Cavium OCTEON III CN71XX Hardware Reference Manual

Contents of this document are subject to change without notice.

CN71XX-HM-0.991E

Cavium Proprietary and Confidential DO NOT COPY

February 2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

PUBLISHED BY
Cavium
2315 North First Street
San Jose, CA 95131
Phone: 408-943-7100
Fax: 408-577-1992
Email: sales@cavium.com
Web: http://www.cavium.com

© 2003-2014 by Cavium, Inc.

All rights reserved. No part of this manual may be reproduced in any form, or transmitted by any means, without the written permission of Cavium, Inc.

Cavium, Inc. makes no warranty about the use of its products, and reserves the right to change this document at any time, without notice. Whereas great care has been taken in the preparation of this manual, Cavium, the publisher, and the authors assume no responsibility for errors or omissions.

OCTEON is a registered trademark of Cavium, Inc.

JEDEC® is a registered trademark of JEDEC Solid State Technology Association.

MIPS, MIPS64, and cnMIPS are registered trademarks of MIPS Technologies. Cavium, Inc is a licensee of cnMIPS. PCI Express and PCIe are registered trademarks of PCI-SIG.

All other trademarks or service marks referred to in this manual are the property of their respective owners.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Table of Contents

Chapter 1 Introduction 35

OCTEON III CN71XX Multicore MIPS64 Processor Family 35
Overview 36
1.1 CN71XX Clocks 39
1.1.1 Standard CN71XX Clocking Scheme 39
1.1.2 Special CN71XX Single-Input Clocking Feature 41
1.2 Principles of Operation 43
1.2.1 CPU Cores 43
1.2.2 Coherent Multicore and I/O L2/DRAM Sharing 43
1.2.3 Core Partitioning 43
1.2.4 Flexible Packet/Control Interfacing 43
1.2.5 In-line Packet-Processing Hardware Acceleration 44
1.2.6 Hardware-Assisted Dynamic Memory Allocation/Deallocation 44
1.2.7 Hardware Work Queuing, Scheduling, Ordering, and Synchronization 44
1.2.8 Essential Quality of Service (QoS) Functions Implemented in Hardware 45
1.2.9 Security Features 45
1.2.10 Coprocessor Accelerators 45
1.2.11 Debug Support 46
1.3 Remaining Chapters 46
1.3.1 CPU Cores 46
1.3.2 Coherent Memory Interconnect and L2 Cache Controller 46
1.3.3 DRAM Controller 46
1.3.4 I/O Interconnect and High-Performance I/O Bridge 46
1.3.5 On-Chip Logic Analyzer Unit 46
1.3.6 Packet Order/Work Unit 47
1.3.7 Free Pool Unit 47
1.3.8 Packet Input Processing/Input Packet Data Unit 47
1.3.9 Packet Output Unit 47
1.3.10 Hyper Finite Automata Unit 47
1.3.11 DMA Packet Interface/Switch Logic Interface (DPI/SLI) 47
1.3.12 Central Interrupt Unit (CIU) 47
1.3.13 Reset Unit (RST) 47
1.3.14 Precision Time Protocol (PTP) Clocks and Timestamping 48
1.3.15 Timer Unit 48
1.3.16 Random Number Generator 48
1.3.17 RAID/De-Dup Unit 48
1.3.18 BCH Unit 48
1.3.19 Key Memory 48
1.3.20 General Serializer/Deserializer (GSER) Interface 48
1.3.21 PCI Express (PCIe) Interface (PEM) 48
1.3.22 SGMII/QSGMII Interface 49
1.3.23 RXAUI Interface 49
1.3.24 RGMII Interface 49
1.3.25 SATA Interface 49
1.3.26 TDM/PCM Interface 49
1.3.27 Boot-Bus Interface 49
1.3.28 eMMC Interface 49
1.3.29 NAND Flash Interface 49

Cavium Confidential For
Bowen Yang
ALU
6/12/2014

1.3.30 USB Interface 49
1.3.31 System Management Interface (SMI) 50
1.3.32 GPIO Interface 50
1.3.33 UART Interface 50
1.3.34 TWSI Interface 50
1.3.35 Serial Peripheral Interface (SPI) 50
1.3.36 Electrical Specifications 50
1.3.37 AC Characteristics 50
1.3.38 Mechanical Specifications 50
1.3.39 Signal Descriptions 50
1.3.40 Ball Assignments 50
1.3.41 Appendix A 51
1.3.42 Appendix B 51
1.4 Configuration and Status Registers (CSRs) 52
1.4.1 CSR Field Types 55
Chapter 2 cnMIPS III Cores 57
Overview 58
2.1 Summary of Features 59
2.1.1 MIPS64 Version 3.0 Implementation 59
2.1.2 Virtualization Module Implementation 59
2.1.3 Cavium-Specific Architectural Additions 59
2.1.4 Full Privileged Architecture (i.e. Coprocessor 0) Support 61
2.1.5 Full EJTAG Version 5.04 Support 62
2.2 Core Non-Privileged State 63
2.3 Cavium-Specific Instruction Summary 65
2.4 Instruction Set Summary 70
2.5 Virtual Addresses and CVMSEG 78
2.6 Physical Addresses 79
2.7 IOBDMA Operations 84
2.8 Core-Memory Reference Ordering 85
2.9 CSR Ordering 91
2.10 Core Write Buffer 92
2.11 Cryptography 94
2.12 Floating-Point Unit 95
2.12.1 Floating Point Modes and Options 95
2.12.2 Denormals 95
2.12.3 Cause Bits 96
2.12.4 Precise Floating Point Traps 97
2.13 Core Error Handling 98
2.14 Virtualization 99
2.15 Coprocessor 0 Privileged Registers 100
2.15.1 Coprocessor 0 Privileged Register in Root Mode 107
2.15.2 Coprocessor 0 Privileged Register in Guest Mode 131
2.15.3 Cavium-Specific Coprocessor 0 Registers in Root Mode 152
2.15.4 Cavium-Specific Coprocessor 0 Registers in Guest Mode 177
2.16 Coprocessor 1 (Floating Point) Control Registers 186
2.17 EJTAG DRSEG Registers 188
2.18 EJTAG TAP Registers 194
2.19 Pipelines 197
2.20 Special MUL Topics 203
2.21 COP2 Latencies 204

Cavium Confidential For Bowen Yang ALU 02/27/2014

2.22 Hardware Debug Features 206
2.22.1 Multicore Debug Support 207
2.22.2 System Debug Characteristics 208
2.22.3 CPU On-Chip Logic Analyzer (OCLA) Samples 209
2.23 Load-Linked/Store-Conditional and Wide Atomic Memory Write Operations 209
2.23.1 Ordinary (MIPS-defined) Atomic Memory Sequences 209
2.23.2 Wide Atomic Memory Write Operations 210
2.23.3 Constraints for a Wide Atomic Memory Write Sequence: 211
2.24 Exceptions 212

Section 1: OCTEON III Cache, System Memory, and Buses 217

Chapter 3 Coherent Memory Interconnect and Level 2 Cache Controller 219

3.1 Coherent Memory Interconnect (CMI) 220
3.1.1 CMI Overview 220
3.1.2 CMI Buses 220
3.1.3 CMI Description 221
3.1.4 CMI Memory Coherence Support 221
3.1.5 Coherent Memory Interconnect Transaction Examples 221
3.1.6 CMI Transactions 223
3.2 Level-2 Cache Controller (L2C) 231
3.2.1 L2 Cache and Data Store 231
3.2.2 L2C Memory Coherence 232
3.2.3 L2 Cache Indexing (Set Selection) 233
3.2.4 L2 Cache Replacement and Way Partitioning 234
3.2.5 Memory Input Queue QOS 234
3.2.6 COMMIT and FILL Bus Arbitration 235
3.2.7 Core COP0 Register Access 235
3.2.8 Main Memory L2/DRAM Physical Address Space 236
3.2.9 L2C+LMC Internal L2/DRAM Address Space 236
3.2.10 L2 Cache-Block Locking 236
3.2.11 L2 Cache Block Flush and Unlocking 237
3.2.12 L2C ECC Codes 237
3.3 Enumerations 238
3.4 L2C Registers 239

Chapter 4 DRAM Controller (LMC) 259

Overview 260
4.1 Internal and DRAM Part Addressing 261
4.2 DRAM Read Example 263
4.3 DRAM Write Example 267
4.4 DRAM Programming 270
4.5 DRAM Refreshes, DRAM ZQ Calibration, Internal ZQ Calibration, and Internal DLL Fine-Tuning 270
4.6 DRAM Scheduler Performance 271
4.7 DRAM Chip Select, CKE, and ODT Signals 271
4.8 LMC Initialization Sequence 272
4.8.1 DDR PLL Initialization 273
4.8.2 LMC CK Initialization 274
4.8.3 LMC DRESET Initialization 274
4.8.4 LMC RESET Initialization 275
4.8.5 Early LMC Initialization 276
4.8.6 LMC Offset Training 277
4.8.7 LMC Internal Vref Training 277
4.8.8 LMC Deskew Training 277

Cavium Confidential For Bowen Yang ALU 02/27/2014

4.8.9 LMC Write Leveling 277
4.8.10 LMC Read Leveling 279
4.8.11 Final LMC Initialization 281
4.9 DRAM ECC Codes 282
4.10 Dynamic Precharge Power-Down 283
4.11 Non-Homogeneous Ranks 283
4.12 Address Mirroring 283
4.13 CK Speed Programming 284
4.14 Example System Configuration 285
4.15 Enumerations 286
4.16 LMC Registers 286
Chapter 5 I/O Interconnect (IOI), I/O Bridge (IOB) and Fetch and Add Unit (FAU) 341
5.1 I/O Interconnect 342
5.1.1 I/O Interconnect Overview 342
5.1.2 IOI Flow Examples 343
5.2 IOB Architecture Overview 345
5.3 Don't-Write-Back Engine 346
5.4 Fetch and Add Unit (FAU) 346
5.5 Fetch-and-Add Operations 348
5.5.1 Load Operations 348
5.5.2 IOBDMA Operations 350
5.5.3 Store Operations 352
5.6 I/O Bridge Registers 353
Chapter 6 On-Chip Logic Analyzer Unit (OCLA) 365
Overview 366
6.1 Features 366
6.2 Architecture 366
6.2.1 Debug Bus Transmitters (DTX) 367
6.2.2 Aligner 368
6.2.3 Matchers 368
6.2.4 FSMs 369
6.2.5 State Outputs 370
6.2.6 Capturing FIFOs 370
6.2.7 Capture to DDR / L2 Cache 372
6.2.8 Capture Transitions 372
6.3 Enumerations 374
6.4 OCLA Registers 375
6.4.1 General Registers 376
6.4.2 DDR/L2 Overflow Stack 380
6.4.3 Per-Matcher Registers 382
6.4.4 Per-FSM Registers 383
6.4.5 Per-Capture Half Registers 384
6.4.6 Data Registers 385
6.5 DTX Registers 386
Section 2: OCTEON III Coprocessors 389
Chapter 7 Packet Order / Work Unit (POW) 391
Overview 392
7.1 POW Work Flow, Operations, and Ordering 393
7.2 Example Software Architecture Using POW Ordering and Synchronization 400

Cavium Confidential For Bowen Yang ALU 02/27/2014

7.2.1 Defragmentation 403
7.2.2 IPsec Decryption 403
7.2.3 Lookup 403
7.2.4 Process 404
7.2.5 IPsec Encryption 404
7.2.6 Output Queue 404
7.3 POW Internal Architecture 404
7.4 Work-Queue Entry Format 407
7.5 Core and Fetch-and-Add Pending Switch Bits 408
7.6 POW Interrupts 409
7.6.1 Per-Packet/Work Receipt Interrupts 409
7.6.2 Input-Queue Threshold Interrupts 412
7.7 POW QOS Features 413
7.7.1 Thresholds 413
7.7.2 Scheduling 413
7.8 POW Debug Visibility 415
7.9 POW Performance Considerations 417
7.10 Forward Progress Constraints 418
7.11 POW Operations 419
7.11.1 Load Operations 419
7.11.2 IOBDMA Operations 426
7.11.3 Store Operations 426
7.12 POW ECC Codes 427
7.13 POW Registers 428
Chapter 8 Free Pool Unit (FPA) 439
Overview 440
8.1 Free Pool Unit Operations 442
8.1.1 Load Operations 442
8.1.2 IOBDMA Operations 442
8.1.3 Store Operations 443
8.2 FPA Registers 444
Chapter 9 Packet Input Processing/Input Packet Data Unit (PIP/IPD) 455
Overview 456
9.1 Input Ports 456
9.2 Input Packet Formats and Pre-IP Parsing 457
9.2.1 Packet Instruction Header 458
9.2.2 DPI Instruction-to-Packet Conversion 461
9.2.3 Parse Mode and Skip Length Selection 462
9.2.4 PIP/IPD L2 Parsing and Is_IP Determination 463
9.2.5 Pre-IP Parsing Summary 464
9.2.6 Packet Input CRC 466
9.2.7 Packet Length Checks 466
9.2.8 Legal SKIP Values 469
9.3 Packet Buffering 470
9.4 Packet Scheduling 475
9.4.1 RAWFULL and RAWSCHED Packets 475
9.4.2 Bit-Select Extractor 475
9.4.3 QOS 476
9.4.4 GRP 477
9.4.5 TT 477
9.4.6 Tag 477

Cavium Confidential For Bowen Yang ALU 02/27/2014

9.5 Work-Queue Entry 478
9.6 Input Packet Data Unit (IPD) Quality of Service 500
9.7 PIP/IPD Per-QOS Admission Control 502
9.8 Per-Port and QOS Threshold Interrupts 505
9.9 PIP Registers 508
9.9.1 PIP Statistics Counters 528
9.9.2 PIP Inbound Statistics Registers 532
9.10 IPD Registers 533
Chapter 10 Packet-Output Processing Unit (PKO) 547
Overview 548
10.1 Output Ports 550
10.2 Output Packet Format and TCP/UDP Checksum Insertion 551
10.3 PKO Output Queue 552
10.4 PKO Commands 553
10.5 PKO Queue-Arbitration Algorithm 560
10.6 PKO-Internal Port-to-Engine Arbitration Algorithm 563
10.7 PKO Don't-Write-Back (DWB) Calculation 565
10.8 PKO-External Port-Level Backpressure 566
10.9 PKO-Internal Port Rate-Limiting 567
10.10 PKO Performance 569
10.11 PKO Operations 570
10.11.1 Store Operations 570
10.12 PKO Registers 571
Chapter 11 Hyper Finite Automata (HFA) 591
Overview 592
11.1 Introduction 592
11.2 Hardware Features 593
11.3 Theory of Operation 594
11.3.1 HFA Memory Debug Access 595
11.3.2 HFA Graph Types 595
11.4 External Data Structures 595
11.4.1 HFA Instruction Queue 595
11.4.2 HFA Instruction Formats 598
11.4.3 Detailed IWORD Field Definitions 601
11.4.4 HFA Result Format 609
11.4.5 HFA Graph Formats 611
11.4.6 Loading Graphs 612
11.4.7 Freeing Cacheable HFA Graphs 616
11.5 HFA Registers 616
Chapter 12 DMA Packet Interface Unit (DPI) and Switch Logic Interface Unit (SLI) 627
Overview 628
12.1 DPI Instruction Input From an External Host 629
12.1.1 DPI Instruction Formats 630
12.1.2 DPI Input Packets 633
12.1.3 DPTR Formats 635
12.1.4 DPI Instruction Input Ring Backpressure 638
12.1.5 DPI Instruction Input Initialization 639
12.2 DPI Packet Output From CN71XX 640
12.2.1 DPI Packet-Output Info-Pointer Mode 644
12.2.2 DPI Packet-Output Buffer-Pointer-Only Mode 645

Cavium Confidential For Bowen Yang ALU 02/27/2014

12.2.3 DPI Packet Output Initialization 646
12.3 DPI DMA Queues and Engines 648
12.3.1 DPI DMA Instruction-Header Format 648
12.3.2 DPI DMA Local-Pointer Format 651
12.3.3 DPI Components and Processing 653
12.3.4 DPI DMA Instruction Queues 655
12.3.5 DMA Queue-to-Engine Mapping 656
12.3.6 DPI DMA Instruction Ordering and Completion 656
12.3.7 DPI DMA Engine Don't-Write-Back Calculation 658
12.3.8 DPI Host Output Queuing Via the DMA Engine 658
12.3.9 DPI DMA Engine Performance 659
12.3.10 DMA Configuration and Initialization 661
12.4 SLI MAC BAR1/BAR2 Target Mode L2C/DRAM Load/Store Operations 663
12.5 SLI Core-initiated Request Logic 665
12.6 CN71XX as a MAC Interrupt Destination 667
12.6.1 CN71XX Receiving Emulated INTA/INTB/INTC/INTD 667
12.6.2 CN71XX Receiving Message-Signaled Interrupts 668
12.7 Endian Swapping 670
12.7.1 Pass-Through Mode 670
12.7.2 64-bit Byte-Swap Mode 671
12.7.3 32-bit Byte-Swap Mode 671
12.7.4 32-bit Exchange Mode 672
12.8 SLI Bus Operations 673
12.8.1 Load/Store Operations 673
12.8.2 IOBDMA Operations 673
12.8.3 RSL Access Space (DID = 0x18) 674
12.8.4 PCIe Configuration Space (DID = 0x19) 674
12.8.5 PCIe I/O Space (DID = 0x1A) 674
12.8.6 MAC Memory Space (DID = 0x1B, 0x1C, 0x1D, 0x1E) 675
12.8.7 SLI CSRs (DID = 0x1F) 676
12.9 Configuration Summary 676
12.10 DPI Registers 678
12.11 SLI Registers 693
Chapter 13 Central Interrupt Unit (CIU) 731
Overview 732
13.1 Central Interrupt Collection and Distribution 732
13.2 Per-Core Mailbox Registers 739
13.3 Per-Core Watchdog Timers 739
13.4 Ten General Timers 741
13.5 Core Availability and Reset 741
13.6 Core Debug-Mode Observability 741
13.7 Core Debug-Interrupt Generation 742
13.8 Core Non-Maskable Interrupt Generation 742
13.9 Chip Soft-Reset Initiation 742
13.10 CIU Registers 743
Chapter 14 Reset Unit (RST) 785
Overview 786
14.1 Core Availability and Reset 786
14.1.1 Chip Soft-Reset Initiation 786
14.2 Reset/Fuse Registers 787
Chapter 15 Precision Time Protocol (PTP) Clocks and Timestamping 799

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview 800
15.1 PTP Clock 800
15.2 PTP Clock Outputs 802
15.3 CN71XX Outbound PTP Timestamp Architecture 803
15.4 GMX Inbound PTP Timestamp Support 803
15.5 PIP/IPD Inbound PTP Timestamp Support 803
15.6 PKO and GMX Outbound Timestamp Support 803
15.7 PTP Timestamp 804
15.8 PTP Event Counter 804
15.9 Controlling External Clocks 805
15.10 Timestamp Registers 806
Chapter 16 Timer (TIM) 811
Overview 812
16.1 Timer Features 812
16.2 TIM Support 813
16.3 Software Responsibilities 815
16.4 Timer Registers 816
Chapter 17 Random-Number Generator (RNG), Random-Number Memory (RNM) 821
Overview 822
17.1 RNG/RNM Operations 824
17.1.1 RNG/RNM Load Operation 824
17.1.2 IOBDMA Operations 825
17.2 RNM Registers 826
Chapter 18 RAID Unit (RAD) 829
Overview 830
18.1 Pipe Calculations 831
18.2 Instruction Queue 833
18.3 Instruction Format 834
18.3.1 CWORD Format 835
18.3.2 OWORD Format 836
18.3.3 IWORD Format 837
18.3.4 RESP Format 838
18.4 Operations 839
18.5 Performance 839
18.6 Ordering 840
18.7 RAD Registers 840
Chapter 19 BCH Unit 847
Overview 848
19.1 Instruction Queue 848
19.2 Instruction Format 850
19.2.1 CWORD Format 850
19.2.2 OWORD Format 851
19.2.3 IWORD Format 851
19.2.4 RESP Format 853
19.3 Operations 853
19.4 Ordering 854
19.5 BCH Registers 854
Chapter 20 Key Memory Unit (KEY) 857

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview 858
20.1 Key Registers 858
Section 3: CN71XX I/O Interfaces 861
Chapter 21 General Serializer/Deserializer (GSER) Interface 863
Overview 864
21.1 Initializing the DLMs 864
21.1.1 DLM0 Initialization 864
21.1.2 DLM1/2 Initialization: 865
21.2 GSER Registers 867
Chapter 22 PCI Express Interface (PEM) 893
Overview 894
22.1 Summary of PEM Features 894
22.2 PEM as a PCIe Receiver 896
22.2.1 Receiving PCIe TLP I/O Space Requests 896
22.2.2 Receiving PCIe TLP Memory Space Requests 896
22.2.3 Receiving PCIe TLP Configuration-Space Requests 903
22.2.4 Receiving PCIe TLP Messages 903
22.2.5 Receiving PCIe TLP Completions 904
22.2.6 Receiving PCIe DLLP Packets 905
22.3 PEM as a PCIe Transmitter 905
22.3.1 Transmitting PCIe TLP I/O Space Requests 905
22.3.2 Transmitting PCIe TLP Memory Space Requests 905
22.3.3 Transmitting PCIe TLP Configuration Space Requests 905
22.3.4 Transmitting PCIe TLP Messages 906
22.3.5 Transmitting PCIe TLP Completions 906
22.3.6 Transmitting PCIe DLLP Packets 907
22.4 PEM0 Configuration-EEPROM Interface 907
22.4.1 EEPROM Programming 908
22.5 PCIe Reset Sequences 911
22.5.1 Fundamental PCIe Reset Sequences 911
22.5.2 Soft Reset for PCIe 916
22.5.3 PCIe Link-Down or Hot Reset 916
22.6 PCIe Configuration Summary 918
22.7 PCIe Configuration Registers 920
22.7.1 EP Mode PCIe Configuration Registers 920
22.7.2 RC Mode PCIe Configuration Registers 958
22.8 PEM Registers 996
Chapter 23 SGMII / 1000BASE-X / QSGMII Interfaces 1013
Overview 1014
23.1 Introduction 1014
23.2 GMX Packet-Interface Buffering 1015
23.3 PCS Autonegotiation, Mode, and GMX Configuration 1015
23.3.1 Autonegotiation 1016
23.3.2 PCS Mode Determination 1017
23.3.3 GMX Configuration Based on PCS Mode 1018
23.4 SGMII/1000BASE-X/QSGMII Features 1018
23.4.1 QSGMII Mode 1018
23.4.2 SerDes Lane Polarity 1018
23.4.3 Flow Control 1019
23.4.4 Receive Packet Dropping 1023
23.4.5 Receive Packet Processing and Format 1025

Cavium Confidential For Bowen Yang ALU 02/27/2014

23.4.6 Transmit Packet Processing 1029
23.4.7 Unidirectional Transport 1034
23.5 Wake-on-LAN (WOL) 1034
23.5.1 Packet Detection 1034
23.5.2 Operation 1036
23.6 Errors/Exceptions 1036
23.6.1 Receive Error/Exception Checks 1037
23.6.2 Transmit Error/Exception Checks 1039
23.6.3 Transmit Error Propagation 1039
23.6.4 PCS Error/Exception Checks 1040
23.7 Reset, Autonegotiation, Initialization, and Link Status Changes 1041
23.7.1 GMX/PCS Reset and Enables 1041
23.7.2 Autonegotiation Execution 1042
23.7.3 Link Bring-Up 1043
23.7.4 Link Bring-Down 1044
23.7.5 Link Status and Changes 1044
23.8 Statistics 1045
23.9 Loopback 1046
23.10 Enumerations 1047
23.11 GMX Registers 1047
23.12 PCS Registers 1083
Chapter 24 Reduced 10 Gigabit Ethernet Attachment Unit Interface (RXAUI) 1093
Overview 1094
24.1 Introduction 1094
24.2 GMX Packet Interface Buffering 1094
24.3 RXAUI Features 1095
24.3.1 RXAUI Lane Polarity and Swapping 1095
24.3.2 Flow Control 1095
24.3.3 Receive Packet Dropping 1107
24.3.4 Receive Packet Processing and Format 1109
24.3.5 Transmit Packet Processing 1113
24.3.6 Unidirectional Transport 1116
24.4 Wake-on-LAN (WOL) 1117
24.4.1 Packet Detection 1117
24.4.2 Operation 1118
24.5 Errors/Exceptions 1119
24.5.1 Receive Errors/Exception Checks 1119
24.5.2 Transmit Errors/Exception Checks 1122
24.5.3 Transmit Error Propagation 1123
24.6 Reset, Initialization, and Link Status Changes 1123
24.6.1 GMX/PCSX Reset and Enables 1123
24.6.2 Link Bring-Up 1124
24.6.3 Link Bring-Down 1125
24.6.4 Link Status and Changes 1125
24.7 Statistics 1126
24.8 Loopback 1127
24.9 Enumerations 1128
24.10 GMX Registers 1129
24.11 PCSX Registers 1167
Chapter 25 Reduced Gigabit Media-Independent Interface (RGMII) 1173
Overview 1174

Cavium Confidential For Bowen Yang ALU 02/27/2014

25.1 Introduction 1174
25.2 GMX Packet-Interface Buffering 1174
25.3 AGL Features 1175
25.3.1 Flow Control 1175
25.3.2 MAC Receive Preamble Processing 1178
25.3.3 Received Packet FCS Processing 1178
25.3.4 MAC Receive Packet Dropping 1179
25.3.5 Receive-Packet Inspection 1181
25.3.6 Receive Link Status 1182
25.3.7 MAC Packet Transmission 1182
25.3.8 MAC Transmit-Packet Options 1183
25.3.9 MAC Collision Processing 1183
25.3.10 Bursts 1183
25.3.11 Connectivity 1183
25.4 Wake-on-LAN (WOL) 1184
25.4.1 Packet Detection 1184
25.4.2 Operation 1185
25.5 AGL Errors/Exceptions 1185
25.5.1 AGL MAC Receive Error/Exception Checks 1186
25.5.2 AGL MAC Transmit Error/Exception Checks 1188
25.6 AGL Interrupts 1188
25.7 Configuration, Reset, and Link Status Changes 1189
25.7.1 AGL Bring-up Sequence 1189
25.7.2 AGL Interface link status changes 1190
25.8 Statistics 1191
25.9 AGL Registers 1192
Chapter 26 Serial AT Attachment Interface (SATA) 1223
Overview 1224
26.1 Architecture 1224
26.1.1 UCTL 1225
26.1.2 UAHC 1225
26.2 Initialization 1226
26.3 Clocks and Reset 1226
26.3.1 Cold Reset 1227
26.4 Endian-Swapping 1228
26.5 Interrupt Handler 1228
26.5.1 AHCI Interrupts 1229
26.5.2 UCTL Interrupts 1229
26.6 Host-Mode Programming Model 1229
26.7 Enumerations 1230
26.8 Registers 1230
26.8.1 SATA UCTL Registers 1231
26.8.2 SATA UAHC Registers 1237
Chapter 27 TDM/PCM Interface 1253
Overview 1254
27.1 Signal Usage 1255
27.2 Clocking 1256
27.2.1 BCLK Generation 1256
27.2.2 FSYNC Generation 1257
27.2.3 BCLK Reception 1257
27.2.4 FSYNC Reception 1257

Cavium Confidential For Bowen Yang ALU 02/27/2014

27.2.5 Examples BCLK/FSYNC Waveforms ... 1258
27.3 TDM Engines ... 1259
27.3.1 TDM Engine Configuration ... 1260
27.3.2 DMA Engines ... 1261
27.4 Initialization Sequence ... 1265
27.5 TDM/PCM Registers ... 1266
Chapter 28 Boot Bus ... 1273
Overview ... 1274
28.1 Addresses ... 1275
28.2 Address Matching and Regions ... 1276
28.3 Reset Configuration and Booting ... 1277
28.4 Region Timing ... 1278
28.4.1 Static-Timed Read Sequences ... 1280
28.4.2 Static-Timed Write Sequences ... 1281
28.4.3 Dynamic-Timed Sequences ... 1282
28.5 Request Queuing ... 1283
28.6 Connections ... 1283
28.7 Operations ... 1284
28.7.1 Load Operations ... 1284
28.7.2 IOBDMA Operations ... 1285
28.7.3 Store Operations ... 1286
28.8 Boot-Bus Registers ... 1287
Chapter 29 eMMC Interface Unit ... 1297
Overview ... 1298
29.1 Access via CSR ... 1300
29.1.1 Command/Response Type Override ... 1300
29.1.2 SD Card Support ... 1301
29.2 eMMC Access Via DMA ... 1302
29.2.1 DMA Write Operations ... 1302
29.2.2 DMA Read Operations ... 1303
29.3 eMMC DMA Interrupts ... 1304
29.4 Booting from an eMMC/SD Device ... 1304
29.5 Software Sequences ... 1306
29.5.1 Power Cycle eMMC Card ... 1307
29.5.2 Reset Card ... 1307
29.5.3 Initialize Card – eMMC Only ... 1307
29.5.4 Initialize Card – eMMC and SD ... 1308
29.5.5 Switch Operating Mode ... 1310
29.5.6 Initiate a DMA Transfer ... 1310
29.5.7 Resume a DMA transfer ... 1310
29.5.8 Terminate a DMA transfer ... 1311
29.6 eMMC Registers ... 1311
Chapter 30 NAND Flash Interface Unit ... 1323
Overview ... 1324
30.1 NAND Flash Access Via the NDF_CMD Queue ... 1326
30.2 Detailed NDF_CMD Command Descriptions ... 1329
30.2.1 NOP Command ... 1330
30.2.2 SET_TM_PAR Command ... 1330
30.2.3 WAIT Command ... 1330
30.2.4 BUS_ACQ Command ... 1331
30.2.5 BUS_REL Command ... 1332

Cavium Confidential For Bowen Yang ALU 02/27/2014

30.2.6 CHIP_EN Command 1333
30.2.7 CHIP_DIS Command 1334
30.2.8 CLE_CMD Command 1335
30.2.9 ALE_CMD Command 1336
30.2.10 RD_CMD Command 1338
30.2.11 RD_EDO_CMD Command 1340
30.2.12 WR_CMD Command 1342
30.2.13 WAIT_STATUS and WAIT_STATUS_ALE Commands 1343
30.3 NDF_CMD Command Examples 1346
30.3.1 NAND Flash Cache-Block Read Sequence 1347
30.3.2 Block-Erase Sequence 1349
30.3.3 Page-Program Sequence 1350
30.4 NAND Flash Registers 1351
Chapter 31 Universal Serial Bus Unit (USB) 1357
Overview 1358
31.1 Architecture 1359
31.1.1 UCTL 1359
31.1.2 UAHC 1360
31.1.3 UPHY 1360
31.2 Initialization 1360
31.2.1 Cold Reset 1362
31.2.2 Hot Reset 1364
31.2.3 Soft Reset 1364
31.2.4 Dynamically Powering Down PHYs 1365
31.2.5 Statically Powering Down PHYs 1365
31.2.6 Suspend/Sleep/LPM 1365
31.3 Endian-Swapping 1366
31.4 Interrupt Handler 1366
31.4.1 UAHC Interrupts 1366
31.5 Board Implementation 1367
31.5.1 Power Supplies and ESD Protection 1367
31.5.2 Reference Clocks 1367
31.5.3 Reference Resistor 1368
31.6 Enumerations 1369
31.7 Registers 1373
31.7.1 UPHY Registers 1373
31.7.2 UCTL Registers 1375
31.7.3 UAHC Registers 1393
Chapter 32 System Management Interface (SMI) 1473
Overview 1474
32.1 SMI/MDIO Interface 1474
32.2 SMI Registers 1476
Chapter 33 GPIO Unit 1481
Overview 1482
33.1 GPIO Operations 1483
33.1.1 Reading the GPIO Bus 1484
33.1.2 Writing the GPIO Bus 1484
33.1.3 GPIO Interrupts 1484
33.1.4 GPIO Clock Generator 1484
33.2 Glitch Filters 1485
33.3 GPIO Registers 1486

Cavium Confidential For Bowen Yang ALU 02/27/2014

Chapter 34 UART Interface 1495
Overview 1496
34.1 Pin Usage 1496
34.2 UART (RS232) Serial Protocol 1497
34.3 Interrupts 1498
34.4 AutoFlow Control 1498
34.4.1 UART AutoRTS 1498
34.4.2 UART AutoCTS 1499
34.4.3 UART Programmable THRE Interrupt 1500
34.5 UART Registers 1501
Chapter 35 TWSI Interface 1517
Overview 1518
35.1 CN71XX High-Level Controller as a Master 1519
35.2 CN71XX High-Level Controller as a Slave 1523
35.3 Direct TWSI Core Usage 1525
35.3.1 Master Transmit Mode 1526
35.3.2 Master Receive Mode 1528
35.3.3 Slave Transmit Mode 1530
35.3.4 Slave Receive Mode 1531
35.3.5 TWSI Core Flow Diagrams 1532
35.4 TWSI Control Registers 1534
35.4.1 TWSI Slave Address Register 1534
35.4.2 TWSI Slave Extended-Address Register 1535
35.4.3 TWSI Data Register 1535
35.4.4 TWSI Control Register 1535
35.4.5 TWSI Status Register 1537
35.4.6 TWSI Master Clock Register 1538
35.4.7 TWSI Clock Control Register 1538
35.4.8 TWSI Software Reset Register 1539
35.5 TWSI Registers 1539
Chapter 36 SPI/MPI Unit 1545
Overview 1546
36.1 Pin Usage 1547
36.2 SPI/MPI Configuration 1547
36.2.1 Clock Generation 1547
36.2.2 Chip Select Signals 1548
36.2.3 Polling/Interrupt-Based Reception 1549
36.2.4 Other Fields in MPI_CFG 1549
36.3 SPI/MPI Usage 1549
36.3.1 MPI_DAT(0..8) and MPI_WIDE_DAT Registers 1550
36.3.2 Using the MPI_TX Register 1550
36.3.3 Using the MPI_STS Register 1550
36.4 SPI as a Boot Device 1551
36.5 Examples 1552
36.5.1 Example 1: Reading a Single Byte From Device Address 0x04 1552
36.5.2 Example 2: Writing a Single Byte to Register 0x04 1552
36.5.3 Example 3: Writing Ten Bytes to Registers 0x09–0x00 1553
36.5.4 Example 4: Reading 17 Bytes From Registers 0x11–0x00 1554
36.6 SPI/MPI Registers 1555
Section 4: OCTEON III Electrical/Mechanical/Physical 1559
Chapter 37 Electrical Specifications 1561

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview 1562
37.1 Absolute-Maximum Ratings 1562
37.1.1 Absolute Maximum Storage Temperatures 1562
37.2 Recommended Operating Conditions 1563
37.2.1 Supply Voltages for the Chip-Core Voltage and External Interfaces 1563
37.2.2 Supply Voltage for the On-Chip PLLs 1564
37.3 Power Sequencing 1564
37.3.1 Power Up in Default Mode 1565
37.3.2 Power Up in SerDes Common-Clock Mode 1566
37.3.3 Power Down 1567
37.4 Power Consumption 1567
37.5 DC Electrical Characteristics 1569
37.5.1 DDR3 Bidirectional I/O for the DDR3 Memory Interface 1569
37.5.2 3.3V CMOS Bidirectional and Point-to-Point I/O for the Miscellaneous Interface 1570
37.5.3 1.8V/2.5V CMOS Bidirectional I/O for RGMII Interface 1570
Chapter 38 AC Characteristics 1571
38.1 Input Clocks 1572
38.1.1 Inputs for Core/Coprocessor/DDR Reference-Clock 1572
38.2 DDR3 SDRAM Interface Timing Specifications 1573
38.2.1 DDR3 AC Test Load 1573
38.2.2 DDR3 Output-Clock Timing Specifications 1573
38.2.3 DDR3 Output Timing Specifications 1574
38.2.4 DDR3 Input Timing Specifications 1576
38.3 SerDes Timing Specifications 1577
38.3.1 DLM Common Reference Clock 1577
38.3.2 PCIe Interface (Gen1) Timing Specifications 1578
38.3.3 PCIe Interface (Gen2) Timing Specifications 1579
38.3.4 SGMII Interface Timing Specifications 1580
38.3.5 RXAUI Interface Timing Specifications 1581
38.3.6 SATA Interface Timing Specifications 1582
38.4 RGMII Interface 1583
38.4.1 RGMII Clock Input Timing 1583
38.4.2 RGMII Signal Timing 1583
38.5 EEPROM Interface 1585
38.5.1 EEPROM Read Cycle 1585
38.6 Boot Bus Interface 1587
38.7 EMMC Interface 1587
38.8 JTAG Interface 1588
38.9 TDM/PCM Interface 1588
38.10 MPI/SPI Interface 1589
38.11 TWSI Interface 1590
38.12 SMI/MDIO Interface 1590
38.13 USB Interface 1591
38.13.1 USB Clock Input Timing 1591
38.13.2 USB AC Specifications 1591
Chapter 39 Mechanical Specifications 1593
Overview 1594
39.1 Landed Ball-Grid-Array Package Diagram 1594
39.2 Package Thermal Specifications 1596
Chapter 40 CN71XX Signal Descriptions 1597
Overview 1598

Cavium Confidential For Bowen Yang ALU 02/27/2014

40.1 DRAM Interface Signals 1599
40.2 SerDes Signals 1600
40.3 RGMII Interface Signals 1601
40.4 USB Interface Signals 1602
40.5 SMI/MDIO Interface Signals 1602
40.6 General Purpose I/O (GPIO) Interface Signals 1603
40.7 TDM/PCM Interface Signals 1603
40.8 Two-Wire Serial Interface (TWSI) Signals 1604
40.9 Clock Signals 1604
40.10 Boot-Bus/eMMC Signals 1605
40.11 SPI Interface Signals 1605
40.12 UART Interface Signals 1606
40.13 EJTAG/JTAG Signals 1606
40.14 Miscellaneous Signals 1607
40.15 Power/Ground/Reserved/No Connect Signals 1607

Chapter 41 Ball Assignments 1609

Overview 1610
41.1 CN71XX Landed Ball-Grid Array 1610
41.2 CN71XX Signal Mapping 1611
41.3 Balls Sorted in Alphabetical Order by Signal Name 1616
41.4 Balls Sorted in Numerical Order and Associated Signal Names 1621

Appendix A Cavium-Specific Core Instructions 1627

A.1 CPU Arithmetic (and Move) Instructions 1633
A.1.1 CPU Insert/Extract Instructions 1635
A.1.2 Large Multiply Instructions 1639
A.1.3 Population Count Instructions 1652
A.1.4 Set Equals/Set Not-Equals Instructions 1654
A.1.5 Q15 Multiply and Accumulate Instructions 1658
A.2 CPU Branch and Jump Instructions 1662
A.2.1 Bit-test Branches Instructions 1662
A.3 CPU Load, Store, Memory, and Control Instructions 1666
A.3.1 Prefetch Instructions 1666
A.3.2 Store Atomic Add Instructions 1671
A.3.3 SYNC Instructions 1675
A.3.4 Unaligned Load/Store Instructions 1682
A.3.5 Integer Indexed Load Instructions 1694
A.3.6 Memory Atomic Fetch and Add Instructions 1706
A.3.7 Memory Atomic Swap Instructions 1710
A.3.8 Memory Atomic Increment Instructions 1714
A.3.9 Memory Atomic Decrement Instructions 1718
A.3.10 Memory Atomic Set Instructions 1722
A.3.11 Memory Atomic Clear Instructions 1726
A.3.12 Zero Cache Block Instructions 1730
A.4 CPU Move and Privileged Instructions 1732
A.5 Coprocessor Move Instructions 1741
A.5.1 CRC Instructions 1741
A.5.2 Security Accelerator Instructions 1760

Appendix B Ordering Information 1893

Cavium Confidential For Bowen Yang ALU 02/27/2014

List of Figures

Figure 1–1 CN71XX Block Diagram 37
Figure 1–2 CN71XX Signals 38
Figure 1–3 CN71XX Detailed Clocking Diagram 40
Figure 1–4 CN71XX Single-Input Clocking Example 41
Figure 2–1 cnMIPS™ III Core Block Diagram 58
Figure 2–2 Forty-Nine-bit Physical Address Format 79
Figure 2–3 Sixty-Four-Bit IOBDMA Store Data Format 84
Figure 2–4 Floating-Point Data Types 95
Figure S1-1 CN71XX Cache, Memory, and Buses 217
Figure 3–1 Coherent Memory Interconnect Diagram 220
Figure 3–2 Fill Transaction 221
Figure 3–3 Store Transaction 222
Figure 3–4 Store to IOB Transaction 223
Figure 3–5 Load to IOB Transaction 223
Figure 3–6 L2C Block Diagram 232
Figure 4–1 SDRAM Physical Address 261
Figure 4–2 Cache-block Read Transaction Example 263
Figure 4–3 Cache-block Write Transaction Example 267
Figure 4–4 Memory System Configuration 285
Figure 5–1 IOI Block Diagram 342
Figure 5–2 IOI Flows for a Packet-Data Input 343
Figure 5–3 IOI Flows for Packet Data Output 344
Figure 5–4 IOI Flow: Memory Allocate 344
Figure 5–5 IOI Flow: Memory Free 345
Figure 5–6 I/O Bridge Block Diagram 345
Figure 5–7 FAU Block Diagram 346
Figure 6–1 OCLA Complex Architecture 366
Figure 6–2 DTX Architecture 367
Figure 6–3 FSM PLA 369
Figure 6–4 Overall State Transition Example 373
Figure S2-1 CN71XX Coprocessors 389
Figure 7–1 States of Work Flowing Through the POW 394
Figure 7–2 The POW States Visible to a Core 396
Figure 7–3 Architecture for the Firewall/VPN 401
Figure 7–4 Software Execution Sequence 402
Figure 7–5 Internal POW Architecture Components 405
Figure 7–6 Format Requirements for the Work-Queue Entry 407
Figure 8–1 FPA Free Memory Pointers 440
Figure 8–2 FPA Free Memory Pointers 441
Figure 9–1 Packet Input Format Modes 457
Figure 9–2 Packet Instruction Header Format (PKT_INST_HDR) 458
Figure 9–3 DPI Instruction to Packet Transformation 461
Figure 9–4 Supported L2 HDR Types in Skip-to-L2 Mode 463
Figure 9–5 Length-Check Algorithm 468
Figure 9–6 PIP/IPD Hardware Allocating Multiple Buffers 471
Figure 9–7 Format Of 64-bit Buffer Pointer 472
Figure 9–8 Packet Alignment for IPv4 and IPv6 Packets 473
Figure 9–9 Bit-Select Extractor 476
Figure 9–10 PIP/IPD Hardware Work-Queue Entry 480
Figure 9–11 Work-Queue Entry Format; Word 2 Cases 481
Figure 9–12 Work-Queue Entry Format; Word 4–15 Cases 482

Cavium Confidential For Yiwen Yang ALU 02/27/2014

Figure 9–13 PIP/IPD Packet-Drop Probability ... 504
Figure 9–14 Port/QOS Combination Values ... 505
Figure 10–1 PKO Conceptual Architecture ... 548
Figure 10–2 PKO Internal Architecture ... 550
Figure 10–3 Format Requirements for Packets ... 551
Figure 10–4 Structure of Output Queue ... 552
Figure 10–5 Usage of the Linked Packet Construction Mode ... 554
Figure 10–6 Packet Segments Connected with Gather Mode ... 554
Figure 10–7 Format of Commands in the PKO Output Queues ... 556
Figure 11–1 HFA IQ Format ... 596
Figure 11–2 Next-Chunk Pointer Format ... 596
Figure 11–3 HFA Instruction Format: Graph CacheLoad (ITYPE = 0x1) ... 599
Figure 11–4 HFA Instruction Format: Graph Walk (ITYPE = 0x4) ... 600
Figure 11–5 HFA Instruction Format: Graph Free (ITYPE = 0x3) ... 601
Figure 11–6 IWORD0 ... 602
Figure 11–7 HTE Gather-List-Entry Format ... 603
Figure 11–8 IWORD1 ... 605
Figure 11–9 IWORD2 ... 606
Figure 11–10 IWORD3 ... 608
Figure 11–11 HFA Result Format ... 609
Figure 11–12 Node-Cache Structure ... 613
Figure 11–13 Graph-Cache Data Structures ... 614
Figure 12–1 DPI/SLI Block Diagram ... 628
Figure 12–2 Instruction Formats ... 630
Figure 12–3 Input Packet Format ... 633
Figure 12–4 Structure of a DPI Gather Component Used in Indirect Gather Mode ... 634
Figure 12–5 PKO Output Queues to DPI Rings Example Mapping ... 641
Figure 12–6 Buffer/Info Pointer Pair ... 643
Figure 12–7 Format Written to the Info Pointer ... 645
Figure 12–8 DPI DMA Instruction Format ... 648
Figure 12–9 DMA Instruction-Header Format ... 649
Figure 12–10 DPI DMA Local-Pointer Format ... 651
Figure 12–11 DPI Component Format Example When HDR[NLST] = 9 ... 653
Figure 12–12 MAC Length-Field Format ... 654
Figure 12–13 Example: Queue Linked-List Memory Chunks ... 655
Figure 12–14 CN71XX Handling of INTA/INTB/INTC/INTD Inputs ... 668
Figure 12–15 SLI Bit Selection for Each SLI_PCIE_MSI_RCV* Write ... 669
Figure 13–1 CN71XX Interrupt Distribution (CIU_INTn_SUM0/4) ... 733
Figure 13–2 CN71XX Interrupt Distribution (CIU_INTn_SUM1) ... 734
Figure 13–3 CN71XX PCIe Interrupt Distribution ... 735
Figure 13–4 CN71XX GPIO Interrupt Distribution ... 736
Figure 13–5 CN71XX SUM2 Interrupt Distribution ... 736
Figure 15–1 PTP Clock ... 800
Figure 15–2 Digital Tuning to the PTP Clock ... 801
Figure 16–1 CN71XX Timer ... 812
Figure 16–2 Bucket Data Structure ... 814
Figure 17–1 RNG Diagram ... 823
Figure 18–1 RAD Block Diagram ... 831
Figure 18–2 RAD Calculations For a Pipe ... 831
Figure 18–3 Structure of RAD Instruction Queue in L2/DRAM ... 833
Figure 18–4 RAD Instruction Format ... 834
Figure 18–5 RAD CWORD Instruction Format ... 835
Figure 18–6 RAD OWORD Instruction Format ... 836
Figure 18–7 Non-Zero Detect Result ... 837
Figure 18–8 RAD IWORD Instruction Format ... 837
Figure 18–9 RAD RESP Instruction Format ... 838
Figure 19–1 Structure of BCH Instruction Queue in L2/DRAM ... 848
Figure 19–2 BCH Instruction Format ... 850
Figure 19–3 BCH CWORD Instruction Format ... 850
Figure 19–4 BCH OWORD Instruction Format ... 851
Figure 19–5 Output Parity/Correction Data Addressed by OWORD[PTR] ... 851

Cavium Confidential For Bowen Zang 02/27/2014

Figure 19–6 BCH IWORD Instruction Format 852
Figure 19–7 Input Parity/Correction Data Addressed by IWORD[PTR] 852
Figure 19–8 BCH RESP Instruction Format 853
Figure 19–9 Operation-Status Format 853
Figure S3–1 CN71XX I/O Interfaces 861
Figure 22–1 CN71/70XX PEM Block Diagram 894
Figure 22–2 Memory Space Requests: PEM0 in EP Mode, PEM1/2 in RC Mode 897
Figure 22–3 Memory Space Requests When PEM Ports are in RC Mode 901
Figure 22–4 Configuration EEPROM Operation 907
Figure 22–5 EEPROM Entry 907
Figure 22–6 EEPROM Example with Five-Entry Table 908
Figure 22–7 PCIe Port Reset Sequence in RC Mode 912
Figure 22–8 PCIe Port Warm Reset Sequence in EP Mode 914
Figure 22–9 Cold Reset Sequence in EP Mode 915
Figure 23–1 CN71XX SGMII Interface Block Diagram 1014
Figure 23–2 Remote-Device Flow Control 1019
Figure 23–3 CN71XX Flow Control 1021
Figure 23–4 Receive FIFO Parameters 1022
Figure 23–5 Generated Pause Packet Format 1022
Figure 23–6 DMAC Filter Algorithm 1024
Figure 23–7 Converting Code Groups to Bytes 1025
Figure 23–8 100 Mbit Byte Stream Reduction Example 1026
Figure 23–9 Conversion from Byte Stream to Packet Bytes 1027
Figure 23–10 Packet Formats 1028
Figure 23–11 Outbound Conversion to Code Groups 1031
Figure 23–12 PCS Loopback 1046
Figure 23–13 RX FIFO Values 1075
Figure 24–1 CN71XX Packet Interfaces Block Diagram 1094
Figure 24–2 Remote-Device Flow Control 1095
Figure 24–3 CN71XX Flow Control 1101
Figure 24–4 Receive FIFO Parameters 1102
Figure 24–5 Generated Pause Packet Format 1103
Figure 24–6 Generated PFC Frame Format 1104
Figure 24–7 Converting Code Groups to Bytes 1110
Figure 24–8 Conversion from Byte Stream to Packet Bytes 1110
Figure 24–9 Packet Formats 1111
Figure 24–10 Outbound Conversion to Code Groups 1114
Figure 24–11 PCS Loopback 1127
Figure 24–12 RX FIFO Values 1154
Figure 25–1 CN71XX Packet Interfaces Block Diagram 1174
Figure 25–2 Remote-Device Flow Control 1175
Figure 25–3 CN71XX Flow Control 1177
Figure 25–4 Receive FIFO Parameters 1177
Figure 25–5 Generated Pause Packet Format 1178
Figure 25–6 Packet Formats 1181
Figure 26–1 SATA Subsystem Block Diagram 1225
Figure 27–1 TDM/PCM Block Diagram 1255
Figure 27–2 FSYNC Sampling 1258
Figure 27–3 BCLK Waveforms 1259
Figure 27–4 TDM-Engine Block Diagram 1259
Figure 27–5 Data Sampling/Data Driving 1261
Figure 28–1 Boot-Bus Hardware 1274
Figure 28–2 Static-Timed Read Sequence 1280
Figure 28–3 Static-Timed Write Sequence 1281
Figure 28–4 Dynamic-Timed Read Sequence 1282
Figure 28–5 Dynamic-Timed Write Sequence 1282
Figure 28–6 Sample Boot-Bus Connection 1 (ALE, 8W) 1283
Figure 29–1 eMMC Interface Block Diagram 1298
Figure 29–2 GPIO_8 as VDD Control 1305
Figure 29–3 GPIO_8 as EMMC_RSTn 1305

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 29–4 eMMC and SD Initialization Sequence ... 1309
Figure 30–1 NAND Flash Interface ... 1325
Figure 30–2 Sixty-Four-bit NDF Store Operation ... 1326
Figure 30–3 BUS_ACQ Signal Timing ... 1331
Figure 30–4 BUS_REL Signal Timing ... 1332
Figure 30–5 CHIP_EN Signal Timing ... 1333
Figure 30–6 CHIP_DIS Signal Timing ... 1334
Figure 30–7 CLE_CMD Signal Timing ... 1335
Figure 30–8 ALE_CMD Signal Timing ... 1337
Figure 30–9 RD_CMD Signal Timing ... 1339
Figure 30–10 RD_EDO_CMD Signal Timing ... 1341
Figure 30–11 ALE_CMD Signal Timing ... 1342
Figure 30–12 WAIT_STATUS Signal Timing ... 1345
Figure 30–13 WAIT_STATUS_ALE Signal Timing ... 1345
Figure 31–1 USB Subsystem Diagram ... 1359
Figure 32–1 SMI/MDIO Transaction as Master ... 1474
Figure 32–2 SMI/MDIO Interface Read Timing Characteristics ... 1475
Figure 32–3 SMI/MDIO Interface Write Timing Characteristics ... 1476
Figure 33–1 GPIO Cell ... 1483
Figure 34–1 Serial Data Format ... 1497
Figure 34–2 Receiver Serial Data Sample Points ... 1497
Figure 34–3 AutoRTS Timing ... 1499
Figure 34–4 AutoCTS Timing ... 1499
Figure 35–1 TWSI Architecture ... 1518
Figure 35–2 Supported TWS HLC Transaction Steps ... 1518
Figure 35–3 Legend for Diagrams ... 1519
Figure 35–4 HLC as Master, Initial-Address Step ... 1520
Figure 35–5 HLC as Master, Address-Extension Step ... 1520
Figure 35–6 HLC as Master, Read-Bytes Step ... 1521
Figure 35–7 HLC as Master, Write-Bytes Step ... 1522
Figure 35–8 HLC as Slave, Initial-Address Step ... 1523
Figure 35–9 HLC as Slave, Address-Extension Step ... 1524
Figure 35–10 HLC as Slave, Read/Write-Bytes Step ... 1525
Figure 35–11 Master-Mode Flow Diagram ... 1532
Figure 35–12 Slave-Mode Flow Diagram ... 1533
Figure 36–1 SPI/MPI Overview ... 1546
Figure 36–2 SPI_CS<3:0> and MPI_CFG[CSLATE] ... 1549
Figure 37–1 PROPOSED Power Sequencing for Default Mode ... 1565
Figure 37–2 PROPOSED Power Sequencing for SerDes Common-Clock Mode ... 1566
Figure 38–1 CK vs Command Timing Diagram ... 1575
Figure 38–2 WRITE Timing Diagram ... 1575
Figure 38–3 RGMII Transmit Multiplexing and Timing Diagram ... 1584
Figure 38–4 RGMII Receive Multiplexing and Timing Diagram ... 1584
Figure 38–5 EEPROM AC Timing ... 1586
Figure 38–6 EMMC Command Timing Diagram ... 1587
Figure 38–7 EMMC Data Timing Diagram ... 1587
Figure 38–8 JTAG Signal I/O Timing Diagram ... 1588
Figure 38–9 MPI/SPI Signal I/O Timing Diagram ... 1589
Figure 38–10 TWSI Signal Parameters ... 1590
Figure 39–1 640-LBGA Package Diagram (Top View) ... 1594
Figure 39–2 640-LBGA Package Diagram (Bottom and Side Views) ... 1595
Figure 41–1 640 BGA Ball Assignment Diagram (Bottom View) ... 1610
Figure 41–2 640 LBGA Preliminary Signal Map (Top View) ... 1611
Figure 41–3 CN71XX Signal Map, Upper Left Quadrant (Top View) ... 1612
Figure 41–4 CN71XX Signal Map, Upper Right Quadrant (Top View) ... 1613
Figure 41–5 CN71XX Signal Map, Lower Left Quadrant (Top View) ... 1614
Figure 41–6 CN71XX Signal Map, Lower Right Quadrant (Top View) ... 1615

Cavium Confidential For Bowen Yang ALU 02/27/2014

List of Tables

Table 1	Revision History	29
Table 2	MIPS Publications	33
Table 1–1	SerDes Clocking Requirements	42
Table 1–2	CN71XX CSR Types	54
Table 2–1	MIPS Publications	57
Table 2–2	CPU Visible State Resident in Each cnMIPS III Core	63
Table 2–3	CPU Arithmetic (and Move) Instructions	65
Table 2–4	CPU Branch and Jump Instructions	66
Table 2–5	CPU Load, Store, and Memory Control Instructions	66
Table 2–6	CPU Move and Privileged Instructions	67
Table 2–7	Coprocessor Move Instructions	67
Table 2–8	CPU Arithmetic Instructions	70
Table 2–9	CPU Branch and Jump Instructions	70
Table 2–10	CPU Instruction Control Instructions	71
Table 2–11	CPU Load, Store, and Memory Control Instructions	71
Table 2–12	SYNC Instruction Stype Encodings	74
Table 2–13	CPU Logical Instructions	74
Table 2–14	CPU Insert/Extract Instructions	74
Table 2–15	CPU Move Instructions	75
Table 2–16	CPU Shift Instructions	75
Table 2–17	CPU Trap Instructions	75
Table 2–18	FPU Arithmetic Instructions	75
Table 2–19	FPU Branch Instructions	76
Table 2–20	FPU Compare Instructions	76
Table 2–21	FPU Convert Instructions	76
Table 2–22	FPU Load, Store, and Memory Control Instructions	76
Table 2–23	FPU Move Instructions	76
Table 2–24	Coprocessor Branch Instructions	77
Table 2–25	Coprocessor Execute Instructions	77
Table 2–26	Coprocessor Load and Store Instructions	77
Table 2–27	Coprocessor Move Instructions	77
Table 2–28	Privileged Instructions	77
Table 2–29	EJTAG Instructions	78
Table 2–30	cnMIPS III Core Physical Addresses	80
Table 2–31	Physical Memory Addressing Operations	87
Table 2–32	Execution Ordering Rules	87
Table 2–33	cnMIPS™ III cores Floating Point Modes	95
Table 2–34	Core Error Handling Mechanisms	98
Table 2–35	Coprocessor 0 Register Summary: Root Context	100
Table 2–36	Coprocessor 0 Register Summary: Guest Context	104
Table 2–37	Hypervisor Exception Cause Values	116
Table 2–38	Performance Counter Control Register Events	126
Table 2–39	Performance Counter Control Register Events	148
Table 2–40	ErrCtl Behavior	171
Table 2–41	EJTAG DRSEG Registers Summary	188
Table 2–42	EJTAG TAP Registers Summary	194
Table 2–43	Integer Instruction Classes	197
Table 2–44	Pipeline Classification for CP0 Register Access	198
Table 2–45	Floating-Point Instruction Classes	199
Table 2–46	Instruction Latency/Busy Cycles for Pipeline 0	200
Table 2–47	Instruction Latency/Busy Cycles for Pipeline 1	201
Table 2–48	3DES Unit Latencies	204
Table 2–49	KAS Unit Latencies	204

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 2–50 AES Unit Latencies 204
Table 2–51 CVM_MT_GFM_XORMUL1 Latencies 204
Table 2–52 CRC Unit Latencies 205
Table 2–53 Hash Latencies 205
Table 2–54 SNOW3G Latencies 205
Table 2–55 SMS4 Latencies 205
Table 2–56 CAMELLIA Unit Latencies 206
Table 2–57 ZUC Latencies 206
Table 2–58 MIPS Debug Features 206
Table 2–59 EJTAG Debug Features 206
Table 2–60 Breakpoint Match Behavior 207
Table 2–61 cnMIPS III Core Exceptions 212
Table 3–1 Coherent Memory Interconnect Transactions 223
Table 3–2 L2C Registers 239
Table 3–3 Supported RD/SEL Values 245
Table 4–1 Chip-Select Signals Usage Summary 271
Table 4–2 Read/Write ODT Configuration 272
Table 4–3 DRAM 64-Bit ECC Code 282
Table 4–4 DRAM Extended 64-bit ECC Code (LMC(0)_CONFIG[ECC_ADR]=1) 282
Table 4–5 Address to Gray Code Mapping 282
Table 4–6 Sample Programming Settings 284
Table 4–7 LMC Registers 286
Table 4–8 Mask-to-WODT Conversion 315
Table 4–9 LMC(0)_SLOT_CTL0 Hardware-Calculated Minimums 320
Table 4–10 LMC(0)_SLOT_CTL1 Hardware-Calculated Minimums 322
Table 4–11 LMC(0)_SLOT_CTL2 Hardware-Calculated Minimums 323
Table 4–12 Mask-to-RODT Conversion 330
Table 5–1 IOB Registers 353
Table 6–1 OCLA Complexes 366
Table 6–2 Example Capture Sequence 371
Table 6–3 OCLA Registers 375
Table 6–4 DTX Registers 386
Table 7–1 POW Operations 398
Table 7–2 POW 11-Bit ECC Code 427
Table 7–3 POW Registers 428
Table 8–1 FPA Registers 444
Table 9–1 PKT_INST_HDR Parsing Selection Override 460
Table 9–2 Receive Errors/Exceptions 467
Table 9–3 SKIP Maximum Values (No Caveats Needed) 469
Table 9–4 SKIP Value Ranges (QOS L4 Watcher Caveat) 469
Table 9–5 SKIP Value Ranges (QOS DiffServ/Watcher Caveat) 470
Table 9–6 SKIP Maximum Values (All Caveats) 470
Table 9–7 Port/QOS Value Mapping to [INT] and [ENB] Bits 506
Table 9–8 PIP Registers 508
Table 9–9 IPD Registers 533
Table 9–10 Port QOS Interrupt Registers 546
Table 10–1 Reset/Default Internal Ports-to-Engines Mapping 563
Table 10–2 PKO-Internal Port to PKO-External Port Mapping for DPI and Loopback 563
Table 10–3 Selectable External Backpressure Sources 566
Table 10–4 Internal Buffer Space 569
Table 10–5 PKO Registers 571
Table 10–6 GMX Per-Engine Local Storage 573
Table 11–1 HFA Registers 616
Table 12–1 DPTR Formats 635
Table 12–2 PKO Output Queues to DPI Rings Example Mapping 641
Table 12–3 DPI DMA Transfer Types 648
Table 12–4 DPI Registers 678
Table 12–5 SLI Registers 693
Table 13–1 CIU Registers 743
Table 14–1 MIO_FUS, RST Registers 787
Table 15–1 PTP Registers 806
Table 16–1 Timer Registers 816

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 17–1 RNM Registers ... 826
Table 18–1 RAD Registers ... 840
Table 19–1 BCH Registers ... 854
Table 20–1 Key Registers ... 858
Table S3–1 DLM Configurations ... 862
Table 21–1 DLM0 Data-Rate Configuration Table ... 865
Table 21–2 DLM2 SATA Data-Rate Configuration Table ... 866
Table 21–3 DLM1/2 PCIe Data-Rate Configuration Table ... 866
Table 21–4 GSER Registers ... 867
Table 22–1 Transaction ID Fields ... 905
Table 22–2 MIO_RST_CNTLn Reset Settings ... 911
Table 22–3 PCICONFIGEP0 Registers ... 920
Table 22–4 PCICONFIGRC(0..2) Registers ... 958
Table 22–5 PEM Registers ... 996
Table 23–1 Recommended GMX Configuration Settings ... 1018
Table 23–2 Flow Control Conditions ... 1021
Table 23–3 FIFO Memory Allocation ... 1030
Table 23–4 Receive Errors/Exceptions ... 1037
Table 23–5 Transmit Errors/Exceptions ... 1039
Table 23–6 PCS Errors/Exceptions ... 1040
Table 23–7 Description of Different Enable Modes ... 1041
Table 23–8 CN71XX Statistics Gathering ... 1045
Table 23–9 GMX Registers ... 1047
Table 23–10 GMX Decisions ... 1055
Table 23–11 PCS Registers ... 1083
Table 24–1 Flow Control Conditions ... 1102
Table 24–2 Receive Errors/Exceptions ... 1119
Table 24–3 Transmit Errors/Exceptions ... 1122
Table 24–4 Description of Different Enable Modes ... 1124
Table 24–5 CN71XX RXAUI Statistics ... 1126
Table 24–6 GMX Registers ... 1129
Table 24–7 GMX Decisions ... 1137
Table 24–8 PCSX Registers ... 1167
Table 25–1 Flow Control Conditions ... 1177
Table 25–2 Transmit-Packet Options ... 1183
Table 25–3 Receive Errors/Exceptions ... 1186
Table 25–4 Transmit Errors/Exceptions ... 1188
Table 25–5 AGL Statistics Gathering ... 1191
Table 25–6 AGL Registers ... 1192
Table 25-7 GMX Decisions ... 1200
Table 26–1 Maximum ACLK Frequency ... 1226
Table 26–2 SATA Registers (UCTL) ... 1231
Table 26–3 UAHC Register Address Map ... 1237
Table 26–4 SATA Registers (UAHC) ... 1237
Table 27–1 Signal Functionality ... 1255
Table 27–2 BCLK/FSYNC Generation ... 1256
Table 27–3 Sample BCLK Frequency ... 1256
Table 27–4 Superframe Memory-Region Example ... 1262
Table 27–5 LSB/MSB Bit Placement ... 1262
Table 27–6 PCM/TDM Registers ... 1266
Table 28–1 Exception Vectors ... 1275
Table 28–2 PCIe Expansion-ROM Address Range ... 1275
Table 28–3 Boot-Bus Pin Straps ... 1278
Table 28–4 Bus Operation ... 1278
Table 28–5 Boot-Bus Timing Parameters ... 1279
Table 28–6 Boot-Bus Fixed-Timing Parameters ... 1279
Table 28–7 Boot-Bus Registers ... 1287
Table 29–1 eMMC and Boot-bus Shared Signals ... 1299
Table 29–2 Boot-bus Signal Mapping to eMMC Devices ... 1299
Table 29–3 Command and Response Types ... 1301
Table 29–4 eMMC vs. SD Command/Response Types ... 1302
Table 29–5 eMMC Registers ... 1311
Table 30–1 NAND Flash Signals ... 1324

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 30–2 NAND Flash Commands 1329
Table 30–3 NAND Flash Registers 1351
Table 31–1 Power Supply Requirements 1367
Table 31–2 Reference-Clock Configurations 1367
Table 31–3 USBH-UCTL Registers 1375
Table 31–4 UAHC Register-Address Map 1393
Table 31–5 USBH-UAHC Registers 1393
Table 32–1 SMI Registers 1476
Table 33–1 GPIO Registers 1486
Table 33–2 Clock Speed Output 1492
Table 34–1 Shared UART/MPI/SPI Signals 1496
Table 34–2 UART Registers 1501
Table 35–1 Slave Internal-Address Register 1524
Table 35–2 Validity Indication for Slave Read Operations 1525
Table 35–3 TWSI_STAT Status Codes (after 7-Bit or First Part of 10-Bit Address) 1526
Table 35–4 TWSI_STAT Status Codes (after Second Part of 10-Bit Address) 1527
Table 35–5 TWSI_STAT Status Codes (after Repeated START Transmission) 1528
Table 35–6 TWSI_STAT Status Codes (Master Receive Mode) 1529
Table 35–7 TWSI_STAT Status Codes (Master Receive – Data Received) 1530
Table 35–8 TWSI_SLAVE_ADD Bit Description 1534
Table 35–9 TWSI_SLAVE_EXTADD Bit Description 1535
Table 35–10 TWSI_CTL Bit Description 1535
Table 35–11 TWSI Core-State Codes 1537
Table 35–12 TWSI_CLK Bit Description 1538
Table 35–13 TWSI_CLKCTL Bit Description 1538
Table 35–14 TWSI Registers 1539
Table 36–1 SPI/MPI Signals 1547
Table 36–2 SPI/MPI Registers 1555
Table 37–1 Voltage Absolute Maximum Ratings 1562
Table 37–2 Absolute Maximum Storage Temperatures 1562
Table 37–3 Recommended Operating Temperatures 1563
Table 37–4 Recommended Operating Supply Voltages 1563
Table 37–5 Power Supplies for Multiple PLLs 1564
Table 37–6 CN71XX Maximum Core Power Specification 1567
Table 37–7 Typical DDR3 Memory-Interface Supply Current 1567
Table 37–8 Worst-Case DDR3 Memory-Interface Supply Current 1567
Table 37–9 Miscellaneous Supply Current 1568
Table 37–10 DLM Interface Thermal Supply Current 1568
Table 37–11 Input-Receiver Specifications 1569
Table 37–12 Output-Driver Specifications 1569
Table 37–13 3.3V/2.5V CMOS Point-to-Point I/O for Miscellaneous Interfaces 1570
Table 37–14 2.5V/3.3V CMOS for RGMII/MII/SMI/MDIO Interface 1570
Table 38–1 Core/Coprocessor/DDR Reference-Clock 1572
Table 38–2 DDR3 Output-Clock Timing Specifications 1573
Table 38–3 DDR3 Output Timing Specifications 1574
Table 38–4 DDR3 Input Timing Specifications 1576
Table 38–5 SerDes Common Reference-Clock Inputs 1577
Table 38–6 DLM Reference-Clock Input: PCIe (Gen1) 1578
Table 38–7 PCIe (Gen1) TX Specifications 1578
Table 38–8 PCIe (Gen1) RX Specifications 1578
Table 38–9 DLM Reference-Clock Input: PCIe (Gen2) 1579
Table 38–10 PCIe (Gen2) TX Specifications 1579
Table 38–11 PCIe (Gen2) RX Specifications 1579
Table 38–12 DLM Reference-Clock Input: SGMII 1580
Table 38–13 SGMII TX Specifications 1580
Table 38–14 SGMII RX Specifications 1580
Table 38–15 DLM Reference-Clock Input: RXAUI 1581
Table 38–16 RXAUI TX Specifications 1581
Table 38–17 RXAUI RX Specifications 1581
Table 38–18 DLM Reference-Clock Input: SATA 1582
Table 38–19 SATA TX Specifications 1582
Table 38–20 SATA RX Specifications 1582

Cavium Confidential For Boyen Yang 02/27/2014

Table 38–21 RGMII Reference-Clock Input 1583
Table 38–22 RGMII Clock Periods 1583
Table 38–23 RGMII Timing Parameters Without DLL Circuit Engaged 1583
Table 38–24 RGMII Timing Parameters With DLL Circuit Engaged 1584
Table 38–25 EEPROM Cycle Timing 1585
Table 38–26 EMMC Command-Timing Parameters 1587
Table 38–27 EMMC Data-Timing Parameters 1587
Table 38–28 JTAG Signal I/O Timing Parameters 1588
Table 38–29 MPI/SPI Signal I/O Timing Parameters 1589
Table 38–30 TWSI Signal Parameters 1590
Table 38–31 USB 3.0 TX Specifications 1591
Table 39–1 Thermal Package Specification for CN71XX 1596
Table 40–1 CN71XX Interfaces 1597
Table 40–2 CN71XX Signal Types 1598
Table 40–3 DRAM Interface Signals 1599
Table 40–4 DLM Common Signals 1600
Table 40–5 PCIe-Specific Signals 1600
Table 40–6 DLM Signals 1601
Table 40–7 RGMII Signals 1601
Table 40–8 USB Interface Signals 1602
Table 40–9 SMI/MDIO Interface Signals 1602
Table 40–10 GPIO Interface Signals 1603
Table 40–11 PCM/TDM Interface Signals 1603
Table 40–12 TWSI Signals 1604
Table 40–13 Clock Signals 1604
Table 40–14 Boot Bus Signals 1605
Table 40–15 SPI Signals 1605
Table 40–16 UART Interface Signals 1606
Table 40–17 eJTAG/JTAG Signals 1606
Table 40–18 Miscellaneous Signals 1607
Table 40–19 Power/Ground/Reserved/No Connect Signals 1607
Table A–1 Cavium-Specific Instructions 1628
Table 42–2 Hint-Field Values 1666

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Preface

In this Preface

- Revision History
- Related Documentation
- Symbols Used
- User Comments
- About This Book
- MIPS Technologies
- Contact Us

Revision History

The revision history is shown in Table 1.

Table 1 Revision History

Version	Date	Changes

0.95E	7/2013	• Initial version of CN71XX HRM.
		• Cores, chapter 2:
		  • in Section 2.15.3, added text to Root.PowThrottle[POWLIM]; added text to Root.CvmMemCtl2 [SCCTR].
		  • in Section 2.15.4, removed Guest.PowThrottle; added text to Guest.CvmMemCtl2 [SCCTR].
		• Added LMC, chapter 4.
		• Removed WOL chapter (WOL descriptions added to SGMII and RGMII chapters).
		• Added PEM, chapter 19
		• SGMII, chapter 20: added Section 23.5, “Wake-on-LAN (WOL)”
		• RGMII, chapter 22: added Section 22.4, “Wake-on-LAN (WOL)”
		• TDM, chapter 24: in Table 27–1, corrected the association between the TDM signals and the GPIO signals.
		• Added USB, chapter 28.
		• Electrical Specifications, chapter 34, added Tables 37–1 and 37–2.
		• Signal Description, chapter 37: in Table 40–11, corrected association between the TDM signals and the GPIO signals.

0.96E	7/2013	• Introduction, chapter 1: in Figure 1–2, corrected PCIE* signal names.
		• LMC, chapter 4: added the CSRs.
		• Added BCH, chapter 19
		• PEM, chapter 21: changed V3.0-compliant with V2.0-compliant.
		• Added RXAUI chapter 23.
		• SGMII/QSGMII, chapter 21: in Section 23.12 updated PCS registers.
		• Electrical Specs, chapter 36:
		  • In Table 37–4, corrected the Min and Max for VDDH_DLM*n*; removed the original footnote 3 and added new footnote 3
		  • in Table 37–9, improved the description of VDD_DLM*n*/VDDTX_DLM*n*.
		  • In 37.5.1, corrected the text before Table 37–12.
		• AC Characteristics, chapter 37:
		  • In Section 38.3, replaced TBDs with values in various tables.
		  • Added SATA specifications in Section 38.3.6.
		• Signal Descriptions, chapter 39: corrected PCIE0* signal names in Table 40–5.
		• Appendix B: added Order number for 1.5 GHz part, and corrected the definition of AAP.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 1 Revision History (Continued)

Version	Date	Changes
0.961E	8/2013	• Added DLM chapter, now Chapter 21. • Electrical Specifications, 37, in Table 37–9, corrected the values for USB*_VDD33.

0.98E	10/2013	• Cores, chapter 2: updated from Sections 2.5 through 2.22.
		• L2C, chapter 3:
		  • in Section 3.3, updated L2C_INTSN_E; added L2C_TAD_PRF_SEL_E.
		  • in Section 3.4, added L2C_XMC_CMD[QOS], L2C_TAD(0..7)_TAG[SBLKDTY]; updated L2C_BIG_CTL[MAXDRAM].
		• LMC, chapter 4:
		  • in Section 4.8.6, added a new step 1.
		  • in Section 4.13, modified the formula for DDR PLL speed and corrected/updated Table 4–6.
		• POW, chapter 7: updated the CSRs, Section 7.13.
		• FPA, chapter 8: updated the CSRs, Section 8.2.
		• PKO, chapter 10: updated text and CSRs.
		• CIU, chapter 13: in Figure 13–2 corrected the inputs.
		• Reset, chapter 14: in Section 14.2, removed RST_OCX and added RST_PP_POWER.
		• DLM, chapter 21: removed 156.25 MHz from Table 21–3.
		• Boot Bus, chapter 28:
		  • Throughout the chapter, removed incorrect references to BOOT_AD signals outside of the <7:0> range and corrected references to the BOOT_ALE<3:0> signals.
		  • In Section 28.6, changed the text from 16-bit wide boot bus to 8-bit wide.
		  • In Section 28.8, updated the registers to reflect the design.
		• eMMC, chapter 29: In Sections 29.5.7 and 29.5.8, corrected step 1.
		• NAND Flash, chapter 30:
		  • Throughout the chapter, removed incorrect references to BOOT_AD signals outside of the <7:0> range.
		  • In Section 30.2.12, updated the WR_CMD equation.
		  • Updated the CHIP_EN line of the sequences in Sections 30.3.1, 30.3.2, and 30.3.3; removed section 29.3.1 (early boot reset sequence from a NAND flash device).
		• USB, chapter 31: corrected text and tables in Section 31.5.
		• SMI, chapter 32: in Section 32.2, corrected SMI_DRV_CTL[PCTL,NCTL].
		• GPIO, chapter 33:
		  • updated Figure 33–1.
		  • in Section 33.3, updated the CSRs.
		• AC Characteristics, chapter 38:
		  • for EEPROM, corrected Table 38–25 and Figure 38–5
		  • corrected Figure 38–9 to show timing for SPI_CK (IDLELO = 1).
		• Signal Descriptions, chapter 40: in Table 40–14, corrected the Up/DN column for BOOT_AD<7:0>.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 1 Revision History (Continued)

Version	Date	Changes

0.985E	11/2013	• CMI-L2C, chapter 3: added chapter text and updated the registers.
		• LMC, chapter 4:
		  • Updated Section 4.8.7, Section 4.8.8, Section 4.8.9
		  • In Section 3.4, updated LMC(0)_NXM[CS_MASK], LMC(0)_RESET_CTL, LMC(0)_DLL_CTL3[DLL90_BYTE_SEL]; LMC(0)_TIMING_PARAMS1, LMC(0)_MODEREG_PARAMS0, LMC(0)_SLOT_CTL*, LMC(0)_RODT_MASK; added LMC(0)_BIST_CTL, LMC(0)_INT_EN.
		• Added chapter 5, I/O Interconnect (IOI), I/O Bridge (IOB) and Fetch and Add Unit (FAU).
		• PIP, chapter 9: in Section 9.5, added a new code of 15 to WORD2[RE]; corrected Section 9.8, Per-Port and QOS Threshold Interrupts.
		• DPI/SLI, chapter 12: in Section 12.10, updated DPI_INT_REG, DPI_INT_EN, DPI_CTL, DPI_REQ_ERR_RST_EN, DPI_SLI_PRT(0..2)_CFG, DPI_PKT_ERR_RSP, DPI_DMA_PP*_CNT; removed DPI_DMA0/1_TIM in Section 12.11, removed SLI_DBG_SELECT, SLI_DBG_DATA,SLI_BIST_STATUS; updated SLI_DATA_OUT_CNT. Corrected the field names of SLI_MSI_RCV*, SLI_S2M_PORT(0..2)_CTL; added SLI_MAC_CREDIT_CNT2.
		• CIU, chapter 13: corrected miscellaneous figures and text throughout the chapter; in Section 13.10, added CIU_PP_RST_PENDING. Updated CIU_PCI_INTA.
		• RST, chapter 14: in Section 14.2, updated MIO_FUS_DAT3[L2C_CRIP], MIO_FUS_TGG.
		• Precision Time Protocol (PTP) Clocks and Timestamping, chapter 15: updated the chapter text; in Section 15.10, updated MIO_PTP_CLOCK_CFG.
		• TIM, chapter 16: in Section 16.4, updated TIM_REG_READ_IDX.
		• RNG/RNM, chapter 17: in Section 17.2, updated RNM_CTL_STATUS.
		• RAD, chapter 18: in Section 18.7, updated RAD_REG_CTL.
		• BCH, chapter 19:
		  • Corrected Figure 19–1.
		  • In the paragraph immediately following Figure 19–1, corrected the number of instructions shown in Figure 19–1.
		  • In Section 19.3 in the description for dbell_cnt, changed “from 4 to 36 words” to “always four words.”
		  • In Section 19.5, updated BCH_CTL.
		• KEY, chapter 20: in Section 20.1, updated the registers.
		• SerDes Dual-Lane Module (DLM), chapter 21: in Section 21.2, updated GSER(0)_DLM(0..2)_MPLL_MULTIPLIER, GSER(0)_DLM(0)_REF_SSP_EN, GSER(0)_PCIE_PIPE_RST; added multiple GSER* registers.
		• PEM, chapter 22: in Section 22.7, updated PCIEEP(0..2)_CFG002[RID],PCIE*(0..2)_CFG065[UCIES], PCIERC(0..2)_CFG020[PVM], PCIERC(0..2)_CFG029[ETFS], PCIERC(0..2)_CFG040[CDL], PCIERC(0..2)_CFG452[LME], PCIERC(0..2)_CFG455[M_CFG1_FILT] PCIERC(0..2)_CFG466, ; added PCIEEP(0..2)_CFG024/025.In Section 22.8, updated PEM(0..2)_TLP_CREDITS[SLI_P], PEM(0..2)_CFG.
		• SGMII / QSGMII / 1000BASE-X Interfaces, chapter 23: in Section 23.11 updated GMX(0..1)_TX(0..3)_PAUSE_PKT_INTERVAL, GMX(0..1)_INF_MODE[MODE].
		• RXAUI, chapter 24: removed references to GMX1. In Section 24.10, updated the reset value of GMX(0)_TX(0..3)_PAUSE_PKT_INTERVAL[INTERVAL]; updated GMX(0)_INF_MODE; added GMX(0)_TX_INT_REG[XCHANGE].
		• RGMII, chapter 25: in Section 25.9, updated AGL_GMX_RX(0)_FRM_CHK[LENERR], AGL_GMX_RX(0)_FRM_MIN, AGL_GMX_RX(0)_FRM_MAX, AGL_GMX_TX_OVR_BP, AGL_GMX_TX_BP, AGL_GMX_RX_TX_STATUS, AGL_PRT(0)_CTL, AGL_PRT(0)_CTL.
		• Boot bus, chapter 28: in Section 28.8, updated MIO_BOOT_REG_CFG(1..7), MIO_BOOT_BIST_STAT.
		• EMMC, chapter 29: in Section 29.6, updated MIO_EMM_CFG, MIO_EMM_BUF_IDX.
		• USB, chapter 31, in Section 31.7, updated USBDRD(0..1)_UCTL_CTL, USBDRD(0..1)_UCTL_BIST_STATUS, USBDRD(0..1)_UAHC_GHWPARAMS0, USBDRD(0..1)_UAHC_GTXFIFOSIZ(0..3), USBDRD(0..1)_UAHC_GRXFIFOSIZ(0..2).
		• GPIO, chapter 33: in Section 33.1.4, updated the steps for configuring the GPIO clock feature.
		• Electrical Specifications, chapter 37: in Table 37–13, corrected the value for I_{LEAK} (DC); added the supply current to Table 37–9 for VDD_DLMC.
		• Signal Descriptions, chapter 40: in Table 40–3, corrected the pullup and pulldown values for DDR_COMP_UP/DN.

0.986E	12/2013	• RST, chapter 14: corrected the text in Section 14.1.
		• RGMII, chapter 25: in Section 25.7.1, updated step 2a in the AGL bring-up sequence.
		• Electrical Specifications, chapter 37:
		  • In Table 37–9, updated the supply current for USB0/1_VDD33.
		  • Corrected values in Table 37–11, Input-Receiver Specifications for DDR.
		•

Cavium Confidential For Review Only 10/17/2014

Table 1 Revision History (Continued)

Version	Date	Changes

0.99E	1/2014	• Introduction, chapter 1:
		  • in the list of Features, added ZUC to the hardware cryptographic bullet.
		  • made corrections to Figures 1–4 and 1–4.
		  • added the clock-selection registers to Table 1–1.
		  • in Table 1–2, added NCB to the CSR types for SATA.
		• Cores, chapter 2: replaced mention of L2C_COP0_MAP with L2C_COP0_ADDR and L2C_COP0_DATA.
		• LMC, chapter 4:
		  • in Overview, changed 16GB of DDR3 SDRAM to 8GB.
		  • in Section 4.1, corrected text in the first paragraph, removed DIMM from Figure 4–1, removed the last bullet about DIMMs, and corrected text in the paragraph immediately following the bullets.
		  • in Section 4.8, changed “eight” to “eleven” in the paragraph immediately following the numbered list.
		  • in Section 4.16, modified the Field Description of LMC(0)_SEQ_CTL[SEQ_SEL] and removed RDIMM text from the descriptions of LMC(0)_DIMM(0..1)_PARAMS, LMC(0)_DIMM_CTL, and LMC(0)_DDR4_DIMM_CTL.
		• IOI/IOB, chapter 5:
		  • added AGL and SATA to Figure 5–1.
		  • removed mention of SATA from Figure 5–2.
		• OCLA, chapter 6:
		  • in Section 6.1, corrected the Feature list.
		  • in Section 6.2, corrected the number in the first sentence.
		  • in Section 6.2.6, corrected the descriptions for OCLA_CAP_CTL_S and corrected the text following this structure.
		• PIP/IPD, chapter 9: in Section 9.1, removed RXAUI from the Port 16–19 bullet.
		• PKO, chapter 10: corrected the contents of Table 10–3 and the bullets that follow; updated the CSRs.
		• DPI/SLI, chapter 12: in Section 12.10, corrected the Description of DPI_DMA_CONTROL[COMMIT_MODE].
		• CIU, chapter 13:
		  • removed extraneous signals from Figure 13–1.
		  • added BCH path to Figure 13–3.
		• RST, chapter 14: corrected the text in section 14.1 and removed the INTSN enumeration.
		• PTP, chapter 15: in Section 15.2, in step 6 of the example, changed 0x1 to 0x2.
		• GSER, chapter 21: updated the CSRs.
		• SGMII, chapter 23: in Section 23.4.3, reordered the text and added Figures 23–2 and 23–3.
		• RXAUI, chapter 24:
		• RGMII, chapter 25:
		  • updated Section 25.3.1.
		  • updated AGL_PRT(0)_CTL
		• SATA, chapter 26: in Section 26.8.1, for SATA_UCTL_CTL[A_CLKDIV_SEL], updated the requirements for the coprocessor-clock frequency.
		• BOOT, chapter 28: removed mention of 16-bit mode and nonmultiplexed mode.
		• EMMC, chapter 29: updated MIO_EMM_CFG.
		• USB, chapter 31: updated the CSRs.
		• AC Characteristics, 38: added timing specifications to USB, Section 38.13.
		• Mechanical Specs, chapter 39: updated package specifications in Figure 39-2.
		• Signal Descriptions, chapter 40: in Table 39-18, added descriptions for VRM_I2_CLK and VRM_I2_DATA.

0.991E	1/2014	• Cores, chapter 2:
		  • added TLBR and TLBWI to Table 2–6.
		  • updated the descriptions of Root.Ebase and Guest.Ebase in Section 2.15.
		• LMC, chapter 4: in Section 4.16, added new field to LMC(0)_MODEREG_PARAMS2.
		• DPI/SLI, chapter 12: in Section 12.11, updated multiple CSRs.
		• CIU, chapter 13: in Section 13.10, updated multiple CSRs.
		• PEM, chapter 22:
		  • corrected the descriptions of cs2 and address in Figure 22–5.
		  • in Sections 22.7.1 and 22.7.2, updated PCIE*(0..2)_CFG031.
		• EMMC, chapter 29: in Section 29.6, updated MIO_EMM_WDOG.
		• USB, chapter 31: updated multiple CSRs.
		• GPIO, chapter 33: in Section 33.3, added table to the description of GPIO_CLK_SYNCE(0..1).
		• Electrical Characteristics, chapter 37: in Tables 37–4 and 37–6, changed core voltage for 1.2 GHz to 0.88V; updated current values.

Cavium Confidential For Bowen Yang ALU 02/03/2014

Related Documentation

For additional information, refer to these documents.

Cavium, Inc.'s OCTEON III chip architecture is based on officially licensed MIPS Inc. Technology. For reference to MIPS Technology, you should procure the following books:

Table 2 MIPS Publications

Publication	Document Number
EJTAG Specification Revision 6.10, February 17, 2013	MD00047
MIPS64® Architecture For Programmers Volume I: Introduction to the MIPS64® Architecture Revision 5.01, December 15, 2012	MD00083
MIPS64® Architecture For Programmers Volume II: The MIPS64® Instruction Set Revision 5.01, December 15, 2012	MD00087
MIPS64® Architecture For Programmers Volume III: The MIPS64® and microMIPS64™ Privileged Resource Architecture Revision 5.01, December 16, 2012	MD00091
MIPS64® Architecture For Programmers Volume IV-i: Virtualization Module of the MIPS64® Architecture, Revision 1.02, February 19, 2013	MD00847

For additional information regarding MIPS® technology, refer to "MIPS Technologies" on page 34.

Symbols Used

The **Notes** in this manual follow standard ANSI recommendations.

NOTE: This format is used to emphasize important factors.

User Comments

We welcome your comments about this document. You can reach us by e-mail at info@cavium.com.

About This Book

The data and illustrations found in this book are not binding. We reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be construed as a commitment by Cavium, Inc.

Cavium, Inc. assumes no responsibility for any errors that may appear in this document. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us at info@cavium.com.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of the Publisher, Cavium, Inc.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIPS Technologies

MIPS architecture is the exclusive property of MIPS Technologies, Inc. and is protected under the following trademarks: MIPS64.

MIPS64 is among the registered trademarks of MIPS Technologies, Inc. in the United States and other countries.

Various MIPS RISC processor manuals and additional information about MIPS products can be found at the MIPS URL:

http://www.mips.com

Comments or questions on the MIPS64® Architecture or MIPS documents should be directed to:

Director of MIPS Architecture
MIPS Technologies, Inc.
955 East Arques Avenue
Sunnyvale, CA 94085
or via E-mail to architecture@mips.com.

Contact Us

Cavium
2315 North First Street
San Jose, CA 95131
Telephone: 408-943-7100
Fax: 408-577-1992
Email: info@cavium.com

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 1

Introduction

OCTEON III CN71XX Multicore MIPS64 Processor Family

Features

- Highly-integrated networking security/application processor
- MIPS64® integer instruction set, highly programmable
- High-performance architecture with up to four cnMIPS™ III processor cores
- Dual control- and data-plane support
- Standalone (i.e. self-bootable) or device support
- Core frequency up to 1.6 GHz
- Excellent performance/watt
- Hardware cryptographic (MD5, SHA-1, SHA-2, SHA-3 SHA-256, SHA-512, SHA-512, DES/3DES, AES, KASUMI, SNOW 3G, SMS4, ZUC, Camellia, modular exponentiation, and Galois field multiplication) and CRC acceleration
- Hardware packet-processing acceleration
- Hardware work queueing, scheduling, ordering, and synchronization
- Hardware TCP acceleration, including checksum and timer
- High-bandwidth on-chip memory system including a 512KB four-way set-associative L2 cache
- Fully coherent memory system
- A 36-bit DDR3/DDR4 DRAM interface, up to 1333 MHz data rate
- Support for DDR3L 1.35-mode operation
- Deep-packet-inspection acceleration with third-generation hardware regular expression acceleration HFA engines
- Two SATA 1.5/3.0/6.0 Gb/s ports
- Two integrated USB 3.0/2.0 MAC/PHYs
- One SPI interface
- Wake-on-LAN support
- Up to two Ethernet interfaces, each configurable as SGMII, QSGMII (four time-interleaved ports per lane), or RXAUI
- Up to three controllers supporting PCI Express® (PCIe®) 2.0 interfaces with ×1/×2/×4 lanes
- One RGMII packet interface
- Secure on-chip key memory
- Cryptographic random-number generator
- Support for NOR flash boot
- Support for eMMC/SD flash interfaces
- Support for NAND flash interfaces with ECC
- Hardware RAID acceleration

Cavium Confidential For Bowei Yang ALU 02/27/2014

Overview

The Cavium OCTEON III CN71XX Multicore cnMIPS64 processors are a family of processors targeted at intelligent networking, wireless, control-plane, and storage applications. The CN71XX is targeted for many applications, but is particularly well-suited for the following applications and standards:

- Wireline and Wireless Broadband Gateways
- Triple Play Gateways
- 802.11 N/AC Access Points
- In-line packet processing
- Firewall (FW)
- Virtual Private Network (VPN)/IPsec/SSL
- Intrusion Detection / Prevention (IDS)
- Security Appliances
- Secure Network Interface Card (NIC)
- Secure Services Card
- SMB iSCSI Appliances
- Storage Shelf Controller
- SSD Controllers
- Unified Threat Management Appliances
- Control Plane Line Card
- Industrial/Military/Aero
- Home Network Attached Storage (NAS) and SMB NAS Appliances

The OCTEON III family has a base architecture that is the same for all members. Software written for the OCTEON III family can scale from top to bottom. Existing software written for the OCTEON II CN6XXX family runs on the OCTEON III family.

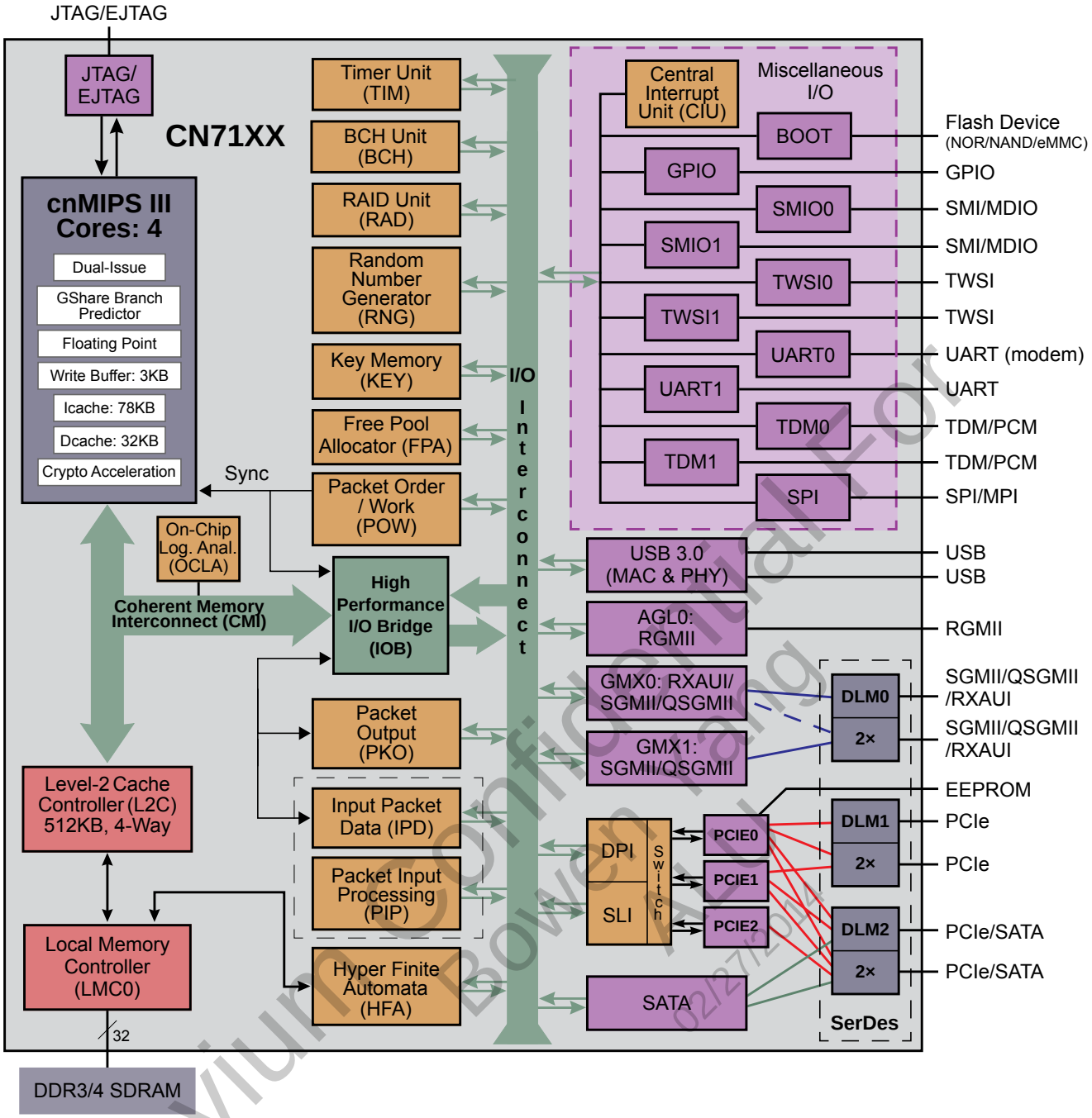
All OCTEON III family members use the same cnMIPS III CPU core and the same on-chip architecture. The cnMIPS III core, hereafter referred to as the core, is an enhanced high-performance dual-issue implementation of the standard MIPS64® integer instruction set with additional networking and wireless acceleration instructions. CN71XX hardware preserves the same software interface to all on-chip hardware units between all family members.

Figure 1–1 shows the block diagram of the CN71XX chip. The blocks on the left side of the chip (i.e. the CPU cores, the coherent memory interconnect (CMI), the level-2 cache (L2C), and the DRAM controller (LMC)) implement an on-chip multiprocessor and coherent shared-memory system. The right side of the chip contains the I/O interconnect (IOI) and I/O interface blocks together with configurable I/O and coprocessing hardware accelerator blocks. These two sides of the chip connect through the high-performance I/O bridge (IOB), creating a system-on-a-chip suited for packet-processing, networking, storage, and security applications.

The family is a scalable architecture that covers a wide range of performance and I/O configurations. CN71XX family members support the following industry-standard I/O interfaces: DDR3/DDR4 DRAM, PCI Express 2.0, SGMII, QSGMII, RXAUI, SATA, UART, JTAG (IEEE 1149.1), and AC-JTAG (IEEE 1149.6). CN71XX also implements a boot-bus interface that can interface with NOR or eMMC flash devices.

Cavium Confidential For Bowen Yang 05/27/2014

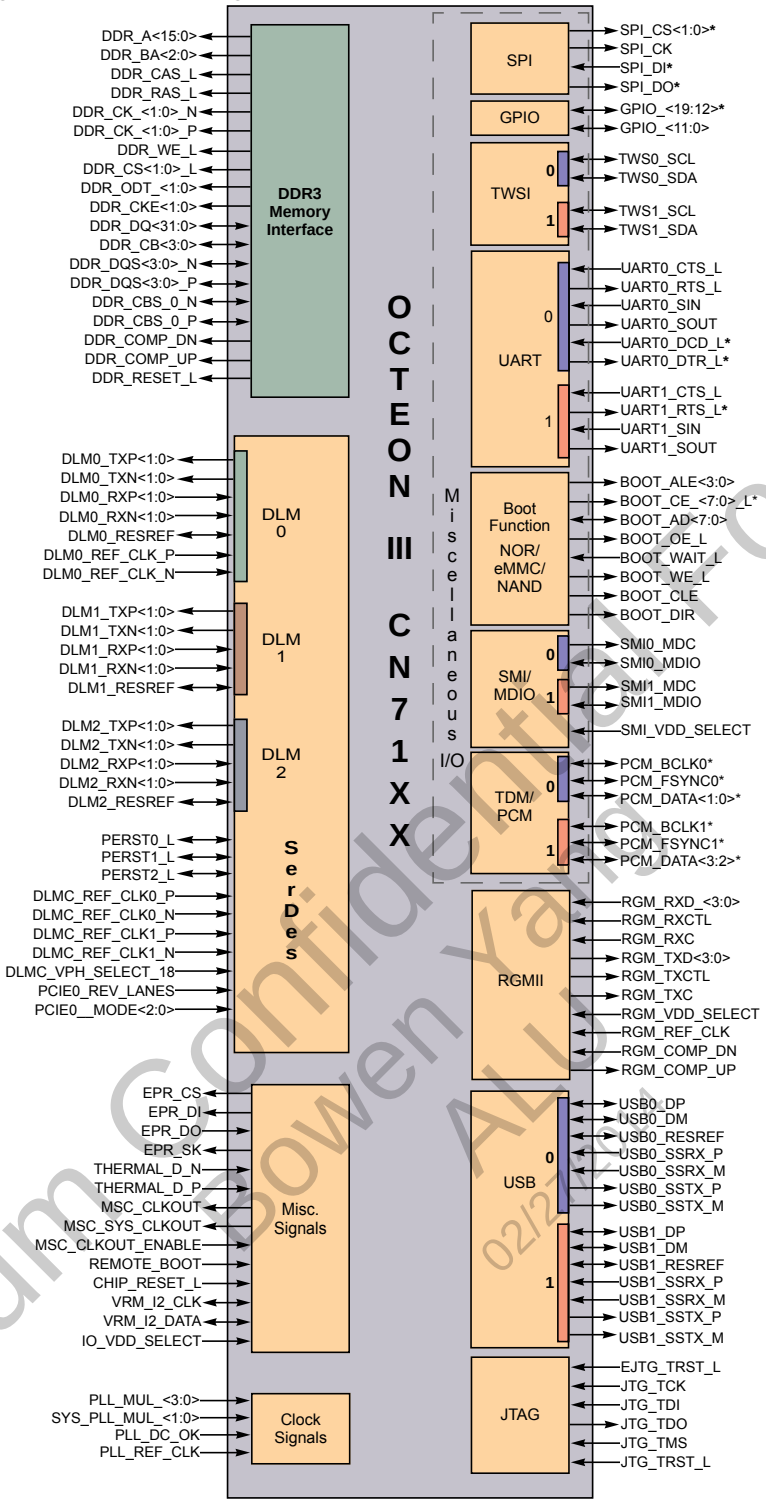
Figure 1–1 CN71XX Block Diagram



Cavium Confidential For Bowen Tang AEU 02/27/2014

Figure 1–2 shows the full CN71XX pinout.

Figure 1–2 CN71XX Signals



* Indicates that signals are multiplexed.

Cavium Confidential For Bowen Yang ALL 02/2

1.1 CN71XX Clocks

This section describes the CN71XX clocking structure, which is based on three clock signals: the core clock, the coprocessor clock, and the DDR clock. The standard clocking scheme is shown in Section 1.1.1, and a special CN71XX single-input clocking scheme is shown in Section 1.1.2

Core Clock The core PLL generates the core clock (RCLK), which is used by the cnMIPS III cores, the CMI, L2C, and OCLA. The supported frequencies of the core clock are listed in the description of the PLL_MUL_<3:0> signals (Table 40–13). Software can determine the core-clock speed by reading the value of RST_BOOT[C_MUL] and placing that value in the following formula:

Core-Clock Frequency = PLL_REF_CLK (= 50 MHz) × [C_MUL]

Coprocessor Clock The coprocessor PLL generates the coprocessor clock (SCLK), which is used by the rest of the coprocessors and the non-SerDes I/O interfaces. The supported frequencies of the coprocessor clock are listed in the description of the SYS_PLL_MUL_<1:0> signals (Table 40–13). Software can determine the coprocessor clock speed by reading the value of RST_BOOT[PNR_MUL] and placing that value in the following formula:.

Coprocessor-Clock Frequency = PLL_REF_CLK (= 50 MHz) × [PNR_MUL]

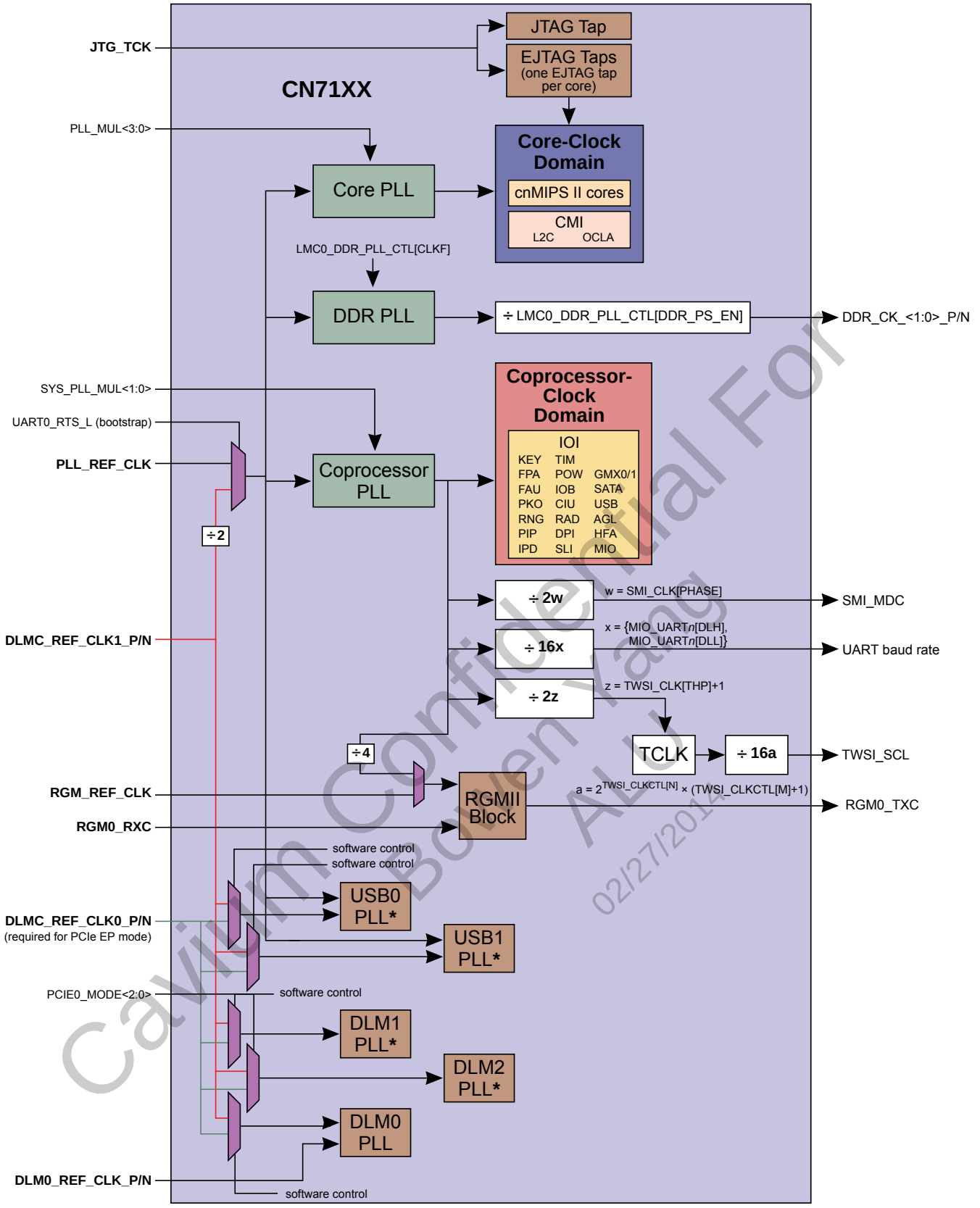
DDR Clock The DDR PLL generates a clock signal for the LMC, with its frequency based on its post-scalar divide (see LMC(0)_DDR_PLL_CTL).

1.1.1 Standard CN71XX Clocking Scheme

CN71XX has a single reference-clock signal (PLL_REF_CLK) that feeds into the three PLLs (core PLL, coprocessor PLL, and DDR PLL) that provide the clock signals for the rest of the chip. Each PLL has its own set of PLL_REF_CLK multipliers, allowing each to generate its own frequency. Figure 1–3 shows the CN71XX clock scheme.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 1–3 CN71XX Detailed Clocking Diagram

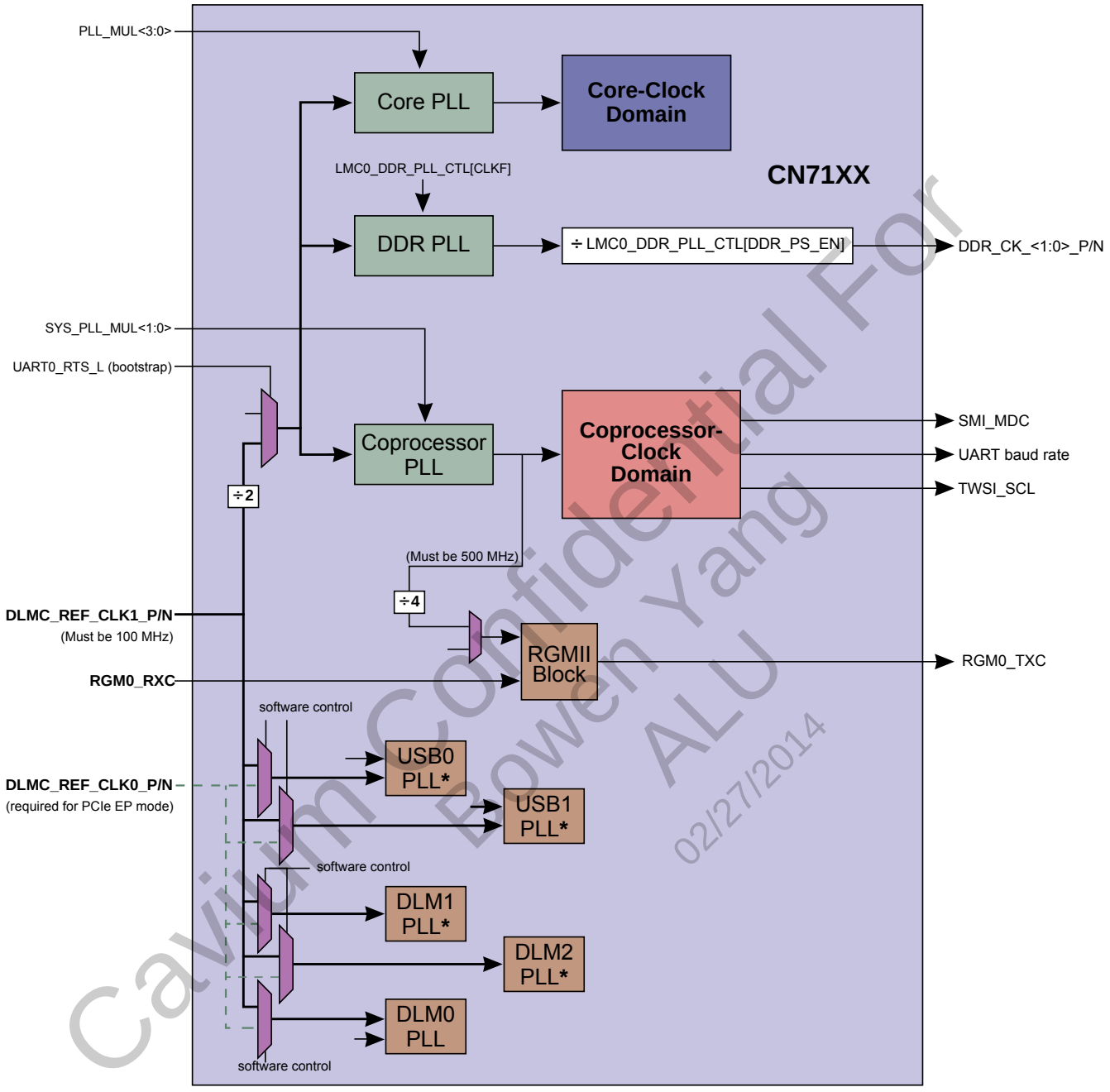


Cavium Confidential For Bowen Yang ALU 02/27/2014

1.1.2 Special CN71XX Single-Input Clocking Feature

CN71XX has a special clocking feature that allows a single reference-clock signal to be the clock source for all internal clock signals. The clock signal is DLMC_REF_CLK1_P/N and in this case it must be set for 100 MHz. An example of how this feature can be used is shown in Figure 1–4.

Figure 1–4 CN71XX Single-Input Clocking Example



This configuration supports all modes of operation except RXAUI and SGMII @ 3.125 Gb/s.

(DLMC_REF_CLK0_P/N is required for PCIe EP mode)

Table 1–1 shows the required reference-clock frequencies for all the SerDes interfaces.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 1–1 SerDes Clocking Requirements

SerDes Standard	Ref. Clock Frequencies	Spread Spectrum	REF_CLK Signals[1]	Clock Selection
SGMII 1.25 Gb/s	50 MHz 100 MHz 125 MHz	non-SSC	DLMC_REF_CLK0 DLMC_REF_CLK1 DLM0_REF_CLK	When GSER(0)_DLM(0)_REF_USE_PAD[REF_USE_PAD] = 1, use DLM0_REF_CLK. When GSER(0)_DLM(0)_REF_USE_PAD[REF_USE_PAD] = 0, use DLMC_REF_CLK0 or DLMC_REF_CLK1. Then, the reference-clock source is specified by GSER(0)_DLM(0)_REFCLK_SEL[REFCLK_SEL]: 0 = use DLMC_REF_CLK0 1 = use DLMC_REF_CLK1
SGMII 2.50 Gb/s	50 MHz 100 MHz 125 MHz			
SGMII 3.125 Gb/s	125 MHz 156.25 MHz			
QSGMII 5.0 Gb/s	50 MHz 100 MHz 125 MHz			
RXAUI 6.250 Gb/s	125 MHz 156.25 MHz			
PCIe, Gen 1/2	100 MHz	SSC[2] or non-SSC	DLMC_REF_CLK0 DLMC_REF_CLK1	If the system boots in EP mode, PEM0 uses DLMC_REF_CLK0. When in RC mode, the reference-clock source is specified by GSER(0)_DLM(1..2)_REFCLK_SEL[REFCLK_SEL]: 0 = use DLMC_REF_CLK0 1 = use DLMC_REF_CLK1
SATA	100 MHz	SSC or non-SSC	DLMC_REF_CLK0 DLMC_REF_CLK1	The reference-clock source is specified by GSER(0)_DLM(2)_REFCLK_SEL[REFCLK_SEL]: 0 = use DLMC_REF_CLK0 1 = use DLMC_REF_CLK1
USB 2.0/HS[3]	50 MHz	non-SSC	PLL_REF_CLK	The reference-clock source is specified by USBDRD(0..1)_UCTL_CTL[REF_CLK_SEL]: 0x0 = the reference-clock source for both PLLs is DLMC_REF_CLK0. 0x1 = the reference-clock source for both PLLs is DLMC_REF_CLK1. 0x2 = the reference-clock source for the SuperSpeed PLL is DLMC_REF_CLK0, the reference clock source for the highspeed PLL is PLL_REF_CLK. 0x3 = reference-clock source for the SuperSpeed PLL is DLMC_REF_CLK1, the reference clock source for the highspeed PLL is PLL_REF_CLK.
	100 MHz	non-SSC	DLMC_REF_CLK0 DLMC_REF_CLK1	
USB 3.0/SS[4]	50 MHz 100 MHz 125 MHz	SSC or non-SSC	DLMC_REF_CLK0 DLMC_REF_CLK1	

1. If DLMC_REF_CLK1 is 100 MHz non-SSC, then it can be used as the source for the system PLL reference clock. This is done by strapping the UART0_RTL_L ball high.
2. Use of an SSC clock for PCIe requires a shared clock for both the RC and EP.
3. PLL_REF_CLK (50 MHz) can only be used to support HS operation. HS operation using DLMC_REF_CLKn requires a 100 MHz non-SSC clock.
4. SS operation requires a compatible clock on DLMC_REF_CLK*n*.

Cavium Confidential For Bowen Yang 2020-04-15 AUG

1.2 Principles of Operation

This section lists some of the principles of the CN71XX architecture.

1.2.1 CPU Cores

The CN71XX's CPU cores are full-functionality and high-performance MIPS64® integer implementations. These cores directly support industry-standard C/C++ (and other) programming environments.

These cores and the CN71XX system-on-a-chip also have all the necessary requirements, including TLBs, to boot and run full- functionality operating systems.

These core features allow for feature-rich usages of the CN71XX, and also are essential to simplify programming tasks. The CN71XX has the high performance necessary for "data-plane" applications, yet avoids the code/data size restrictions present in some competing "data-plane" processors.

CN71XX includes many core instructions beyond the standard MIPS64® integer instructions. These include instructions to accelerate packet processing, security processing, and memory/cache processing.

There is extensive hardware acceleration on CN71XX, but the CPU cores direct the complicated higher-layer application-specific processing.

1.2.2 Coherent Multicore and I/O L2/DRAM Sharing

Shared main memory (implemented via the level-2 cache and the DRAM) is the primary communication media for bulk transfers in the CN71XX. I/O and coprocessor devices DMA packet and other data in/out of this memory.

CN71XX's level-2 cache is shared by all of the CPU cores instructions/data and hardware device DMA accesses. The L2 cache can be partitioned, and can be bypassed on a reference-by-reference basis. I/O devices can DMA data directly into L2 cache.

The CN71XX hardware always maintains L2-cache and CPU-core-data-cache coherence with respect to all DMA and other core accesses.

1.2.3 Core Partitioning

Software and chip configuration can partition the cores to perform different functions. For example, some of the cores may run an operating system while others perform data-plane functions, or different cores may execute different portions of the data-plane services.

1.2.4 Flexible Packet/Control Interfacing

Packets and control information can flow to/from the CN71XX via any of the SGMII/ QSGMII/RXAUI, or PCIe interfaces.

Internally, CN71XX can support three SerDes dual-lane modules (DLMs), for a total of six SerDes lanes.

Cavium Confidential For Bowen Yang ALU 04/27/2014

1.2.5 In-line Packet-Processing Hardware Acceleration

The CN71XX has in-line hardware to offload from the cores all data movement, many common packet-parsing functions, and other important calculations. The in-line hardware completely offloads the work from the cores.

The CN71XX in-line packet-processing hardware units complete the following tasks before a core receives a packet:

- Allocate DRAM buffers to hold the packet bytes.
- Send packet data into these DRAM buffers via DMA operations in a format convenient to upper-level software. This can enable copy-free core software.
- Parse the layer-two and layer-three/IP packet, checking for common exception conditions. This is done for every ingress packet. The layer-four TCP/UDP checksum checks are included, among many others.
- Perform optional mutual exclusion via a programmable 5-tuple classification.

When software has finished processing the packet, CN71XX in-line hardware performs the following tasks:

- DMA the packet-send command from the selected output queue (and free the available queue space).
- DMA packet data from L2/DRAM. This includes multiple modes to gather non-contiguous packet data.
- Generate the TCP/UDP checksum. (CN71XX can perform this very efficiently; the packet data is read out of L2/DRAM memory only once to both calculate the checksum and send the packet off-chip.)
- Free the DRAM buffers.

1.2.6 Hardware-Assisted Dynamic Memory Allocation/Deallocation

CN71XX contains free pools that contain pointers to available packet buffers. The hardware can automatically allocate and free packet buffers. The queues that the cores use to submit instructions to the various on-chip coprocessors are also dynamically allocated in chunks. The coprocessor hardware automatically frees the memory chunks containing the instructions after it processes the instructions contained in the chunk. The cores can also use the free pools at any time. Refer to Chapter 8.

1.2.7 Hardware Work Queuing, Scheduling, Ordering, and Synchronization

CN71XX hardware maintains a work queue in an on-chip hardware unit. This structure provides a primary on-chip communication mechanism between the cores and the hardware units on CN71XX. Both hardware and core software can contribute work to the work queue. The core software can request work at any time, which it becomes aware of either through polling or by interrupts. The hardware schedules the work for the cores.

For example, the in-line input packet processing hardware presents an input packet to the CPU cores by creating and submitting a work queue entry to the queue. The core software receives the packet by obtaining the associated work.

CN71XX has many ordering/synchronization mechanisms available to core software, but one important one is closely integrated with the work queueing/scheduling hardware. This hardware orders/synchronizes based on tag values associated with the packet/work. The inline packet processing hardware can create the initial tag for

Cavium Confidential For Bowen Yang 2/27/2014

a piece of work from up to a 7-tuple hash. Core software can fully direct the subsequent tag changes needed for the packet/work, and is also in control of the initial tag that it creates.

The work-queuing hardware not only synchronizes, but also schedules and deschedules (in conjunction with core software) work based on tag values.

1.2.8 Essential Quality of Service (QoS) Functions Implemented in Hardware

CN71XX has some essential QoS capabilities implemented directly in hardware:

- The in-line input packet processing hardware can classify packets into one of the eight input work queues using default values, DSA/VLAN priorities and IP Diffserv values, configurable on a per-port basis. The system can also directly select the work queue on a per-packet basis.
- The in-line input packet processing hardware may discard an input packet before buffering and presenting it to a core. The hardware implements both a random early discard (RED) algorithm and a threshold algorithm to decide when or if to discard an input packet. The RED-like algorithm can be configured differently for each of the 1024 Aura levels (representing QoS flows), and the threshold algorithm can be configured differently for different ports.
- Each output port can be configured to have multiple queues. The queues can be configured with different priorities. The hardware implements both static priority and weighted round-robin.

Remaining QoS functions are implemented by core software on CN71XX. For example, hardware never decides to drop an outgoing packet – only software does.

1.2.9 Security Features

The cores internally include acceleration for the common security algorithms: MD5/SHA-1/SHA-2/SHA-3/SHA-256/SHA-512, DES/3DES, AES (all modes), AES-GCM, KASUMI, CRC (up to 32 bits), SNOW 3G, ZUC, SMS4, GFM, Camellia, and vector multiplication for fast modular exponentiation needed for RSA/Diffie-Hellman operations.

CN71XX contains a number of features for high FIPS-level certification, including on-chip key memory and restricted PCIe host access.

1.2.10 Coprocessor Accelerators

There are also other available coprocessors that have not already been mentioned:

- Timer (TIM): timer support is particularly useful for TCP implementations.
- RAID (RAD): optional hardware acceleration support for RAID 3/4/5/6
- Key memory
- True Random Number Generator (RNG)
- Hyper/Deterministic Finite Automata (HFA)

Cavium Confidential For Bowen Yang ALU 02/27/2014

1.2.11 Debug Support

CN71XX has numerous debug features, including:

- Two UARTs for connection to an external console.
- Full core support for the MIPS EJTAG standard (including single-step) and an external JTAG interface with one internal TAP controller per core and an overall JTAG controller for boundary scan..
- Multicore debug support.
- Visibility into critical hardware scheduling structures.
- Visibility into hardware/software interface structures

1.3 Remaining Chapters

The remainder of this chapter briefly describes the contents of each remaining chapter in this book.

1.3.1 CPU Cores

This chapter describes the CPU cores that implement the MIPS64® integer instruction set, including internal architecture, the CN71XX-specific enhancements and configuration registers, and the interactions between the cores and the rest of the CN71XX. Refer to Chapter 2.

Section 1: OCTEON III Cache, Memory, and Buses

This section contains chapters that describe the core elements of the CN71XX architecture.

1.3.2 Coherent Memory Interconnect and L2 Cache Controller

This chapter describes the components that implement the CN71XX memory system. This includes a description of the coherent memory interconnect (CMI) and the internal architecture and configuration registers of the level-2 cache and controller (L2C). Refer to Chapter 3.

1.3.3 DRAM Controller

This chapter describes the internal architecture and configuration registers of the DRAM controller, and the DDR3/DDR4 pin interface. Refer to Chapter 4.

1.3.4 I/O Interconnect and High-Performance I/O Bridge

This chapter describes the components that implement the I/O Interconnect (IOI) that connects CN71XX I/O devices and coprocessors to the on-chip main memory system. This includes a description of the IOI and the internal architecture and configuration registers of the high-performance I/O bridge (IOB). Refer to Chapter 5.

1.3.5 On-Chip Logic Analyzer Unit

This chapter describes the OCLA unit. Refer to Chapter 6.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Section 2: OCTEON III Coprocessors

This section contains chapters that describe the various coprocessors that make up the CN71XX.

1.3.6 Packet Order/Work Unit

This chapter describes the work-queuing, scheduling, ordering, and synchronization hardware on the CN71XX. This includes the software interface, the internal architecture, and the configuration registers of the packet order/work unit (POW). Refer to Chapter 7.

1.3.7 Free Pool Unit

This chapter describes the CN71XX hardware free-pool implementations. This includes the software interface, the internal architecture, and the configuration registers of the free pool unit (FPA). Refer to Chapter 8.

1.3.8 Packet Input Processing/Input Packet Data Unit

This chapter describes the hardware-centralized input-packet parsing and DMA capabilities. This includes the software interface, the internal architecture, and the configuration registers. These units write input packets from any/all of the RGMII, SGMII, QSGMII, RXAUI, and PCIe interfaces into CN71XX's main memory, and also create/submit work-queue entries for the input packet. Refer to Chapter 9.

1.3.9 Packet Output Unit

This chapter describes the hardware-centralized output packet processing and DMA capabilities. This includes the software interface, the internal architecture, and the configuration registers of the packet output unit (PKO). This unit DMAs packets out from main memory through any of the RGMII, SGMII, QSGMII, RXAUI, and PCIe interfaces. Refer to Chapter 10.

1.3.10 Hyper Finite Automata Unit

This chapter describes the software interface, internal architecture, and configuration registers for the HFA unit. Refer to Chapter 11.

1.3.11 DMA Packet Interface/Switch Logic Interface (DPI/SLI)

Refer to Chapter 12.

1.3.12 Central Interrupt Unit (CIU)

This chapter describes the central interrupt unit (CIU). This includes the software interface and the configuration registers. This unit is a central distributor for core interrupts, external PCIe interrupts, and other controls. Refer to Chapter 13.

1.3.13 Reset Unit (RST)

This chapter describes the reset status, control, and configuration logic. Refer to Chapter 14.

Cavium Confidential For Bowen Yang ALU 9/27/2014

1.3.14 Precision Time Protocol (PTP) Clocks and Timestamping

This chapter describes timestamping in the PTP, GMX, AGL, PIP/IPD, PKO blocks, and MIX blocks. This timestamping is suitable for use in IEEE 1588 Precision Time Protocol (PTP) or other purposes. Refer to Chapter 15.

1.3.15 Timer Unit

This chapter describes the timer unit, which is a hardware mechanism to submit work-queue entries at future times. This includes the software interface, internal architecture, and configuration registers. Refer to Chapter 16.

1.3.16 Random Number Generator

This chapter describes the random number generator (RNG), which includes the internal architecture, the software interface, and the configuration registers. The random number generator creates truly random numbers. Refer to Chapter 17.

1.3.17 RAID/De-Dup Unit

This chapter describes the RAID unit (RAD), its software interface, and the configuration registers. RAD can be used to accelerate de-duplication by performing fast data-block compares. Refer to Chapter 18.

1.3.18 BCH Unit

This chapter describes the BCH unit (BCH, Bose Chaudhuri Hocquenghem cyclic error-correcting code). BCH accelerates both the parity-generation and the error-correction functions. Refer to Chapter 19.

1.3.19 Key Memory

This chapter describes the key memory unit (KEY), including the software interface and control registers. The key memory is a secure on-chip store that can be used for holding critical keys. Refer to Chapter 20.

Section 3: OCTEON III I/O Interfaces

This section contains chapters that describe the various I/O interfaces that are part of the CN71XX.

1.3.20 General Serializer/Deserializer (GSER) Interface

This chapter describes the GSER block, which controls the SerDes dual-lane modules (DLMs). Refer to Chapter 21.

1.3.21 PCI Express (PCIe) Interface (PEM)

This chapter describes the PCIe 3.0 ports and the hardware that creates and services PCIe transactions. This includes a description of all the PCIe transactions that CN71XX can create and service, the PCIe BARs, the software interface, the internal architecture, and the configuration registers. The internal architecture includes ports that interface with switch logic and the centralized packet-input and packet-output hardware, DMA engines, direct-access mechanisms from externally-mastered PCIe transactions, and core direct-access to locally-mastered PCIe transactions. Refer to Chapter 22.

Cavium Confidential For Bowen Yang ALU 02/27/2014

1.3.22 SGMII/QSGMII Interface

This chapter describes the serial Gigabit media-independent interface (SGMII), and the internal architecture and configuration registers of the hardware SGMII unit. This unit primarily interfaces with the centralized packet input and output logic internally. Refer to Chapter 23.

1.3.23 RXAUI Interface

This chapter describes the reduced 10-Gigabit attachment unit (RXAUI) interface, and the internal architecture and configuration registers of the hardware RXAUI unit. This unit primarily interfaces with the centralized packet input and output logic internally. Refer to Chapter 24.

1.3.24 RGMII Interface

This chapter describes the RGMII interface, and the internal architecture and configuration registers of the hardware RGMII unit. Refer to Chapter 25.

1.3.25 SATA Interface

Refer to Chapter 26.

1.3.26 TDM/PCM Interface

Refer to Chapter 27.

1.3.27 Boot-Bus Interface

This chapter describes the external-boot interface, its software interface, and the configuration registers. When CN71XX self-boots, the boot interface has the (nonvolatile, presumably) storage, like flash memories or ROMs, that contain the initial CN71XX boot code. The boot bus can also support other more sophisticated devices such as those that follow the compact flash standard.. Refer to Chapter 28.

1.3.28 eMMC Interface

This chapter describes the eMMC interface unit. Refer to Chapter 29.

1.3.29 NAND Flash Interface

This chapter describes the external-boot interface when it interfaces with NAND flash devices. It includes the NAND flash interface, NAND flash commands, and configuration registers of the hardware NAND flash interface unit. Refer to Chapter 30.

1.3.30 USB Interface

This chapter describes the universal serial bus (USB) interfaces, which support host functions and are fully compliant with the USB 3.0 specification. Refer to Chapter 31.

Cavium Confidential For Bowen Yang ALI 02/27/2014

1.3.31 System Management Interface (SMI)

The CN71XX system management interface is a standard ethernet MDIO interface. The chapter describes the interfaces, their software interfaces, and the configuration registers. Refer to Chapter 32.

1.3.32 GPIO Interface

This chapter describes the external general-purpose I/O (GPIO) interface, its software interface, and the configuration registers. Refer to Chapter 33.

1.3.33 UART Interface

This chapter describes the external UART interfaces, their software interfaces, and the configuration registers. Refer to Chapter 34.

1.3.34 TWSI Interface

This chapter describes the external TWSI interfaces, their software interfaces, and the configuration registers. Refer to Chapter 35.

1.3.35 Serial Peripheral Interface (SPI)

This chapter describes the SPI interface. Refer to Chapter 36.

Section 4: OCTEON III Electrical/Mechanical/Physical

This section contains chapters that describe the electrical, mechanical, and physical aspects of the CN71XX.

1.3.36 Electrical Specifications

This chapter describes the supply voltages, power sequencing, power consumption, and DC electrical characteristics of CN71XX. Refer to Chapter 37.

1.3.37 AC Characteristics

This chapter describes the AC characteristics of the CN71XX I/O interfaces. Refer to Chapter 38.

1.3.38 Mechanical Specifications

This chapter has an overview of the CN71XX physical package (LBGA), and thermal requirements. Refer to Chapter 39.

1.3.39 Signal Descriptions

This chapter describes the CN71XX signals in detail. Refer to Chapter 40.

1.3.40 Ball Assignments

This chapter describes the position of the I/O pins in the CN71XX physical package. Refer to Chapter 41.

Cavium Confidential For Bowen Yang ALU 02/27/2014

1.3.41 Appendix A

This appendix describes in detail the Cavium-specific instructions in the cnMIPS III cores. Refer to Appendix A.

1.3.42 Appendix B

This appendix provides the formatting of the CN71XX part numbers. Refer to Appendix B.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

1.4 Configuration and Status Registers (CSRs)

CN71XX has many CSRs to control and configure the on-chip hardware. Each configuration register can be accessed by one or more of the following mechanisms:

- cores using MFC0/MTC0 instructions
- cores using MFC2/MTC2 instructions
- cores using ordinary load/store instructions to I/O physical addresses
- cores using indirect windowed accesses to PCIe configuration registers via the PEM*_CFG_WR and PEM*_CFG_RD CSRs.
- remote EJTAG/JTAG device using the EJTAG/JTAG TAP interface
- remote PCIe host using direct PCIe configuration accesses maintenance accesses
- remote host using direct PCIe memory space BAR0 accesses
- remote PCIe host using indirect windowed accesses to I/O physical addresses via the SLI_WIN_RD_ADDR, SLI_WIN_RD_DATA, SLI_WIN_WR_ADDR, SLI_WIN_WR_DATA, and SLI_WIN_WR_MASK CSRs.

These configuration registers fall into the following classes:

- **Core coprocessor 0 (CP0) registers**. These CSRs are replicated on each core, and can be accessed by software running on the local core. The CP0 CSRs can also be accessed via L2C_COP0_ADR and L2C_COP0_DAT (see Section 3.4). These are covered in Section 2.16.
- **Core coprocessor 2 (COP2) registers**. These CSRs are replicated on each core, and can only be accessed by software running on the local core. These are covered in Section 2.2 and Appendix A.
- **Core EJTAG registers**. These CSRs are replicated on each core, and can only be accessed by software running on the local core. These are covered in Section 2.17.
  - All CN71XX core EJTAG registers are 64-bits, and must be accessed with 64-bit load/store operations.
- **Core EJTAG TAP registers**. These CSRs are replicated on each core, and can only be accessed by an external EJTAG device via the CN71XX's EJTAG interface. These are covered in Section 2.18.
- **NCB registers**. These registers are accessed directly off the IOI. These registers can be accessed by any core and by remote PCIe devices/hosts through a windowing mechanism via BAR0 read/write operations.
  - Each NCB register has a CN71XX-internal address.
  - All CN71XX NCB registers are 64 or 32 bits, and must be accessed with 64-bit or 32-bit load/store operations.
- **RSL registers**. These registers are accessed indirectly off the IOI. They are similar to NCB registers, except that access to them is much slower as it is indirect access. The hardware can only service an RSL register access about once every 30–40 cycles. These registers can be accessed by any core and by remote PCIe devices/hosts.
  - Each RSL register has a CN71XX-internal address.
  - All CN71XX RSL registers are 64-bits, and must be written with 64-bit store operations, but can be read with any size load.
- **PCICONFIGEP0/1 and PCICONFIGRC0/1 registers**.
  - These registers can be accessed via the PEM0/1_CFG_WR and PEM0/1_CFG_RD CSRs. The ADDR field of these registers matches the PCIe config space address.

Cavium Confidential For Bowen Yang 02/27/14

- All PCIe config registers are 32-bits.

- **PEXP registers**. These windowing registers can only be accessed directly from the PCIe bus, and cannot be accessed by the local cores. Remote PCIe devices/ hosts can access these registers with a read or write operation to CN71XX's BAR0.
- **PEXP_NCB registers**. These registers can be accessed directly from the PCIe bus and from the IOI.
  - Each PEXP_NCB register has both a PCIe bus address and a CN71XX-internal address.
    - Cores can access the PEXP_NCB registers via the CN71XX-internal address.
    - Remote PCIe devices/hosts can access PEXP_NCB registers with a 32-bit or 64-bit PCIe memory space read or write operation to CN71XX's BAR0.
  - Cores can access PEXP_NCB registers via either 64-bit or 32-bit load/store operations.
- **JTAG TAP registers**. These registers can only be accessed by an external device via CN71XX's JTAG interface.
- **TWSI core registers**. These registers control the TWSI core and can only be accessed indirectly via MIO_TWS_SW_TWSI (MIO_TWS_SW_TWSI is an ordinary RSL CSR). TWSI core registers are covered in Chapter 35.

Cavium Confidential For Bowen Yang ALU 02/27/2014

The detailed descriptions of these CSRs are distributed throughout the remaining chapters of this document. Table 1–2 lists the CSRs types that are present in the different chapters.

Table 1–2 CN71XX CSR Types

Chapter	Units	CSR Type
3	Level-Two Cache Controller (L2C)	RSL
4	DRAM Controller	RSL
5	I/O Interconnect, I/O Bridge	RSL
6	On-Chip Logic Analyzer	RSL
7	Packet Order/Work Unit	NCB
8	Free Pool Unit	RSL
9	Packet Input Processing	RSL, NCB
10	Packet Output Processing	RSL
11	Hyper Finite Automata Unit (HFA)	RSL, NCB
12	DPI/SLI	PEXP, PEXP_NCB, NCB
13	Central Interrupt Unit	NCB
14	Reset Unit	NCB
15	Precision Time Protocol (PTP) Clocks and Timestamping	NCB
16	Timer	RSL
17	Random Number Generator, Random Number Memory	RSL
18	RAID	RSL
19	BCH	RSL
20	Key Memory	RSL
21	GSER	RSL
22	PCI Express Interface	PCICONFIGEP, PCICONFIGRC, RSL
23	Serial Gigabit Media Independent Interface (SGMII)	RSL
24	Reduced 10/20 Gigabit Ethernet Attachment Unit Interface (RXAUI)	RSL
25	Reduced Gigabit Media Independent Interface (RGMII)	RSL
26	Serial AT Attachment Unit	RSL, NCB
27	TDM/PCM Interface	NCB
28	Boot Bus	RSL
29	eMMC Interface Unit	RSL
30	NAND Flash Unit	NCB
31	USB	RSL, NCB
32	System Management Interface	RSL
33	GPIO	NCB
34	UART	RSL
35	TWSI Interface	RSL
36	SPI Interface	NCB

NOTE: The listed CSR addresses are all little-endian format. Because most CN71XX CSRs are 64-bit registers, and all 64-bit addresses are endian-neutral, most addresses in this book are endian-neutral.

Cavium Confidential For ALD Bowen Yang 02/27/2014

1.4.1 CSR Field Types

CN71XX has a variety of CSR field types as described below.

- **R/W** – Fields that can be read or written.
- **RO** – Fields that can only be read. Write operations to RO fields are ignored.
- **WO** – Fields that can only be written. Read operations to WO fields return unpredictable data.
- **WR0** – Fields that can be written, but when read, return all 0s.
- **RAZ** – Reserved fields.
- **R/W1C** – Fields that can be read or written with a 1 to clear.
- **R/W1S** – Fields that can be read or written with a 1 to set.
- **RO/WRSL** – Fields that can only be read if accessed from the physical PCIe port, but can be read and written if accessed via PEM*_CFG_WR/ PEM*_CFG_RD.
- **WORSL** – Fields that can only be written and are accessible via the PEM*_CFG_WR registers.
- **RC** – Fields that are cleared after being read. Write operations to RC fields are ignored.
- **RC/W** – Fields that are cleared after being read but can be written.
- **RC/WC** – Fields that are cleared when either read or written.

The above field types, except for WR0, WORSL, and RAZ, can also appear with **/H**, which indicates that the hardware can modify the field directly.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 2

cnMIPS III Cores

This chapter contains the following subjects:

- Overview
- Summary of Features
- Core Non-Privileged State
- Cavium-Specific Instruction Summary
- Instruction Set Summary
- Virtual Addresses and CVMSEG
- Physical Addresses
- IOBDMA Operations
- Core-Memory Reference Ordering
- CSR Ordering
- Core Write Buffer
- Cryptography
- Coprocessor 0 Privileged Registers
- Coprocessor 1 (Floating Point) Control Registers
- EJTAG DRSEG Registers
- EJTAG TAP Registers
- Pipelines
- Special MUL Topics
- COP2 Latencies
- Hardware Debug Features
- Load-Linked/Store-Conditional and Wide Atomic Memory Write Operations
- Exceptions

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium, Inc.'s OCTEON III CN71XX architecture is based on officially licensed MIPS Inc. Technology. For reference to MIPS® Technology found in CN71XX, you must have the following books:

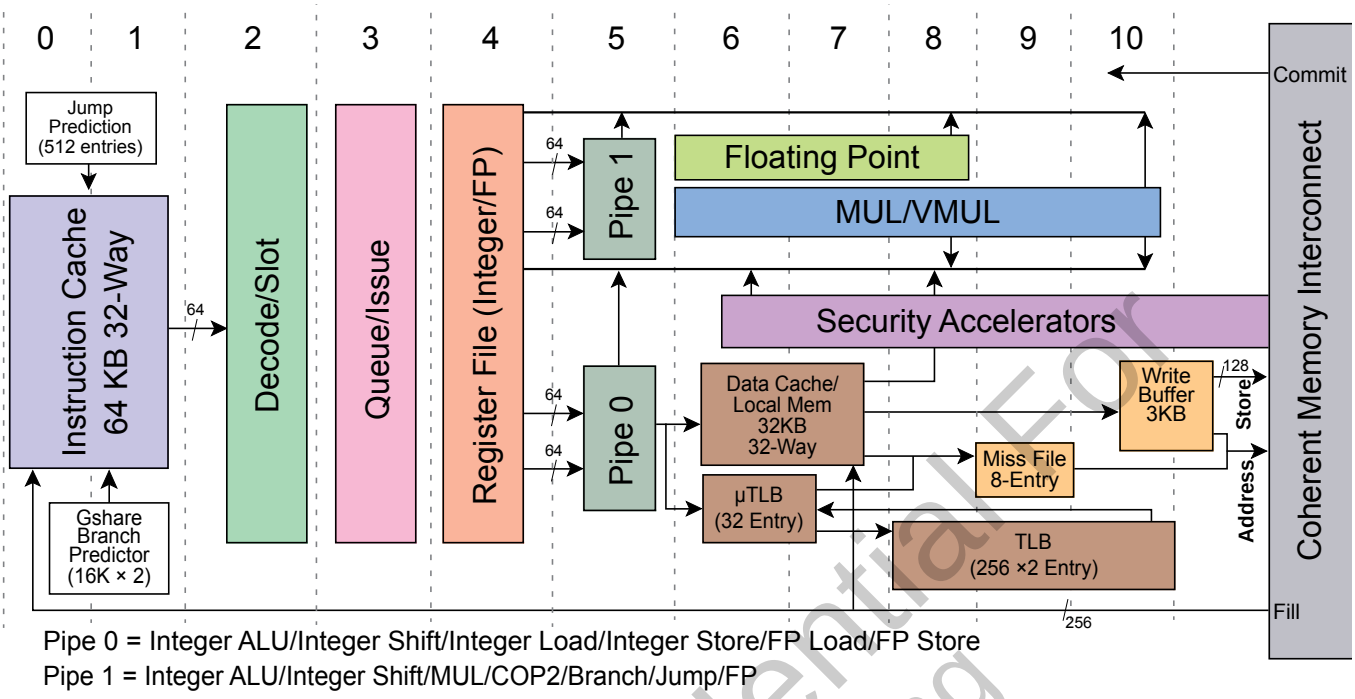
Table 2–1 MIPS Publications

Publication	Doc Number
EJTAG Specification, Revision 6.10, February 17, 2013	MD00047
MIPS64® Architecture For Programmers Volume I: Introduction to the MIPS64® Architecture, Revision 5.01, December 15, 2012	MD00083
MIPS64® Architecture For Programmers Volume II: The MIPS64® Instruction Set, Revision 5.01, December 15, 2012	MD00087
MIPS64® Architecture For Programmers Volume III: The MIPS64® Privileged Resource Architecture, Revision 5.01, December 16, 2012	MD00091
MIPS64® Architecture For Programmers Volume IV-i: Virtualization Module of the MIPS64® Architecture, Revision 1.02, February 19, 2013	MD00847

Overview

The CN71XX's cnMIPS® III cores, shown in Figure 2–1, support MIPS64® version 5 integer instruction set and privileged architecture.

Figure 2–1 cnMIPS™ III Core Block Diagram



cnMIPS III core pipelines are capable of a total of TBD GOPS peak (4 cores @ 1.6 GHz). Instruction fetch/decode operates at two instructions fetched per cycle. Up to two instructions can be issued per cycle, and one branch/jump prediction per cycle. The cnMIPS III cores introduces a 16K × 2 entry Gshare branch predictor. The cnMIPS III cores also enhance the jump predictor to 512 entries and the subroutine return call stack to 12 entries. The cnMIPS III cores also support hardware instruction prefetching and queuing. Integer loads and stores issue on Pipe 0.

The cnMIPS III cores support an IEEE 754-compliant floating point (FP) unit capable of performing up to TBD GFLOPS peak (4 cores @ 1.6 GHz). FP instructions, except for FP loads and stores, are all issued on Pipe 1. Integer instructions can dual issue with FP instructions. Like integer loads and stores, FP loads and stores issue on Pipe 0.

The cnMIPS III core Load/Store units can do a total of TBD billion load/store operations/second (4 cores @ 1.6 GHz). Address translation is supported by a two-level TLB (Translation Lookaside Buffer) with a software-invisible 32-entry micro-TLB (µTLB) and 256×2 entry software-visible architected TLB. To support loads and stores from multiple cores, cnMIPS III architecture has a completely redesigned, highly-scalable, high-performance coherent memory interconnect interface supporting up to TBD Gb/s of data transfer.

Cavium Confidential For Bowen Yang ALB 02/27/2014

2.1 Summary of Features

2.1.1 MIPS64 Version 3.0 Implementation

- Dual-issue, 8+ stage pipeline
- Up to 1.6 GHz in 28 nm process
- Single- and double-precision floating point unit
- Big and little endian support
- Aggressive conditional clocking for minimal power dissipation
- 78KB, 39-way virtual instruction cache (Icache)
  - Bitmask LRU or random replacement
  - Hardware single-bit-error correction
- Hardware instruction prefetching
- 16K entry × 2-bit Gshare branch prediction
- 12-entry subroutine-call return stack
- 512-entry jump prediction
- 32KB, 32-way data cache (Dcache)
  - Three-cycle load-to-use latency
  - Fully coherent with all multicore and I/O DRAM references
  - Duplicated physical tag store for external snoops (cnMIPS III only)
  - Write-through, Invalidate protocol
  - Bitmask LRU replacement
  - Hardware single-bit-error correction
- 32-entry micro-TLB (μTLB) to cache translation from the MTLB
- High-performance coherent memory interconnect interface
- Up to eight outstanding fills
- 3KB merging write-buffer
  - Aggressive merging for highest performance
  - Allows many outstanding stores
  - Marked differentiation for looser ordering via SYNCS or SYNCWS
  - Flushed by a SYNC or SYNCW
  - Partially flushed by a SYNCS or SYNCWS
  - High-performance write synchronization
  - Programmable overflow threshold
  - Programmable merge closure time-out

2.1.2 Virtualization Module Implementation

- Separate Root and Guest CP0 and TLB state
- Onion model with two-level TLB address translation

2.1.3 Cavium-Specific Architectural Additions

- Cavium-specific instruction additions
  - Bit-test branches
  - Bit-field signed extracts
  - Bit-field clear and insert
  - Unsigned byte add

Cavium Confidential For Bowen Yang ALU 02/2/2014

- Set equals, set not-equals
- 32-bit and 64-bit population count
- RDHWR additions
  - 64-bit cycle counter
  - fast SSO tag switch access
- True unaligned loads/stores
- New SYNC instructions
- PREF implementation
  - Many PREFs with different L1/L2 cache behavior
  - Prefetch of TLB entry into the μTLB (cnMIPS III only)
  - Prefetch Indexed (PREFX) (cnMIPS III only, COP1 must be enabled)
- Cache implementation
- Integer indexed loads
- Many memory atomic operations
- Zero cache block
- Q15 Multiply and Accumulate

- Cavium-specific in-core coprocessor 2 security accelerators
  - CRC (any polynomial 32-bit or less)
  - Hash (MD5/SHA-1/SHA-3/SHA-256/SHA-512)
  - 3DES
  - AES
  - KASUMI
  - Galois Field Multiplier
  - SNOW 3G
  - SMS4
  - Camellia
  - ZUC
- Cavium-specific integer multiply unit
  - 64-bit register-direct integer multiply/add
  - 128-bit and 256-bit product multiply/add
  - 192-bit and 448-bit product multiply/add (cnMIPS III only)
- CVMSEG
  - CVMSEG LM = Dcache partition can be a local memory / scratchpad
    - Variable and dynamic size from zero to 54 cache blocks (up to 6912B)
  - CVMSEG I/O = IOBDMA operations = I/O operations into CVMSEG LM
  - Separate kernel/debug, supervisor, user access
- XKPHYS access for supervisor/user applications
  - separate enables for memory and I/O spaces
  - separate enables for supervisor and user
- Read and Execute Inhibit TLB support
- CP0 scratch registers
  - Four in addition to DESAVE, UserLocal in the Root; three in addition to UserLocal in the Guest

Cavium Confidential For Bowen Yang ALU 02/27/2014

2.1.4 Full Privileged Architecture (i.e. Coprocessor 0) Support

- 256-entry TLB
  - Up to 512 pages
  - Page sizes from 4KB up to 256MB (1KB not supported)
- Can be partitioned into separate Root and Guest TLBs
- SEGBITS = 49
- PABITS = 49
- Virtual Icache
- Soft-reset
  - Core-selected CIU_PP_RST[RST,RST0]
  - Chip-wide CIU_SOFT_RST[SOFT_RST]
  - Core-local EJTAG TAP soft-reset (ECR[PrRst])
- Non-maskable interrupts
  - Core-selected CIU_NMI[NMI]
  - CIU watchdog timer expiration for the core
- Interrupts
  - Three interrupt input wires per core (Cause[IP4, IP3, IP2]) from CIU
    - CIU combines all interrupt sources separately for each
    - CIU has per-core mailbox registers for inter-core interrupts
  - No external interrupt controller (EIC)
  - Vectored interrupts not implemented
- Supervisor mode implemented
- Four performance counters
- Two watchpoints
  - One instruction-only, one load/store-only
- Hardware-automatic Icache and Dcache repair (by invalidation) on Icache and Dcache errors
  - Separate Icache and Dcache error registers for diagnostic purposes.
- HWREna support for Cavium-specific RDHWR additions
- No GPR shadow sets
- No user-mode endian reversal
- No reduced-power modes
- MDMX ASE not implemented
- SmartMIPS ASE not implemented
- MIPS16e not implemented
- Trace Logic not implemented

Cavium Confidential For Bowen Yang ALU 02/27/2014

2.1.5 Full EJTAG Version 5.04 Support

- Single-step implemented
- EJTAG DSEG implemented
  - DRSEG
  - DMSEG accesses pass through EJTAG TAP controller
- Four Instruction Breakpoints
  - ASID compare
- Four Data Breakpoints
  - Both load and store value compare with byte lanes implemented
  - ASID compare
- Debug interrupts (DINT) implemented
  - Core-selected CIU_DINT[DINT]
  - Debug interrupts from the EJTAG TAP controller
- Multicore Debug Support
  - Immediate multicore stop on EJTAG breakpoint match
  - Dedicated multicore communication path for low-impact and very fast debugging
  - Global (CIU_GSTOP[GSTOP]) and per-core debug mode counter/ watchdog stop conditions
- EJTAG PC Sampling Implemented
- Full EJTAG TAP controller support
  - Special EJTAG chip pin interface
  - One EJTAG TAP controller per core; EJTAG TAPs from different cores link serially onto the EJTAG chip pin interface
  - DMSEG implemented
  - EJTAGBOOT implemented (for memory-less boot)
  - Core reset support (ECR[PrRst])
  - Peripheral reset not supported
- Imprecise debug data breakpoint on load value compare, otherwise all debug data breakpoints are precise.
- No imprecise bus errors
- No imprecise machine-checks
- No imprecise cache errors
- Internal bus clock never stops
- No soft-reset masking
- Data address sampling not implemented
- Fast debug channel not implemented

Cavium Confidential For Bowen Yang ALU 02/27/2014

2.2 Core Non-Privileged State

Table 2–2 details the non-privileged state of the cnMIPS III cores.

Table 2–2 CPU Visible State Resident in Each cnMIPS III Core

Register	Implemented/ Not Implemented	Comments	
32 general-purpose registers	Implemented		
HI and LO registers	Implemented	**NOTE:** The register contents are unpredictable following any chip or core reset.	
PC	Implemented		
32 general-purpose 64-bit FP registers	Implemented	The 32 64-bit FP registers are treated as such when Status[FR] = 1. When Status[FR] = 0 (also known as paired-register mode), these are treated as 32 32-bit registers (only low 32 bits of each register valid) for single-precision operations and 16 64-bit registers for double-precision operations. Only even register numbers are valid references for double-precision operations in paired register mode.	**cnMIPS III** only
Two FP control registers: FIR and FCSR	Implemented	FIR is the 32-bit FP Implementation register, which identifies the capabilities of the cnMIPS III FP unit. FCSR is a 32-bit FP control and status register. Refer to Section 2.16 on page 186 for more information.	**cnMIPS III** only
MPL0, MPL1, MPL2, MPL3, MPL4, MPL5	Cavium specific	64-bit multiplier registers used by Cavium's multiplier additions. Refer to Section 2.20 on page 203 for more information. **NOTE:** Not present when CvmCtl[NOMUL] = 1. **NOTE:** The register contents are unpredictable following any chip or core reset.	MPL3, MPL4, and MPL5 in **cnMIPS III** only
P0, P1, P2, P3, P4, P5	Cavium specific	64-bit product registers used by Cavium's multiplier additions. Refer to Section 2.20 on page 203 for more information. **NOTE:** The register contents are unpredictable following any chip or core reset. **NOTE:** P0, P1, P2, P3, P4, and P5 are made unpredictable by any of the DDIV, DDIVU, DIV, DIVU, MUL, DMUL, DMULT, MULT, MULTU, MADD, MADDU, MSUB, OR MSUBU instructions. You must save P0, P1, P2, P3, P4, and P5 before any of these instructions are executed when a context switch is necessary. **NOTE:** Not present when CvmCtl[NOMUL] = 1.	P3, P4, and P5 in **cnMIPS III** only
CRCIV<31:0>	Cavium specific	IV for CRC32 engine **NOTE:** The register contents are unpredictable following any chip or core reset.	
CRCPOLY<31:0>	Cavium specific	Polynomial for CRC32 engine **NOTE:** The register contents are unpredictable following any chip or core reset.	
CRCLEN<3:0>	Cavium specific	LEN (number of bytes to add) for CRC32 engine **NOTE:** The register contents are unpredictable following any chip or core reset.	
GFMMUL[1:0] <63:0>	Cavium specific	Galois Field Multiplier multiplier **NOTE:** The register contents are unpredictable following any chip or core reset. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.	
GFMRESINP[1:0] <63:0>	Cavium specific	Galois Field Multiplier result/input **NOTE:** The register contents are unpredictable following any chip or core reset. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.	

Cavium Confidential EO
Bowen Yang
02/21/2014

Table 2–2 CPU Visible State Resident in Each cnMIPS III Core (Continued)

Register	Implemented/ Not Implemented	Comments	
GFMPOLY<15:0>	Cavium specific	Galois Field Multiplier polynomial **NOTE:** The register contents are unpredictable following any chip or core reset. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.	
HASHDAT[14:0] <63:0>	Cavium specific	Input data for 512/1024-bit hash. Also, [7:0] are SNOW3G and ZUC LFSR. Also, all are SHA3 state. **NOTE:** The register contents are unpredictable following any chip or core reset. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.	
HASHDAT[15] <63:0>	Cavium specific	SHA3 state **NOTE:** The register contents are unpredictable following any chip or core reset.	**cnMIPS III** only
HASHDAT[24] <63:0>	Cavium specific	SHA3 state **NOTE:** The register contents are unpredictable following any chip or core reset.	**cnMIPS III** only
HASHIV[7:0]<63:0>	Cavium specific	Hash IV. Also, [3:0] are SNOW3G result and FSM, ZUC result and FSM, and EIA3 MAC (T). Also, all are SHA3 state. **NOTE:** The register contents are unpredictable following any chip or core reset. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.	
3DESKEY[2:0] <63:0>	Cavium specific	3DES key. 3DESKEY[1:0] also is the KASUMI key. **NOTE:** The register contents are unpredictable following any chip or core reset. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.	
3DESIV<63:0>	Cavium specific	3DES IV **NOTE:** The register contents are unpredictable following any chip or core reset. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.	
3DESRESULT <63:0>	Cavium specific	3DES result. Also the KASUMI result. **NOTE:** The register contents are unpredictable following any chip or core reset. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.	
AESKEY[3:0] <63:0>	Cavium specific	AES key. AESKEY[1:0] is also the SMS4 key. **NOTE:** The register contents are unpredictable following any chip or core reset. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.	
AESKEYLEN<1:0>	Cavium specific	AES key length indicator. **NOTE:** The register contents are unpredictable following any chip or core reset. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.	
AESIV[1:0]<63:0>	Cavium specific	AES IV. Also the SMS4 Load IV. **NOTE:** The register contents are unpredictable following any chip or core reset. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.	
AESRESINP[1:0] <63:0>	Cavium specific	AES result/input. Also the SMS4 and CAMELLIA result/input. **NOTE:** The register contents are unpredictable following any chip or core reset. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.	

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 2–2 CPU Visible State Resident in Each cnMIPS III Core (Continued)

Register	Implemented/ Not Implemented	Comments
CVMSEG LM	Cavium specific	Local Scratchpad Memory **NOTE:** The scratchpad contents are unpredictable following any chip or core reset.

2.3 Cavium-Specific Instruction Summary

Table 2–3 through Table 2–7 list all Cavium-specific instructions in the cnMIPS III core.

Table 2–3 CPU Arithmetic (and Move) Instructions

Subcategory	Instruction	Comments	
	BADDU	Unsigned byte add. Refer to Section A.1 on page 1633.	
	DMUL	Register-direct 64-bit multiply. Refer to Section A.1 on page 1633.	
CPU Insert/Extract Instructions	EXTS EXTS32 CINS CINS32	Signed-bit field extract and clear/insert instructions. Refer to Section A.1 on page 1633.	
Large Multiply Instructions	MTM0 MTM1 MTM2 MTP0 MTP1 MTP2	Instructions to move data to/from Cavium-specific multiplier registers. Refer to Section A.1.2 on page 1639. **NOTE:** Not present when CvmCtl[NOMUL] = 1.	Updated for **cnMIPS III**
	V3MULU VMM0 VMULU	Large multiply instructions. Refer to Section A.1.2 on page 1639. **NOTE:** Not present when CvmCtl[NOMUL] = 1.	Updated for **cnMIPS III**
Population Count	DPOP POP	Count the number of ones in a 32-bit (POP) or a 64-bit (DPOP) variable. Refer to Section A.1.3 on page 1652.	
Set Equals /Set Not-Equals	SEQ SEQI SNE SNEI	SEQ and SNE have functionality similar to SLT. Refer to Section A.1.4 on page 1654.	
Q15 Multiply and Accumulate	QMACS.0x QMAC.0x	Q15 multiply and accumulate. Refer to Section A.1.5 on page 1658.	

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 2–4 CPU Branch and Jump Instructions

Subcategory	Instruction	Comments
Bit-test Branches	BBIT0 BBIT032 BBIT1 BBIT132	Bit-test branches. These instructions consume the MIPS LDC2, LWC2, SDC2, and SWC2 major opcodes. Refer to Section A.2.1 on page 1662.

Table 2–5 CPU Load, Store, and Memory Control Instructions

Subcategory	Instruction	Comments	
PREF	PREF 4	NOTL2; Prefetch into L1, do not put the block in L2. Refer to Section A.3 on page 1666.	
	PREF 5, PREF 28	L2; Prefetch into L2, do not put the block in L1. Refer to Section A.3 on page 1666.	
	PREF 24	PREFu; Prefetch TLB translation into the μTLB. Refer to Section A.3 on page 1666.	**cnMIPS III** only
	PREF 25	NUDGE/WRITEBACK_INVALIDATE. Refer to Section A.3 on page 1666.	
	PREF 26	NUDGE_WB. Refer to Section A.3 on page 1666.	
	PREF 27	EVICT_SOON. Refer to Section A.3 on page 1666.	
	PREF 29	DONT_WRITE_BACK (block locations are unpredictable until stored to). Refer to Section A.3 on page 1666.	
	PREF 30	PREPARE_FOR_STORE (block locations are unpredictable until stored to). Refer to Section A.3 on page 1666.	
	PREF 31	PREPARE_FOR_STORE_THROUGH (block locations are unpredictable until stored to). Refer to Section A.3 on page 1666.	
PREFX	PREFX 4	NOTL2; Prefetch into L1, do not put the block in L2. Refer to Section A.3 on page 1666.	**cnMIPS III** only
	PREFX 5, PREFX 28	L2; Prefetch into L2, do not put the block in L1. Refer to Section A.3 on page 1666.	
	PREFX 24	PREFu; Prefetch TLB translation into the μTLB. Refer to Section A.3 on page 1666.	
	PREFX 25	NUDGE/WRITEBACK_INVALIDATE. Refer to Section A.3 on page 1666.	
	PREFX 26	NUDGE_WB. Refer to Section A.3 on page 1666.	
	PREFX 27	EVICT_SOON. Refer to Section A.3 on page 1666.	
	PREFX 29	DONT_WRITE_BACK (block locations are unpredictable until stored to). Refer to Section A.3 on page 1666.	
	PREFX 30	PREPARE_FOR_STORE (block locations are unpredictable until stored to). Refer to Section A.3 on page 1666.	
	PREFX 31	PREPARE_FOR_STORE_THROUGH (block locations are unpredictable until stored to). Refer to Section A.3 on page 1666.	
Store Atomic Add	SAA SAAD	32-bit and 64-bit store atomic add instructions. Refer to Section A.3.2 on page 1671.	
SYNC	SYNCIOBDMA SYNCS SYNCW SYNCWS	Memory reference ordering instructions. Refer to Table 2–12 and Section A.3.3 on page 1675.	
Unaligned Load/Stores	ULD ULW USD USW	Unaligned Load/Store instructions. Included when CvmCtl[USEUN] is set. Refer to Section A.3.4 on page 1682.	

Cavium Confidential For Boyuen Yang AEO 04/27/2014

Table 2–5 CPU Load, Store, and Memory Control Instructions (Continued)

Subcategory	Instruction	Comments
Integer Indexed Loads	LBX LBUX LHX LHUX LWX LWUX LDX	Integer Indexed Load instructions. Refer to Section A.3.5 on page 1694.
Memory Atomic Fetch and Add	LAA LAAD	Memory Atomic Fetch and Add instructions. Refer to Section A.3.6 on page 1706.
Memory Atomic Swap	LAW LAWD	Memory Atomic Swap instructions. Refer to Section A.3.7 on page 1710.
Memory Atomic Increment	LAI LAID	Memory Atomic Increment instructions. Refer to Section A.3.8 on page 1714.
Memory Atomic Decrement	LAD LADD	Memory Atomic Decrement instructions. Refer to Section A.3.9 on page 1718.
Memory Atomic Set	LAS LASD	Memory Atomic Set instructions. Refer to Section A.3.10 on page 1722.
Memory Atomic Clear	LAC LACD	Memory Atomic Clear instructions. Refer to Section A.3.11 on page 1726.
Zero Cache Block	ZCB ZCBT	Zero Cache Block instructions. Refer to Section A.3.12 on page 1730.

Table 2–6 CPU Move and Privileged Instructions

Instruction	Comments
CACHE	Cache manipulation instruction. Refer to Section A.4 on page 1732.
RDHWR 31, RDHWR 30	64-bit cycle counter (31). Fast SSO switch access (30). Refer to Section A.4 on page 1732.
TLBR	Read a µTLB entry. Refer to Section A.4 on page 1732.
TLBWI	Write a µTLB entry. Refer to Section A.4 on page 1732.

Table 2–7 Coprocessor Move Instructions

Subcategory	Instruction	Comments
CRC	CVM_MF_CRC_IV CVM_MF_CRC_IV_REFLECT CVM_MF_CRC_LEN CVM_MF_CRC_POLYNOMIAL CVM_MT_CRC_BYTE CVM_MT_CRC_BYTE_REFLECT CVM_MT_CRC_DWORD CVM_MT_CRC_DWORD_REFLECT CVM_MT_CRC_HALF CVM_MT_CRC_HALF_REFLECT CVM_MT_CRC_IV CVM_MT_CRC_IV_REFLECT CVM_MT_CRC_LEN CVM_MT_CRC_POLYNOMIAL CVM_MT_CRC_POLYNOMIAL_REFLECT CVM_MT_CRC_VAR CVM_MT_CRC_VAR_REFLECT CVM_MT_CRC_WORD CVM_MT_CRC_WORD_REFLECT	Instructions to use the in-core CRC Coprocessor. Refer to Section A.5.1 on page 1741.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 2–7 Coprocessor Move Instructions (Continued)

Subcategory	Instruction	Comments
3DES	CVM_MF_3DES_IV CVM_MF_3DES_KEY CVM_MF_3DES_RESULT CVM_MT_3DES_DEC CVM_MT_3DES_DEC_CBC CVM_MT_3DES_ENC CVM_MT_3DES_ENC_CBC CVM_MT_3DES_IV CVM_MT_3DES_KEY CVM_MT_3DES_RESULT	Instructions to use the in-core 3DES Coprocessor. Refer to Section A.5.2 on page 1760. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.
AES	CVM_MF_AES_INP0 CVM_MF_AES_IV CVM_MF_AES_KEY CVM_MF_AES_KEYLENGTH CVM_MF_AES_RESINP CVM_MT_AES_DEC_CBC0 CVM_MT_AES_DEC_CBC1 CVM_MT_AES_DEC0 CVM_MT_AES_DEC1 CVM_MT_AES_ENC_CBC0 CVM_MT_AES_ENC_CBC1 CVM_MT_AES_ENC0 CVM_MT_AES_ENC1 CVM_MT_AES_IV CVM_MT_AES_KEY CVM_MT_AES_KEYLENGTH CVM_MT_AES_RESINP	Instructions to use the in-core AES Coprocessor. Refer to Section A.5.2 on page 1760. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.
Galois Field Multiplier	CVM_MF_GFM_MUL CVM_MF_GFM_POLY CVM_MF_GFM_RESINP CVM_MT_GFM_MUL CVM_MT_GFM_POLY CVM_MT_GFM_RESINP CVM_MT_GFM_XOR0 CVM_MT_GFM_XORMUL1	Instructions to use the Galois Field Multiplier. Refer to Section A.5.2 on page 1760. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.
Galois Field Multiplier REFLECT	CVM_MF_GFM_RESINP_REFLECT CVM_MT_GFM_MUL_REFLECT CVM_MT_GFM_XOR0_REFLECT CVM_MT_GFM_XORMUL1_REFLECT	Instructions to use the Galois Field Multiplier REFLECT. Refer to Section A.5.2 on page 1760. **NOTE:** Not present when CvmCtl[SNOW3G] = 0.
Hash (MD5/SHA-1/ SHA-256/SHA-512)	CVM_MF_HSH_DAT CVM_MF_HSH_DATW CVM_MF_HSH_IV CVM_MF_HSH_IVW CVM_MT_HSH_DAT CVM_MT_HSH_DATW CVM_MT_HSH_IV CVM_MT_HSH_IVW CVM_MT_HSH_STARTMD5 CVM_MT_HSH_STARTSHA CVM_MT_HSH_STARTSHA256 CVM_MT_HSH_STARTSHA512	Instructions to use the in-core HSH Coprocessor. (MD5/ SHA-1). Refer to Section A.5.2 on page 1760. **NOTE:** Not present when CvmCtl[NOCRYPTO] = 1.
KASUMI	CVM_MF_KAS_KEY CVM_MF_KAS_RESULT CVM_MT_KAS_KEY CVM_MT_KAS_RESULT CVM_MT_KAS_ENC CVM_MT_KAS_ENC_CBC	Instructions to use the in-core KASUMI coprocessor. Refer to Section A.5.2 on page 1760. **NOTE:** Not present when CvmCtl[KASUMI] = 0.

Cavium Confidential For Bowen Yang ALU 02/10/2014

Table 2–7 Coprocessor Move Instructions (Continued)

Subcategory	Instruction	Comments	
SNOW 3G	CVM_MF_SNOW3G_LFSR CVM_MF_SNOW3G_FSM CVM_MF_SNOW3G_RESULT CVM_MT_SNOW3G_LFSR CVM_MT_SNOW3G_FSM CVM_MT_SNOW3G_RESULT CVM_MT_SNOW3G_START CVM_MT_SNOW3G_MORE	Instructions to use the in-core SNOW 3G Coprocessor. Refer to Section A.5.2 on page 1760. **NOTE:** Not present when CvmCtl[SNOW3G] = 0.	
SMS4	CVM_MF_SMS4_IV CVM_MF_SMS4_KEY CVM_MF_SMS4_RESINP CVM_MT_SMS4_DEC_CBC0 CVM_MT_SMS4_DEC_CBC1 CVM_MT_SMS4_DEC0 CVM_MT_SMS4_DEC1 CVM_MT_SMS4_ENC_CBC0 CVM_MT_SMS4_ENC_CBC1 CVM_MT_SMS4_ENC0 CVM_MT_SMS4_ENC1 CVM_MT_SMS4_IV CVM_MT_SMS4_KEY CVM_MT_SMS4_RESINP	Instructions to use the in-core SMS4 Coprocessor. Refer to Section A.5.2 on page 1760. **NOTE:** Not present when CvmCtl[SMS4] = 0.	
ZUC	CVM_MT_ZUC_LFSR CVM_MT_ZUC_FSM CVM_MT_ZUC_RESULT CVM_MT_ZUC_START CVM_MT_ZUC_MORE CVM_MF_ZUC_LFSR CVM_MF_ZUC_FSM CVM_MF_ZUC_RESULT CVM_MF_ZUC_TRESULT	Instructions to use in-core ZUC Coprocessor. Refer to Section A.5.2 on page 1760. NOTE: Not present when CvmCtl[ZUC] = 0	**cnMIPS III** only
Camellia	CVM_MF_CAMELLIA_RESINP CVM_MT_CAMELLIA_RESINP CVM_MT_CAMELLIA_ROUND CVM_MT_CAMELLIA_FL CVM_MT_CAMELLIA_FLINV	Instructions to use in-core CAMELLIA Coprocessor. Refer to Section A.5.2 on page 1760. NOTE: Not present when CvmCtl[CAMELLIA] = 0	**cnMIPS III** only
SHA-3	CVM_MF_SHA3_DAT CVM_MT_SHA3_DAT CVM_MT_SHA3_XORDAT CVM_MT_SHA3_STARTOP	Instructions to use in-core SHA-3 Coprocessor. Refer to Section A.5.2 on page 1760. NOTE: Not present when CvmCtl[NOCRYPTO] = 1	**cnMIPS III** only

Cavium Confidential For Bowen Yang AU 02/27/2014

2.4 Instruction Set Summary

Table 2–8 through Table 2–29 list the complete cnMIPS III core instruction set.

Table 2–8 CPU Arithmetic Instructions

Instruction	Implemented/ Not Implemented	Comments	
ADD, ADDI, DADD, DADDI, DSUB, SUB	Implemented		
ADDU, ADDIU, DADDU, DADDIU, DSUBU, SUBU	Implemented		
BADDU	Implemented (Cavium specific)	Unsigned byte add. Refer to Section A.1 on page 1633.	
CLO, CLZ, DCLO, DCLZ	Implemented		
DDIV, DDIVU, DIV, DIVU	Implemented	These instructions make the Cavium large multiply registers P0, P1, P2, P3, P4, and P5 unpredictable.	
DMUL	Implemented (Cavium specific)	Register-direct 64-bit multiply. Refer to Section A.1 on page 1633. This instruction makes the Cavium large multiply registers P0, P1, P2, P3, P4, and P5 unpredictable.	
DMULT, DMULTU, MULT, MULTU	Implemented	These instructions make the Cavium large multiply registers P0, P1, P2, P3, P4, and P5 unpredictable.	
DPOP	Implemented (Cavium specific)	64-bit population count. Refer to Section A.1.3 on page 1652.	
MUL, MADD, MADDU, MSUB, MSUBU	Implemented	These instructions make the Cavium large multiply registers P0, P1, P2, P3, P4, and P5 unpredictable.	
POP	Implemented (Cavium specific)	32-bit population count. Refer to Section A.1.3 on page 1652.	
QMACS.0x, QMAC.0x	Implemented (Cavium specific)	Q15 Multiply and Accumulate. Refer to Section A.1.5 on page 1658.	
SEB, SEH	Implemented		
SEQ, SEQI, SNE, SNEI	Implemented (Cavium specific)	Equal/not equal comparison. Refer to Section A.1.4 on page 1654.	
SLT, SLTI, SLTIU, SLTU	Implemented		
VMM0, V3MULU, VMULU	Implemented (Cavium specific)	Large multiplies. Refer to Section A.1.2 on page 1639.	Updated for **cnMIPS III**

Table 2–9 CPU Branch and Jump Instructions

Instruction	Implemented/ Not Implemented	Comments
BBIT0, BBIT032, BBIT1, BBIT132	Implemented (Cavium specific)	New bit test instructions not in the MIPS64 architecture. These instructions consume the MIPS LDC2, LWC2, SDC2, and SWC2 major opcodes. They are not, however, considered coprocessor instructions. Refer to Section A.2.1 on page 1662.
BEQ (B), BGEZ, BGTZ, BGEZAL (BAL), BLTZAL, BLEZ, BLTZ, BNE, J, JR, JAL, JALR.HB, JR.HB,	Implemented	
BEQL, BGEZALL, BGEZL, BGTZL, BLEZL, BLTZALL, BLTZL, BNEL	Implemented	Deprecated. Software is strongly encouraged to avoid use of the Branch likely instructions.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 2–9 CPU Branch and Jump Instructions (Continued)

Instruction	Implemented/ Not Implemented	Comments
JALX	Not Implemented	This instruction is not implemented because Cavium does not implement microMIPS64 nor MIPS16e.

Table 2–10 CPU Instruction Control Instructions

Instruction	Implemented/ Not Implemented	Comments
NOP, SSNOP, EHB	Implemented	
PAUSE	Implemented	PAUSE instructions will wake up on a tag switch completion and will periodically wake up at a frequency selected by CvmMemCtl[PAUSETIME], in addition to other reasons. In cnMIPS III, a coherent invalidation signal after a load-locked will cause PAUSE to exit. This can be disabled by setting CvmMemCtl2[DISPAUSELL] = 1.

Table 2–11 CPU Load, Store, and Memory Control Instructions

Instruction	Implemented/ Not Implemented	Comments
LB, LBU, SB	Implemented	
LD, LH, LHU, LW, LWU, SD, SH, SW	Implemented as specified in the MIPS specifications when CvmCtl[REPUN] is clear. Unaligned memory references are automatically completed by hardware when CvmCtl[REPUN] is set.	When CvmCtl[REPUN] is set, unaligned memory addresses are not trapped, they are instead completed by the hardware. The hardware cost is ~10 extra cycles when the different bytes required by the operation cross an aligned 64-bit boundary, so the ULD, ULW, USD, and USW instructions may be advantageous in this case. Refer to Section A.3.4 on page 1682 for descriptions of the ULD, ULW, USD, and USW instructions. Address errors are always generated on accesses to unaligned I/O space addresses.
LL, LLD, SC, SCD	Implemented	Refer to "Load-Linked/Store-Conditional and Wide Atomic Memory Write Operations" on page 209.
LDL, LWL, SDL, SWL LDR, LWR, SDR, SWR	Implemented when CvmCtl[USEUN] is clear.	The LDL, LWL, SDL, and SWL instruction opcodes are otherwise the ULD, ULW, USD, and USW instructions or NOPs when CvmCtl[USEUN] is set. Refer to Section A.3.4 on page 1682 for descriptions of the ULD, ULW, USD, and USW instructions. Not supported to I/O space addresses.
LBE, LBUE, LHE, LHUE, LLE, LWE, LWLE, LWRE, PREFE, SBE, SCE, SHE, SWE, SWLE, SWRE	Not implemented	Segmentation not implemented.

Cavium Confidential For Boyce Yang 09/27/2012

Table 2–11 CPU Load, Store, and Memory Control Instructions (Continued)

Instruction	Implemented/ Not Implemented	Comments		
LBX, LBUX, LHX, LHUX, LWX, LWUX, LDX	Implemented (Cavium specific)	Integer Indexed Load instructions. The LBUX, LHX, and LWX instructions are in the The MIPS® DSP Application-Specific Extension. When CvmCtl[REPUN] is set, unaligned memory addresses are not trapped, they are instead completed by the hardware. The hardware cost is ~10 extra cycles when the different bytes required by the operation cross an aligned 64-bit boundary, so the ULD, ULW, USD, and USW instructions may be advantageous in this case. Refer to Section A.3.4 on page 1682 for descriptions of the ULD, ULW, USD, and USW instructions. For LHX, LHUX, LWX, LWUX, and LDX instructions, address errors are always generated on accesses to unaligned I/O space addresses.		
LAA, LAAD	Implemented (Cavium specific)	Memory Atomic Fetch and Add instructions. Refer to Section A.3.6 on page 1706. Behavior unpredictable on accesses to I/O space addresses.		
LAW, LAWD	Implemented (Cavium specific)	Memory Atomic Swap instructions. Refer to Section A.3.7 on page 1710. Behavior unpredictable on accesses to I/O space addresses.		
LAI, LAID	Implemented (Cavium specific)	Memory Atomic Increment instructions. Refer to Section A.3.8 on page 1714. Behavior unpredictable on accesses to I/O space addresses.		
LAD, LADD	Implemented (Cavium specific)	Memory Atomic Decrement instructions. Refer to Section A.3.9 on page 1718. Behavior unpredictable on accesses to I/O space addresses.		
LAS, LASD	Implemented (Cavium specific)	Memory Atomic Set instructions. Refer to Section A.3.10 on page 1722. Behavior unpredictable on accesses to I/O space addresses.		
LAC, LACD	Implemented (Cavium specific)	Memory Atomic Clear instructions. Refer to Section A.3.11 on page 1726. Behavior unpredictable on accesses to I/O space addresses.		
PREF	Implemented (Cavium specific)	Prefetch instructions. Refer to Section A.3.1 on page 1666.	4, 5, 25, 26,27, 28, 29, 30, 31	
			24	**cnMIPS III** only
PREFX	Implemented (Cavium specific)	Prefetch Indexed. Refer to Section A.3.1 on page 1666. Software must set STATUS[CU1] = 1 to enable this instruction.		**cnMIPS III** only
SAA, SAAD	Implemented (Cavium specific)	32-bit and 64-bit store atomic add instructions. Refer to Section A.3.2 on page 1671.		
SYNC	Implemented	Refer to Table 2–12 on page 74.		
SYNCI	Implemented	Flushes the entire Icache. SYNCI instructions never match a watchpoint/breakpoint on CN71XX.		
SYNCIOBDMA, SYNCS, SYNCW, SYNCWS	Implemented (Cavium specific)	SYNC instructions not included in the MIPS ISA. Refer to Section A.3.3 on page 1675.		

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 2–11 CPU Load, Store, and Memory Control Instructions (Continued)

Instruction	Implemented/ Not Implemented	Comments
ULD, ULW, USD, USW	Cavium-specific instructions that are available when CvmCtl[USEUN] is set	Unaligned Load and Store operations (ULW, USW = 32-bit, ULD, USD = 64-bit). Instructions included when CvmCtl[USEUN] is set. Consume the LDL, LWL, SDL, SWL, LDR, LWR, SDR, SWR opcodes when CvmCtl[USEUN] is set. Refer to Section A.3.4 on page 1682. Address errors are always generated on accesses to unaligned I/O space addresses.
ZCB, ZCBT	Implemented (Cavium specific)	Zero Cache Block instructions. Refer to Section A.3.12 on page 1730.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Table 2–12 SYNC Instruction Stype Encodings

Code	MIPS Stype Name	Cavium Instruction Name	Comments
0x0	SYNC or SYNC 0	—	MIPS SYNC instruction.
0x2	—	SYNCIOBDMA	SYNCIOBDMA. See Section A.3.3 on page 1675.
0x4	SYNC_WMB or SYNC 4	SYNCW	SYNCW. See Section A.3.3 on page 1675
0x5	—	SYNCWS	SYNCWS. See Section A.3.3 on page 1675.
0x6	—	SYNCS	SYNCS. See Section A.3.3 on page 1675.
0x10	SYNC_MB or SYNC 16	—	Implemented same as SYNC.
0x11	SYNC_ACQUIRE or SYNC 17	—	Implemented as a NOP.
0x12	SYNC_RELEASE or SYNC 18	SYNCW	Implemented as a SYNCW. See Section A.3.3 on page 1675.
0x13	SYNC_RMB or SYNC 19	—	Implemented as a NOP.
0x1, 0x3, 0x7–0x9, 0x14–0x1F	—	—	Reserved

Table 2–13 CPU Logical Instructions

Instruction	Implemented/ Not Implemented	Comments
AND, ANDI, LUI, NOR, OR, ORI, XOR, XORI	Implemented	

Table 2–14 CPU Insert/Extract Instructions

Instruction	Implemented/ Not Implemented	Comments
DEXT, DEXTM, DEXTU, DINS, DINSM, DINSU, DSBH, DSHD, EXT, INS, WSBH	Implemented	
EXTS, EXTS32, CINS, CINS32	Implemented (Cavium specific)	Signed bit field extract and clear/insert instructions not included in the MIPS64 architecture. Refer to Section A.1.1 on page 1635.

Cavium Confidential For
BoWen Yang
ALU
02/27/2014

Table 2–15 CPU Move Instructions

Instruction	Implemented/ Not Implemented	Comments	
MFHI, MFLO, MTHI, MTLO, MOVN, MOVZ	Implemented		
MOVF, MOVT	Implemented	Software must set STATUS[CU1] = 1 to enable these instructions.	**cnMIPS III** only
MTM0, MTM1, MTM2, MTP0, MTP1, MTP2	Implemented (Cavium specific)	Instructions to move data to/from Cavium-specific multiplier registers. Refer to Section A.1.2 on page 1639.	Updated for **cnMIPS III**
RDHWR	Implemented	RDHWR 0, 1, 2, 3 already in MIPS Specifications.	
	Implemented	RDHWR 29 already in MIPS Specifications.	
	Implemented (Cavium specific)	RDHWR 30. Refer to Section A.4 on page 1732.	
	Implemented (Cavium specific)	RDHWR 31. Refer to Section A.4 on page 1732.	

Table 2–16 CPU Shift Instructions

Instruction	Implemented/ Not Implemented	Comments
DROTR, DROTR32, DROTRV, ROTR, ROTRV DSLL, DSLL32, DSLLV, DSRA, DSRA32, DSRAV, DSRL, DSRL32, DSRLV, SLL, SLLV, SRA, SRAV, SRL, SRLV	Implemented	

Table 2–17 CPU Trap Instructions

Instruction	Implemented/ Not Implemented	Comments
BREAK, SYSCALL, TEQ, TEQI, TGE, TGEI, TGEIU, TGEU, TLT, TLTI, TLTIU, TLTU, TNE, TNEI	Implemented	

Table 2–18 FPU Arithmetic Instructions

Instruction	Implemented/ Not Implemented	Comments	
ABS.S, ABS.D, ADD.S, ADD.D, DIV.S, DIV.D, MUL.S, MUL.D, NEG.S, NEG.D, SQRT.S, SQRT.D, SUB.S, SUB.D, RECIP.S, RECIP.D, RSQRT.S, RSQRT.D, MADD.S, MADD.D, MSUB.S, MSUB.D, NMADD.S, NMADD.D, NMSUB.S, NMSUB.D	Implemented	Software must set STATUS[CU1] = 1 to enable these instructions.	**cnMIPS III** only
ABS.PS, ADD.PS, DIV.PS, MUL.PD, NEG.PS, SQRT.PS, SUB.PS, RECIP.PS, RSQRT.PS, MADD.PS, MSUB.PS, NMADD.PS, NMSUB.PS	Not Implemented	Paired Single not implemented in cnMIPS III.	

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 2–19 FPU Branch Instructions

Instruction	Implemented/ Not Implemented	Comments	
BC1F, BC1T, BC1FL, BC1TL	Implemented	FP branch instructions. Branch is based on FP condition code value. Software must set STATUS[CU1] = 1 to enable these instructions.	**cnMIPS III** only

Table 2–20 FPU Compare Instructions

Instruction	Implemented/ Not Implemented	Comments	
C.F.S, C.UN.S, C.EQ.S, C.UEQ.S, C.OLT.S, C.ULT.S, C.OLE.S, C.ULE.S, C.SF.S, C.NGLE.S, C.SEQ.S, C.NGL.S, C.LT.S, C.NGE.S, C.LE.S, C.NGT.S, C.F.D, C.UN.D, C.EQ.D, C.UEQ.D, C.OLT.D, C.ULT.D, C.OLE.D, C.ULE.D, C.SF.D, C.NGLE.D, C.SEQ.D, C.NGL.D, C.LT.D, C.NGE.D, C.LE.D, C.NGT.D	Implemented	These FP compare instructions set the FP condition code registers. Software must set STATUS[CU1] = 1 to enable these instructions.	**cnMIPS III** only
C.F.PS, C.UN.PS, C.EQ.PS, C.UEQ.PS, C.OLT.PS, C.ULT.PS, C.OLE.PS, C.ULE.PS, C.SF.PS, C.NGLE.PS, C.SEQ.PS, C.NGL.PS, C.LT.PS, C.NGE.PS, C.LE.PS, C.NGT.PS,	Not Implemented	Paired Single not implemented in cnMIPS III.	

Table 2–21 FPU Convert Instructions

Instruction	Implemented/ Not Implemented	Comments	
CVT.D.L, CVT.D.S, CVT.D.W, CVT.L.D, CVT.L.S, CVT.L.W, CVT.S.D, CVT.S.L, CVT.S.W, CVT.W.D, CVT.W.L, CVT.W.S, CEIL.L.D, CEIL.L.S, CEIL.W.D, CEIL.W.S, FLOOR.L.D, FLOOR.L.S, FLOOR.W.D, FLOOR.W.S, ROUND.L.D, ROUND.L.S, ROUND.W.D, ROUND.W.S, TRUNC.L.D, TRUNC.L.S, TRUNC.W.D, TRUNC.W.S	Implemented	FP conversion instructions. Software must set STATUS[CU1] = 1 to enable these instructions.	**cnMIPS III** only
CVT.PS.S, CVT.S.PL, CVT.S.PU	Not Implemented	Paired Single not implemented in cnMIPS III.	

Table 2–22 FPU Load, Store, and Memory Control Instructions

Instruction	Implemented/ Not Implemented	Comments	
LDC1, LWC1, SDC1, SWC1, LDXC1, LUXC1, LWXC1, SDXC1, SUXC1, SWXC1	Implemented	Software must set STATUS[CU1] = 1 to enable these instructions.	**cnMIPS III** only

Table 2–23 FPU Move Instructions

Instruction	Implemented/ Not Implemented	Comments	
CFC1, CTC1, DMFC1, DMTC1, MFC1, MFHC1, MTC1, MTHC1, MOV.D, MOV.S, MOVT.D, MOVT.S, MOVN.D, MOVN.S, MOVZ.D, MOVZ.S, MOVN, MOVZ	Implemented	Software must set STATUS[CU1] = 1 to enable these instructions.	**cnMIPS III** only
MOV.PS, MOVT.PS, MOVN.PS, MONZ.PS, ALNV.PS, PLL.PS, PLU.PS, PUL.PS, PUU.PS	Not Implemented	Paired Single not implemented	

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 2–24 Coprocessor Branch Instructions

Instruction	Implemented/ Not Implemented	Comments
BC2F, BC2T, BC2FL, BC2TL	Not implemented	

Table 2–25 Coprocessor Execute Instructions

Instruction	Implemented/ Not Implemented	Comments
COP2	Not Implemented	

Table 2–26 Coprocessor Load and Store Instructions

Instruction	Implemented/ Not Implemented	Comments
LDC2, LWC2, SDC2, SWC2	Not Implemented	These opcodes are consumed by the Cavium-specific BBIT0, BBIT032, BBIT1, BBIT132 instructions. Refer to Section A.2.1 on page 1662.

Table 2–27 Coprocessor Move Instructions

Instruction	Implemented/ Not Implemented	Comments
CFC2, CTC2, MFC2, MTC2, MFHC2, MTHC2	Not Implemented	
DMFC2, DMTC2	Implemented (Cavium specific)	Refer to Table 2–7 on page 67 and Section A.5 on page 1741.

Table 2–28 Privileged Instructions

Instruction	Implemented/ Not Implemented	Comments
CACHE	Implemented (Cavium specific)	Cache manipulation instruction, addresses never-match watchpoints/ breakpoints on CN71XX. Refer to Section A.4 on page 1732.
CACHEE	Not implemented	Not required for TLB.
ERET, DI, EI, DMFC0, DMTC0, MFC0, MTC0	Implemented	
RDPGPR, WRPGPR	Implemented	Implemented as a register-to-register copy. The cnMIPS III cores have no shadow register set, so both rd and rt refer to the main register file.
TLBP, TLBWR		A machine-check may occur during a TLB write.
TLBR, TLBWI, (CvmMemCtl2[UTLBRW] = 0)	Implemented	
TLBR, TLBWI (CvmMemCtl2[UTLBRW] = 1)	Implemented (Cavium specific)	These instructions operate on the μTLB using the registers UtlbIndex, UtlbEntry0, UtlbEntry1, and UtlbEntry2. To be used only to debug the μTLB.
WAIT	Implemented	CvmCtl[DISWAIT] changes WAIT into a NOP.
HYPCALL, DMFGC0, DMTGC0, MFGC0, MTGC0, TLBGP, TLBGR, TLBGWI, TLBGWR	Implemented (Virtualization module)	VRM OCLA
TLBGINV, TLBGINVF	Not implemented (Virtualization module)	Not required for TLB.
TLBINV, TLBINVF	Not implemented	Not required for TLB.

Cavium Confidential For Bowen Yang A.L.U 02/27/2014

Table 2–29 EJTAG Instructions

Instruction	Implemented/ Not Implemented	Comments
DERET, SDBBP	Implemented	See "Hardware Debug Features" on page 206.

2.5 Virtual Addresses and CVMSEG

SEGBITS = 49, as per the MIPS specifications.

When CvmMemCtl[CVMSEGENAU/S/K] is set for the appropriate kernel (K), supervisor, (S) or user (U) mode, virtual addresses for load/store operations in the KSEG3 range 0xFFFF FFFF FFFF 8000 to 0xFFFF FFFF FFFF BFFF are called CVMSEG references and are treated specially by the cnMIPS III cores. Instruction references to these addresses are always treated as normal KSEG3 references in the cores. When CvmMemCtl[CVMSEGENAU/S/K] is clear for the appropriate mode, these CVMSEG addresses act as normal MIPS-defined KSEG3 addresses.

CVMSEG is present in the hardware when CvmMemCtl[CVMSEGENAK] is set. CVMSEG has two parts:

CVMSEG LM = 0xFFFF FFFF FFFF 8000 to 0xFFFF FFFF FFFF 9FFF

CVMSEG I/O = 0xFFFF FFFF FFFF A000 to 0xFFFF FFFF FFFF BFFF

CVMSEG LM makes a portion of the Dcache available as a local memory. CvmMemCtl[LMEMSZ] selects the valid size of CVMSEG LM. CvmMemCtl[LMEMSZ] can legally range from 0 to 54 cache blocks (i.e. CVMSEG LM is between 0 and 6912 bytes). The legal CVMSEG LM addresses (when CvmMemCtl[LMEMSZ] > 0) start at virtual address 0xFFFF FFFF FFFF 8000 and can increase up to the maximum possible legal CVMSEG LM virtual address 0xFFFF FFFF FFFF 9AFF (which is only legal when CvmMemCtl[LMEMSZ] is 54). CVMSEG LM references above the range allocated by CvmMemCtl[LMEMSZ] (but at 0xFFFF FFFF FFFF 9FFF or below) cause an address error when CVMSEG is present.

CVMSEG I/O has a limited number of valid addresses:

- 0xFFFF FFFF FFFF A200, and
- 0xFFFF FFFF FFFF B200

  SD instructions to these CVMSEG I/O addresses create IOBDMA commands. IOBDMA commands return data from IOI devices into selected CVMSEG LM locations. Refer to Section 2.7.

When CVMSEG I/O is enabled (i.e. when CvmMemCtl[CVMSEGENAU/S/K] is set for the corresponding mode), accesses to the CVMSEG I/O range 0xFFFF FFFF FFFF A000 – 0xFFFF FFFF FFFF BFFF cause address-error exceptions in all of the following situations:

- The instruction accessing the CVMSEG I/O address is not a naturally-aligned SD instruction.
- The instruction references a CVMSEG I/O address not listed above
- The core is not in kernel or debug modes and CvmMemCtl[CVMSEGIODIS] = 1.
- Root.CvmMemCtl[CVMSEGIODIS] = 1 and the core is in guest mode (this is a root exception taken in guest mode).

Cavium Confidential For Bowen Yang ALU 02/27/2014

Independent Guest.CvmMemCtl[CVMSEGIODIS,CVMSEGENAU/S/K,LMEMSZ] and Root.CvmMemCtl[CVMSEGIODIS,CVMSEGENAU/S/K,LMEMSZ] CVMSEG configuration variables are available, for guest and root mode, respectively. When CVMSEG LM is present in both guest and root modes, CVMSEG LM is shared by the modes - successful accesses to the same CVMSEG LM address are to the same shared memory location, regardless of the mode.

The hardware CVMSEG LM size is:

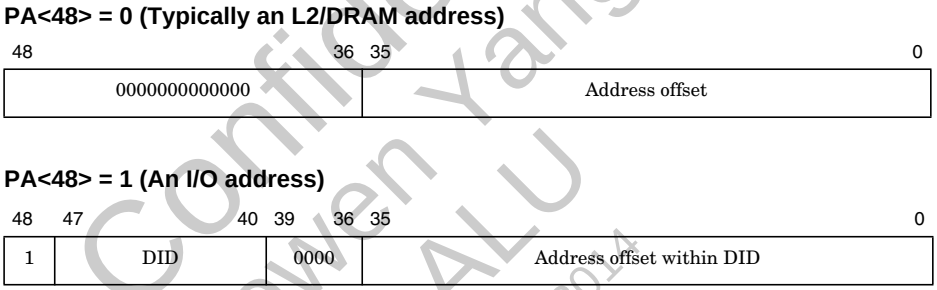
Guest.CvmMemCtl [CVMSEGENAK]	Root.CvmMemCtl [CVMSEGENAK]	Hardware CVMSEG LM Size in Cache blocks
0	0	0
0	1	Root.CvmMemCtl[LMEMSZ]
1	0	Guest.CvmMemCtl[LMEMSZ]0
1	1	MAXIMUM (Guest.CvmMemCtl[LMEMSZ], Root.CvmMemCtl[LMEMSZ])

The larger the hardware CVMSEG LM size, the smaller the data cache size.

2.6 Physical Addresses

The physical address is described in Figure 2–2. The number of physical address bits (PABITS) is 49, as per the MIPS specifications.

Figure 2–2 Forty-Nine-bit Physical Address Format



- **Address offset** - Selects the (typically) L2/DRAM location on the node.
- **DID** - Directs the request to the correct I/O (i.e. non-L2/DRAM) IOI hardware block on the node.
- **Address offset within DID** - IOI device location.

NOTE: CN71XX does not use the MIPS-defined cacheability and coherency attributes to determine cacheability. Only the physical addresses determine cacheability. Except for physical addresses in the range 0x1000 0000 to 0x1FFF FFFF, CN71XX caches all addresses with PA<48> = 0. CN71XX does not cache an address that either has PA<48> = 1 or is in the range 0x1000 0000 to 0x1FFF FFFF. These addresses are called I/O space addresses.

NOTE: The physical addressing implementation does not support uncached DRAM addressing or cached I/O space accesses.

Cavium Confidential For Bowen Yang ALU 8/27/2014

Physical addresses in the range 0x1000 0000 – 0x1FFF FFFF are exceptional. The hardware normally converts these physical addresses to a corresponding boot bus I/O physical address. (This hole conversion does not occur on the first of the two stages of guest virtual-to-physical address translation when Root.CvmVMConfig[DGHT] = 1.)

Physical Address Detail:

Table 2–30 cnMIPS III Core Physical Addresses

Physical Addresses	Mem / Bus	Comment
0x0 0000 0000 0000 to 0x0 0000 0FFF FFFF	DR0 DRAM	Cached (first 256 MB of DRAM) for load/store operations only.
0x0 0000 1000 0000 to 0x0 0000 1FFF FFFF	Boot Bus	Uncached I/O space for load/store/IOBDMA operations. Converted to a boot-bus address if not the first stage of guest address translation (i.e. if a GPA is not being created) or if Root.CvmVMConfig[DGHT] = 0: 0x1 0000 1000 0000 – 0x1 0000 1FFF FFFF
0x0 0000 2000 0000 to 0x0 00FF FFFF FFFF	DR1 DRAM	Cached (all DRAM above first 256 MB of DRAM) for load/store operations only.
0x1 0000 0000 0000 to 0x1 0000 FFFF FFFF	Boot Bus	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 0700 0000 0000 to 0x1 0700 0000 FFFF	CIU, GPIO, MIO PTP, and MPI NCB type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 0700 0100 0000 to 0x1 0700 0100 00A0	NDF NCB type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 0000 0000 to 0x1 1800 00FF FFFF	MIO BOOT, RST, NDF, FUS, TWSI0, TWSI1. UART0, UART1, SMI RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 0600 0000 to 0x1 1800 06FF FFFF	RST RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 0800 0000 to 0x1 1800 08FF FFFF	GMX0 RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 1000 0000 to 0x1 1800 10FF FFFF	GMX1 RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 2000 0000 to 0x1 1800 20FF FFFF	KEY RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 2100 0000 to 0x1 1800 21FF FFFF	VRM RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 2800 0000 to 0x1 1800 28FF FFFF	FPA RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 3700 0000 to 0x1 1800 3700 07F8	HFA (DFA) RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 4000 0000 to 0x1 1800 40XX XXXX	RNM RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 5000 0000 to 0x1 1800 5000 1FFF	PKO RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations only.
0x1 1800 5800 0000 to 0x1 1800 58FF FFFF	TIM RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 68000000 to 0x1 1800 68FF FFFF	USBDRD0 UCTL0 (USB0) RSL CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 69000000 to 0x1 1800 69FF FFFF	USBDRD1 (USB1) UCTL1 RSL CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 6C000000 to 0x1 1800 6CFF FFFF	SATA RSL CSRs	Uncached (I/O space) for load/store/IOBDMA operations.

Cavium Confidential For Bowen Yang ALU 6/27/2014

Table 2–30 cnMIPS III Core Physical Addresses (Continued)

Physical Addresses	Mem / Bus	Comment
0x1 1800 7000 0000 0x1 1800 7000 1FFF	RAD RSL CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 71000000 to 0x1 1800 71FF FFFF	BCH RSL CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 8080 0000 to 0x1 1800 80FF FFFF	L2C RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 8800 0000 to 0x1 1800 88FF FFFF	LMC0 RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 9000 0000 to 0x1 1800 90FF FFFF	GSER0 RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 A000 0000 to 0x1 1800 A000 1FFF	PIP RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 A800 0000 to 0x1 1800 A8FF FFFF	OCLA0 RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 B000 1000 to 0x1 1800 B0FF FFFF	PCS0/PCSX0 RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 B100 1000 to 0x1 1800 B1FF FFFF	PCS1 RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 C000 0000 to 0x1 1800 C0FF FFFF	PEM0 RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 C100 0000 to 0x1 1800 C1FF FFFF	PEM1 RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 C200 0000 to 0x1 1800 C2FF FFFF	PEM2 RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 E000 0000 to 0x1 1800 E000 20A0	AGL RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 F000 0000 to 0x1 1800 F000 FFFF	IOB RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1800 FE00 0000 to 0x1 1800 FEFF FFFF	DTX RSL type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1900 0000 0000 to 0x1 190F FFFF FFFF	PCIe Config/IACK/Special space	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1A00 0000 0000 to 0x1 1A0F FFFF FFFF	PCIe I/O space	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 1B00 0000 0000 to 0x1 1B03 FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID12
0x1 1B04 0000 0000 to 0x1 1B07 FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID13
0x1 1B08 0000 0000 to 0x1 1B0B FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID14
0x1 1B0C 0000 0000 to 0x1 1B0F FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID15
0x1 1C00 0000 0000 to 0x1 1C03 FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID16
0x1 1C04 0000 0000 to 0x1 1C07 FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID17
0x1 1C08 0000 0000 to 0x1 1C0B FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID18
0x1 1C0C 0000 0000 to 0x1 1C0F FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID19
0x1 1D00 0000 0000 to 0x1 1D03 FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID20

Cavium Confidential For Bowell Yang ALU 04/27/2014

Table 2–30 cnMIPS III Core Physical Addresses (Continued)

Physical Addresses	Mem / Bus	Comment
0x1 1D04 0000 0000 to 0x1 1D07 FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID21
0x1 1D08 0000 0000 to 0x1 1D0B FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID22
0x1 1D0C 0000 0000 to 0x1 1D0F FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID23
0x1 1E00 0000 0000 to 0x1 1E03 FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID24
0x1 1E04 0000 0000 to 0x1 1E07 FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID25
0x1 1E08 0000 0000 to 0x1 1E0B FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID26
0x1 1E0C 0000 0000 to 0x1 1E0F FFFF FFFF	PCIe Memory space	Uncached (I/O space) for load/store/IOBDMA operations. See SLI_MEM_ACCESS_SUBID27
0x1 1F00 0000 0000 to 0x1 1F0F FFFF FFFF	SLI PEXP_NCB type CSRs, doorbells	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 2000 0000 0000 to 0x1 2000 0000 1FFF	KEY Memory operations	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 2800 0000 0000 to 0x1 280F FFFF FFFF	FPA Pool 0 Allocate/Free operations	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 2900 0000 0000 to 0x1 290F FFFF FFFF	FPA Pool 1 Allocate/Free operations	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 2A00 0000 0000 to 0x1 2A0F FFFF FFFF	FPA Pool 2 Allocate/Free operations	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 2B00 0000 0000 to 0x1 2B0F FFFF FFFF	FPA Pool 3 Allocate/Free operations	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 2C00 0000 0000 to 0x1 2C0F FFFF FFFF	FPA Pool 4 Allocate/Free operations	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 2D00 0000 0000 to 0x1 2D0F FFFF FFFF	FPA Pool 5 Allocate/Free operations	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 2E00 0000 0000 to 0x1 2E0F FFFF FFFF	FPA Pool 6 Allocate/Free operations	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 2F00 0000 0000 to 0x1 2F0F FFFF FFFF	FPA Pool 7 Allocate/Free operations	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 3700 0000 0000 to 0x1 3707 FFFF FFFF	HFA (DFA) NCB type CSRs and operations	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 4000 0000 0000 to 0x1 4000 0000 07FF	RNG Load/IOBDMA operations	Uncached (I/O space) for IOBDMA operations only.
0x1 4F00 0000 0000 to 0x1 4F00 0000 07FF	IPD NCB type CSRs	Uncached (I/O space) for IOBDMA operations.
0x1 5200 0000 0000 to 0x1 5200 0003 FFFF	PKO doorbell store operations	Uncached (I/O space) for store operations only.
0x1 6000 0000 0000 to 0x1 600F FFFF FFFF	POW getwork load/iobdma operations, store work operations	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 6100 0000 0000 to 0x1 610F FFFF FFFF	POW status load operations, store work operations	Uncached (I/O space) for load/store/IOBDMA operations only.
0x1 6200 0000 0000 to 0x1 6200 0000 FFFF	POW memory load operations	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 6300 0000 0000 to 0x1 630F FFFF FFFF	POW index load operations, store operations	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 6400 0000 0000 to 0x1 6400 0000 0007	POW ALLOC_WE load operations	Uncached (I/O space) for load/IOBDMA operations only.

Cavium Confidential For Bowei Yang ALU 02/27/2014

Table 2–30 cnMIPS III Core Physical Addresses (Continued)

Physical Addresses	Mem / Bus	Comment
0x1 6700 0000 1000 to 0x1 670F FFFF FFFF	POW NCB type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 6800 0000 0000 to 0x1 680F FFFF FFFF	USBDRD0 UAHC0 (USB) NCB type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 6900 0000 0000 to 0x1 690F FFFF FFFF	USBDRD1 UAHC1 (USB) NCB type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 6C00 0000 0000 to 0x1 6C0F FFFF FFFF	SATA NCB type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 7000 0000 0000 to 0x1 700F FFFF FFFF	RAD doorbell store operations	Uncached (I/O space) for store operations only.
0x1 7100 0000 0000 to 0x1 710F FFFF FFFF	BCH doorbell store operations	Uncached (I/O space) for store operations only.
0x1 DF00 0000 0000 to 0x1 DF0F FFFF FFFF	DPI NCB type CSRs	Uncached (I/O space) for load/store/IOBDMA operations.
0x1 F000 0000 0000 to 0x1 F00F FFFF FFFF	FAU operations	Uncached (I/O space) for load/store/IOBDMA operations.

Cavium Confidential For Bowen Yang ALU 02/27/2014

2.7 IOBDMA Operations

In addition to normal load and store operations, IOBDMA operations are a way for cnMIPS cores to interact with IOI devices.

IOBDMA operations move data between CVMSEG LM in the cnMIPS core and the IOI device. An IOBDMA operation can send an address from the cnMIPS core to the IOI device and return a variable-length vector (up to 128 bytes of 64-bit words) from the IOI device into CVMSEG LM. An IOBDMA operation is much like an I/O load operation, but it can be multiword instead of just single-word.

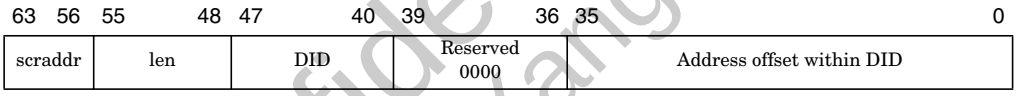
Note that IOBDMA operations cannot reference L2/DRAM addresses; they can only reference I/O addresses.

Section 2.5 describes CVMSEG LM, and it also describes how IOBDMA operations are initiated by stores to CVMSEG I/O addresses.

Unlike normal I/O load operations, IOBDMA operations do not wait for the return data to be deposited into the selected CVMSEG LM addresses before executing subsequent instructions. SYNCIOBDMA or SYNC/SYNCS instructions stall subsequent instructions until the CVMSEG LM addresses are updated for all prior IOBDMA operations, however.

The 64-bit store data format for a CVMSEG I/O store access is shown in Figure 2–3.

Figure 2–3 Sixty-Four-Bit IOBDMA Store Data Format



- **scraddr** - A (64-bit word) CVMSEG LM starting location where the core puts the return data. The return data starts at the CVMSEG LM address

  0xFFFF FFFF FFFF 8000 + (8 × effective_scraddr),

  where effective_scraddr = scraddr + (Root.CvmMemCtl[IOBDMASCRMSB] << 8)

NOTE: IOBDMA operations can start only in the first 2 KB of CVMSEG LM, unless Root.CvmMemCtl[IOBDMASCRMSB] is non-zero. Guest.CvmMemCtl[IOBDMASCRMSB] always equals Root.CvmMemCtl[IOBDMASCRMSB].

- **len** - The number of 64-bit words in the result. The results are placed sequentially in CVMSEG LM from the starting address indicated by scraddr.

  $1 \le len \le 16$

- **DID** - Directs the request to the correct IOI device, as in an I/O physical memory address.
- **Address offset within DID** - IOI device location, as in an I/O physical address.

NOTE: This address is not aligned to any particular size in some cases.

Cavium Confidential For Bowen Yang ALI 2014

Constraints for successful IOBDMA execution:

- The appropriate Guest.CvmMemCtl[CVMSEGENA*] / Root.CvmMemCtl[CVMSEGENA*] must be set enabling CMSEG for the mode.
- If in guest-user or guest-supervisor modes, Guest.CvmMemctl[CVMSEGIODIS] must be clear. Violation of this constraint causes a guest-mode address error exception taken on the SD that initiates the IOBDMA operation.
- If in guest-user, guest-supervisor, guest-kernel, root-user, or root-supervisor modes, Root.CvmMemctl[CVMSEGIODIS] must be clear. Violation of this constraint causes a root-mode address error exception taken on the SD that initiates the IOBDMA operation.
- Reserved fields in the CVMSEG I/O store data must be zero.
- The CVMSEG LM addresses for the return data must be valid, i.e.

  effective_scraddr + len ≤ LMEMSZ × 16,

  where LMEMSZ is Root.CvmMemCtl[LMEMSZ] or Guest.CvmMemCtl[LMEMSZ], depending on root or guest mode, respectively.

  Violation of this constraint could cause an address-error exception to occur on the SD that initiates the IOBDMA operation.
- len must be > 0 and ≤ 16. Violation of this constraint could cause an address error exception to occur on the SD that initiates the IOBDMA operation.
- All return data destined for the CVMSEG LM LMTLINE location must not return at any time during the sequence that initiates a wide atomic memory write sequence. (Section 2.23 describes wide atomic memory write operations and the CVMSEG LM LMTLINE location.) Violation of this requirement might cause the wide-atomic memory-write sequence to fail or take an exception.

2.8 Core-Memory Reference Ordering

L2/DRAM stores may be buffered by cnMIPS III cores until a subsequent SYNC, SYNCS, SYNCW, or SYNCWS instruction issues. The cnMIPS III cores may choose to issue these buffered stores into the system memory in any order when there are no SYNC* instructions issued by the cores.

The cnMIPS III core's store-buffering architecture is important for correct functionality and for best performance. Proper use of the cnMIPS III core's synchronization instructions is essential with the cnMIPS III core's store-buffering architecture, and code optimized for cnMIPS III cores may be different from code optimized for other architectures.

For an example of the architectural importance of the cnMIPS III core's store-buffering behavior, consider lock releases. Lock releases are typically ordinary stores to a lock variable indicating that the lock is no longer held. For best performance with multiple cores, this releasing store should typically be made visible to other cores quickly so that the other cores have the opportunity to obtain the lock soon after it is released.

Without a synchronization instruction (typically, a SYNCW or SYNCWS), the cnMIPS III core does not by default make a store visible quickly since it aggressively buffers stores. Thus, for best lock performance on cnMIPS III core, you should

Cavium Confidential For Bowen Yang ALU 02/27/2014

typically follow the releasing-store instruction with a synchronization instruction. This synchronization instruction is not needed with many other memory architectures.

The description of the SYNCW instruction in Section A.3.3 details a similar producer/consumer example. Section 2.10 discusses more about the cnMIPS III core's store-buffering implementation.

The following bullets provide very broad SYNC, SYNCS, SYNCW, SYNCWS, and SYNCIOBDMA descriptions:

- A SYNC instruction orders all operations. A SYNCS instruction is identical to SYNC, except that it does not order marked L2/DRAM store operations. SYNC/SYNCS instructions can be used to order I/O operations as well as memory operations.
- A SYNCW instruction does not order load operations. SYNCW instructions are considerably faster than SYNC or SYNCS since they do not cause the pipeline to stall; subsequent store operations are buffered until the prior store operations are properly ordered instead. A SYNCWS instruction is identical to SYNCW, except that it does not order marked L2/DRAM store operations. SYNCW/SYNCWS instructions can be used to order I/O operations as well as memory operations, and SYNCW/SYNCWS are considerably faster for I/O and memory synchronization.
- SYNCIOBDMA instructions guarantee that prior IOBDMA requests have arrived at the device (and the CVMSEG LM response has been returned) before any subsequent operation can issue. SYNC and SYNCS instructions also enforce the same ordering constraints as SYNCIOBDMA

Note that store operations to I/O physical addresses (i.e. I/O store operations) are posted operations, which means that the CPU considers these operations to be complete when they reach an I/O ordering point for the device. There is no direct synchronization support available to guarantee that one of these posted operations actually reached a given device.

Non-posted operations to I/O physical addresses (i.e. I/O load and IOBDMA operations) also traverse the ordering points for a device. The execution order of operations on the device generally matches the order of entry at the ordering point for the device, except that posted operations may pass non-posted operations.

The I/O ordering points for posted operations are in the coherent memory interconnect (CMI).

The cnMIPS III core differentiates store operations based on the MIPS cacheability and coherency attribute attached to the store operation as follows:

Cacheability and Coherency Attribute	Relevance to L2/DRAM Store Operation
0–6	Unmarked
7	Marked

System performance can be higher when store operations are unordered or marked. The marked/unmarked differentiation is only used for ordering purposes:

- All L2/DRAM load/store operations are ordered by SYNC and SYNCW instructions. Marked L2/DRAM load/store operations are not ordered by SYNCS and SYNCWS instructions, so SYNCS and SYNCWS can be higher performance.

Cavium Confidential For Bowen Yang ATU 02/27/2014

When software must guarantee that posted I/O operations execute before proceeding, it should execute a non-posted I/O operation to the device (waiting for the result if it is an IOBDMA operation). Other cores can flush the posted I/O operations if they are known to have reached the ordering point.

Non-posted I/O operations can be flushed with a SYNC/SYNCS/SYNCIOBDMA operation.

Table 2–31 classifies all CN71XX physical memory addressing operations.

Table 2–31 Physical Memory Addressing Operations

Mnemonic	Description
DLD	A load instruction to a physical address that is L2/DRAM.
ILD	A load instruction to a physical address that is I/O.
UMST	An unmarked L2/DRAM store, ZCB*, or PREF 29/30/31 instruction. UMST SYNC ordering guarantees that the instruction is visible to all readers.
MMST	A marked L2/DRAM store, ZCB*, or PREF 29/30/31 instruction. MMST SYNC ordering guarantees that the instruction is visible to all readers.
IST	An I/O store operation. IST SYNC ordering guarantees that the IST is posted to the ordering point for the device, which generally guarantees that the IST executes prior to any subsequent ILD/IST/DMA to the device, regardless which core sources the ILD/IST/DMA.
DMA	A IOBDMA operation. DMA SYNCW/SYNCWS ordering guarantees that the operation is posted to the ordering point for the device, which generally guarantees that the DMA executes prior to any subsequent ILD/IST/DMA to the device, regardless which core sources the ILD/IST/DMA. DMA SYNC/SYNCS/SYNCIOBDMA ordering guarantees that the operation has completed with the result returned into CVMSEG LM.

Table 2–32 indicates execution ordering rules for any two possible combinations of these physical memory operations, focusing on the ordering guarantees of SYNC, SYNCS, SYNCW, SYNCWS, and SYNCIOBDMA instructions inserted between the two.

Table 2–32 Execution Ordering Rules

First Operation	Second Operation	Execution Ordering Comments
DLD	Any	The second operation cannot appear to execute before the first (DLD) operation, regardless of the presence or absence of SYNC* instructions.
ILD	Any	The second operation cannot appear to execute before the first (ILD) operation both arrives at its ordering point and completes, regardless of the presence or absence of SYNC* instructions.

Cavium Confidential For Bowen Yang ANU 09/27/2014

Table 2–32 Execution Ordering Rules (Continued)

First Operation	Second Operation	Execution Ordering Comments
UMST	DLD	With respect to the local core, the memory value provided to the DLD is always as if all prior local UMST/MMST operations executed in order before the DLD. A SYNC or SYNCS instruction is required between the two operations to order them for other agents. When the SYNC/SYNCS instruction is present, other agents cannot observe the DLD execution prior to the UMST operation.
	ILD	A SYNC or SYNCS instruction is required between the two operations to order them for other agents. When the SYNC/SYNCS instruction is present, other agents cannot observe the ILD instruction to arrive at its ordering point nor execute prior to the UMST operation.
	UMST or MMST	Unless the store operations reference overlapping bytes, a SYNC, SYNCS, SYNCW, or SYNCWS instruction is required between the two operations to order them for other agents. When the SYNC/SYNCS/SYNCW/SYNCWS instruction is present, other agents cannot observe the second (UMST/MMST) operation to execute prior to the first (UMST) operation.
	IST or DMA	A SYNC, SYNCS, SYNCW, or SYNCWS instruction is required between the two operations to order them for other agents. When the SYNC/SYNCS/SYNCW/SYNCWS instruction is present, other agents cannot observe the IST/DMA instruction to arrive at its ordering point nor execute prior to the UMST execution.
MMST	DLD	With respect to the local core, the memory value provided to the DLD is always as if all prior local UMST/MMST operations executed in order before the DLD. A SYNC instruction is required between the two operations to order them for other agents. When the SYNC instruction is present, other agents cannot observe the DLD execution prior to the MMST operation.
	ILD	A SYNC instruction is required between the two operations to order them for other agents. When the SYNC instruction is present, other agents cannot observe the ILD instruction to arrive at its ordering point nor execute prior to the MMST operation.
	UMST or MMST	Unless the store operations reference overlapping bytes, a SYNC or SYNCW instruction is required between the two operations to order them for other agents. When the SYNC/SYNCW instruction is present, other agents cannot observe the second (UMST/MMST) operation to execute prior to the first (MMST) operation.
	IST or DMA	A SYNC or SYNCW instruction is required between the two operations to order them for other agents. When the SYNC/SYNCW instruction is present, other agents cannot observe the IST/DMA instruction to arrive at its ordering point nor execute prior to the MMST execution.

Cavium Confidential Property
Bowen Yang 2014

Table 2–32 Execution Ordering Rules (Continued)

First Operation	Second Operation	Execution Ordering Comments
IST	DLD	A SYNC or SYNCS instruction is required between the two operations to order them for other agents. When the SYNC/SYNCS is present, other agents cannot observe the DLD execution before the IST operation arrives at the ordering point for its device.
	ILD	When the IST and the ILD operations reference the same device, the ILD operation arrives at its ordering point after the IST, and generally executes after the IST, with or without SYNC* instructions. Otherwise, when the IST and ILD reference different devices, other agents cannot observe the ILD to arrive at the ordering point for its device nor execute prior to the IST arriving at the ordering point for its device, with or without SYNC* instructions.
	UMST or MMST	A SYNC, SYNCS, SYNCW, or SYNCWS instruction is required between the two operations to order them for other agents. When the SYNC/SYNCS/SYNCW/SYNCWS is present, other agents cannot observe the UMST/MMST execution before the IST arrives at the ordering point for its device.
	IST or DMA	When the IST and the IST/DMA operations reference the same device, the IST/DMA operation arrives at the ordering point after the IST operation and generally executes after the IST operation, with or without SYNC* instructions. Otherwise, when the IST and IST/DMA operations reference different devices, other agents cannot observe the IST/DMA operation arrive at the ordering point for its device nor execute prior to the IST operation arriving at the ordering point for its device, with or without SYNC* instructions.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 2–32 Execution Ordering Rules (Continued)

First Operation	Second Operation	Execution Ordering Comments
DMA	DLD	A SYNC, SYNCS, or SYNCIOBDMA instruction is required between the two operations to order them. When the SYNC/SYNCS/ SYNCIOBDMA instruction is present, the DMA arrives at its ordering point, executes and its result is returned before the DLD executes.
	ILD	When the DMA and the ILD reference the same device, the ILD operation arrives at the ordering point in order with the DMA, and generally execute after the DMA, with or without SYNC* instructions. Otherwise, when the DMA and ILD reference different devices, other agents cannot observe the ILD to arrive at the ordering point for its device nor execute prior to the DMA arriving at the ordering point for its device, with or without SYNC* instructions. When a SYNC/SYNCS/SYNCIOBDMA operation is present between the two operations, other agents cannot observe the ILD to arrive at its ordering point nor execute before the DMA arrives at its ordering point, executes, and its result is returned.
	UMST or MMST	A SYNCIOBDMA, SYNC, SYNCS, SYNCW, or SYNCWS instruction is required between the two operations to order them for other agents. When a SYNC/SYNCS/SYNCIOBDMA instruction is present, other agents cannot observe the UMST/MMST execution before the DMA arrives at its ordering point, executes, and its result is returned. When a SYNCW/SYNCWS instruction is present, other agents cannot observe the UMST/MMST execution before the DMA operation arrives at the ordering point for its device.
	IST or DMA	When the DMA and the IST/DMA operations reference the same device,the IST/DMA operation arrives at the ordering point in order with the DMA operation, and generally executes after the DMA operation (except that the posted IST operation might pass the DMA operation), with or without SYNC* instructions. Otherwise, when the DMA and IST/DMA operations reference different devices, other agents cannot observe the IST/DMA operation arrive at the ordering point for its device nor execute prior to the DMA arriving at the ordering point for its device, with or without SYNC* instructions. When a SYNC/SYNCS/SYNCIOBDMA is present between the two operations, other agents cannot observe the IST/DMA operation arrive at its ordering point nor execute before the DMA arrives at its ordering point, executes, and its result is returned.

The earlier descriptions in this section, including Tables 2–31 and 2–32, refer to I/O operations destined for the same device as “generally” executed in the order the operation reach the ordering point for the device. Some devices can be accessed through different methods. Specifically, a single device might support a combination of NCB type CSRs, RSL type CSRs, and/or operations. The CN71XX hardware commonly reorders RSL type CSR accesses relative to the other two access methods after the ordering point, so special handling might required to force a particular execution order when mixing RSL type CSR accesses with the other methods. Section 2.9 describes this in more detail.

Cavium Confidential
Boyen Yang For
02/12/24

2.9 CSR Ordering

In the CN71XX design, when the device has CSRs of type RSL, the execution order of the I/O operations might not match the device ordering point order of the operations. This is because CN71XX services RSL-type CSR references indirectly off the IOI, and this indirect mechanism is much slower than direct accesses via the IOI. There is effectively one shared ordering point for all I/O operations to RSL-type CSRs. Later device-direct references may bypass earlier RSL references.

For example, the FPA (described in Chapter 8) has RSL-type CSRs as well as IOI direct FPA operations to free/allocate buffers. Without synchronization, a prior CSR write operation that configures FPA might be delayed until after a subsequent FPA free operation, causing FPA to malfunction. To avoid these malfunctions, software should insert synchronization to force completion of prior RSL-type write operations before issuing the subsequent FPA operations.

Whenever it is necessary for cnMIPS III core software to guarantee completion of prior store operations to RSL-type CSRs before subsequent operations, as in the FPA example just described, the core can perform either of the following operations:

- Issue a load request to any RSL-type CSR
- Issue an IOBDMA request to any RSL-type CSR followed by a SYNCIOBDMA instruction

As CN71XX services all load/store/IOBDMA operations to any RSL-type CSR in the same order that they appear on the IOI, these load/IOBDMA operations flush all prior store/ load/IOBDMA operations. In general, write operations to RSL-type CSRs should be infrequent in CN71XX, and liberal doses of this RSL synchronization should not cause performance problems.

Cavium Confidential For Bowen Yang ALU 02/27/2014

2.10 Core Write Buffer

The behavior of the write buffers in the cnMIPS III cores is important for correct functionality and for best performance. There are 24 write-buffer entries in each core, and each can merge multiple individual stores up to a full 128-byte cache line.

The write buffer can only stall the instruction pipeline when SYNC/SYNCS instructions execute or when there are no write-buffer entries available. SYNCW/SYNCWS instructions never stall the instruction pipeline, and are queued and complete in parallel with instruction issue.

A SYNC/SYNCW instruction forces all write-buffer entries to be sent to the CMI. A SYNCS/SYNCWS instruction does the same, except that marked L2/DRAM write-buffer entries are not sent to the CMI by SYNCS/SYNCWS instructions. All sent write-buffer entries commit before any instruction following a SYNC/SYNCS instruction issues. Following a SYNCW/SYNCWS instruction, the core does not send any subsequently created write-buffer entries to the CMI until after all previously sent write-buffer entries commit. (Note that commit order is, effectively, the store order that is visible to the other cores.)

The write buffer aggressively merges L2/DRAM store operations. It is desirable to merge multiple store operations into one write-buffer entry so fewer write-buffer entries are sent to the CMI. If an L2/DRAM store operation is in the same 128-byte (naturally aligned) cache block, the store operation merges into the existing entry. Otherwise, store operations allocate new write-buffer entries. A write-buffer entry is marked if its merged store operations are all marked. A store operation cannot merge into any write-buffer entry that has already been sent to the CMI.

L2/DRAM store operations normally remain in the write buffer until one of the following occurs:

- A SYNC or SYNCW instruction executes.
- A SYNCS or SYNCWS instruction executes and the write-buffer entry merged at least one unmarked store.
- The write buffer entry times out. CvmMemCtl[WBFLTIME,WBFLTIME2] controls the expiration interval of unmarked L2/DRAM write buffer entries. The expiration interval starts when the write buffer entry is last referenced (unless CvmMemCtl[DISMRGCLRWBTO] is set, when it starts when the write buffer entry is allocated). The timeout interval can be as small as 64 cycles, and as large 500K cycles. (Larger CvmMemCtl[WBFLTIME,WBFLTIME2] values result in larger timeouts.) Marked L2/DRAM write buffer entries use the maximum timeout interval (unless CvmMemCtl[DISMARKWBLONGTO] is set, when the timeout of marked entries is the same as other entries).
- A load/PREF/PREFX instruction to the same cache block misses in the (L1) Dcache.
- An SC/SCD instruction that the cnMIPS III core does not locally fail executes to the same cache block.
- A PREF/PREFX PREPARE_FOR_STORE/ PREPARE_FOR_STORE_THROUGH/NUDGE/NUDGE_WB/EVICT_SOON/ DONT_WRITE_BACK or ZCB/ZCBT instruction to the same cache block executes.

Cavium Confidential For Ruiwen Yang 10/22/2014

- The write buffer entry is ejected to make space for other write-buffer entries. The core ejects blocks to make space when there are more than CvmMemCtl[WBTHRESH,WBFLTIME2] active write-buffer entries.

  Smaller CvmMemCtl[WBTHRESH,WBFLTIME2] values eject blocks sooner, reducing merging but leaving more write-buffer entries available for later use and possibly reducing the cost of subsequent SYNC/SYNCS/SYNCW/SYNCWS instructions.

  Larger CvmMemCtl[WBTHRESH,WBFLTIME2] values eject blocks later, increasing merging but may result in more pipeline stalls because write-buffer entries are not available.

- A memory atomic or store atomic instruction to the same cache block executes.

SC/SCD instructions are treated specially in the write buffer. SC/SCD instructions that do not locally fail evict any matching write-buffer entry that was previously in the write buffer, and create a write-buffer entry that is immediately pushed to the CMI. If the SC/SCD instruction is the successful termination of a wide atomic write instruction, the write-buffer entry includes both the SC/SCD data and the LMTLINE store data. Section 2.23 describes SC/SCD usages and functionality in more detail.

I/O write and IOBDMA operations create write-buffer entries. But they never merge with any other accesses, and are always immediately pushed to the CMI, through the I/O bridge, and onto the IOI. It largely appears as if the I/O write operations and IOBDMA operations bypass the write buffer.

PREF/PREFX NUDGE/EVICT_SOON/DONT_WRITE_BACK, store atomic, and memory atomic operations (see Section A.3) evict any matching write-buffer entry that was previously in the write buffer, then create a new write buffer entry that is immediately pushed to the CMI. It largely appears as if these operations bypass the write buffer.

PREF/PREFX PREPARE_FOR_STORE/PREPARE_FOR_STORE_THROUGH or ZCB/ZCBT operations (see Section A.3) evict any matching write-buffer entry that was previously in the write buffer, then create new write buffer entries like ordinary stores do. But the newly-created write-buffer entry is a full cache block write. Note that PREPARE_FOR_STORE* operations do not update the L1 Dcache, so it is possible for the Dcache copy of the block to become inconsistent with the value that is eventually written into L2/DRAM by the write buffer created for the PREPARE_FOR_STORE*. The specification of PREPARE_FOR_STORE* defines the value of the bytes in the cache block to be unpredictable after the PREPARE_FOR_STORE*, so this is acceptable. If software needs these bytes set to a predictable value (in both the Dcache and the L2/DRAM), it must eventually follow the PREPARE_FOR_STORE* with stores to the bytes of interest in the cache block (or with stores to the cache block by another core or device, which invalidates the Dcache). ZCB/ZCBT operations invalidate a matching data cache block, so immediately zero the block from the local cores perspective.

CvmMemCtl[NOMERGE,ALLSYNCW] are additional debug bits. When CvmMemCtl[NOMERGE] is set, write operations do not merge and are forced immediately to the L2. When both CvmMemCtl[NOMERGE,ALLSYNCW] are set, these stores are additionally forced to commit in order.

PREF/PREFX NUDGE_WB operations (see Section A.3) evict any matching write-buffer entry that was previously in the write buffer. They do not create a new write-buffer entry.

Cavium Confidential For Broadcom ALD Use Only 2/19/14

2.11 Cryptography

cnMIPS III cores can include large multiply, DES/3DES, AES, Galois Field Multiplier (GFM), MD5, SHA-1, SHA-256, SHA-512, KASUMI, SNOW3G, SMS4, CAMELLIA, and ZUC primitive cryptographic acceleration units coupled with their corresponding user-visible processor state.

cnMIPS III cores indicate the availability of these functions via CP0 register fields CvmCtl[NOMUL,NOCRYPTO,KASUMI,SNOW3G,SMS4,CAMELLIA,ZUC].

When these cryptographic functions are currently available in the cnMIPS III core, CvmCtl[NOMUL,NOCRYPTO] are both 0, and CvmCtl[KASUMI,SNOW3G,SMS4,CAMELLIA,ZUC] are all one. When these cryptographic functions are not currently available, CvmCtl[NOMUL,NOCRYPTO] are both 1, and CvmCtl[KASUMI,SNOW3G,SMS4,CAMELLIA,ZUC] are all zero.

CvmCtl[KASUMI,SNOW3G,SMS4,CAMELLIA,ZUC] specifically indicate the availability of KASUMI, SNOW3G, SMS4, CAMELLIA, and ZUC accelerators, respectively. CvmCtl[KASUMI,SNOW3G,SMS4,CAMELLIA,ZUC] always equal the inverse of CvmCtl[NOCRYPTO] on cnMIPS III cores, since cnMIPS III cores include KASUMI, SNOW3G, SMS4, CAMELLIA, and ZUC acceleration whenever cryptographic functions are available.

CN71XX parts are one of two variants:

- Cryptography-enabled (AAP variant): cryptographic functions are available at all times on all cores. (MIO_FUS_DAT2[NOMUL,NOCRYPTO] will both be clear in these chips.)
- Cryptography-disabled (CP variant): cryptographic functions are not available at any time on any core. (MIO_FUS_DAT2[NOMUL,NOCRYPTO] will both be set in these chips.)

When available in the cnMIPS III cores, software can access the cryptographic accelerators and state via standard MIPS coprocessor 2 instructions.

Section 2.2 above describes the user-visible state in cnMIPS III cores. This state is available with cryptographic functions:

- MPL0, MPL1, MPL2, MPL3, MPL4, MPL5, P0, P1, P2, P3, P4, and P5 large multiply state
- 3DESKEY, 3DESIV, and 3DESRESULT DES/3DES and KASUMI state
- AESKEY, AESKEYLEN, AESIV, and AESRESINP AES, SMS4, and CAMELLIA state
- GFMMUL, GFMRESINP, and GFMPOLY GFM state
- HASHDAT and HASHIV MD5, SHA-1, SHA-256, and SHA-512 state

Sections 2.3, 2.4, A.1.2, and A.5.2 describes all the instructions that access the accelerators and state when cryptographic functions are available.

Cavium Confidential For Bowen Yang ALU 02/5/2014

2.12 Floating-Point Unit

cnMIPS™ III introduces a hardware floating-point unit that is fully compatible with the MIPS Revision 5 specification and IEEE 754 standard (1985 version). The floating-point unit and associated instructions are accessed via coprocessor 1. The MIPS specification and IEEE standard allow some implementation choices, which are described below.

2.12.1 Floating Point Modes and Options

Table 2–33 shows the different floating point modes software can configure the cnMIPS™ III cores in. cnMIPS™ III supports true 64-bit floating point mode as well as the paired register mode for backward compatibility.

Table 2–33 cnMIPS™ III cores Floating Point Modes

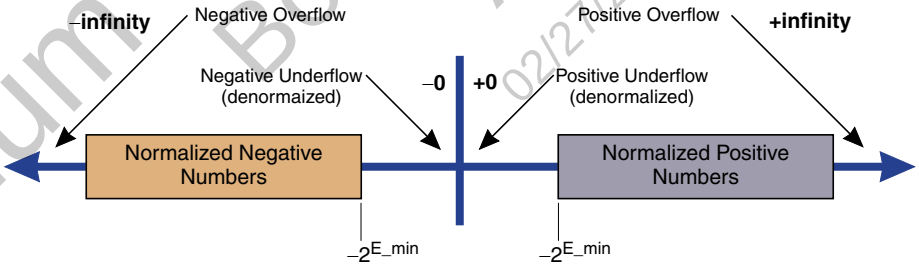
Set By:	Hardware	Hardware	Software	Software
COP0/COP1 state	Config1[FP]	FIR[F64]	Status[CU1]	Status[FR]
COP1 turned off by software	1	1	0	Don't Care
COP1 turned on by software, 32-bit mode with 64-bit FP data types (paired register mode)	1	1	1	0
COP1 turned on by software, true 64-bit mode	1	1	1	1
	Read-only	Read-only	Read/write	Read/write

cnMIPS™ III cores support both single- and double-precision data types, but not paired single data types.

2.12.2 Denormals

Denormalized numbers fill the gap between the 0 and the minimum number representable by IEEE 754 floating-point format (Figure 2–4). Denormalized numbers allow graceful degradation of precision when the floating-point number underflows.

Figure 2–4 Floating-Point Data Types



cnMIPS™ III cores implement denormalized numbers in hardware whenever possible, but rely on floating-point unimplemented traps for cases where it is difficult for the hardware to support denormalized numbers easily.

Cavium Confidential For Bowen Yang ALU 02/27/2014

cnMIPS™ III cores take Unimplemented Floating-Point traps for the following operations when flush to zero (FCSR[FS]) is 0 and at least one of the source operands is a denormal number:

- single- and double-precision multiplication
- single- and double-precision division
- single- and double-precision square root
- single- and double-precision reciprocal
- single- and double-precision reciprocal square root.

cnMIPS™ III cores also raise an unimplemented trap if the sum of unbiased exponents of the operands is less than equal to zero when FCSR[FS] = 0 and at least one of the operands is non-zero.

The MIPS specification treats multiply and add variants (e.g., MADD.S) as if the multiplication and addition were two separate operations (logically). Hence, the multiply portion of the multiply and add variants do not support denormalized source operands in hardware (along with the restrictions on results described above). However, the add portion does support denormalized operands (e.g., if register FR is denormal).

2.12.3 Cause Bits

The COP1 Floating-Point Control and Status Register (FCSR) allows for six cause bits that indicate the source of a floating-point trap.

- Unimplemented (E)
- Invalid (V)
- Divide-by-zero (Z)
- Overflow (O)
- Underflow (U)
- Inexact (I)—

If the Flush-to-zero bit (FCSR[FS]) bit is set, then the Inexact Cause bit has special treatment when traps are not enabled.

Floating-point traps are enabled by the corresponding Enable bits in the same FCSR register. Unimplemented traps cannot be masked and are always taken. If a particular trap is not enabled and the corresponding Cause bit is set, then there is a set of cumulative Flag bits that record the event. Only one Cause bit is set on a floating-point trap based on the following precedence: E, V, Z, O, U, and I. Thus, for example, if E is not, V is enabled and set, then all other Cause bits (Z, O, U, and I) must be zero.

Cavium Confidential For: Edward Yang 8/2/2016 4

The precedence rules and special treatment for various Cause bits are the following:

Unimplemented (E) If the unimplemented cause bit is set, then the hardware clears all other Cause bits.

Invalid (V) If the Invalid Cause bit is set, then the hardware clears Z, O, and U bits (E should already be clear). The Inexact bit is cleared if it was a true inexact operation (i.e. not caused by flush to zero) and Invalid trap was not enabled.

Divide-by-zero (Z) If the Divide-by-zero Cause bit is set, then the hardware clears the O and U bits (E and V should already be clear). The Inexact bit is cleared if it was a true inexact operation (i.e., not caused by flush to zero) and Divide-by-zero trap was not enabled.

Overflow (O) If the Overflow bit is set, then the Underflow bit is cleared (E, V, and Z should already be clear). If the Overflow trap is enabled and both Overflow and Inexact Cause bits are set, then the Overflow bit takes precedence and the hardware clears the Inexact Cause bit.

Underflow (U) If the Underflow Cause bit is set, the E, V, Z, and O should already be clear. If the Underflow trap is enabled and both Underflow and Inexact Cause bits are set, then the Underflow bit takes precedence and the hardware clears the Inexact Cause bit.

Inexact (I) The Inexact Cause bit is set either because of an inexact result or if the result is flushed to zero when FCSR[FS] = 1. The rules for clearing the Inexact Cause bit are described above.

2.12.4 Precise Floating Point Traps

Floating-point traps are always precise, but when floating-point traps are enabled, certain floating-point operations slow down. The following two rules apply:

1. If a floating-point instruction is the first instruction in program order to issue in a cycle, then the hardware cannot issue any another instruction in the same cycle.

2. In the three cycles following a floating-point instruction, only the following instructions can be issued by hardware:
   - integer add instructions
   - integer logical instructions
   - integer shift instructions
   - integer conditional move instructions
   - integer branch instructions
   - integer jump instructions
   - floating-point instructions

   No load or store instructions (neither integer nor floating point) can be issued.

Cavium Confidential For Bowen Yang ALU 02/27/2014

2.13 Core Error Handling

cnMIPS™ III cores have extended cnMIPS™ II's error handling features to allow privileged software to better control how errors in the chip are controlled and exercised. Six structures within the cnMIPS™ III have error protection (Table 2–34). Each structure with error protection has four sets of control and status bits: the Log bit (captures if there was an error), the Disable bit (allows software to disable error protection), No Software Invoked bit (allows software error handler to be not invoked on a correctable error), and Force Error bit (allows software to inject an error into the structure to invoke the hardware and software error flow). The Error Handler specifies which error handler is called when the No Software Invoked bit is set to 0.

Table 2–34 Core Error Handling Mechanisms

Structure	Error Protection	Control and Status Bits				Error Handler
		Log	Disable	No Software Invoked (for correctable error)	Force Error	
L1 Instruction Cache	Parity (recovers)	Icache CacheErr[0]	Icache CacheErr[18]	Icache CacheErr[17]	Icache CacheErr[31]	Cache-error exception handler
L1 Data Cache	Parity (recovers)	ErrCtl[0]	ErrCtl[21]	ErrCtl[17]	ErrCtl[11]	Cache-error exception handler
µTLB	Parity (recovers)	ErrCtl[16]	CvmMemCtl2[1]	ErrCtl[18]	Not needed, software can directly set parity bits in µTLB debug mode	Cache-error exception handler
TLB	Parity (recovers)	ErrCtl[14]	CvmMemCtl[44]	Not applicable, TLB is software managed	ErrCtl[15]	General exception handler with ExcCode = 24 (machine check)
Miss Address File (MAF) Load Tags	Parity (recovers)	ErrCtl[19]	ErrCtl[20]	ErrCtl[25]	ErrCtl[24]	General exception handler with ExcCode = 7 (bus error)
Write Buffer	SECDED ECC	ErrCtl[9]	ErrCtl[22]	ErrCtl[23]	ErrCtl[12]: single bit ErrCtl[13]: double bit	General exception handler with ExcCode = 7 (bus error)

Cavium Confidential For Bowen Jiang ALU 02/27/2014

2.14 Virtualization

The cnMIPS™ III processor implements the MIPS architecture specification for processor virtualization. In the MIPS architecture, the “onion model” provides a guest context and a root context. A guest operating system would typically execute in the guest context; while a hypervisor would typically execute in the root context.

Most MIPS processor registers in the coprocessor 0 root context are duplicated in the guest to provide separate contexts for guest and root software. The CP0 registers in root mode are defined in Section 2.15.1; the CP0 registers in guest mode are defined in Section 2.15.2. The Cavium-specific CP0 registers are defined in Section 2.15.3 and Section 2.15.4.

A root context includes all the traditional CP0 state associated with the MIPS architecture and some additional state to support virtualization. Root CP0 registers are only visible to the root mode software and not visible to guest mode software.

The guest CP0 context consists of a subset of the registers available in the root CP0 context. All guest registers are visible to root software though D/MTGC0 and D/MFGC0 instructions. Guest registers may have access permissions different from root when accessed by the guest. From the guest’s perspective, registers may be as follows:

- Read only
- Read/write
- Read only and write generates a Guest Privileged Sensitive Instruction Exception to the root
- Read or write generate a Guest Privileged Sensitive Instruction Exception to the root.

In addition, the behavior of some CP0 registers in the guest may differ from the same CP0 registers in the root. A few CP0 registers, such as those associated with the watch exception and performance counting, may be assigned to either guest or root context by software executing in the root context.

Cavium Confidential For Bowen Yang ALU 02/27/2014

2.15 Coprocessor 0 Privileged Registers

MIPS CP0 is implemented as follows:

- 256-entry TLB
- SEGBITS = PABITS = 49.
- EJTAG is implemented.
- CN71XX maintains a virtual Icache.

Section 2.15.3 details the Coprocessor 0 registers in the cnMIPS™ III cores.

Table 2–35 Coprocessor 0 Register Summary: Root Context

Reg. #	Select	Register	Implemented/ Not implemented	L2C_COP0_ADR/ L2C_COP0_DAT Access	Comments	
0	0	Root.Index	Implemented	—	See page 107.	
0	0	Root.UtlbIndex	Implemented (Cavium-Specific)	—	When CvmMemCtl2[UTLBRW] is set, access to Index register is substituted with access to UtlbIndex. See page 152.	**cnMIPS III** only
0	1	MVPControl	Not Implemented	—	MIPS MT ASE not implemented.	
0	2	MVPConf0	Not Implemented	—	MIPS MT ASE not implemented.	
0	3	MVPConf1	Not Implemented	—	MIPS MT ASE not implemented.	
1	0	Root.Random	Implemented	—	See page 107.	
1	1	VPEControl	Not Implemented	—	MIPS MT ASE not implemented.	
1	2	VPEConf0	Not Implemented	—	MIPS MT ASE not implemented.	
1	3	VPEConf1	Not Implemented	—	MIPS MT ASE not implemented.	
1	4	YQMask	Not Implemented	—	MIPS MT ASE not implemented.	
1	5	VPESchedule	Not Implemented	—	MIPS MT ASE not implemented.	
1	6	VPEScheFBack	Not Implemented	—	MIPS MT ASE not implemented.	
1	7	VPEOpt	Not Implemented	—	MIPS MT ASE not implemented.	
2	0	Root.EntryLo0	Implemented	—	See page 108.	
2	0	Root.UtlbEntry1	Implemented (Cavium-specific)	—	When CvmMemCtl2[UTLBRW] = 1, access to EntryLo0 register is substituted with access to 1. See page 152.	**cnMIPS III** only
2	1	TCStatus	Not Implemented	—	MIPS MT ASE not implemented.	
2	2	TCBind	Not Implemented	—	MIPS MT ASE not implemented.	
2	3	TCRestart	Not Implemented	—	MIPS MT ASE not implemented.	
2	4	TCHalt	Not Implemented	—	MIPS MT ASE not implemented.	
2	5	TCContext	Not Implemented	—	MIPS MT ASE not implemented.	
2	6	TCSchedule	Not Implemented	—	MIPS MT ASE not implemented.	
2	7	TCScheFBack	Not Implemented	—	MIPS MT ASE not implemented.	
3	0	Root.EntryLo1	Implemented	—	See page 108.	
3	0	Root.UtlbEntry2	Implemented (Cavium-specific)	—	See page 152.	**cnMIPS III** only
3	7	TCOpt	Not Implemented	—	MIPS MT ASE not implemented.	
4	0	Root.Context	Implemented	—	See page 108.	
4	1	ContextConfig	Not Implemented	—		

Cavium Confidential For Bowen Yang ALD only

Table 2–35 Coprocessor 0 Register Summary: Root Context (Continued)

Reg. #	Select	Register	Implemented/ Not implemented	L2C_COP0_ADR/ L2C_COP0_DAT Access	Comments	
4	2	Root.UserLocal	Implemented	RO	See page 109. Read by RDHWR 29.	
4	3	XContextConfig	Not Implemented	—		
5	0	Root.PageMask	Implemented	—	See page 109.	
5	1	Root.PageGrain	Implemented	—	See page 110.	
6	0	Root.Wired	Implemented	—	See page 110.	
6	1	SRSConf0	Not Implemented	—	SmartMIPS ASE not implemented.	
6	2	SRSConf1	Not Implemented	—	SmartMIPS ASE not implemented.	
6	3	SRSConf2	Not Implemented	—	SmartMIPS ASE not implemented.	
6	4	SRSConf3	Not Implemented	—	SmartMIPS ASE not implemented.	
6	5	SRSConf4	Not Implemented	—	SmartMIPS ASE not implemented.	
7	0	Root.HWREna	Implemented	RO	See page 110.	
8	0	Root.BadVAddr	Implemented	—	See page 111.	
8	1	Root.BadInstr	Implemented		See page 111.	**cnMIPS III** only
8	2	Root.BadInstrP	Implemented		See page 111.	**cnMIPS III** only
9	0	Root.Count	Implemented	RO	See page 111.	
9	6	Root.CvmCount	Implemented (Cavium-specific)	RO	A 64-bit cycle counter. See page 169. Read by RDHWR 31.	
9	7	Root.CvmCtl	Implemented (Cavium-specific)	RO	Controls the Icache and instruction issue. See page 154.	
10	0	Root.EntryHi	Implemented	—	See page 112.	
10	0	Root.UtlbEntry0	Implemented (Cavium-specific)	—	When CvmMemCtl2[UTLBRW] is set, access to EntryHi register is substituted with access to UtlbEntry0. See page 156.	**cnMIPS III** only
10	4	GuestCtl1	Not implemented	—	Guest ID not implemented.	
10	5	Root.GuestCtl2	Implemented		See page 112.	**cnMIPS III** only
11	0	Root.Compare	Implemented	RO	See page 112.	
11	4	Root.GuestCtl0Ext	Implemented	RO	See page 113.	**cnMIPS III**
11	6	Root.PowThrottle	Implemented (Cavium-specific)	R/W	See page 157.	
11	7	Root.CvmMemCtl	Implemented (Cavium-specific)	—	Controls the cores' memory system. See page 159.	Updated for **cnMIPS III**
12	0	Root.Status	Implemented	RO	See page 114.	
12	1	Root.IntCtl	Implemented	RO	See page 115.	
12	2	Root.SRSCtl	Implemented	RO	See page 115.	
12	3	SRSMap	Not Implemented	—	No shadow register set.	
12	4	View_IPL	Not Implemented	—	MIPS MCU ASE not implemented.	
12	5	SRSMap2	Not Implemented	—	MIPS MCU ASE not implemented.	
12	6	Root.GuestCtl0	Implemented		See page 115.	**cnMIPS III** only
12	7	Root.GTOffset	Implemented		See page 117	**cnMIPS III** only

Cavium Confidential For Bowen Yang ALU 01/27/2014

Table 2–35 Coprocessor 0 Register Summary: Root Context (Continued)

Reg. #	Select	Register	Implemented/ Not implemented	L2C_COP0_ADR/ L2C_COP0_DAT Access	Comments	
13	0	Root.Cause	Implemented	RO	See page 118.	
13	4	View_RIPL	Not Implemented	—	MIPS MCU ASE not implemented.	
14	0	Root.EPC	Implemented	RO	See page 118.	
15	0	Root.PRId	Implemented	RO	See page 119.	
15	1	Root.Ebase	Implemented	RO	See page 119.	
15	2	CDMMBase	Not Implemented	—		
15	3	CMGCRBase	Not Implemented	—		
16	0	Root.Config	Implemented	—	See page 119.	
		Root.PCDebug	Implemented (Cavium-Specific) for L2C_COP0_ADR/ L2C_COP0_DAT Access	RO	Read-only program counter debug information that can only be accessed via L2C_COP0_ADR/ L2C_COP0_DAT CSR read operations. See page 162.	
16	1	Root.Config1	Implemented	—	See page 120.	
		Root.PCDebug1	Implemented (Cavium-Specific) for L2C_COP0_ADR/ L2C_COP0_DAT Access	RO	Read-only program counter debug information that can only be accessed via L2C_COP0_ADR/ L2C_COP0_DAT CSR read operations. See page 163.	
16	2	Root.Config2	Implemented	—	See page 120.	
		Root.PCDebug2	Implemented (Cavium-Specific) for L2C_COP0_ADR/ L2C_COP0_DAT Access	RO	Read-only program counter debug information that can only be accessed via L2C_COP0_ADR/ L2C_COP0_DAT CSR read operations. See page 163.	
16	3	Root.Config3	Implemented	—	See page 121.	
		Root.PCDebug3	Implemented (Cavium-Specific) for L2C_COP0_ADR/ L2C_COP0_DAT Access	RO	Read-only program counter debug information that can only be accessed via L2C_COP0_ADR/ L2C_COP0_DAT CSR read operations. See page 164.	
16	4	Root.Config4	Implemented	—	See page 122.	
		Root.PCDebug4	Implemented (Cavium-Specific) for L2C_COP0_ADR/ L2C_COP0_DAT Access	RO	Read-only program counter debug information that can only be accessed via L2C_COP0_ADR/ L2C_COP0_DAT CSR read operations. See page 164.	
16	5	Root.Config5	Implemented (Cavium-Specific)	—	See page 122.	**cnMIPS III** only
16	6	Root.CvmMemCtl2	Implemented	—	Controls the cores' memory system. Extension to CvmMemCtl register. See page 165	**cnMIPS III** only
16	7	Root.CvmVMConfig	Implemented (Cavium-Specific)	—	See page 167.	**cnMIPS III** only
17	0	LLAddr	Not Implemented	—		
18	0,1	Root.WatchLo	Two Implemented	WatchLo0 RO	One instruction-only, one data-only. See page 123.	
19	0,1	Root.WatchHi	Two Implemented	WatchHi0 RO	One instruction-only, one data-only. See page 123.	

Cavium Confidential For Bowen Yang AU 02/24/2017

Table 2–35 Coprocessor 0 Register Summary: Root Context (Continued)

Reg. #	Select	Register	Implemented/ Not implemented	L2C_COP0_ADR/ L2C_COP0_DAT Access	Comments	
20	0	Root.XContext	Implemented	—	See page 124.	
22	0	Root.Multicore Debug	Implemented (Cavium-specific)	RO	See page 169. Used for multicore debugging. See "Hardware Debug Features" on page 206 for more details.	Updated for **cnMIPS III**
22	2	Root.CvmCountOffset	Implemented (Cavium-specific)		See page 171.	**cnMIPS III** only
23	0	Root.Debug	Implemented	RO	See page 125.	
23	1	TraceControl	Not implemented	—	PDtrace not implemented.	
23	2	TraceControl2	Not implemented	—	PDtrace not implemented.	
23	3	UserTraceData1	Not implemented	—	PDtrace not implemented.	
23	4	TraceIBPC	Not Implemented	—	PDtrace not implemented.	
23	5	TraceDBPC	Not Implemented	—	PDtrace not implemented.	
23	6	Root.Debug2	Implemented	RO	See page 125.	
24	0	Root.DEPC	Implemented	RO	See page 126.	
24	2	TraceControl3	Not Implemented	—	PDtrace not implemented.	
24	3	UserTraceData2	Not Implemented	—	PDtrace not implemented.	
25	0,1,2,3, 4,5,6,7	Root.PerfCnt	Four Implemented	RO	Eight total registers. See page 126.	Select [4,5,6,7] **cnMIPS III** only
26	0	Root.ErrCtl	Implemented	—	Implemented to latch errors and control how/whether software is informed of hardware errors. See page 171.	**cnMIPS III** only
27	0	Root.CacheErr	Implemented (Cavium-specific)	RO	Implemented to latch Icache error information. See page 172.	Updated for **cnMIPS III**
27	1			—	Implemented to latch Dcache error information. See page 173.	Updated for **cnMIPS III**
27	2	Root.IcacheDebug	Implemented (Cavium-specific	RO	Implemented to latch Icache BIST error information. See page 173.	**cnMIPS III** only
28	0	Root.TagLo	Implemented (Cavium-specific)	RO	(Icache) Implemented. See page 173.	Updated for **cnMIPS III**
28	2			—	(Dcache) Implemented. See page 174.	
28	1	Root.DataLo	Implemented (Cavium-specific)	RO	(Icache) Implemented. See page 175.	
28	3			—	(Dcache) Implemented. See page 175.	
28	4	Root.TagLo1	Implemented (Cavium-specific)	RO	(Dcache) Implemented. See page 174.	

Cavium Confidential For Bowen Yang AUG 07/21/2014

Table 2–35 Coprocessor 0 Register Summary: Root Context (Continued)

Reg. #	Select	Register	Implemented/ Not implemented	L2C_COP0_ADR/ L2C_COP0_DAT Access	Comments	
29	0	Root.TagHi	Implemented (Cavium-specific)	RO	(Icache) implemented. See page 176.	Updated for **cnMIPS III**
29	2			—	(Dcache) implemented. See page 176.	Updated for **cnMIPS III**
29	1	Root.DataHi	Implemented (Cavium-specific)	RO	(Icache) implemented. See page 175	
29	3			—	(Dcache) implemented. See page 176	
30	0	Root.ErrorEPC	Implemented	RO	See page 129.	
31	0	Root.DESAVE	Implemented	RO	See page 129.	
31	2	Root.KScratch1	Implemented	RO	See page 130.	
31	3	Root.KScratch2	Implemented	RO	See page 130.	
31	4	Root.KScratch3	Implemented (Cavium-specific)	RO	See page 130.	
31	5	Root.KScratch4	Implemented	RO	See page 130	**cnMIPS III** only

Table 2–36 Coprocessor 0 Register Summary: Guest Context

Register Number	Select	Register	Implemented/ Not implemented	L2C_COP0_ADR/ L2C_COP0_DAT* Access	Comments	
0	0	Guest.Index	Implemented	—	See page 131	**cnMIPS III** only
0	1	MVPControl	Not Implemented	—	MIPS MT ASE not implemented.	
0	2	MVPConf0	Not Implemented	—	MIPS MT ASE not implemented.	
0	3	MVPConf1	Not Implemented	—	MIPS MT ASE not implemented.	
1	0	Guest.Random	Implemented	—	See page 131.	**cnMIPS III** only
1	1	VPEControl	Not Implemented	—	MIPS MT ASE not implemented.	
1	2	VPEConf0	Not Implemented	—	MIPS MT ASE not implemented.	
1	3	VPEConf1	Not Implemented	—	MIPS MT ASE not implemented.	
1	4	YQMask	Not Implemented	—	MIPS MT ASE not implemented.	
1	5	VPESchedule	Not Implemented	—	MIPS MT ASE not implemented.	
1	6	VPEScheFBack	Not Implemented	—	MIPS MT ASE not implemented.	
1	7	VPEOpt	Not Implemented	—	MIPS MT ASE not implemented.	
2	0	Guest.EntryLo0	Implemented	—	See page 132.	**cnMIPS III** only
2	1	TCStatus	Not Implemented	—	MIPS MT ASE not implemented.	
2	2	TCBind	Not Implemented	—	MIPS MT ASE not implemented.	
2	3	TCRestart	Not Implemented	—	MIPS MT ASE not implemented.	
2	4	TCHalt	Not Implemented	—	MIPS MT ASE not implemented.	
2	5	TCContext	Not Implemented	—	MIPS MT ASE not implemented.	
2	6	TCSchedule	Not Implemented	—	MIPS MT ASE not implemented.	

Cavium Confidential For Bowen Yang ALU 04/27/2014

Table 2–36 Coprocessor 0 Register Summary: Guest Context (Continued)

Register Number	Select	Register	Implemented/ Not implemented	L2C_COP0_ADR/ L2C_COP0_DAT* Access	Comments	
2	7	TCScheFBack	Not Implemented	—	MIPS MT ASE not implemented.	
3	0	Guest.EntryLo1	Implemented	—	See page 132.	**cnMIPS III** only
3	7	TCOpt	Not Implemented	—	MIPS MT ASE not implemented.	
4	0	Guest.Context	Implemented	—	See page 133.	**cnMIPS III** only
4	1	ContextConfig	Not Implemented	—		
4	2	Guest.UserLocal	Implemented	RO	See page 133. Read by RDHWR 29.	**cnMIPS III** only
4	3	XContextConfig	Not Implemented	—		
5	0	Guest.PageMask	Implemented	—	See page 133.	**cnMIPS III** only
5	1	Guest.PageGrain	Implemented	—	See page 134.	**cnMIPS III** only
6	0	Guest.Wired	Implemented	—	See page 134.	**cnMIPS III** only
6	1	SRSConf0	Not Implemented	—	SmartMIPS ASE not implemented.	
6	2	SRSConf1	Not Implemented	—	SmartMIPS ASE not implemented.	
6	3	SRSConf2	Not Implemented	—	SmartMIPS ASE not implemented.	
6	4	SRSConf3	Not Implemented	—	SmartMIPS ASE not implemented.	
6	5	SRSConf4	Not Implemented	—	SmartMIPS ASE not implemented.	
7	0	Guest.HWREna	Implemented	RO	See page 134.	**cnMIPS III** only
8	0	Guest.BadVAddr	Implemented	—	See page 135.	**cnMIPS III** only
8	1	Guest.BadInstr	Implemented		See page 135.	**cnMIPS III** only
8	2	Guest.BadInstrP	Implemented		See page 135.	**cnMIPS III** only
9	0	Guest.Count	Implemented	RO	See page 136.	**cnMIPS III** only
9	6	Guest.CvmCount	Implemented (Cavium-specific)	RO	A 64-bit cycle counter. See page 177. Read by RDHWR 31.	**cnMIPS III** only
9	7	Guest.CvmCtl	Implemented (Cavium-specific)	RO	Controls the Icache and instruction issue. See page 177.	**cnMIPS III** only
10	0	Guest.EntryHi	Implemented	—	See page 136.	**cnMIPS III** only
11	0	Guest.Compare	Implemented	RO	See page 136.	**cnMIPS III** only
11	7	Guest.CvmMem Ctl	Implemented (Cavium-specific)	—	Controls the cores' memory system. See page 180.	Updated for **cnMIPS III**
12	0	Guest.Status	Implemented	RO	See page 137.	**cnMIPS III** only
12	1	Guest.IntCtl	Implemented	RO	See page 139.	**cnMIPS III** only
12	2	Guest.SRSCtl	Implemented	RO	See page 139.	**cnMIPS III** only
12	3	SRSMap	Not Implemented	—	No shadow register set.	
12	4	View_IPL	Not Implemented	—	MIPS MCU ASE not implemented.	
12	5	SRSMap2	Not Implemented	—	MIPS MCU ASE not implemented.	
13	0	Guest.Cause	Implemented	RO	See page 140.	**cnMIPS III** only

Cavium Confidential For Bowen Yang ALU 02/27/2014

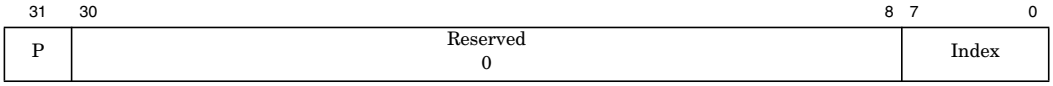
Table 2–36 Coprocessor 0 Register Summary: Guest Context (Continued)

Register Number	Select	Register	Implemented/ Not implemented	L2C_COP0_ADR/ L2C_COP0_DAT* Access	Comments	
13	4	View_RIPL	Not Implemented	—	MIPS MCU ASE not implemented.	
14	0	Guest.EPC	Implemented	RO	See page 141.	**cnMIPS III** only
15	0	Guest.PRId	Implemented	RO	See page 141.	**cnMIPS III** only
15	1	Guest.Ebase	Implemented	RO	See page 141.	**cnMIPS III** only
15	2	CDMMBase	Not Implemented	—		
15	3	CMGCRBase	Not Implemented	—		
16	0	Guest.Config	Implemented	—	See page 142.	**cnMIPS III** only
16	1	Guest.Config1	Implemented	—	See page 142.	**cnMIPS III** only
16	2	Guest.Config2	Implemented	—	See page 143.	**cnMIPS III** only
16	3	Guest.Config3	Implemented	—	See page 144.	**cnMIPS III** only
16	4	Guest.Config4	Implemented	—	See page 145.	**cnMIPS III** only
16	5	Guest.Config5	Implemented (Cavium-Specific)	—	See page 145.	**cnMIPS III** only
16	6	Guest.CvmMem Ctl2	Implemented	—	Controls the cores' memory system. Extension to CvmMemCtl register. See page 184.	**cnMIPS III** only
17	0	LLAddr	Not Implemented	—		
18	0,1	Guest.WatchLo	Two Implemented	WatchLo0 RO	One instruction-only, one data-only. See page 146.	**cnMIPS III** only
19	0,1	Guest.WatchHi	Two Implemented	WatchHi0 RO	One instruction-only, one data-only. See page 146.	**cnMIPS III** only
20	0	Guest.XContext	Implemented	—	See page 147.	**cnMIPS III** only
25	0,1,2,3, 4,5,6,7	Guest.PerfCnt	Four Implemented	RO	Eight total registers. See page 147.	Select [4,5,6,7] **cnMIPS III** only
30	0	Guest.ErrorEPC	Implemented	RO		**cnMIPS III** only
31	2	Guest.KScratch1	Implemented	RO	See page 151.	**cnMIPS III** only
31	3	Guest.KScratch2	Implemented	RO	See page 151.	**cnMIPS III** only
31	4	Guest.KScratch3	Implemented (Cavium-specific)	RO	See page 151.	**cnMIPS III** only

Cavium Confidential For Bowen Yang ALU 02/27/2014

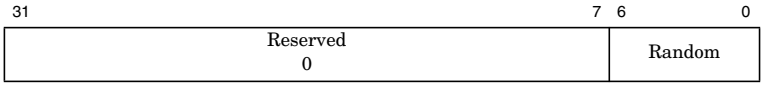
2.15.1 Coprocessor 0 Privileged Register in Root Mode

Root.Index Register: CP0 Register 0, Select 0



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	P	RO	0	Probe Failure. Hardware writes this bit during execution of the RTLBP instruction to indicate whether a RTLB match occurred.
<30:8>	—	RAZ	0x0	Reserved
<7:0>	Index	R/W	—	RTLB index. Software writes this field to provide the index to the RTLB entry referenced by the RTLBR and RTLBWI instructions.

Root.Random Register: CP0 Register 1, Select 0

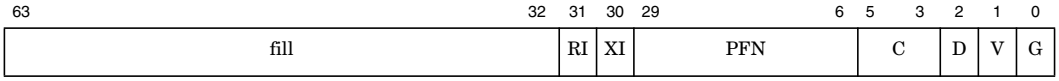


Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:8>	—	RAZ	0x0	Reserved
<7:0>	Random	RO	TLB Entries–1	TLB random index

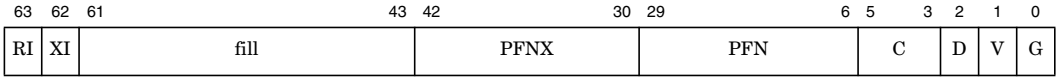
Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Root.EntryLo0, Root.EntryLo1 Registers: CP0 Registers 2 and 3, Select 0

When Root.PageGrain[ELPA] = 0:

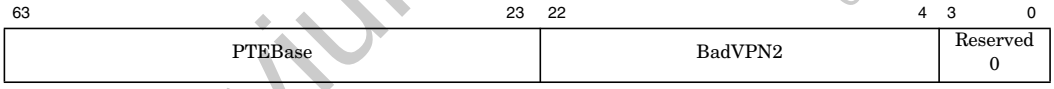


When Root.PageGrain[ELPA] = 1:



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:32>/ <61:43>	fill	RO	0x0	Ignored on write, return 0x0 on read.
<31>/ <63>	RI	R/W	0	Read inhibit. If this bit is set to 1 in a TLB entry, any load instruction that reads data on the virtual page causes a TLB invalid exception, even if the V (valid) bit is set. This bit is writable only if PageGrain[RIE] = 1. If PageGrain[RIE] = 0, this bit is set to 0 on any write to the EntryLo0/1 register, regardless of the value written.
<30>/ <62>	XI	R/W	0	Execute inhibit. If this bit is set to 1 in a TLB entry, any attempt to fetch an instruction causes a TLB invalid exception, even if the V (valid) bit is set. This bit is writable only if PageGrain[XIE] = 1. If PageGrain[XIE] = 0, this bit is set to 0 on any write to the EntryLo0/1 register, regardless of the value written.
<42:30>	PFNX	R/W	0x0	When PageGrain[ELPA] = 1, the hardware uses the full address width, with PFNX expanding the PFN field to the full physical address. When PageGrain[ELPA] = 0, the PFNX is always written to 0x0 on a mtc0/dmtc0 operation to EntryLo0/1. Otherwise, the hardware uses the full address width in all cases. See "Root.PageGrain Register: CP0 Register 5, Select 1" on page 110. • cnMIPS™ III cores implement a 49-bit physical address requiring a 13-bit PFNX field. • cnMIPS™ III cores do not support 1KB pages.
<29:6>	PFN	R/W	—	Page frame number
<5:3>	C	R/W	—	Coherency attribute of page. Refer to Section 2.8.
<2>	D	R/W	—	Dirty bit (indicates page is writable).
<1>	V	R/W	—	Valid bit
<0>	G	R/W	—	Global bit

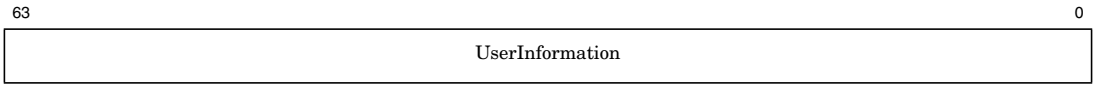
Root.Context Register: CP0 Register 4, Select 0



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:23>	PTEBase	R/W	—	This field is for use by the operating system and is normally written with a value that allows the operating system to use the Context Register as a pointer into the current PTE array in memory.
<22:4>	BadVPN2	RO	0x0	Bad Virtual Page Number/2 field is written by hardware on a miss.
<3:0>	—	RAZ	0x0	Reserved

Cavium Confidential For Bowen Yang ALU 02/27/2014

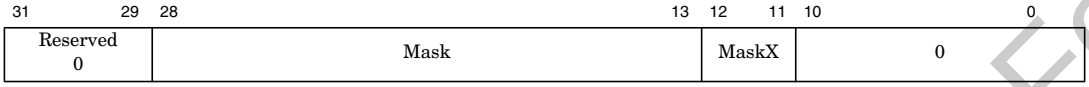
Root.UserLocal Register: CP0 Register 4, Select 2



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:0>	UserInformation	R/W	—	This field contains software information that is not interpreted by the hardware.

Root.PageMask Register: CP0 Register 5, Select 0

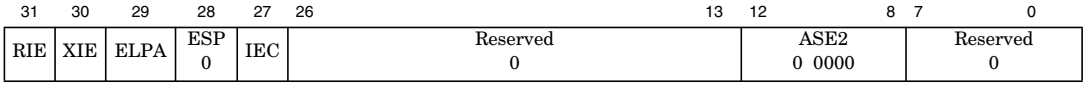
The PageMask register is a read/write register used for reading from and writing to the TLB.



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:29>	—	RAZ	0x0	Reserved. Ignored on write; read as zero.
<28:13>	Mask	R/W	—	The Mask field is a bit mask in which a 1 bit indicates that the corresponding bit of the virtual address should not participate in the TLB match. CN71XX supports the full mask range, and supports all page sizes 4KB, 8KB, 16KB, ... 64MB, 128MB, and 256MB.
<12:11>	MaskX	RO	0x0	CN71XX does not support 1KB pages, so MaskX is not used (write operations are ignored, read operations are 0x0).
<10:0>	—	RAZ	0x0	Reserved. Ignored on write; read as zero.

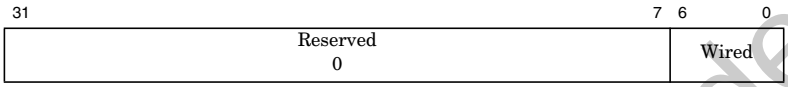
Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Root.PageGrain Register: CP0 Register 5, Select 1



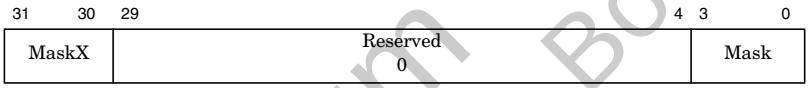
Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	RIE	R/W	0	Read inhibit enable. When set to 1, EntryLo0/1[RI] is enabled. When cleared to 0, EntryLo0/1[RI] is disabled and is not writable by software.
<30>	XIE	R/W	0	Execute inhibit enable. When set to 1, EntryLo0/1[XI] is enabled. When cleared to 0, EntryLo0/1[XI] is disabled and is not writable by software.
<29>	ELPA	R/W	0	Enables support for large physical addresses. As PABITS = 49 for CN71XX, ELPA should normally be enabled on CN71XX. See "Root.EntryLo0, Root.EntryLo1 Registers: CP0 Registers 2 and 3, Select 0" on page 108 for description for the only effect of PageGrain[ELPA].
<28>	ESP	RAZ	0	1KB pages not implemented.
<27>	IEC	R/W	0	Enables unique exception codes for the Read-Inhibit and Execute-Inhibit exceptions.
<28:13>	—	RAZ	0x0	Reserved
<12:8>	ASE	RAZ	0x0	ASE not implemented.
<7:0>	—	RAZ	0x0	Reserved

Root.Wired Register: CP0 Register 6, Select 0



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:8>	—	RAZ	0x0	Reserved
<7:0>	Wired	R/W	0x0	TLB wired boundary.

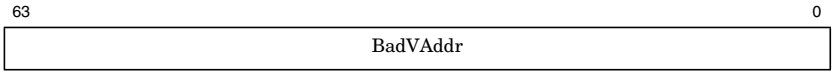
Root.HWREna Register: CP0 Register 7, Select 0



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:30>	MaskX	R/W	0x0	cnMIPS™ II cores include Cavium-specific instruction RDHWR 31, 30, so a MaskX field exists at <31:30> to enable them. Refer to Section A.4 for the RDHWR description.
<29>	Mask29	R/W	0	Enables access by the RDHWR instruction to the UserLocal register.
<28:4>	—	RAZ	0x0	Reserved
<3:0>	Mask	R/W	0x0	Each bit in this field enables access by the RDHWR instruction to a particular hardware register.

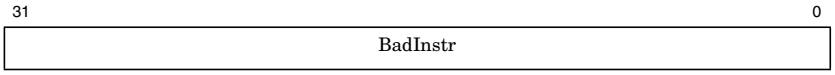
Cavium Confidential For Bowen Yang ALU 02/27/2014

Root.BadVAddr Register: CP0 Register 8, Select 0



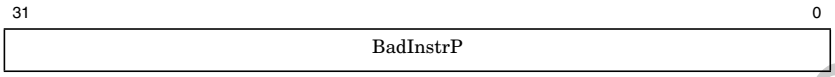
Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:0>	BadVAddr	RO	0x0	Bad virtual address

Root.BadInstr Register: CP0 Register 8, Select 1



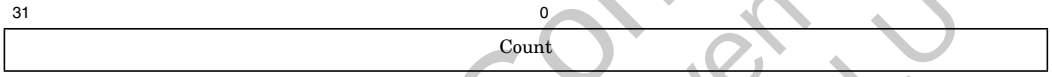
Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:0>	BadInstr	RO	—	Contains the instruction that caused an exception.

Root.BadInstrP Register: CP0 Register 8, Select 2



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:0>	BadInstrP	RO	—	Contains the instruction prior to the one that encountered an exception.

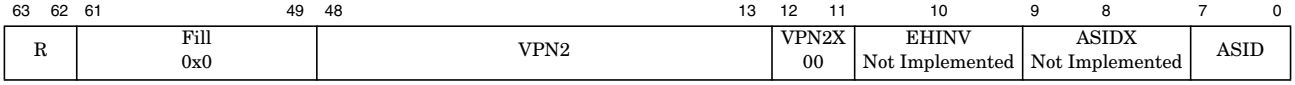
Root.Count Register: CP0 Register 9, Select 0



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:0>	Count	R/W	0x0	Interval counter. The Count register is zeroed during reset and increments by one every cycle.

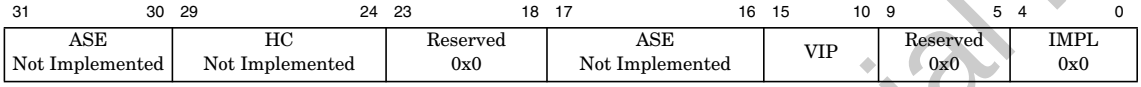
Cavium Confidential For Bowen Yang ALU 02/2/2014

Root.EntryHi Register: CP0 Register 10, Select 0



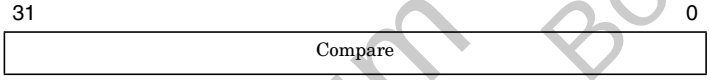
Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:62>	R	R/W	—	Virtual memory region
<61:49>	Fill	RAZ	0x0	Fill bits reserved for expansion of the virtual address space.
<48:13>	VPN2	R/W	—	This field is written by hardware on a TLB exception or on a TLB read.
<12:11>	VPN2X	RAZ	0x0	Not used. 1KB pages not implemented.
<10>	EHINV	RAZ	0	TLB invalidation. Not implemented.
<9:8>	ASIDX	RAZ	0x0	Address space identifier extended. Not implemented.
<7:0>	ASID	R/W	—	Address space identifier

Root.GuestCtl2 Register: CP0 Register 10, Select 5



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:30>	ASE	RO	0x0	MCU module extension for HC. Not implemented.
<29:24>	HC	RO	0x0	Hardware clear for VIP. Not implemented.
<23:18>	—	RAZ	0x0	Reserved
<17:16>	ASE	RO	0x0	MCU module extension for VIP. Not implemented.
<15:10>	VIP	R/W	0x0	Virtual interrupt pending. Corresponds to Guest.Cause[IP<7:2>].
<9:5>	—	RAZ	0x0	Reserved
<4:0>	IMPL	RAZ	0x0	Reserved

Root.Compare Register: CP0 Register 11, Select 0



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:0>	Compare	R/W	—	Interval count compare value

Cavium Confidential For Bowen Yang ALU 02/27/2014

Root.GuestCtl0Ext Register: CP0 Register 11, Select 4

31 — 5	4	3	2	1	0
Reserved 0x0	CGI Not Implemented	FCD	OG Not Implemented	BG Not Implemented	MG Not Implemented

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:5>	—	RAZ	0x0	Reserved
<4>	CGI	RAZ	0	Definition of Root.GuestCtl0[CG] does not change. Not implemented.
<3>	FCD	RO	0	Field change disable. When set to 1, disables guest software/hardware field change exceptions (GSFC/GHFC).
<2>	OG	RAZ	0	Other GPSI enable. Not implemented.
<1>	BG	RAZ	0	Bad register GPSI enable. Not implemented.
<0>	MG	RAZ	0	MMU GPSI enable. Not implemented.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Root.Status Register: CP0 Register 12, Select 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17 16	15 10	9 8	7	6	5	4 3	2	1	0
CU3 0	CU2	CU1 0	CU0	RP 0	FR 0	RE 0	MX 0	PX	BEV	TS	SR	NMI	ASE 0	Impl 00	IM7...IM2	IM1, IM0	KX	SX	UX	KSU	ERL	EXL	IE

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	CU3	RAZ	0	CU3 is no longer used and is reserved for future use.
<30>	CU2	R/W	—	Control access to coprocessor 2
<29>	CU1	R/W	0	Controls access to coprocessor 1 (FP and Prefetch Indexed Instructions). Implemented.
<28>	CU0	R/W	—	Control access to coprocessor 0
<27>	RP	RAZ	0	Reduced power mode is not implemented.
<26>	FR	R/W	0	FR = 1 provides true 64-bit FPU. FR = 0 provides paired register mode with 32 32-bit FP registers or 16 64-bit FP registers with FP registers numbered as 0, 2, 4, ... (only even numbered registers legal for 64-bit FP and FR = 0).
<25>	RE	RAZ	0	Reverse-endian mode is not implemented.
<24>	MX	RAZ	0	MDMX™ is not implemented.
<23>	PX	R/W	—	Enables access to 64-bit operations in User mode, without enabling 64-bit addressing.
<22>	BEV	R/W	1	Controls the location of exception vectors.
<21>	TS	R/W	0	Indicates that the TLB has detected a match on multiple entries (i.e machine-check).
<20>	SR	R/W	—	Set the first time that a core comes out of reset following a chip soft reset that was due to either CIU_SOFT_RST CSR write (refer to Section TBD) or due to watchdog timer (refer to Section TBD). Otherwise, clear when the core comes out of reset. (Refer also to MIO_RST_BOOT[LBOOT]).
<19>	NMI	R/W	0	Indicates that the entry through the reset exception vector was due to an NMI exception.
<18>	ASE	RO	0	MCU ASE is not implemented; reads as zero.
<17:16>	Impl	RO	0x0	Ignored on write and read as zeros.
<15:10>	IM7...IM2	R/W	—	Interrupt Masks. When set, these mask bits enable the corresponding Cause[IP7..IP2] interrupts.
<9:8>	IM1, IM0	R/W	—	Interrupt Masks. These are used to control software interrupts.
<7>	KX	R/W	—	Enables access to 64-bit Kernel Segments and use of the XTLB Refill Vector for references to Kernel Segments.
<6>	SX	R/W	—	Enables access to 64-bit Supervisor Segments and use of the XTLB Refill Vector for references to Supervisor Segments.
<5>	UX	R/W	—	Enables access to 64-bit User Segments, Use of the XTLB Refill Vector for references to User Segments and Execution of instructions which perform 64-bit operations while the processor is operating in User Mode.
<4:3>	KSU	R/W	—	The encoding of this field denotes the base operating mode of the processor.
<2>	ERL	R/W	1	Error Level
<1>	EXL	R/W	—	Exception Level
<0>	IE	R/W	—	Interrupt Enable

NOTE: The Dcache (virtual tags only, not physical tags) is invalidated whenever the Status[ERL] value changes.

Cavium Confidential For Bowen Yang 02/21/2014

Root.IntCtl Register: CP0 Register 12, Select 1

31 29	28 26	25 23	22 14	13 10	9 5	4 0
IPTI	IPPCI	IPFDC	MCUASE 0	Reserved 0	VS 0	Reserved 0

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:29>	IPTI	RO	0x7	This field is a read-only copy of CvmCtl[IPTI].
<28:26>	IPPCI	RO	0x7	This field is a read-only copy of CvmCtl[IPPCI].
<25:23>	IPFDC	RAZ	0x0	Fast debug channel not implemented
<22:14>	MCUASE	RO	0x0	MCU ASE is not implemented; reads as zero.
<13:10>	—	RAZ	0x0	Reserved
<9:5>	VS	RAZ	0x0	Vectored interrupt not implemented
<4:0>	—	RAZ	0x0	Reserved

Root.SRSCtl Register: CP0 Register 12, Select 2

cnMIPS™ II cores have no GPR shadow sets so the entire SRSCtl register is read as zero.

31 30	29 26	25 22	21 18	17 16	15 12	11 10	9 6	5 4	3 0
Reserved 00	HSS 00 00	Reserved 00 00	EICSS 00 00	Reserved 00	ESS 0000	Reserved 00	PSS 00 00	Reserved 00	CSS 0000

NOTE: SRS map is not implemented.

Root.GuestCtl0 Register: CP0 Register 12, Select 6

31	30	29	28	27 26	25	24	23	22	21	20	19	18	17 16	15 10	9	8	7	6 2	1	0
GM	RI	MC	CP0	AT	GT	CG	CF	G1	CVMSW	CVMGT	G0E	PT	ASE	PIP	RAD	DRG	G2	GExcCode	SFC2	SFC1

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	GM	R/W	0	Guest mode. The processor is in guest mode when GM = 1, Root.Status[EXL] = 0 and Root.Status[ERL] = 0 and Root.Debug[DM] = 0.
<30>	RI	R/W	0	Guest reserved instruction redirect. 1 = reserved-instruction exceptions during guest-mode execution result in a guest reserved instruction redirect exception, taken in root mode. 0 = reserved-instruction exceptions during guest-mode execution are taken in guest mode.
<29>	MC	R/W	0	Guest mode-change exception enable. The purpose of this enable is to provide root-software control over certain mode-changing events within guest context that may be frequent in guest context by causing field change exceptions. 1 = During guest-mode execution, a hardware-initiated change to Guest.Status[EXL] triggers a GHFC exception, and a software-initiated change to Guest.Status[UM/KSU] triggers a GSFC exception. 0 = During guest-mode execution, a hardware-initiated change to Guest.Status[EXL] does not trigger a GHFC exception, a software-initiated change to Guest.Status[UM/KSU] does not trigger a GSFC exception.

Cavium Confidential For Bowen Yang ALU 04/27/2014

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<28>	CP0	R/W	0	Guest access to coprocessor 0. 1 = guest-kernel use of a guest privileged sensitive instructions is permitted, subject to all other exception conditions. 0 = guest-kernel use of a guest privileged sensitive instruction triggers a GPSI exception. CP0 has no other effect on the operation of coprocessor 0 in guest mode.
<27:26>	AT	RO	0x3	Guest MMU under guest control. Guest and root MMU are both implemented and active in hardware. Guest TLB resources are: • TLB-related instructions: TLBWR, TLBWI, TLBR, TLBP, TLBINV, TLBINVF. • supporting registers: Index, Random, EntryLo0, EntryLo1, EntryHi, Context, XContext, ContextConfig, PageMask, PageGrain, SegCtl0, SegCtl1, SegCtl2, PWBase, PWField, PWSize, PWCtl.
<25>	GT	R/W	0	Timer register access. 1 = guest kernel read access from Count and guest-kernel read or write access to Compare is permitted. Guest read requests from CC using RDHWR are permitted in any mode. 0 = guest kernel accesses to Count or Compare registers, or read requests from CC with RDHWR trigger a GPSI exception.
<24>	CG	RO	0	A GPSI exception results from the use of the CACHE instruction.
<23>	CF	R/W	0	Configuration register access. 1 = guest-kernel access to Config0–7 is permitted. 0 = guest-kernel access to Config0–7 triggers a GPSI exception.
<22>	G1	RO	0	GuestCtl1 not implemented.
<21>	CVMSW	R/W		RDHWR 30 instruction allowed: 1 = allowed, 0 = triggers a GPSI exception. Cavium only.
<20>	CVMGT	R/W		RDHWR 31 instruction allowed: 1 = allowed, 0 = RDHWR 31 or a guest kernel read operation to CvmCount triggers a GPSI exception. Cavium only.
<19>	G0E	RO	1	GuestCtl0Ext is present.
<18>	PT	RO	1	Pending-interrupt passthrough (PIP) is supported. Interrupts can be assigned to root or guest.
<17:16>	ASE	RAZ	0x0	Reserved.
<15:10>	PIP	R/W	0x0	Pending-interrupt passthrough. Controls how external interrupts are passed through to the guest CP0 context. Interpreted as a bit mask and applies one-to-one to Guest.Cause[IP<7:2>].
<9>	RAD	RO	1	Root ASID is used to de-alias root TLB entries, while guest TLB contains only one context at any given time.
<8>	DRG	RO	0	Root software cannot access guest entries directly.
<7>	G2	RO	1	GuestCtl2 is implemented.
<6:2>	GExcCode	RO	—	Hypervisor exception cause code. Refer to Table 2–37.
<1>	SFC2	R/W	0	Guest software-field-change exception enable for guest.status.CU2. This bit provides root-software control over guest COP2-enable-related field change exception. Guest software can use status.CU2 for COP2-specific context switching. 1 = exception is not taken, 0 = exception is taken
<0>	SFC1	R/W	0	Guest software-field-change exception enable for guest.status.CU1. This bit provides root-software control over guest COP1-enable-related field change exception. Guest software can use status.CU1 for COP1-specific context switching. 1 = exception is not taken, 0 = exception is taken

Table 2–37 Hypervisor Exception Cause Values

Value	Mnemonic	Description
0x0	GPSI	Guest privileged sensitive instruction. Taken when execution of a GPSI was attempted from guest-kernel mode, but the instruction was not enabled for guest-kernel mode.
0x1	GSFC	Guest software field-change event.
0x2	HC	Hypercall.

Cavium Confidential For Boyen Yang ALU 02/27/2014

Table 2–37 Hypervisor Exception Cause Values

Value	Mnemonic	Description
0x3	GRR	Guest reserved instruction redirect. A reserved instruction or MDMX unusable exception would be taken in guest mode. When GuestCtl0[RI] = 1, this root-mode exception is raised before the guest-mode exception can be taken.
0x4	GCvmSegAcc	Guest attempted a CvmSeg access but Root.CvmMemCtl[CVMSEGIODIS] is set. For this exception, Root.Cause[ExcCode] is set to 0x1B (guest exit). Cavium only.
0x5–0x7	—	Reserved.
0x8	GVA	Guest-mode-initiated root TLB exception has guest virtual address available. Set when a guest-mode-initiated TLB translation results in a root TLB-related exception occurring in root mode and the guest physical address is not available.
0x9	GHFC	Guest-hardware field-change event.
0xA	GPA	Guest-mode-initiated root TLB exception has guest physical address available. Set when a guest-mode-initiated TLB translation results in a root TLB-related exception occurring in root mode and the guest physical address is available.
0xB–0x1F	—	Reserved.

Root.GTOffset Register CP0 Register 12, Select 7

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:0>	GTOffset	R/W	0x0	2s-complement offset from Root.Count[Count].

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Root.Cause Register CP0 Register 13, Select 0

31	30	29 28	27	26	25 24	23	22	21	20 18	17 16	15 13	12 10	9 8	7	6 2	1 0
BD	TI	CE	DC	PCI	ASE 00	IV	WP	FDCI 0	Reserved 0 00	ASE 00	IP7...IP5	IP4...IP2	IP1, IP0	Reserved 0	ExcCode	Reserved 00

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	BD	RO	0	Indicates whether the last exception taken occurred in a branch-delay slot.
<30>	TI	RO	0	Timer Interrupt; indicates whether timer interrupt is present.
<29:28>	CE	RO	0x0	Coprocessor unit number referenced when a coprocessor unusable exception is taken.
<27>	DC	R/W	0	Disable count register.
<26>	PCI	RO	0	Implemented; four performance counters implemented. The performance counters might cause root interrupts when they are active in the root context.
<25:24>	ASE	RO	0x0	MCU ASE is not implemented; reads as zero.
<23>	IV	R/W	—	Indicates whether an interrupt exception uses the general exception vector or a special interrupt vector.
<22>	WP	R/W	—	Implemented (two watch registers implemented).
<21>	FDCI	RAZ	0	Fast debug channel not implemented.
<20:18>	—	RAZ	0x0	Reserved.
<17:16>	ASE	RO	0x0	MCU ASE is not implemented; reads as zero.
<15:13>	IP7...IP5	RO	—	Indicate an interrupt is pending. May be set for timer interrupts (Cause[TI] is set in this case) and/or performance counter interrupts (Root.Cause[PCI] is set in this case). Root.CvmCtl[IPTI,IPPCI] select the IP bits used for timer and performance counter interrupts, respectively.
<12:10>	IP4...IP2	RO	—	Indicate an interrupt is pending. may assert due to timer and performance counter interrupts, and may additionally assert due to the CIU interrupt distribution. Refer to Chapter TBD.
<9:8>	IP1, IP0	R/W	—	Indicate an interrupt is pending. These are used to control the request for software interrupts.
<7>	—	RAZ	0	Reserved
<6:2>	ExcCode	RO	0x0	Exception code. This field is compliant with MIPS® documentation. Please refer to the MIPS® documentation for cause code value assignment.
<1:0>	—	RAZ	0x0	Reserved.

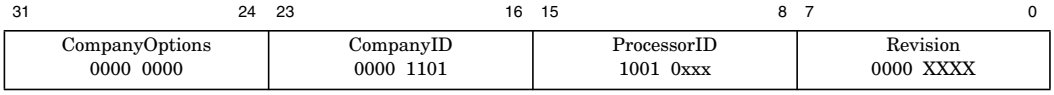
Root.EPC Register: CP0 Register 14, Select 0

63	0
EPC	

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:0>	EPC	R/W	—	Exception program counter

Cavium Confidential For Bowen Yang ALD 02/27/2014

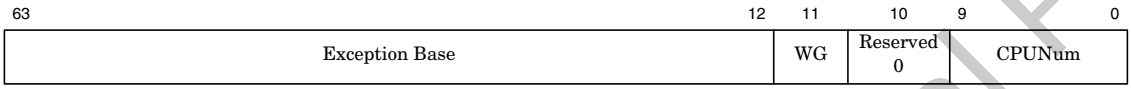
Root.PRId Register: CP0 Register 15, Select 0



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:24>	CompanyOptions	RO	0x0	0x0
<23:16>	CompanyID	RO	0xD	13 (0xD)
<15:8>	ProcessorID	RO	0x96	0x96 = CN71XX
<7:0>	Revision	RO	0x0	0x0 = pass 1.0, 0x1 = pass 1.1, 0x2 = pass 1.2, 0x8 = pass 2.0, 0x9 = pass 2.1, 0xA = pass 2.2

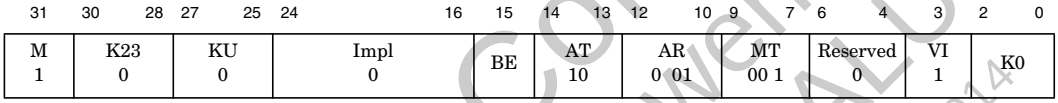
Root.EBase Register: CP0 Register 15, Select 1

EBase is only supported to KSEG0, KSEG1, and XKPHYS addresses.



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:12>	Exception Base	R/W	0xFFFFFFFF80000	Base address of the exception vectors when Root.Status[BEV] = 0. Bits<29:12> are always writable. Bits<63:30> are writable only when WG = 1.
<11>	WG	R/W		When set, bits<63:30> of Exception Base are writable
<10>	—	RAZ	0	Reserved
<9:0>	CPUNum	RO	—	Specifies the cnMIPS III core ID.

Root.Config Register: CP0 Register 16, Select 0



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	M	RO	1	Indicates that the Root.Config1 register is present.
<30:28>	K23	RAZ	0x0	Fixed Mapping MMU not implemented.
<27:25>	KU	RAZ	0x0	Fixed Mapping MMU not implemented.
<24:16>	Impl	RO	0x0	Read as zero.
<15>	BE	RO	1	$\overline{\text{Root.CvmCtl[LE]}}$
<14:13>	AT	RO	0x2	MIPS 64 with access to all address segments
<12:10>	AR	RO	0x1	Release 2, Release 3/MIPSr3, or Release 5.
<9:7>	MT	RO	0x1	Standard TLB
<6:4>	—	RAZ	0x0	Reserved
<3>	VI	RO	1	Icache is virtual.
<2:0>	K0	R/W	0x2	Field is implemented, and the only effect is on the ordering of root-mode kseg0 stores. Section 2.8 describes the CN71XX use of the MIPS cacheability and coherency attributes.

Cavium Confidential For Bowen Yang ALU 03/27/2014

Root.Config1 Register: CP0 Register 16, Select 1

31	30 25	24 22	21 19	18 16	15 13	12 10	9 7	6	5	4	3	2	1	0
M	MMU Size–1	IS	IL	IA	DS	DL	DA	C2	MD	PC	WR	CA	EP	FP
1	111 111	0 00	11 0	111	000	110	11 1	1	0	1	1	0	1	0

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	M	RO	1	Indicates that the Root.Config2 register is present.
<30:25>	MMU Size –1	RO	0x3F	This field is the lower six bits of Root.CvmVMConfig[RootMMUSizeM1]. "Root.Config4 Register: CP0 Register 16, Select 4" on page 122 extends this field. Reset value is 256 entries.
<24:22>	IS	RO	0x0	The reset value denotes 64 Icache sets per way per the MIPS specification. The actual Cavium implementation is 16 sets per way.[1]
<21:19>	IL	RO	0x6	128-byte Icache line size
<18:16>	IA	RO	0x7	The reset value denotes 8-way Icache per the MIPS specification. The actual Cavium implementation is 39-way Icache.[1]
<15:13>	DS	RO	0x0	The reset value denotes 64 Dcache sets per way per the MIPS specification. The actual Cavium implementation is eight sets per way.[1]
<12:10>	DL	RO	0x0	128-byte Dcache line size
<9:7>	DA	RO	0x7	The reset value denotes 8-way Dcache per the MIPS specification. The actual Cavium implementation is 32-way Dcache.[1]
<6>	C2	RO	1	Coprocessor 2 implemented
<5>	MD	RAZ	0	MDMX ASE not implemented
<4>	PC	RO	1	Four Performance Counter registers implemented
<3>	WR	RO	1	Two Watch registers implemented
<2>	CA	RAZ	0	MIPS16e not implemented
<1>	EP	RO	1	EJTAG implemented
<0>	FP	RO	1	FPU implemented

1. This mismatch exists because MIPS does not have an encoding for Cavium's implementation.

Root.Config2 Register: CP0 Register 16, Select 2

31	30 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
M	TU	TS	TL	TA	SU	SS	SL	SA
1	000	0000	0000	0000	0000	0100	0110	0111

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	M	RO	1	Root.Config 3 is present.
<30:28>	TU	RAZ	0x0	Not implemented
<27:24>	TS	RO	0x0	No tertiary cache
<23:20>	TL	RO	0x0	No tertiary cache
<19:16>	TA	RO	0x0	No tertiary cache
<15:12>	SU	RAZ	0x0	Not implemented
<11:8>	SS	RO	0x4	1024 sets per way in secondary cache
<7:4>	SL	RO	0x6	128-byte cache line in secondary cache
<3:0>	SA	RO	0x7	The value denotes 8-way set-associative secondary cache per the MIPS specification. The actual Cavium implementation is 4-way set-associative secondary cache.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Root.Config3 Register: CP0 Register 16, Select 3

31	30	29	28	27	26	25	24	23	22 21	20 18	17	16	1514
M	BPG	CMGCR	MSAP	BP	BI	SC	PW	VZ	IPLW	MMAR	MCU	ISAOnExc	ISA
1	0	0	0	1	1	0	0	1	0	0	0	0	00

13	12	11	10	9	8	7	6	5	4	3	2	1	0
ULRI	RXI	DSP2P	DSPP	CTXTC	ITL	LPA	VEIC	VInt	SP	CDMM	MT	SM	TL
1	1	0	0	0	0	1	0	0	0	0	0	0	0

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	M	RO	1	Config4 is present.
<30>	BPG	RO	0	Big Pages feature not implemented
<29>	CMGCR	RO	0	CM GCR space not implemented
<28>	MSAP	RO	0	MIPS SIMD Architecture (MSA) not implemented
<27>	BP	RO	1	BadInstrP register implemented
<26>	BI	RO	1	BadInstr register implemented
<25>	SC	RO	0	Segment Control not implemented
<25>	PW	RO	0	Hardware Page Table Walk not implemented
<23>	VZ	RO	1	MIPS virtualization ASE implemented
<22:21>	IPLW	RO	0x0	MCU ASE not implemented
<20:18>	MMAR	RO	0x0	microMIPS64 not implemented
<17>	MCU	RO	0	MCU ASE not implemented
<16>	ISAOnExc	RO	0	MIPS64 is used on entrance to an exception vector.
<15:14>	ISA	RO	0x0	Only MIPS64 Instruction Set is implemented.
<13>	ULRI	RO	1	UserLocal register implemented
<12>	RXI	RO	1	Read/execute inhibit function implemented
<11>	DSP2P	RO	0	Revision 2 of the MIPS DSP ASE not implemented
<10>	DSPP	RO	0	MIPS DSP ASE not implemented
<9>	CTXTC	RO	0	ContextConfig is not implemented.
<8>	ITL	RO	0	MIPS IFlowTrace not implemented
<7>	LPA	RO	1	Large physical address support is implemented, and the PageGrain register exists.
<6>	VEIC	RO	0	No external interrupt controller
<5>	VInt	RO	0	Vectored interrupts not implemented
<4>	SP	RO	0	1KB page size not implemented
<3>	CDMM	RO	0	Common Device Memory Map not implemented
<2>	MT	RO	0	MIPS MT ASE not implemented
<1>	SM	RO	0	SmartMIPS™ not implemented
<0>	TL	RO	0	Trace Logic not implemented

Cavium Confidential For Bowen Yang 02/27/2014

Root.Config4 Register: CP0 Register 16, Select 4

This register is only present in cnMIPS III.

31	30 29	28	27 24	23 16	15 14	13 8	7 0
M	IE	AE	VTLBSizeExt	KScrExist	MMUExtDef	Reserved	MMUSizeExt
0	0x0	0	0x0	0x3C	01	0x0	0000 0001

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	M	RO	1	Config5 is present.
<30:29>	IE	RO	0x0	TLBINV, TLBINVF, EntryHi[EHINV] not implemented.
<28>	AE	RO	0	EntryHI[ASID] is not extended to 10 bits.
<27:24>	VTLBSizeExt	RO	0x0	FTLB/VTLB not implemented.
<23:16>	KScrExist	RO	0x3C	Indicates how many scratch registers are available to root kernel-mode software. Each bit represents a select for Register 31. (Bit 16 represents select 0; bit 23 represents select 7.) If the bit is set, the associated scratch register is implemented and available for kernel-mode software. Root scratch registers exist for select 2, 3, 4, and 5.
<15:14>	MMUExtDef	RO	0x1	MMU extension definition. Set to 0x1 (bits <7:0> used as MMUSizeExt).
<13:8>	—	RAZ	0x0	Reserved
<7:0>	MMUSizeExt	RO	0x3	Extension of Root.Config1[MMUSize–1] field. Concatenate this field to the left of the most significant bit of Root.Config1[MMUSize–1] to indicate the actual root MMUSize–1. Reset value is 256 entries.

Root.Config5 Register: CP0 Register 16, Select 5

This register is only present in cnMIPS III.

31	30	29	28	27	26	25 2	1	0
M	K	CV	EVA	Reserved	MSAEn	Reserved	P32	NFExist
0	0	0	0x0	0x0	0	0x0	0	0

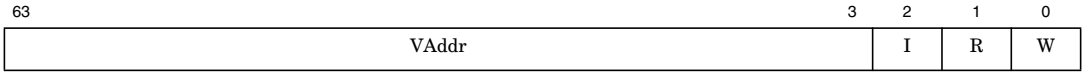
Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	M	RO	0	Config6 is not present.
<30>	K	RO	0	
<29>	CV	RO	0	
<28>	EVA	RO	0	
<27>	MSAEn	RO	0	
<26:3>	—	RAZ	0x0	Reserved
<2>	UFR	RO	0	User-mode FR instructions not allowed.
<1>	P32	RO	0	
<0>	NFExist	RO	0	Nested Fault feature not implemented.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Root.WatchLo Register: CP0 Register 18, select 0, 1

Two watchpoints are implemented; the first (select 0) is instruction-only, and the second (select 1) data-only.

Watch.Lo select 0 (Watch.Lo select 1) is fully available for guest use if Guest.Config1[WR] = 1 and Root.WatchHi0[WM<1>] = 1 (or Root.WatchHi1[WM<1>] = 1).



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:3>	VAddr	R/W	—	Specifies the virtual address to match
<2>	I	R/W	0	Implemented at select = 0 (only first watchpoint is instruction)
<1>	R	R/W	0	Implemented at select = 1 (only second watchpoint is data)
<0>	W	R/W	0	Implemented at select = 1 (only second watchpoint is data)

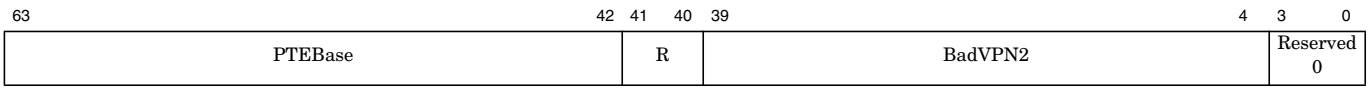
Root.WatchHi Register: CP0 Register 19, select 0, 1

Watch.Hi select 0 (Watch.Hi select 1) is fully available for guest use if Guest.Config1[WR] = 1 and Root.WatchHi0[WM<1>] = 1 (or Root.WatchHi1[WM<1>] = 1).

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	M	RO	0	Set for select = 0, clear for select = 1 (2 watchpoints implemented)
<30>	G	R/W	—	If this bit is one, any address that matches that specified in the WatchLo register will cause a watch exception. If this bit is zero, the ASID field of the WatchHi register must match the ASID field of the EntryHi register to cause a watch exception.
<29:28>	WM	R/W	0x0	This field is used for Root management of Watch functionality in an implementation supporting the Virtualization ASE. Watch based on GPA is not supported (only RVA/GVA Watch is supported), and the only legal values are 0x0 and 0x2: 0x0 = watch is RVA only, 0x2 = watch is GVA only.
<27:24>	—	RAZ	0x0	Reserved
<23:16>	ASID	R/W	—	ASID value which is required to match that in the EntryHi register if the G bit is zero in the WatchHi register.
<15:12>	—	RAZ	0x0	Reserved
<11:3>	Mask	R/W	—	Implemented in the maximum width
<2>	I	W1C	0	Set by hardware when an instruction fetch condition matches the values in this watch register pair.
<1>	R	W1C	0	Set by hardware when a load condition matches the values in this watch register pair.
<0>	W	W1C	0	Set by hardware when a store condition matches the values in this watch register pair.

Cavium Confidential For Edward Yang 02/10/2014

Root.XContext Register: CP0 Register 20, Select 0



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:42>	PTEBase	R/W	—	This field is for use by the operating system and is normally written with a value that allows the operating system to use the Context Register as a pointer into the current PTE array in memory.
<41:40>	R	RO	0x0	Region field. Contains bits 63...62 of the virtual address.
<39:4>	BadVPN2	RO	0x0	Bad Virtual Page Number/2 field is written by hardware on a miss.
<3:0>	—	RAZ	0x0	Reserved

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Root.Debug Register: CP0 Register 23, Select 0

31	30	29	28	27	26	25	24	23	22	21	20	19
DBD	DM	NoDCR 0	LSNM	Doze 0	Halt 0	CountDM	IBusEP 0	MCheckP 0	CacheEP 0	DBusEP	IEXI	DDBSImpr 0

18	17 15	14 10	9	8	7	6	5	4	3	2	1	0
DDBLImpr	EJTAGver 101	DExcCode	NoSSt 0	SSt	OffLine 0	DIBImpr	DINT	DIB	DDBS	DDBL	DBp	DSS

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	DBD	RO	—	Indicates whether the last debug exception or exception in Debug Mode occurred in a branch or jump delay slot.
<30>	DM	RO	0	Indicates that the processor is operating in Debug Mode.
<29>	NoDCR	RO	0	dseg present
<28>	LSNM	R/W	0	Implemented
<27>	Doze	RO	0	No low power mode
<26>	Halt	RO	0	Internal bus clock never stopped
<25>	CountDM	R/W	0	Implemented
<24>	IBusEP	RO	0	No imprecise bus errors
<23>	MCheckP	RO	0	No imprecise machine checks
<22>	CacheEP	R/W1	0	No imprecise cache errors
<21>	DBusEP	R/W1	0	Implemented
<20>	IEXI	R/W	0	Implemented
<19>	DDBSImpr	RO	0	No imprecise Debug Data breakpoint on store
<18>	DDBLImpr	RO	0	Implemented for load value breakpoints
<17:15>	EJTAGver	RO	0x5	EJTAG version 5.0
<14:10>	DExcCode	RO	—	Implemented
<9>	NoSSt	RO	0	Single step is implemented
<8>	SSt	R/W	0	Implemented
<7>	OffLine	RAZ	0	MIPS MT ASE not implemented.
<6>	DIBImpr	RO	0	Implemented
<5>	DINT	RO	—	Implemented
<4>	DIB	RO	—	Implemented
<3>	DDBS	RO	—	Implemented
<2>	DDBL	RO	—	Implemented
<1>	DBp	RO	—	Implemented
<0>	DSS	RO	—	Implemented

Root.Debug2 Register: CP0 Register 23, Select 6

This register is only present in cnMIPS III.

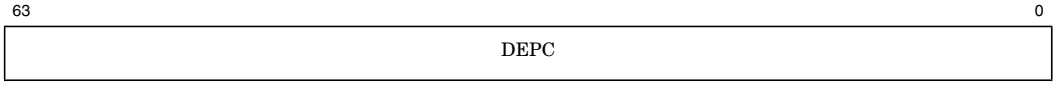
63 4	3	2	1	0
Reserved 0	Prm 0	DQ 0	Tup 0	PaCo 0

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:4>	—	RAZ	0x0	Reserved
<3>	Prm	RAZ	0	Complex breakpoints not implemented.
<2>	DQ	RAZ	0	Complex breakpoints not implemented.
<1>	Tup	RAZ	0	Complex breakpoints not implemented.
<0>	PaCo	RAZ	0	Complex breakpoints not implemented.

Cavium Confidential For Bowen Yang ALU 02/27/2014

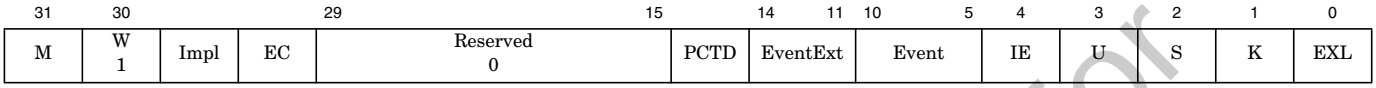
Root.DEPC Register: CP0 Register 24, Select 0

Debug exception program counter register. For imprecise debug exceptions and imprecise exceptions in debug mode, the DEPC register contains the address at which execution is resumed when returning to non-debug mode.



Root.PerfCntCtl Register: CP0 Register 25, Select 0, 2, 4, 6

Performance counter control register. PerfCntCtl *n* is fully available for guest use if Guest.Config1[PC] = 1 and Root.PerfCntCtl *n*[EC] = 0x2 or 0x3.



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	M	RO	0/1	1 for sel = 0 (PerfCntCtl1 is present), 1 for sel = 2 (PerfCntCtl2 is present), 1 for sel = 4 (PerfCntCtl3 is present), 0 for sel = 6 (PerfCntCtl4 is not present)
<30>	W	RO	1	1 (64-bit counter)
<29:25>	Impl	RO	0x0	
<24:23>	EC	R/W	R/W	Event class: 0x0 = root events counted; active in root context 0x1 = root intervention events counted; active in root context 0x2 = guest events counted; active in guest context 0x3 = guest events plus root intervention events counted; active in guest context
<22:16>	—	RAZ	0x0	Reserved
<15>	PCTD	RAZ	0	
<14:11>	EventExt	R/W	0x0	The event selection is the concatenation of the Event and EventExt fields, i.e., EventExt | Event.
<10:5>	Event	R/W	—	Selects the event to be counted by the corresponding Counter Register. See Table 2–38. See also EventExt.
<4>	IE	R/W	0	Interrupt enable.
<3>	U	R/W	—	Enables event counting in user mode.
<2>	S	R/W	—	Enables event counting in supervisor mode.
<1>	K	R/W	—	Enables event counting in kernel mode.
<0>	EXL	R/W	—	Enables event counting when the EXL bit in the Status register is one and the ERL bit in the Status register is zero.

Table 2–38 Performance Counter Control Register Events

Value	Event	Meaning
0x0	—	Reserved
0x1	PERF_CNT_CLK	Conditionally clocked cycles (as opposed to count/cvm_count which count even with no clock cycles)
0x2	PERF_CNT_ISSUE	Instructions issued but not retired
0x3	PERF_CNT_RET	Instructions retired
0x4	PERF_CNT_NISSUE	Cycles no issue
0x5	PERF_CNT_SISSUE	Cycles single issue
0x6	PERF_CNT_DISSUE	Cycles dual issue
0x7	PERF_CNT_IFI	Cycle ifetch issued (but not necessarily commit to pp_mem)

Cavium Confidential For Bowen Yang ALD 10/27/2014

Table 2–38 Performance Counter Control Register Events (Continued)

Value	Event	Meaning
0x8	PERF_CNT_BR	Branches retired, excluding branch likelies
0x9	PERF_CNT_BRMIS	Branch mispredicts, excluding branch likelies
0xA	PERF_CNT_J	Jumps retired, does not include J/JAL
0xB	PERF_CNT_JMIS	Jumps mispredicted, does not include J/JAL
0xC	PERF_CNT_REPLAY	Mem Replays
0xD	PERF_CNT_IUNA	Cycles idle due to unaligned_replays
0xE	PERF_CNT_TRAP	Traps are replays caused by the memory system, as well as exceptions and PERF_CNT_HAZTR's. Branch/jump mispredict do not cause these traps.
0xF	—	Reserved
0x10	PERF_CNT_UULOAD	Unexpected unaligned loads (REPUN=1)
0x11	PERF_CNT_UUSTORE	Unexpected unaligned store (REPUN=1)
0x12	PERF_CNT_ULOAD	Unaligned loads (REPUN=1 or USEUN=1)
0x13	PERF_CNT_USTORE	Unaligned store (REPUN=1 or USEUN=1)
0x14	PERF_CNT_EC	Exec clocks (must set CvmCtl[DISCE] for accurate timing)
0x15	PERF_CNT_MC	Mul clocks (must set CvmCtl[DISCE] for accurate timing)
0x16	PERF_CNT_CC	Crypto clocks (must set CvmCtl[DISCE] for accurate timing)
0x17	PERF_CNT_CSRC	Issue_csr clocks (must set CvmCtl[DISCE] for accurate timing)
0x18	PERF_CNT_CFETCH	Icache committed fetches (demand+prefetch)
0x19	PERF_CNT_CPREF	Icache committed prefetches
0x1A	PERF_CNT_ICA	Icache aliases
0x1B	PERF_CNT_II	Icache invalidates
0x1C	PERF_CNT_IP	Icache parity error
0x1D	PERF_CNT_CIMISS	Cycles idle due to imiss (must set CvmCtl[DISCE] for accurate timing)
0x1E	—	Reserved
0x1F	—	Reserved
0x20	PERF_CNT_WBUF	Number of write buffer entries created
0x21	PERF_CNT_WDAT	Number of write buffer data cycles used (may need to set CvmCtl[DISCE] for accurate counts)
0x22	PERF_CNT_WBUFLD	Number of write buffer entries forced out by loads
0x23	PERF_CNT_WBUFFL	Number of cycles that there was no available write buffer entry (may need to set CvmCtl[DISCE] and CvmMemCtl[MCLKALWYS] for accurate counts)
0x24	PERF_CNT_WBUFTR	Number of stores that found no available write buffer entries
0x25	PERF_CNT_BADD	Number of address bus cycles used (may need to set CvmCtl[DISCE] for accurate counts)
0x26	PERF_CNT_BADDL2	Number of address bus cycles not reflected (i.e. destined for L2) (may need to set CvmCtl[DISCE] for accurate counts)
0x27	PERF_CNT_BFILL	Number of fill bus cycles used (may need to set CvmCtl[DISCE] for accurate counts)
0x28	PERF_CNT_DDIDS	Number of Dstream DIDs created
0x29	PERF_CNT_IDIDS	Number of Istream DIDs created
0x2A	PERF_CNT_DIDNA	Number of cycles that no DIDs were available (may need to set CvmCtl[DISCE] and CvmMemCtl[MCLKALWYS] for accurate counts)
0x2B	PERF_CNT_LDS	Number of load issues
0x2C	PERF_CNT_LMLDS	Number of local memory load issues
0x2D	PERF_CNT_IOLDS	Number of I/O load issues
0x2E	PERF_CNT_DMLDS	Number of loads that were not prefetches and missed in the cache
0x2F	—	Reserved

Cavium Confidential For Bowen Yang 02/27/2014

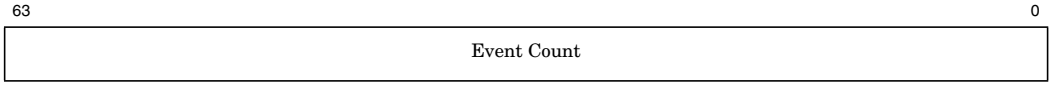
Table 2–38 Performance Counter Control Register Events (Continued)

Value	Event	Meaning	
0x30	PERF_CNT_STS	Number of store issues	
0x31	PERF_CNT_LMSTS	Number of local memory store issues	
0x32	PERF_CNT_IOSTS	Number of I/O store issues	
0x33	PERF_CNT_IOBDMA	Number of IOBDMA operations	
0x34	—	Reserved	
0x35	PERF_CNT_DTLB	Number of dstream TLB refill, invalid, or modified exceptions	
0x36	PERF_CNT_DTLBAD	Number of dstream TLB address errors	
0x37	PERF_CNT_ITLB	Number of istream TLB refill, invalid, or address error exceptions	
0x38	PERF_CNT_SYNC	Number of SYNC stall cycles (may need to set CvmCtl[DISCE] for accurate counts)	
0x39	PERF_CNT_SYNCIOB	Number of SYNCIOBDMA stall cycles (may need to set CvmCtl[DISCE] for accurate counts)	
0x3A	PERF_CNT_SYNCW	Number of SYNCWs or SYNCWSs	
0x3B	PERF_CNT_DUTLB	Number of dstream misses in the µTLB.	**cnMIPS III** only
0x3C	PERF_CNT_IUTLB	Number of istream misses in the µTLB.	**cnMIPS III** only
0x3D	PERF_CNT_CDMISS	Cycles idle due to demand (non-prefetch) Dstream misses	**cnMIPS III** only
0x3E -0x3F	—	Reserved	
0x40	PERF_CNT_ERETMIS	D/eret mispredicts	
0x41	PERF_CNT_LIKMIS	Branch likely mispredicts	
0x42	PERF_CNT_HAZTR	Hazard traps due to *MTC0's to CvmCtl, Perf counter control, EntryHi, or CvmMemCtl register	
0x43	PERF_CNT_FPUNIMPTRAP	Number of FP unimplemented traps	
0x44	PERF_CNT_FPHAZARDTRAP	Number of FP hazard traps	
0x45	PERF_CNT_FPARITHEXC	Number of FP Arithmetic Exceptions	
0x51	PERF_CNT_FPMOVC1	Number of retired MTC1, DMTC1, MFHC1, MTHC1, MFC1, DMFC1 instructions	
0x52	PERF_CNT_FPCOPYC1	Number of retired CFC1 and CTC1 instructions	
0x53	PERF_CNT_FPCOMPARE	Number of retired FP C.Cond.fmt instructions	
0x54	PERF_CNT_FPBRANCH	Number of retired BC1T, BC1F, BC1TL, and BC1FL instructions	
0x55	PERF_CNT_FPMOV	Number of retired FP move instructions	
0x56	PERF_CNT_FPABSNEG	Number of retired ABS.fmt and NEG.fmt instructions	
0x57	PERF_CNT_FPADDSUB	Number of retired ADD.fmt and SUB.fmt instructions	
0x58	PERF_CNT_FPCVT	Number of retired FP format conversion instructions (e.g., CVT.S.D)	
0x59	PERF_CNT_FPMUL	Number of retired MUL.fmt instructions	
0x5A	PERF_CNT_FPMADD	Number of retired MADD.fmt variant instructions	
0x5B	PERF_CNT_FPDIVRECIP	Number of retired DIV.fmt and RECIP.fmt instructions	
0x5C	PERF_CNT_FPSQRTRSQRT	Number of retired SQRT.fmt and RSQRT.fmt instructions	
0x5D	PERF_CNT_FPLOAD	Number of retired FP load instructions	
0x5E	PERF_CNT_FPSTORE	Number of retired FP store instructions	
0x5F	PERF_CNT_FPALL	Number of retired FP instructions (all)	

Cavium Confidential For Bowen Yang ALD 04/27/2014

Root.PerfCntCtr Register: CP0 Register 25, select 1, 3, 5, 7

PerfCntCtl *n* is fully available for guest use if Guest.Config1[PC] = 1 and Root.PerfCntCtl *n*[EC] = 0x2 or 0x3.



Performance counter counter register. Increments once for each event that is enabled by the corresponding PerfCntCtl Register. When the MSB is one, a pending interrupt request is ORed with those from other performance counters and indicated by the PCI bit in the Cause register.

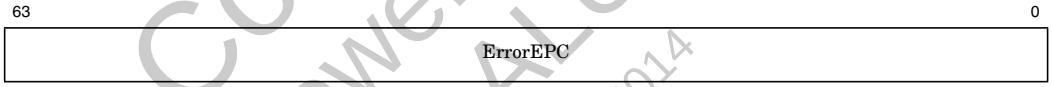
To save power, the implementation may not update the performance counter counter registers every cycle when CvmCtl[DCICLK] is clear—the current counter register value may not include all the recent updates.

To ensure that a performance counter counter register contains all of the recent updates when CvmCtl[DCICLK] is clear, either:

- issue a MTC0/DMTC0 to the corresponding performance counter control register, or
- issue a MFC0/DMFC0 to the performance counter counter register

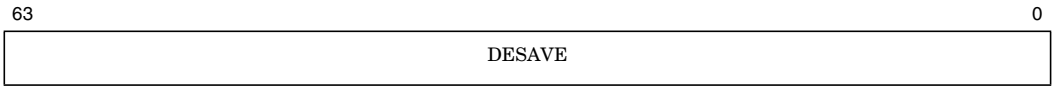
Note that with the latter case (a single MFC0/DMFC0), the counter register will update two cycles after the instruction, but the destination GPR of the instruction may not contain the updated value. To get the most recent value of the performance counter counter register when CvmCtl[DCICLK] is clear, execute three consecutive MFC0/DMFC0 operations to the register, all with the same destination GPR.

Root.ErrorEPC Register: CP0 Register 30, Select 0



The ErrorEPC register contains the virtual address at which instruction processing can resume after servicing an error.

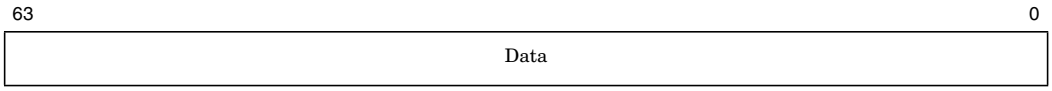
Root.DESAVE Register: CP0 Register 31, Select 0



The Debug Exception Save (DESAVE) register is a read/write register that functions as a simple scratchpad register.

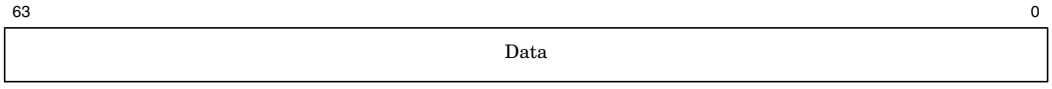
Cavium Confidential For Bowen Yang ALU 02/27/2014

Root.KScratch1 Register: CP0 Register 31, Select 2



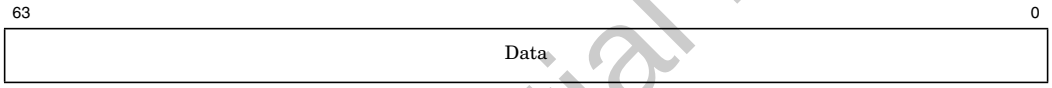
The KScratch1 register is a read/write register available for scratch pad storage by kernel mode software.

Root.KScratch2 Register: CP0 Register 31, Select 3



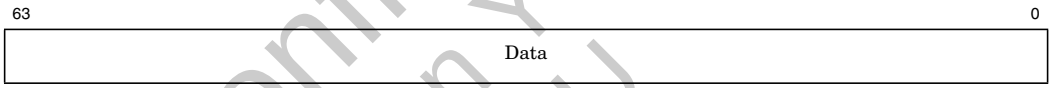
The KScratch2 register is a read/write register available for scratch pad storage by kernel mode software.

Root.KScratch3 Register: CP0 Register 31, Select 4



The KScratch3 register is a read/write register available for scratch pad storage by kernel mode software.

Root.KScratch4 Register: CP0 Register 31, Select 5



The KScratch4 register is a read/write register available for scratch pad storage by kernel mode software.

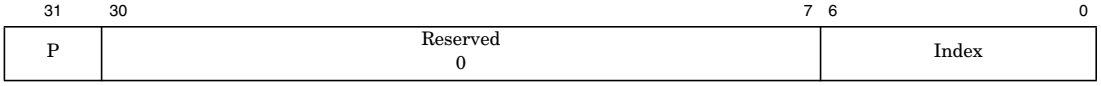
Cavium Confidential For
Bowen Yang
ALU
02/27/2014

2.15.2 Coprocessor 0 Privileged Register in Guest Mode

The set of coprocessor 0 registers available in guest mode can also be accessed from root mode. In the following field-description tables, the **Field TypeG** column specifies the field access when in guest mode, while the **Field TypeR** column specifies the field access when in root mode

Guest.Index Register: CP0 Register 0, Select 0

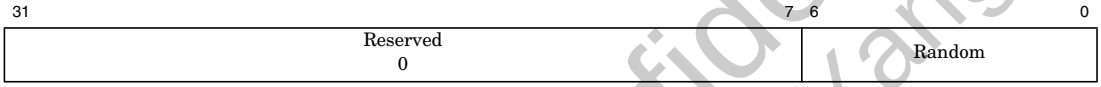
When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31>	P	RO	R/W	0	Probe failure. Hardware writes this bit during execution of the GTLBP instruction to indicate whether a GTLB match occurred.
<30:8>	—	RAZ	RAZ	0x0	Reserved
<7:0>	Index	R/W	R/W	—	GTLB index. Software writes this field to provide the index to the GTLB entry referenced by the TLBR and TLBWI instructions.

Guest.Random Register: CP0 Register 1, Select 0

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.



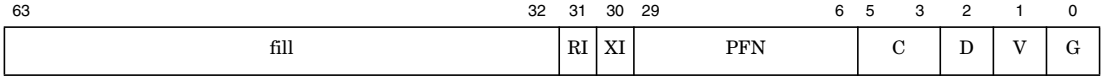
Bit Pos	Field Name	Field TypeG	Field TypeG	Reset Value	Field Description
<31:8>	—	RAZ	RAZ	0x0	Reserved.
<7:0>	Random	RO	RO	TLB entries–1	TLB random index.

Cavium Confidential For Bowen Yang ALU 02/27/2014

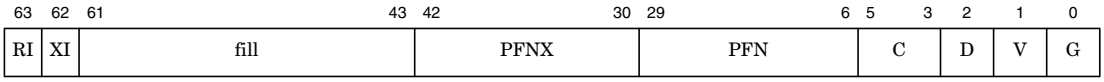
Guest.EntryLo0, EntryLo1 Registers: CP0 Registers 2 and 3, Select 0

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

When Guest.PageGrain[ELPA] = 0:



When Guest.PageGrain[ELPA] = 1:

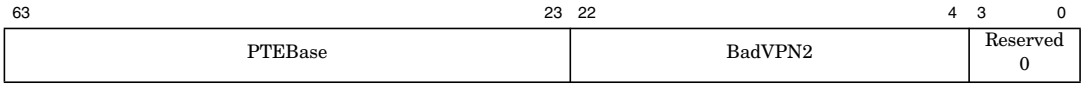


Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<63:32>/ <61:43>	fill	RO	RO	0x0	Ignored on write, return 0x0 on read.
<31>/ <63>	RI	R/W	R/W	0	Read inhibit. If this bit is set to 1 in a TLB entry, any load instruction that reads data on the virtual page causes a TLB invalid exception, even if the V (valid) bit is set. This bit is writable only if PageGrain[RIE] = 1. If PageGrain[RIE] = 0, this bit is set to 0 on any write to the EntryLo0/1 register, regardless of the value written.
<30/62>	XI	R/W	R/W	0	Execute inhibit. If this bit is set to 1 in a TLB entry, any attempt to fetch an instruction causes a TLB invalid exception, even if the V (valid) bit is set. This bit is writable only if PageGrain[XIE] = 1. If PageGrain[XIE] = 0, this bit is set to 0 on any write to the EntryLo0/1 register, regardless of the value written.
<42:30>	PFNX	R/W	R/W	0x0	When PageGrain[ELPA] = 1, the hardware uses the full address width, with PFNX expanding the PFN field to the full physical address. When PageGrain[ELPA] = 0, the PFNX is always written to 0x0 on a mtc0/ dmtc0 operation to EntryLo0/1. Otherwise, the hardware uses the full address width in all cases. See "Root.PageGrain Register: CP0 Register 5, Select 1" on page 110. • cnMIPS III cores implement a 49-bit physical address requiring a 13-bit PFNX field. • cnMIPS III cores do not support 1KB pages.
<29:6>	PFN	R/W	R/W	—	Page frame number
<5:3>	C	R/W	R/W	—	Coherency attribute of page. Refer to Section 2.8.
<2>	D	R/W	R/W	—	Dirty bit (indicates page is writable).
<1>	V	R/W	R/W	—	Valid bit
<0>	G	R/W	R/W	—	Global bit

Cavium Confidential for Steven Tang ALU 02/27/2014

Guest.Context Register: CP0 Register 4, Select 0

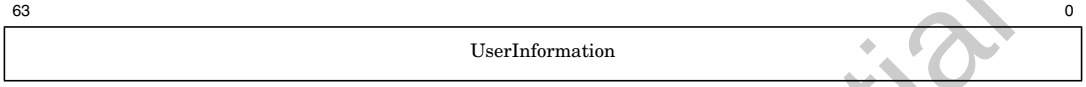
When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<63:23>	PTEBase	R/W	R/W	—	This field is for use by the operating system and is normally written with a value that allows the operating system to use the Context Register as a pointer into the current PTE array in memory.
<22:4>	BadVPN2	RO	R/W	0x0	Bad Virtual Page Number/2 field is written by hardware on a miss.
<3:0>	—	RAZ	RAZ	0x0	Reserved

Guest.UserLocal Register: CP0 Register 4, Select 2

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<63:0>	UserInformation	R/W	R/W	—	This field contains software information that is not interpreted by the hardware.

Guest.PageMask Register: CP0 Register 5, Select 0

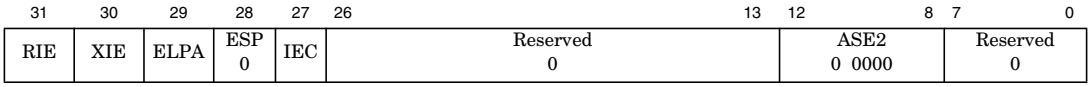
The PageMask register is a read/write register used for reading from and writing to the TLB. When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31:29>	—	RAZ	RAZ	0x0	Reserved. Ignored on write; read as zero.
<28:13>	Mask	R/W	R/W	—	The Mask field is a bit mask in which a 1 bit indicates that the corresponding bit of the virtual address should not participate in the TLB match. CN71XX supports the full mask range, and supports all page sizes 4KB, 8KB, 16KB, ... 64MB, 128MB, and 256MB.
<12:11>	MaskX	RO	RO	0x0	CN71XX does not support 1KB pages, so the MaskX field is not used (write operations are ignored, reads as 0).
<10:0>	—	RAZ	RAZ	0x0	Reserved. Ignored on write; read as zero.

Cavium Confidential For Bowen Yang ALI 02/27/2014

Guest.PageGrain Register: CP0 Register 5, Select 1

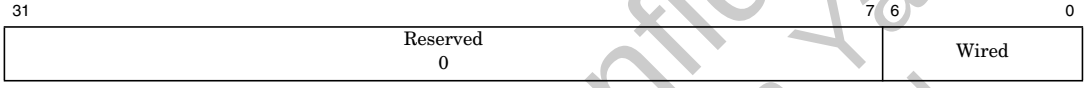
When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31>	RIE	R/W	R/W	0	Read inhibit enable. When set to 1, EntryLo0/1[RI] is enabled. When cleared to 0, EntryLo0/1[RI] is disabled and is not writable by software.
<30>	XIE	R/W	R/W	0	Execute inhibit enable. When set to 1, EntryLo0/1[XI] is enabled. When cleared to 0, EntryLo0/1[XI] is disabled and is not writable by software.
<29>	ELPA	R/W	R/W	0	Enables support for large physical addresses. As PABITS = 49 for CN71XX, ELPA should normally be enabled on CN71XX. See "Root.EntryLo0, Root.EntryLo1 Registers: CP0 Registers 2 and 3, Select 0" on page 108 for description for the only effect of PageGrain[ELPA].
<28>	ESP	RAZ	RAZ	0	1KB pages not implemented.
<27>	IEC	R/W	R/W	0	Enables unique exception codes for the Read-Inhibit and Execute-Inhibit exceptions.
<28:13>	—	RAZ	RAZ	0x0	Reserved
<12:8>	ASE2	RAZ	RAZ	0x0	ASE not implemented.
<7:0>	—	RAZ	RAZ	0x0	Reserved

Guest.Wired Register: CP0 Register 6, Select 0

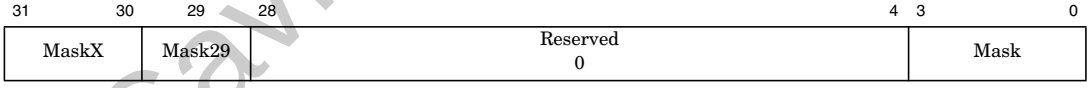
When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeG	Reset Value	Field Description
<31:8>	—	RAZ	RAZ	0x0	Reserved
<7:0>	Wired	R/W	R/W	0x0	TLB wired boundary.

Guest.HWREna Register: CP0 Register 7, Select 0

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

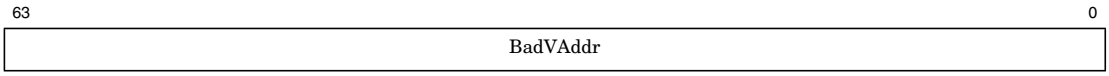


Bit Pos	Field Name	Field TypeG	Field TypeG	Reset Value	Field Description
<31:30>	MaskX	R/W	R/W	0x0	cnMIPS III cores include Cavium-specific instruction RDHWR 31, 30, 29, so a MaskX field exists at <31:30> to enable them. Refer to Section A.4 for the RDHWR description.
<29>	Mask29	R/W	R/W	0	Enables access by the RDHWR instruction to the UserLocal register.
<28:4>	—	RAZ	RAZ	0x0	Reserved
<3:0>	Mask	R/W	R/W	0x0	Each bit in this field enables access by the RDHWR instruction to a particular hardware register.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Guest.BadVAddr Register: CP0 Register 8, Select 0

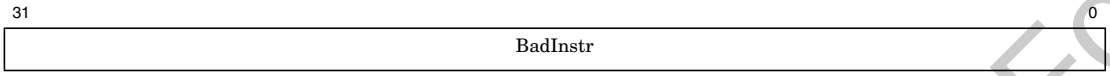
When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<63:0>	BadVAddr	RO	R/W	0x0	Bad virtual address

Guest.BadInstr Register: CP0 Register 8, Select 1

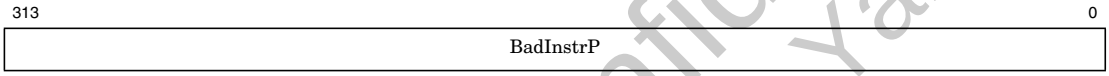
When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31:0>	BadInstr	RO	R/W	—	Contains the instruction that was executing when an exception occurred.

Guest.BadInstrP Register: CP0 Register 8, Select 2

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

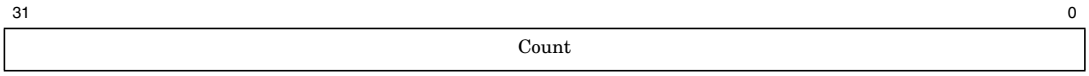


Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31:0>	BadInstrP	RO	R/W	—	Contains the instruction prior to the one that was executing when an exception occurred.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Guest.Count Register: CP0 Register 9, Select 0

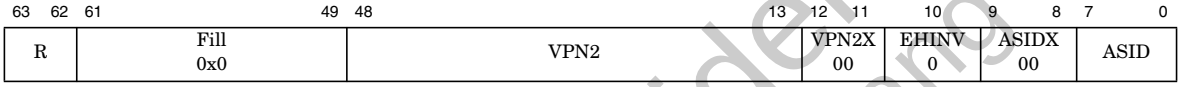
When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31:0>	Count	RO	RO	0x0	Interval counter. The Count register is zeroed during reset and increments by one every cycle. The guest view of Count is computed from Root.Count + Root.GTOffset. Guest write access always generates a GPSI exception. • When Root.GuestCtl0[GT] = 0, mfc0 access or RDHWR from CC causes a GPSI exception. • When Root.GuestCtl0[GT] = 1, read access is permitted with mfc0 or RDHWR from CC. The root cannot directly write Count but can write to Root.GTOffset or Root.Count to change the value of this register.

Guest.EntryHi Register: CP0 Register 10, Select 0

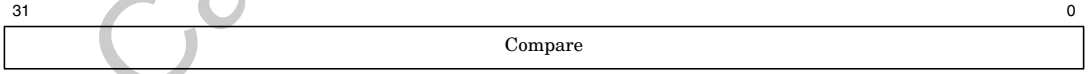
When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<63:62>	R	R/W	R/W	—	Virtual memory region
<61:49>	Fill	RAZ	RAZ	0x0	Fill bits reserved for expansion of the virtual address space.
<48:13>	VPN2	R/W	R/W	—	This field is written by hardware on a TLB exception or on a TLB read.
<12:11>	VPN2X	RAZ	RAZ	0x0	Not used. 1KB pages not implemented.
<10>	EHINV	RAZ	RAZ	0	TLB invalidation not implemented.
<9:8>	ASIDX	RAZ	RAZ	0x0	ASIB field not extended.
<7:0>	ASID	R/W	R/W	—	Address space identifier

Guest.Compare Register: CP0 Register 11, Select 0

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31:0>	Compare	When Root.GuestCtl0[GT] = 1: R/W When Root.GuestCtl0[GT] = 0: GPSI	R/W	—	Interval count compare value.

Cavium Confidential For Baofei Yang 02/27/2014

Guest.Status Register: CP0 Register 12, Select 0

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17 16	15 10	9 8	7	6	5	4 3	2	1	0
CU3 0	CU2	CU1 0	CU0	RP 0	FR 0	RE 0	MX 0	PX	BEV	TS	SR	NMI	ASE 0	Impl 00	IM7...IM2	IM1, IM0	KX	SX	UX	KSU	ERL	EXL	IE

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31>	CU3	RAZ	RAZ	0	CU3 is no longer used and is reserved for future use.
<30>	CU2	RO, R/W	R/W	—	Control access to coprocessor 2. If a guest write request to Guest.Status would cause a change in CU2 and both Root.GuestCtl0[SFC2] = 0 and Root.GuestCtl0Ext[FCD] = 0, a guest software field change (GSFC) exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<29>	CU1	RO, R/W	R/W	0	Controls access to coprocessor 1 (FP and Prefetch Indexed Instructions). If a guest write request to Guest.Status would cause a change in CU1 and both Root.GuestCtl0[SFC1] = 0 and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<28>	CU0	RO, R/W	R/W	—	Control access to coprocessor 0. If a guest write request to Guest.Status would cause a change in CU0 and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<27>	RP	RAZ	RAZ	0	Reduced power mode is not implemented.
<26>	FR	RO, R/W	R/W	0	FR = 1 provides true 64-bit FPU. FR = 0 provides paired register mode with 32 32-bit FP registers or 16 64-bit FP registers with FP registers numbered as 0, 2, 4, ... (only even numbered registers legal for 64-bit FP and FR = 0). If a guest write request to Guest.Status would cause a change in FR and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<25>	RE	RAZ	RAZ	0	Reverse-endian mode is not implemented.
<24>	MX	RAZ	RAZ	0	MDMX™ is not implemented.
<23>	PX	RO, R/W	R/W	—	Enables access to 64-bit operations in User mode, without enabling 64-bit addressing. If a guest write request to Guest.Status would cause a change in PX and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<22>	BEV	RO, R/W	R/W	1	Controls the location of exception vectors. If a guest write request to Guest.Status would cause a change in BEV and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<21>	TS	RO	R/W	0	Indicates that the TLB has detected a match on multiple entries (i.e machine-check).
<20>	SR	RO	R/W	—	Set the first time that a core comes out of reset following a chip soft reset that was due to either CIU_SOFT_RST CSR write (refer to Section TBD) or due to watchdog timer (refer to Section TBD). Otherwise, clear when the core comes out of reset. (Refer also to MIO_RST_BOOT[LBOOT]). If a guest write request to Guest.Status would cause a change in SR and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.

Cavium Confidential For Ruwen Yang ALU 8/27/2014

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<19>	NMI	RO, R/W	R/W	0	Indicates that the entry through the reset exception vector was due to an NMI exception. If a guest write request to Guest.Status would cause a change in NMI and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<18>	ASE	RO	RO	0	MCU ASE is not implemented; reads as zero.
<17:16>	Impl	RO	RO	0x0	Ignored on write and read as zeros.
<15:10>	IM7–IM2	R/W	R/W	—	Interrupt masks. When set, these mask bits enable the corresponding Cause[IP7..IP2] interrupts.
<9:8>	IM1, IM0	R/W	R/W	—	Interrupt Masks. These are used to control software interrupts.
<7>	KX	RO, R/W	R/W	—	Enables access to 64-bit Kernel Segments and use of the XTLB Refill Vector for references to Kernel Segments. If a guest write request to Guest.Status would cause a change in KX and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<6>	SX	RO, R/W	R/W	—	Enables access to 64-bit Supervisor Segments and use of the XTLB Refill Vector for references to Supervisor Segments. If a guest write request to Guest.Status would cause a change in SX and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<5>	UX	RO, R/W	R/W	—	Enables access to 64-bit User Segments, Use of the XTLB Refill Vector for references to User Segments and Execution of instructions which perform 64-bit operations while the processor is operating in User Mode. If a guest write request to Guest.Status would cause a change in UX and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<4:3>	KSU	RO, R/W	R/W	—	The encoding of this field denotes the base operating mode of the processor. If a guest write request to Guest.Status would cause a change in KSU and both Root.GuestCTL0[MC] = 1 and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<2>	ERL	RO, R/W	R/W	1	Error level. If a guest write request to Guest.Status would cause a change in ERL and both Root.GuestCTL0[FCD] = 0 and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<1>	EXL	R/W	R/W	—	Exception level.
<0>	IE	R/W	R/W	—	Interrupt enable.

NOTE: The Dcache (virtual tags only, not physical tags) is invalidated whenever the Status[ERL] value changes.

Cavium Confidential For Bowen Yang 02/27/2014

Guest.IntCtl Register: CP0 Register 12, Select 1

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

31 29	28 26	25 23	22 14	13 10	9 5	4 0
IPTI	IPPCI	IPFDC	MCU ASE 0	Reserved 0	VS 0	Reserved 0

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31:29>	IPTI	RO	RO	0x7	This field is a read-only copy of CvmCtl[IPTI].
<28:26>	IPPCI	RO	RO	0x7	This field is a read-only copy of CvmCtl[IPPCI].
<25:23>	IPFDC	RAZ	RAZ	0x0	Fast debug channel not implemented
<22:14>	MCU ASE	RO	RO	0x0	MCU ASE is not implemented; reads as zero.
<13:10>	—	RAZ	RAZ	0x0	Reserved
<9:5>	VS	RAZ	RAZ	0x0	Vectored interrupt not implemented
<4:0>	—	RAZ	RAZ	0x0	Reserved

Guest.SRSCtl Register: CP0 Register 12, Select 2

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

31 30	29 26	25 22	21 18	17 16	15 12	11 10	9 6	5 4	3 0
Reserved 00	HSS 00 00	Reserved 00 00	EICSS 00 00	Reserved 00	ESS 0000	Reserved 00	PSS 00 00	Reserved 00	CSS 0000

NOTE: SRS map is not implemented.

cnMIPS™ III cores have no GPR shadow sets so the entire SRSCtl register is read as zero.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Guest.Cause Register: CP0 Register 13, Select 0

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

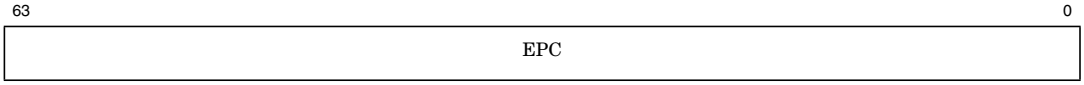
31	30	29 28	27	26	25 24	23	22	21	20 18	17 16	15 13	12 10	9 8	7	6 2	1 0
BD	TI	CE	DC	PCI	ASE 00	IV	WP	FDCI 0	Reserved 0 00	ASE 00	IP7...IP5	IP4...IP2	IP1, IP0	Reserved 0	ExcCode	Reserved 00

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31>	BD	RO	R/W	0	Indicates whether the last exception taken occurred in a branch-delay slot.
<30>	TI	RO	R/W	0	Timer Interrupt; indicates whether timer interrupt is present. Cleared by writing to the Guest.Compare register.
<29:28>	CE	RO	R/W	0x0	Coprocessor unit number referenced when a coprocessor unusable exception is taken.
<27>	DC	RO, R/W	R/W	0	Disable count register. If a guest write request to this register change the value of DC and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<26>	PCI	RO	RO	0	Implemented; up to four performance counters implemented. This bit is set to 1 if the following occur: • the MSB of a performance counter active in the guest context is set to 1 • Root.PerfCntCtl[IE] = 1 • Root.PerfCntCtl[EC] = 0x2 or 0x3.
<25:24>	ASE	RO	RO	0x0	MCU ASE is not implemented; reads as zero.
<23>	IV	RO, R/W	R/W	—	Indicates whether an interrupt exception uses the general exception vector or a special interrupt vector. If a guest write request to this register change the value of IV and Root.GuestCtl0Ext[FCD] = 0, a GSFC exception occurs. Else this field is writable by the guest without causing a GSFC exception.
<22>	WP	R/W	R/W	—	Implemented (two watch registers implemented).
<21>	FDCI	RAZ	RAZ	0	Fast debug channel not implemented.
<20:18>	—	RAZ	RAZ	0x0	Reserved
<17:16>	ASE	RAZ	RAZ	0x0	MCU ASE is not implemented; reads as zero.
<15:13>	IP7...IP5	RO	RO	—	Indicate an interrupt is pending. May be set for timer interrupts (Guest.Cause[TI] is set in this case) and/or performance counter interrupts (Guest.Cause[PCI] is set in this case). Root.CvmCtl[IPTI,IPPCI] select the IP bits used for timer and performance counter interrupts, respectively.
<12:10>	IP4...IP2	RO	RO	—	Indicate an interrupt is pending. may assert due to timer and performance counter interrupts, and may additionally assert due to the CIU interrupt distribution. Refer to Chapter TBD.
<9:8>	IP1, IP0	R/W	R/W	—	Indicate an interrupt is pending. These are used to control the request for software interrupts.
<7>	—	RAZ	RAZ	0	Reserved
<6:2>	ExcCode	RO	R/W	0x0	Exception code. This field is compliant with MIPS® documentation. Please refer to the MIPS® documentation for cause code value assignment. (See Table 2–37.)
<1:0>	—	RAZ	RAZ	0x0	Reserved

Cavium Confidential For Bowen Yang ARD 07/7/2014

Guest.EPC Register: CP0 Register 14, Select 0

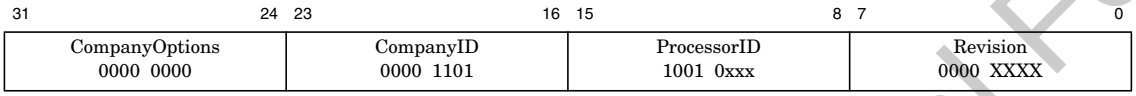
Guest Exception Program Counter register. When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<63:0>	EPC	R/W	R/W	—	Exception program counter.

Guest.PRId Register: CP0 Register 15, Select 0

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31:24>	CompanyOptions	RO	RO	0x0	0x0
<23:16>	CompanyID	RO	RO	0xD	13 (0xD)
<15:8>	ProcessorID	RO	RO	0x96	0x96 = CN71XX.
<7:0>	Revision	RO	RO	0x0	0x0 = pass 1.0, 0x1 = pass 1.1, 0x2 = pass 1.2, 0x8 = pass 2.0, 0x9 = pass 2.1, 0xA = pass 2.2

Guest.EBase Register: CP0 Register 15, Select 1

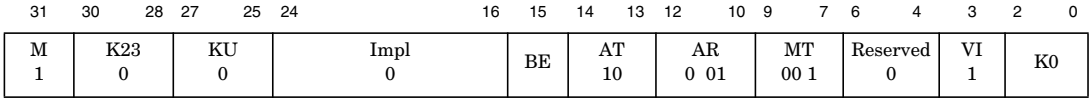
EBase is only supported to KSEG0, KSEG1, and XKPHYS addresses. When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<63:12>	Exception Base	R/W	R/W	0xFFFFFFFF80000	Base address of the exception vectors when Root.Status[BEV] = 0. Bits<29:12> are always writable. Bits<63:30> are writable only when WG = 1.
<11>	WG	R/W	R/W		When set, bits<63:30> of Exception Base are writable
<10>	—	RAZ	RAZ	0	Reserved
<9:0>	CPUNum	RO	R/W	—	Specifies the cnMIPS™ II core ID.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Guest.Config Register: CP0 Register 16, Select 0

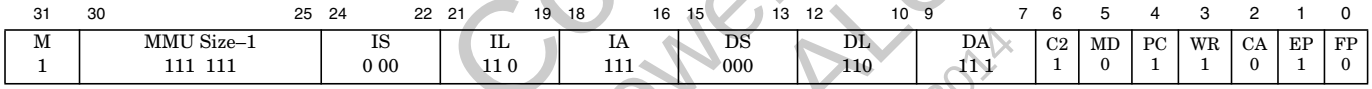
When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception. When Root.GuestCtl[CF] = 0, write accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31>	M	RO	RO	1	Indicates that the Guest.Config1 register is present.
<30:28>	K23	RAZ	RAZ	0x0	Fixed Mapping MMU not implemented.
<27:25>	KU	RAZ	RAZ	0x0	Fixed Mapping MMU not implemented.
<30:16>	Impl	RO	RO	0x0	Read as zero.
<15>	BE	RO	RO	1	$\overline{\text{CvmCtl[LE]}}$
<14:13>	AT	RO	RO	0x2	MIPS 64 with access to all address segments
<12:10>	AR	RO	RO	0x1	Release 2, Release 3/MIPSr3, or Release 5.
<9:7>	MT	RO	RO	0x1	Standard TLB
<6:4>	—	RAZ	RAZ	0x0	Reserved
<3>	VI	RO	RO	1	Icache is virtual.
<2:0>	K0	R/W	R/W	0x2	Field is implemented, and the only effect is on the ordering of guest-mode kseg0 stores. Section 2.8 describes the CN63XX use of the MIPS cacheability and coherency attributes.

Guest.Config1 Register: CP0 Register 16, Select 1

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception. When Root.GuestCtl[CF] = 0, write accesses to this register generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31>	M	RO	RO	1	Indicates that the Guest.Config2 register is present.
<30:25>	MMU Size –1	RO	RO	0x3F	This field is the lower 6 bits of the real guest MMUSize–1; the real guest MMUSize is 255 – Root.CvmVMConfig[RootMMUSizeM1]. "Guest.Config4 Register: CP0 Register 16, Select 4" on page 145 extends this field. Reset value is 0x0 entries.
<24:22>	IS	RO	RO	0x0	The value denotes 64 Icache sets per way per the MIPS specification. The actual Cavium implementation is 16 sets per way.[1]
<21:19>	IL	RO	RO	0x6	128-byte Icache line size
<18:16>	IA	RO	RO	0x7	The value denotes 8-way Icache per the MIPS specification. The actual Cavium implementation is 39-way Icache.[1]
<15:13>	DS	RO	RO	0x0	The value denotes 64 Dcache sets per way per the MIPS specification. The actual Cavium implementation is 8 sets per way.[1]
<12:10>	DL	RO	RO	0x0	128-byte Dcache line size
<9:7>	DA	RO	RO	0x7	The value denotes 8-way Dcache per the MIPS specification. The actual Cavium implementation is 32-way Dcache.[1]

Cavium Confidential For Bowen Yang ALU 2/27/2014

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<6>	C2	RO	RO	1	Coprocessor 2 implemented
<5>	MD	RAZ	RAZ	0	MDMX ASE not implemented
<4>	PC	RO	R/W	0	Up to four Performance Counter registers implemented
<3>	WR	RO	R/W	0	Up to two Watch registers implemented
<2>	CA	RAZ	RAZ	0	MIPS16e not implemented
<1>	EP	RO	RO	0	EJTAG not implemented in the guest context.
<0>	FP	RO	RO	1	FPU implemented

1. This mismatch exists because MIPS does not have an encoding for Cavium's implementation.

Guest.Config2 Register: CP0 Register 16, Select 2

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception. When Root.GuestCtl[CF] = 0, write accesses to this register generate a GPSI exception.

31	30 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
M	TU	TS	TL	TA	SU	SS	SL	SA
1	000	0000	0000	0000	0000	0100	0110	0111

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31>	M	RO	RO	1	Guest.Config 3 is present.
<30:28>	TU	RAZ	RAZ	0x0	Not implemented
<27:24>	TS	RO	RO	0x0	No tertiary cache
<23:20>	TL	RO	RO	0x0	No tertiary cache
<19:16>	TA	RO	RO	0x0	No tertiary cache
<15:12>	SU	RAZ	RAZ	0x0	Not implemented
<11:8>	SS	RO	RO	0x4	1024 sets per way in secondary cache
<7:4>	SL	RO	RO	0x6	128-byte cache line in secondary cache
<3:0>	SA	RO	RO	0x7	The value denotes 8-way set-associative secondary cache per the MIPS specification. The actual Cavium implementation is 4-way set-associative secondary cache.

Cavium Confidential For Bowen Yang ALD 02/27/2014

Guest.Config3 Register: CP0 Register 16, Select 3

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception. When Root.GuestCtl[CF] = 0, write accesses to this register generate a GPSI exception.

31	30	29	28	27	26	25	24	23	22 21	20 18	17	16	15 14
M	BPG	CMGCR	MSAP	BP	BI	SC	PW	VZ	IPLW	MMAR	MCU	ISAOnExc	ISA
1	0	0	0	1	1	0	0	0	0	0	0	0	00

13	12	11	10	9	8	7	6	5	4	3	2	1	0
ULRI	RXI	DSP2P	DSPP	CTXTC	ITL	LPA	VEIC	VInt	SP	CDMM	MT	SM	TL
1	1	0	0	0	0	1	0	0	0	0	0	0	0

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31>	M	RO	RO	1	Config4 is present.
<30>	BPG	RO	RO	0	Big Pages feature not implemented
<29>	CMGCR	RO	RO	0	CM GCR space not implemented
<28>	MSAP	RO	RO	0	MIPS SIMD Architecture (MSA) not implemented
<27>	BP	RO	RO	1	BadInstrP register implemented
<26>	BI	RO	RO	1	BadInstr register implemented
<25>	SC	RO	RO	0	Segment Control not implemented
<24>	PW	RO	RO	0	Hardware Page Table Walk not implemented
<23>	VZ	RO	RO	0	MIPS virtualization ASE implemented
<22:21>	IPLW	RO	RO	0x0	MCU ASE not implemented
<20:18>	MMAR	RO	RO	0x0	microMIPS64 not implemented
<17>	MCU	RO	RO	0	MCU ASE not implemented
<16>	ISAOnExc	RO	RO	0	MIPS64 is used on entrance to an exception vector.
<15:14>	ISA	RO	RO	0x0	Only MIPS64 Instruction Set is implemented.
<13>	ULRI	RO	RO	1	UserLocal register implemented
<12>	RXI	RO	RO	1	Read/execute inhibit function implemented
<11>	DSP2P	RO	RO	0	Revision 2 of the MIPS DSP ASE not implemented
<10>	DSPP	RO	RO	0	MIPS DSP ASE not implemented
<9>	CTXTC	RO	RO	0	ContextConfig is not implemented.
<8>	ITL	RO	RO	0	MIPS IFlowTrace not implemented
<7>	LPA	RO	RO	1	Large physical address support is implemented, and the PageGrain register exists.
<6>	VEIC	RO	RO	0	No external interrupt controller
<5>	VInt	RO	RO	0	Vectored interrupts not implemented
<4>	SP	RO	RO	0	1KB page size not implemented
<3>	CDMM	RO	RO	0	Common Device Memory Map not implemented
<2>	MT	RO	RO	0	MIPS MT ASE not implemented
<1>	SM	RO	RO	0	SmartMIPS™ not implemented
<0>	TL	RO	RO	0	Trace Logic not implemented

Cavium Confidential For Sweeney ALO 04/27/2014

Guest.Config4 Register: CP0 Register 16, Select 4

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception. When Root.GuestCtl[CF] = 0, write accesses to this register generate a GPSI exception.

This register is only present in cnMIPS III.

31	30–24	23–16	15–14	13–8	7–0
M 1	Reserved	KScrExist	MMUExtDef 01	Reserved	MMUSizeExt 0000 0001

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31>	M	RO	RO	1	Config5 is present.
<30:29>	IE	RO	RO	0x0	Reserved
<28>	AE	RO	RO	0	
<27:24>	VTLBSizeExt	RO	RO	0x0	
<23:16>	KScrExist	RO	RO	0x1C	Indicates how many scratch registers are available to guest kernel-mode. Each bit represents a select for Register 31. (Bit 16 represents select 0; bit 23 represents select 7.) If the bit is set, the associated scratch register is implemented and available for kernel-mode software. Guest.Scratch registers exist for select 2, 3, and 4.
<15:14>	MMUExtDef	RO	RO	0x1	MMU Extension Definition. Set to 0x1 (bits <7:0> used as MMUSizeExt).
<13:8>	—	RAZ	RAZ	0x0	Reserved
<7:0>	MMUSizeExt	RO	RO	0x3	Extension of Guest.Config1[MMUSize–1] field. Concatenate this field to the left of the Guest.Config1[MMUSize–1] to indicate the actual root MMUSize–1. Reset value is 0x0 entries.

Guest.Config5 Register: CP0 Register 16, Select 5

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception. When Root.GuestCtl[CF] = 0, write accesses to this register generate a GPSI exception.

This register is only present in cnMIPS III.

31	30	29	28	27–2	1	0
M 0	K 0	CV 0	EVA 0	Reserved 0x0	P32 0	NFExist 0

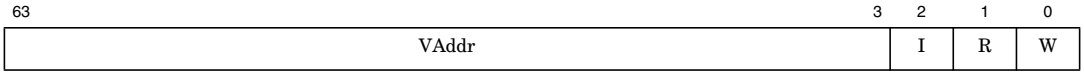
Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31>	M	RO	RO	1	
<30>	K	RO	RO	0x0	
<29>	CV	RO	RO	0	
<28>	EVA	RO	RO	0x0	
<27>	MSAEn	RO	RO	0	
<26:3>	—	RAZ	RAZ	0x0	Reserved
<2>	UFR	RO	RO	0	User-mode FR instructions not allowed.
<1>	P32	RO	RO	0x1	
<0>	NFExist	RO	RO	0x1	

Cavium Confidential For Bowen Yang ALU 02/27/2014

Guest.WatchLo Register: CP0 Register 18, select 0, 1

Two watchpoints can be implemented in the guest context; the first instruction-only and the second data-only. When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

The Root.WatchLo and Root.WatchHi registers are available to the guest when the root sets Guest.Config1[WR] to 1 and Root.WatchHi[WM] = 0x2 (GVA Watch). Otherwise guest access to a WatchLo or WatchHi pair generate a GPSI exception.

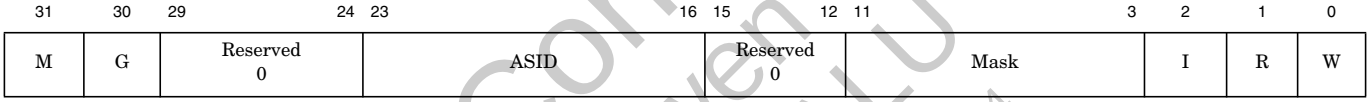


Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<63:3>	VAddr	R/W	R/W	—	Specifies the virtual address to match
<2>	I	R/W	R/W	0	Can be implemented at select = 0 (only the first watchpoint can be an instruction)
<1>	R	R/W	R/W	0	Can be implemented at select = 1 (only the second watchpoint can be data)
<0>	W	R/W	R/W	0	Can be implemented at select = 1 (only the second watchpoint can be data)

Guest.WatchHi Register: CP0 Register 19, select 0, 1

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

Root WatchLo and WatchHi registers are available to the guest when the root sets Guest.Config1[WR] to 1 and Root.WatchHi[WM] = 0x2 (GVA Watch). Otherwise guest access to a WatchLo or WatchHi pair generate a GPSI exception.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31>	M	RO	RO	0	For select 0, M = Root.WatchHi1[WM<1>]. For select 1, M = 0. Up to two watch points are available in the guest context.
<30>	G	R/W	R/W	—	If this bit is one, any address that matches that specified in the WatchLo register will cause a watch exception. If this bit is zero, the ASID field of the WatchHi register must match the ASID field of the EntryHi register to cause a watch exception.
<29:24>	—	RAZ	RAZ	0x0	Reserved
<23:16>	ASID	R/W	R/W	—	ASID value which is required to match that in the EntryHi register if the G bit is zero in the WatchHi register.
<15:12>	—	RAZ	RAZ	0x0	Reserved
<11:3>	Mask	R/W	R/W	—	Implemented in the maximum width
<2>	I	W1C	W1C	0	Set by hardware when an instruction fetch condition matches the values in this watch register pair.
<1>	R	W1C	W1C	0	Set by hardware when a load condition matches the values in this watch register pair.
<0>	W	W1C	W1C	0	Set by hardware when a store condition matches the values in this watch register pair.

Cavium Confidential For Bowen Yang ALU 02/7/2014

Guest.XContext Register: CP0 Register 20, Select 0

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

63 — 42	41 — 40	39 — 4	3 — 0
PTEBase	R	BadVPN2	Reserved 0

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<63:42>	PTEBase	R/W	R/W	—	This field is for use by the operating system and is normally written with a value that allows the operating system to use the Context Register as a pointer into the current PTE array in memory.
<41:40>	R	RO	R/W	0x0	Region field. Contains bits <63:62> of the virtual address.
<39:4>	BadVPN2	RO	R/W	0x0	Bad Virtual Page Number/2 field is written by hardware on a miss.
<3:0>	—	RAZ	RAZ	0x0	Reserved

Guest.PerfCntCtl Register: CP0 Register 25, Select 0, 2, 4, 6

When Root.GuestCtl[CP0] = 0, accesses to this register generates a GPSI exception.

If Guest.Config1[PC] = 1 and Root.PerfCntCtl[EC] = 0x2 or 0x3, then the performance counter is fully accessible by the guest. Otherwise, guest access generates a GPSI exception.

31	30	29 — 23	22 — 16	15	14 — 11	10 — 5	4	3	2	1	0
M	W 1	Impl 0x0	Reserved 0x0	PCTD	EventExt	Event	IE	U	S	K	EXL

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<31>	M	RO	RO	0/1	For select = 0, 2, 4, M = Root.PerfCntCtl*(n+1)*[EC<1>]. For select 6, M = 0. Up to four performance counters are available in the guest context.
<30>	W	RO	RO	1	1 (64-bit counter)
<29:23>	Impl	RO	RO	0x0	
<22:16>	—	RAZ	RAZ	0x0	Reserved
<15>	PCTD	RO	RO	0	
<14:11>	EventExt	R/W	R/W	0x0	The event selection is the concatenation of the Event and EventExt fields, i.e., EventExt|Event.
<10:5>	Event	R/W	R/W	—	Selects the event to be counted by the corresponding Counter Register. See Table 2–38. See also EventExt.
<4>	IE	R/W	R/W	0	Interrupt Enable.
<3>	U	R/W	R/W	—	Enables event counting in User Mode.
<2>	S	R/W	R/W	—	Enables event counting in Supervisor Mode.
<1>	K	R/W	R/W	—	Enables event counting in Kernel Mode.
<0>	EXL	R/W	R/W	—	Enables event counting when the EXL bit in the Status register is one and the ERL bit in the Status register is zero.

Cavium Confidential For Bowen Yang ALU 04/27/2014

Table 2–39 Performance Counter Control Register Events

Value	Event	Meaning
0x0	—	Reserved
0x1	PERF_CNT_CLK	Conditionally clocked cycles (as opposed to count/cvm_count which count even with no clocks)
0x2	PERF_CNT_ISSUE	Instructions issued but not retired
0x3	PERF_CNT_RET	Instructions retired
0x4	PERF_CNT_NISSUE	Cycles no issue
0x5	PERF_CNT_SISSUE	Cycles single issue
0x6	PERF_CNT_DISSUE	Cycles dual issue
0x7	PERF_CNT_IFI	Cycle ifetch issued (but not necessarily commit to pp_mem)
0x8	PERF_CNT_BR	Branches retired, excluding branch likelies
0x9	PERF_CNT_BRMIS	Branch mispredicts, excluding branch likelies
0xA	PERF_CNT_J	Jumps retired, does not include J/JAL
0xB	PERF_CNT_JMIS	Jumps mispredicted, does not include J/JAL
0xC	PERF_CNT_REPLAY	Mem Replays
0xD	PERF_CNT_IUNA	Cycles idle due to unaligned_replays
0xE	PERF_CNT_TRAP	Traps are replays caused by the memory system, as well as exceptions and PERF_CNT_HAZTR's. Branch/jump mispredict do not cause these traps.
0xF	—	Reserved
0x10	PERF_CNT_UULOAD	Unexpected unaligned loads (REPUN=1)
0x11	PERF_CNT_UUSTORE	Unexpected unaligned store (REPUN=1)
0x12	PERF_CNT_ULOAD	Unaligned loads (REPUN=1 or USEUN=1)
0x13	PERF_CNT_USTORE	Unaligned store (REPUN=1 or USEUN=1)
0x14	PERF_CNT_EC	Exec clocks (must set CvmCtl[DISCE] for accurate timing)
0x15	PERF_CNT_MC	Mul clocks (must set CvmCtl[DISCE] for accurate timing)
0x16	PERF_CNT_CC	Crypto clocks (must set CvmCtl[DISCE] for accurate timing)
0x17	PERF_CNT_CSRC	Issue_csr clocks (must set CvmCtl[DISCE] for accurate timing)
0x18	PERF_CNT_CFETCH	Icache committed fetches (demand+prefetch)
0x19	PERF_CNT_CPREF	Icache committed prefetches
0x1A	PERF_CNT_ICA	Icache aliases
0x1B	PERF_CNT_II	Icache invalidates
0x1C	PERF_CNT_IP	Icache parity error
0x1D	PERF_CNT_CIMISS	Cycles idle due to imiss (must set CvmCtl[DISCE] for accurate timing)
0x1E	—	Reserved
0x1F	—	Reserved
0x20	PERF_CNT_WBUF	Number of write buffer entries created
0x21	PERF_CNT_WDAT	Number of write buffer data cycles used (may need to set CvmCtl[DISCE] for accurate counts)
0x22	PERF_CNT_WBUFLD	Number of write buffer entries forced out by loads
0x23	PERF_CNT_WBUFFL	Number of cycles that there was no available write buffer entry (may need to set CvmCtl[DISCE] and CvmMemCtl[MCLKALWYS] for accurate counts)
0x24	PERF_CNT_WBUFTR	Number of stores that found no available write buffer entries
0x25	PERF_CNT_BADD	Number of address bus cycles used (may need to set CvmCtl[DISCE] for accurate counts)
0x26	PERF_CNT_BADDL2	Number of address bus cycles not reflected (i.e. destined for L2) (may need to set CvmCtl[DISCE] for accurate counts)
0x27	PERF_CNT_BFILL	Number of fill bus cycles used (may need to set CvmCtl[DISCE] for accurate counts)

Cavium Confidential For ALD Bowen Yang 02/27/2014

Table 2–39 Performance Counter Control Register Events (Continued)

Value	Event	Meaning	
0x28	PERF_CNT_DDIDS	Number of Dstream DIDs created	
0x29	PERF_CNT_IDIDS	Number of Istream DIDs created	
0x2A	PERF_CNT_DIDNA	Number of cycles that no DIDs were available (may need to set CvmCtl[DISCE] and CvmMemCtl[MCLKALWYS] for accurate counts)	
0x2B	PERF_CNT_LDS	Number of load issues	
0x2C	PERF_CNT_LMLDS	Number of local memory load issues	
0x2D	PERF_CNT_IOLDS	Number of I/O load issues	
0x2E	PERF_CNT_DMLDS	Number of loads that were not prefetches and missed in the cache	
0x2F	—	Reserved	
0x30	PERF_CNT_STS	Number of store issues	
0x31	PERF_CNT_LMSTS	Number of local memory store issues	
0x32	PERF_CNT_IOSTS	Number of I/O store issues	
0x33	PERF_CNT_IOBDMA	Number of IOBDMA operations	
0x34	—	Reserved	
0x35	PERF_CNT_DTLB	Number of dstream TLB refill, invalid, or modified exceptions	
0x36	PERF_CNT_DTLBAD	Number of dstream TLB address errors	
0x37	PERF_CNT_ITLB	Number of istream TLB refill, invalid, or address error exceptions	
0x38	PERF_CNT_SYNC	Number of SYNC stall cycles (may need to set CvmCtl[DISCE] for accurate counts)	
0x39	PERF_CNT_SYNCIOB	Number of SYNCIOBDMA stall cycles (may need to set CvmCtl[DISCE] for accurate counts)	
0x3A	PERF_CNT_SYNCW	Number of SYNCWs or SYNCWSs	
0x3B	PERF_CNT_DUTLB	Number of dstream misses in the µTLB.	**cnMIPS III** only
0x3C	PERF_CNT_IUTLB	Number of istream misses in the µTLB**.**	**cnMIPS III** only
0x3D	PERF_CNT_CDMISS	Cycles idle due to demand (non-prefetch) Dstream misses	**cnMIPS III** only
0x3E -0x3F	—	Reserved	
0x40	PERF_CNT_ERETMIS	D/eret mispredicts	
0x41	PERF_CNT_LIKMIS	Branch likely mispredicts	
0x42	PERF_CNT_HAZTR	Hazard traps due to *MTC0's to CvmCtl, Perf counter control, EntryHi, or CvmMemCtl register	
0x43	PERF_CNT_FPUNIMPTRAP	Number of FP unimplemented traps	
0x44	PERF_CNT_FPHAZARDTRAP	Number of FP hazard traps	
0x45	PERF_CNT_FPARITHEXC	Number of FP Arithmetic Exceptions	
0x51	PERF_CNT_FPMOVC1	Number of retired MTC1, DMTC1, MFHC1, MTHC1, MFC1, DMFC1 instructions	
0x52	PERF_CNT_FPCOPYC1	Number of retired CFC1 and CTC1 instructions	
0x53	PERF_CNT_FPCOMPARE	Number of retired FP C.Cond.fmt instructions	
0x54	PERF_CNT_FPBRANCH	Number of retired BC1T, BC1F, BC1TL, and BC1FL instructions	
0x55	PERF_CNT_FPMOV	Number of retired FP move instructions	
0x56	PERF_CNT_FPABSNEG	Number of retired ABS.fmt and NEG.fmt instructions	
0x57	PERF_CNT_FPADDSUB	Number of retired ADD.fmt and SUB.fmt instructions	
0x58	PERF_CNT_FPCVT	Number of retired FP format conversion instructions (e.g., CVT.S.D)	
0x59	PERF_CNT_FPMUL	Number of retired MUL.fmt instructions	
0x5A	PERF_CNT_FPMADD	Number of retired MADD.fmt variant instructions	

Cavium Confidential For Bowen Yang ALU 02/27/2014

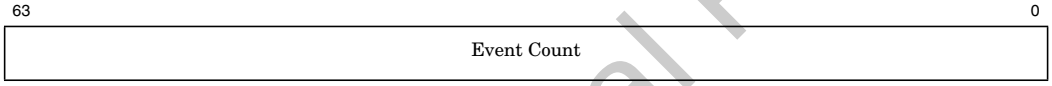
Table 2–39 Performance Counter Control Register Events (Continued)

Value	Event	Meaning
0x5B	PERF_CNT_FPDIVRECIP	Number of retired DIV.fmt and RECIP.fmt instructions
0x5C	PERF_CNT_FPSQRTRSQRT	Number of retired SQRT.fmt and RSQRT.fmt instructions
0x5D	PERF_CNT_FPLOAD	Number of retired FP load instructions
0x5E	PERF_CNT_FPSTORE	Number of retired FP store instructions
0x5F	PERF_CNT_FPALL	Number of retired FP instructions (all)

Guest.PerfCntCtr Register: CP0 Register 25, select 1, 3, 5, 7

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

If Guest.Config1[PC] = 1 and Root.PerfCntCtl[EC] = 0x2 or 0x3, then the performance counter is fully accessible by the guest. Otherwise, guest access generates a GPSI exception.



Increments once for each event that is enabled by the corresponding Control Register. When the MSB is one, a pending interrupt request is ORed with those from other performance counters and indicated by the PCI bit in the Cause register.

To save power, the implementation may not update the performance counter counter registers every cycle when CvmCtl[DCICLK] is clear—the current counter register value may not include all the recent updates.

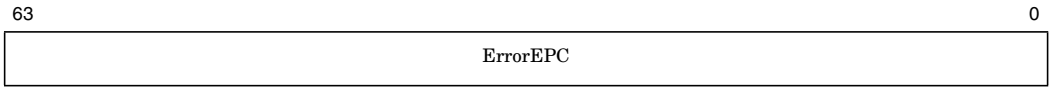
To ensure that a performance counter counter register contains all of the recent updates when CvmCtl[DCICLK] is clear, either:

- issue a MTC0/DMTC0 to the corresponding performance counter control register, or
- issue a MFC0/DMFC0 to the performance counter counter register

Note that with the latter case (a single MFC0/DMFC0), the counter register will update two cycles after the instruction, but the destination GPR of the instruction may not contain the updated value. To get the most recent value of the performance counter counter register when CvmCtl[DCICLK] is clear, execute three consecutive MFC0/DMFC0 operations to the register, all with the same destination GPR.

Guest.ErrorEPC Register: CP0 Register 30, Select 0

The ErrorEPC register contains the virtual address at which instruction processing can resume after servicing an error. When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

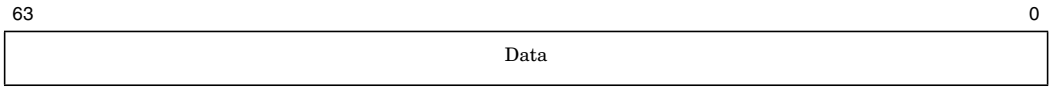


Cavium Confidential For Rowen Yang ALU 08/27/2014

Guest.KScratch1 Register: CP0 Register 31, Select 2

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

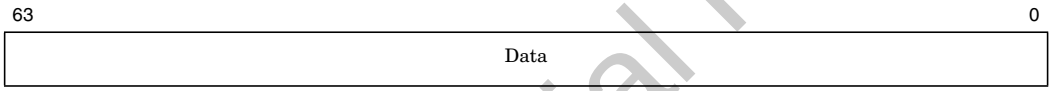
The KScratch1 register is a read/write register available for scratch pad storage by kernel mode software.



Guest.KScratch2 Register: CP0 Register 31, Select 3

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

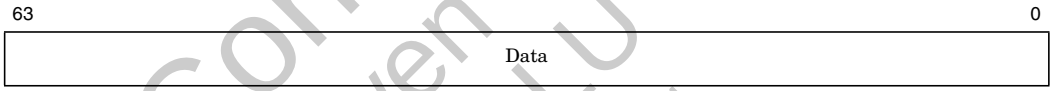
The KScratch2 register is a read/write register available for scratch pad storage by kernel mode software.



Guest.KScratch3 Register: CP0 Register 31, Select 4

When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception.

The KScratch3 register is a read/write register available for scratch pad storage by kernel mode software.

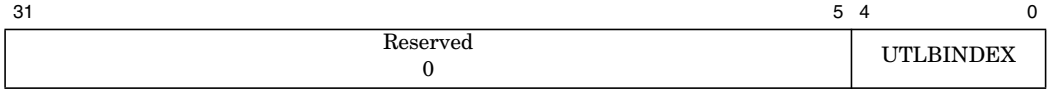


Cavium Confidential For
Bowen Yang
ALU
02/27/2014

2.15.3 Cavium-Specific Coprocessor 0 Registers in Root Mode

Root.UtlbIndex Register: Implemented at reg# = 0, sel = 0 when CvmMemCtl2[UTLBRW] = 1

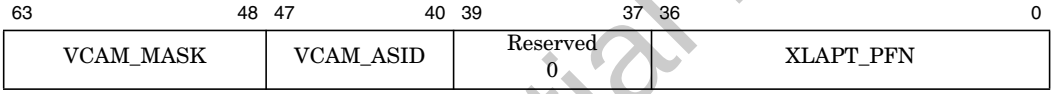
This register is used only to debug the µTLB.



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:5>	—	RAZ	0x0	Reserved
<4:0>	UTLBINDEX	R/W	—	Index into µTLB

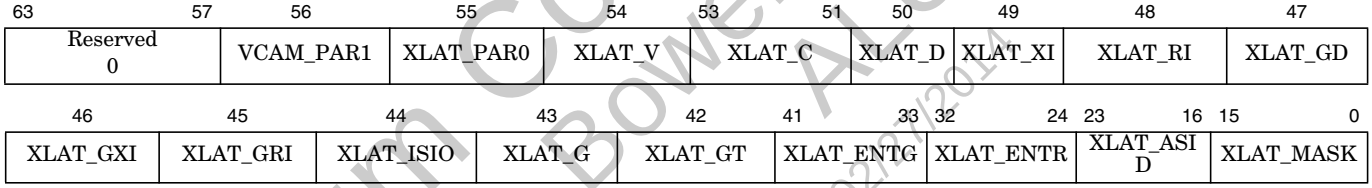
Root.UtlbEntry1 Register: Implemented at reg# = 2, sel = 0 when CvmMemCtl2[UTLBRW] = 1

This register is used only to debug the µTLB



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:48>	VCAM_MASK	R/W	—	Mask bits in µTLB CAM
<47:40>	VCAM_ASID	R/W	—	ASID bits in µTLB CAM
<39:37>	—	RAZ	0	Reserved
<36:0>	XLAT_PFN	R/W	—	PFN bits in in µTLB RAM

Root.UtlbEntry2 Register: Implemented at reg# = 3, sel = 0 when CvmMemCtl2[UTLBRW] = 1



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:57>	—	RAZ	0	Reserved
<56>	XLAT_PAR1	R/W	—	Mask bits in µTLB CAM
<55>	XLAT_PAR0	R/W	—	ASID bits in µTLB CAM
<54>	XLAT_V	R/W	—	Parity 1 bit in µTLB RAM
<53:51>	XLAT_C	R/W	—	Parity 0 bit in µTLB RAM
<50>	XLAT_D	R/W	—	Guest XI bit in µTLB RAM
<49>	XLAT_XI	R/W	—	Guest RI bit in µTLB RAM
<48>	XLAT_RI	R/W	—	Guest D bit in µTLB RAM
<47>	XLAT_GD	R/W	—	Isio bit in µTLB RAM
<46>	XLAT_GXI	R/W	—	ASID bits in µTLB RAM
<45>	XLAT_GRI	R/W	—	G bit in µTLB RAM
<44>	XLAT_ISIO	R/W	—	Gt bit in µTLB RAM

Cavium Confidential For Bowen Yang ALU 02/17/2014

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<43>	XLAT_G	R/W	—	EntryG bits in µTLB RAM
<42>	XLAT_GT	R/W	—	EntryR bits in µTLB RAM
<41:33>	XLAT_ENTG	R/W	—	V bit in µTLB RAM
<32:24>	XLAT_ENTR	R/W	—	D bit in µTLB RAM
<23:16>	XLAT_ASID	R/W	—	C bits in µTLB RAM
<15:0>	XLAT_MASK	R/W	—	XI bit in µTLB RAM

This register is used only to debug the µTLB. XLAT_PAR1 and XLAT_PAR0 are computed as follows:

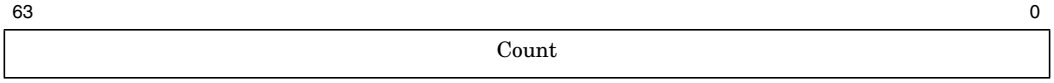
FULL_BIT_ARRAY = {UtlbEntry0[VCAM_R],
UtlbEntry0[VCAM_VPN <41:21> only],
UtlbEntry2[XLAT_GXI],
UtlbEntry2[XLAT_GRI],
UtlbEntry2[XLAT_GD],
UtlbEntry2[XLAT_ISIO],
UtlbEntry2[XLAT_ASID],
UtlbEntry2[XLAT_G],
UtlbEntry2[XLAT_GT],
UtlbEntry2[XLAT_ENTG],
UtlbEntry2[XLAT_ENTR],
UtlbEntry2[XLAT_V],
UtlbEntry2[XLAT_D],
UtlbEntry2[XLAT_C],
UtlbEntry2[XLAT_XI],
UtlbEntry2[XLAT_RI],
UtlbEntry2[XLAT_MASK],
UtlbEntry1[XLAT_PFN]};

UtlbEntry2[XLAT_PAR1] = XOR of all odd array entries of FULL_BIT_ARRAY
UtlbEntry2[XLAT_PAR0] = XOR of all even array entries of FULL_BIT_ARRAY.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Root.CvmCount Register: Implemented at reg# = 9, sel = 6

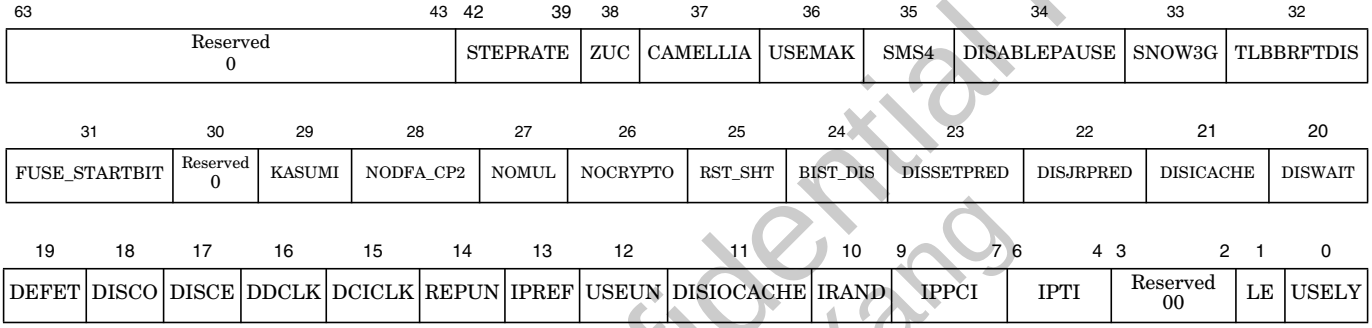
Reset to 0x0. Increments by one every cycle. See "Root.Multicore Debug Register: Implemented at reg# =22, select =0" on page 169 for more details about the behavior of CvmCount during debug.



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:0>	count	RW	0x0	Cycle counter. Value may also be read at RDHWR 31. Refer to Section A.4.

Root.CvmCtl Register: Implemented at reg# = 9, select = 7

Controls the Icache and instruction issue.



Bit Pos	Field Name	Field Type	Reset Value	Field Description	
<63:43>	—	RAZ	0x0	Reserved	
<42:39>	STEPRATE	R/W	0x0	If non-zero, issue an instruction at most every $2^{STEPRATE}$ cycles. If non-zero, Multicore Debug[CvGSTP] and [CvDm] must be zero.	**cnMIPS III** only
<38>	ZUC	RO	1	When set, ZUC operations are enabled. CvmCtl[ZUC] = 0 whenever CvmCtl[NOCRYPTO] = 1	**cnMIPS III** only
<37>	CAMELLIA	RO	1	When set, CAMELLIA operations are enabled. CvmCtl[CAMELLIA] = 0 whenever CvmCtl[NOCRYPTO] = 1	**cnMIPS III** only
<36>	USEMAK	R/W	0	If set, AES operations use MAK for key. USEMAK must not be set when RNM_CTL_STATUS[DIS_MAK] is set. When USEMAK is set, the only legal AES/SMS4 instructions are: CVM_MF_AES_RESINP (both of them) CVM_MT_AES_ENC0 CVM_MT_AES_ENC1	
<35>	SMS4	RO	1	When set, SMS4 operations are enabled. CvmCtl[SMS4] = 0 whenever CvmCtl[NOCRYPTO] = 1.	
<34>	DISABLEPAUSE	R/W	0	When set, the PAUSE instruction is a NOP.	
<33>	SNOW3G	RO	1	When set, SNOW3G and GFM REFLECT operations are enabled. CvmCtl[SNOW3G] = 0 whenever CvmCtl[NOCRYPTO] = 1.	
<32>	TLBBRFTDIS	R/W	0	If clear, following a TLB/XTLB refill exception, conditional branches are forced to predict not-taken until a subsequent mispredict or ERET.	
<31>	FUSE_STARTBIT	RO	1	Always reads as 1.	

Cavium Confidential For Peilen Yang AU 02/27/2014

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<30>	—	RAZ	0	Reserved
<29>	KASUMI	RO	1	When set, KASUMI operations are enabled. CvmCtl[KASUMI] = 0 whenever CvmCtl[NOCRYPTO] = 1.
<28>	NODFA_CP2	RO	1	Read only and reads as 1.
<27>	NOMUL	RO	0	When set, large mul operations are disabled.
<26>	NOCRYPTO	RO	0	When set, crypto operations are disabled.
<25>	RST_SHT	RO	0	When set, reset delays are short (should be 0).
<24>	BIST_DIS	RO	0	When set, BIST is disabled (should be 0).
<23>	DISSETPRED	R/W	0	When set, disables power-saving feature of Icache.
<22>	DISJRPRED	R/W	0	When set, JALs no longer push the jump stack (i.e. the jump stack remains frozen).
<21>	DISICACHE	R/W	0	When set, Icache is disabled (operates in pass-through mode). Normally, this bit should not be set. When DISICACHE = 0, and is being written to 1, one of the following must be true: ● The linear instruction stream including the MTC0/DMTC0 instruction that writes CvmCtl[DISICACHE] = 1 is running out of I/O space or DMSEG ● The MTC0/DMTC0 instruction that writes CvmCtl[DISICACHE] = 1 is followed immediately with a SYNCI instruction (in the same cache block). ● There was a one cache line backward branch before the MTC0/DMTC0 instruction that writes CvmCtl[DISICACHE] = 1
<20>	DISWAIT	R/W	0	When set, WAIT is disabled
<19>	DEFET	R/W	0	When set, fetch under fill is disabled
<18>	DISCO	R/W	0	When set, disable the conditional clocks (i.e. always clock) mul/crypto
<17>	DISCE	R/W	0	When set, disable the conditional clocks (i.e. always clock) issue
<16>	DDCLK	R/W	0	When set, disable the conditional clocks (i.e. always clock) exec
<15>	DCICLK	R/W	0	When set, disable the conditional clocks (i.e. always clock) issue unit CSRs
<14>	REPUN	R/W	0	When set, LBX+LBUX+LH+LHX+LHU+LHUX+LW+LWX+LWU+LWUX+ LD+LDX+SH+SW+SD unaligned accesses handled by hardware
<13>	IPREF	R/W	0	When set, instruction prefetching is disabled. Cavium recommends that instruction prefetching be disabled until the memory system and TLB has been fully initialized.
<12>	USEUN	R/W	0	When set, CVM ULW+ULD+USW+USD insts, else MIPS LWL+LWR+LDL+ LDR+SWL+SWR+SDL+SDR
<11>	DISIOCACHE	R/W	0	When set, the iocache feature is disabled, which cause noncompliance for EJTAG and may cause extra I/O-space fetches.
<10>	IRAND	R/W	0	When set, the Icache uses random replacement, else use “bitmask LRU”
<9:7>	IPPCI	R/W	0x7	Selects the Cause[IP] bit to which the Performance Counter Interrupt request is merged. Legal values are CvmCtl[IPPCI] = 2 .. 7, where CvmCtl[IPPCI]=*n* selects Cause[IP<*n*>] for the interrupt. IntCtl[IPPCI] is a read-only copy of CvmCtl[IPPCI].
<6:4>	IPTI	R/W	0x7	Selects the Cause[IP] bit to which the Timer Interrupt request is merged. Legal values are CvmCtl[IPTI] = 2 .. 7, where CvmCtl[IPTI]=*n* selects Cause[IP<*n*>] for the interrupt. IntCtl[IPTI] is a read-only copy of CvmCtl[IPTI].
<3:2>	—	RAZ	0x0	Reserved

Cavium Confidential For Baiwen Yang ALU 02/27/2014

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<1>	LE	R/W	0	When set, the core defaults to LE (i.e. Config[BE] is a read-only copy of the inverse of CvmCtl[LE].) Any CvmCtl write instruction (a DMTC0 or MTC0) that modifies the CvmCtl[LE] value must follow the following rules or may cause unpredictable results: **1.** The instruction must not be in a delay slot. **2.** The instruction must be colocated next to a NOP instruction within the same 64-bit word.
<0>	USELY	R/W	0	When clear use *L for unaligned opcode for BE, *R for LE. When set unaligned opcode is always the *L opcodes

Root.UtlbEntry0 Register: Implemented at reg# = 10, sel = 0 when CvmMemCtl2[UTLBRW] = 1

This register is used only to debug the µTLB.

63 61	60 52	51	50 42	41 5	4	3	2 1	0
Reserved	VCAM_ENTR	VCAM_GTI	VCAM_ENTG	VCAM_VPN	VCAM_G	VCAM_GT	VCAM_R	VCAM_VAL

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:61>	—	RAZ	0	Reserved
<60:52>	VCAM_ENTR	R/W	—	EntryR in µTLB CAM
<51>	VCAM_GTI	R/W	—	GTI bit in µTLB CAM
<50:42>	VCAM_ENTG	R/W	—	EntryG in µTLB CAM
<41:5>	VCAM_VPN	R/W	—	VPN bits in µTLB CAM
<4>	VCAM_G	R/W	—	G bit in µTLB CAM
<3>	VCAM_GT	R/W	—	Gt bit in µTLB CAM
<2:1>	VCAM_R	R/W	—	R bits in µTLB CAM
<0>	VCAM_VAL	R/W	—	Valid bit in µTLB CAM

Cavium Confidential For Bowen Yang ALU 02/27/2014

Root.PowThrottle Register: Power Throttle Register, Implemented at reg# = 11, select = 6

63 56	55 48	47 40	39 32	31 29	28	27	26 24	23 16	15 8	7 0
MAXPOW	POWER	THROTT	HRMPOWADJ	Reserved 0	OVRRD	DISTAG	PERIOD	POWLIM	MAXTHR	MINTHR

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:56>	MAXPOW	RO	—	In combination with HRMPOWADJ, indicates the maximum power of the cnMIPS core. This is the typical maximum value for POWER. See HRMPOWADJ. HRMMAXPOW = MAXPOW – HRMPOWADJ A constant for a given core, but may be different for different chips and/or cores.
<55:48>	POWER	RO	—	An estimate of the current power consumption by the core. The POWER value returned is an average, not instantaneous. The average is weighted exponentially, where the half-life of the weighting function is (177000 >> PERIOD) cycles. (Every (1024 >> PERIOD) cycles, POWER = (POWER × 255/256) + (P/256), where P is the estimated average power over the last (1024 >> PERIOD) cycles.) POWER will typically fall in the range 0 – HRMMAXPOW, with POWER equal to 0 indicating minimum power, and POWER equal to HRMMAXPOW indicating the maximum power.
<47:40>	THROTT	RO	—	Throttling value used during prior interval. Throttling forces a CPU core idle for a portion of time, which will reduce power consumption. When THROTT is equal to 0, a core is idle most of the time and consumes minimum power. When THROTT is equal to 0xFF, a core is never idled to reduce power. Increasing THROTT from 0 to 0xFF decreases the forced idle time linearly. The idling is forced at the end of the (1024 >> PERIOD) cycle interval—no idling occurs at the beginning of an interval. This forced idling cannot occur for more than (1024 >> PERIOD) consecutive cycles. The hardware generally uses a THROTT value that is as large as possible (in order to maximize performance) subject to the following constraints (in priority order): • iMINTHR ≤ THROTT ≤ MAXTHR • Power limits from the POWLIM feedback control system.
<39:32>	HRMPOWADJ	RO	—	In combination with MAXPOW, indicates the maximum power of the cnMIPS core. Refer to MAXPOW. A constant for a given core, but may be different for different chips and/or cores.
<31:29>	—	RAZ	—	Must be zero.
<28>	OVRRD	R/W	0	Must be cleared to 0.
<27>	DISTAG	R/W	0	Must be cleared to 0.
<26:24>	PERIOD	R/W	0x2	Selects the interval over which power is limited as follows: 0x0 Power limit interval is 1024 cycles 0x1 Power limit interval is 512 cycles 0x2 Power limit interval is 256 cycles 0x3 Power limit interval is 128 cycles 0x4..0x7 Reserved Smaller PERIOD values make the core idle for more cycles fewer times, or more coarsely, when necessary. Larger PERIOD values make the core idle for fewer cycles more times, or more fine-grained, when necessary. The PERIOD values should be set the same across all cores within a single chip. Changes to PERIOD can cause power glitches in the core for up to two subsequent intervals.

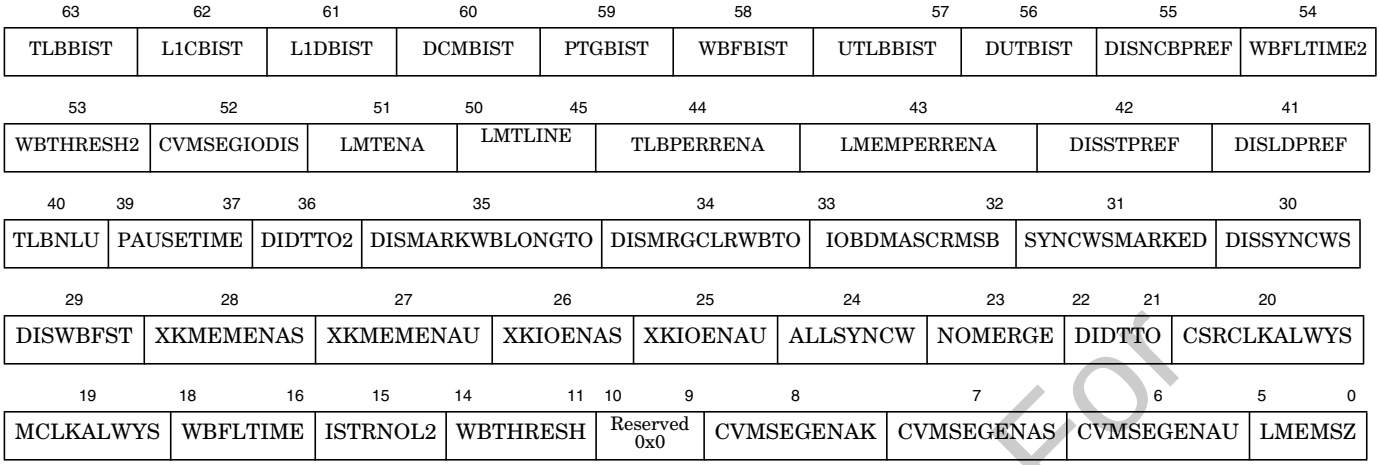
Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<23:16>	POWLIM	R/W	0xFF	A power limiter for the core. A limiter of the power consumption of the core. This power limiting is implemented by a closed-loop feedback control system for the POWER power approximation. The direct output of the POWLIM feedback control system is the THROTT value. The power consumed by the core (estimated currently by the POWER value) is an indirect output of the POWLIM feedback control system. POWLIM is not used by the hardware when iMINTHR is equal to MAXTHR. MAXTHR and iMINTHR threshold requirements always supersede POWLIM limits. (For maximum POWLIM feedback control freedom, set MINTHR = 0 and MAXTHR = 0xFF.) POWLIM equal to 0 forces the core to consume near minimum power, and POWLIM ≥ HRMMAXPOW typically disables any power limiting by POWLIM. (If HRMPOWADJ ≠ 0, power may still be limited by hardware throttling logic, though.) Increasing the POWLIM value from 0 to HRMMAXPOW increases the power that the core is allowed to consume linearly (roughly) from minimum to maximum.
<15:8>	MAXTHR	R/W	0xFF	Maximum allowed THROTT value. MAXTHR must be ≥ MINTHR.
<7:0>	MINTHR	R/W	0x0	Minimum allowed THROTT value. MINTHR must be ≤ MAXTHR. iMINTHR is the internal MINTHR value used by the hardware power throttling logic: iMINTHR = (HRMPOWADJ ≠ 0) ? 0 : MINTHR

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Root.CvmMemCtl Register: Cvm Memory Control Register, Implemented at reg# = 11, select = 7

Controls the cnMIPS™ cores' memory system. Also, see Table 2–34 for error handling bits.



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63>	TLBBIST	RO	0	1 = BIST fail, 0 = BIST pass
<62>	L1CBIST	RO	0	1 = BIST fail, 0 = BIST pass
<61>	L1DBIST	RO	0	1 = BIST fail, 0 = BIST pass
<60>	DCMBIST	RO	0	1 = BIST fail, 0 = BIST pass
<59>	PTGBIST	RO	0	1 = BIST fail, 0 = BIST pass
<58>	WBFBIST	RO	0	1 = BIST fail, 0 = BIST pass
<57>	UTLBBIST	RO	0	1 = BIST fail, 0 = BIST pass
<56>	DUTBIST	RO	0	1 = BIST fail, 0 = BIST pass
<55>	DISNCBPREF	R/W	0	1 = NCB prefetches (from SSO) are disabled and silently dropped. 0 = NCB prefetches (from SSO) are allowed.
<54>	WBFLTIME2	R/W	0	Used in conjunction with WBFLTIME field (see below)
<53>	WBTHRESH2	R/W	1	Used in conjunction with WBTHRESH field (see below)
<52>	CVMSEGIODIS	R/W	0	Used to disable non-kernel accesses to CVMSEG I/O. When set: • root mode CVMSEG I/O accesses cause address-error exceptions when in root-user and root-supervisor modes • guest mode CVMSEG I/O accesses cause Root.GuestCtl0[GExcCode] = GCvmSegAcc exceptions when in guest-user, guest-supervisor, or guest-kernel modes and a guest-mode address-error exception does not occur.
<51>	LMTENA	RAZ	0	Reserved.
<50:45>	LMTLINE	R/W	—	This is the cacheline in CVMSEG LM used in wide atomic store operations (see Section 2.23.2). LMTLINE must not change value in a wide atomic memory write sequence.
<44>	TLBPERRDIS	R/W	0	When set, TLB parity error checking is disabled.

Cavium Confidential For Bowen Yang ALU 6/27/2014

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<43>	LMEMPERRENA	R/W	0	When set, CVMSEG LM parity errors are enabled. To enable CVMSEG LM parity errors, software must follow these constraints: • Whenever CVMSEG LM is enabled with a non-zero CvmMemCtl[LMEMSZ], or if CvmMemCtl[LMEMSZ] is increased when CVMSEG LM is enabled, write all the newly-allocated CVMSEG LM bytes to zero before loading them. This constraint is not needed if it is always true, for all newly allocated 64-bit words, that no bytes in the word are loaded from before all bytes in the word are stored to. ("Stored to" could be via an IOBDMA.) • Deadlock must be avoided. The hardware cannot recover from CVMSEG LM parity errors, and the parity error remains following the associated Cache Error on Data Access exception. The Cache Error on Data Access exception handler should check if the way/set indicate the block is in CVMSEG LM. If yes, software recovery may be required if the error is not to be considered fatal.
<42>	DISSTPREF	R/W	0	If set, NUDGE/WRITEBACK_INVALIDATE, NUDGE_WB, EVICT_SOON, L2, DONT_WRITE_BACK, PREPARE_FOR_STORE, and PREPARE_FOR_STORE_THROUGH prefetch operations become NOPs.
<41>	DISLDPREF	R/W	0	If set, NORMAL and NOTL2 prefetch operations become NOPs.
<40>	TLBNLU	R/W	0	TLB replacement policy: 0 = bitmask LRU; 1 = NLU. • Bitmask LRU maintains a mask of recently used TLB entries and avoids them as new entries are allocated. • NLU simply guarantees that the next allocation is not the last used TLB entry. This field selects the TLB replacement algorithm of the root TLB: bitmask LRU or NLU. If the root TLB uses bitmask LRU, then the guest TLB uses NLU. If the root TLB uses NLU, then the guest TLB uses bitmask LRU.
<39:37>	PAUSETIME	R/W	0x0	Selects the bit in the counter used for releasing a PAUSE. This counter trips every $2^{(8+PAUSETIME)}$ cycles. If not already released, the cnMIPS III core will always release a given PAUSE instruction within $2^{(8+PAUSETIME)}$. If the counter trip happens to line up, the cnMIPS III core may release the PAUSE instantly.
<36>	DIDTTO2	R/W	0x0	This field is an extension of CvmMemCtl[DIDTTO]. See the description for "DIDTTO" on page 161.
<35>	DISMARKWBLONGTO	R/W	0	If set, marked write-buffer entries time out the same as other entries; if clear, marked write-buffer entries use the maximum timeout.
<34>	DISMRGCLRWBTO	R/W	0	If set, a merged store does not clear the write-buffer entry timeout state.
<33:32>	IOBDMASCRMSB	R/W	0x0	Two bits that are the MSBs of the resultant CVMSEG LM word location for an IOBDMA. The other 8 bits come from the SCRADDR field of the IOBDMA.
<31>	SYNCWSMARKED	R/W	0	If set, SYNCWS and SYNCS only order marked stores; if clear, SYNCWS and SYNCS only order unmarked stores. SYNCWSMARKED has no effect when DISSYNCWS is set.
<30>	DISSYNCWS	R/W	0	If set, SYNCWS acts as SYNCW and SYNCS acts as SYNC.
<29>	DISWBFST	R/W	0	If set, no stall happens on write buffer full. **NOTE:** This bit should always be cleared to 0.
<28>	XKMEMENAS [1]	R/W	0	If set (and SX set), supervisor-level loads/stores can use XKPHYS addresses with VA<48>==0

Cavium Confidential For Beverly Yang 2021/8/14

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<27>	XKMEMENAU [1]	R/W	0	If set (and UX set), user-level loads/stores can use XKPHYS addresses with VA<48>==0
<26>	XKIOENAS [1]	R/W	0	If set (and SX set), supervisor-level loads/stores can use XKPHYS addresses with VA<48>==1
<25>	XKIOENAU [1]	R/W	0	If set (and UX set), user-level loads/stores can use XKPHYS addresses with VA<48>==1
<24>	ALLSYNCW	R/W	0	If set, all stores act as SYNCW (NOMERGE must be set when this is set) RW, reset to 0 if set, all stores act as SYNCW (NOMERGE must be set when this is set).
<23>	NOMERGE	R/W	0	If set, no stores merge, and all stores reach the coherent bus in order.
<22:21>	DIDTTO [2]	R/W	0x0	{CvmMemCtl[DIDTTO2], CvmMemCtl[DIDTTO]} selects the bit in the counter for DID timeouts. Taken together, these fields are encoded as follows: 0 = 2^{31}, 1 = 2^{30}, 2 = 2^{29}, 3 = 2^{14}, 4 = 2^{28}, 5 = 2^{27}, 6 = 2^{26}, 7 = 2^{25}. Actual timeout is between 1× and 2× this interval. For example, with {CvmMemCtl[DIDTTO2], CvmMemCtl[DIDTTO]} = 3, expiration interval is between 16K and 32K core clock cycles.
<20>	CSRCKALWYS	R/W	0	If set, the (mem) CSR clock never turns off.
<19>	MCLKALWYS	R/W	0	If set, the core's memory clock never turns off.
<18:16>	WBFLTIME	R/W	0x4	{WBFLTIME2,WBFLTIME} (aka WBFLTIME) selects the write buffer flush timeout WBFLTIME is a 5-bit signed value that selects the bit in a counter used for write buffer flush timeouts. (WBFLTIME+11) is the bit position in an internal counter used to determine expiration. A write buffer entry expires between 1x and 2x this interval. For example, with WBFLTIME = 0, a write buffer entry will expire between 2K and 4K cycles after it is allocated. With WBFLTIME = -1 (i.e. WBFLTIME2=1, WBFLTIME=0xF), a write buffer entry will expire between 1K and 2K cycles after it is allocated. Legal values range from a min of -5 (WBFLTIME2=0x1,WBFLTIME=0x3) up to max of +7 (WBFLTIME2=0x0,WBFLTIME=0x7).
<15>	ISTRNOL2	R/W	0	If set, do not put Istream in the L2 cache. This bit determines whether the core services L1 instruction cache misses with LDI or LDT CMI transactions. See Section 3.4
<14:11>	WBTHRESH	R/W	0x8	The write buffer threshold is a concatenation of the two fields: <WBTHRESH2, WHTHRESH> The legal range of the write buffer threshold is 2 - 22. Values 0, 1, and 23-31 are illegal. The reset value of the threshold is 12 (= 0xC, WBTHRESH2 = 0x0, WBTHRESH = 0xC).
<10:9>	—	RAZ	0x0	Reserved
<8>	CVMSEGENAK[1]	R/W	0	If set, CVMSEG is available for load/store operations in kernel/debug mode. CVMSEGENAK must be enabled when either CVMSEGENAS or CVMSEGENAU are set. Whenever CVMSEGENAK transitions 0→1, software must also invalidate the Dcache if it is possible that any CVMSEG virtual addresses are currently present in the data cache. CVMSEG virtual addresses are KSEG3 virtual addresses in the range 0xFFFF FFFF FFFF 8000 to 0xFFFF FFFF FFFF BFFF. Either an L1 Dcache Virtual Tag Invalidate or an L1 Dcache Invalidate can be used to remove the CVMSEG addresses from the data cache (refer to the CACHE instruction definition in Section A.4).
<7>	CVMSEGENAS[1]	R/W	0	If set, CVMSEG is available for load/store operations in supervisor mode.

Cavium Confidential Pre-Release 02/27/2014

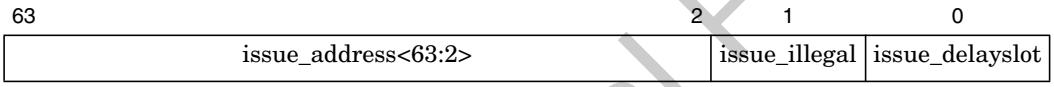
Bit Pos	Field Name	Field Type	Reset Value	Field Description
<6>	CVMSEGENAU[1]	R/W	0	If set, CVMSEG is available for load/store operations in user mode.
<5:0>	LMEMSZ	R/W	0x0	Size of local memory in cache blocks; 54 (6912 bytes) is max legal value.

1. Note that both CvmMemCtl[CVMSEGENA*] and CvmMemCtl[XK*] enable the corresponding range (CVMSEG and XKPHYS with VA<48>==1) only for loads/store operations, and NOT for instruction references.
2. Note that DID time-outs are the only source of bus-error exceptions. These bus-error exceptions may not be recoverable, but the handler can extract information for diagnostic purposes.

Root.PCDebug Register: Implemented at reg# = 16, select = 0

This register contains program counter debug information.

NOTE: Unlike other CP0 registers, the only access to this register is via L2C_COP0_ADR read operations. See L2C_COP0_ADR and L2C_COP0_DAT in Section 3.4.



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:2>	issue_address <63:2>	RO	—	This field can contain debug information as follows: • virtual address<63:2> (PC) of the next instruction to issue (5a in pipeline) • PC of an instruction executing/replaying • PC being filled into the instruction cache • unpredictable Bits<58:50> are copies of bit<49>.
<1>	issue_illegal	RO	—	Set when issue_address is an illegal PC.
<0>	issue_ delayslot	RO	—	Set when issue_address is in a delayslot (prior instruction can be either taken or not taken).

Cavium Confidential For Bowen Yan ALU 02/27/2014

Root.PCDebug1 Register: Implemented at reg# = 16, select = 1

This register contains program counter debug information.

NOTE: Unlike other CP0 registers, the only access to this register is via L2C_COP0_ADR read operations. See L2C_COP0_ADR and L2C_COP0_DAT in Section 3.4.

63	0
fetch_address<63:0>	

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:0>	fetch_address<63:0>	RO	—	VA <63:0> (PC) being fetched from the instruction cache (3a in pipeline). Bits<58:50> are copies of bit<49>. Bits<1:0> are RAZ.

Root.PCDebug2 Register: Implemented at reg# = 16, select = 2

This register contains program counter debug information.

NOTE: Unlike other CP0 registers, the only access to this register is via L2C_COP0_ADR read operations. See L2C_COP0_ADR and L2C_COP0_DAT in Section 3.4.

63	2	1	0
fill_address<63:2>		fill_illegal	Reserved 0

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:2>	fill_address<63:2>	RO	—	Virtual address<63:2> being filled into instruction cache (4a in pipeline). Valid when Root.PCDebug3[waiting_for_ifill] = 1. Bits<58:50> are copies of bit<49>.
<1>	fill_illegal	RO	—	Set when fill_address is an illegal PC.
<0>	—	RAZ	—	Reserved

Cavium Confidential For Bowen Yang ALU 02/12/14

Root.PCDebug3 Register: Implemented at reg# = 16, select = 3

This register contains program counter debug information.

NOTE: Unlike other CP0 registers, the only access to this register is via L2C_COP0_ADR read operations. See L2C_COP0_ADR and L2C_COP0_DAT in Section 3.4.

63 … 3	2	1	0
Reserved 0	mem_stall	waiting_for_pfill	waiting_for_ifill

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:3>	—	RAZ	—	Reserved.
<2>	mem_stall_3a	RO	—	Stall term from L1 memory system.
<1>	waiting_for_pfill	RO	—	When waiting_for_ifill = 1, indicates whether the instruction-cache fill operation is due to a prefetch operation.
<0>	waiting_for_ifill	RO	—	Set when there is an outstanding instruction-cache fill operation.

Root.PCDebug4 Register: Implemented at reg# = 16, select = 4

This register contains program counter debug information.

NOTE: Unlike other CP0 registers, the only access to this register is via L2C_COP0_ADR read operations. See L2C_COP0_ADR and L2C_COP0_DAT in Section 3.4.

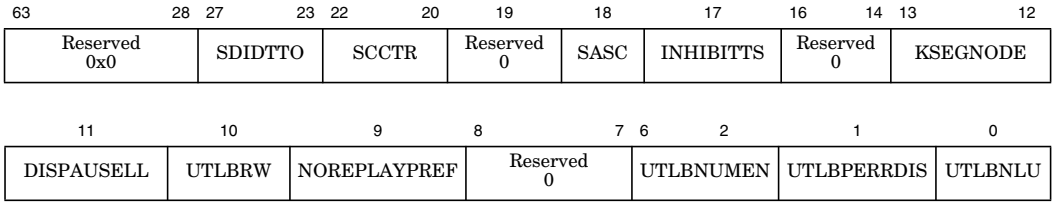
63	62 … 55	54 … 49	48 … 0
Reserved 0	commit_ASID	commit_address<63:59>	commit_address<48:0>

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63>	—	RAZ	—	Reserved.
<62:55>	commit_ASID	RO	—	ASID of commit_address.
<54:49>	commit_address <63:59>	RO	—	VA<63:59> of the last committed instruction (11a in pipeline).
<48:0>	commit_address <48:0>	RO	—	VA<48:0> of last committed instruction (11a in pipeline). Bits<1:0> are RAZ.

Cavium Confidential For Bowen Zang ALU 2/27/2014

Root.CvmMemCtl2 Register: Implemented at reg# = 16, select = 6

Also, see Table 2–34 for error handling bits.



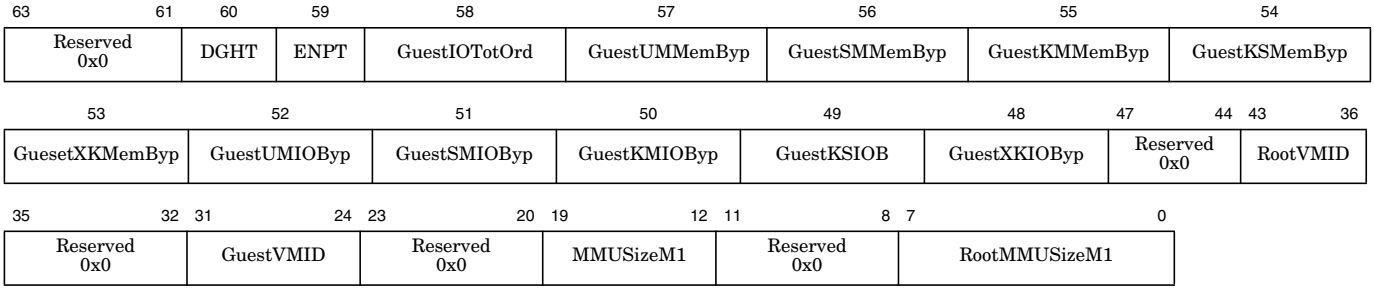
Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:28>	—	RAZ	0x0	Reserved.
<27:23>	SDIDTTO	R/W	0x10	Not used by the core hardware. Valid range is 0xC–0x1F.
<22:20>	SCCTR	RO	0	Store-conditional cache miss counter. Counts the number of consecutive store conditional operations that locally failed solely because the cache block was not present in the Dcache. Always 0x0 following SC/SCD execution when SASC = 1. When SASC=0: • SCCTR can range 0–7 after SC/SCD execution. The core increments the counter whenever it locally fails a SC/SCD operation solely because the cache block was not present in the Dcache, and clears it otherwise. • When SCCTR = 7, the core sends SC/SCD operations that are not locally known to succeed or fail to L2C for final success/fail resolution, as if SASC = 1. • When SCCTR ≠ 7, the core sends only SC/SCD operations that are not locally known to succeed or fail to L2C if the cache block is present in the Dcache when the SC/SCD executes. For the latest SCCTR value, a *MFC0 CvmMemCtl2 must issue a minimum of three cycles after the last SC/SCD operation.
<19>	—	RAZ	0	Reserved.
<18>	SASC	R/W	0	Send all store conditional operations. When set, the core sends all store-conditional operations that are not locally known to succeed or fail (based only on CPU-local information that does not include whether the block was written by another agent, which is indicated by whether the cache block was invalidated from the Dcache) to L2C for final success/failure determination. When clear, the core may (or may not) locally fail a store conditional operation solely because the cache block was not present in the data cache when the store-conditional operation executes. See SCCTR.
<17>	INHIBITTS	R/W	0	Inhibit multi-TLB-entry-match machine checks. When set, a multi-match machine check cannot occur as the result of TLBW* execution and Status[TS] will never be set by hardware. In general, the TLB hardware might malfunction in unpredictable ways when multiple matches are present in a TLB. Consequently, this bit should be set only in very controlled circumstances. Cavium recommends that this bit is ever set only during a full context switch of all guest TLB entries by the root software (i.e. hypervisor), and that this root software is certain that root TLB misses cannot occur while this switch code is executing, and that this root software is also certain that the guest TLB cannot have more than one TLB entry that matches any given GVA after the switch is complete.
<16:14>	—	RAZ	0x0	Reserved.

Cavium Confidential For Binyen Yang 02/27/2014

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<13:12>	KSEGNODE	RAZ	0x0	Reserved.
<11>	DISPAUSELL	R/W	0x0	When set to 1, PAUSE will not exit on invalidation signal after a load-locked instruction
<10>	UTLBRW	R/W	0x0	When set to 1, Index, EntryLo0, EntryLo1, and EntryHi COP0 registers map to UtlbIndex, UtlbEntry1, UtlbEntry2, and UtlbEntry0 registers, respectively. Typical use of this field is to debug the µTLB.
<9>	NOREPLAYPREF	R/W	0x0	When set to 1, PREF and PREFX instructions become a NOP when they miss in the µTLB. When set to zero, PREF and PREFX instructions replay on µTLB miss. The PREF & PREFX hint = 24 is not affected by this field.
<8:7>	Reserved	RAZ	0x0	Reserved
<6:2>	UTLBNUMEN	R/W	0x1F	Number of µTLB Entries minus 1. 0x1 in this field means 2 entries in the µTLB, Reset value is 0x1F or 31 (so µTLB resets to 32 entries). Illegal value is 0x0 (minium 2 entries needed in the µTLB).
<1>	UTLBPERRDIS	R/W	0x0	When set, µTLB parity errors are disabled.
<0>	UTLBNLU	R/W	0x0	µTLB replacement policy: 0 = bitmask LRU; 1 = NLU. This field selects between the µTLB replacement policies: bitmask LRU or NLU. Bitmask LRU maintains a mask of recently used µTLB entries and avoids them as new entries are allocated. NLU simply guarantees that the next allocation is not the last used µTLB entry.

Cavium Confidential
Bowen Yang
ALU
02/27/2014

Root.CvmVMConfig Register: Implemented at reg# = 16, select = 7



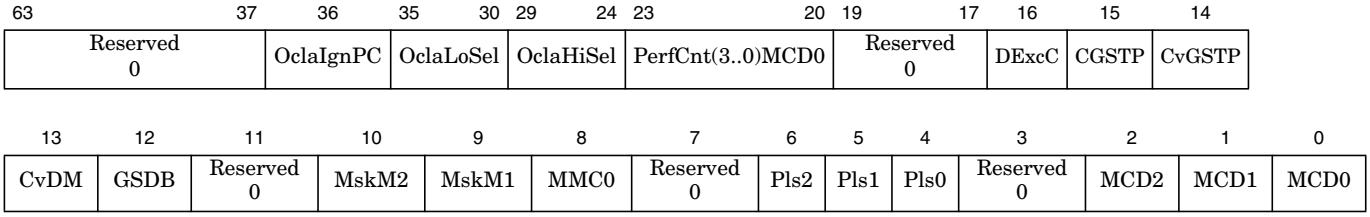
Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:61>	—	RAZ	0x0	Reserved.
<60>	DGHT	R/W	0	Disable guest hole translation . Hole translation (i.e. boot-bus hole conversion) is the conversion of a PA/GPA with bit<48> = 0 and bits<39:28> = 0x001 to a boot-bus address 0x1_0000_1XXX_XXXX, where: • The Xs are <27:0> of the PA/GPA DGHT determines whether this conversion occurs during the first phase of guest address translation (i.e. during the conversion of a GVA to a GPA): 0 = the hole translation occurs during GPA creation 1 = the hole translation does not occur during GPA creation Hole translation always occurs during RVA→RPA and GPA→RPA conversions. Hole GPAs are only possible when DGHT = 1. A hole GPA, like other GPAs with bit<48> = 0, is subject to a memory root translation bypass (i.e. GuestXKMemByp, GuestKSMemByp, GuestKMMemByp, GuestSMMemByp, or GuestUMMemByp).
<59>	ENPT	R/W	0	Enable pause instruction trap. When set to 1, enable guest pause instructions to trap to root with GPSI.
<58>	GuestIOTotOrd	R/W	0	Reserved.
<57>	GuestUMMemByp	R/W	0	Root translation bypass for XUSEG or USEG GVAs to memory GPAs. When set, if the guest translation from an XUSEG or USEG GVA produces a GPA with GPA<48> = 0 (after guest hole translation, if any), the translation bypasses root translation (i.e. the RPA is the GPA).
<56>	GuestSMMemByp	R/W	0	Root translation bypass for XSSEG or KSSEG GVAs to Memory GPAs. When set, if the guest translation from an XSSEG or KSSEG GVA produces a GPA with GPA<48> = 0 (after guest hole translation, if any), the translation bypasses root translation (i.e. the RPA is the GPA).
<55>	GuestKMMemByp	R/W	0	Root translation bypass for XKSEG/KSEG3 GVAs to Mem GPAs. When set, if the guest translation from an XKSEG or KSEG3 GVA produces a GPA with GPA<48> = 0 (after guest hole translation, if any), the translation bypasses root translation (i.e. the RPA is the GPA)
<54>	GuestKSMemByp	R/W	0	Root translation bypass for KSEG0/KSEG1 GVAs to Mem GPAs. When set, if the guest translation from a KSEG0 or KSEG1 GVA produces a GPA with GPA<48> = 0 (after guest hole translation, if any), the translation bypasses root translation (i.e. the RPA is the GPA)

Cavium Confidential For Power Point Only 02/25/2014

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<53>	GuestXKMemByp	R/W	0	Root translation bypass for XKPHYS GVAs to Mem GPAs. When set, if the guest translation from an XKPHYS GVA produces a GPA with GPA<48> = 0 (after guest hole translation, if any), the translation bypasses root translation (i.e. the RPA is the GPA)
<52>	GuestUMIOByp	R/W	0	Root translation bypass for XUSEG or USEG GVAs to I/O GPAs. When set, if the guest translation from an XUSEG or USEG GVA produces a GPA with GPA<48> = 1 (after guest hole translation, if any), the translation bypasses root translation (i.e. the RPA is the GPA)
<51>	GuestSMIOByp	R/W	0	Root translation bypass for XSSEG or KSSEG GVAs to I/O GPAs. When set, if the guest translation from an XSSEG or KSSEG GVA produces a GPA with GPA<48> = 1 (after guest hole translation, if any), the translation bypasses root translation (i.e. the RPA is the GPA)
<50>	GuestKMIOByp	R/W	0	Root translation bypass for XKSEG/KSEG3 GVAs to I/O GPAs. When set, if the guest translation from an XKSEG or KSEG3 GVA produces a GPA with GPA<48> = 1 (after guest hole translation, if any), the translation bypasses root translation (i.e. the RPA is the GPA)
<49>	GuestKSIOByp	R/W	0	Root translation bypass for KSEG0/KSEG1 GVAs to I/O GPAs. When set, if the guest translation from a KSEG0 or KSEG1 GVA produces a GPA with GPA<48> = 1 (after guest hole translation, if any), the translation bypasses root translation (i.e. the RPA is the GPA)
<48>	GuestXKIOByp	R/W	0	Root translation bypass for XKPHYS GVAs to I/O GPAs. When set, if the guest translation from an XKPHYS GVA produces a GPA with GPA<48> = 1 (after guest hole translation, if any), the translation bypasses root translation (i.e. the RPA is the GPA)
<47:44>	—	RAZ	0x0	Reserved.
<43:36>	RootVMId	R/W	0x0	Reserved.
<35:32>	—	RAZ	0x0	Reserved.
<31:24>	GuestVMId	R/W	0x0	Reserved.
<23:20>	—	RAZ	0x0	Reserved.
<19:12>	MMUSizeM1	RO	0xFF	The sum of the sizes of the guest and root TLBs – 1.
<11:8>	—	RAZ	0x0	Reserved.
<7:0>	RootMMUSizeM1	R/W	0xFF	Determines the size of the root TLB. Determines the MMUSizeM1 values in Root.Config1, Root.Config4, Guest.Config1, and Guest.Config4. The core has logically separate guest and root TLBs, with MMUSizem1 + 1 total TLB entries partitioned between them. The number of entries in the root TLB is RootMMUSizem1 + 1, leaving MMUSizeM1 – RootMMUSizeM1 in the guest TLB. Out of hardware reset, the guest TLB is empty, and the root TLB has all TLB entries. The N TLB entries available in each TLB are indexed 0..N–1 at all times. The performance cost of changing RootMMUSizeM1 might be high (e.g. the data cache and µTLB might be flushed), so RootMMUSizeM1 should not be changed too frequently. When RootMMUSizeM1 does change, one of the TLBs gains TLB entries, and all the newly acquired entries must be written (via TLBWI/TLBGWI in the context of the gaining TLB) immediately after.

Cavium Confidential
Robin Yang
2014-04-04

Root.Multicore Debug Register: Implemented at reg# =22, select =0



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:37>	—	RAZ	0x0	Reserved
<36>	OclaIgnPC	R/W	0	Ignore PC updates. 1 = changes only in ocla_data<63:53,0> cause OCLA to receive a new debug entry; changes to ocla_data<52:1> do not. 0 = any changes to ocla_data<63:0> cause OCLA to receive a new debug entry. OclaIgnPC only affects debug selects < 0x20.
<35:30>	OclaLoSel	R/W	0x0	OCLA low-priority debug select. See NOTE 1: on page 170.
<29:24>	OclaHiSel	R/W	0x0	OCLA high-priority debug select. OclaHiSel<5> (bit<29>) must be 0. See NOTE 1: on page 170.
<23>	PerfCnt3MCD0	R/W	0	Performance counter 3 debug exception enable. When set, the hardware generates an MCD0 pulse whenever <63> of the corresponding performance counter counter register changes value from 0 to 1.
<22>	PerfCnt2MCD0	R/W	0	Performance counter 2 debug exception enable. When set, the hardware generates an MCD0 pulse whenever <63> of the corresponding performance counter counter register changes value from 0 to 1.
<21>	PerfCnt1MCD0	R/W	0	Performance counter 1 debug exception enable. When set, the hardware generates an MCD0 pulse whenever <63> of the corresponding performance counter counter register changes value from 0 to 1.
<20>	PerfCnt0MCD0	R/W	0	Performance counter 0 debug exception enable. When set, the hardware generates an MCD0 pulse whenever <63> of the corresponding performance counter counter register changes value from 0 to 1.
<19:17>	—	RAZ	0x0	Reserved
<16>	DExcC	RO	0	Set to one when Debug[DExcCode] is valid and should be interpreted, i.e. is set for debug mode exceptions.
<15>	CGSTP	R/W	0	If set, Count does not increment when GSTOP is set
<14>	CvGSTP	R/W	0	If set, CvmCount does not increment when GSTOP is set
<13>	CvDM	R/W	0	If set, CvmCount does not increment when Debug[DM] set
<12>	GSDB	R/W	0	If one, SDBBP execution pulses global MCD0 wire
<11>	—	RAZ	0	Reserved
<10>	MskM2	R/W	0	When set, pulses on the global MCD2 wire set MCD2.
<9>	MskM1	R/W	0	When set, pulses on the global MCD1 wire set MCD1.
<8>	MskM0	R/W	0	When set, pulses on the global MCD0 wire set MCD0.
<7>	—	RAZ	0	Reserved
<6>	Pls2	RAZ/W	0	Writing a 1 pulses global MCD2 wire.
<5>	Pls1	RAZ/W	0	Writing a 1 pulses global MCD1 wire.
<4>	Pls0	RAZ/W	0	Writing a 1 pulses global MCD0 wire.
<3>	—	RAZ	0	Reserved
<2>	MCD2	R/W1C	0	Reset to zero. If set, enter debug mode.
<1>	MCD1	R/W1C	0	Reset to zero. If set, enter debug mode.
<0>	MCD0	R/W1C	0	Reset to zero. If set, enter debug mode.

Cavium Confidential For Power Vang ALU 9/27/2014

NOTE 1: The core preferentially sends OCLA new debug entries for OclaHiSel before it sends OCLA new debug entries for OclaLoSel. Except when both OclaHiSel and OclaLoSel are 0x0, OclaHiSel should never equal OclaLoSel.

OclaHiSel/OclaLoSel encodings:

0x00 = Low power state. OCLA will not receive a new debug entry

0x02 = Last committed PC

```
ocla_data<71>    = 0
ocla_data<70:65> = low 6 bits of EBase[CPUNum]
ocla_data<64>    = 1 if entry is for OclaHiSel, 0 if entry is for OclaLoSel
ocla_data<63>    = set if an enabled performance counter 3 event occurs on a given cycle
ocla_data<62>    = set if an enabled performance counter 2 event occurs on a given cycle
ocla_data<61>    = set if an enabled performance counter 1 event occurs on a given cycle
ocla_data<60:55> = low 6 bits of EntryHi[ASID] (either root or guest)
ocla_data<54:49> = PC<63:58> (RVA or GVA)
ocla_data<48:2>  = PC<48:2>
ocla_data<1>     = set if PC is illegal
ocla_data<0>     = set if an enabled performance counter 0 event occurs on a given cycle
```

0x21 = Context[PTEBase] plus modes

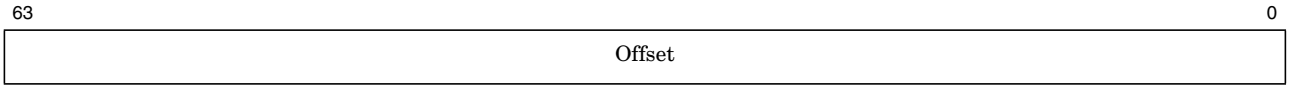
```
ocla_data<71>    = 0
ocla_data<70:65> = low 6 bits of EBase[CPUNum]
ocla_data<64>    = 1 if entry is for OclaHiSel, 0 if entry is for OclaLoSel
ocla_data<63:23> = Context[PTEBase] (Guest.Context if guest mode, else Root.Context)
ocla_data<22:8>  = all 0s
ocla_data<7>     = set if guest mode, else clear
ocla_data<6>     = Status[EXL] (Guest.Status if guest mode, else Root.Status)
ocla_data<5>     = Status[ERL] (Guest.Status if guest mode, else Root.Status)
ocla_data<4>     = Status[UX] (Guest.Status if guest mode, else Root.Status)
ocla_data<3>     = Status[SX] (Guest.Status if guest mode, else Root.Status)
ocla_data<2>     = Status[KX] (Guest.Status if guest mode, else Root.Status)
ocla_data<1:0>   = mode:
                   0x3 = debug
                   0x2 = user
                   0x1 = supervisor
                   0x0 = kernel
```

0x22 = XContext[PTEBase] plus modes

```
ocla_data<71>    = 0
ocla_data<70:65> = low 6 bits of EBase[CPUNum]
ocla_data<64>    = 1 if entry is for OclaHiSel, 0 if entry is for OclaLoSel
ocla_data<63:42> = XContext[PTEBase] (Guest.Context if guest mode, else Root.Context)
ocla_data<41:8>  = all 0s
ocla_data<7>     = set if guest mode, else clear
ocla_data<6>     = Status[EXL] (Guest.Status if guest mode, else Root.Status)
ocla_data<5>     = Status[ERL] (Guest.Status if guest mode, else Root.Status)
ocla_data<4>     = Status[UX] (Guest.Status if guest mode, else Root.Status)
ocla_data<3>     = Status[SX] (Guest.Status if guest mode, else Root.Status)
ocla_data<2>     = Status[KX] (Guest.Status if guest mode, else Root.Status)
ocla_data<1:0>   = mode:
                   0x3 = debug
                   0x2 = user
                   0x1 = supervisor
                   0x0 = kernel
```

Cavium Confidential For Bowen Yang ALU 09/27/2014

Root.CvmCountOffset Register: Implemented at reg# = 22, select = 2



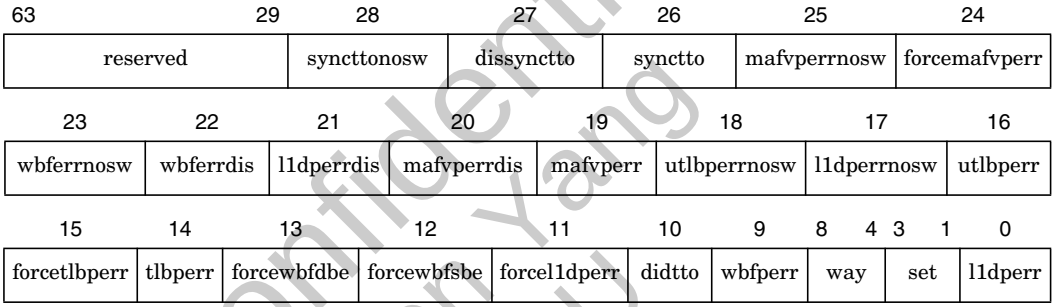
Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:0>	Offset	R/W		Offset added to Root.CvmCount to create Guest.CvmCount.

Root.ErrCtl Register: Implemented at reg# = 26, select = 0

Table 2–40 summarizes the ErrCtl register behavior. Also, see Table 2–34 for error handling bits.

Table 2–40 ErrCtl Behavior

Error	Field(s) Written by Hardware
Load to cached memory has parity error	l1dperr, set, way
Bus error exception due to DID timeout	didtto
Bus error exception due to write buffer parity error	wbfperr



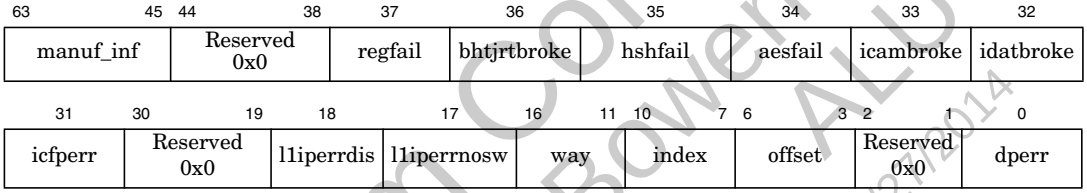
Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:29>	—	RAZ	0	Reserved
<28>	syncttonosw	R/W	0	0 = software receives a bus-error exception when a commit/sync counter timeout occurs. 1 = software does not receive a bus-error exception when a commit/sync counter timeout occurs.
<27>	dissynctto	R/W	0	When set to 1, the commit/sync counter timeout is infinite. A synctto event is generally serious, so this field often should be cleared to 0.
<26>	synctto	R/W1C	0	Commit/sync counter timeout. Root.CvmMemCtl[DIDTTO,DIDTTO2] select the timeout length. DIDTTO bus-error exceptions can be asynchronous (i.e. could happen at any instruction). When hardware sets synctto, it also clears the commit/sync counter. This field can be set without exception when syncttonosw = 1.
<25>	mafvperrnosw	R/W	1	When cleared to 0, software is invoked on MAF Vtag parity error
<24>	forcemafvperr	WR0	0	When set to 1, hardware forces a MAF Vtag parity error on the next MAF Vtag read. Hardware clears this bit after it inserts the error
<23>	wbferrnosw	R/W	1	When cleared to 0, software is invoked on Write Buffer correctable error.
<22>	wbferrdis	R/W	0	When set to 1, ECC check on Write Buffer is disabled
<21>	l1dperrdis	R/W	0	When set to 1, parity check on Dcache is disabled
<20>	mafvperrdis	R/W	0	When set to one, MAF Vtag parity error check is disabled

Cavium Confidential For ALU Power Yang 02/27/2014

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<19>	mafvperr	R/W1C	0	Hardware sets this bit on a load MAF Vtag parity error
<18>	utlbperrnosw	R/W	1	When cleared to 0, trap handler invoked on µTLB parity error. Invocation suppressed when set to one.
<17>	l1dperrnosw	R/W	0	When cleared to 0, trap handler invoked on data cache parity error. Invocation suppressed when set to one.
<16>	utlbperr	R/W1C	0	Hardware sets this bit on a µTLB parity error machine-check. This bit is helpful during a machine check exception.
<15>	forcetlbperr	WR0	0	Write of 1 causes bad parity to be written into TLB on the next TLB write. Hardware clears this bit after it inserts the error.
<14>	tlbperr	R/W1C	0	Hardware sets this bit on a TLB parity error machine-check. This bit is helpful during a machine check exception.
<13>	forcewbfdbe	WR0	0	Writing this bit to 1 causes a write buffer double bit ECC error on next write buffer read. Hardware clears this bit after it inserts the error.
<12>	forcewbfsbe	WR0	0	Writing this bit to 1 causes a write buffer single bit ECC error on next write buffer read. Hardware clears this bit after it inserts the error.
<11>	forcel1dperr	WR0	0	Writing this bit to 1 causes a Dcache parity error on next Dcache read. Hardware clears this bit after it inserts the error.
<10>	didtto	R/W1C	0	Asserted on DID timeout. This bit is helpful during a bus error exception.
<9>	wbferr	R/W1C	0	Asserted on write buffer parity error. This bit is helpful during a bus error exception.
<8:4>	way	RO	0x0	Way in Dcache
<3:1>	set	RO	0x0	Set (VA<9:7>) in DCache
<0>	l1dperr	R/W1C	0	Asserted on DCache parity error. This bit is helpful to determine whether the instruction or data caches caused a CacheErr exception.

Root.CacheErr (Icache) Register: Implemented at reg# = 27, select = 0

Implemented to latch Icache error information. Also, see Table 2–34 for error handling bits.

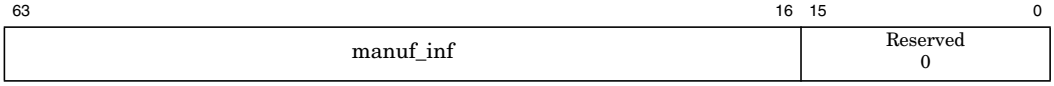


Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:45>	manuf_inf	RO	0	Manufacturing information; these bits are unpredictable and may change.
<44:38>	—	RAZ	0	Reserved
<37>	regfail	RO	0	When set, register file has a fatal error (can only be set in manufacturing mode).
<36>	bhtjrtbroke	RO	0	When set, BHT or JRT has a fatal error.
<35>	hshfail	RO	0	When set, HSH ROM BIST has a fatal error.
<34>	aesfail	RO	0	When set, the AES Unit has a fatal error.
<33>	icambroke	RO	0	When set, the icache tags have a fatal error.
<32>	idatbroke	RO	0	When set, the icache data has a fatal error.
<31>	icfperr	R/W1C	0	When set to 1, all icache fills have bad parity; this field is cleared by hardware on parity error.
<30:19>	—	RAZ	0	Reserved
<18>	l1iperrdis	R/W	0	When set to 1, icache parity checking is disabled
<17>	l1iperrnosw	R/W	0	When set to zero, trap handler invoked on instruction cache parity error. Invocation suppressed when set to one.

Cavium Confidential For Bowen Yang ALU 02/21/2014

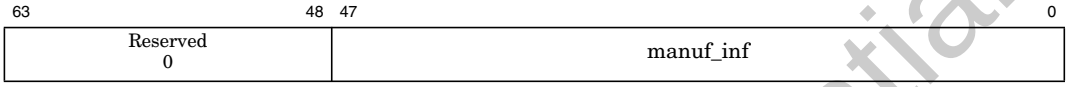
Bit Pos	Field Name	Field Type	Reset Value	Field Description
<16:11>	way	R/W	0	Contains way (0–36) after parity error
<10:7>	index	R/W	0	Contains <10:7> of address after parity error.
<6:3>	offset	R/W	0	Contains <6:3> of address after parity error.
<2:1>	—	RAZ	0	Reserved
<0>	dperr	R/W	0	Data parity error

Root.CacheErr (Dcache) Register: Implemented at reg# = 27, select = 1



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:16>	manuf_inf	RO	0x0	Manufacturing information; these bits are unpredictable and might change.
<15:0>	—	RAZ	0	Reserved

Root.ICacheDebug Register: Implemented at reg# = 27, select = 2



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:48>	—	RAZ	0	Reserved
<47:0>	manuf_inf	RO	0x0	Manufacturing information; these bits are unpredictable and might change.

Root.TagLo Register (Icache): Implemented at reg# = 28, select = 0 (Icache)

Icache implemented.

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:62>	R	R/W	0x0	<63:62> of virtual address
<61:60>	—	RAZ	0x0	Reserved
<59:52>	asid	R/W	0x0	ASID
<51:49>	—	RAZ	0x0	Reserved
<48:11>	tag	R/W	0x0	<48:10> of the virtual address
<10:3>	—	RAZ	0x0	Reserved
<2>	Gt	R/W	0	Gt bit, set for a GVA, clear for an RVA
<1>	G	R/W	0	G bit
<0>	valid	R/W	0	valid bit

Cavium Confidential For Bowen Yang ALU 02/27/2014

Root.TagLo Register (Dcache): Implemented at reg# = 28, select = 2 (Dcache)

Dcache implemented.

63	62 61 60	59 52	51 49	48 10	9 8	7	6	5	4	3	2	1	0
R	Reserved 0	ASID	Reserved 0	vtag	Reserved 0	LM_AND_LMA_A 1	GT 1	GTI 1	LM_NAND_LMA 1	LMA 1	LM 1	G 1	valid 1

Because the CVMSEGLM range overlaps the KSEG3 range, both can exist simultaneously in the Dcache. Thus, there are three types of cache lines that are differentiated using the LM and LMA bits:

LM	LMA	Cacheline Type
0	0	normal (neither KSEG3 nor CVMSEGLM)
0	1	illegal
1	0	KSEG3
1	1	CVMSEGLM

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:62>	R	RO	0x0	<63:62> of virtual address
<61:60>	—	RAZ	0x0	Reserved; must be zero.
<59:52>	ASID	RO	0x0	ASID
<51:49>	—	RAZ	0x0	Reserved; must be zero.
<48:10>	vtag	RO	0x0	<48:10> of the virtual address
<9:8>	—	RAZ	0x0	Reserved; must be zero.
<7>	LM_AND_LMA_A	RO	0	Logical AND of the LM and LMA bits.
<6>	GT	RO	0	GT bit, set for a GVA, clear for an RVA.
<5>	GTI	RO	0	A second copy of the GT bit.
<4>	LM_NAND_LMA	RO	0	Inverse of LM_AND_LMA_A.
<3>	LMA	RO	0	LMA bit. See description above.
<2>	LM	RO	0	LM bit. See description of above.
<1>	G	RO	0	G bit
<0>	valid	RO	0	Valid bit for the virtual tag

Root.TagLo1 Register (Dcache): Implemented at reg# = 28, select = 4 (Dcache)

Dcache implemented.

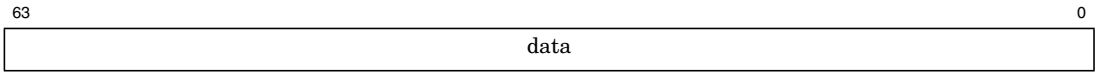
63 18	17 9	8 0
Reserved 0	ENTR	ENTG

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:18>	—	RO	0x0	Reserved.
<17:9>	ENTR	RO	Undefined	Cavium use only. These bits are unpredictable and might change.
<8:0>	ENTG	RO	Undefined	Cavium use only. These bits are unpredictable and might change.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Root.DataLo Register (Icache): Implemented at reg# = 28, select = 1 (Icache)

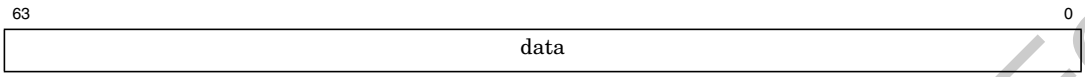
Icache implemented.



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:0>	data	R/W	0x0	Icache data <63:0>.

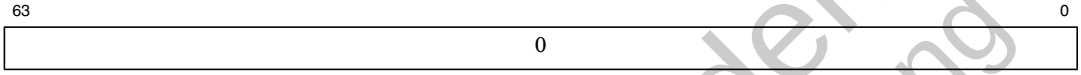
Root.DataLo Register (Dcache): Implemented at reg# = 28, select = 3 (Dcache)

The 64 bits read from the CACHE instruction. Dcache implemented.



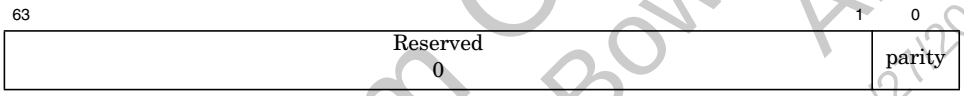
Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:0>	data	RO	0x0	Data from the Dcache.

Root.TagHi Register (Icache): Implemented at reg# = 29, select = 0 (Icache)



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:0>	—	RAZ	0x0	

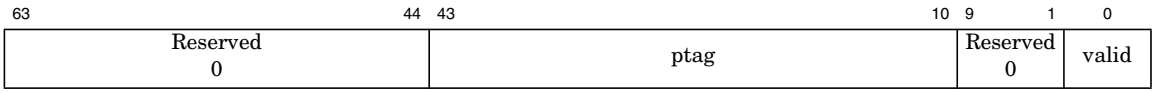
Root.DataHi Register (Icache): Implemented at reg# = 29, select = 1 (Icache)



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:1>	—	RAZ	0x0	Reserved; must be zero.
<0>	parity	R/W	0	Icache parity bit

Cavium Confidential For Bowen Yang ALU 02/21/2014

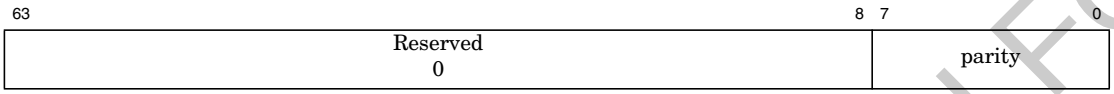
Root.TagHi Register (Dcache): Implemented at reg# = 29, select = 2 (Dcache)



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:44>	—	RAZ	0x0	Reserved; must be zero.
<43:10>	ptag	RO	0	Bits<43:10> of the physical address.
<9:1>	—	RAZ	0x0	Reserved; must be zero.
<0>	valid	RO	0	Valid bit for physical tag./

Root.DataHi Register (Dcache): Implemented at reg# = 29, select = 3 (Dcache)

The eight parity bits read from the CACHE instruction. Dcache implemented.



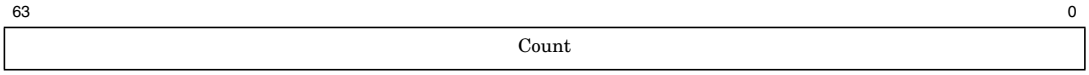
Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:8>	—	RAZ	0x0	Reserved; must be zero.
<7:0>	parity	RO	0x0	Dcache parity bits read during the last execution of a CACHE idxldtag instruction.

Cavium Confidential For Bowen Yang ALU 02/27/2014

2.15.4 Cavium-Specific Coprocessor 0 Registers in Guest Mode

Guest.CvmCount Register: Implemented at reg# = 9, sel = 6

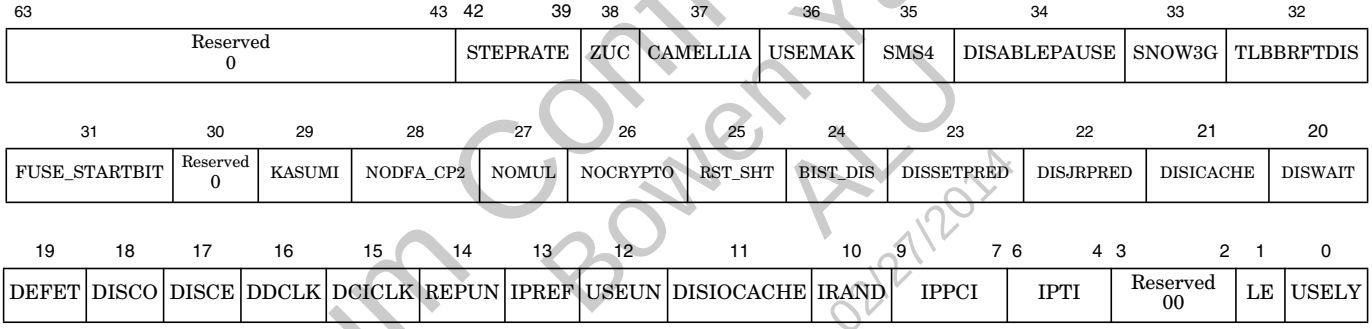
When Root.GuestCtl[CP0] = 0, accesses to this register generate a GPSI exception. Reset to 0x0. Increments by one every cycle. See "Root.Multicore Debug Register: Implemented at reg# =22, select =0" on page 169 for more details about the behavior of CvmCount during debug.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<63:0>	CvmCount	RO	RO	0x0	Interval counter. Guest.CvmCount is Root.CvmCount + Root.CvmCountOffset. Guest write access always generates a GPSI exception. • When Root.GuestCtl0[CVMGT] = 0, guest mfc0 or RDHWR 31 cause a GPSI exception. • When Root.GuestCtl0[CVMGT] = 1, read access is permitted with mfc0 or RDHWR 31. The root cannot directly write Count but can write to Root.CvmCountOffset or Root.CvmCount to change the value of this register.

Guest.CvmCtl Register: Implemented at reg# = 9, select = 7

Controls the Icache and instruction issue.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description	
<63:43>	—	RAZ	RAZ	0x0	Reserved	
<42:39>	STEPRATE	RO	RO	0x0	If non-zero, issue an instruction at most every $2^{STEPRATE}$ cycles. If non-zero, MulticoreDebug[CvGSTP] and [CvDm] must be zero. Same value as Root.CvmCtl[STEPRATE].	**cnMIPS III** only
<38>	ZUC	RO	RO	1	When set, ZUC operations are enabled. CvmCtl[ZUC] = 0 whenever CvmCtl[NOCRYPTO] = 1. Same value as Root.CvmCtl[ZUC].	**cnMIPS III** only
<37>	CAMELLIA	RO	RO	1	When set, CAMELLIA operations are enabled. CvmCtl[CAMELLIA] = 0 whenever CvmCtl[NOCRYPTO] = 1. Same value as Root.CvmCtl[CAMELLIA].	**cnMIPS III** only

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<36>	USEMAK	RO	RO	0	If set, AES operations use MAK for key. USEMAK must not be set when RNM_CTL_STATUS[DIS_MAK] is set. When USEMAK is set, the only legal AES/SMS4 instructions are: CVM_MF_AES_RESINP (both of them) CVM_MT_AES_ENC0 CVM_MT_AES_ENC1 Same value as Root.CvmCtl[USEMAK].
<35>	SMS4	RO	RO	1	When set, SMS4 operations are enabled. CvmCtl[SMS4] = 0 whenever CvmCtl[NOCRYPTO] = 1. Same value as Root.CvmCtl[SMS4].
<34>	DISABLEPAUSE	RO	RO	0	When set, the PAUSE instruction is a NOP. Same value as Root.CvmCtl[DISABLEPAUSE].
<33>	SNOW3G	RO	RO	1	When set, SNOW3G and GFM REFLECT operations are enabled. CvmCtl[SNOW3G] = 0 whenever CvmCtl[NOCRYPTO] = 1. Same value as Root.CvmCtl[SNOW3G].
<32>	TLBBRFTDIS	RO	RO	0	If clear, following a TLB/XTLB refill exception, conditional branches are forced to predict not-taken until a subsequent mispredict or ERET. Same value as Root.CvmCtl[TLBBRFTDIS].
<31>	FUSE_STARTBIT	RO	RO	1	Always reads as 1.
<30>	—	RAZ	RAZ	0	Reserved
<29>	KASUMI	RO	RO	1	When set, KASUMI operations are enabled. CvmCtl[KASUMI] = 0 whenever CvmCtl[NOCRYPTO] = 1. Same value as Root.CvmCtl[KASUMI].
<28>	NODFA_CP2	RO	RO	1	Read only and reads as 1.
<27>	NOMUL	RO	RO	0	When set, large mul operations are disabled. Same value as Root.CvmCtl[NOMUL].
<26>	NOCRYPTO	RO	RO	0	When set, crypto operations are disabled. Same value as Root.CvmCtl[NOCRYPTO].
<25>	RST_SHT	RO	RO	0	When set, reset delays are short (should be 0). Same value as Root.CvmCtl[RST_SHT].
<24>	BIST_DIS	RO	RO	0	When set, BIST is disabled (should be 0). Same value as Root.CvmCtl[BIST_DIS].
<23>	DISSETPRED	RO	RO	0	When set, disables power-saving feature of Icache. Same value as Root.CvmCtl[DISSETPRED].
<22>	DISJRPRED	RO	RO	0	When set, JALs no longer push the jump stack (i.e. the jump stack remains frozen). Same value as Root.CvmCtl[DISJRPRED].
<21>	DISICACHE	RO	RO	0	When set, Icache is disabled (operates in pass-through mode). Normally, this bit should not be set. When DISICACHE = 0, and is being written to 1, one of the following must be true: • The linear instruction stream including the MTC0/DMTC0 instruction that writes CvmCtl[DISICACHE] = 1 is running out of I/O space or DMSEG • The MTC0/DMTC0 instruction that writes CvmCtl[DISICACHE] = 1 is followed immediately with a SYNCI instruction (in the same cache block). • There was a one cache line backward branch before the MTC0/DMTC0 instruction that writes CvmCtl[DISICACHE] = 1 Same value as Root.CvmCtl[DISICACHE].

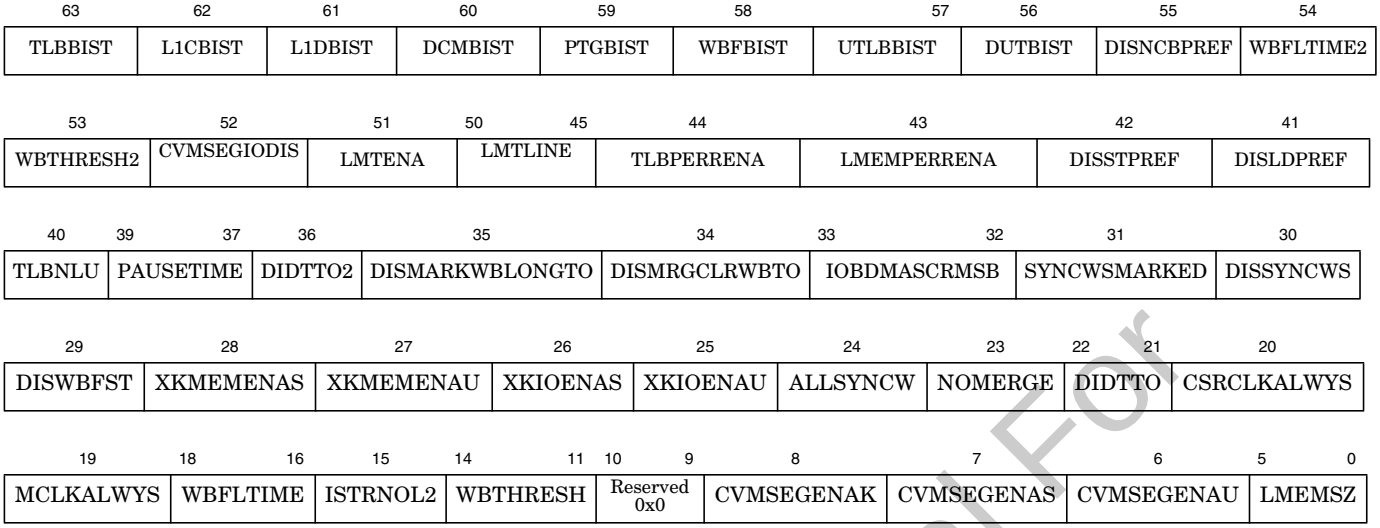
Cavium Confidential For Robert Yang ALU 2/25/2014

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<20>	DISWAIT	RO	RO	0	When set, WAIT is disabled. Same value as Root.CvmCtl[DISWAIT].
<19>	DEFET	RO	RO	0	When set, fetch under fill is disabled. Same value as Root.CvmCtl[DEFET].
<18>	DISCO	RO	RO	0	When set, disable the conditional clocks (i.e. always clock) mul/crypto. Same value as Root.CvmCtl[DISCO].
<17>	DISCE	RO	RO	0	When set, disable the conditional clocks (i.e. always clock) issue. Same value as Root.CvmCtl[DISCE].
<16>	DDCLK	RO	RO	0	When set, disable the conditional clocks (i.e. always clock) exec. Same value as Root.CvmCtl[DDCLK].
<15>	DCICLK	RO	RO	0	When set, disable the conditional clocks (i.e. always clock) issue unit CSRs. Same value as Root.CvmCtl[DCICLK].
<14>	REPUN	R/W	R/W	0	When set, LBX+LBUX+LH+LHX+LHU+LHUX+LW+LWX+LWU+LWUX+LD+LDX+SH+SW+SD unaligned accesses handled by hardware.
<13>	IPREF	RO	RO	0	When set, instruction prefetching is disabled. Same value as Root.CvmCtl[IPREF].
<12>	USEUN	R/W	R/W	0	When set, CVM ULW+ULD+USW+USD insts, else MIPS LWL+LWR+LDL+ LDR+SWL+SWR+SDL+SDR
<11>	DISIOCACHE	RO	RO	0	When set, the iocache feature is disabled, which cause noncompliance for EJTAG and may cause extra I/O-space fetches. Same value as Root.CvmCtl[DISIOCACHE].
<10>	IRAND	RO	RO	0	When set, the Icache uses random replacement, else use “bitmask LRU”. Same value as Root.CvmCtl[IRAND].
<9:7>	IPPCI	R/W	R/W	0x7	Selects the Cause[IP] bit to which the Performance Counter Interrupt request is merged. Legal values are CvmCtl[IPPCI] = 2 .. 7, where CvmCtl[IPPCI]=*n* selects Cause[IP<*n*>] for the interrupt. IntCtl[IPPCI] is a read-only copy of CvmCtl[IPPCI].
<6:4>	IPTI	R/W	R/W	0x7	Selects the Cause[IP] bit to which the Timer Interrupt request is merged. Legal values are CvmCtl[IPTI] = 2 .. 7, where CvmCtl[IPTI]=*n* selects Cause[IP<*n*>] for the interrupt. IntCtl[IPTI] is a read-only copy of CvmCtl[IPTI].
<3:2>	—	RAZ	RAZ	0x0	Reserved
<1>	LE	RO	RO	0	When set, the core defaults to LE (i.e. Config[BE] is a read-only copy of the inverse of CvmCtl[LE].) Same value as Root.CvmCtl[LE].
<0>	USELY	R/W	R/W	0	When clear use *L for unaligned opcode for BE, *R for LE. When set unaligned opcode is always the *L opcodes

Cavium Confidential For Power ALU 8/21/2014

Guest.CvmMemCtl Register: Cvm Memory Control Register, Implemented at reg# = 11, select = 7

Controls the cnMIPS™ cores' memory system. Also, see Table 2–34 for error handling bits.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<63>	TLBBIST	RO	RO	0	1 = BIST fail, 0 = BIST pass. Same value as Root.CvmMemCtl[TLBBIST].
<62>	L1CBIST	RO	RO	0	1 = BIST fail, 0 = BIST pass. Same value as Root.CvmMemCtl[L1CBIST].
<61>	L1DBIST	RO	RO	0	1 = BIST fail, 0 = BIST pass. Same value as Root.CvmMemCtl[L1DBIST].
<60>	DCMBIST	RO	RO	0	1 = BIST fail, 0 = BIST pass. Same value as Root.CvmMemCtl[DCMBIST].
<59>	PTGBIST	RO	RO	0	1 = BIST fail, 0 = BIST pass. Same value as Root.CvmMemCtl[PTGBIST].
<58>	WBFBIST	RO	RO	0	1 = BIST fail, 0 = BIST pass. Same value as Root.CvmMemCtl[WBFBIST].
<57>	UTLBBIST	RO	RO	0	1 = BIST fail, 0 = BIST pass. Same value as Root.CvmMemCtl[UTLBBIST].
<56>	DUTBIST	RO	RO	0	1 = BIST fail, 0 = BIST pass. Same value as Root.CvmMemCtl[DUTBIST].
<55>	DISNCBPREF	RO	RO	0	1 = NCB prefetches (from SSO) are disabled and silently dropped. 0 = NCB prefetches (from SSO) are allowed. Same value as Root.CvmMemCtl[DISNCBPREF].
<54>	WBFLTIME2	RO	RO	0	Used in conjunction with WBFLTIME field (see below). Same value as Root.CvmMemCtl[WBFLTIME2].
<53>	WBTHRESH2	RO	RO	1	Used in conjunction with WBTHRESH field (see below). Same value as Root.CvmMemCtl[WBTHRESH2].
<52>	CVMSEGIODIS	R/W	R/W	0	When set, guest CVMSEG I/O accesses cause guest-mode address-error exceptions when in guest-user or guest-supervisor modes. **NOTE:** Root.CvmMemCtl[CVMSEGIODIS] can cause a root exception on a guest mode CVMSEG I/O access.
<51>	LMTENA	RO	RAZ	0	Reserved.

Cavium Confidential For Bowen Yang 5/22/2

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<50:45>	LMTLINE	RO	RO	—	This is the cacheline in CVMSEG LM used in wide atomic store operations (see Section 2.23.2). LMTLINE must not change value in a wide atomic memory write sequence. Same value as Root.CvmMemCtl[LMTLINE].
<44>	TLBPERRDIS	RO	RO	0	When set, TLB parity error checking is disabled. Same value as Root.CvmMemCtl[TLBPERRDIS].
<43>	LMEMPERRENA	RO	RO	0	When set, CVMSEG LM parity errors are enabled. To enable CVMSEG LM parity errors, software must follow these constraints: • Whenever CVMSEG LM is enabled with a non-zero CvmMemCtl[LMEMSZ], or if CvmMemCtl[LMEMSZ] is increased when CVMSEG LM is enabled, write all the newly-allocated CVMSEG LM bytes to zero before loading them. This constraint is not needed if it is always true, for all newly allocated 64-bit words, that no bytes in the word are loaded from before all bytes in the word are stored to. ("Stored to" could be via an IOBDMA.) • Deadlock must be avoided. The hardware cannot recover from CVMSEG LM parity errors, and the parity error remains following the associated Cache Error on Data Access exception. The Cache Error on Data Access exception handler should check if the way/set indicate the block is in CVMSEG LM. If yes, software recovery may be required if the error is not to be considered fatal. Same value as Root.CvmMemCtl[LMEMPERRENA].
<42>	DISSTPREF	RO	RO	0	If set, NUDGE/WRITEBACK_INVALIDATE, NUDGE_WB, EVICT_SOON, L2, DONT_WRITE_BACK, PREPARE_FOR_STORE, and PREPARE_FOR_STORE_THROUGH prefetch operations become NOPs. Same value as Root.CvmMemCtl[DISSTPREF].
<41>	DISLDPREF	RO	RO	0	If set, NORMAL and NOTL2 prefetch operations become NOPs. Same value as Root.CvmMemCtl[DISLDPREF].
<40>	TLBNLU	RO	RO	0	TLB replacement policy: 0 = bitmask LRU; 1 = NLU. This field selects between the TLB replacement policies: bitmask LRU or NLU. Bitmask LRU maintains a mask of recently used TLB entries and avoids them as new entries are allocated. NLU simply guarantees that the next allocation is not the last used TLB entry. Opposite value as Root.CvmMemCtl[TLBNLU]. If the root TLB is bitmask LRU, then the guest TLB is NLU. If the root TLB is NLU, then the guest TLB is bitmask LRU.
<39:37>	PAUSETIME	RO	RO	0x0	Selects the bit in the counter used for releasing a PAUSE. This counter trips every $2^{(8+PAUSETIME)}$ cycles. If not already released, the cnMIPS III core will always release a given PAUSE instruction within $2^{(8+PAUSETIME)}$. If the counter trip happens to line up, the cnMIPS III core may release the PAUSE instantly. Same value as Root.CvmMemCtl[PAUSETIME].
<36>	DIDTTO2	RO	RO	0x0	This field is an extension of CvmMemCtl[DIDTTO]. See the description for "DIDTTO" on page 161. Same value as Root.CvmMemCtl[DIDTTO2].
<35>	DISMARKWB LONGTO	RO	RO	0	If set, marked write-buffer entries time out the same as other entries; if clear, marked write-buffer entries use the maximum timeout. Same value as Root.CvmMemCtl[DISMARKWBLONGTO].

Cavium Confidential For Bowen Ning 9/27/2014

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<34>	DISMRGCLRWBTO	RO	RO	0	If set, a merged store does not clear the write-buffer entry timeout state. Same value as Root.CvmMemCtl[DISMRGCLRWBTO].
<33:32>	IOBDMASCRMSB	RO	RO	0x0	Two bits that are the MSBs of the resultant CVMSEG LM word location for an IOBDMA. The other 8 bits come from the SCRADDR field of the IOBDMA. Same value as Root.CvmMemCtl[IOBDMASCRMSB].
<31>	SYNCWSMARKED	RO	RO	0	If set, SYNCWS and SYNCS only order marked stores; if clear, SYNCWS and SYNCS only order unmarked stores. SYNCWSMARKED has no effect when DISSYNCWS is set. Same value as Root.CvmMemCtl[SYNCWSMARKED].
<30>	DISSYNCWS	RO	RO	0	If set, SYNCWS acts as SYNCW and SYNCS acts as SYNC. Same value as Root.CvmMemCtl[].
<29>	DISWBFST	RO	RO	0	If set, no stall happens on write buffer full. Same value as Root.CvmMemCtl[DISSYNCWS]. **NOTE:** This bit should always be cleared to 0.
<28>	XKMEMENAS [1]	R/W	R/W	0	If set (and SX set), supervisor-level loads/stores can use XKPHYS addresses with VA<48>==0
<27>	XKMEMENAU [1]	R/W	R/W	0	If set (and UX set), user-level loads/stores can use XKPHYS addresses with VA<48>==0
<26>	XKIOENAS [1]	R/W	R/W	0	If set (and SX set), supervisor-level loads/stores can use XKPHYS addresses with VA<48>==1
<25>	XKIOENAU [1]	R/W	R/W	0	If set (and UX set), user-level loads/stores can use XKPHYS addresses with VA<48>==1
<24>	ALLSYNCW	RO	RO	0	If set, all stores act as SYNCW (NOMERGE must be set when this is set) RW, reset to 0 if set, all stores act as SYNCW (NOMERGE must be set when this is set). Same value as Root.CvmMemCtl[ALLSYNCW].
<23>	NOMERGE	RO	RO	0	If set, no stores merge, and all stores reach the coherent bus in order. Same value as Root.CvmMemCtl[NOMERGE].
<22:21>	DIDTTO [2]	RO	RO	0x0	{CvmMemCtl[DIDTTO2], CvmMemCtl[DIDTTO]} selects the bit in the counter for DID timeouts. Taken together, these fields are encoded as follows: 0 = 2^{31}, 1 = 2^{30}, 2 = 2^{29}, 3 = 2^{14}, 4 = 2^{28}, 5 = 2^{27}, 6 = 2^{26}, 7 = 2^{25}. Actual timeout is between 1× and 2× this interval. For example, with {CvmMemCtl[DIDTTO2], CvmMemCtl[DIDTTO]} = 3, expiration interval is between 16K and 32K core clock cycles. Same value as Root.CvmMemCtl[DIDTTO].
<20>	CSRCKALWYS	RO	RO	0	If set, the (mem) CSR clock never turns off. Same value as Root.CvmMemCtl[CSRCKALWYS].
<19>	MCLKALWYS	RO	RO	0	If set, the core's memory clock never turns off. Same value as Root.CvmMemCtl[MCLKALWYS].

Cavium Confidential Pdf
Bowen Yang @ALD 2014

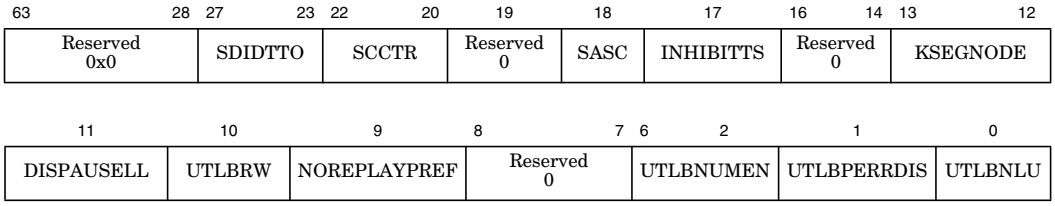
Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<18:16>	WBFLTIME	RO	RO	0x4	{WBFLTIME2,WBFLTIME} (aka WBFLTIME) selects the write buffer flush timeout WBFLTIME is a 5-bit signed value that selects the bit in a counter used for write buffer flush timeouts. (WBFLTIME+11) is the bit position in an internal counter used to determine expiration. A write buffer entry expires between 1x and 2x this interval. For example, with WBFLTIME = 0, a write buffer entry will expire between 2K and 4K cycles after it is allocated. With WBFLTIME = -1 (i.e. WBFLTIME2=1, WBFLTIME=0xF), a write buffer entry will expire between 1K and 2K cycles after it is allocated. Legal values range from a min of -5 (WBFLTIME2=0x1,WBFLTIME=0x3) up to max of +7 (WBFLTIME2=0x0,WBFLTIME=0x7). Same value as Root.CvmMemCtl[WBFLTIME].
<15>	ISTRNOL2	RO	RO	0	If set, do not put Istream in the L2 cache. This bit determines whether the core services L1 instruction cache misses with LDI or LDT CMI transactions. Same value as Root.CvmMemCtl[ISTRNOL2]. See Section 3.4
<14:11>	WBTHRESH	RO	RO	0x8	The write buffer threshold is a concatenation of the two fields: <WBTHRESH2, WHTHRESH> The legal range of the write buffer threshold is 2 - 22. Values 0, 1, and 23-31 are illegal. The reset value of the threshold is 12 (= 0xC, WBTHRESH2 = 0x0, WBTHRESH = 0xC). Same value as Root.CvmMemCtl[WBTHRESH].
<10:9>	—	RAZ	RAZ	0x0	Reserved.
<8>	CVMSEGENAK[1]	R/W	R/W	0	If set, CVMSEG is available for load/store operations in kernel/debug mode. CVMSEGENAK must be enabled when either CVMSEGENAS or CVMSEGENAU are set. Whenever CVMSEGENAK transitions 0→1, software must also invalidate the Dcache if it is possible that any CVMSEG virtual addresses are currently present in the data cache. CVMSEG virtual addresses are KSEG3 virtual addresses in the range 0xFFFF FFFF FFFF 8000 to 0xFFFF FFFF FFFF BFFF. Either an L1 Dcache Virtual Tag Invalidate or an L1 Dcache Invalidate can be used to remove the CVMSEG addresses from the data cache (refer to the CACHE instruction definition in Section A.4).
<7>	CVMSEGENAS[1]	R/W	R/W	0	If set, CVMSEG is available for load/store operations in supervisor mode.
<6>	CVMSEGENAU[1]	R/W	R/W	0	If set, CVMSEG is available for load/store operations in user mode.
<5:0>	LMEMSZ	R/W	R/W	0x0	Size of local memory in cache blocks; 54 (6912 bytes) is max legal value.

1. Note that both CvmMemCtl[CVMSEGENA*] and CvmMemCtl[XK*] enable the corresponding range (CVMSEG and XKPHYS with VA<48>==1) only for loads/store operations, and NOT for instruction references.
2. Note that DID time-outs are the only source of bus-error exceptions. These bus-error exceptions may not be recoverable, but the handler can extract information for diagnostic purposes.

Cavium Confidential For Beijing Huawei 2012/7/2

Guest.CvmMemCtl2 Register: Implemented at reg# = 16, select = 6

Also, see Table 2–34 for error handling bits.



Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<63:24>	—	RAZ	RAZ	0x0	Reserved.
<27:23>	SDIDTTO	RO	RO	0x10	Not used by the core hardware. Valid range is 0xC–0x1F. Same value as Root.CvmMemCtl2[SDIDTTO].
<22:20>	SCCTR	RO	RO	0	Store-conditional cache miss counter. Counts the number of consecutive store conditional operations that locally failed solely because the cache block was not present in the Dcache. Always 0x0 following SC/SCD execution when SASC = 1. When SASC=0: • SCCTR can range 0–7 after SC/SCD execution. The core increments the counter whenever it locally fails a SC/SCD operation solely because the cache block was not present in the Dcache, and clears it otherwise. • When SCCTR = 7, the core sends SC/SCD operations that are not locally known to succeed or fail to L2C for final success/fail resolution, as if SASC = 1. • When SCCTR ≠ 7, the core sends only SC/SCD operations that are not locally known to succeed or fail to L2C if the cache block is present in the Dcache when the SC/SCD executes. For the latest SCCTR value, a *MFC0 CvmMemCtl2 must issue a minimum of three cycles after the last SC/SCD operation. Same value as Root.CvmMemCtl2[SCCTR].
<19>	—	RAZ	RAZ	0	Reserved.
<18>	SASC	RO	RO	0	Send all store conditional operations. When set, the core sends all store-conditional operations that are not locally known to succeed or fail (based only on CPU-local information that does not include whether the block was written by another agent, which is indicated by whether the cache block was invalidated from the Dcache) to L2C for final success/failure determination. When clear, the core may (or may not) locally fail a store conditional operation solely because the cache block was not present in the data cache when the store-conditional operation executes. See SCCTR. Same value as Root.CvmMemCtl2[SASC].

Cavium Confidential For Bowen Feng ALU 9/17/2014

Bit Pos	Field Name	Field TypeG	Field TypeR	Reset Value	Field Description
<17>	INHIBITTS	RO	RO	0	Inhibit multi-TLB-entry-match machine checks. When set, a multi-match machine check cannot occur as the result of TLBW* execution and Status[TS] will never be set by hardware. In general, the TLB hardware might malfunction in unpredictable ways when multiple matches are present in a TLB. Consequently, this bit should be set only in very controlled circumstances. Cavium recommends that this bit is ever set only during a full context switch of all guest TLB entries by the root software (i.e. hypervisor), and that this root software is certain that root TLB misses cannot occur while this switch code is executing, and that this root software is also certain that the guest TLB cannot have more than one TLB entry that matches any given GVA after the switch is complete. Same value as Root.CvmMemCtl2[INHIBITTS].
<16:14>	—	RAZ	RAZ	0x0	Reserved.
<13:12>	KSEGNODE	R/W	R/W	0x0	Reserved.
<11>	DISPAUSELL	RO	RO	0x0	When set to 1, PAUSE will not exit on invalidation signal after a load-locked instruction. Same value as Root.CvmMemCtl2[DISPAUSELL].
<10>	UTLBRW	RO	RO	0x0	When set to 1, Index, EntryLo0, EntryLo1, and EntryHi COP0 registers map to UtlbIndex, UtlbEntry1, UtlbEntry2, and UtlbEntry0 registers, respectively. Typical use of this field is to debug the µTLB. Same value as Root.CvmMemCtl2[UTLBRW].
<9>	NOREPLAYPREF	RO	RO	0x0	When set to 1, PREF and PREFX instructions become a NOP when they miss in the µTLB. When set to zero, PREF and PREFX instructions replay on µTLB miss. The PREF & PREFX hint = 24 is not affected by this field. Same value as Root.CvmMemCtl2[NOREPLAYPREF].
<8:7>	Reserved	RAZ	RAZ	0x0	Reserved
<6:2>	UTLBNUMEN	RO	RO	0x1F	Number of µTLB Entries minus 1. 0x1 in this field means 2 entries in the µTLB, Reset value is 0x1F or 31 (so µTLB resets to 32 entries). Illegal value is 0x0 (minium 2 entries needed in the µTLB). Same value as Root.CvmMemCtl2[UTLBNUMEN].
<1>	UTLBPERRDIS	RO	RO	0x0	When set, µTLB parity errors are disabled. Same value as Root.CvmMemCtl2[UTLBPERRDIS].
<0>	UTLBNLU	RO	RO	0x0	µTLB replacement policy: 0 = bitmask LRU; 1 = NLU. This field selects between the µTLB replacement policies: bitmask LRU or NLU. Bitmask LRU maintains a mask of recently used µTLB entries and avoids them as new entries are allocated. NLU simply guarantees that the next allocation is not the last used µTLB entry. Same value as Root.CvmMemCtl2[UTLBNLU].

Cavium Confidential For Beken Use Only

2.16 Coprocessor 1 (Floating Point) Control Registers

Floating Point Implementation Register: FIR, CP1 Control Register 0

This register contains information identifying the capabilities of the floating point unit, the floating point processor identification, and the revision level of the floating

31 29	28	27 24	23	22	21	20	19	18	17	16	15 8	7 0
Reserved 0x0	UFRP 0	Impl 0x0	Has2008 0	F64 1	L 1	W 1	3D 0	PS 0	D 1	S 1	ProcessorID	Revision

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:29>	—	—	0	Reserved.
<28>	UFRP	RAZ	0	User mode FR access. Not implemented.
<27:24>	Impl	R	0	Not used.
<23>	Has2008	RAZ	0	IEEE-754-2008 features. Not implemented.
<22>	F64	R	1	FPU is 64 bits.
<21>	L	R	1	L fixed point (implemented).
<20>	W	R	1	W fixed point (implemented).
<19>	3D	R	0	MIPS-3D ASE (not implemented).
<18>	PS	R	0	Paired single floating-point data type (not implemented).
<17>	D	R	1	Double-precision floating-point data type and instructions (implemented).
<16>	S	R	1	Single-precision floating-point data type and instructions (implemented).
<15:8>	ProcessorID	R	0x96	0x95 = CN78XX, 0x96 = CN71XX/CN70XX
<7:0>	Revision	RAZ	0x0	

Cavium Confidential For Bowen Yang ALU 02/27/2014

Floating Point Control and Status Register: FCSR, CP1 Control Register 31

This register controls the operation of the floating-point unit

31 25	24	23	22 21	20	19	18	17 12	11 7	6 2	1 0
FCC	FS	FCC	Impl	Rsvd	ABS2008	NAN2008	Cause	Enables	Flags	RM
			00	0	0	0				

7	6	5	4	3	2	1		0																	
									E	V	Z	O	U	I	V	Z	O	U	I	V	Z	O	U	I

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:25, 23>	FCC	R/W	—	Floating-point condition codes. These bits record the result of floating-point compares and are tested for floating-point conditional branches and conditional moves. The FCC bit to use is specified in the compare, branch, or conditional move instruction.
<24>	FS	R/W	—	Flush to zero. When FS = 1, denormalized operands and results are flushed to zero instead of causing an unimplemented operation exception. The Inexact Cause bit is set to 1 when the operand or result is flushed to zero.
<22:21>	Impl	R	0x0	Reserved.
<20>	—	R	0x0	Reserved.
<19>	ABS2008	RAZ	0x0	Read as 0. IEEE-754-2008 is not implemented.
<18>	NAN2008	RAZ	0x0	Read as 0. IEEE-754-2008 is not implemented.
<17:12>	Cause	R/W	—	Cause bits. These bits indicate the exception conditions that arise during execution of an FPU arithmetic instruction. A bit is set to 1 if the corresponding exception condition arises during the execution of an instruction and is set to 0 otherwise. By reading the registers, the exception condition caused by the preceding FPU arithmetic instruction can be determined.
<11:7>	Enables	R/W	—	Enable bits. These bits control whether or not a exception is taken when an IEEE exception condition occurs for any of the five conditions. The exception occurs when both an Enable bit and the corresponding Cause bit are set either during an FPU arithmetic operation or by moving a value to FCSR or one of its alternative representations. Note that Cause bit E has no corresponding Enable bit; the non-IEEE Unimplemented Operation exception is defined by MIPS as always enabled.
<6:2>	Flags	R/W	—	Flag bits. This field shows any exception conditions that have occurred for completed instructions since the flag was last reset by software. When a FPU arithmetic operation raises an IEEE exception condition that does not result in a Floating Point Exception (i.e. the Enable bit was off), the corresponding bit(s) in the Flag field are set, while the others remain unchanged. Arithmetic operations that result in a Floating Point Exception (i.e., the Enable bit was on) do not update the Flag bits. This field is never reset by hardware and must be explicitly reset by software.
<1:0>	RM	R/W	—	Rounding Mode. 0 = round to nearest, 1 = round to zero, 2 = round to positive infinity, 3 = round to negative infinity.

Cavium Confidential For Broadcom Use Only 02-Feb-2014

2.17 EJTAG DRSEG Registers

The memory-mapped EJTAG registers are located in the debug register segment (drseg), which is a subsegment of the debug segment (dseg). They are accessible by the debug software when the processor is executing in Debug Mode. These registers provide both miscellaneous debug control and control of hardware breakpoints.

Table 2–41 EJTAG DRSEG Registers Summary

Register	Implemented/Not Implemented	Comments
Debug Control Register (DCR)	Implemented	See page 189
Instruction Breakpoint Status (IBS)	Implemented	See page 190
Instruction Breakpoint Address (IBA)	Four implemented	See page 190
Instruction Breakpoint Address Mask (IBM)	Four implemented	See page 190
Instruction Breakpoint ASID (IBASID)	Four implemented	See page 191
Instruction Breakpoint Control (IBC)	Four implemented	See page 191
Data Breakpoint Status (DBS)	Implemented	See page 192
Data Breakpoint Address (DBA)	Four implemented	See page 192
Data Breakpoint Address Mask (DBM)	Four implemented	See page 192
Data Breakpoint ASID (DBASID)	Four implemented	See page 193
Data Breakpoint Control (DBC)	Four implemented	See page 193
Data Breakpoint Value (DBV)	Four implemented	See page 193

Cavium Confidential For Bowen Yang ALU 02/27/2014

Debug Control Register (DCR)

Address: Offset 0x0000

63 32	31 30	29	28 27	26	25 24	23	22	21 19	18
Reserved 0	EJTAG_Brk_Override 0 / Rsvd 0	ENM	PCnoGID 0 / PCnoTCID 0	PCIM 0	PCnoASID 0 / DASQ 0	DASe 0	DAS 0	Reserved 0	FDCImpl 0

17	16	15	14	13 12	11	10	9	8 6	5	4	3	2	1	0
DataBrk 1	InstBrk 1	IVM 1	DVM 0	Reserved 0	RDVec 0	CBT 0	PCS 1	PCR	PCSe	IntE	NMIE	NMIpend	SRstE 0	ProbEn

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:32>	—	RAZ	0x0	Reserved.
<31>	EJTAG_Brk_Override	RAZ	0	Not implemented.
<30>	—	RAZ	0	Reserved.
<29>	ENM	RO	0	$\overline{\text{CvmCtl[LE]}}$
<28>	PCnoGID	RAZ	0	Not implemented.
<27>	PCnoTCID	RAZ	0	Not implemented. MIPS MT ASE not implemented.
<26>	PCIM	RAZ	0	Not implemented.
<25>	PCnoASID	RAZ	0	Not implemented.
<24>	DASQ	RAZ	0	Not implemented. Data address sampling not implemented.
<23>	DASe	RAZ	0	Not implemented. Data address sampling not implemented.
<22>	DAS	RAZ	0	Not implemented. Data address sampling not implemented.
<21:19>	—	RAZ	0x0	Reserved.
<18>	FDCImpl	RAZ	0	Fast debug channel not implemented.
<17>	DataBrk	RO	1	4 data hardware breakpoints implemented.
<16>	InstBrk	RO	1	4 instruction hardware breakpoints implemented.
<15>	IVM	RO	1	Inverted data value match on data hardware breakpoints implemented.
<14>	DVM	RAZ	0	Data value store on a data value breakpoint match not implemented. (Not needed since load data value breakpoints are imprecise.)
<13:12>	—	RAZ	0x0	Reserved.
<11>	RDVec	RAZ	0	Not implemented.
<10>	CBT	RAZ	0	Complex breakpoint block not implemented.
<9>	PCS	RO	1	PC sampling is supported.
<8:6>	PCR	R/W	0	PC Sampling rate. Values 0 to 7 map to values 2^5 to 2^{12} cycles, respectively (i.e. a PC sample is written out every 32, 64, 128, 256, 512, 1024, 2048, or 4096 cycles respectively). The external probe or software is allowed to set this value to the desired sample rate.
<5>	PCSe	R/W	0	Implemented (PC Sampling enable).
<4>	IntE	R/W	1	Hardware and software interrupt enable for Non-Debug Mode.
<3>	NMIE	R/W	1	Non-Maskable Interrupt (NMI) enable for Non-Debug Mode.
<2>	NMIpend	RO	0	Indication for pending NMI.
<1>	SRstE	RAZ	0	Soft-reset masking not implemented.
<0>	ProbEn	RO	1/0	Indicates ECR[ProbEn].

Cavium Confidential For Bowen Yang ALU 9/7/2014

Instruction Breakpoint Status (IBS) Register

Address: Offset 0x1000

63 31	30	29 28	27 24	23 16	15	14 4	3 0
Reserved 0	ASIDsup 1	Reserved 0	BCN 0100	Reserved 0	IBPshare 0	BS14...BS4 0	BS3...BS 0

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:31>	—	RAZ	0x0	Reserved
<30>	ASIDsup	RO	1	ASID compare implemented
<29:28>	—	RAZ	0x0	Reserved
<27:24>	BCN	RO	0x4	4 instruction breakpoints implemented
<23:16>	—	RAZ	0x0	Reserved
<15>	IBPshare	RAZ	0	Not implemented. MIPS MT ASE not implemented.
<14:4>	BS [14:4]	RAZ	0x0	Not implemented
<3:0>	BS [3:0]	R/W	—	Implemented

Instruction Breakpoint Address (IBA0...3) Register

Address: Offset 0x1100 + 0x100 × n

63 0
IBA

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:0>	IBA	R/W	—	Instruction Breakpoint Address (IBA0...3) register has the virtual address used in the condition for instruction breakpoint 0...3. It is located at drseg segment offset 0x1100 + 0x100 × n.

Instruction Breakpoint Address Mask (IBM0...3) Register

Address: Offset 0x1108 + 0x100 × n

63 0
IBM

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:0>	IBM	R/W	—	Instruction Breakpoint Address Mask (IBM0...3) register has the address compare mask used in the condition for instruction breakpoint 0...3. The address that is masked is in the IBA0...3 register. The IBM0...3 register is located at drseg segment offset 0x1108 + 0x100 × 0...3.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Instruction Breakpoint ASID (IBASID0...3) Register

Address: Offset 0x1110 + 0x100 × n

63 … 32	31 … 24	23	22 … 21	20	19 … 12	11 … 8	7 … 0
Reserved 0x0	GuestID 0	UGID	EASID 0	MGPA 0	Reserved 0x0	VPE 0	ASID

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:32>	—	RAZ	0x0	Reserved
<31:24>	GuestID	RO	0x0	GuestID<7:1> always 0x0. GuestID<0>: 1 = match guest only, 0 = match root only
<23>	UGID	RO	1	GuestID field is always used for the match comparisons.
<22:21>	EASID	RO	0x0	Extended ASID not implemented.
<20>	MGPA	RO	0	Match on Guest Physical address not implemented.
<19:12>	—	RAZ	0x0	Reserved
<11:8>	VPE	RAZ	0x0	Not Implemented. MIPS MT ASE not implemented.
<7:0>	ASID	R/W	—	The Instruction Breakpoint ASID (IBASID0...3) register has the ASID value used in the compare for instruction breakpoint 0...3. It is located at drseg segment offset 0x1110 + 0x100 × 0...3.

Instruction Breakpoint Control (IBC0...3) Register

Address: Offset 0x1118 + 0x100 × n

63 … 32	31 … 24	23	22	21 … 4	3	2	1	0
Reserved 0	TC 0	ASIDuse	TCuse 0	Reserved 0	VPEuse 0	TE	Reserved 0	BE

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:32>	—	RAZ	0x0	Reserved
<31:24>	TC	RAZ	0x0	Not implemented. MIPS MT ASE not implemented.
<23>	ASIDuse	R/W	—	Implemented
<22>	TCuse	RAZ	0	Not implemented. MIPS MT ASE not implemented.
<21:7>	—	RAZ	0x0	Reserved
<6>	HWArts	RAZ	0	Not implemented. Address range breakpoints are not implemented.
<5>	EXCL	RAZ	0	Not implemented. Address range breakpoints are not implemented.
<4>	HWART	RAZ	0	Not implemented. Address range breakpoints are not implemented.
<3>	VPEuse	RAZ	0	Not implemented. MIPS MT ASE not implemented.
<2>	TE	R/W	0	Use instruction breakpoint 0...3 as triggerpoint. See Table 2–60.
<1>	—	RAZ	0	Reserved
<0>	BE	R/W	0	Use instruction breakpoint 0...3 as breakpoint. See Table 2–60.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Data Breakpoint Status (DBS) Register

Address: Offset offset 0x2000.

63 … 31	30	29	28	27…24	23 … 16	15	14 … 4	3 … 0
Reserved 0	ASIDsup 1	NoSVmatch 0	NoLVmatch 0	BCN 0100	Reserved 0	DBPshare 0	BS14...BS4 0	BS3...BS0

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:31>	—	RAZ	0x0	Reserved
<30>	ASIDsup	RO	1	ASID compare implemented
<29>	NoSVmatch	RO	0	Store values compare supported
<28>	NoLVmatch	RO	0	Load values compare supported
<27:24>	BCN	RO	0x4	4 data breakpoints implemented
<23:16>	—	RAZ	0x0	Reserved
<15>	DBPshare	RAZ	0	Not implemented. MIPS MT ASE not implemented.
<14:4>	BS14...BS4	RAZ	0x0	Not implemented
<3:0>	BS3...BS0	R/W0	—	Implemented

Data Breakpoint Address (DBA0...3) Register

Address: Offset 0x2100 + 0x100 × n

63 … 0
DBA

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:0>	DBA	R/W	—	Data Breakpoint Address (DBA0...3) register has the virtual address used in the condition for data breakpoint 0...3. This register is located at drseg segment offset 0x2100 + 0x100 × 0...3.

Data Breakpoint Address Mask (DBM0...3) Register

Address: Offset 0x2108 + 0x100 × n

63 … 0
DBM

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:0>	DBM	R/W	—	Data Breakpoint Address Mask (DBM0...3) register has the address compare mask used in the condition for data breakpoint 0...3. The address that is masked is in the DBA0...3 register. The DBM0...3 register is located at drseg segment offset 0x2108 + 0x100 × 0...3.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Data Breakpoint ASID (DBASID0...3) Register

Address: Offset 0x2110 + 0x100 × n

63 … 32	31 … 24	23	22 … 21	20	19 … 16	15 … 8	7 … 0
Reserved 0x0	GuestID 0	UGID	EASID 0	MGPA 0	VPE 0	TCval 0	ASID

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:32>	—	RAZ	0x0	Reserved
<31:24>	GuestID	RO	0x0	GuestID<7:1> always 0x0. GuestID<0>: 1 = match guest only, 0 = match root only
<23>	UGID	RO	1	GuestID field is always used for the match comparisons.
<22:21>	EASID	RO	0x0	Extended ASID not implemented.
<20>	MGPA	RO	0	Match on Guest Physical address not implemented.
<19:16>	VPE	RAZ	0x0	Not implemented. MIPS MT ASE not implemented.
<15:8>	TCval	RAZ	0x0	Not implemented. MIPS MT ASE not implemented.
<7:0>	ASID	R/W	—	The Data Breakpoint ASID (DBASID0...3) register has the ASID value used in the compare for data breakpoint 0...3. It is located at drseg segment offset 0x2110 + 0x100 × 0...3.

Data Breakpoint Control (DBC0...3) Register

Address: Offset 0x2118 + 0x100 × n

63 … 32	31 … 24	23	22	21 … 14	13	12	11 … 4	3	2	1	0
Reserved 0	TC 0	ASIDuse 1	TCuse 0	BAI	NoSB	NoLB	BLM	VPEuse 0	TE	IVM	BE

Bit Pos	Field Name	Field Type	Reset Value	Field Description
<63:32>	—	RAZ	0x0	Reserved
<31:24>	TC	RAZ	0x0	Not implemented
<23>	ASIDuse	R/W	1	1 (ASID compare Implemented)
<22>	TCuse	RAZ	0	Not implemented
<21:14>	BAI[7:0]	R/W	—	Implemented (byte lines in value compare)
<13>	NoSB	R/W	—	Controls whether condition for data breakpoint is ever fulfilled on a store access.
<12>	NoLB	R/W	—	Controls whether condition for data breakpoint is ever fulfilled on a load access:
<11:4>	BLM[7:0]	R/W	—	Byte lines in value compare
<3>	VPEuse	RAZ	0	Not implemented
<2>	TE	R/W	0	Use instruction breakpoint *n* as triggerpoint. See Table 2–60.
<1>	IVM	R/W	—	Used to indicate that the data value match should be inverted.
<0>	BE	R/W	0	Use instruction breakpoint *n* as breakpoint. See Table 2–60.

Data Breakpoint Value (DBV0...3) Register

Address: Offset 0x2120 + 0x100 × n

63 … 0
DBVn

The Data Breakpoint Value (DBV0...3) register has the value used in the condition for data breakpoint 0...3.

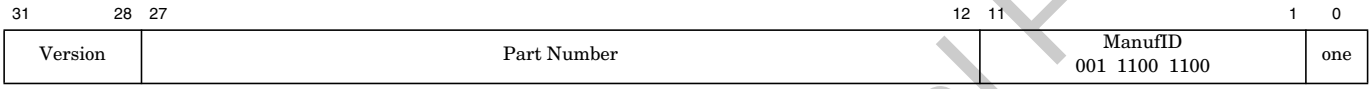
Cavium Confidential FOR Bowen Yang ALU 02/27/2014

2.18 EJTAG TAP Registers

Table 2–42 EJTAG TAP Registers Summary

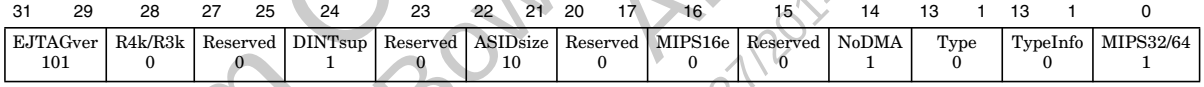
Register	Implemented/Not Implemented	Comments
Device ID	Implemented	See page 194
Implementation	Implemented	See page 194
Data	Implemented	See page 195
Address	Implemented	See page 195
EJTAG Control	Implemented	See page 195
PC Sample	Implemented	See page 196
EJTAG Boot Indication	Implemented	See page 196
Bypass	Implemented	See page 196
Fastdata	Implemented	See page 196

Device ID Register



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:28>	Version	RO	0x0	Identifies the version of a specific device as follows: CN71XX pass 1.0 = 0x0
<27:12>	Part Number	RO	0x9600	Identifies the part number of a specific device as follows: CN78XX pass 1.x = 0x9500 CN71XX/CN70XX pass 1.x = 0x9600
<11:1>	ManufID	RO	0x1CC	0x1CC
<0>	one	RO	1	Ignored on write; returns 1 on read.

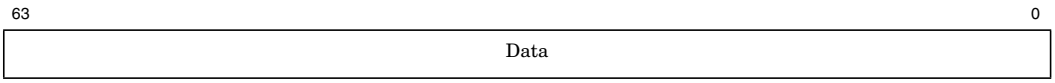
Implementation Register (TAP Instruction IMPCODE)



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31:29>	EJTAGver	RO	0x5	EJTAG Version 5.0
<28>	R4k/R3k	RO	0	R4K privileged environment
<27:25>	—	RAZ	0x0	Reserved
<24>	DINTsup	RO	1	DINT is supported from TAP.
<23>	—	RAZ	0	Reserved
<22:21>	ASIDsize	RO	0x2	8-bit ASID
<20:17>	—	RAZ	0x0	Reserved
<16>	MIPS16e	RO	0	No MIPS16e support
<15>	—	RAZ	0	Reserved
<14>	NoDMA	RO	1	Implemented
<13:11>	Type	RAZ	0x0	Not implemented
<10:1>	TypeInfo	RAZ	0x0	Not implemented
<0>	MIPS32/64	RO	1	64-bit processor

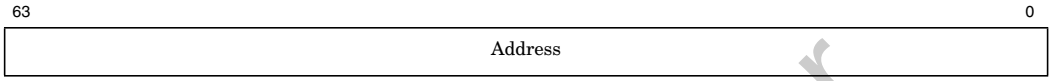
Cavium Confidential For Bowen Yang ALU 02/27/2014

Data Register (TAP Instruction DATA, ALL, or FASTDATA)



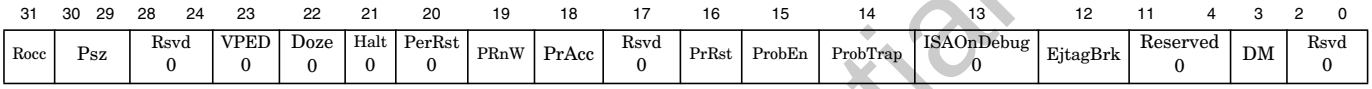
The read/write Data register is used for opcode and data transfers during processor accesses. A TAP write to the Data register is ignored when there is no DMSEG access pending.

Address Register (TAP Instruction ADDRESS or ALL)



The read-only Address register provides the address for a processor access.

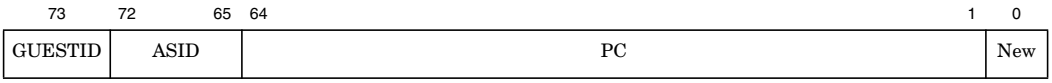
EJTAG Control Register (ECR) (TAP Instruction CONTROL or ALL)



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<31>	Rocc	R/W0	1	This bit must be cleared to acknowledge that the reset was detected.
<30:29>	Psz	RO	—	Indicates the size of a pending processor access, in combination with the Address register.
<28:24>	—	RAZ	0x0	Reserved
<23>	VPED	RAZ	0	MIPS MT ASE not implemented.
<22>	Doze	RO	0	No low power mode
<21>	Halt	RO	0	Internal bus clock does not stop
<20>	PerRst	RAZ	0	Peripheral reset not implemented
<19>	PRnW	RO	—	Indicates read or write of a pending processor access.
<18>	PrAcc	R/W0	0	Indicates a pending processor access and controls finishing of a pending processor access.
<17>	—	RAZ	0	Reserved
<16>	PrRst	R/W	0	Implemented (core reset supported)
<15>	ProbEn	R/W	1/0	Controls whether the probe handles accesses to the dmseg segment through servicing of processors accesses. Resets to one if EJTAGBOOT, otherwise resets to zero.
<14>	ProbTrap	R/W	1/0	Controls location of the debug exception vector. Resets to one if EJTAGBOOT, otherwise resets to zero.
<13>	ISAOnDebug	RO	0	Determines the ISA to be used on a debug exception when ProbTrap=1; reads as 0 (microMIPS not implemented).
<12>	EjtagBrk	R/W1	1/0	Requests a Debug Interrupt exception to the processor when this bit is written as 1. Resets to one if EJTAGBOOT, otherwise resets to zero.
<11:4>	—	RAZ	0x0	Reserved
<3>	DM	RO	0	Indicates if the processor is in Debug Mode.
<2:0>	—	RAZ	0x0	Reserved

Cavium Confidential For Bowen Yang A.U 02/27/2014

PC Sample Register (TAP Instruction PCSAMPLE)



Bit Pos	Field Name	Field Type	Reset Value	Field Description
<73>	GUESTID	RO	—	1 = the sampled PC is a GVA, 0 = the sampled PC is a RVA
<72:65>	ASID	RO	—	Address Space ID of the sampled PC.
<64:1>	PC	RO	—	Program counter value.
<0>	New	R/W0	—	Processor writes a 1 to this field whenever a new sample is written into this register. The probe replaces with a 0 when it reads out the sample value. Used to detect a duplicate sample read operation on the probe side.

EJTAG Boot Indication

EJTAG Boot is Implemented.

Bypass Register

Implemented. The Bypass register is a 1-bit read-only register, which provides a minimum shift path through the TAP.

Fastdata Register

Implemented. The width of the Fastdata register is 1-bit. During a Fastdata access, the Fastdata register is written and read, i.e., a bit is shifted in and a bit is shifted out.

Cavium Confidential For Bowen Yang ALU 02/27/2014

2.19 Pipelines

The cnMIPS III cores are dual-issue processors. Either pipe may issue the first of the two instructions in the program. With the cnMIPS III cores, both Pipeline 0 and Pipeline 1 have dedicated instruction classes as defined below.

The integer instruction classes are shown in Table 2–43 and the pipeline classification for CP0 register access is shown in Table 2–44.

Table 2–43 Integer Instruction Classes

Instr. Class	Instructions
Arith	= CPU Arithmetic (Table 2–8, except Imul*/Idiv and Set instructions listed below), CPU Logical Instructions (Table 2–13), CPU Instruction Control (Table 2–10), CPU Insert/Extract Instructions (Table 2–14), CPU shift (Table 2–16)
Set	= SLT, SLTI, SLTU, SLTIU, SEQ, SNE, SEQI, SNEI
Cmov	= MOVN, MOVZ
J	= Jumps: J, JR, JAL, JALR.HB, JR.HB, CPU Trap Instructions (Table 2–9)
Br	= Ordinary branches: BEQ (B), BGEZ, BGTZ, BGEZAL (BAL), BLTZAL, BLEZ, BLTZ, BNE, BBIT0, BBIT032, BBIT1, BBIT132
BrL	= Deprecated branch likelies: BEQL, BGEZALL, BGEZL, BGTZL, BLEZL, BLTZALL, BLTZL, BNEL
Load/Store	= CPU Load, Store, and Memory Control Instructions (Table 2–11, except ULD,ULW,USD,USW)
ULoad/UStore	= ULD,ULW,USD,USW
MFCP0I	= MFC0 or DMFC0 from I, SI, or TI Type CP0 register, RDHWR, RDPGPR, WRPGPR
MTCP0I	= MTC0 or DMTC0 to I Type CP0 register
MTCP0SI	= MTC0 or DMTC0 to SI Type CP0 register, EI, DI
MTCP0TI	= MTC0 or DMTC0 to TI Type CP0 register, PAUSE, WAIT, SYNCI
MFCP0M	= MFC0 or DMFC0 from M, SM, or TM Type CP0 register
MTCPOM	= MTC0 or DMTC0 to M Type CP0 register, ERET, DERET, SDBBP, TLBWI, TLBWR
MTCP0SM	= MTC0 or DMTC0 to SM Type CP0 register
MTCP0TM	= MTC0 or DMTC0 to TM Type CP0 register, TLBR, TLBP, CACHE
MFCP2	= DMFC2
MTCP2	= DMTC2
Imul	= MUL, DMUL, VMULU, MFHI, MFLO, POP, DPOP, CLZ, CLO, DCLZ, DCLO, MTM0, MTM1, MTM2, MTP0, MTP1, MTP2, MULT, MULTU, MTHI, MTLO, MADD, MADDU, MSUB, MSUBU, QMAC.00, QMAC.01, QMAC.02, QMAC.03, QMACS.00, QMACS.01, QMACS.02, QMACS.03
Imul2	= DMULT, DMULTU
Imul3	= V3MULU
Imul4	= VMM0
Idiv	= DIV, DIVU
IdivD	= DDIV, DDIVU

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 2–44 Pipeline Classification for CP0 Register Access

CP0 Register	Type	CP0 Register	Type	CP0 Register	Type
Index	M	IntCtl	SI	PerfCnt Counter0	I
UtlbIndex	M	SRSCtl	SI	PerfCnt Control1	TI
Random	M	GuestCtl0	SI	PerfCnt Counter1	I
EntryLo0	M	GTOffset	SI	PerfCnt Control2	TI
UtlbEntry1	M	Cause	SI	PerfCnt Counter2	I
EntryLo1	M	EPC	I	PerfCnt Control3	TI
UtlbEntry2	M	PrID	I	PerfCnt Counter3	I
Context	M	EBase	I	ErrCtl	TM
UserLocal	I	Config	M	CacheErr (icache)	I
PageMask	M	Config1	M	CacheErr (dcache)	M
PageGrain	M	Config2	M	IcacheDebug	I
Wired	M	Config3	M	TagLo (icache)	I
HWREna	I	Config4	M	DataLo (icache)	I
BadInstr	I	Config5	M	TagLo (dcache)	M
BadInstrP	I	CvmVMConfig	TM	DataLo (dcache)	M
BadVAddr	M	CvmMemCtl2	TM	TagLo1 (dcache)	M
Count	I	WatchLo0	SI	DataHi (icache)	I
CvmCount	I	WatchLo1	SM	TagHi (dcache)	M
CvmCtl	TI	WatchHi0	SI	DataHi (dcache)	M
EntryHi	TM	WatchHi1	SM	ErrorEPC	I
UtlbEntry0	TM	XContext	M	DESAVE	I
GuestCtl2	I	MultiCoreDebug	I	KScratch1	I
Compare	I	CvmCountOffset	I	KScratch2	I
GuestCtl0Ext	I	Debug	TI	KScratch3	I
PowThrottle	I	Debug2	TI	KScratch4	I
CvmMemCtl	TM	DEPC	I		
Status	SI	PerfCnt Control0	TI		

Cavium Confidential For Bowen Yang ALU 02/27/2014

The floating-point instruction classes are shown in Table 2–45.

Table 2–45 Floating-Point Instruction Classes

Instr. Class	Instructions
FPAddSubCvt	ADD.S, ADD.D, SUB.S, SUB.D, {CONVERT}.D.L,{CONVERT}.D.S, {CONVERT}.D.W, {CONVERT}.L.D, {CONVERT}.L.S, {CONVERT}.L.W, {CONVERT}.S.D, {CONVERT}.S.L, {CONVERT}.S.W, {CONVERT}.W.D, {CONVERT}.W.L, {CONVERT}.W.S, where {CONVERT} = CVT, ROUND, TRUNC, CEIL, or FLOOR
FPAbsNegMov	ABS.S, ABS.D, NEG.S, NEG.D, MOV.S, MOV.D, MOVF.S, MOVF, D, MOVT.S, MOVT.D, MOVN.S, MOVN.D, MOVZ.S, MOVZ.D, MOVN, MOVZ
FPMul	MUL.S, MUL.D
FPMadd	MADD.S, MADD.D, MSUB.S, MSUB.D, NMADD.S, NMADD.D, NMSUB.S, NMSUB.D
FPCond	BC1F, BC1T, BC1FL, BC1TL, C.{COND}.S, C.{COND}.D, where {COND} = F, UN, EQ,UEQ, OLT, ULT, OLE, ULE, SF, NGLE, SEQ, NGL, LT, NGE, LE, NGT
FPDivSingle	DIV.S, RECIP.S
FPDivDouble	DIV.D, RECIP.D
FPSqrtSingle	SQRT.S
FPSqrtDouble	SQRT.D
FPRsqrtSingle	RSQRT.S
FPRsqrtDouble	RSQRT.D
FPMTC1	CTC1, DMTC1, MTC1, MTHC1
FPMFC1	DMFC1,MFC1, MFHC1
FPCFC1	CFC1
FPLoad	LDC1, LWC1, LDXC1, LUXC1, LWXC1
FPStore	SDC1, SWC1, SDXC1, SUXC1, SWXC1

Cavium Confidential For Bowen Yang ALI 02/27/2014

Table 2–46 shows the instruction latency and the busy cycles for the instruction classes in pipeline 0.

Table 2–46 Instruction Latency/Busy Cycles for Pipeline 0

Class	Result Latency (core cycles)	Unit Busy (core cycles)	Comments
Arith	1	1	Fully pipelined.
Set	0 or 1	1	A Set class instruction on pipe 0 may dual-issue with a Br class instruction on pipe 1 that consumes the Set instruction result. The Br instruction must consume the result register via the rs instruction field. Otherwise, the Set result latency is always 1 cycle. Fully pipelined.
Cmov	3	1	Fully pipelined.
Load/Store	3	1	Fully pipelined.
ULoad/UStore	4	2	Takes two cycles on pipe 0. No instruction can dual-issue on pipe 1 during the first of the two cycles.
MFCP0M	3	1	Can only dual issue with subsequent Arith, Set, and Cmov instructions. MFCP0M must be at least 3 cycles after MTCPOM instructions. MFCP0M must be at least 2 cycles after TLBR, TLBP, or TLBWI.
MTCPOM	N/A	1	Can only dual issue with subsequent Arith, Set, and Cmov instructions.
MTCP0SM	N/A	4	Can only dual issue with subsequent Arith, Set, and Cmov instructions. The next instruction may execute in 4 cycles.
MTCP0TM	N/A	11	Can only dual issue with subsequent Arith, Set, and Cmov instructions. A pipe flush occurs after the instruction issue. The next instruction may execute in 11 cycles.
FPLoad	4	1	Fully pipelined, but load cannot issue in same cycle if a Floating Point instruction prior in program order has issued in the same cycle.
FPStore	4	1	Fully pipelined, but Pipeline 1 cannot issue any other instruction in same cycle.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 2–47 shows the instruction latency and the busy cycles for the instruction classes in pipeline 1.

Table 2–47 Instruction Latency/Busy Cycles for Pipeline 1

Class	Result Latency (core cycles)	Unit Busy (core cycles)	Comments
Arith	1	1	Fully pipelined.
Set	1	1	Fully pipelined.
Cmov	3	1	Fully pipelined.
J	1	1	For JAL, the cnMIPS III core pushes $31 onto the subroutine call stack. For JR $31 and JR.HB $31, the cnMIPS III core predicts the target from the top of its 8-entry subroutine return call stack and pops the stack. For JALR $31 and JALR.HB $31, the cnMIPS III core predicts the target from its jump table, pushes $31 onto the subroutine call return stack, and updates the jump table with the actual target. For other JR and JALR instructions, the cnMIPS III core predicts the target from the jump table and updates the jump table with the actual target. vpc<13:6> XOR vpc<21:14> selects one of the 256 jump table locations to be referenced by JR, JALR, and JALR.HB instructions. With J, JAL, and correctly-predicted jumps, the cnMIPS III core may issue target instructions as soon as the next cycle. With any mispredicted jump or taken trap instruction, the cnMIPS III core flushes the pipe. A mispredicted jump target may issue 7 cycles after the delay slot of the branch issues (8 cycles after the jump issues). The delay slot cannot dual-issue with their corresponding JR, JR.HB, JALR, and JALR.HB instructions.
Br	1	1	The cnMIPS III core always predicts unconditional branch (B or BAL) instructions taken. Otherwise, it dynamically predicts whether the branch is taken, updating the prediction based on the actual. vpc<13:3> XOR (vpc<21:14> << 3) selects one of the 2048 2-bit branch prediction table entries. With unconditional and correctly-predicted branches, the cnMIPS III core may issue target instructions as soon as the next cycle. A mispredicted branch target may issue 7 cycles after the delay slot of the branch issues. For BLTZAL and BGEZAL (BAL), the cnMIPS III core pushes $31 onto the subroutine stack. Can dual-issue with and consume the result of a Set. The Br instruction must consume the Set result via the rs instruction field.
BrL	1	1	Deprecated instructions.
MFCP0I	1	1	Can only dual issue with subsequent Arith, Set, and Cmov instructions. MFCP0I cannot issue until 4 cycles after MTCP0I. MFCP0I cannot issue until 4 cycles after an EI, DI, or ERET instruction.
MTCP0I	N/A	1	Must single-issue. MTCP0I cannot issue until 3 cycles after the last Load/Store or 4 cycles after the last ULoad/UStore instruction.
MTCP0SI	N/A	4	Must single-issue. The next instruction may execute in 4 cycles.
MTCP0TI	N/A	11	Must single-issue. A pipe flush occurs after the instruction issue. The next instruction may execute in 11 cycles.
MFCP2	6	1	Fully pipelined.
MTCP2	N/A	See Section 2.21	MTCP2 instructions can cause CRC and crypto operations to start, which can have various unit busy times. See Section 2.21.
Imul	6	1	Fully pipelined.

Cavium Confidential For Bowen Yang ALU 02/21/2014

Table 2–47 Instruction Latency/Busy Cycles for Pipeline 1 (Continued)

Class	Result Latency (core cycles)	Unit Busy (core cycles)	Comments
Imul2	N/A	2	No Imul, Imul2, Imul3, Imul4, Idiv nor IdivD may issue for 2 cycles.
Imul3	6	3	No Imul, Imul2, Imul3, Imul4, Idiv nor IdivD may issue for 3 cycles.
Imul4	6	4	No Imul, Imul2, Imul3, Imul4, Idiv nor IdivD may issue for 4 cycles.
Idiv	N/A	18	No Imul, Imul2, Imul3, Imul4, Idiv nor IdivD may issue for 18 cycles.
IdivD	N/A	35	No Imul, Imul2, Imul3, Imul4, Idiv nor IdivD may issue for 35 cycles.
FPAddSubCvt	4	1	Fully pipelined. Issue may be delayed by one clock if register file write port busy.
FPAbsNegMov	2	1	Fully pipelined. Issue may be delayed by one clock if register file write port busy.
FPMul	5	1	Fully pipelined. Issue may be delayed by one clock if register file write port busy.
FPMadd	9	2	Partially pipelined. Issue may be delayed by one clock if register file write port busy.
FPCond	1	1	Fully pipelined. Issue may be delayed by one clock if register file write port busy.
FPDivSingle	12	9	Partially pipelined. Issue may be delayed by one clock if register file write port busy.
FPDivDouble	22	19	Partially pipelined. Issue may be delayed by one clock if register file write port busy.
FPSqrtSingle	16	13	Partially pipelined. Issue may be delayed by one clock if register file write port busy.
FPSqrtDouble	30	27	Partially pipelined. Issue may be delayed by one clock if register file write port busy.
FPRsqrtSingle	27	24	Partially pipelined. Issue may be delayed by one clock if register file write port busy.
FPRsqrtDouble	51	48	Partially pipelined. Issue may be delayed by one clock if register file write port busy.
FPMTC1	2	1	Fully pipelined. Issue may be delayed by one clock if register file write port busy.
FPMFC1	6	1	Fully pipelined
FPCFC1	6	1	CFC1 will not issue for 12 cycles after the last FPMadd issue and 5 cycles after any other Floating Point instruction.

Cavium Confidential For Bowen Yang ALU 02/27/2014

2.20 Special MUL Topics

The following sequence can be used to save/restore multiplier context.

```
// save multiplier context P0-P5, MPL0-MPL5, HI, LO
v3mulu   $1, $0, $0
v3mulu   $2, $0, $0
v3mulu   $3, $0, $0
v3mulu   $4, $0, $0
sd       $1, 0($5)   // P0
v3mulu   $1, $0, $0
sd       $2, 8($5)   // P1
v3mulu   $2, $0, $0
sd       $3, 16($5)  // P2
ori      $3, $0, 1
v3mulu   $3, $3, $0
sd       $4, 24($5)  // P3
v3mulu   $4, $0, $0
sd       $1, 32($5)  // P4
v3mulu   $1, $0, $0
sd       $2, 40($5)  // P5
v3mulu   $2, $0, $0
sd       $3, 48($5)  // MPL0
v3mulu   $3, $0, $0
sd       $4, 56($5)  // MPL1
v3mulu   $4, $0, $0
sd       $1, 64($5)  // MPL2
mflo     $1
sd       $2, 72($5)  // MPL3
mfhi     $2
sd       $3, 80($5)  // MPL4
sd       $4, 88($5)  // MPL5
sd       $1, 96($5)  // LO
sd       $2, 104($5) // HI

// restore multiplier context P0-P5, MPL0-MPL5, HI, LO
ld       $1, 96($5)  // LO
ld       $2, 104($5) // HI
ld       $3, 48($5)  // MPL0
ld       $4, 72($5)  // MPL3
MTLO     $1
ld       $1, 56($5)  // MPL1
MTHI     $2
ld       $2, 80($5)  // MPL4
MTM0     $3, $4
ld       $3, 64($5)  // MPL2
ld       $4, 88($5)  // MPL5
MTM1     $1, $2
ld       $1, 0($5)   // P0
ld       $2, 24($5)  // P3
MTM2     $3, $4
ld       $3, 8($5)   // P1
ld       $4, 32($5)  // P4
MTP0     $1, $2
ld       $1, 16($5)  // P2
ld       $2, 40($5)  // P5
MTP1     $3, $4
MTP2     $1, $2
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

2.21 COP2 Latencies

Table 2–48 through Table 2–55 list the latencies (i.e. the number of clock cycles until the next issue to the unit) of various COP2 class of instructions (DMFC2 and DMTC2, refer to Table 2–7).

NOTE: If a COP2-class instruction is not listed in this section, it has a latency of one cycle.

Table 2–48 lists the 3DES unit latencies.

Table 2–48 3DES Unit Latencies

Operation	Number of Cycles	
	If followed by DMFC2	If followed by DMTC2
CVM_MT_3DES_DEC	48	49
CVM_MT_3DES_DEC_CBC	48	49
CVM_MT_3DES_ENC	48	49
CVM_MT_3DES_ENC_CBC	48	49

Table 2–49 lists the KASUMI unit latencies.

Table 2–49 KAS Unit Latencies

Operation	Number of Cycles	
	If followed by DMFC2	If followed by DMTC2
CVM_MT_KAS_ENC	32	33
CVM_MT_KAS_ENC_CBC	32	33

Table 2–50 lists the latencies of the COP2 instructions that use the AES unit; they depend on key length, encrypt or decrypt, whether the operation is the first after a key load, and whether the operation is followed by a DMFC2 or a DMTC2. Latencies are listed for the following operations:

- CVM_MT_AES_DEC_CBC1
- CVM_MT_AES_DEC1
- CVM_MT_AES_ENC_CBC1
- CVM_MT_AES_ENC1

Table 2–50 AES Unit Latencies

AES Key Length	First Operation After Key Load?	Encrypt/ Decrypt	Number of Cycles	
			If followed by DMFC2	If followed by DMTC2
128-bit	No	—	32	34
	Yes	Decrypt	142	144
	Yes	Encrypt	66	68
192-bit	No	—	38	40
	Yes	Decrypt	166	168
	Yes	Encrypt	75	77
256-bit	No	—	44	46
	Yes	Decrypt	207	209
	Yes	Encrypt	103	105

Table 2–51 lists the latencies of the CVM_MT_GFM_XORMUL1 instruction.

Table 2–51 CVM_MT_GFM_XORMUL1 Latencies

GFM POLYNOMIAL Length	Number of Cycles	
	If followed by DMFC2	If followed by DMTC2
64-bit	18	19
128-bit	34	35

Cavium Confidential For Bowen Yang BALU 02/27/2014

Table 2–52 lists the CRC unit latencies.

Table 2–52 CRC Unit Latencies

Operation	Number of Cycles	
	If followed by DMFC2	If followed by DMTC2
CVM_MT_CRC_BYTE	2	2
CVM_MT_CRC_BYTE_REFLECT	2	2
CVM_MT_CRC_DWORD	4	4
CVM_MT_CRC_DWORD_REFLECT	4	4
CVM_MT_CRC_HALF	2	2
CVM_MT_CRC_HALF_REFLECT	2	2
CVM_MT_CRC_VAR	4	4
CVM_MT_CRC_VAR_REFLECT	4	4
CVM_MT_CRC_WORD	2	2
CVM_MT_CRC_WORD_REFLECT	2	2
CVM_MT_CRC_POLYNOMIAL	4	4
CVM_MT_CRC_POLYNOMIAL_REFLECT	4	4

Table 2–53 lists the Hash (MD5/SHA-1/SHA-3/SHA-256/SHA-512) unit latencies.

Table 2–53 Hash Latencies

Operation	Number of Cycles	
	If followed by DMFC2	If followed by DMTC2[1]
CVM_MT_HSH_STARTMD5	200	192
CVM_MT_HSH_STARTSHA	90	82
CVM_MT_HSH_STARTSHA256	140	132
CVM_MT_HSH_STARTSHA512	172	161
CVM_MT_SHA3_STARTOP	25	25

1. For information on the legal order of DMTC2 operations, see the **Restrictions** sections for the CVM_MT_HSH_START* instructions in the Hash (MD5/SHA-1/SHA-256/SHA-512) section on page 1805 of Appendix A.

Table 2–54 lists the SNOW3G unit latencies.

Table 2–54 SNOW3G Latencies

Operation	Number of Cycles	
	If followed by DMFC2	If followed by a DMTC2
CVM_MT_SNOW3G_START	73	73
CVM_MT_SNOW3G_MORE	7	7

Table 2–55 lists the latencies of the COP2 instructions that use the SMS4 unit, which depend on whether the operation is the first after a key load, and whether the operation is followed by a DMFC2 or a DMTC2. Latencies are listed for the following operations:

- CVM_MT_SMS4_DEC_CBC1
- CVM_MT_SMS4_DEC1
- CVM_MT_SMS4_ENC_CBC1
- CVM_MT_SMS4_ENC1

Table 2–55 SMS4 Latencies

First Operation After Key Load?	Number of Cycles	
	If followed by DMFC2	If followed by DMTC2
No	65	66
Yes	135	136

Cavium Confidential For Bowen Yang ALU 02/27/2015

Table 2–56 lists the CAMELLIA unit latencies.

Table 2–56 CAMELLIA Unit Latencies

Operation	Number of Cycles	
	If followed by DMFC2	If followed by DMTC2
CVM_MT_CAMELLIA_ROUND	1	1
CVM_MT_CAMELLIA_FL	4	4
CVM_MT_CAMELLIA_FLINV	4	4

Table 2–57 lists the ZUC unit latencies.

Table 2–57 ZUC Latencies

Operation	Number of Cycles	
	If followed by DMFC2	If followed by a DMTC2
CVM_MT_ZUC_START	73	73
CVM_MT_ZUC_MORE	7	7

2.22 Hardware Debug Features

The cnMIPS III core's hardware debug support includes the traditional MIPS debug features, MIPS EJTAG features, and OCLA features. *Refer to EJTAG Specification, Revision 5.04*. The traditional MIPS debug features provide compatibility with existing debuggers. The cnMIPS III core's EJTAG support includes cnMIPS III core-specific extensions that enable concurrent multicore debugging.

Non-EJTAG cnMIPS™ II Core Hardware Features

The traditional MIPS debug features are listed in Table 2–58, and the EJTAG debug features are listed in Table 2–59.

Table 2–58 MIPS Debug Features

Feature	Implemented / Comments
BREAK instruction	Implemented
Trap instructions	Implemented
Watch registers	one instruction-only, one data-only implemented
MMU Traps	Implemented

EJTAG Hardware Debug Features:

The EJTAG hardware-debug features are listed in Table 2–59.

Table 2–59 EJTAG Debug Features

Feature	Implemented / Comments
EJTAG dmseg	Implemented
EJTAG drseg	Implemented
EJTAG inst breakpoints	Four implemented
EJTAG data breakpoints	Four implemented, both address and value. The EJTAG data value breakpoints are not precise for load operations.
EJTAG single step	Implemented
EJTAG debug interrupts	Implemented. No external wire, only through the TAP.

CN71XX includes a single JTAG external (i.e. pin) interface for EJTAG support, though each core has it's own EJTAG TAP controller.

Cavium Confidential For Bowen Yang ALU 02/07/2014

2.22.1 Multicore Debug Support

The cnMIPS III core hardware supports very fast debug interrupts, called multicore debug (i.e. MCD) interrupts, to aid debugging parallel applications.

The hardware includes three global wires, called MCD wires, that each core can pulse every cycle. Each core also samples the “wired-OR” value, each wire every cycle. When this value is a 1, each core that is not mask-disabled (via MulticoreDebug[Mask_MCD0,Mask_MCD1,Mask_MCD2]) sets a state bit corresponding to the wire (i.e. sets MulticoreDebug[MCD0,MCD1,MCD2]).

If any of MulticoreDebug[MCD0,MCD1,MCD2] are non-zero on a given core (and the core is not already in debug mode), a debug exception is requested on the core. Software can clear MulticoreDebug[MCD0, MCD1, MCD2] by writing a one to them; this is necessary to ensure that no further debug interrupts occur after exiting the debug handler.

These MCD interrupts occur at the same priority level as the debug interrupts (i.e. DINT) described in the MIPS spec. The exception location is also the same as a debug interrupt and MCD0, MCD1, and MCD2 bits are similar to the DINT bit, but the detailed behavior is different:

- The DINT bit is read-only, but MulticoreDebug[MCD0, MCD1, MCD2] must be cleared by the debug handler
- DINT must be clear when MulticoreDebug[DExcC] is set, but MulticoreDebug[MCD0,MCD1,MCD2] need not be.

There are four ways that the global MCD wires can be pulsed:

1. Software can write a one to the Pls0, Pls1, or Pls2 bits to pulse any combination of the three MCD wires.
2. If MulticoreDebug[GSDB] is set and a SDBBP instruction is executed, the hardware pulses the MCD0 wire.
3. If IBCn/DBCn[TE] is set and a breakpoint matches, the hardware pulses the MCD0 wire. Table 2–60 describes the detailed behavior on a breakpoint match based on the IBCn/DBCn[BE,TE] value:

Table 2–60 Breakpoint Match Behavior

BE	TE	Comment
0	0	Nothing happens on a match
0	1	MCD0 is pulsed on a match. BS bits are also set in IBS/DBS. No direct local exception occurs. (This mode may not be used.)
1	0	A local breakpoint exception occurs due to the breakpoint match, causing the core to enter debug mode. MCD0 is not pulsed. BS bits are set in IBS/DBS. (This mode will be used when debugging, but not multicore.)
1	1	A local breakpoint exception occurs due to the breakpoint match, causing the core to enter debug mode. MCD0 is also pulsed. BS bits are also set in IBS/DBS. (This mode will be used when debugging multicore.)

4. The on-chip logic analyzer (OCLA) can pulse any combination of the three MCD wires.

The debug exception handler can follow these rules to determine the cause(s) of a given debug exception after reading the Debug and/or MulticoreDebug registers:

1. Any of MulticoreDebug[MCD0,MCD1,MCD2] could be set at any time, indicating that the corresponding MCD state bit is set.

Cavium Confidential For Bowen Yang ALU 02/27/2014

2. If MulticoreDebug[DExcC] is set, all of Debug[DDBSImpr, DDBLImpr, DINT, DIB, DDBS, DDBL, DBp, DSS] will be clear, and Debug[DExcCode] will contain a valid code. (This is the case for a debug mode exception.)
3. If none of Debug[DDBSImpr,DDBLImpr, DINT, DIB, DDBS, DDBL, DBp, DSS] are set, then the exception was either due to MCD*, or MulticoreDebug[DExcC] being set and Debug[DExcCode] is valid.
4. No more than one of Debug[DIB, DDBS, DDBL, DBp, DSS] can be set.
5. If MulticoreDebug[DExcC] is clear, any combination of Debug[DDBLImpr, DINT] may be set.
6. At least one of Debug[DDBLImpr, DINT, DIB, DDBS, DDBL, DBp, DSS] and MulticoreDebug[MCD0, MCD1, MCD2, DExcC] will be set.

2.22.2 System Debug Characteristics

Note that the Root.Count and Root.CvmCount CP0 registers do not increment if CIU_GSTOP[GSTOP] is set and the Root.MultiCoreDebug[CGSTP, CvGSTP] bits, respectively, are set in the core. GSTOP provides a way for debug code to stop all cycle counters on all cores. Additionally, Root.CvmCount does not increment if the core is in debug mode (i.e. if Root.Debug[DM] = 1) when Root.MultiCoreDebug[CvDM] = 1 in the core, and Root.Count does not increment if the core is in debug mode when Root.Debug[CountDM] = 1 in the core.

Though cnMIPS III cores implement these clock management functions to make the core instruction flow while debugging as similarly as possible to the non-debug instruction flow, there still are many ways that debug timing and instruction flows will differ from non-debug. No other CN71XX hardware is affected by debug mode or GSTOP: packets may continue to arrive, hardware coprocessor units will continue to operate, and external interrupts (e.g. PCIe) may arrive while the cores are in debug mode and/or GSTOP is enabled.

Another significant factor affecting debug/nondebug instruction flow is the impact of delayed load/IOBDMA operations. The cores can issue these delayed load/IOBDMA operations as follows:

- Fetch & Add Unit (FAU) requests have the tagwait feature. With this feature, the hardware attempts to delay servicing the FAU request until after the last pending tag switch occurs. Refer to “Fetch and Add Unit (FAU)” on page 280.
- New work requests have a wait feature. With this feature, the hardware attempts to delay the new work request until work is available.

These delayed load/IOBDMA operations have time-outs that fail the request. The time-out is necessary to ensure that subsequent SYNCIOBDMA, SYNCIOALL, or SYNC requests do not hang forever. (If they did hang, the core would no longer be able to execute any instructions.) These time-outs are necessarily shorter than any other time-outs so that the cores can respond quickly to interrupts, exceptions, and any other flow redirections. During debug it may not only be probable for these time-outs to expire, but also perhaps necessary for these time-outs to expire, to make forward progress. Refer to Section 7.10, “Forward Progress Constraints”. This may cause considerable changes to the instruction flow while debugging.

Note that, the core-debug-mode handler need not use noncacheable references to avoid invalidating load-linked/store-conditional sequences, as the MIPS specifications suggest. The debug handler can use ordinary cached loads/stores without invalidating load-linked / store conditional sequences.

Cavium Confidential For Review Only 02/7/2014

2.22.3 CPU On-Chip Logic Analyzer (OCLA) Samples

The cnMIPS III cores can sample internal information at a rate of about one sample every 33 cycles and store them as OCLA debug entries. These samples can include the PC. When possible and there is a difference in the data to sample, the core sends a sample. Chapter 6 describes OCLA.

MultiCoreDebug[OclaLoSel,OclaHiSel] selects what is sampled and sent to OCLA. The hardware preferentially sends OclaHiSel samples to OCLA before it sends an OclaLoSel sample to OCLA.

When MultiCoreDebug[OclaIgnPC] is set, the PC can change without generating an OCLA debug entry.

2.23 Load-Linked/Store-Conditional and Wide Atomic Memory Write Operations

This section describes both ordinary atomic memory sequences and Wide Atomic Memory Write sequences, both of which use the load-linked and store-conditional instructions.

Both sequences begin with a load-linked instruction. A wide atomic memory write sequence starts with an LL/LLD with a destination register of $0. Other LL/LLD instructions initiate an ordinary atomic memory sequence. In MIPS, the register $0 always has value zero. So one could say that an LL/LLD $0 has no destination register nor result, and we can say that wide atomic memory write sequences are initiated by an LL/LLD with no result.

LL/LLD instructions to I/O space physical addresses cannot initiate any sequence, nor can SC/SCD instructions to I/O space physical addresses terminate any sequence.

2.23.1 Ordinary (MIPS-defined) Atomic Memory Sequences

LL and LLD instructions (with a non-$0 destination/result) to DRAM set the lock flag. The hardware can clear the lock flag under any of the following conditions:

- Another core writes the same (128-byte) cache block, causing the block to be invalidated from this cache.
- ERET execution.
- SC/SCD execution.
- A store operation to DRSEG, DMSEG, CVMSEG LM, or CVMSEG I/O.
- a memory atomic or store atomic add instruction executes.
- PREF/PREFX 29, 30, or 31.
- The start of a wide atomic memory write sequence (i.e. LL/LLD $0).
- TLBWI/TLBWR execution.
- Increase in the maximum hardware CVMSEG LM size.
- Anything that invalidates the entire Dcache:
  - CACHE (op == 1, 9, 17, 21)
  - Status[ERL] change (guest or root)

Cavium Confidential For Bowen Yang ABU 2/27/2014

2.23.2 Wide Atomic Memory Write Operations

The largest store instruction in the MIPS64 instruction set is 64-bits. Wide atomic memory write operations can effectively implement larger store instructions, such as 128-bit or 256-bit, via multi-instruction sequences. Importantly, the wide atomic store operation either executes atomically, or doesn't execute at all, much as if it were a single instruction rather than an instruction sequence.

A terminating SC/SCD instruction provides data for the wide atomic write operation. On a success, the hardware combines the CVMSEG LM LMTLINE write operations during the sequence with the SC/SCD write operation, and writes them all to memory atomically. LMTLINE is a naturally aligned 128-byte cache block of data in CVMSEG LM selected by Root.CvmMemCtl[LMTLINE]. CVMSEG LM, its size, and its enables are described in Section 2.5. Software must position the LMTLINE store operations such that their relative position in the LMTLINE cache block match their relative position within the cache block when written to memory. When enabled, the first CVMSEG LM LMTLINE location at the CVMSEG LM virtual address:

0xFFFF FFFF FFFF 8000 + (Root.CvmMemCtl[LMTLINE] << 7)

and the remaining LMTLINE data is in subsequent addresses.

NOTE: Guest.CvmMemCtl[LMTLINE] cannot be modified and always equals Root.CvmMemCtl[LMTLINE].

The hardware guarantees atomicity by failing the SC/SCD instruction that terminates the wide atomic memory write sequence when necessary.

Here is an example wide atomic memory write-instruction sequence implementing an atomic 128-bit memory write:

```
// assume:
// $1    = the (naturally-aligned) memory address to write the 128-bit word to
// $4,$5 = the 128-bit word to be written
ANDI $2, $1, 0x7f // 0x7f because LMTLINE is 128 bytes again:
LLD  $0, 0($1)    // initiate the sequence
OR   $3, $4, $4
SD   $3, LMTLINECONSTANT($2) // 64 of the 128-bits written to LMTLINE
OR   $3, $5, $5
SCD  $3, 8($1)   // hardware combines these 64-bits with the 64-bits in LMTLINE
BEQ  $3, again   // try again if an interrupt occurs
NOP
```

For example, if Root.CvmMemCtl[LMTLINE] were 0, LMTLINECONSTANT would be –32768 (integer), which would fit in the offset field of the SD instruction.

In the example sequence, the ANDI instruction determines the offset of the memory location into LMTLINE. The LLD $0 initiates the sequence, also providing the virtual address of the cache block that will be written in memory on success. The SD instruction stores 64-bits to the LMTLINE location corresponding to where it will be written into memory. The SCD instruction provides the additional 64-bits and the virtual address. Upon success in this example, the hardware combines the two 64-bit store operations and performs one atomic 128-bit write to memory.

This example could be extended to writing any naturally-aligned word size up to and including 1024-bits by including more SD instructions to LMTLINE in the sequence.

Cavium Confidential For Bowen Yang ALU 04/27/2014

2.23.3 Constraints for a Wide Atomic Memory Write Sequence:

- An ERET while a wide atomic memory write is active terminates it. Any interrupt while a wide atomic memory write sequence is active might also terminate the sequence, and may leave the value of the LMTLINE locations written as part of the sequence prior to the interrupt unpredictable.
- Any SC/SCD terminates an active wide atomic memory sequence.
- The start of an ordinary atomic memory sequence (i.e. LL/LLD non-$0) terminates the wide atomic memory sequence. An ordinary atomic memory sequence cannot be in-flight simultaneously with a wide atomic memory sequence.
- Results are unpredictable unless there is at least one SW/SD to LMTLINEn in the sequence between the LL/LLD and SC/SCD.
- Results are unpredictable unless the CVMSEG LM LMTLINE locations are all written in lowest-to-highest address order. Results are also unpredictable if bits<6:0> of the effective address of the SC/SCD instruction that terminates the wide atomic write sequence is less than or equal to bits<6:0> of any CVMSEG LM LMTLINE store operations in the sequence.
- If LLD initiates the wide atomic write sequence, results are unpredictable if anything other than a SCD terminates the sequence, or if anything other than naturally aligned SDs are used to store to LMTLINE during the sequence.
- If LL initiates the wide atomic write sequence, results are unpredictable if anything other than an SC terminates the sequence, or if anything other than naturally aligned SWs are used to store to LMTLINE during the sequence.
- Results are unpredictable if <63:7> of the LL/LLD effective virtual address does not match <63:7> of the SC/SCD effective virtual address.
- The CVMSEG LM addresses for the LMTLINE data must be valid, i.e.

  Root.CvmMemCtl[LMTLINE] < LMEMSZ,

  where LMEMSZ is Root.CvmMemCtl[LMEMSZ] or Guest.CvmMemCtl[LMEMSZ], depending on root or guest mode, respectively.

- Results are unpredictable if any IOBDMA operation results return into the LMTLINE during the wide atomic memory write sequence, or if any IOBDMA operation that will return any results into LMTLINE is created during the sequence.
- Wide atomic memory write sequences share the CVMSEG LM LMTLINE data with LMTDMAs and LMTSTs. An LMTDMA/LMTST initiated while a wide atomic memory write sequence is active might cause the wide atomic memory write sequence to malfunction.

  Software must ensure that either there is no active wide atomic memory write operation when it executes the LMTDMA/LMTST, or that it cannot succeed. (A subsequent ERET, for example, can ensure that an inflight wide atomic memory write operation cannot succeed.) A wide atomic memory write sequence initiated in the middle of an LMTDMA/LMTST initiation sequence might cause an exception to occur on the initiation of the LMTDMA/LMTST.

  However, a wide atomic memory write operation can execute in an interrupt handler without disturbing an interrupted LMTDMA/LMTST operation, provided the LMTLINE is restored after the wide atomic memory write. Section 2.23.2 describes other LMTLINE save/restore requirements.

Cavium Confidential PDF Bowen Yang 6/27/2014

- The values of the LMTLINE locations written during the wide atomic write sequence are unpredictable after a successful sequence. (Since an interrupt can also make the LMTLINE locations that were previously written during a wide atomic write sequence unpredictable, software should also consider the value of the written LMTLINE locations to be unpredictable after a failing wide atomic store sequence.) The value of the LMTLINE locations corresponding to bits<6:0> of the SC/SCD address are unpredictable after a successful wide atomic write sequence.
- A TLBMOD exception may or may not occur when attempting a LL/LLD $0 to a memory location for which write access is not allowed. A TLBRI exception may or may not occur when attempting a LL/LLD $0 to a memory location for which read access is not allowed.

2.24 Exceptions

Table 2–61 lists all the MIPS-defined and Cavium-specific exceptions, and whether they are implemented on the cnMIPS III core. This lists shows the relative priority of each exception, highest to lowest.

Table 2–61 cnMIPS III Core Exceptions

Exception	Implemented/ Not Implemented	Comments
Reset	Implemented	
Soft Reset	Implemented	cnMIPS III cores can be reset from a chip-wide warm reset, chip-wide soft reset (CIU_SOFT_RST[SOFT_RST]), watchdog time-out expiration, core-selected soft reset (CIU_PP_RST[RST,RST0]), or core-local EJTAG TAP soft reset (ECR[PrRst]). In the core-selected and core-local soft reset cases (where only the local core is reset), the coherent memory interconnect and other coherent bus users are not reset. If the core being reset is actively using the bus the chip behavior could be adversely affected. Before performing a core-selected or core-local soft reset, ensure that the core being reset has executed a SYNCIOBDMA instruction, and has entered an idle loop.
Debug Single Step	Implemented	
Debug Interrupt (DINT)	Implemented	cnMIPS III cores can receive core-selected debug interrupts (CIU_DINT[DINT]), EJTAG TAP ECR[EjtagBrk] debug interrupts, and can take a debug interrupt from EJTAGBOOT.
Multicore Debug (MCD) Interrupt	Cavium-specific	Multicore debug interrupts are very similar to DINT, except that Debug[DINT] is not set after the interrupt is taken. (MultiCoreDebug[MCD*] is set instead.)
Debug Data Break Load/Store Imprecise (DDBLImpr/ DDBSImpr)DDBLImpr	DDBLImpr implemented. DDBSImpr not implemented.	cnMIPS III core's EJTAG breakpoints are all precise except for load value matches.
Non-Maskable Interrupts	Implemented	cnMIPS III core may receive core-selected non-maskable interrupts (CIU_NMI[NMI]) or may receive a non-maskable interrupt from the CIU watchdog for the core.
Machine Check	Implemented	cnMIPS III core implements synchronous machine-checks on TLB write operations that create multiple matching entries. Debug mode machine-checks can occur. TLB parity errors also cause MCheck ExcCodes, but can be distinguished from this exception by Status[TS] or CacheErr (Dcache) [tlbperr]. See "TLB parity error during mapped instruction fetch" and "TLB parity error during data access" below.

Cavium Confidential For Cavium Employee Use Only

Table 2–61 cnMIPS III Core Exceptions (Continued)

Exception	Implemented/ Not Implemented	Comments
Interrupt	Implemented for Cause[IP4,IP3,IP2].	cnMIPS III cores receive three interrupt wires from CIU - Cause[IP4,IP3,IP2]. The CIU combines all possible chip-wide interrupt wires into Cause[IP4,IP3,IP2]. This includes: per-group work-queue interrupts, GPIO interrupts, 2 mailbox (inter-core) interrupts, UART interrupts, PCIe interrupts, PCIe MSI interrupts, watchdog interrupts, TWSI interrupts, RML interrupts, trace buffer interrupts, GMX/IPD packet-drop interrupts, general timer interrupts. Cause[IP2], Cause[IP3], and Cause[IP4] can be programmed independently to include any combination of these interrupt sources.
Deferred Watch	Implemented	
Debug Instruction Break	Implemented	CN71XX may take an instruction breakpoint for the instruction in the delay slot of a branch-likely instruction, even though the branch was taken, and therefore the instruction was not really executed. (Note that branch-likely instructions are deprecated in the MIPS architecture and generally should not be used.)
Watch on Instruction Fetch	Implemented	CN71XX may take a watch on an instruction fetch for the instruction in the delay slot of a branch-likely instruction, even though the branch was taken, and therefore the instruction was not really executed. (Note that branch-likely instructions are deprecated in the MIPS architecture and generally should not be used.)
Address Error in Instruction Fetch	Implemented	cnMIPS III core may flush the Icache on an address error on an instruction fetch. Debug mode address error instructions can occur.
TLB Parity Error During Mapped Instruction Fetch	Cavium Specific	cnMIPS III core takes a synchronous machine check exception when it detects a TLB parity error during a mapped instruction fetch. Software can distinguish TLB parity errors from other machine checks by Status[TS] or CacheErr (Dcache) [tlbperr]. To recover from a TLB parity error, software should completely re-initialize the TLB, including any wired entries, and flush the L1 caches. See also the Machine Check exception.
TLB/XTLB Refill on Instruction Fetch	Implemented	Debug mode TLB/XTLB refill on instruction fetch exceptions can occur.
TLB Invalid on Instruction Fetch	Implemented	Debug mode TLB invalid on instruction fetch exceptions can occur.
TLB Execute-Inhibit on Instruction Fetch	Implemented	Debug mode TLB Execute-Inhibit on instruction fetch exceptions can occur.
Cache Error on Instruction Fetch	Implemented	The cnMIPS III core automatically corrects the error by flushing the Icache, so the cache error handling software need not attempt to correct the error. The Cache error register (register = 27, select = 0) contains diagnostic information. Debug-mode cache error on instruction fetch exceptions can occur.
Bus Error on Instruction Fetch	Implemented	CN71XX instruction fetch bus errors only occur on a time-out of an outstanding instruction fetch fill. These errors are generally non-recoverable. Debug mode bus error on instruction fetch exceptions can occur.
Debug Breakpoint (SDBBP execution)	Implemented	Debug mode debug breakpoint exceptions can occur.
Coprocessor Unusable	Implemented	CN71XX may generate Coprocessor unusable exceptions for all of Coprocessors 0, 1, and 2. Debug mode coprocessor unusable exceptions can occur.
MDMX Unusable	Not Implemented	CN71XX does not implement the MDMX ASE.
Reserved Instruction	Implemented	Debug mode reserved instruction exceptions can occur.
Integer Overflow	Implemented	Debug mode Integer overflow exceptions can occur.
Trap	Implemented	Debug mode Trap exceptions can occur.
System Call	Implemented	Debug mode system call exceptions can occur.
Breakpoint	Implemented	Debug mode breakpoint exceptions can occur.

Cavium Confidential For Broadwell Ying ALD 12/23/2014

Table 2–61 cnMIPS III Core Exceptions (Continued)

Exception	Implemented/ Not Implemented	Comments
Floating Point	Not Implemented	cnMIPS™ II cores do not implement floating-point.
Coprocessor 2	Not Implemented	cnMIPS III core's coprocessor 2 units do not generate exceptions.
Precise Debug Data Break	Implemented	All debug data break exceptions are precise except for load value match breakpoints.
Watch on Data Access	Implemented	
Address Error on Data Access	Implemented with Cavium-specific additions.	When CvmCtl[REPUN] is clear, the core generates address errors for the LD, LW, LWU, LH, LHU, SD, SW, and SH instructions exactly as per the MIPS specs. LBX, LBUX, LHX, LHUX, LWX, LWUX act similarly. When CvmCtl[REPUN] is set, address error exceptions due to address alignment in physical memory or CVMSEG LM (NOTE: NOT I/O or DSEG) locations do not happen for the LD, LDX, LW, LWU, LWX, LWUX, LH, LHU, LHX, LHUX, LBX, LBUX, SD, SW, and SH instructions since the core hardware automatically completes the memory unaligned loads/stores. When CvmCtl[REPUN] is set, the core generates address errors for all unaligned IO/DSEG accesses for the LD, LDX, LW, LWU, LWX, LWUX, LH, LHU, LHX, LHUX, LBX, LBUX, SD, SW, and SH instructions, as per the MIPS specifications. However, the priority of these unaligned address errors is lower than the priority of TLB parity, TLB refill, TLB invalid, and TLB modified exceptions when CvmCtl[REPUN] is set, unlike the MIPS specifications. When CvmCtl[USEUN] is set, the core generates address errors for all unaligned IO/DSEG accesses for the ULD, ULW, USD, and USW instructions. The priority of these unaligned address errors is lower than the priority of TLB parity, TLB refill, TLB invalid, and TLB modified exceptions. When enabled (by the appropriate CvmMemCtl[CVMSEGENA*]), CVMSEG I/O references by anything other than an aligned SD to the 0xFFFF FFFF FFFF A200 virtual address (i.e. any non-IOBDMA) cause address errors. When enabled (by the appropriate CvmMemCtl[CVMSEGENA*]), CVMSEG LM references larger than that allowed by CvmMemCtl[LMEMSZ] cause address errors. When enabled (by the appropriate CvmMemCtl[CVMSEGENA*]), all LL, LLD, SAA, SAAD, SC, and SCD accesses to CVMSEG cause address errors. Debug mode address error on data access exceptions can occur.
TLB Parity Error During Mapped Data Access	Cavium Specific	cnMIPS III core takes a synchronous machine-check exception when it detects a TLB parity error during a mapped data access. Software can distinguish TLB parity errors from other machine checks by Status[TS] or CacheErr (Dcache) [tlbperr]. To recover from a TLB parity error, software should completely reinitialize the TLB, including any wired entries, and flush the L1 caches. See also the Machine Check exception.
TLB/XTLB Refill on Data Access	Implemented	Debug mode TLB/XTLB Refill on Data Access exceptions can occur.
TLB Invalid on Data Access	Implemented	Debug mode TLB Invalid on Data Access exceptions can occur.
TLB Modified on Data Access	Implemented	Debug mode TLB Modified on Data Access exceptions can occur. TLB Read-Inhibit exceptions are lower priority than TLB Modified exceptions for Cavium-specific Memory Atomic instructions.
TLB Read-Inhibit on Data Access	Implemented	Debug mode TLB Read-Inhibit on data access exceptions can occur. TLB Read-Inhibit exceptions are lower priority than TLB Modified exceptions for Cavium-specific Memory Atomic instructions.

Cavium Confidential For Boyen Yang ALU 9/27/2014

Table 2–61 cnMIPS III Core Exceptions (Continued)

Exception	Implemented/ Not Implemented	Comments
Cache Error on Data Access	Implemented	The cnMIPS III core automatically corrects the error by flushing the Dcache, so the cache error handling software need not attempt to correct the error. The Cache error register (register = 27, select = 1) contains diagnostic information. Debug mode Cache Error on Data Access exceptions can occur.
Bus Error on Data Access	Implemented	Data access bus errors only occur on a time-out of an outstanding Dcache fill or a write-buffer parity error. These errors are generally non-recoverable. The Cache error register (register = 27, select = 1) contains diagnostic information. Debug mode Bus error on Data Access exceptions can occur.
Precise Debug Data Value Compare Break	Not implemented	Debug Data Value Compare breakpoints are imprecise on the core.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

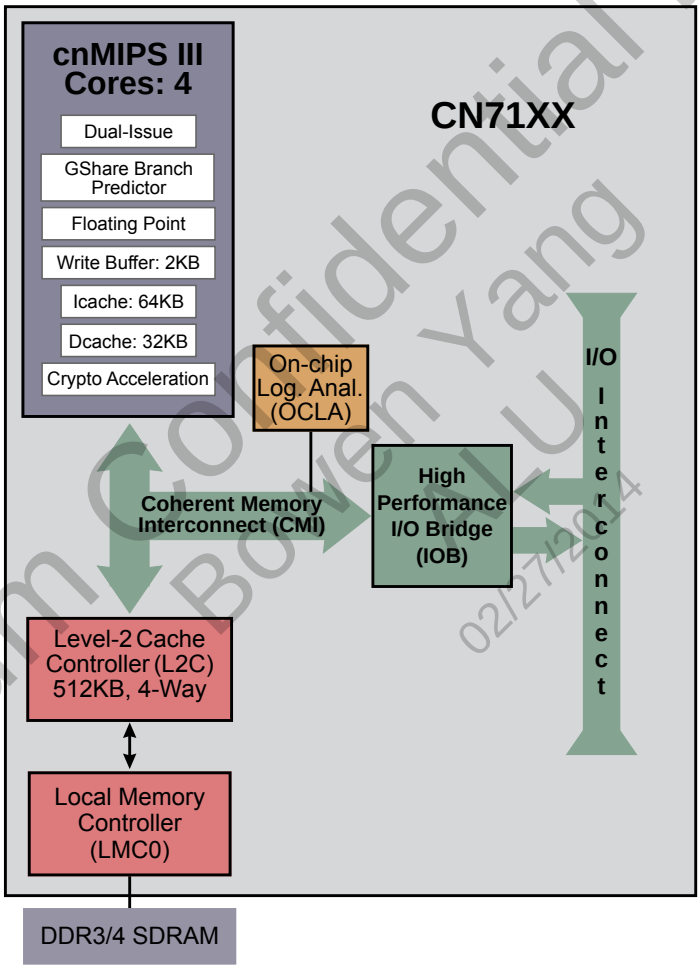
Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Section 1: OCTEON III Cache, System Memory, and Buses

This section contains chapters that describe the core elements of the CN71XX architecture that interface with the cnMIPS III cores (refer to Figure S1-1). The chapters in this section are the following:

- Coherent Memory Interconnect (CMI)/Level-Two Cache (L2C)
- Local-Memory Controller (LMC)
- I/O Interconnect (IOI)/High-Performance I/O Bridge (IOB)/Fetch and Add Unit (FAU)
- On-chip Logic Analyzer unit (OCLA)

Figure S1-1 CN71XX Cache, Memory, and Buses



Cavium Confidential For Bowen Yang 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 3

Coherent Memory Interconnect and Level 2 Cache Controller

This chapter describes the following topics:

- Coherent Memory Interconnect (CMI)

  Section 3.1 discusses the coherent memory interconnect (CMI) and contains the following subjects:

  - CMI Buses
  - CMI Description
  - CMI Memory Coherence Support
  - CMI Transactions

- Level-2 Cache Controller (L2C)

  Section 3.2 discusses the Level-2 cache controller (L2C) and contains the following subjects:

  - L2 Cache and Data Store
  - L2C Memory Coherence
  - L2 Cache Indexing (Set Selection)
  - L2 Cache Replacement and Way Partitioning
  - Memory Input Queue QOS
  - COMMIT and FILL Bus Arbitration
  - Core COP0 Register Access
  - Main Memory L2/DRAM Physical Address Space
  - L2C+LMC Internal L2/DRAM Address Space
  - L2 Cache-Block Locking
  - L2 Cache Block Flush and Unlocking
  - L2C ECC Codes

- Enumerations
- L2C Registers

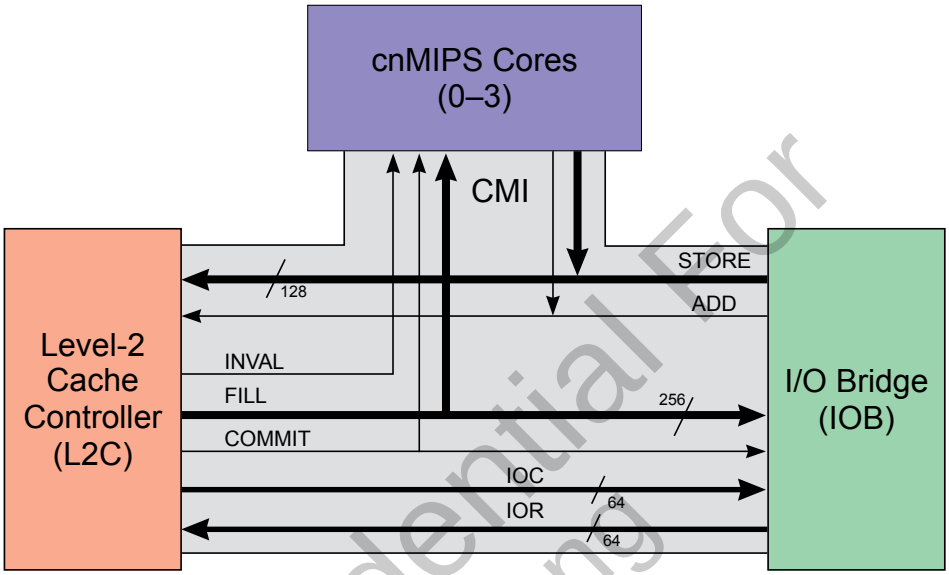
Cavium Confidential For Bowen Yang ALU 02/27/2014

3.1 Coherent Memory Interconnect (CMI)

3.1.1 CMI Overview

The coherent memory interconnect (CMI), shown in Figure 3–1, is the communication channel for all memory and I/O transactions between the four cores, the high-performance I/O bridge (IOB), and the level-2 cache controller (L2C). It runs at the core-clock frequency.

Figure 3–1 Coherent Memory Interconnect Diagram



3.1.2 CMI Buses

The CMI is comprised of seven buses: the address/command bus (ADD), the store data bus (STORE), the commit/response control bus (COMMIT), the fill data bus (FILL), the invalidate bus (INVAL), the I/O command/data bus (IOC), and the I/O response/data bus (IOR).

- The ADD bus carries address and control information that initiate CMI transactions.
- The STORE bus carries the store data required by the ADD bus transfer.
- The COMMIT bus carries required response control information.
- The FILL bus carries required response data. It can also carry invalidate information.
- The INVAL bus carries addresses from the L2C to the CPU core, causing the corresponding cache block to be invalidated from the CPU L1 data cache.
- The IOC bus carries I/O transaction request address+data from the cores to the IOB. L2C constructs the IOC transfer from ADD/STORE bus transfers.
- The IOR bus carries I/O transaction response data from the IOB back to the cores. L2C completes the I/O transaction by COMMIT/FILL bus transfers that contain the data.

Cavium Confidential For Bowen Yang ALU 08/27/2014

3.1.3 CMI Description

The CMI is a split-transaction highly pipelined interconnect. All of the ADD, STORE, COMMIT, FILL, INVAL, IOC, and IOR buses are decoupled by large queues. This decoupling allows for variable timing between the different operations required to complete different CMI transactions. Consequently, CN71XX schedules the individual buses at maximum efficiency for highest bandwidth CMI operation. (Some COMMIT bus operations are coupled, at fixed timing, to later FILL bus cycles.)

NOTE: CMI transactions transfer no more than 128 bytes of data at a time. This is the cache-block size.

3.1.4 CMI Memory Coherence Support

The CMI contains write-invalidate coherence support. The core-data caches are write-through. L2C initiates CMI invalidations to core data caches when other CMI sources update cached blocks. The cores contain large write buffers to minimize the number of CMI write operations, and support a weakly-consistent memory model to maximize performance. The CMI commit information allows weakly-consistent bus sources (like the cores) to order operations, as necessary.

Memory requests to coherent memory space are directed to the L2C. Memory requests can be initiated by either a core or the IOB. The IOB initiates memory requests on behalf of I/O agents located on the other side of the IOB. Memory responses are returned by the L2C to the source of the original request, either the cores or IOB.

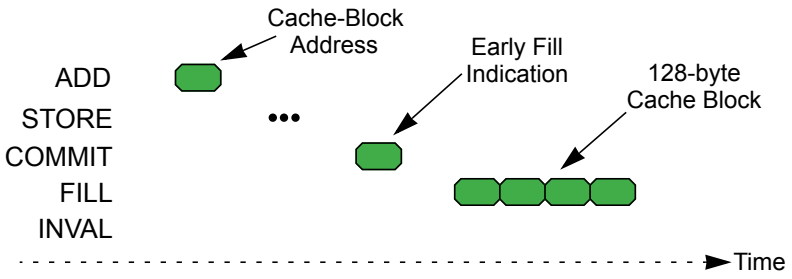
3.1.5 Coherent Memory Interconnect Transaction Examples

CMI Transaction Example

FILL Transaction

Figure 3–2 shows the steps of a complete fill transaction, a cache-block-load operation, on the CMI. A CMI transaction that either puts a cache line into either the L1 instruction or data caches after a cache miss is a fill. A CMI transaction that returns data to IOB is also a fill. (The LDD, PSL1, LDI, LDT, and LDE transactions in Table 3–1 are fills.) The first step (not shown in the figure) for a fill transaction, as with all CMI transactions, is that the core or IOB arbitrates for the ADD bus so it can initiate the transaction. This arbitration is necessary to guarantee that only one of the cores/IOB drives the ADD bus on a given cycle. The ADD bus arbitration is fair (round robin). Once the core or IOB wins arbitration, it puts control information indicating that the transaction is a fill, together with the address of the cache block, onto the ADD bus in a single cycle.

Figure 3–2 Fill Transaction



The L2C receives the ADD bus information, possibly queues it, and eventually services the transaction. In this case, servicing consists of consulting the L2 cache and/or DRAM to obtain a copy of the cache block. As Figure 3–2 indicates, once the

Cavium Confidential For Boren Yang 07/26/2016

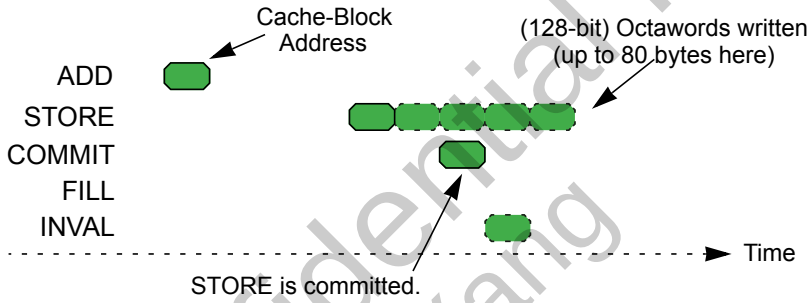
L2C has a copy of the block, it sends an early fill indication on the COMMIT bus. Starting five cycles later, the L2C then transfers the 128-byte cache block on the FILL bus in four consecutive cycles to complete the CMI transaction. The number of cycles required to complete the fill transaction varies depending on a number of factors, including whether the L2C found the block in the L2 cache or in memory.

CMI Transaction Example

Store Transaction

Figure 3–3 shows a store transaction on the CMI. Store transactions occur as stores exit the cores/IOB and are seen by other cores and I/O. The ADD bus cycle indicates a store transaction, and contains the address of the store as well as the number of 128-bit octaword transfers required on the STORE bus. The L2C schedules the STORE bus cycles needed to complete the transaction later, once the STORE bus is available and buffer space is available to receive the data. The cores/IOB drive the store data of the transaction onto the STORE bus during their scheduled cycles. This example is a case where five STORE bus cycles are required for the complete store. The number of STORE bus cycles may range from as small as one, up to a possible eight to transfer an entire 128-byte cache block. Each STORE bus cycle transfers between one and 16 bytes.

Figure 3–3 Store Transaction



The L2C buffers the ADD and STORE bus information and services the write operation. L2C places a commit operation on the COMMIT bus. If necessary, L2C also sends out an invalidate operation on the INVAL bus. The commit indicates that the store is visible to all bus users at this time, and may be sent long before the actual store operation is completely retired. For example, L2C can send the commit operation for a store even though the store is not yet deposited in the cache/DRAM, provided that any subsequent CMI transaction will see the updated store value. The commit operation for a store cannot be sent long before the invalidate operation on the INVAL bus, though, as the invalidates must appear to be complete for the store operation to be committed. If present, the invalidation operation might cause a data-cache-block invalidate in one or more CPU cores.

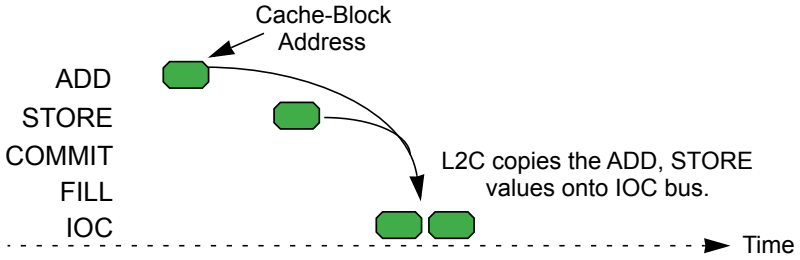
CMI Transaction Example

Store Transaction to IOB

Figure 3–4 shows a store-to-IOB transaction. Core store instructions to I/O addresses generate stores to IOB. In these store to IOB transactions, L2C buffers the ADD and STORE bus values and forwards them onto the IOC bus. IOB latches the IOC bus transfer and completes the transaction.

Cavium Confidential For Baoen Yang 12-27-2014

Figure 3–4 Store to IOB Transaction



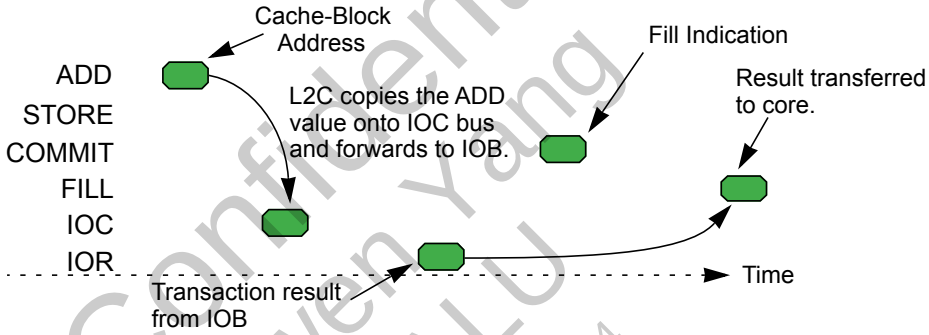
CMI Transaction Example

Load Transaction to IOB

Figure 3–5 shows a load-to-IOB transaction. Core load instructions to I/O addresses generate loads to IOB. The complete transaction involves four distinct phases:

- The first phase is an ADD bus transfer that initiates the transaction.
- The second phase is an IOC bus transfer where the L2C forwards the ADD bus information to IOB.
- The third phase is an IOR bus transfer from IOB that contains the (up to 64-bits of) transaction result for the load.
- The fourth (final) phase is a COMMIT and FILL bus pair including the result forwarded by L2C to the requesting core.

Figure 3–5 Load to IOB Transaction



3.1.6 CMI Transactions

Table 3–1 shows the CMI's transactions.

Table 3–1 Coherent Memory Interconnect Transactions

Transaction	Description
Data Cache Fill (LDD)	Fill the cache block from L2/DRAM into the L1 data cache of the requesting core. (A later store from another core or IOB will invalidate the block in the L1 data cache.) Put the block into the L2 cache if it wasn't already present. Only cores initiate these transactions.
Data Cache Fill-Through (PSL1)	Fill the cache block from L2/DRAM into the L1 data cache of the requesting core. (A later store from another core or IOB will invalidate the block in the L1 data cache.) Do not put the block into the L2 cache. Only cores initiate these transactions (and only via PREF/PREFX 4 instructions – refer to Section A.3).

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 3–1 Coherent Memory Interconnect Transactions (Continued)

Transaction	Description
Instruction Cache / I/O Fill (LDI)	Fill the cache block from L2/DRAM into the L1 instruction cache or IOB. Put the cache block into the L2 cache if it wasn't already present. Both cores and IOB initiate these transactions. The cores only fetch instructions via these transactions.
Instruction Cache / I/O Fill-Through (LDT)	Fill the cache block from L2/DRAM into the L1 instruction cache or to IOB. Do not put the cache block into the L2 cache. Both cores and IOB initiate these transactions. The cores only fetch instructions via these transactions.
Data Cache Fill Exclusive (LDE)	When initiated by a core (via PREF/PREFX 7 only, refer to Section A.3.1), LDE is functionally equivalent to LDD. When initiated by IOB, LDE is functionally equivalent to LDI.
Store-Partial (STP)	Store to some of the bytes in the cache block. Do not modify the bytes in the cache block that are not stored. Invalidate all L1 data cache copies of the block, except for the L1 data cache of the initiator (if the initiator was a core). Put the cache block into the L2 cache if it wasn't already present. Both cores and IOB initiate these transactions.
Store-Full (STF, STFIL1)	Store to all bytes in the cache block. Zero the bytes not transferred on the STORE bus (or masked off). Invalidate all L1 data cache copies of the block, except for the L1 data cache of the initiator (if the initiator was a core). The core that initiates a STFIL1 self-invalidates the block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Both cores and IOB initiate STF transactions. Only cores initiate STFIL1 transactions (and only via ZCB instructions – refer to Section A.3.12).
Store-Through (STT, STTIL1)	Store to all bytes in the cache block. Zero the bytes not transferred on the STORE bus (or masked off). Invalidate all L1 data cache copies of the block, except for the L1 data cache of the initiator (if the initiator was a core). The core that initiates a STTIL1 self-invalidates the block from its L1 data cache, if necessary. Do not put the cache block into the L2 cache. Both cores and IOB initiate STT transactions. (The cores only initiate STTs via PREF/PREFX 31 instructions – refer to Section A.3.) Only cores initiate STTIL1 transactions (and only via ZCBT instructions – refer to Section A.3.12).
Store-Conditional (STC)	If the current value of the memory location matches the register result value of the LL (LLD) instruction, store to the 32-bits (64-bits) and invalidate all other L1 data cache copies of the block in all other cores. If the current value of the memory location does not match the LL/LLD instruction value, fail the transaction, neither storing nor invalidating. On success, put the cache block into the L2 cache if it wasn't already present. Only cores initiate these transactions (and only via SC/SCD instructions that do not terminate a wide atomic write).
L2 Cache Index Writeback Invalidate (WBIL2I)	If the index-selected L2 cache block is dirty (i.e. if VALID and DIRTY bits are set), write it back to the DRAM. (The ADD bus identifies the L2 cache block via an index and way, rather than an address, for this transaction.) Invalidate the selected L2 cache block (i.e. clear the VALID, USED, and LOCK bits). Only cores initiate these transactions (and only via CACHE 3 instructions – refer to Section A.4).
L2 Cache Index Writeback (WBL2I)	If the index-selected cache block is dirty (i.e. if VALID and DIRTY bits are set), clear its DIRTY and USE bit after first writing the block to DRAM. (Clearing the USE bit makes it more likely that the cache block will be replaced in the future.) (The ADD bus identifies the cache block via an index and way, rather than an address, for this transaction, and this is an ADD-bus-only transaction on the CMI.) Do not clear the USE bit if the block is locked. This is an ADD-bus-only transaction on the CMI. Only cores initiate these transactions (and only via CACHE 15 instructions – refer to Section A.4).

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 3–1 Coherent Memory Interconnect Transactions (Continued)

Transaction	Description
L2 Cache Index Load Tag (LTGL2I)	Read the tag and state for the selected L2 cache block into L2C_TAD0_TAG. (The ADD bus identifies the cache block via an index and way, rather than an address, for this transaction.) Only cores initiate these transactions (and only via CACHE 7 instructions – refer to Section A.4).
L2 Cache Index Store Tag (STGL2I)	Write the tag and state for the index-selected L2 cache block to the current L2C_TAD0_TAG value. (The ADD bus identifies the cache block via an index and way, rather than an address, for this transaction.) Only cores initiate these transactions (and only via CACHE 11 instructions – refer to Section A.4).
L2 Cache Hit Invalidate (INVL2)	If the cache block is present in the L2 cache, invalidate it (i.e. clear the VALID, USED, and LOCK bits). Do nothing if the cache block is not present in the L2 cache. If dirty data for the location was present in the L2 cache, it is lost. The current value of the cache block is unpredictable after the INVL2 transaction until the block is later stored to, as stale copies of the block may be present in the L1 data caches. Only cores initiate these transactions (and only via CACHE 19 instructions – refer to Section A.4).
L2 Cache Hit Writeback Invalidate (WBIL2)	If the cache block is present in the L2 cache, invalidate it (i.e. clear the VALID, USED, and LOCK bits) after first writing the block to DRAM if it was dirty (i.e. if VALID and DIRTY bits were set). Do nothing if the cache block is not present in the L2 cache. Only cores initiate these transactions (and only via CACHE 23 instructions – refer to Section A.4).
L2 Cache Fetch and Lock (LCKL2)	Set the LOCK (and USE) bit for the block. Put the cache block into the L2 cache if it wasn't already present and there is an L2 cache way available. LCKL2 is a NOP if it misses in the L2 cache when there are no ways available. Only cores initiate these transactions (and only via CACHE 31 instructions – refer to Section A.4).
Don't-Write-Back (DWB)	If the cache block is present in the L2 cache, clear the DIRTY bit. Do nothing if the cache block is not present in the L2 cache. If dirty data for the location was present in the L2 cache, it may be lost. The current value of the cache block is unpredictable after the DWB transaction until the block is later stored to, as stale copies of the block may be present in the L1 data caches. Both cores and IOB initiate DWB transactions. (The cores only initiate DWBs via PREF/PREFX 29 instructions – refer to Section A.3.)
Prefetch Into L2 (PL2)	Put the cache block into the L2 cache if it wasn't already present. This is an ADD-bus-only transaction on the CMI. Only cores initiate these transactions (and only via PREF/PREFX 5/28 instructions – refer to Section A.3).
Evict Soon from L2 (RPL2)	If the cache block is present in the L2 cache (and not locked), clear its USE bit. (Clearing the USE bit makes it more likely that the cache block will be replaced in the future.) Do nothing if the cache block is (locked or) not present in the L2 cache. This is an ADD-bus-only transaction on the CMI. Only cores initiate these transactions (and only via PREF/PREFX 27 instructions – refer to Section A.3).
L2 Cache Nudge / Hit Writeback (WBL2)	If the cache block is present in the L2 cache, clear its DIRTY and USE bits after first writing the block to DRAM if it was dirty (i.e. if VALID and DIRTY bits were set). (Clearing the USE bit makes it more likely that the cache block will be replaced in the future.) Do nothing if the cache block is not present in the L2 cache. Do not clear the USE bit if the block is locked. This is an ADD-bus-only transaction on the CMI. Only cores initiate these transactions (and only via PREF/PREFX 25 and CACHE 27 instructions – refer to Section A.3 and Section A.4).

Cavium Confidential For Bowen Yang ALD 6/27/2014

Table 3–1 Coherent Memory Interconnect Transactions (Continued)

Transaction	Description
Store-Atomic Add (SAA64, SAA32)	Read the current value of a naturally-aligned 32-bit or 64-bit word in L2/DRAM, and atomically add a value to it and update L2/DRAM with the result. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate these transactions (and only via SAA/SAAD instructions – refer to Section A.3.2).
L2 Cache Atomic Fetch and Add (FAA32, FAA64)	Return the current value of a naturally-aligned 32-bit or 64-bit word in L2/DRAM, and atomically add a value to it and update L2/DRAM with the result. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate these transactions (and only via LAA/LAAD instructions – refer to Section A.3.6).
L2 Cache Atomic Fetch and Increment (INCR64, INCR32)	Return the current value of a naturally-aligned 32-bit or 64-bit word in L2/DRAM, and atomically add one to it and update L2/DRAM with the result. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate INCR64 and INCR32 transactions (only via LAI/LAID instructions – refer to Section A.3.8).
L2 Cache Atomic Fetch and Decrement (DECR64, DECR32)	Return the current value of a naturally-aligned 8-bit, 16-bit, 32-bit, or 64-bit word in L2/DRAM, and atomically subtract one to it and update L2/DRAM with the result. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate DECR64 and DECR32 transactions (only via LAD/LADD instructions -– refer to Section A.3.9).
L2 Cache Atomic Swap (FAS64, FAS32)	Return the current value of a naturally-aligned 32-bit or 64-bit word in L2/DRAM, and atomically store to it. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate these transactions (and only via LAW/LAWD instructions – refer to Section A.3.7).
L2 Cache Atomic Fetch and Set (SET64, SET32)	Return the current value of a naturally-aligned 8-bit, 16-bit, 32-bit, or 64-bit word in L2/DRAM, and atomically set all bits in it. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate SET64 and SET32 transactions (only via LAS/LASD instructions – refer to Section A.3.10).
L2 Cache Atomic Fetch and Clear (CLR64, CLR32)	Return the current value of a naturally-aligned 8-bit, 16-bit, 32-bit, or 64-bit word in L2/DRAM, and atomically clear all bits in it. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate CLR64 and CLR32 transactions (only via LAC/LACD instructions – refer to Section A.3.11).

Cavium Confidential For Bowen Yang ALi 2017/2/7 09:14

Table 3–1 Coherent Memory Interconnect Transactions (Continued)

Transaction	Description
Instruction Cache / I/O Fill (LDI)	Fill the cache block from L2/DRAM into the L1 instruction cache or IOB. Put the cache block into the L2 cache if it wasn't already present. Both cores and IOB initiate these transactions. The cores only fetch instructions via these transactions.
Instruction Cache / I/O Fill-Through (LDT)	Fill the cache block from L2/DRAM into the L1 instruction cache or to IOB. Do not put the cache block into the L2 cache. Both cores and IOB initiate these transactions. The cores only fetch instructions via these transactions.
Data Cache Fill Exclusive (LDE)	When initiated by a core (via PREF/PREFX 7 only, refer to Section A.3.1), LDE is functionally equivalent to LDD. When initiated by IOB, LDE is functionally equivalent to LDI.
Store-Partial (STP)	Store to some of the bytes in the cache block. Do not modify the bytes in the cache block that are not stored. Invalidate all L1 data cache copies of the block, except for the L1 data cache of the initiator (if the initiator was a core). Put the cache block into the L2 cache if it wasn't already present. Both cores and IOB initiate these transactions.
Store-Full (STF, STFIL1)	Store to all bytes in the cache block. Zero the bytes not transferred on the STORE bus (or masked off). Invalidate all L1 data cache copies of the block, except for the L1 data cache of the initiator (if the initiator was a core). The core that initiates a STFIL1 self-invalidates the block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Both cores and IOB initiate STF transactions. Only cores initiate STFIL1 transactions (and only via ZCB instructions – refer to Section A.3.12).
Store-Through (STT, STTIL1)	Store to all bytes in the cache block. Zero the bytes not transferred on the STORE bus (or masked off). Invalidate all L1 data cache copies of the block, except for the L1 data cache of the initiator (if the initiator was a core). The core that initiates a STTIL1 self-invalidates the block from its L1 data cache, if necessary. Do not put the cache block into the L2 cache. Both cores and IOB initiate STT transactions. (The cores only initiate STTs via PREF/PREFX 31 instructions – refer to Section A.3.) Only cores initiate STTIL1 transactions (and only via ZCBT instructions – refer to Section A.3.12).
Store-Conditional (STC)	If the current value of the memory location matches the register result value of the LL (LLD) instruction, store to the 32-bits (64-bits) and invalidate all other L1 data cache copies of the block in all other cores. If the current value of the memory location does not match the LL/LLD instruction value, fail the transaction, neither storing nor invalidating. On success, put the cache block into the L2 cache if it wasn't already present. Only cores initiate these transactions (and only via SC/SCD instructions that do not terminate a wide atomic write).
L2 Cache Index Writeback Invalidate (WBIL2I)	If the index-selected L2 cache block is dirty (i.e. if VALID and DIRTY bits are set), write it back to the DRAM. (The ADD bus identifies the L2 cache block via an index and way, rather than an address, for this transaction.) Invalidate the selected L2 cache block (i.e. clear the VALID, USED, and LOCK bits). Only cores initiate these transactions (and only via CACHE 3 instructions – refer to Section A.4).
L2 Cache Index Writeback (WBL2I)	If the index-selected cache block is dirty (i.e. if VALID and DIRTY bits are set), clear its DIRTY and USE bit after first writing the block to DRAM. (Clearing the USE bit makes it more likely that the cache block will be replaced in the future.) (The ADD bus identifies the cache block via an index and way, rather than an address, for this transaction, and this is an ADD-bus-only transaction on the CMI.) Do not clear the USE bit if the block is locked. This is an ADD-bus-only transaction on the CMI. Only cores initiate these transactions (and only via CACHE 15 instructions – refer to Section A.4).

Cavium Confidential For Edwin Yang ALU 02/27/2014

Table 3–1 Coherent Memory Interconnect Transactions (Continued)

Transaction	Description
L2 Cache Index Load Tag (LTGL2I)	Read the tag and state for the selected L2 cache block into L2C_TAD0_TAG. (The ADD bus identifies the cache block via an index and way, rather than an address, for this transaction.) Only cores initiate these transactions (and only via CACHE 7 instructions – refer to Section A.4).
L2 Cache Index Store Tag (STGL2I)	Write the tag and state for the index-selected L2 cache block to the current L2C_TAD0_TAG value. (The ADD bus identifies the cache block via an index and way, rather than an address, for this transaction.) Only cores initiate these transactions (and only via CACHE 11 instructions – refer to Section A.4).
L2 Cache Hit Invalidate (INVL2)	If the cache block is present in the L2 cache, invalidate it (i.e. clear the VALID, USED, and LOCK bits). Do nothing if the cache block is not present in the L2 cache. If dirty data for the location was present in the L2 cache, it is lost. The current value of the cache block is unpredictable after the INVL2 transaction until the block is later stored to, as stale copies of the block may be present in the L1 data caches. Only cores initiate these transactions (and only via CACHE 19 instructions – refer to Section A.4).
L2 Cache Hit Writeback Invalidate (WBIL2)	If the cache block is present in the L2 cache, invalidate it (i.e. clear the VALID, USED, and LOCK bits) after first writing the block to DRAM if it was dirty (i.e. if VALID and DIRTY bits were set). Do nothing if the cache block is not present in the L2 cache. Only cores initiate these transactions (and only via CACHE 23 instructions – refer to Section A.4).
L2 Cache Fetch and Lock (LCKL2)	Set the LOCK (and USE) bit for the block. Put the cache block into the L2 cache if it wasn't already present and there is an L2 cache way available. LCKL2 is a NOP if it misses in the L2 cache when there are no ways available. Only cores initiate these transactions (and only via CACHE 31 instructions – refer to Section A.4).
Don't-Write-Back (DWB)	If the cache block is present in the L2 cache, clear the DIRTY bit. Do nothing if the cache block is not present in the L2 cache. If dirty data for the location was present in the L2 cache, it may be lost. The current value of the cache block is unpredictable after the DWB transaction until the block is later stored to, as stale copies of the block may be present in the L1 data caches. Both cores and IOB initiate DWB transactions. (The cores only initiate DWBs via PREF/PREFX 29 instructions – refer to Section A.3.)
Prefetch Into L2 (PL2)	Put the cache block into the L2 cache if it wasn't already present. This is an ADD-bus-only transaction on the CMI. Only cores initiate these transactions (and only via PREF/PREFX 5/28 instructions – refer to Section A.3).
Evict Soon from L2 (RPL2)	If the cache block is present in the L2 cache (and not locked), clear its USE bit. (Clearing the USE bit makes it more likely that the cache block will be replaced in the future.) Do nothing if the cache block is (locked or) not present in the L2 cache. This is an ADD-bus-only transaction on the CMI. Only cores initiate these transactions (and only via PREF/PREFX 27 instructions – refer to Section A.3).
L2 Cache Nudge / Hit Writeback (WBL2)	If the cache block is present in the L2 cache, clear its DIRTY and USE bits after first writing the block to DRAM if it was dirty (i.e. if VALID and DIRTY bits were set). (Clearing the USE bit makes it more likely that the cache block will be replaced in the future.) Do nothing if the cache block is not present in the L2 cache. Do not clear the USE bit if the block is locked. This is an ADD-bus-only transaction on the CMI. Only cores initiate these transactions (and only via PREF/PREFX 25 and CACHE 27 instructions – refer to Section A.3 and Section A.4).

Cavium Confidential For Bowen Yang ALD 3/27/2014

Table 3–1 Coherent Memory Interconnect Transactions (Continued)

Transaction	Description
Store-Atomic Add (SAA64, SAA32)	Read the current value of a naturally-aligned 32-bit or 64-bit word in L2/DRAM, and atomically add a value to it and update L2/DRAM with the result. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate these transactions (and only via SAA/SAAD instructions – refer to Section A.3.2).
L2 Cache Atomic Fetch and Add (FAA32, FAA64)	Return the current value of a naturally-aligned 32-bit or 64-bit word in L2/DRAM, and atomically add a value to it and update L2/DRAM with the result. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate these transactions (and only via LAA/LAAD instructions – refer to Section A.3.6).
L2 Cache Atomic Fetch and Increment (INCR64, INCR32)	Return the current value of a naturally-aligned 32-bit or 64-bit word in L2/DRAM, and atomically add one to it and update L2/DRAM with the result. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate INCR64 and INCR32 transactions (only via LAI/LAID instructions – refer to Section A.3.8).
L2 Cache Atomic Fetch and Decrement (DECR64, DECR32)	Return the current value of a naturally-aligned 8-bit, 16-bit, 32-bit, or 64-bit word in L2/DRAM, and atomically subtract one to it and update L2/DRAM with the result. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate DECR64 and DECR32 transactions (only via LAD/LADD instructions -– refer to Section A.3.9).
L2 Cache Atomic Swap (FAS64, FAS32)	Return the current value of a naturally-aligned 32-bit or 64-bit word in L2/DRAM, and atomically store to it. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate these transactions (and only via LAW/LAWD instructions – refer to Section A.3.7).
L2 Cache Atomic Fetch and Set (SET64, SET32)	Return the current value of a naturally-aligned 8-bit, 16-bit, 32-bit, or 64-bit word in L2/DRAM, and atomically set all bits in it. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate SET64 and SET32 transactions (only via LAS/LASD instructions – refer to Section A.3.10).
L2 Cache Atomic Fetch and Clear (CLR64, CLR32)	Return the current value of a naturally-aligned 8-bit, 16-bit, 32-bit, or 64-bit word in L2/DRAM, and atomically clear all bits in it. Invalidate all L1 data cache copies of the surrounding cache block, except for the L1 data cache of the initiator. The core that initiates the transaction self-invalidates the surrounding block from its L1 data cache, if necessary. Put the cache block into the L2 cache if it wasn't already present. Only cores initiate CLR64 and CLR32 transactions (only via LAC/LACD instructions – refer to Section A.3.11).

Cavium Confidential For Bowen Yang ALi 2017/2/7 09:14

Table 3–1 Coherent Memory Interconnect Transactions (Continued)

Transaction	Description
Store to IOB (IOBST64, IOBST32, IOBST16, IOBST8)	Forward a naturally-aligned address and corresponding 8-bit, 16-bit, 32-bit, or 64-bit word onto the IOC bus. Either service the I/O store directly in IOB or forward it onto the IOI for another OCTEON I/O device. Only cores initiate these transactions.
Load to IOB (IOBLD8, IOBLD16, IOBLD32, IOBLD64)	Forward a naturally-aligned address onto the IOC bus. Either service the I/O load directly in IOB or forward it onto the IOI and service it in another CN71XX I/O device. Return the 8-bit, 16-bit, 32-bit, or 64-bit result via an IOR single-cycle transfer. Forward this IOR response onto the COMMIT/FILL busses to complete the transaction. Only cores initiate these transactions.
IOBDMA to IOB (IOBDMA)	Forward a naturally-aligned address onto the IOC bus. Either service the I/O vector load directly in IOB or forward it onto the IOI for another CN71XX I/O device. The total IOBDMA length may be as large as 255 64-bit words. Respond with a series of $1 \le n \le 16$ word responses. Return each n-vector of 64-bit results via an IOR n-cycle transfer. Forward each IOR response onto the COMMIT/FILL buses until the transaction is complete. Only cores initiate these transactions (refer to Section 2.7).
IOBADDR	Zero-length IOBDMAs. Correct code should never generate these on CN71XX.

Cavium Confidential For Bowen Yang ALU 02/27/2014

3.2 Level-2 Cache Controller (L2C)

This section discusses the L2C and contains the following subjects:

- L2 Cache and Data Store
- L2C Memory Coherence
- L2 Cache Indexing (Set Selection)
- L2 Cache Replacement and Way Partitioning
- Memory Input Queue QOS
- COMMIT and FILL Bus Arbitration
- Core COP0 Register Access
- Main Memory L2/DRAM Physical Address Space
- L2C+LMC Internal L2/DRAM Address Space
- L2 Cache-Block Locking
- L2 Cache-Block Locking
- L2 Cache Block Flush and Unlocking
- L2C ECC Codes

3.2.1 L2 Cache and Data Store

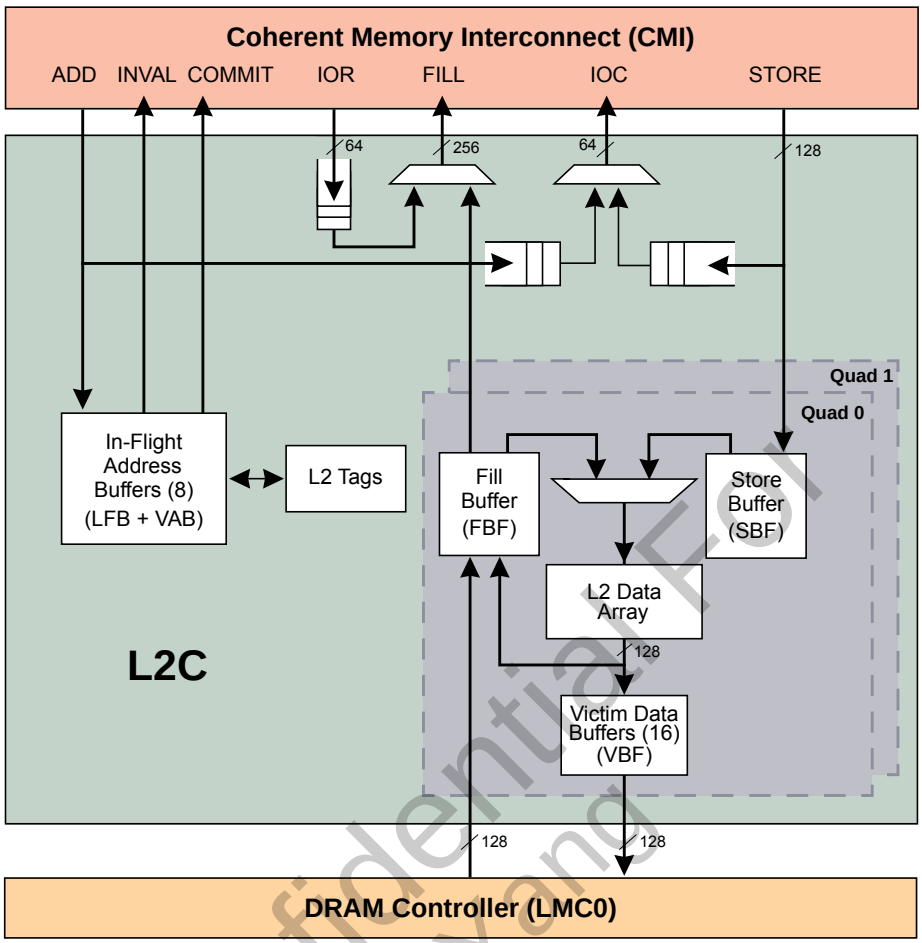
The L2C block diagram is shown in Figure 3–6. The L2C contains CN71XX's shared on-chip cache. The L2 cache has the following specifications:

- 512KB
- four-way set-associative with a 128-byte cache block
- write-back
- both the on-chip data and tags are protected by SECDED ECC.

The L2 cache is shared by all of the cores and the I/O components on CN71XX, though it can be bypassed using particular CMI transactions and can also be partitioned.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 3–6 L2C Block Diagram



3.2.2 L2C Memory Coherence

L2C maintains CN71XX's memory-reference coherence.

- It returns the latest copy of the block for every fill request, whether the block is in the L2 cache, in DRAM, or in-flight somewhere.
- It also sends an invalidate to the CPU cores for store requests to cache blocks that are present in a data cache when the request was created by another core or an I/O component, via the IOB.

L2C also services CMI I/O transactions that are necessary to transfer commands and/or data between the cores and the IOB.

Figure 3–6 is a block diagram of the L2C. The figure shows the CMI interface on the top and the DRAM controller interface on the bottom. The CMI interface is 640 bits wide, the LMC DRAM interface is 256 bits wide, and the internal L2 cache data interfaces are 256 bits wide in each quad (a total of 512 bits including all quads). Section 3.1 describes the CMI. Chapter 4 describes the LMC DRAM controller. All data transfers between the L2C and the LMC move full 128-byte cache blocks.

L2C has eight L2/DRAM input queues that receive ADD bus memory transactions: one queue for each of eight quality-of-service (QOS) levels. Separate QOS levels can improve L2C response to higher-priority transactions. Section 3.2.5 describes this in more detail.

Cavium Confidential FCI
Bowen Yang
ALU
02/27/2014

L2C also contains separate queues to facilitate Core ↔ IOB I/O transactions:

- One set to forward core ADD/STORE bus I/O information to IOC, and
- Another queue to forward IOB IOR I/O responses to COMMIT/FILL

L2C holds and processes up to eight simultaneous L2/DRAM transactions in its in-flight address buffer (LFB).

For L2/DRAM fill transactions, L2C returns data from either the L2 cache or memory via the fill buffer (FBF) associated with the LFB entry. For L2/DRAM store transactions, L2C first deposits STORE bus data into a store buffer (SBF) entry associated with the LFB entry, then either updates the cache or writes a full cache block store straight to DRAM. Except for stores to all bytes in a cache block, all L2/DRAM transactions that miss in the L2 cache require a DRAM fill. Partial cache block stores need the DRAM fill to obtain the bytes that are not stored. L2C puts DRAM fill data into the FBF, then writes it to the L2 cache, if necessary, and forwards it onto the FILL bus, if necessary.

3.2.3 L2 Cache Indexing (Set Selection)

There are 1024 sets in the L2 cache. Each set contains four 128-byte cache blocks, one per available way. Every L2/DRAM reference must select a set.

L2C implements two set-indexing algorithms, an unaliased algorithm (L2C_CTL[DISIDXALIAS] = 1), and an aliased one (L2C_CTL[DISIDXALIAS] = 0). The aliased algorithm is recommended for most CN71XX applications.

- The unaliased algorithm is the typical set-selection algorithm:

  index<9:0> = address<16:7>

  where address is the physical byte address of the L2/DRAM location.

- The aliased set-selection algorithm additionally bit-wise exclusive-ORs upper address bits into the index:

  index<2:0> = address<9:7> ⊕ address<19:17> ⊕ address<14:12>
  index<9:3> = address<16:10> ⊕ address<26:20>

Both the unaliased and aliased algorithms spread contiguous cache blocks across sequential sets. The aliased algorithm also spreads cache blocks with different upper address bits across different sets. This is advantageous when memory addressing is not strictly contiguous or not random.

For example, if packet buffers are 2KB and always naturally-aligned, but only the first 256 bytes of every packet buffer is typically used, the unaliased algorithm can only ever use two sets out of every sixteen, but the aliased algorithm evenly distributes the blocks across all sets, provided the naturally aligned 2KB buffers cover many 128KB-aligned regions. This can have a very large performance effect.

NOTE: L2C_CTL[DISIDXALIAS] must change value only when the L2 cache is known to be completely empty, such as at boot time.

Cavium Confidential For Bowen Yang ALU 02-27-2014

3.2.4 L2 Cache Replacement and Way Partitioning

L2C implements its replacement policy using one USED bit per L2 cache block. There are four USED bits per each of the 1024 sets, one per way. All sixteen ways are normally "available," but way partitioning, described below, can remove ways from consideration. Locked blocks, described in Section 3.2.10, also remove ways from replacement consideration.

When L2C references a block, it sets the USED bit for the block to 1. When the USED bits for all available ways in a set are all 1s, L2C clears to 0 the USED bits for all the available ways in the set, except for the last block referenced. When L2C replaces a cache block, it picks the first available way that has a USED bit of 0 and does not hold a locked block. If the USED bits of all available ways in a set are all 1s on a replacement operation, L2C replaces the first available way that does not hold a locked block.

Way partitioning can prevent specific cores or I/O devices from polluting the L2 cache. Four-bit MASK values configure the way partitioning. There are a total of five 4-bit [MASK] fields for the different sources of L2/DRAM transactions on the CMI:

- Four for the cores: one for each of the cores (L2C_WPAR_PP(0..3)[MASK])
- One for all the IOB-initiated transactions (L2C_WPAR_IOB0[MASK])

When a bit is set in one of the mask fields, a memory command issued from the corresponding core or IOB cannot select that way for replacement. It is legal to set all mask bits to 1, preventing the corresponding core or IOB from ever replacing an L2 block. Note that there can be significant performance effects from this, however, as all partial-block-store transactions will require full cache block DRAM read followed by write in this case.

NOTE: Way partitioning does not directly affect cache coherence and does not directly affect cache-hit processing in any way. Way partitioning only directly affects cache replacement.

3.2.5 Memory Input Queue QOS

As Section 3.2.2 mentions and as is shown in Figure 3–6, new L2/DRAM requests arrive at L2C via the CMI ADD bus and enter one of eight memory input queues – one queue for each of the eight QOS levels. L2C does not process these new transactions while they reside in one of the input queues, but it later makes them one of the (up to) eight in-flight transactions/addresses that it can simultaneously process. All of these L2C in-flight transactions remain in-flight until they are completed, and completion may require DRAM accesses. Consequently, entry into the L2C in-flight buffer is a critical scheduling point.

L2C_QOS_PP(0..3) and L2C_QOS_IOB0 programming select one of the eight QOS levels for each core and for two classes (normal and Don't-Write-Back) of IOB L2C requests.

When L2C_CTL[XMC_ARB_MODE] = 1, L2C requests that enter the QOS = 0 queue do not participate in any weighted-round-robin scheduling and are statically higher-priority than other L2C requests. This special mode can be appropriate in many circumstances, such as to ensure that critical I/O — like packet input and output — are not blocked.

Cavium Confidential For Bowen Yang ALU 10/7/2014

The remainder of this section assumes L2C_CTL[XMC_ARB_MODE] = 0, where all L2C requests participate in the weighted-round-robin algorithm.

The L2C memory QOS scheduling works in rounds. There are 256 rounds. Each of the eight QOS levels can participate in various combinations of the 256 rounds. Before completing a round, L2C moves one request from every QOS queue that participates in a round and currently has a request available.

L2C_QOS_WGT[WGT0, WGT1, WGT2, WGT3, WGT4, WGT5, WGT6, WGT7] determines the weight of each QOS queue. The weight determines the rounds that a given QOS level participates in. A queue with higher weight is higher priority because it will participate in more rounds. A queue with lower weight is lower priority.

New L2C requests can bypass the eight QOS levels when all input queues are empty and the in-flight buffer is not full. The round-based weighted-round-robin scheduling (and the L2C_CTL[XMC_ARB_MODE] mode) is most used after the in-flight buffer is full.

3.2.6 COMMIT and FILL Bus Arbitration

The COMMIT/FILL bus arbiter can operate in one of the following modes:

- round-robin mode (L2C_CTL[RSP_ARB_MODE]=0)
- static-priority mode (L2C_CTL[RSP_ARB_MODE]=1).

Round-robin is more fair, but the fixed priority mode prioritizes IOR above other FILL bus usages, and this may be advantageous to minimize core-initiated I/O latencies.

3.2.7 Core COP0 Register Access

The L2C_COP0_ADR and L2C_COP0_DAT registers allow some Coprocessor 0 register values, as well as additional debug information like the current program counter, from inside each CPU core to be observed/accessed. For more information on the COP0 registers, see Section 2.15.

This L2C_COP0_ADR/L2C_COP0_DAT access is useful for many purposes, including:

- Debug observability, and
- Write access to the COP0 PowThrottle register. This makes it easy for centralized power-management software to control the power on all CPU cores.

Cavium Confidential For Bowen Yang ALU 02/27/2014

3.2.8 Main Memory L2/DRAM Physical Address Space

Section 2.6 describes the full 49-bit CN71XX physical address space. Most of the lower addresses in this 49-bit space are L2/DRAM addresses. The cores and all other CN71XX units address L2/DRAM with these lower addresses.

The following architected address ranges provide space for up to 128 GB of L2/DRAM physical addresses:

DR0: 0x0 0000 0000 0000 to 0x0 0000 0FFF FFFF
= lower 256MB of DDR3 SDRAM

DR1: 0x0 0000 2000 0000 to 0x0 0080 0FFF FFFF
= upper 511.75GB of DDR3 SDRAM

- With 256MB of DDR3 SDRAM, only DR0 is present.
- With more than 256MB of DDR3 SDRAM, DR0 is present in entirety, and the lower portion of DR1 is present, according to the amount of DRAM attached.

CN71XX can attach at most 16GB of SDRAM, so it can never entirely fill DR1.

Some CN71XX units provide only 36-bit L2/DRAM addressing.

3.2.9 L2C+LMC Internal L2/DRAM Address Space

Before much processing, L2C converts the 40-bit L2/DRAM addresses described in Section 3.2.8 into a 39-bit address in the address space called the L2C+LMC internal L2/DRAM address space. This address space is contiguous and cannot be fully utilized by CN71XX. It includes DR0 locations at an unchanged address. It also includes tDR1 after hole-removal: a DR1 location at CN71XX physical address *X* exists at address (*X* – 256 MB) in this L2C+LMC internal L2/DRAM address space.

L2/DRAM addresses between DR0 and DR1 are called HOLE addresses. HOLE addresses do not exist in the L2C+LMC internal address space. (A HOLE address Y aliases to address Y in L2C+LMC internal DRAM address space.) L2C+LMC internal addresses beyond the limit selected by L2C_BIG_CTL[MAXDRAM] are called BIG addresses in CN71XX.

L2C can flag HOLE and BIG accesses L2C_TAD(0..0)_INT[HOLERD, HOLEWR, BIGRD, BIGWR] and CIU_CIB_L2C_RAW[TADX_INT_BIGRD, TADX_INT_BIGWR, TADX_INT_HOLERD, TADX_INT_HOLEWR]. L2C prevents write operations to HOLE and BIG addresses from modifying any existing L2/DRAM locations.

3.2.10 L2 Cache-Block Locking

L2C can lock individual cache blocks into the L2 cache. A locked block is not replaced (until it is later flushed from the cache – See Section 3.2.11 – or until the chip is reset), so fast access to the block is guaranteed when it is locked.

L2C locks a block in the cache when it services a LCKL2 transaction. Table 3–1 describes LCKL2 CMI transactions. L2C normally receives LCKL2 transactions from cores via L2 Cache Fetch and Lock instructions (CACHE 31 – refer to Section A.4) and the CMI, and can also receive LCKL2 transactions via the L2C_XMC_CMD CSR.

Cavium Confidential For Bowen Yang ALD 02/11/2014

If L2C does not find the block in the cache while servicing the LCKL2, the locked block resides in the way selected by the normal replacement algorithm for the source core after the transaction. If L2C finds the block already in the cache while servicing the LCKL2, the locked block resides in the way that previously held the block after L2C completes the LCKL2. It can be difficult to predict the exact way that a locked block will be placed in, though the selection can be influenced with way partitioning (refer to Section 3.2.4) while locking.

Software can confirm the lock and determine the L2 cache way that L2C placed the block in via the L2 Cache Index Load Tag instructions (CACHE 7 – refer to Section A.4).

Section 3.2.11 describes how to unlock L2 cache blocks.

3.2.11 L2 Cache Block Flush and Unlocking

It may be necessary to flush blocks from the L2 cache. Two major ways are address-based, and index-based. The address-based flush instructions refer to specific L2/DRAM blocks by address, do nothing when the block is not currently present in the L2 cache, and are most useful to flush small address ranges. The index-based flush instructions refer to specific L2 cache locations by index and way, and is most useful to flush the entire L2 cache. It is possible for the L2 cache location selected by the indexed-based flush instructions to hold any L2/DRAM block from the index.

Address-based flush mechanisms:

- A NUDGE instruction (i.e. PREF/PREFX 25 – See Section A.3) or a L2 Cache Hit Writeback instruction (i.e. CACHE 27 – See Section A.4) creates a WBL2 CMI transaction that causes L2C to write a matching block back to L2/DRAM if it was dirty, and replace it soon (if it isn't locked).
- An L2 Cache Hit Writeback Invalidate instruction (i.e. CACHE 23 – See Section A.4) creates a WBIL2 CMI transaction that causes L2C to write a matching block back to L2/DRAM if it was dirty, and remove it from the cache (unlocking it, if necessary).
- An L2 Cache Hit Invalidate instruction (i.e. CACHE 19 – See Section A.4) creates a INVL2 CMI transaction that causes L2C to invalidate a matching block without any write-back (unlocking it, if necessary), which makes the current value of the block unpredictable if it was dirty in the L2 cache.

Index-based flush mechanisms:

- An L2 Cache Index Writeback instruction (i.e. CACHE 15 - See Section A.4) creates a WBL2I CMI transaction that causes L2C to write a block back to L2/DRAM if it was valid and dirty, and replace it soon.
- An L2 Cache Index Writeback Invalidate instruction (i.e. – See Section A.4) creates a WBIL2I CMI transaction that causes L2C to write a block back to L2/DRAM if it was valid and dirty, and invalidate it (and unlock it, if necessary).

Except for NUDGE and L2 Cache Index Writeback, all instructions that flush the L2 Cache can also unlock L2 cache blocks.

3.2.12 L2C ECC Codes

TBD

Cavium Confidential For Boyen Yang ATU 02/27/2014

3.3 Enumerations

L2C Quad Error Priority Enumeration
L2C_DAT_ERRPRIO_E

Enumerates the different quad error priorities

Value	Mnemonic	Description
0x0	NBE	No error.
0x1	FBFSBE	FBF single-bit error.
0x2	SBFSBE	SBF single-bit error.
0x3	L2DSBE	L2D single-bit error.
0x4	FBFDBE	FBF double-bit error.
0x5	SBFDBE	SBF double-bit error.
0x6	L2DDBE	L2D double-bit error.

L2C TAD Performance Counter Select Enumeration
L2C_TAD_PRF_SEL_E

Enumerates the different TAD performance counter selects. .

Value	Mnemonic	Description
0x0	NONE	Nothing (disabled).
0x1	L2T_HIT	L2 tag hit.
0x2	L2T_MISS	L2 tag miss.
0x3	L2T_NOALLOC	L2 tag no-allocate (forced).
0x4	L2_VIC	L2 victim.
0x5	SC_FAIL	Store-conditional fail.
0x6	SC_PASS	Store-conditional pass.
0x7	LFB_OCC	LFB occupancy (each cycle adds number of LFBs valid).
0x8	WAIT_LFB	LFB wait LFB (each cycle adds number of LFBs waiting for other LFBs).
0x9	WAIT_VAB	LFB wait VAB (each cycle adds number of LFBs waiting for VAB).
0x80 | α<<4	QD(01)_IDX	Quad(01) index bus in use.
0x81 | α<<4	QD(01)_RDAT	Quad(01) read data bus in use.
0x82 | α<<4	QD(01)_BNKS	Quad(01) number of banks in use (0–4/cycle).
0x83 | α<<4	QD(01)_WDAT	Quad(01) write data flops in use (0–4/cycle).

L2C Tag Error Priority Enumeration
L2C_TAG_ERRPRIO_E

Enumerates the different TAG error priorities. .

Value	Mnemonic	Description
0x0	NBE	No error.
0x1	NOWAY	TAG NOWAY error.
0x2	SBE	TAG single-bit error.
0x3	DBE	TAG double-bit error.

Cavium Confidential For Bowen Yang ALU 02/27/2014

3.4 L2C Registers

The L2C registers are listed in Table 3–2.

Table 3–2 L2C Registers

Register	Address[1]	CSR Type[2]	Detailed Description
L2C_CTL	0x0001180080800000	RSL	See page 240
L2C_QOS_WGT	0x0001180080800008	RSL	See page 240
L2C_ECC_CTL	0x0001180080800010	RSL	See page 241
L2C_TAD_CTL	0x0001180080800018	RSL	See page 242
L2C_XMC_CMD	0x0001180080800028	RSL	See page 243
L2C_BIG_CTL	0x0001180080800030	RSL	See page 244
L2C_COP0_ADR	0x0001180080800038	RSL	See page 244
L2C_COP0_DAT	0x0001180080800040	RSL	See page 246
L2C_XMC(0)_PFC	0x0001180080800800+α*0x40	RSL	See page 246
L2C_XMD(0)_PFC	0x0001180080800808+α*0x40	RSL	See page 246
L2C_RSC(0)_PFC	0x0001180080800810+α*0x40	RSL	See page 246
L2C_RSD(0)_PFC	0x0001180080800818+α*0x40	RSL	See page 246
L2C_INV(0)_PFC	0x0001180080800820+α*0x40	RSL	See page 247
L2C_IOC(0)_PFC	0x0001180080800828+α*0x40	RSL	See page 247
L2C_IOR(0)_PFC	0x0001180080800830+α*0x40	RSL	See page 247
L2C_WPAR_PP(0..3)	0x0001180080840000+α*0x8	RSL	See page 247
L2C_WPAR_IOB(0)	0x0001180080840200+α*0x8	RSL	See page 247
L2C_QOS_PP(0..3)	0x0001180080880000+α*0x8	RSL	See page 248
L2C_QOS_IOB(0)	0x0001180080880200+α*0x8	RSL	See page 248
L2C_TAD(0)_PRF	0x0001180080A00008+α*0x40000	RSL	See page 249
L2C_TAD(0)_TAG	0x0001180080A00010+α*0x40000	RSL	See page 250
L2C_TAD(0)_DLL	0x0001180080A00018+α*0x40000	RSL	See page 250
L2C_TAD(0)_INT	0x0001180080A00028+α*0x40000	RSL	See page 251
L2C_TAD(0)_PFC(0..3)	0x0001180080A00400+α*0x40000+β*0x8	RSL	See page 252
L2C_TAD(0)_ERR	0x0001180080A007D0+α*0x40000	RSL	See page 252
L2C_TQD(0)_ERR	0x0001180080A007D8+α*0x40000	RSL	See page 253
L2C_TTG(0)_ERR	0x0001180080A007E0+α*0x40000	RSL	See page 253
L2C_TBF(0)_BIST_STATUS	0x0001180080A007E8+α*0x40000	RSL	See page 254
L2C_TDT(0)_BIST_STATUS	0x0001180080A007F0+α*0x40000	RSL	See page 254
L2C_TTG(0)_BIST_STATUS	0x0001180080A007F8+α*0x40000	RSL	See page 254
L2C_MCI(0)_INT	0x0001180080C00028+α*0x40000	RSL	See page 255
L2C_MCI(0)_ERR	0x0001180080C007F0+α*0x40000	RSL	See page 255
L2C_MCI(0)_BIST_STATUS	0x0001180080C007F8+α*0x40000	RSL	See page 256
L2C_CBC(0)_INT	0x0001180080E00028+α*0x40000	RSL	See page 256
L2C_CBC(0)_IOCERR	0x0001180080E007E8+α*0x40000	RSL	See page 257
L2C_CBC(0)_RSDERR	0x0001180080E007F0+α*0x40000	RSL	See page 257
L2C_CBC(0)_BIST_STATUS	0x0001180080E007F8+α*0x40000	RSL	See page 258

1. For registers with multiple versions, α represents the values within the specified range.
2. RSL-type registers are accessed indirectly across the I/O Interconnect.

Cavium Confidential For Bowei Yang ALU 02/27/2014

Level 2 Cache Control Register
L2C_CTL

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved.
<31:29>	OCLA_QOS	R/W	0x7	0x7	QOS level for the transactions from OCLA to L2C.
<28>	—	RAZ	—	—	Reserved.
<27>	DISSTGL2I	R/W	0	0	Disable STGL2Is from changing the tags.
<26:25>	—	RAZ	—	—	Reserved.
<24>	DISCCLK	R/W	0	0	Disable conditional clocking in L2C PNR blocks.
<23:16>	—	RAZ	—	—	Reserved.
<15>	RSP_ARB_MODE	R/W	0	0	Arbitration mode for RSC/RSD bus. 0 = round-robin; 1 = static priority. 1. IOR data 2. STIN/FILLs 3. STDN/SCDN/SCFL
<14>	XMC_ARB_MODE	R/W	0	0	Arbitration mode for ADD bus QOS queues. 0 = fully determined through QOS, 1 = QOS0 highest priority; QOS 1–7 use normal mode.
<13:6>	RDF_CNT	R/W	0x0	0x0	Defines the sample point of the LMC response data in the DDR-clock/core-clock crossing. For optimal performance set to 10 × (DDR-clock period/core-clock period) – 1. To disable set to 0. All other values are reserved.
<5:2>	—	RAZ	—	—	Reserved.
<1>	DISECC	R/W	0	0	Tag and data ECC disable.
<0>	DISIDXALIAS	R/W	0	0	Index alias disable.

Level 2 Cache QOS Weight Register
L2C_QOS_WGT

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:56>	WGT7	R/W	0xFF	0xFF	Weight for QOS level 7.
<55:48>	WGT6	R/W	0xFF	0xFF	Weight for QOS level 6.
<47:40>	WGT5	R/W	0xFF	0xFF	Weight for QOS level 5.
<39:32>	WGT4	R/W	0xFF	0xFF	Weight for QOS level 4.
<31:24>	WGT3	R/W	0xFF	0xFF	Weight for QOS level 3.
<23:16>	WGT2	R/W	0xFF	0xFF	Weight for QOS level 2.
<15:8>	WGT1	R/W	0xFF	0xFF	Weight for QOS level 1.
<7:0>	WGT0	R/W	0xFF	0xFF	Weight for QOS level 0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

L2C ECC Control Register
L2C_ECC_CTL

Flip ECC bits to generate single-bit or double-bit ECC errors in all instances of a given memory type. Encodings are as follows.

0x0 = No error.
0x1 = Single-bit error on ECC[0].
0x2 = Single-bit error on ECC[1].
0x3 = Double-bit error on ECC[1:0].

L2DFLIP allows software to generate L2DSBE, L2DDBE, VBFSBE, and VBFDBE errors for the purposes of testing error handling code. When one (or both) of these bits are set, a PL2 that misses in the L2 will fill with the appropriate error in the first two OWs of the fill. Software can determine which OW pair gets the error by choosing the desired fill order (address<6:5>). A PL2 that hits in the L2 will not inject any errors. Therefore sending a WBIL2 prior to the PL2 is recommended to make a miss likely. (If multiple processors are involved, software must be sure that no other processor or I/O device can bring the block into the L2).

To generate a VBFSBE or VBFDBE, software must first get the cache block into the cache with an error using a PL2 that misses the L2. Then a store partial to a portion of the cache block without the error must change the block to dirty. Then, a subsequent WBL2/WBIL2/victim will trigger the VBFSBE/VBFDBE error.

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved.
<9:8>	L2DFLIP	R/W	0x0	0x0	Generate an ECC error in the L2D.
<7:6>	L2TFLIP	R/W	0x0	0x0	Generate an ECC error in the L2T.
<5:4>	RDFFLIP	R/W	0x0	0x0	Generate an ECC error in RDF memory.
<3:2>	XMDFLIP	R/W	0x0	0x0	Generate an ECC error in all corresponding CBC XMD memories.
<1:0>	IOCCMDFLIP	R/W	0x0	0x0	Generate an ECC error in all corresponding IOCCMD memories.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Level 2 Cache TAD Control Register
L2C_TAD_CTL

- If MAXLFB is ≠ 0, VBF_THRESH should be less than MAXLFB.
- If MAXVBF is ≠ 0, VBF_THRESH should be less than MAXVBF.

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:11>	—	RAZ	—	—	Reserved
<10:8>	VBF_THRESH	R/W	0x0	0x0	VBF threshold. When the number of in-use VBFs exceeds this number the L2C TAD increases the priority of all its write operations in the LMC.
<7>	—	RAZ	—	—	Reserved.
<6:4>	MAXVBF	R/W	0x0	0x0	Maximum VBFs in use at once (0 means 8, 1–7 as expected).
<3>	—	RAZ	—	—	Reserved.
<2:0>	MAXLFB	R/W	0x0	0x0	Maximum VABs/LFBs in use at once (0 means 8, 1–7 as expected).

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Level 2 Cache ADD Bus Command Register L2C_XMC_CMD

Note the following:

1. The ADD bus command chosen must not be a IOB-destined command or operation is UNDEFINED.
2. The ADD bus command will have SID forced to IOB, DID forced to L2C, no virtualization checks performed (always pass), and xmdmsk forced to 0. Note that this implies that commands that REQUIRE a STORE cycle (STP, STC, SAA, FAA, FAS) should not be used or the results are unpredictable. The sid = IOB means that the way partitioning used for the command is L2C_WPAR_IOB(0). Neither L2C_QOS_IOB(0) or L2C_QOS_PP(0..3) are used for these commands.
3. Any FILL responses generated by the ADD bus command are ignored. Generated STINs, however, will correctly invalidate the required cores.
4. Any L2D read generated by the ADD bus command records the syndrome information in L2C_TAD(0)_ECC0/1. If ECC is disabled prior to the CSR write, this provides the ability to read the ECC bits directly. If ECC is not disabled, this should log zeros (assuming no ECC errors were found in the block).
5. A write that arrives while the INUSE bit is set will block until the INUSE bit clears. This gives software two options when needing to issue a stream of write operations to L2C_XMC_CMD: polling on the INUSE bit, or allowing hardware to handle the interlock — at the expense of locking up the RSL bus for potentially tens of cycles at a time while waiting for an available LFB/VAB entry. Note that when the INUSE bit clears, the only ordering it implies is that software can send another ADD bus command. Subsequent commands may complete out of order relative to earlier commands.
6. The address written to L2C_XMC_CMD is a physical address. L2C performs hole removal and index aliasing (if enabled) on the written address and uses that for the command. This hole removed/index aliased address is what is returned on a read of the L2C_XMC_CMD register.

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	INUSE	RO/H	0	0	Set to 1 by hardware upon receiving a write; cleared when command has issued (not necessarily completed, but ordered relative to other traffic) and hardware can accept another command.
<62:56>	CMD	R/W	0x0	—	Command to use for simulated ADD bus request. A new request can be accepted.
<55:47>	—	RAZ	—	—	Reserved
<46:44>	QOS	R/W	0x0	—	QOS level to use for simulated ADD bus request.
<43:40>	NODE	R/W	0x0	—	Reserved; must be zero.
<39:0>	ADDR	R/W	0x0	—	Address to use for simulated ADD bus request. (The address written to L2C_XMC_CMD is a physical address. L2C performs hole removal and index aliasing (if enabled) on the written address and uses that for the command. This hole-removed/index-aliased address is what is returned on a read of L2C_XMC_CMD.)

Cavium Confidential For Power Yang ALU 02/27/2014

Level 2 Cache Big Memory Control Register L2C_BIG_CTL

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:4>	MAXDRAM	R/W	0x9	0x4	Amount of configured DRAM. 0x0 = reserved 0x1 = 512 MB 0x2 = 1 GB 0x3 = 2 GB 0x4 = 4 GB 0x5 = 8 GB 0x6 = 16 GB 0x7 = 32 GB 0x8 = 64 GB 0x9 = 128 GB 0xA = 256 GB 0xB = 512 GB 0xC–0xF= reserved Violations of this limit causes L2C to set L2C_INT_REG[BIGRD/BIGWR]. BIGRD interrupts can occur during normal operation as the cores are allowed to prefetch to nonexistent memory locations. Therefore, BIGRD is for informational purposes only. When a HOLERD/BIGRD occurs or HOLEWR/BIGWR blocks a store operation, L2C_TAD(0)_ERR is loaded. L2C_TAD(0)_ERR is not locked for a BIGRD, however.
<3:1>	—	RAZ	—	—	Reserved
<0>	DISBIG	R/W	0	0	Disable the BIG/HOLE logic. When set, the BIG/HOLE is logic disabled completely. When clear, BIGWR and HOLEWR block stores and BIGRD/HOLERD is reported.

Level 2 Cache COP0 Address Register L2C_COP0_ADR

This register provides the address of the COP0 register to read/write when L2C_COP0_DAT is accessed. The [RD] and [SEL] fields where *rd* and *sel* are as defined in Table 2–35 (root register number and select) and Table 2–36 (guest register number and select).

If a COP0 register cannot be accessed by this mechanism, the write is silently ignored and the read data will be 0x2BAD2BAD2BAD2BAD.

If the PPID is outside of the range of 0–3, 255 the write operations are ignored and read requests return 0x2BAD2BAD2BAD2BAD.

If the PPID is outside of the range of 0–3, 255 or if the core in question is in reset, a write operation is ignored and read requests time out the RSL bus.

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:24>	—	RAZ	—	—	Reserved
<23:16>	PPID	R/W	0x0	0x0	Core to access; use 0xFF for broadcast write. Broadcast reads are unpredictable.
<15>	—	RAZ	—	—	Reserved
<14:9>	MBZ	R/W	0x0	0x0	Must be written to zero.
<8>	ROOT	R/W	0	0	If 1, root register is accessed, if 0 guest register is accessed.

Cavium Confidential For Bowen Yang Only 02/21/2017

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<7:3>	RD	R/W	0x0	0x0	COP0 register number
<2:0>	SEL	R/W	0x0	0x0	COP0 sel

The supported [RD]/[SEL] values are shown in Table 3–3.

Table 3–3 Supported RD/SEL Values

RD Value	SEL Value	Type	Description	RD Value	SEL Value	Type	Description	RD Value	SEL Value	Type	Description
4	2	RO	COP0 UserLocal	16	1	RO	PC Fetch Debug	25	5	RO	COP0 PerfCnt Counter2
7	0	RO	COP0 HWREna	16	2	RO	PC Fill Debug	25	6	RO	COP0 PerfCnt Control0
9	0	RO	COP0 Count	16	3	RO	PC Misc Debug	25	7	RO	COP0 PerfCnt Counter0
9	6	RO	COP0 CvmCount	18	0	RO	COP0 WatchLo0	27	0	RO	COP0 CacheErr(Icache)
9	7	RO	COP0 CvmCtl	19	0	RO	COP0 WatchhI0	27	2	RO	COP0 IcacheDebug
11	0	RO	COP0 Compare	22	0	RO	COP0 MultiCoreDebug	28	0	RO	COP0 DataLo(Icache)
11	6	R/W	COP0 PowThrottle	22	1	RO	COP0 VoltageMonitor	28	1	RO	COP0 DataLo(Icache)
12	0	RO	COP0 Status	23	0	RO	COP0 Debug	29	1	RO	COP0 DataHi(Icache)
12	1	RO	COP0 IntCtl	23	6	RO	COP0 Debug2	30	0	RO	COP0 ErrorEPC
12	2	RO	COP0 SRSCtl	24	0	RO	COP0 DEPC	31	0	RO	COP0 DESAVE
13	0	RO	COP0 Cause	25	0	RO	COP0 PerfCnt Control0	31	2	RO	COP0 Scratch
14	0	RO	COP0 EPC	25	1	RO	COP0 PerfCnt Counter0	31	3	RO	COP0 Scratch1
15	0	RO	COP0 PrID	25	2	RO	COP0 PerfCnt Control1	31	4	RO	COP0 Scratch2
15	1	RO	COP0 EBase	25	3	RO	COP0 PerfCnt Counter1				
16	0	RO	PC Issue Debug	25	4	RO	COP0 PerfCnt Control2				

Notes:

PC Issue Debug (RD = 16, SEL = 0):

<63:2> (pc0_5a<63:2>) is often VA<63:2> of the next instruction to issue, but can also be the VA of an instruction executing/replaying on pipe 0, or can also be a VA being filled into the instruction cache, or can also be unpredictable. <61:49> RAZ

<1> set when illegal VA

<0> set when VA is delayslot (prior branch may be either taken or not taken)

PC Fetch Debug (RD = 16, SEL = 0):

<63:0> (fetch_address_3a) VA being fetched from the instruction cache. <61:49>, <1:0> RAZ

PC Fill Debug (RD = 16, SEL = 0):

<63:0> (fill_address_4a<63:2>) VA<63:2> being filled into instruction cache is valid when waiting_for_ifill_4a is set (see PC Misc Debug Info). <61:49> RAZ

<1> set when illegal VA

<0> RAZ

PC Misc Debug (RD = 16, SEL = 0): <63:3> RAZ

<2> (mem_stall_3a) stall term from L1 memory system

<1> (waiting_for_pfill_4a) when waiting_for_ifill_4a is set, indicates whether instruction cache fill is due to a prefetch

<0> (waiting_for_ifill_4a) set when there is an outstanding instruction cache fill

Cavium Confidential For Bowen Yang ALU 02/27/2014

L2C COP0 Data Access Register
L2C_COP0_DAT

Provides data access for the COP0 register specified by the L2C_COP0_ADR register. See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	DATA	R/W/H	0x0	0x0	Data to write to specified COP0 register or return data for a read.

Level 2 Cache ADD bus Performance Counter Registers
L2C_XMC(0)_PFC

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	COUNT	R/W/H	0x0	—	Current counter value

Level 2 Cache STORE bus Performance Counter Registers
L2C_XMD(0)_PFC

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	COUNT	R/W/H	0x0	—	Current counter value

Level 2 Cache COMMIT Bus Performance Counter Registers
L2C_RSC(0)_PFC

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	COUNT	R/W/H	0x0	—	Current counter value

Level 2 Cache FILL Bus Performance Counter Registers
L2C_RSD(0)_PFC

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	COUNT	R/W/H	0x0	—	Current counter value

Cavium Confidential For Bowen Yang ALU 02/27/2014

Level 2 Cache IOC Performance Counter Registers L2C_INV(0)_PFC

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	COUNT	R/W/H	0x0	—	Current counter value

Level 2 Cache IOC Performance Counter Register L2C_IOC(0)_PFC

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	COUNT	R/W/H	0x0	—	Current counter value

Level 2 Cache IOR Performance Counter Register L2C_IOR(0)_PFC

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	COUNT	R/W/H	0x0	—	Current counter value

Level 2 Cache Core Way Partitioning Registers L2C_WPAR_PP(0..3)

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	MASK	R/W	0x0	0x0	Way partitioning mask (1 means do not use). The read value of MASK includes bits set because of the L2C cripple fuses.

Level 2 Cache IOB Way Partitioning Registers L2C_WPAR_IOB(0)

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	MASK	R/W	0x0	0x0	Way partitioning mask (1 means do not use). The read value of MASK includes bits set because of the L2C cripple fuses.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Level 2 Cache Core QOS Level Registers
L2C_QOS_PP(0..3)

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved
<2:0>	LVL	R/W	0x0	0x0	QOS level to use for this core.

Level 2 Cache IOB QOS Level Registers
L2C_QOS_IOB(0)

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved
<6:4>	DWBLVL	R/W	0x0	0x0	QOS level for DWB commands.
<3>	—	RAZ	—	—	Reserved
<2:0>	LVL	R/W	0x0	0x0	QOS level for non-DWB commands.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Level 2 Cache TAD Performance Counter Control Registers
L2C_TAD(0)_PRF

All four counters are equivalent and can use any of the defined selects. The CNT*n*SEL legal values are:

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:24>	CNT3SEL	R/W	0x0	0x1	Selects event to count for L2C_TAD(0)_PFC3. Enumerated by L2C_TAD_PRF_SEL_E.
<23:16>	CNT2SEL	R/W	0x0	0x1	Selects event to count for L2C_TAD(0)_PFC2. Enumerated by L2C_TAD_PRF_SEL_E.
<15:8>	CNT1SEL	R/W	0x0	0x1	Selects event to count for L2C_TAD(0)_PFC1. Enumerated by L2C_TAD_PRF_SEL_E.
<7:0>	CNT0SEL	R/W	0x0	0x1	Selects event to count for L2C_TAD(0)_PFC0. Enumerated by L2C_TAD_PRF_SEL_E.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Level 2 Cache Tag Data Registers
L2C_TAD(0)_TAG

This register holds the tag information for LTGL2I and STGL2I commands.

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:60>	SBLKDTY	R/W/H	0x0	0x0	Sub-block dirty bits.
<59:56>	—	RAZ	—	—	Reserved
<55:48>	BUSINFO	R/W/H	0x0	0x0	The bus information bits
<47>	—	RAZ	—	—	Reserved
<46:40>	ECC	R/W/H	0x0	—	The tag ECC.
<39:17>	TAG	R/W/H	0x0	0x0	The tag. The tag is the corresponding bits from the L2C+LMC internal L2/DRAM byte address.
<16:4>	—	RAZ	—	—	Reserved
<3>	USED	R/W/H	0	0	The LRU use bit. If setting the LOCK bit, the USE bit should also be set or the operation is undefined.
<2>	VALID	R/W/H	0	0	The valid bit
<1>	DIRTY	R/W/H	0	0	The dirty bit
<0>	LOCK	R/W/H	0	0	The lock bit. If setting the LOCK bit, the USE bit should also be set or the operation is undefined.

Level 2 Cache TAD DLL Observability Register
L2C_TAD(0)_DLL

This register provides the parameters for DLL observability. See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:4>	DLL_SETTING	RO/H	0x0	0x1	DLL setting.
<3:1>	DLL_STATE	RO/H	0x0	0x1	DLL state.
<0>	DLL_LOCK	RO/H	0	1	DLL lock: 1 = locked, 0 = unlocked.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Level 2 Cache TAD Interrupt Registers
L2C_TAD(0)_INT

This register is for TAD-based interrupts. See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:34>	—	RAZ	—	—	Reserved
<33>	RTGDBE	R/W1C/H	0	0	RTG double-bit error
<32>	RTGSBE	R/W1C/H	0	0	RTG single-bit error
<31:17>	—	RAZ	—	—	Reserved
<16>	WRDISLMC	R/W1C/H	0	0	Illegal write to disabled LMC error. A DRAM write arrived before the LMC(s) were enabled.
<15>	RDDISLMC	R/W1C/H	0	0	Illegal read to disabled LMC error. A DRAM read arrived before the LMC(s) were enabled.
<14>	BIGRD	R/W1C/H	0	0	Read reference past L2C_BIG_CTL[MAXDRAM] occurred.
<13>	BIGWR	R/W1C/H	0	0	Write reference past L2C_BIG_CTL[MAXDRAM] occurred.
<12>	HOLERD	R/W1C/H	0	0	Read reference to 256MB hole occurred.
<11>	HOLEWR	R/W1C/H	0	0	Write reference to 256MB hole occurred.
<10>	NOWAY	R/W1C/H	0	0	No way was available for allocation. L2C sets NOWAY during its processing of a transaction whenever it needed/wanted to allocate a WAY in the L2 cache, but was unable to. When this bit = 1, it is (generally) not an indication that L2C failed to complete transactions. Rather, it is a hint of possible performance degradation. (For example, L2C must read-modify-write DRAM for every transaction that updates some, but not all, of the bytes in a cache block, misses in the L2 cache, and cannot allocate a WAY.) There is one “failure” case where L2C sets NOWAY: when it cannot leave a block locked in the L2 cache as part of a LCKL2 transaction. See L2C_TTG(0)_ERR for logged information.
<9>	TAGDBE	R/W1C/H	0	0	TAG double-bit error occurred. See L2C_TTG(0)_ERR for logged information.
<8>	TAGSBE	R/W1C/H	0	0	TAG single-bit error occurred. See L2C_TTG(0)_ERR for logged information.
<7:6>	—	RAZ	—	—	Reserved
<5>	FBFDBE	R/W1C/H	0	0	FBF double-bit error occurred. See L2C_TQD(0)_ERR for logged information.
<4>	FBFSBE	R/W1C/H	0	0	FBF single-bit error occurred. See L2C_TQD(0)_ERR for logged information.
<3>	SBFDBE	R/W1C/H	0	0	SBF double-bit error occurred. See L2C_TQD(0)_ERR for logged information.
<2>	SBFSBE	R/W1C/H	0	0	SBF single-bit error occurred. See L2C_TQD(0)_ERR for logged information.
<1>	L2DDBE	R/W1C/H	0	0	L2D double-bit error occurred. See L2C_TQD(0)_ERR for logged information.
<0>	L2DSBE	R/W1C/H	0	0	L2D single-bit error occurred. See L2C_TQD(0)_ERR for logged information.

Cavium Confidential For Broadcom Review ALU 6/27/2014

Level 2 Cache TAD Performance Counter Registers L2C_TAD(0)_PFC(0..3)

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	COUNT	R/W/H	0x0	—	Current counter value

Level 2 Cache TAD Request Error Info Registers L2C_TAD(0)_ERR

This register records error information for HOLE* and BIG* interrupts. The first non-BIGRD error will lock the register until the logged error type is cleared; BIGRD never locks the register. See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	BIGRD	RO/H	0	0	Logged information is for a BIGRD error.
<62>	BIGWR	RO/H	0	0	Logged information is for a BIGWR error.
<61>	HOLERD	RO/H	0	0	Logged information is for a HOLERD error.
<60>	HOLEWR	RO/H	0	0	Logged information is for a HOLEWR error.
<59:58>	—	RAZ	—	—	Reserved
<57:51>	CMD	RO/H	0x0	—	XMC command of request causing error.
<50:44>	SOURCE	RO/H	0x0	—	XMC “source” of request causing error. If SOURCE<6>==0, SOURCE<5:0> = PPID else SOURCE<3:0> is BUSID of IOB which made the request.
<43:40>	NODE	RO/H	0x0	—	Always zero.
<39:0>	ADDR	RO/H	0x0	—	XMC address causing the error. This field is the physical address after hole removal and index aliasing (if enabled). (The hole is between DR0 and DR1. Remove the hole by subtracting 256MB from all L2/DRAM physical addresses ≥ 512 MB.)

Cavium Confidential For Bowen Yang ALD 02/27/2014

Level 2 Cache TAD Quad Error Information Registers L2C_TQD(0)_ERR

This register records error information for all L2D/SBF/FBF errors.

An error locks the L2DIDX, and SYN fields and set the bit corresponding to the error received. DBE errors take priority and overwrite an earlier logged SBE error. Only one of SBE/DBE is set at any given time and serves to document which error the L2DIDX/SYN is associated with.

The syndrome is recorded for DBE errors, though the utility of the value is not clear.

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	L2DDBE	RO/H	0	0	L2DIDX/SYN corresponds to a double-bit L2D ECC error.
<62>	SBFDBE	RO/H	0	0	L2DIDX/SYN corresponds to a double-bit SBF ECC error.
<61>	FBFDBE	RO/H	0	0	L2DIDX/SYN corresponds to a double-bit FBF ECC error.
<60>	L2DSBE	RO/H	0	0	L2DIDX/SYN corresponds to a single-bit L2D ECC error.
<59>	SBFSBE	RO/H	0	0	L2DIDX/SYN corresponds to a single-bit SBF ECC error.
<58>	FBFSBE	RO/H	0	0	L2DIDX/SYN corresponds to a single-bit FBF ECC error.
<57:40>	—	RAZ	—	—	Reserved.
<39:32>	SYN	RO/H	0x0	—	Error syndrome.
<31:18>	—	RAZ	—	—	Reserved.
<17:15>	QDNUM	RO/H	0x0	—	Quad containing the error.
<14>	QDHLF	RO/H	0	—	Quad half of the containing the error.
<13:0>	L2DIDX	RO/H	0x0	—	Index within the quad-half containing the error.

Level 2 Cache TAD Tag Error Information Registers L2C_TTG(0)_ERR

This register records error information for all TAG SBE/DBE/NOWAY errors.

The priority of errors (lowest to highest) is NOWAY, SBE, DBE. An error locks the SYN, WAY, and L2IDX fields for equal or lower priority errors until cleared by software.

The syndrome is recorded for DBE errors, though the utility of the value is not clear.

A NOWAY error does not change the value of the SYN field, and leaves WAY unpredictable. L2IDX[16:7] is the L2 block index associated with the command which had no way to allocate.

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	TAGDBE	RO/H	0x0	0x0	Information refers to a double-bit TAG ECC error.
<62>	TAGSBE	RO/H	0x0	0x0	Information refers to a single-bit TAG ECC error.
<61>	NOWAY	RO/H	0x0	0x0	Information refers to a NOWAY error.
<60:39>	—	RAZ	—	—	Reserved.
<38:32>	SYN	RO/H	0x0	0x0	Syndrome for the single-bit error.

Cavium Confidential For Bowen Yang ALU 02/2/2012

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:19>	—	RAZ	—	—	Reserved.
<18:17>	WAY	RO/H	0x0	—	Way of the L2 block containing the error
<16:7>	L2IDX	RO/H	0x0	—	Index of the L2 block containing the error
<6:0>	—	RAZ	—	—	Reserved

Level 2 Cache TAD Quad Buffer BIST Status Registers
L2C_TBF(0)_BIST_STATUS

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	VBFFL	RO	0x0	0x0	BIST failure status for VBF ({QD7H1,QD7H0, ... , QD0H1, QD0H0})
<47:32>	SBFFL	RO	0x0	0x0	BIST failure status for SBF ({QD7H1,QD7H0, ... , QD0H1, QD0H0})
<31:16>	FBFRSPFL	RO	0x0	0x0	BIST failure status for FBF RSP port ({QD7H1,QD7H0, ... , QD0H1, QD0H0})
<15:0>	FBFWRPFL	RO	0x0	0x0	BIST failure status for FBF WRP port ({QD7H1,QD7H0, ... , QD0H1, QD0H0})

Level 2 Cache TAD Data BIST Status Registers
L2C_TDT(0)_BIST_STATUS

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	L2DFL	RO	0x0	0x0	BIST failure status for L2D ({QD7H1,QD7H0, ... , QD0H1, QD0H0})

Level 2 Cache TAD Tag BIST Status Registers
L2C_TTG(0)_BIST_STATUS

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:32>	RTGFL	RO	0x0	0x0	Always 0x0.
<31:18>	—	RAZ	—	—	Reserved
<17>	LRULFBFL	RO	0	0	Reserved, always zero.
<16>	LRUFL	RO	0	0	BIST failure status for tag LRU
<15:4>	—	RAZ	—	—	Reserved
<3:0>	TAGFL	RO	0x0	0x0	BIST failure status for TAG ways 0–3.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Level 2 Cache MCI Interrupt Registers
L2C_MCI(0)_INT

This register is for MCI-based interrupts. See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	VBFDBE	R/W1C/H	0	0	VBF double-bit error occurred. See L2C_MCI(0)_ERR for logged information.
<0>	VBFSBE	R/W1C/H	0	0	VBF single-bit error occurred. See L2C_MCI(0)_ERR for logged information.

Level 2 Cache MCI Error Information Registers
L2C_MCI(0)_ERR

This register records error information for all MCI errors.

An error locks the VBF4, INDEX, and SYN0/1 fields and set the bit corresponding to the error received. VBFDBE errors take priority and will overwrite an earlier logged VBFSBE error. The information from exactly one VBF read is present at any given time and serves to document which error(s) were present in the read with the highest priority error.

The syndrome is recorded for DBE errors, though the utility of the value is not clear.

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	VBFDBE1	RO/H	0	0	INDEX/SYN1 corresponds to a double-bit VBF ECC error
<62>	VBFDBE0	RO/H	0	0	INDEX/SYN0 corresponds to a double-bit VBF ECC error
<61>	VBFSBE1	RO/H	0	0	INDEX/SYN1 corresponds to a single-bit VBF ECC error
<60>	VBFSBE0	RO/H	0	0	INDEX/SYN0 corresponds to a single-bit VBF ECC error
<59:48>	—	RAZ	—	—	Reserved
<47:40>	SYN1	RO/H	0x0	0x0	Error syndrome for QW1 ([127:64]).
<39:32>	SYN0	RO/H	0x0	0x0	Error syndrome for QW0 ([63:0]).
<31:12>	—	RAZ	—	—	Reserved
<11>	VBF4	RO/H	0	—	When 1, errors were from VBF (4+a), when 0, from VBF (0+a).
<10:4>	INDEX	RO/H	0x0	—	VBF index which was read and had the error(s).
<3:0>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang ALU 02/2/2014

Level 2 MCI BIST Status (DCLK) Registers
L2C_MCI(0)_BIST_STATUS

If clear BIST is desired, CLEAR_BIST must be written to 1 before START_BIST is written to 1 using a separate CSR write operation.

CLEAR_BIST must not be changed after writing START_BIST to 1 until the BIST operation completes (indicated by START_BIST returning to 0) or operation is undefined.

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	START_BIST	R/W/H	0	0	When written to 1, starts BIST. remains 1 until BIST is complete.
<62>	CLEAR_BIST	R/W	0	0	When BIST is triggered, run clear BIST.
<61:2>	—	RAZ	—	—	Reserved
<1:0>	VBFFL	RO	0x0	0x0	BIST failure status for VBF0–1. VBFFL<1> is always 0.

Level 2 Cache CBC Interrupt Registers
L2C_CBC(0)_INT

This register is for CBC-based interrupts. See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	IOCCMDDBE	R/W1C/H	0	0	IOCCMD double-bit error occurred. See L2C_CBC(0)_IOCERR for logged information.
<2>	IOCCMDSBE	R/W1C/H	0	0	IOCCMD single-bit error occurred. See L2C_CBC(0)_IOCERR for logged information.
<1>	RSDDBE	R/W1C/H	0	0	RSD double-bit error occurred. See L2C_CBC(0)_IOCERR for logged information.
<0>	RSDSBE	R/W1C/H	0	0	RSD single-bit error occurred. See L2C_CBC(0)_IOCERR for logged information.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Level 2 Cache CBC Error Information Registers
L2C_CBC(0)_IOCERR

This register records error information for all CBC IOC errors.

An error locks the INDEX, and SYN fields and set the bit corresponding to the error received. CMDDBE errors take priority and overwrite an earlier logged CMDSBE error. Only one of CMDSBE/CMDDBE is set at any given time and serves to document which error the INDEX/SYN is associated with.

The syndrome is recorded for DBE errors, though the utility of the value is not clear.

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	CMDDBE	RO/H	0	0	INDEX/SYN corresponds to a double-bit IOCCMD ECC error.
<62>	CMDSBE	RO/H	0	0	INDEX/SYN corresponds to a single-bit IOCCMD ECC error.
<61:40>	—	RAZ	—	—	Reserved
<39:32>	SYN	RO/H	0x0	0x0	Error syndrome.
<31:3>	—	RAZ	—	—	Reserved
<2:0>	XMCNUM	RO/H	0x0	0x0	Indicates the XMC that had the error.

Level 2 Cache RSD Error Information Registers
L2C_CBC(0)_RSDERR

This register records error information for all CBC RSD errors.

An error locks the INDEX, and SYN fields and set the bit corresponding to the error received. RSDDBE errors take priority and overwrite an earlier logged RSDSBE error. Only one of RSDSBE/RSDDBE is set at any given time and serves to document which error the INDEX/SYN is associated with.

The syndrome is recorded for DBE errors, though the utility of the value is not clear.

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	RSDDBE	RO/H	0	0	INDEX/SYN corresponds to a double-bit RSD ECC error
<62>	RSDSBE	RO/H	0	0	INDEX/SYN corresponds to a single-bit RSD ECC error
<61:40>	—	RAZ	—	—	Reserved
<39:32>	SYN	RO/H	0x0	0x0	Error syndrome.
<31:9>	—	RAZ	—	—	Reserved
<8:6>	TADNUM	RO/H	0x0	0x0	Indicates the TAD FIFO containing the error.
<5:4>	QWNUM	RO/H	0x0	0x0	Indicates the QW containing the error.
<3:0>	RSDNUM	RO/H	0x0	0x0	Indicates the RSD that had the error.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Level 2 Cache CBC BIST Status Registers
L2C_CBC(0)_BIST_STATUS

See Table 3–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:34>	—	RAZ	—	—	Reserved
<33:32>	IOCCMDFL	RO	0x0	0x0	BIST failure status for IOCCMD0–1.
<31:0>	RSDFL	RO	0x0	0x0	BIST failure status for RSDQW0–31.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 4

DRAM Controller (LMC)

This chapter describes the following topics:

- Overview
- Internal and DRAM Part Addressing
- DRAM Read Example
- DRAM Write Example
- DRAM Programming
- DRAM Refreshes, DRAM ZQ Calibration, Internal ZQ Calibration, and Internal DLL Fine-Tuning
- DRAM Scheduler Performance
- DRAM Chip Select, CKE, and ODT Signals
- LMC Initialization Sequence
- DRAM ECC Codes
- Dynamic Precharge Power-Down
- Non-Homogeneous Ranks
- Address Mirroring
- CK Speed Programming
- Example System Configuration
- Enumerations
- LMC Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The LMC DDR3 SDRAM controller conforms to the JEDEC DDR3 SDRAM specification. (Refer to the JEDEC standard: DDR3 SDRAM Specification JESD79-3E, July 2010).

The LMC can communicate with the HFA unit; the HFA interconnect can be used for reading HFA graphs in DRAM. Chapter 11 discusses more about HFA graph read operations.

CN71XX has the following LMC DDR3 SDRAM controller features:

- Up to 8 GB of attached DDR3 SDRAM
- 32-bit interface (36 bits wide with optional SECDED ECC)
- Data rates up to 1333 MHz
- Two ranks / chip selects
- ×8 and ×16 DRAM parts supported
- Non-homogeneous rank/chip-selects
- Low-address cache-block bank selection with optional hashing
- Low-latency and high-bandwidth HFA graph read interfaces
- Aggressive 8-bank reordering scheduling
  - Entire 128-byte cache-block read/write operations via two consecutive read/write CAS operations
  - Bank autoprecharge following each cache block read/write operations
- 1T and 2T address/commands (see Micron Technical Note TN-41-08)
- Per-rank ODT enable pins
  - Separate programmability for read operations and write operations
- ¼ cycle granularity programmable delays for read-leveling and write-leveling
- Per-rank address mirroring
- Automatic precharge power-down mode entry/exit to reduce DRAM power consumption when idle
- Address bus protection via its optional inclusion in SECDED ECC code
- Nonexistent DRAM address checking
- Automatic memory state preservation (i.e. self-refresh entry) on chip warm or soft reset

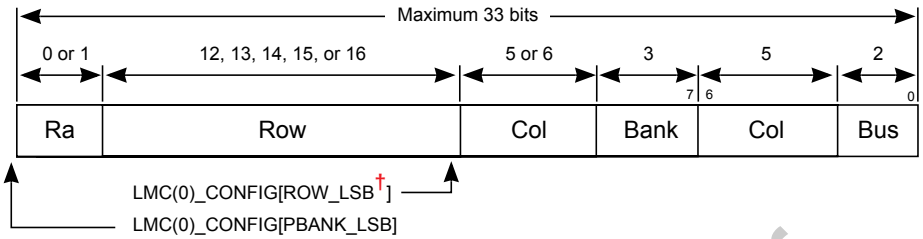
Cavium Confidential For Bowen Yang ALU 02/27/2014

4.1 Internal and DRAM Part Addressing

LMC uses L2C+LMC internal addresses to identify attached DRAM. Section 3.2.9 in Chapter 3 describes the 39-bit L2C+LMC internal addresses.

Figure 4–1 shows the partitioning of these addresses into DRAM part components.

Figure 4–1 SDRAM Physical Address



Bus = selects the byte on the 36-bit DDR3 bus
Col = column address for the DDR3 part (10 or 11 bits)
Bank = bank address for the DDR3 part (3 bits)
Row = row address for the DDR3 part (12, 13, 14, 15, or 16 bits)
Ra = optional rank address for a dual-rank DIMM (present when LMC(0)_CONFIG[RANK_ENA] = 1 (one bit wide if it exists))

NOTE:	† **ROW_LSB** is either LMC(0)_CONFIG[ROW_LSB] or LMC(0)_DUAL_MEMCFG[ROW_LSB].

- The two bits of the DRAM part address select the DRAM part bytes on the 32-bit bus.
- Bits <9:7> select the bank (with others when LMC(0)_CONTROL[XOR_BANK] is set).
- The column bits for the DRAM parts fill in the remaining of the low address bits, followed by the row bits for the DRAM parts.
- The rank within DIMM selection field (if more than one rank is present and LMC(0)_CONFIG[RANK_ENA]=1) is the next bits.

LMC can never utilize all of the 39-bit L2C + LMC internal address space. L2C_BIG_CTL[MAXDRAM] should match the amount of DRAM attached, and the upper address bits are unused/zero. Also, when different ranks are different sizes, the DIMM/RANK fields must be programmed for the largest rank, and there are nonexistent addresses within the smaller ranks. With proper LMC(0)_NXM[CS_MASK,MEM_MSB_D*_R*] programming, LMC identifies accesses to all these nonexistent addresses. LMC completes read operations to nonexistent addresses after it aliases them to existing DRAM locations. LMC blocks write operations to nonexistent addresses and sets LMC(0)_INT[NXM_WR_ERR] to indicate the error (assuming LMC(0)_CONTROL[NXM_WRITE_EN] = 0).

The DDR3 part bank LMC selects to service a request is not just the Bank field shown in Figure 4–1 when LMC(0)_CONTROL[XOR_BANK] is set. When LMC(0)_CONTROL[XOR_BANK] is set to one, LMC supports hashed bank selection:

bank<2:0> = byte_address<9:7> ⊕ byte_address<14:12>

Most applications will find it advantageous to use hashed bank selection (i.e. to set LMC(0)_CONTROL[XOR_BANK] = 1).

Cavium Confidential For Bowen Yang ALU 09/12/2014

LMC's bank selection has the following properties:

- All of the locations within a single cache block reside in the same row.
- Sequential cache blocks are always/usually in different banks.

  Always when LMC(0)_CONTROL[XOR_BANK] = 0

  Usually when LMC(0)_CONTROL[XOR_BANK] = 1.
- Sequential cache blocks are usually in the same rank/DIMM.
- Bank distribution for skewed access when LMC(0)_CONTROL[XOR_BANK] = 1.
  - Suppose an application uses many buffers whose size is 4KB (or larger) and a power-of-two.
  - Suppose, further, that the buffers are naturally-aligned based on their size and that the application only ever uses the first two (128B) cache blocks in any given buffer.

  With LMC(0)_CONTROL[XOR_BANK] = 0, only banks 0 and 1 are used by the described application.

  With LMC(0)_CONTROL[XOR_BANK] = 1, however, all four/eight banks are used by the described application.

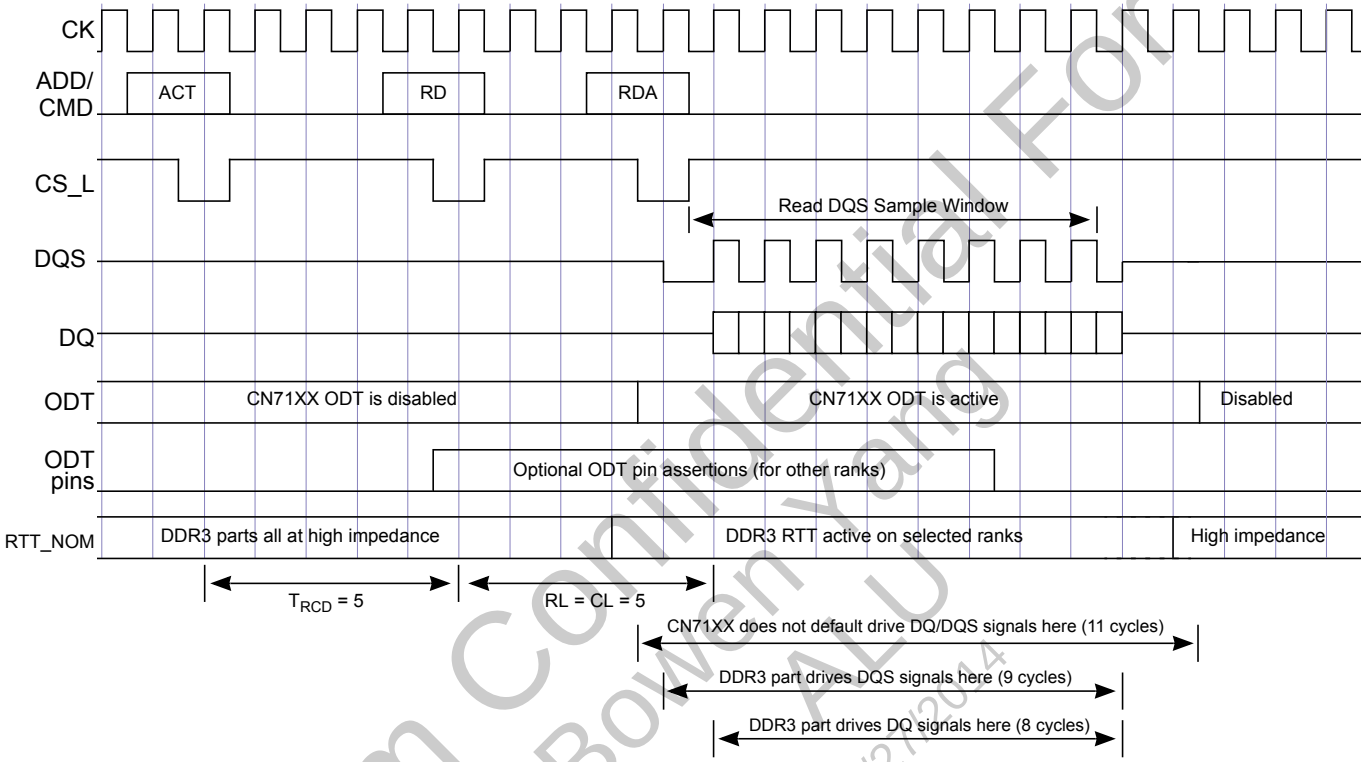
Cavium Confidential For Bowen Yang ALU 02/27/2014

4.2 DRAM Read Example

CN71XX always reads entire 128-byte cache blocks on the main-memory DDR3 interface. There is a single address/command bus to drive all commands to all DIMMs/parts. The entire 32/36-bit interface operates as one, though the bytes from/to individual DRAM parts can have different skews. Figure 4–2 shows an isolated cache-block read transaction for an impossible idealized case with no delays/skews other than the clock edges. We will describe ways that LMC compensates for real conditions below. A cache-block read transaction consists of one bank activate, four column-address commands, and the data transfer of the 128-byte cache block on the DRAM data bus.

Note that Figure 4–2 shows only two column-address commands. The behavior of four consecutive column-address commands is similar.

Figure 4–2 Cache-block Read Transaction Example



Cavium Confidential For Bowen Yang ALU 02/27/2014

The signals in Figure 4–2 are defined as follows:

CK indicates the behavior of the CN71XX DDR_CK_*_P/DDR_CK_*_N signals.

ADD/CMD indicates the behavior of the CN71XX DDR_CAS_L, DDR_RAS_L, DDR_A*, DDR_BA*, and DDR_WE_L signals.

CS_L indicates the behavior of the CN71XX DDR_DIMM*_CS*_L signals for the selected rank. Other CN71XX DDR_DIMM*_CS*_L signals are inactive.

DQS indicates the behavior of the CN71XX DDR_DQS_*_P/DDR_DQS_*_N and DDR_CBS_0_P/DDR_CBS_0_N signals.

DQ indicates the behavior of the CN71XX DDR_DQ* and DDR_CB* signals.

ODT indicates when the CN71XX on-die-termination resistance is enabled for both DQS and DQ.

ODT pins indicates the behavior of the CN71XX DDR_DIMM*_ODT_* signals.

RTT_NOM indicates when the on-die-termination on the DDR3 parts is enabled. This termination resistance value is selected by MR1[Rtt_Nom] in the DRAM part, which should match one of LMC(0)_MODEREG_PARAMS1[RTT_NOM_*].

Figure 4–2 assumes the following:

- 2T addressing mode (LMC(0)_CONTROL[DDR2T] = 1)
- T_{RCD} = 5 cycles (LMC(0)_TIMING_PARAMS1[TRCD] = 4)
- CL = CWL = 5 cycles, AL = 0 (LMC(0)_MODEREG_PARAMS0[CL] = 2, LMC(0)_MODEREG_PARAMS0[CWL] = 0, LMC(0)_MODEREG_PARAMS0[AL] = 0)
- Zero read skews/leveling and zero board delays (LMC(0)_RLEVEL_RANK(0..3)[BYTE*] = 0)
- Minimum-sized read window for DQ/DQS (LMC(0)_CONTROL[FPRCH2, BPRCH] = 0)
- Not in registered-DIMM mode (LMC(0)_CONTROL[RDIMM_ENA] = 0)
- The longer ODT pin assertion (LMC(0)_CONTROL[RODT_BPRCH] = 1)

The cache-block read in Figure 4–2 starts with CN71XX driving the activate (ACT) command onto ADD/CMD to activate the DRAM bank. LMC is in 2T addressing mode in this example, so it drives the ACT for two consecutive cycles on ADD/CMD, asserting the chip-select (CS_L) for only the second of the two. This 2T addressing mode provides extra set-up time for ADD/CMD, which may be heavily-loaded with many attached DRAM parts. (See Micron Technical Note TN-41-08.)

The next event in Figure 4–2 is the RD CAS command. The example shows the RD spaced the minimum distance from the ACT. This will happen for an isolated cache block read, but in other circumstances, LMC may schedule a larger interval between the ACT and the RD. An entire 128B cache block read requires two CAS operations, and LMC always schedules the second CAS exactly four cycles after the first. The second is always an RDA CAS command, which provides the bank closure and precharge that are necessary following the ACT that activated the bank.

The ODT pin behavior in Figure 4–2 relates closely to the CAS commands. The optional ODT pin assertion can improve electrical behavior in systems with more than one DIMM. The figure assumes CL = CWL, so LMC asserts the ODT pins the same cycle as the first CAS operation. The ODT pins selected in LMC(0)_RODT_MASK assert for 11 cycles. The RTT_NOM row shows when selected DRAM parts turn on their on-die-termination in response to the ODT pin assertion, which is CWL – 2 cycles later.

Whenever the DDR3 DQS/DQ pins are idle, the bus is tri-stated. Figure 4–2 shows the minimum 11-cycle read interval for this. LMC always opens the same read window for all of DQS/DQ. The ODT row in Figure 4–2 shows that LMC enables CN71XXs on-die-termination for both DQS/DQ during a read operation. LMC(0)_COMP_CTL2[RODT_CTL] selects the CN71XX on-die-termination resistance. LMC(0)_COMP_CTL2[DQX_CTL] selects the drive strength of CN71XX's default-drive at most times. The only exception is that if LMC(0)_PHY_CTL[TS_STAGGER] = 1, then the CN71XX default drive strength is 60 Ohm, rather than full strength, for one cycle each time the default-drive re-activates.

Figure 4–2 shows the DDR3 parts driving the DQS pins for 9 cycles, and the DQ pins for 8 cycles to return the data and complete the cache block read. For 2 (3) cycles of the read window, neither CN71XX nor any DRAM parts directly drive the DQS (DQ)

Cavium Confidential For Bowen Yang 02-21-2014

pins, though both CN71XX's ODT and the ODT/RTT_NOM of other DRAM parts may be active. The DRAM parts eventually drive 16 DQS edges along with the returning DQ data. CN71XX must properly detect these DQS edges in order to properly sample the DQ pins. Figure 4–2 depicts the Read DQS Sample Window, which is where CN71XX can receive the DQS edges. LMC opens the Read DQS Sample Window for a byte just before it expects the first DQS edge from the DRAM part. CN71XX closes the Read DQS Sample Window for a byte when it receives the last DQS edge for the byte. The CN71XX read logic ignores all DQS activity outside of the Read DQS Sample Window.

As we mentioned, Figure 4–2 shows an impossible idealized case with no delays/skews. In the remainder of this section, we indicate the effect of various real delays/skews and other parameter changes on the components of Figure 4–2.

- ADD/CMD bus delays

  The direct effect of ADD/CMD bus delays to Figure 4–2 is to shift the RTT_NOM row to the right by the amount of the delay.

  Of course, ADD/CMD bus delays must indirectly increase DRAM part DQS/DQ response delays ...

- DQS/DQ response delays

  DQS/DQ response delays cause the interval that the DRAM part drives DQS and DQ in Figure 4–2 to shift to the right by the amount of the delay. LMC(0)_RLEVEL_RANK(0..3)[BYTE*] should be correspondingly increased to allow LMC to properly capture the delayed DQS/DQ response data.

- LMC(0)_RLEVEL_RANK(0..3)[BYTE*] increase

  Real delays will likely require nonzero LMC(0)_RLEVEL_RANK(0..3)[BYTE*] values. There is an independent LMC(0)_RLEVEL_RANK(0..3)[BYTE*] value for each byte, which means CN71XX can tolerate different delays between DRAM parts. Referring to Figure 4–2, the positionings of the read interval, Read DQS Sample Window, and DDR_DIMM*_ODT_* activation are all byte-specific.

  LMC(0)_RLEVEL_RANK(0..3)[BYTE*] is a 6-bit value indicating the amount of delay LMC should expect for the byte, measured in units of ¼ CK. Bits <5:2> indicate the number of whole CKs, and bits <1:0> indicate the number of ¼ CKs additional.

  Whenever <5:0> increases by *N*, the Read DQS Sample Window opens *N*/4 CKs later than Figure 4–2 shows.

  Whenever <5:2> increases by *N*, the read interval and the CN71XX ODT activation for DQS and DQ both shift *N* CKs to the right. (The width of the read interval does not change.)

- LMC(0)_TIMING_PARAMS1[TRCD] increase

  An increase by *N* shifts the ACT and corresponding CS_L in Figure 4–2 to the left by *N* CKs.

- LMC(0)_MODEREG_PARAMS0[CL] increase

  An increase by *N* shifts the entire ADD/CMD and CS_L rows in Figure 4–2 to the left by *N* CKs.

Cavium Confidential For Beverly Yang ALU 6/27/2014

- LMC(0)_MODEREG_PARAMS0[CWL] increase

  An increase by *N* shifts the entire ODT pins row in Figure 4–2 to the left by *N* CKs. (Note that CWL cannot increase more than CL does from the values assumed in the figure.)

- LMC(0)_CONTROL[FPRCH2] increase

  An increase by *N* causes the read interval and the CN71XX ODT activation for DQS and DQ to start *N* CKs sooner than in Figure 4–2 for all bytes. The width of both these intervals grow by *N* CKs from that shown in Figure 4–2.

- LMC(0)_CONTROL[BPRCH] increase

  An increase by *N* causes the read no-drive interval and the CN71XX ODT activation for DQS and DQ to complete *N* CKs later than in Figure 4–2 for all bytes. The width of both these intervals grow by *N* CKs from that shown in Figure 4–2.

- LMC(0)_CONTROL[RDIMM_ENA] set

  Shifts the entire ADD/CMD and CS_L rows in Figure 4–2 to the left by 1 CK. This is necessary for registered DIMMs.

- LMC(0)_CONTROL[RODT_BPRCH] = 0 causes the CN71XX ODT pins and RTT_NOM active windows to be inactive one cycle sooner. The width of both of these intervals decreases by one CK.

  If both DIMM slots are populated and the board delay between them is greater than ¼ CK cycle, LMC(0)_CONTROL[RODT_BPRCH] = 1 ensures RTT on the inner DIMM will remain high while CN71XX reads from the outer DIMM.

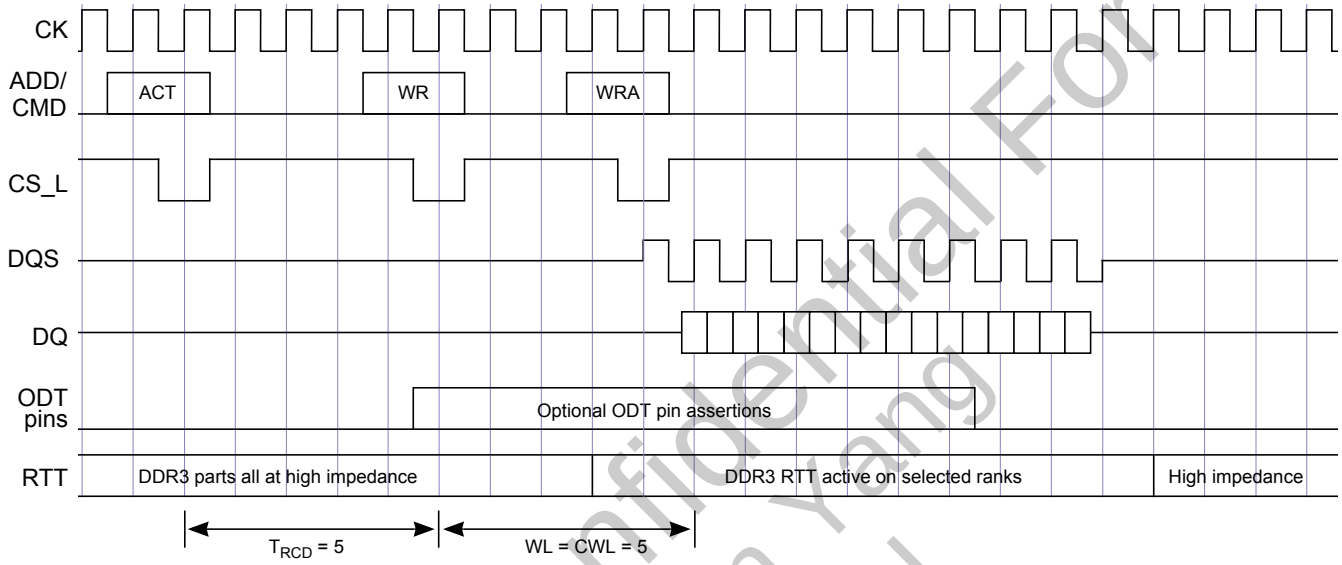
Cavium Confidential For Bowen Yang ALU 02/27/2014

4.3 DRAM Write Example

CN71XX always writes entire 128-byte cache blocks on the main-memory DDR3 interface. There is a single address/command bus to drive all commands to all DIMMs/parts. The entire 32/36-bit interface operates as one, though the bytes from/to individual DRAM parts can have different skews. Figure 4–3 shows an isolated cache-block write transaction for an idealized case with no delays/skews other than the clock edges. We will describe ways that LMC compensates for real conditions below. A cache-block write transaction consists of one bank activate, four column-address commands, and the data transfer of the 128-byte cache block on the DRAM data bus.

Note that Figure 4–3 shows only two column-address commands. The behavior of four consecutive column-address commands is similar.

Figure 4–3 Cache-block Write Transaction Example



Cavium Confidential For Bowen Yang ALU 02/17/2014

The signals in Figure 4–3 are defined as follows:

CK indicates the behavior of the CN71XX DDR_CK_*_P/DDR_CK_*_N signals.

ADD/CMD indicates the behavior of the CN71XX DDR_CAS_L, DDR_RAS_L, DDR_A*, DDR_BA*, and DDR_WE_L signals.

CS_L indicates the behavior of the CN71XX DDR_DIMM*_CS*_L for the selected rank. Other CN71XX DDR_DIMM*_CS*_L signals are inactive.

DQS indicates the behavior of the CN71XX DDR_DQS_*_P/DDR_DQS_*_N and DDR_CBS_0_P/DDR_CBS_0_N signals.

DQ indicates the behavior of the CN71XX DDR_DQ* and DDR_CB* signals.

ODT pins indicates the behavior of the CN71XX DDR_DIMM*_ODT_* signals.

RTT indicates when the on-die-termination on the DDR3 parts is enabled. This termination resistance value is selected by MR1[Rtt_Nom] or MR2[Rtt_WR] in the DRAM part, which should match one of LMC(0)_MODEREG_PARAMS1[RTT_NOM_*] and LMC(0)_MODEREG_PARAMS1[RTT_WR_*], respectively. According to the rules in the JEDEC DDR3 specifications, MR1[Rtt_Nom] selects

the termination resistance value used by the DRAM part during the RTT active interval in most cases. The exception is when dynamic ODT is enabled and the ODT pin is active for the DRAM part that is the destination for the write, in which case MR2[Rtt_WR] selects the termination resistance value for the first 10 CK cycles of the RTT active interval in the figure for the destination part.

Figure 4–3 assumes the following:

- 2T addressing mode (LMC(0)_CONTROL[DDR2T] = 1)
- T_{RCD} = 5 cycles (LMC(0)_TIMING_PARAMS1[TRCD] = 4)
- CWL = 5 cycles, AL = 0 (LMC(0)_MODEREG_PARAMS0[CWL] = 0, LMC(0)_MODEREG_PARAMS0[AL] = 0, LMC(0)_CONFIG[EARLY_DQX] = 0)
- Zero write leveling (LMC(0)_WLEVEL_RANK(0..3)[BYTE*] = 0)
- Not in registered-DIMM mode (LMC(0)_CONTROL[RDIMM_ENA] = 0)
- The longer ODT pin assertion (LMC(0)_CONTROL[WODT_BPRCH] = 1)

The cache-block write transaction in Figure 4–3 starts with CN71XX driving the activate (ACT) command onto ADD/CMD to activate the DRAM bank. LMC is in 2T addressing mode in this example, so it drives the ACT for two consecutive cycles on ADD/CMD, asserting the chip-select (CS_L) for only the second of the two. This 2T addressing mode provides extra set-up time for ADD/CMD, which may be heavily-loaded with many attached DRAM parts. (See Micron Technical Note TN-41-08.)

The next event in Figure 4–3 is the WR CAS command. The example shows the WR spaced the minimum distance from the ACT. This will happen for an isolated cache block write, but in other circumstances, LMC may schedule a larger interval between the ACT and the WR. An entire 128B cache-block write requires two CAS operations, and LMC always schedules the second CAS exactly four cycles after the first. The second is always an WRA CAS command, which provides the bank closure and precharge that are necessary following the ACT that activated the bank.

The ODT pin behavior in Figure 4–3 relates closely to the CAS commands. The optional ODT pin assertion can improve electrical behavior. LMC asserts the ODT pins the same cycle as the first CAS operation. The ODT pins selected by LMC(0)_WODT_MASK assert for 11 cycles. The RTT row shows when selected DRAM parts turn on their on-die-termination in response to the ODT pin assertion, which is CWL – 2 cycles later.

Whenever the DDR3 DQS/DQ pins are idle, the DQ and DQS buses are tri-stated. LMC(0)_COMP_CTL2[DQX_CTL] selects CN71XX's drive strength in all of Figure 4–3.

Figure 4–3 shows CN71XX driving 18 DQS edges along with the DQ data aligned to the last 16 edges to complete the cache-block write transaction.

As we mentioned, Figure 4–3 shows an idealized case with no delays/skews. In the remainder of this section, we indicate the effect of various real delays/skews and other parameter changes on the components of Figure 4–3.

- ADD/CMD bus delays

  The direct effect of ADD/CMD bus delays to Figure 4–3 is to shift the RTT row to the right by the amount of the delay.

- ADD/CMD and DQS/DQ delays

  These effects can not directly be seen in Figure 4–3.

Cavium Confidential
Bowen Yang
ALU
06/27/2014

Most likely, the ADD/CMD bus delays will be larger than the DQS/DQ delays, and LMC(0)_WLEVEL_RANK(0..3)[BYTE*] should be correspondingly increased to allow the DRAM part to properly capture the DQS/DQ data.

Less likely, there may be some boards with CK delays smaller than the DQS/DQ delays. In this case, LMC(0)_CONFIG[EARLY_DQX] can be set to one, and LMC(0)_WLEVEL_RANK(0..3)[BYTE*] adjusted to allow the DRAM part to properly capture the DQS/DQ data.

- LMC(0)_WLEVEL_RANK(0..3)[BYTE*] increase

  Real delays may require nonzero LMC(0)_WLEVEL_RANK(0..3)[BYTE*] values. There is an independent LMC(0)_WLEVEL_RANK(0..3)[BYTE*] value for each byte, which means CN71XX can tolerate different delays between DRAM parts. Referring to Figure 4–3, the positionings of the DQS and DQ rows are byte-specific.

  LMC(0)_WLEVEL_RANK(0..3)[BYTE*] is a 5-bit value indicating the amount of delay LMC should use for the byte, measured in units of 1/8 CK. Whenever it increases by *N*, the DQS and DQ rows in Figure 4–3 shift to the right by *N*/8 CKs. (Note that the low bit of LMC(0)_WLEVEL_RANK(0..3)[BYTE*] must be zero during normal operation, so LMC really only provides ¼ CK delay precision, not 1/8 CK.)

- LMC(0)_CONFIG[EARLY_DQX] set

  Shifts the entire DQS and DQ rows in Figure 4–3 to the left by 1 CK for all bytes.

- LMC(0)_TIMING_PARAMS1[TRCD] increase

  An increase by *N* shifts the ACT and corresponding CS_L in Figure 4–3 to the left by *N* CKs.

- LMC(0)_MODEREG_PARAMS0[CWL] increase

  An increase by *N* shifts all the DQS, DQ, and RTT rows in Figure 4–3 to the right by *N* CKs.

- LMC(0)_CONTROL[RDIMM_ENA] set

  Shifts the entire ADD/CMD and CS_L rows in Figure 4–3 to the left by 1 CK. This is necessary for registered DIMMs.

- LMC(0)_CONTROL[WODT_BPRCH] = 0 causes the CN71XX ODT pins and RTT active windows to be inactive one cycle sooner. The width of both of these intervals decreases by one CK.

  If both DIMM slots are populated and the board delay between them is greater than ¼ CK cycle, LMC(0)_CONTROL[WODT_BPRCH] = 1 ensures RTT on the inner DIMM will remain high while CN71XX writes to the outer DIMM.

Cavium Confidential For Bowen Yang ALU 06/27/2014

4.4 DRAM Programming

Most of the parameters to the CN71XX DRAM controller come from DRAM part and DIMM specifications. If desired, CN71XX's TWSI interface can be used to read the SPD of the DIMMs, refer to Chapter 31. These part/DIMM parameters go into fields in the LMC(0)_CONFIG, LMC(0)_CONTROL, LMC(0)_TIMING_PARAMS0, LMC(0)_TIMING_PARAMS1, LMC(0)_MODEREG_PARAMS0, LMC(0)_MODEREG_PARAMS1, LMC(0)_DUAL_MEMCFG, LMC(0)_NXM, LMC(0)_SLOT_CTL0, LMC(0)_SLOT_CTL1, LMC(0)_SLOT_CTL2, LMC(0)_WODT_MASK, LMC(0)_RODT_MASK, LMC(0)_DIMM(0..1)_PARAMS, and LMC(0)_DIMM_CTL.

4.5 DRAM Refreshes, DRAM ZQ Calibration, Internal ZQ Calibration, and Internal DLL Fine-Tuning

LMC periodically interrupts its normal processing to issue one DRAM refresh command (REF). When it does that, it also optionally completes a short DDR3 DRAM ZQ (external) calibration command (ZQCS) followed by an internal ZQ calibration and internal DLL fine-tuning.

LMC(0)_CONFIG[REF_ZQCS_INT] configures both the REF command interval and the ZQ calibration interval. LMC executes REF commands more frequently than ZQ calibration. LMC executes all of internal and external ZQ calibration and DLL fine-tuning at the same time/frequency (when enabled).

LMC asserts the chip-selects for all ranks asserted in LMC(0)_CONFIG[INIT_STATUS] in sequence when it sends the REF command, which means that all attached parts on all ranks normally get the REF command within a few CKs.

LMC(0)_CONTROL[EXT_ZQCS_DIS] optionally disables the periodic DDR3 ZQCS command. LMC also asserts the chip-selects for all ranks asserted in LMC(0)_CONFIG[INIT_STATUS] when it sends the ZQCS command, which means that all attached parts on all ranks normally get the ZQCS command within a few CKs. In CN71XX systems, all attached DDR3 DRAM must have a separate 240 Ohm external resistor for ZQ calibration.

LMC(0)_CONTROL[INT_ZQCS_DIS] optionally disables periodic internal ZQ calibration. LMC(0)_DLL_CTL3[FINE_TUNE_MODE] enables internal DLL fine-tuning.

LMC does not perform any refreshes or ZQ calibration between a self-refresh entry and exit.

Cavium Confidential For Robert Yang Aug 27/2014

4.6 DRAM Scheduler Performance

The LMC DRAM scheduler aggressively reorders both read and write operations for optimal performance. The oldest references are higher priority, but if bank or other conflicts prevent LMC from scheduling the oldest, then LMC selects a younger reference.

LMC typically prioritizes read operations over write operations. It does, however, prioritize write operations over read operations if one of the following scenarios occur:

- The most recent transaction was a write operation, and there have not been LMC(0)_CONTROL[MAX_WRITE_BATCH] consecutive write operations
- The L2C wants a write to a cache block that is currently being written to memory to complete quickly (and LMC(0)_CONTROL[ELEV_PRIO_DIS] = 0).

Regardless of the priority between read and write operations, LMC still may schedule either read or write operations if bank or other conflicts prevent one type from being scheduled.

The LMC scheduler can differently schedule read/write operations (1) in the same rank, (2) in the same DIMM, but different rank, and (3) in different DIMMs. Refer to LMC(0)_SLOT_CTL0, LMC(0)_SLOT_CTL1, and LMC(0)_SLOT_CTL2 CSRs. LMC can schedule references of the same type (read or write) in the same rank as was last scheduled at highest-priority, followed by remaining references of the same type in the same DIMM, followed by other references of the same type. This can maximize performance.

4.7 DRAM Chip Select, CKE, and ODT Signals

CN71XX has two chip-select signals (DDR__CS<1:0>_L), supporting two ranks of DRAM devices.

LMC's usage of the chip-select signals assigned for a DIMM depends on LMC(0)_CONFIG[RANK_ENA] and LMC(0)_CONTROL[RDIMM_ENA] as outlined in Table 4–1.

Table 4–1 Chip-Select Signals Usage Summary

LMC(0)_CONFIG [RANK_ENA]	LMC(0)_CONTROL [RDIMM_ENA]	Description
0	0	Only the first chip-select of the pair (i.e. DDR_CS0_L) is used. The second chip-select of the pair does not assert.
0	1	Only the first chip-select of the pair (i.e. DDR_CS0_L) is used during normal operation. Both chip-selects assert simultaneously to write the JEDEC standard SSTE32882 control words during a power-up/init sequence.
1	0	Both chip-selects of the pair assert largely independently during normal operation.
1	1	Both chip-selects of the pair assert largely independently during normal operation. Both chip-selects assert simultaneously to write the JEDEC standard SSTE32882 control words during a power-up/init sequence.

Cavium Confidential For Bowen Yang ALU 02/27/2014

CN71XX has two CKE signals (DDR_CKE0/1). The chip-selects and CKE signals must be used together on the same DDR3 DRAM parts as follows:

DDR_CS0_L ↔ DDR_CKE0
DDR_CS1_L ↔ DDR_CKE1

CN71XX has two ODT output signals (DDR_ODT_<1:0>) to drive DDR3-part ODT inputs, one output signal per chip-select signal or DIMM/rank. Figures 4–2, and 4–3 show where it is possible for CN71XX to assert its ODT outputs for isolated read and write transactions.

Table 4–2 shows the CN71XX configuration variables that determine whether each ODT signal asserts for read and write operations to different DIMMs/ranks.

Table 4–2 Read/Write ODT Configuration

Read/Write on Chip Select #	LMC(0)_RODT_MASK Bits	LMC(0)_WODT_MASK Bits
DDR_CS0_L	RODT_D0_R0<3:0>	WODT_D0_R0<3:0>
DDR_CS1_L	RODT_D0_R1<3:0>	WODT_D0_R1<3:0>

4.8 LMC Initialization Sequence

There are eleven parts to the LMC initialization procedure:

1. DDR PLL initialization
2. LMC CK initialization
3. LMC DRESET initialization
4. LMC RESET initialization
5. Early LMC initialization
6. LMC offset training
7. LMC internal V_{ref} training
8. LMC deskew training
9. LMC write leveling
10. LMC read leveling
11. Final LMC initialization

During a cold reset, all eleven initializations should be executed in sequence. The DDR PLL and LMC CK initializations need only be performed once if the SDRAM-clock speed and parameters do not change. Subsequently, it is possible to execute only DRESET and subsequent steps, or to execute only the final three steps.

The remainder of this section covers these initialization steps in sequence.

Cavium Confidential For Bowen Yang ALU 07/21/2014

4.8.1 DDR PLL Initialization

Perform the following eight substeps to initialize the DDR PLL:

1. If not done already, write all fields in LMC(0)_DDR_PLL_CTL and LMC(0)_DLL_CTL2 to their reset values, including:
   - LMC(0)_DDR_PLL_CTL[DDR_DIV_RESET] = 1
   - LMC(0)_DLL_CTL2[DRESET] = 1

   This substep is not necessary after a chip cold reset/power up.
2. If the current DRAM contents are not preserved (see LMC(0)_RESET_CTL[DDR3PSV]), this is also an appropriate time to assert the RESET# pin of the DRAM parts. If desired, write LMC(0)_RESET_CTL[DDR3RST] = 0 without modifying any other LMC(0)_RESET_CTL fields to assert the DDR_RESET_L pin. No action is required here to assert DDR_RESET_L following a chip cold reset/power up. Refer to Section 4.8.4.
3. Without changing any other LMC(0)_DDR_PLL_CTL values, write LMC(0)_DDR_PLL_CTL[CLKF, CLKR] with values that give a desired DDR PLL speed. The LMC(0)_DDR_PLL_CTL[CLKF, CLKR] values should be selected in conjunction with the post-scalar divider values for LMC (LMC(0)_DDR_PLL_CTL[DDR_PS_EN]) so that the desired LMC CK speeds are is produced. Section 4.13 describes LMC(0)_DDR_PLL_CTL[CLKF, CLKR] and LMC(0)_DDR_PLL_CTL[DDR_PS_EN] programmings that produce the desired LMC CK speed. Section 4.8.2 describes LMC CK initialization, which can be done separately from the DDR PLL initialization described in this section.

   The LMC(0)_DDR_PLL_CTL[CLKF, CLKR] values must not change after this point without restarting this SDRAM PLL initialization sequence.
4. Read LMC(0)_DDR_PLL_CTL and wait for the result.
5. Wait a minimum of 3 µs.
6. Write LMC(0)_DDR_PLL_CTL[RESET_N] = 1 without changing any other LMC(0)_DDR_PLL_CTL values.
7. Read LMC(0)_DDR_PLL_CTL and wait for the result.
8. Wait a minimum of 25 µs.

Cavium Confidential For Bowen Yang ALU 02/27/2014

4.8.2 LMC CK Initialization

DDR PLL initialization must be completed prior to starting LMC CK initialization.

Perform the following seven substeps to initialize the LMC CK:

1. Without changing any other LMC(0)_DDR_PLL_CTL values, write LMC(0)_DDR_PLL_CTL[DDR_DIV_RESET] = 1 and LMC(0)_DDR_PLL_CTL[DDR_PS_EN] with the appropriate value to get the desired LMC CK speed. Section Section 4.13 discusses CLKF/CLKR and DDR_PS_EN programmings.

   The LMC(0)_DDR_PLL_CTL[DDR_PS_EN] must not change after this point without restarting this LMC CK initialization sequence.
2. Without changing any other fields in LMC(0)_DDR_PLL_CTL, write LMC(0)_DDR_PLL_CTL[DDR4_MODE] = 0.
3. Read LMC(0)_DDR_PLL_CTL and wait for the result.
4. Wait a minimum of 1 µs.
5. Without changing any other fields in LMC(0)_DDR_PLL_CTL, write LMC(0)_DDR_PLL_CTL[PHY_DCOK] = 1.
6. Read LMC(0)_DDR_PLL_CTL and wait for the result.
7. Wait a minimum of 20 µs.
8. Without changing any other LMC(0)_COMP_CTL2 values, write LMC(0)_COMP_CTL2[CK_CTL,CONTROL_CTL] to the desired DDR_CK_*_P/DDR_DIMM*_CS*_L/DDR_DIMM*_ODT_* drive strength.
9. Read LMC(0)_DDR_PLL_CTL and wait for the result.
10. Wait a minimum of 200 ns.
11. Without changing any other LMC(0)_DDR_PLL_CTL values, write LMC(0)_DDR_PLL_CTL[DDR_DIV_RESET] = 0.
12. Read LMC(0)_DDR_PLL_CTL and wait for the result.
13. Wait a minimum of 200 ns.

4.8.3 LMC DRESET Initialization

Both the DDR PLL and LMC CK initializations must be completed prior to starting this LMC DRESET initialization.

1. If not done already, write LMC(0)_DLL_CTL2 to its reset value, including LMC(0)_DLL_CTL2[DRESET] = 1.
2. Without changing any other LMC(0)_DLL_CTL2 fields, write LMC(0)_DLL_CTL2 [DLL_BRINGUP] = 1.
3. Read LMC(0)_DLL_CTL2 and wait for the result.
4. Wait for a minimum of 10 LMC CK cycles.
5. Without changing any other fields in LMC(0)_DLL_CTL2, write LMC(0)_DLL_CTL2[QUAD_DLL_ENA] = 1.

   LMC(0)_DLL_CTL2[QUAD_DLL_ENA] must not change after this point without restarting the LMC DRESET initialization sequence.
6. Read LMC(0)_DLL_CTL2 and wait for the result.
7. Wait a minimum of 10 µs.

Cavium Confidential For Bowen Kang ALU 02/27/2014

8. Without changing any other fields in LMC(0)_DLL_CTL2, write LMC(0)_DLL_CTL2[DLL_BRINGUP] = 0.

   LMC(0)_DLL_CTL2[DLL_BRINGUP] must not change after this point without restarting the LMC DRESET initialization sequence.

9. Read LMC(0)_DLL_CTL2 and wait for the result.

10. Without changing any other fields in LMC(0)_DLL_CTL2, write LMC(0)_DLL_CTL2[DRESET] = 0.

    LMC(0)_DLL_CTL2[DRESET] must not change after this point without restarting the LMC DRESET initialization sequence.

After completing LMC DRESET initialization, all LMC CSRs may be accessed. Prior to completing LMC DRESET initialization, only LMC(0)_DDR_PLL_CTL, LMC(0)_DLL_CTL2, LMC(0)_RESET_CTL, and LMC(0)_COMP_CTL2 LMC CSRs can be accessed.

4.8.4 LMC RESET Initialization

The purpose of this step is to assert/deassert the RESET# pin at the DRAM parts.

It may be appropriate to skip this step if the DRAM parts are in self refresh and are currently preserving their contents. (Software can determine this via LMC(0)_RESET_CTL[DDR3PSV] in some circumstances.) The remainder of this section assumes that the DRAM contents need not be preserved.

The remainder of this section assumes that the CN71XX DDR_RESET_L pin is attached to the RESET# pin of the attached DRAM parts, as will be appropriate in many systems.

(In other systems, such as ones that can preserve DRAM part contents while CN71XX is powered down, it will not be appropriate to directly attach the CN71XX DDR_RESET_L pin to DRESET# of the DRAM parts, and this section may not apply.)

Perform the following six substeps for LMC reset initialization:

1. If not done already, assert DDR_RESET_L pin by writing LMC(0)_RESET_CTL[DDR3RST] = 0 without modifying any other LMC(0)_RESET_CTL fields.

2. Read LMC(0)_RESET_CTL and wait for the result.

3. Wait until RESET# assertion-time requirement from JEDEC DDR3 specification is satisfied (200 µs during a power-on ramp, 100ns when power is already stable).

4. Deassert DDR_RESET_L pin by writing LMC(0)_RESET_CTL[DDR3RST] = 1 without modifying any other LMC(0)_RESET_CTL fields.

5. Read LMC(0)_RESET_CTL and wait for the result.

6. Wait a minimum of 500µs. This guarantees the necessary T = 500µs delay between DDR_RESET_L deassertion and DDR_CKE* assertion.

Cavium Confidential For Bowen Yang ALU 12/2/2014

4.8.5 Early LMC Initialization

All of DDR PLL, LMC CK, and LMC DRESET initializations must be completed prior to starting this LMC initialization sequence.

Perform the following five substeps for early LMC initialization:

1. Software must ensure there are no pending DRAM transactions.

2. Write LMC(0)_CONFIG, LMC(0)_CONTROL, LMC(0)_TIMING_PARAMS0, LMC(0)_TIMING_PARAMS1, LMC(0)_MODEREG_PARAMS0, LMC(0)_MODEREG_PARAMS1, LMC(0)_DUAL_MEMCFG, LMC(0)_NXM, LMC(0)_WODT_MASK, LMC(0)_RODT_MASK, LMC(0)_COMP_CTL2, LMC(0)_PHY_CTL, LMC(0)_DIMM0/1_PARAMS, and LMC(0)_DIMM_CTL with appropriate values. All sections in this chapter can be used to derive proper register settings.

3. Without changing any other fields in LMC(0)_CONFIG, write LMC(0)_CONFIG[RANKMASK] then write both LMC(0)_SEQ_CTL[SEQ_SEL,INIT_START] = 1 with a single CSR write operation. LMC(0)_CONFIG[RANKMASK] bits should be set to indicate the ranks that will participate in the sequence.

   The LMC(0)_SEQ_CTL[SEQ_SEL] value should select power-up/init or self-refresh exit, depending on whether the DRAM parts are in self-refresh and whether their contents should be preserved. While LMC performs these sequences, it will not perform any other DRAM transactions. When the sequence is complete, hardware sets the LMC(0)_CONFIG[INIT_STATUS] bits for the ranks that have been initialized.

   If power-up/init is selected immediately following a DRESET assertion, LMC executes the sequence described in the “Reset and Initialization Procedure” section of the JEDEC DDR3 specification. This includes activating CKE, writing all four DDR3 mode registers on all selected ranks, and issuing the required ZQCL command. The LMC(0)_CONFIG[RANKMASK] value should select all ranks with attached DRAM in this case. If LMC(0)_CONTROL[RDIMM_ENA] = 1, LMC writes the JEDEC standard SSTE32882 control words selected by LMC(0)_DIMM_CTL[DIMM*_WMASK] between DDR_CKE* signal assertion and the first DRAM mode register write operation. LMC(0)_DIMM_CTL[DIMM*_WMASK] should be cleared to 0 if the corresponding DIMM is not present.

   If self-refresh exit is selected, LMC executes the required SRX command followed by a refresh and ZQ calibration. Section 4.5 describes behavior of a REF + ZQCS. LMC does not write the DDR3 mode registers as part of this sequence, and the mode register parameters must match at self-refresh entry and exit times.

4. Read LMC(0)_SEQ_CTL and wait for LMC(0)_SEQ_CTL[SEQ_COMPLETE] to be set.

5. Read LMC(0)_CONFIG[INIT_STATUS]) and confirm that all ranks have been initialized.

Cavium Confidential For Bowen Yang 02/27/2014

4.8.6 LMC Offset Training

LMC requires input-receiver offset training.

1. Write LMC(0)_PHY_CTL[DAC_ON] = 1.
2. Write LMC(0)_SEQ_CTL[SEQ_SEL] = 0x0B and LMC(0)_SEQ_CTL[INIT_START] = 1.
3. Wait for LMC(0)_SEQ_CTL[SEQ_COMPLETE] to be set to 1.

4.8.7 LMC Internal V_{ref} Training

LMC requires input-reference-voltage training. This sequence should be re-executed if the DC profile during the read operation is adjusted afterward. This includes changes on any of the following registers: LMC(0)_COMP_CTL2[RODT_CTL], LMC(0)_MODREG_PARAMS1[RTT_NOM_*xx*, DIC_*xx*], LMC(0)_RODT_MASK, and LMC(0)_MODEREG_PARAMS2[RTT_PARK_*xx*].

1. Write LMC(0)_EXT_CONFIG[VREFINT_SEQ_DESKEW] = 0.
2. Write LMC(0)_SEQ_CTL[SEQ_SEL] = 0x0a and LMC(0)_SEQ_CTL[INIT_START = 1.
3. Wait for LMC(0)_SEQ_CTL[SEQ_COMPLETE] to be set to 1.

4.8.8 LMC Deskew Training

LMC requires input-read-data deskew training. This sequence should be re-executed if the previous sequence is rerun.

1. Write LMC(0)_EXT_CONFIG[VREFINT_SEQ_DESKEW] = 1.
2. Write LMC(0)_SEQ_CTL[SEQ_SEL] = 0x0A and LMC(0)_SEQ_CTL[INIT_START = 1.
3. Wait for LMC(0)_SEQ_CTL[SEQ_COMPLETE] to be set to 1.

4.8.9 LMC Write Leveling

LMC supports an automatic write leveling like that described in the JEDEC DDR3 specifications separately per byte-lane.

All of DDR PLL, LMC CK, LMC DRESET, and early LMC initializations must be completed prior to starting this LMC write-leveling sequence.

There are many possible procedures that will write-level all the attached DRAM parts. One possibility is for software to simply write the desired values into LMC(0)_WLEVEL_RANK(0..3). This section describes one possible sequence that uses LMC's autowrite-leveling capabilities.

1. If overall DQS/DQ delays, including on board and on DRAM module if any, to the first DRAM on the fly-by chain may be more than the CK trace delays to the first DRAM on the fly-by chain, then ensure that LMC(0)_CONFIG[EARLY_DQX] is set at this point.

Do the remaining steps 2–7 separately for each rank *i* with attached DRAM.

2. Write LMC(0)_WLEVEL_RANK*i* = 0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

3. Without changing any other fields in LMC(0)_WLEVEL_CTL, write LMC(0)_WLEVEL_CTL[LANEMASK] to select the byte lane(s) with attached DRAM(s).

4. Without changing any other fields in LMC(0)_CONFIG,
   - write LMC(0)_SEQ_CTL[SEQ_SEL] to select write-leveling
   - write LMC(0)_CONFIG[RANKMASK] = (1 << *i*)
   - write LMC(0)_SEQ_CTL[INIT_START] = 1

   LMC will initiate write-leveling at this point. Assuming LMC(0)_WLEVEL_CTL [SSET] = 0, LMC first enables write-leveling on the selected DRAM rank via a MR1 write, then sequences through and accumulates write-leveling results for eight different delay settings twice, starting at a delay of zero in this case since LMC(0)_WLEVEL_RANK*i*[BYTE*<4:3>] = 0, increasing by 1/8 CK each setting, covering a total distance of one CK, then disables the write-leveling via another MR1 write.

   After the sequence through 16 delay settings is complete:
   - LMC sets LMC(0)_WLEVEL_RANK*i*[STATUS] = 3
   - LMC sets LMC(0)_WLEVEL_RANK*i*[BYTE*<2:0>] (for all ranks selected by LMC(0)_WLEVEL_CTL[LANEMASK]) to indicate the first write leveling result of 1 that followed result of 0 during the sequence, except that the LMC always writes LMC(0)_WLEVEL_RANK*i*[BYTE*<0>]=0.
   - Software can read the eight write-leveling results from the first pass through the delay settings by reading LMC(0)_WLEVEL_DBG[BITMASK] (after writing LMC(0)_WLEVEL_DBG[BYTE]). (LMC does not retain the write-leveling results from the second pass through the eight delay settings. They should often be identical to the LMC(0)_WLEVEL_DBG[BITMASK] results, though.)

5. Wait until LMC(0)_WLEVEL_RANK*i*[STATUS] ≠ 2.

   LMC will have updated LMC(0)_WLEVEL_RANK*i*[BYTE*<2:0>] for all byte lanes selected by LMC(0)_WLEVEL_CTL[LANEMASK] at this point. LMC(0)_WLEVEL_RANK*i*[BYTE*<4:3>] will still be the value that software wrote in substep 2 above, which is 0.

6. Calculate LMC(0)_WLEVEL_RANK*i*[BYTE*<4:3>] settings for all byte lanes on all ranks with attached DRAM.

   At this point, all byte lanes on rank *i* with attached DRAM should have been write-leveled, and LMC(0)_WLEVEL_RANK*i*[BYTE*<2:0>] has the result for each byte lane.

   But note that the write-leveling sequence will only determine the delay modulo the CK cycle time, and cannot determine how many additional CK cycles of delay are present. Software must calculate the number of CK cycles, or equivalently, the LMC(0)_WLEVEL_RANK*i*[BYTE*<4:3>] settings.

   This BYTE*<4:3> calculation is system/board specific.

   Many techniques can be used to calculate write-leveling BYTE*<4:3> values, including:
   - Known values for some byte lanes.
   - Relative values for some byte lanes relative to others.

Cavium Confidential For Edward Yang ALI 02/27/2014

For example, suppose lane *X* is likely to require a larger write-leveling delay than lane *Y*. A BYTE*X*<2:0> value that is much smaller than the BYTE*Y*<2:0> value may then indicate that the required lane *X* delay wrapped into the next CK, so BYTE*X*<4:3> should be set to BYTE*Y*<4:3>+1.

When ECC DRAM is not present (i.e. when DRAM is not attached to the DDR_CBS_0_* and DDR_CB<7:0> chip signals, or the DDR_DQS_<4>_* and DDR_DQ<35:32> chip signals), write LMC(0)_WLEVEL_RANK*[BYTE8] = LMC(0)_WLEVEL_RANK*[BYTE0], using the final calculated BYTE0 value. Write LMC(0)_WLEVEL_RANK*[BYTE4] = LMC(0)_WLEVEL_RANK*[BYTE0], using the final calculated BYTE0 value.

7. Initialize LMC(0)_WLEVEL_RANK* values for all unused ranks.

Let rank *i* be a rank with attached DRAM.

For all ranks *j* that do not have attached DRAM, set LMC(0)_WLEVEL_RANK*j* = LMC(0)_WLEVEL_RANK*i*.

4.8.10 LMC Read Leveling

LMC supports an automatic read-leveling separately per byte-lane using the DRAM multipurpose register predefined pattern for system calibration defined in the JEDEC DDR3 specifications.

All of DDR PLL, LMC CK, and LMC DRESET, and early LMC initializations must be completed prior to starting this LMC read-leveling sequence.

Software could simply write the desired read-leveling values into LMC(0)_RLEVEL_RANK(0..3). This section describes a sequence that uses LMC's autoread-leveling capabilities.

When LMC does the read-leveling sequence for a rank, it first enables the DRAM multipurpose register predefined pattern for system calibration on the selected DRAM rank via a MR3 write, then executes 64 RD operations at different internal delay settings, then disables the predefined pattern via another MR3 write operation. LMC determines the pass or fail of each of the 64 settings independently for each byte lane, then writes appropriate LMC(0)_RLEVEL_RANK(0..3)[BYTE*] values for the rank.

After read-leveling for a rank, software can read the 64 pass/fail indications for one byte lane via LMC(0)_RLEVEL_DBG[BITMASK]. Software can observe all pass/fail results for all byte lanes in a rank via separate read-leveling sequences on the rank with different LMC(0)_RLEVEL_CTL[BYTE] values.

The 64 pass/fail results will typically have failures for the low delays, followed by a run of some passing settings, followed by more failures in the remaining high delays. LMC sets LMC(0)_RLEVEL_RANK(0..3)[BYTE*] to one of the passing settings. First, LMC selects the longest run of successes in the 64 results. (In the unlikely event that there is more than one longest run, LMC selects the first one.) Then if LMC(0)_RLEVEL_CTL[OFFSET_EN] = 1 and the selected run has more than LMC(0)_RLEVEL_CTL[OFFSET] successes, LMC selects the last passing setting in the run minus LMC(0)_RLEVEL_CTL[OFFSET]. Otherwise LMC selects the middle setting in the run (rounding earlier when necessary). We expect the read-leveling sequence to produce good results with the reset values LMC(0)_RLEVEL_CTL [OFFSET_EN]=1, LMC(0)_RLEVEL_CTL[OFFSET] = 2.

The read-leveling sequence has the following steps:

Cavium Confidential For Bowen Yang Aug 27/2015

1. Select desired LMC(0)_RLEVEL_CTL[OFFSET_EN,OFFSET,BYTE] settings.

Do the remaining substeps 2–4 separately for each rank *i* with attached DRAM.

2. Without changing any other fields in LMC(0)_CONFIG,
   - write LMC(0)_SEQ_CTL[SEQ_SEL] to select read-leveling
   - write LMC(0)_CONFIG[RANKMASK] = (1 << *i*)
   - write LMC(0)_SEQ_CTL[INIT_START] = 1

   This initiates the previously-described read-leveling.

3. Wait until LMC(0)_RLEVEL_RANK*i*[STATUS] ≠ 2

   LMC will have updated LMC(0)_RLEVEL_RANK*i*[BYTE*] for all byte lanes at this point.

   If ECC DRAM is not present (i.e. when DRAM is not attached to the DDR_CBS_0_* and DDR_CB<7:0> chip signals, or the DDR_DQS_<4>_* and DDR_DQ<35:32> chip signals), write LMC(0)_RLEVEL_RANK*[BYTE8] = LMC(0)_RLEVEL_RANK*[BYTE0]. Write LMC(0)_RLEVEL_RANK*[BYTE4] = LMC(0)_RLEVEL_RANK*[BYTE0].

4. If desired, consult LMC(0)_RLEVEL_DBG[BITMASK] and compare to LMC(0)_RLEVEL_RANK*i*[BYTE*] for the lane selected by LMC(0)_RLEVEL_CTL[BYTE]. If desired, modify LMC(0)_RLEVEL_CTL[BYTE] to a new value and repeat so that all BITMASKs can be observed.

5. Initialize LMC(0)_RLEVEL_RANK* values for all unused ranks.

   Let rank *i* be a rank with attached DRAM.

   For all ranks *j* that do not have attached DRAM, set LMC(0)_RLEVEL_RANK*j* = LMC(0)_RLEVEL_RANK*i*.

This read-leveling sequence can help select the proper CN71XX ODT resistance value (LMC(0)_COMP_CTL2[RODT_CTL]). A hardware-generated LMC(0)_RLEVEL_RANK*i*[BYTE*j*] value (for a used byte lane *j*) that is drastically different from a neighboring LMC(0)_RLEVEL_RANK*i*[BYTE*k*] (for a used byte lane *k*) can indicate that the CN71XX ODT value is not optimal. It is possible to simultaneously optimize both LMC(0)_COMP_CTL2[RODT_CTL] and LMC(0)_RLEVEL_RANK*n*[BYTE*] values by performing this read-leveling sequence for several LMC(0)_COMP_CTL2[RODT_CTL] values and selecting the one with the best LMC(0)_RLEVEL_RANK*n*[BYTE*] profile for the ranks.

NOTE: The optimal ODT settings for LMC(0)_COMP_CTL2[RODT_CTL], LMC(0)_MODREG_PARAMS1[RTT_NOM_*xx*], and LMC(0)_RODT_MASK should be determined by board analysis, simulation, and testing.

Cavium Confidential Pdf
Bowen Yang
2017-xx-xx

4.8.11 Final LMC Initialization

Early LMC initialization, LMC write-leveling, and LMC read-leveling must be completed prior to starting this final LMC initialization.

LMC hardware updates the LMC(0)_SLOT_CTL0, LMC(0)_SLOT_CTL1, LMC(0)_SLOT_CTL2 CSRs with minimum values based on the selected read-leveling and write-leveling settings. Software should not write the final LMC(0)_SLOT_CTL0, LMC(0)_SLOT_CTL1, and LMC(0)_SLOT_CTL2 values until after the final read-leveling and write-leveling settings are written.

Software must ensure the LMC(0)_SLOT_CTL0, LMC(0)_SLOT_CTL1, and LMC(0)_SLOT_CTL2 CSR values are appropriate for this step. These CSRs select the minimum gaps between read operations and write operations of various types.

Software must not reduce the values in these CSR fields below the values previously selected by the LMC hardware (during write-leveling and read-leveling steps above).

All sections in this chapter may be used to derive proper settings for these registers.

For minimal read latency, L2C_CTL[EF_ENA,EF_CNT] should be programmed properly. This should be done prior to the first read.

Cavium Confidential Eor
Bowen Yang
ALU
02/27/2014

4.9 DRAM ECC Codes

LMC(0)_CONFIG[ECC_ENA] enables ECC protection (and necessitates that all 9 lanes be populated). LMC(0)_CONFIG[ECC_ADR] enables the ECC address bus protection feature.

Table 4–3 shows the LMC 64-bit ECC code for core 64-bit references.

Table 4–3 DRAM 64-Bit ECC Code

cccccccc 7:0	Data Bits 63:0																
	63:60	59:56	55:52	51:48	47:44	43:40	39:36	35:32	31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0	
00000001	1011	0100	1101	0001	1011	0100	1101	0001	0100	1011	0010	1110	0100	1011	0010	1110	0XB4D1B4D144BE4B2E
00000010	0001	0101	0101	0111	0001	0101	0101	0111	0001	0101	0101	0111	0001	0101	0101	0111	0x1557155711571557
00000100	1010	0110	1001	1001	1010	0110	1001	1001	1010	0110	1001	1001	1010	0110	1001	1001	0XA699A699AA69A699
00001000	0011	1000	1110	0011	0011	1000	1110	0011	0011	1000	1110	0011	0011	1000	1110	0011	0X38E338E3338338E3
00010000	1100	0000	1111	1100	1100	0000	1111	1100	1100	0000	1111	1100	1100	0000	1111	1100	0XC0FCC0FCCC0CC0FC
00100000	1111	1111	0000	0000	1111	1111	0000	0000	1111	1111	0000	0000	1111	1111	0000	0000	0XFF00FF00FFF0FF00
01000000	1111	1111	0000	0000	0000	0000	1111	1111	1111	1111	0000	0000	0000	0000	1111	1111	0XFF0000FFFFF000FF
10000000	0000	0000	1111	1111	1111	1111	0000	0000	1111	1111	0000	0000	0000	0000	1111	1111	0X00FFFF00FFF000FF
00000000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	
xxxxxxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	
84210000	7766	6666	9999	9988	BBAA	AAAA	5555	5544	FFEE	EEEE	1111	1100	3322	2222	DDDD	DDCC	← Syndromes
00008421	50DB	8742	DB87	42AF	50DB	8742	DB87	42AF	41CA	9653	CA96	53BE	41CA	9653	CA96	53BE	

Table 4–4 shows the DRAM Extended 64-bit ECC Code. Table 4–5 shows the mapping from address to Gray Code.

Table 4–4 DRAM Extended 64-bit ECC Code (LMC(0)_CONFIG[ECC_ADR]=1)

GrayCode Burst 3:0		Address 34:7							
1101	0xD	0010	1010	1010	1110	0101	0101	0101	0x2AAE555
1101	0xD	1101	1101	0001	0110	1011	1010	0010	0xDD16BA2
1110	0xC	1001	0101	0101	1011	0010	1010	1011	0x955B2AB
0110	0x6	1110	0101	0110	0011	1100	1010	1100	0xE563CAC
0110	0x6	1111	1001	1000	0011	1111	0011	0000	0xF983F30
0110	0x6	1111	1110	0000	0011	1111	1100	0000	0xFE03FC0
0001	0x1	0000	0000	0000	0011	1111	1111	1111	0x0003FFF
0110	0x6	1111	1111	1111	1100	0000	0000	0000	0x7FFC000
0000		0000	0000	0000	0000	0000	0000	0000	
xxxx		xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	
0BB4		3BBB	BAA9	9888	8877	7777	6655	4444	← Syndromes
7FC3		EA96	3E1E	1C96	53FC	A963	E1E1	C965	

Table 4–5 Address to Gray Code Mapping

Address<6:3>	GrayCode<3:0>	Address<6:3>	GrayCode<3:0>
0x0	0x0	0x8	0xC
0x1	0x1	0x9	0xD
0x2	0x3	0xA	0xF
0x3	0x2	0xB	0xE
0x4	0x6	0xC	0xA
0x5	0x7	0xD	0xB
0x6	0x5	0xE	0x9
0x7	0x4	0xF	0x8

Cavium Confidential Bowen Yang ATG 02/27/2014 EDR

4.10 Dynamic Precharge Power-Down

LMC can dynamically put the DDR3 DRAM parts into precharge power-down mode when enabled and the DRAM interface is idle.

When LMC(0)_CONFIG[IDLEPOWER] ≠ 0, dynamic power-down is enabled. If dynamic powerdown is enabled and the LMC has been idle for:

$2^{(2+\text{IDLEPOWER})}$ CKs,

then LMC automatically initiates a precharge power-down entry SEQUENCE that causes the DDR parts to enter power-down mode by deasserting DDR_CKE*. After entering dynamic power-down, LMC keeps the DRAM parts in power-down as long as it is idle. This can save system power when the CN71XX DDR interface is not used.

When LMC needs to issue a REF/ZQCS to the DRAM devices while they are in dynamic power-down, LMC automatically initiates a precharge power-down exit SEQUENCE that takes the DRAM parts out of power-down. After initiating the required REF/ZQCS, LMC also automatically initiates a precharge power-down entry SEQUENCE that puts the DRAM parts back into power-down as quickly as allowed. Refer to Section 4.5 above.

LMC(0)_MODEREG_PARAMS0[PPD] determines whether the DRAM DLL is disabled during the precharge power-down. The DDR3 DRAM parts save more power when the DLL is disabled, but the exit SEQUENCE takes longer.

Refer to LMC(0)_CONFIG[SEQUENCE] CSR description.

When a new DRAM load/store request arrives (from L2C) during dynamic powerdown, LMC automatically initiates a precharge power-down exit SEQUENCE that takes the DRAM parts out of power-down and returns to normal operation. This costs an additional TXP or TXPDLL CKs beyond what it would when LMC is idle, but not in dynamic power-down.

4.11 Non-Homogeneous Ranks

The DDR3 DRAM attached to different ranks must use many common timing parameters, but the different ranks can have different read-leveling, write-leveling, and, referring to Figure 4–1, ROW_LSB programming. LMC(0)_DUAL_MEMCFG specifies one alternate ROW_LSB and which ranks use the alternate.

Consequently, different DRAM ranks may have different capacities, and there may be holes in the address space. The LMC(0)_NXM CSR description and Section 4.1 above describe the resultant nonexistent addresses and how to configure LMC for them.

4.12 Address Mirroring

LMC supports the address mirroring required by some DIMMs. When LMC(0)_CONFIG[MIRRMASK] bits are set, LMC swizzles address bits <8:3> and bank bits <1:0>. Refer to the LMC(0)_CONFIG[MIRRMASK] CSR description.

Cavium Confidential For Bowen Yang @ 2/27/2014 4:41 U

4.13 CK Speed Programming

The LMC CK speed is determined by the DDR PLL speed and the post-scalar divider.

The DDR PLL multiplies the received PLL_REF_CLK:

$$\text{DDR PLL speed (MHz)} = \frac{(\text{LMC(0)_DDR_PLL_CTL[CLKF]} + 1) \times 50}{\text{LMC(0)_DDR_PLL_CTL[CLKR]} + 1}$$

The DDR PLL speed must reside between 1 GHz and 2.4 GHz.

The LMC CK speed is a divide-down of the DDR PLL output:

$$\text{LMC CK speed (MHz)} = \frac{\text{DDR PLL Speed (MHz)}}{\text{(LMC(0)_DDR_PLL_CTL[DDR_PS_EN] divider)} + n}$$

where n = one of the following values:

n = 1 if LMC(0)_DDR_PLL_CTL[DDR_PS_EN] = 0x0 – 0x7

n = 2 if LMC(0)_DDR_PLL_CTL[DDR_PS_EN] = 0x8

n = 3 if LMC(0)_DDR_PLL_CTL[DDR_PS_EN] = 0x9

Table 4–6 lists some suggested settings:

Table 4–6 Sample Programming Settings

LMC CK Speed / Data Rate	LMC(0)_DDR_PLL_CTL [CLKF]	LMC(0)_DDR_PLL_CTL [CLKR]	LMC(0)_DDR_PLL_CTL [DDR_PS_EN]
333/666MHz	39 (0x27)	0	0x5 (divide by 6)
400/800 MHz	47 (0x2F)	0	0x5 (divide by 6)
533/1066 MHz	63 (0x3F)	2	0x1 (divide by 2)
533/1066 MHz	127 (0x7F)	2	0x3 (divide by 4)
666/1333 MHz	39 (0x27)	0	0x2 (divide by 3)
666/1333 MHz	79 (0x4F)	2	0x1 (divide by 2)

Section 4.8.2 above describes the LMC CK initialization sequence.

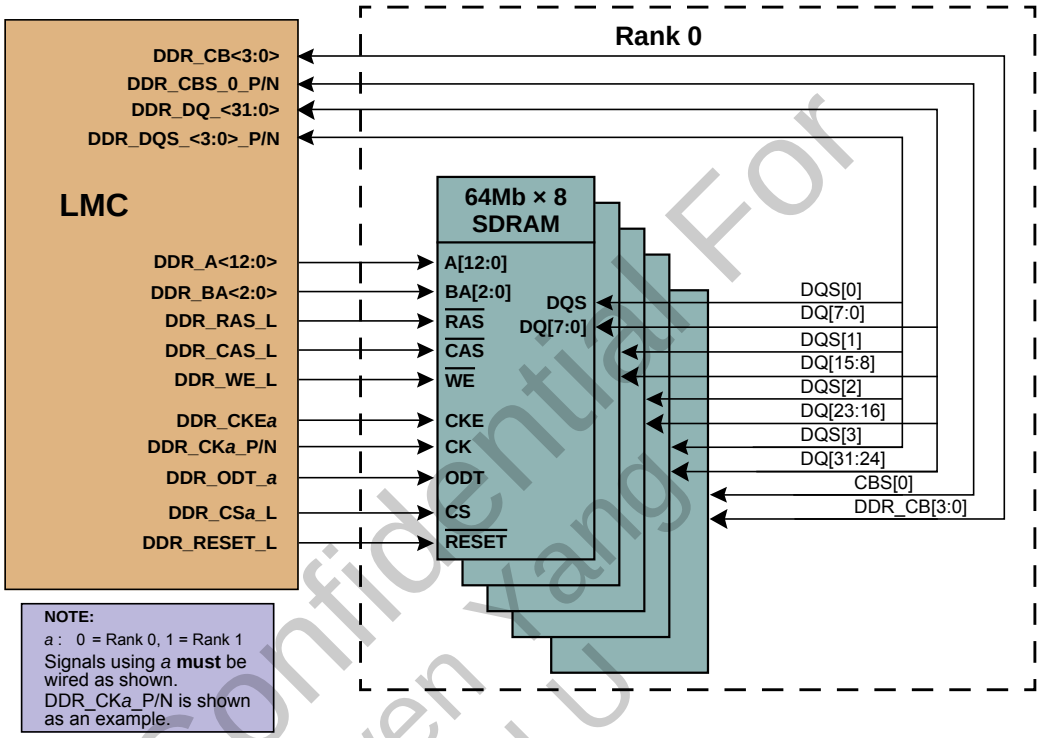
Cavium Confidential For Bowen Yang ALU 02/27/2014

4.14 Example System Configuration

Figure 4–4 shows a system configuration with CN71XX and two dual-ranked DIMMs (only Rank 0 is shown).For a description of the memory signals, see Table 36–3.

Note that this is a typical wiring. There are many other possibilities. For example, boards that plan to either power-down CN71XX or cold reset CN71XX while the DRAM parts are in self-refresh will need to drive DDR3 RESET# from the board rather than with DDR_RESET_L.

Figure 4–4 Memory System Configuration



Cavium Confidential For Bowen Yang ALU 02/27/2014

4.15 Enumerations

4.16 LMC Registers

The LMC registers are listed in Table 4–7.

Table 4–7 LMC Registers

Register	Address	CSR Type[1]	Detailed Description
LMC(0)_FADR	0x0001180088000020 | α << 24	RSL	See page 288
LMC(0)_NXM_FADR	0x0001180088000028 | α << 24	RSL	See page 288
LMC(0)_EXT_CONFIG	0x0001180088000030 | α << 24	RSL	See page 289
LMC(0)_ECC_SYND	0x0001180088000038 | α << 24	RSL	See page 290
LMC(0)_SEQ_CTL	0x0001180088000048 | α << 24	RSL	See page 290
LMC(0)_MODEREG_PARAMS2	0x0001180088000050 | α << 24	RSL	See page 293
LMC(0)_MODEREG_PARAMS3	0x0001180088000058 | α << 24	RSL	See page 294
LMC(0)_TIMING_PARAMS2	0x0001180088000060 | α << 24	RSL	See page 295
LMC(0)_MR_MPR_CTL	0x0001180088000068 | α << 24	RSL	See page 296
LMC(0)_MPR_DATA0	0x0001180088000070 | α << 24	RSL	See page 296
LMC(0)_MPR_DATA1	0x0001180088000078 | α << 24	RSL	See page 297
LMC(0)_MPR_DATA2	0x0001180088000080 | α << 24	RSL	See page 297
LMC(0)_DUAL_MEMCFG	0x0001180088000098 | α << 24	RSL	See page 297
LMC(0)_NXM	0x00011800880000C8 | α << 24	RSL	See page 298
LMC(0)_DIMM(0..1)_DDR4_PARAMS0	0x00011800880000D0 | α << 24 | β << 3	RSL	See page 299
LMC(0)_BIST_CTL	0x0001180088000100 | α << 24	RSL	See page 299
LMC(0)_DIMM(0..1)_DDR4_PARAMS1	0x0001180088000140 | α << 24 | β << 3	RSL	See page 299
LMC(0)_RESET_CTL	0x0001180088000180 | α << 24	RSL	See page 300
LMC(0)_CONFIG	0x0001180088000188 | α << 24	RSL	See page 301
LMC(0)_CONTROL	0x0001180088000190 | α << 24	RSL	See page 305
LMC(0)_TIMING_PARAMS0	0x0001180088000198 | α << 24	RSL	See page 307
LMC(0)_TIMING_PARAMS1	0x00011800880001A0 | α << 24	RSL	See page 308
LMC(0)_MODEREG_PARAMS0	0x00011800880001A8 | α << 24	RSL	See page 311
LMC(0)_WODT_MASK	0x00011800880001B0 | α << 24	RSL	See page 315
LMC(0)_COMP_CTL2	0x00011800880001B8 | α << 24	RSL	See page 316
LMC(0)_DLL_CTL2	0x00011800880001C8 | α << 24	RSL	See page 318
LMC(0)_IFB_CNT	0x00011800880001D0 | α << 24	RSL	See page 318
LMC(0)_OPS_CNT	0x00011800880001D8 | α << 24	RSL	See page 318
LMC(0)_DCLK_CNT	0x00011800880001E0 | α << 24	RSL	See page 318
LMC(0)_INT_EN	0x00011800880001E8 | α <<24	RSL	See page 319
LMC(0)_INT	0x00011800880001F0 | α << 24	RSL	See page 319
LMC(0)_SLOT_CTL0	0x00011800880001F8 | α << 24	RSL	See page 320
LMC(0)_SLOT_CTL1	0x0001180088000200 | α << 24	RSL	See page 322
LMC(0)_SLOT_CTL2	0x0001180088000208 | α << 24	RSL	See page 323
LMC(0)_PHY_CTL	0x0001180088000210 | α << 24	RSL	See page 324
LMC(0)_DLL_CTL3	0x0001180088000218 | α << 24	RSL	See page 325
LMC(0)_DDR_PLL_CTL	0x0001180088000258 | α << 24	RSL	See page 326

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 4–7 LMC Registers (Continued)

Register	Address	CSR Type[1]	Detailed Description
LMC(0)_MODEREG_PARAMS1	0x0001180088000260 | α << 24	RSL	See page 327
LMC(0)_RODT_MASK	0x0001180088000268 | α << 24	RSL	See page 330
LMC(0)_DIMM(0..1)_PARAMS	0x0001180088000270 | α << 24 | β << 3	RSL	See page 332
LMC(0)_RLEVEL_RANK(0..3)	0x0001180088000280 | α << 24 | β << 3	RSL	See page 333
LMC(0)_RLEVEL_CTL	0x00011800880002A0 | α << 24	RSL	See page 334
LMC(0)_RLEVEL_DBG	0x00011800880002A8 | α << 24	RSL	See page 335
LMC(0)_WLEVEL_RANK(0..3)	0x00011800880002B0 | α << 24 | β << 3	RSL	See page 335
LMC(0)_WLEVEL_CTL	0x0001180088000300 | α << 24	RSL	See page 337
LMC(0)_WLEVEL_DBG	0x0001180088000308 | α << 24	RSL	See page 337
LMC(0)_DIMM_CTL	0x0001180088000310 | α << 24	RSL	See page 338
LMC(0)_SCRAMBLE_CFG0	0x0001180088000320 | α << 24	RSL	See page 338
LMC(0)_SCRAMBLE_CFG1	0x0001180088000328 | α << 24	RSL	See page 338
LMC(0)_SCRAMBLED_FADR	0x0001180088000330 | α << 24	RSL	See page 339
LMC(0)_DDR4_DIMM_CTL	0x00011800880003F0 | α << 24	RSL	See page 340

1. RSL-type registers are accessed indirectly across the I/O Interconnect.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

LMC Failing Address Register (SEC/DED) LMC(0)_FADR

This register captures only the first transaction with ECC errors. A DED error can over-write this register with its failing addresses if the first error was a SEC. If you write LMC(0)_CONFIG → SEC_ERR/DED_ERR, it clears the error bits and captures the next failing address. If FDIMM is 1, that means the error is in the high DIMM.

LMC(0)_FADR captures the failing pre-scrambled address location (split into DIMM, bunk, bank, etc.). If scrambling is off, LMC(0)_FADR also captures the failing physical location in the DRAM parts. LMC(0)_SCRAMBLED_FADR captures the actual failing address location in the physical DRAM parts.

- If scrambling is on, LMC(0)_SCRAMBLED_FADR contains the failing physical location in the DRAM parts (split into DIMM, bunk, bank, etc);
- If scrambling is off, the pre-scramble and post-scramble addresses are the same; so the contents of LMC(0)_SCRAMBLED_FADR match the contents of LMC(0)_FADR.

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:40>	—	RAZ	—	—	Reserved
<39:38>	FILL_ORDER	RO/H	0x0	0x0	Fill order for failing transaction.
<37>	FDIMM	RO/H	0	0	Failing DIMM number
<36>	FBUNK	RO/H	0	0	Failing rank number
<35:32>	FBANK	RO/H	0x0	0x0	Failing bank number: bits<3:0>
<31:14>	FROW	RO/H	0x0	0x0	Failing row address; bits<17:0>
<13:0>	FCOL	RO/H	0x0	0x0	Failing column address<13:0>. Technically, represents the address of the 64-bit data that had an ECC error, i.e., FCOL[0] is always 0. Can be used in conjunction with LMC(0)_CONFIG[DED_ERR] to isolate the 64-bit chunk of data in error.

LMC NXM Failing Address Register LMC(0)_NXM_FADR

This register captures only the first transaction with a NXM error while an interrupt is pending, and only captures a subsequent event once the interrupt is cleared by writing a 1 to LMC(0)_INT[NXM_ERR]. It captures the actual L2C-LMC address provided to the LMC that caused the NXM error. A read or write NXM error is captured only if enabled using the NXM event enables. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:39>	—	RAZ	—	—	Reserved
<38>	NXM_SRC	RO/H	0	—	Indicates the source of the operation that caused a NXM error: 0 = L2C, 1 = HFA
<37>	NXM_TYPE	RO/H	0	—	Indicates the type of operation that caused NXM error: 0 = read, 1 = write
<36:0>	NXM_FADDR	RO/H	0x0	—	Failing L2C-LMC address. Bits<3:0> are always 0s for an HFA access, and bits<4:0> are always 0s for an L2C access. Bits<5:4> represent the fill order for an L2C read operation, and the start point within a cache line for a write operation.

Cavium Confidential For Bowen Yang ALU 08/27/2014

LMC Extended Configuration Register LMC(0)_EXT_CONFIG

This register has additional configuration and control bits for the LMC. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:21>	—	RAZ	—	—	Reserved
<20>	VREFINT_SEQ _DESKEW	R/W	0	0	Personality bit to change the operation of what is normally the internal vref training sequence into the deskew training sequence.
<19>	READ_ENA_BPRCH	R/W	0	0	Enable the pad receiver for one cycle longer than normal during read operations.
<18>	READ_ENA_FPRCH	R/W	0	0	Enable the pad receiver starting one cycle earlier than normal during read operations.
<17>	SLOT_CTL_RESET _FORCE	WO	0	0	Write 1 to reset the slot-control override for all slot-control registers. After writing a 1 to this bit, slot-control registers will update with changes made to other timing-control registers. This is a one-shot operation; it automatically returns to 0 after a write to 1.
<16:8>	REF_INT_LSBS	R/W	0x0	0x0	refresh-interval value least-significant bits. The default is 0x0, but it can be set to a non-zero value to get a more precise refresh interval.
<7>	DRIVE_ENA_BPRCH	R/W	0	0	Drive DQx for one cycle longer than normal during write operations.
<6>	DRIVE_ENA_FPRCH	R/W	0	0	Drive DQX starting one cycle earlier than normal during write operations.
<5:4>	DLCRAM_FLIP_SYND	R/W	0x0	0x0	Reserved.
<3>	DLCRAM_COR_DIS	R/W	0	0	Reserved.
<2>	DLC_NXM_RD	R/W	0	0	When set, enable NXM events for HFA read operations.
<1>	L2C_NXM_RD	R/W	0	0	When set, enable NXM events for L2C read operations.
<0>	L2C_NXM_WR	R/W	1	1	When set, enable NXM events for L2C write operations.

Cavium Confidential For Bowen Yang ALD 02/27/2014

LMC MRD ECC Syndromes Register LMC(0)_ECC_SYND

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:24>	MRDSYN3	RO/H	0x0	0x0	MRD ECC Syndrome Quad3. MRDSYN3 corresponds to {DQ[31:0]_c3_p1, DQ[31:0]_c3_p0}.
<23:16>	MRDSYN2	RO/H	0x0	0x0	MRD ECC Syndrome Quad2. MRDSYN2 corresponds to {DQ[31:0]_c2_p1, DQ[31:0]_c2_p0}.
<15:8>	MRDSYN1	RO/H	0x0	0x0	MRD ECC Syndrome Quad1. MRDSYN1 corresponds to {DQ[31:0]_c1_p1, DQ[31:0]_c1_p0}.
<7:0>	MRDSYN0	RO/H	0x0	0x0	MRD ECC Syndrome Quad0. MRDSYN0 corresponds to {DQ[31:0]_c0_p1, DQ[31:0]_c0_p0}.

LMC Sequence Control Register LMC(0)_SEQ_CTL

This register is used to initiate the various control sequences in the LMC. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved.
<5>	SEQ_COMPLETE	RO/H	0	0	Sequence complete. This bit is cleared when INIT_START is set to a 1 and then is set to 1 when the sequence is completed.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<4:1>	SEQ_SEL	R/W	0x0	0x0	Selects the sequence that LMC runs after a 0→1 transition on INIT_START. 0x0 = Power-up/initialization: LMC(0)_CONFIG[RANKMASK] selects participating ranks (should be all ranks with attached DRAM). DDR*_DIMM*_CKE* signals are activated (if not already active). The DRAM registers MR0, MR1, MR2, and MR3 are written in the selected ranks. 0x1 = Read-leveling: LMC(0)_CONFIG[RANKMASK] selects the rank to be read-leveled. MR3 written in the selected rank. 0x2 = Self-refresh entry: LMC(0)_CONFIG[INIT_STATUS] selects the participating ranks (should be all ranks with attached DRAM). MR1 and MR2 are written in the selected ranks if LMC(0)_CONFIG[SREF_WITH_DLL] = 1. DDR*_DIMM*_CKE* signals de-activated. 0x3 = Self-refresh exit: LMC(0)_CONFIG[RANKMASK] must be set to indicate participating ranks (should be all ranks with attached DRAM). DDR*_DIMM*_CKE* signals activated. MR0, MR1, MR2, and MR3 are written in the participating ranks if LMC(0)_CONFIG[SREF_WITH_DLL] = 1. LMC(0)_CONFIG[INIT_STATUS] is updated for ranks that are selected. 0x6 = Write-leveling: LMC(0)_CONFIG[RANKMASK] selects the rank to be write-leveled. LMC(0)_CONFIG[INIT_STATUS] must indicate all ranks with attached DRAM. MR1 and MR2 written in the LMC(0)_CONFIG[INIT_STATUS]-selected ranks. 0x7 = Reserved.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
					(continued) 0x8 = MRW Mode register write sequence. 0x9 = MPR MPR register read or write sequence. 0xA = VREFINT Vref internal training sequence, also used as deskew training sequence when LMC(0)_EXT_CONFIG[VREFINT_SEQ_DESKEW] is set. 0xB = Offset training Offset training sequence. Self-refresh entry SEQ_SELs can also be automatically generated by hardware upon a chip warm- or soft-reset sequence when LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT] are set. LMC writes the LMC(0)_MODEREG_PARAMS0 and LMC(0)_MODEREG_PARAMS1 field values to the mode registers in the DRAM parts (i.e. MR0, MR1, MR2, and MR3) as part of some of these sequences. Refer to the LMC(0)_MODEREG_PARAMS0 and LMC(0)_MODEREG_PARAMS1 descriptions for more details. If there are two consecutive power-up/initializations without a DRESET assertion between them, LMC asserts DDR_DIMM*_CKE as part of the first power-up/initialization, and continues to assert DDR_DIMM*_CKE through the remainder of the first and the second power-up/initialization. If DDR_DIMM*_CKE deactivation and reactivation is needed for a second power-up/initialization, a DRESET assertion is required between the first and the second initializations.
<0>	INIT_START	WO	0	0	A 0→1 transition starts the DDR memory sequence that is selected by SEQ_SEL. This register is a one-shot and clears itself each time it is set.

Cavium Confidential For Bowen Yang ALU 02/27/2014

LMC Mode Register Parameters Register 2
LMC(0)_MODEREG_PARAMS2

These parameters are written into the DDR4 mode registers. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:41>	—	RAZ	—	—	Reserved.
<40>	—	RAZ	—	—	Reserved.
<40>	VREFDQ_TRAIN_EN	R/W	0	0	VREF training mode enable, used for all ranks.
<39>	VREF_RANGE_11	R/W	0	0	VREF range for rank 3.
<38:33>	VREF_VALUE_11	R/W	0x0	0x0	VREF value for rank 3.
<32:30>	RTT_PARK_11	R/W	0x0	0x0	RTT park value for rank 3.
<29>	VREF_RANGE_10	R/W	0	0	VREF range for rank 2.
<28:23>	VREF_VALUE_10	R/W	0x0	0x0	VREF value for rank 2.
<22:20>	RTT_PARK_10	R/W	0x0	0x0	RTT park value for rank 2.
<19>	VREF_RANGE_01	R/W	0	0	VREF range for rank 1.
<18:13>	VREF_VALUE_01	R/W	0x0	0x0	VREF value for rank 1.
<12:10>	RTT_PARK_01	R/W	0x0	0x0	RTT park value for rank 1.
<9>	VREF_RANGE_00	R/W	0	0	VREF range for rank 0.
<8:3>	VREF_VALUE_00	R/W	0x0	0x0	VREF value for rank 0.
<2:0>	RTT_PARK_00	R/W	0x0	0x0	RTT park value for rank 0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

LMC Mode Register Parameters Register 3
LMC(0)_MODEREG_PARAMS3

These parameters are written into the DDR4 mode registers. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:33>	—	RAZ	—	—	Reserved.
<32:31>	MPR_FMT	R/W	0x0	0x0	MPR format.
<30:29>	WR_CMD_LAT	R/W	0x0	0x0	Write command latency when CRC and DM are both enabled.
<28:26>	FGRM	R/W	0x0	0x0	Fine granularity refresh mode.
<25>	TEMP_SENSE	R/W	0	0	Temperature sensor readout enable.
<24>	PDA	R/W	0	0	Per DRAM addressability.
<23>	GD	R/W	0	0	Gear-down mode.
<22>	CRC	R/W	0	0	CRC mode.
<21:20>	LPASR	R/W	0x0	0x0	LP auto self refresh.
<19:17>	TCCD_L	R/W	0x1	0x1	TCCD_L timing parameter: 0x0 = 4 0x3 = 7 0x1 = 5 0x4 = 8 0x2 = 6 0x5–0x7 = reserved.
<16>	RD_DBI	RO	0	0	Read DBI, must be 0.
<15>	WR_DBI	RO	0	0	Write DBI, must be 0.
<14>	DM	R/W	0	0	Data mask enable.
<13>	CA_PAR_PERS	R/W	0	0	Command/address persistent parity error mode.
<12>	ODT_PD	R/W	0	0	ODT in PD mode.
<11:9>	PAR_LAT_MODE	R/W	0x0	0x0	Parity latency mode.
<8>	WR_PREAMBLE	R/W	0	0	Write preamble, 0 = one nCK, 1 = two nCK.
<7>	RD_PREAMBLE	R/W	0	0	Write preamble, 0 = one nCK, 1 = two nCK.
<6>	SRE_ABORT	R/W	0	0	Self refresh abort.
<5:3>	CAL	R/W	0x0	0x0	CS-to-CMD/ADDR latency mode (cycles).
<2>	VREF_MON	R/W	0	0	Internal VREF monitor: 0 = disable, 1 = enable.
<1>	TC_REF	R/W	0	0	Temperature controlled refresh range: 0 = normal, 1 = extended.
<0>	MAX_PD	R/W	0	0	Maximum power-down mode: 0 = disable, 1 = enable.

Cavium Confidential For Bowen Yang AL U 02/7/2014

LMC Timing Parameters Register 2
LMC(0)_TIMING_PARAMS2

This register sets timing parameters for DDR4. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:15>	—	RAZ	—	—	Reserved.
<14:11>	TRTP	R/W	0x5	—	Specifies the T_{RTP} parameter, in cycles. Set this field as follows: RNDUP[T_{RTP}(ns) / T_{CYC}(ns)] – 1, For DDR3, typical = max(4 nCK, 7.5ns). For DDR4 the T_{RTP} parameter is dictated by the T_{WR} MR bits.
<10:7>	T_RW_OP_MAX	R/W	0x7	0x7	Specifies the maximum delay for a read or write operation to complete, used to set the timing of MRW and MPR operations. Set this field as follows: RNDUP[Maximum operation delay (cycles) / 8] Typical = 0x7.
<6:3>	TWTR_L	R/W	0x6	0x6	Specifies T_{WTR_L} constraints. Set this field as follows: RNDUP[T_{WTR_L}(ns) / T_{CYC}(ns)] – 1 where T_{WTR_L} is from the JEDEC DDR4 spec, and T_{CYC}(ns) is the DDR clock frequency (not the data rate). Typical = MAX(4 nCK, 7.5 ns)
<2:0>	TRRD_L	R/W	0x4	0x4	Specifies T_{RRD_L} constraints. Set this field as follows: RNDUP[T_{RRD_L}(ns) / T_{CYC}(ns)] – 1, where T_{RRD_L} is from the JEDEC DDR4 spec, and T_{CYC}(ns) is the DDR clock frequency (not the data rate). Typical = MAX(4 nCK, 7.5 ns) 0x0 = reserved 0x1 = two T_{CYC} 0x2 = three T_{CYC} 0x3 = four T_{CYC} 0x4 = five T_{CYC} 0x5 = six T_{CYC} 0x6 = seven T_{CYC} 0x7 = eight T_{CYC}

Cavium Confidential For Bowen Yang ALU 02/27/2014

LMC MR Write and MPR Control Register LMC(0)_MR_MPR_CTL

This register sets timing parameters for DDR4. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:52>	—	RAZ	—	—	Reserved.
<51>	MPR_WHOLE_ BYTE_ENABLE	R/W	0	0	Reserved.
<50:47>	MPR_BYTE_SELECT	R/W	0x0	0x0	Reserved.
<46:45>	MPR_BIT_SELECT	R/W	0x0	0x0	Select which of four bits to read for each nibble of DRAM data. Typically all four bits from a ×4 device, or all eight bits from a ×8 device, or all 16 bits from a ×16 device carry the same data, but this field allows selection of which device bit will be used to read the MPR data.
<44>	MPR_WR	R/W	0	0	MPR sequence will perform a write operation when set.
<43:42>	MPR_LOC	R/W	0x0	0x0	MPR location select for MPR sequence. Only makes a difference for DDR4.
<41>	MR_WR_PDA_ ENABLE	R/W	0x0	0x0	PDA write enable. When set, MRW operations use PDA, enabled by MR_WR_PDA_MASK per device. Only available for DDR4 devices.
<40:23>	MR_WR_PDA_MASK	R/W	0x0	0x0	PDA mask. If MR_WR_PDA_ENABLE = 1 and there is a 1 in the bit for this mask value, then the corresponding DRAM device is enabled for the PDA MR write operation. Bit<23> corresponds to the lowest order, ×4 device, and bit<40> corresponds to the highest order ×4 device, for a total of up to 18 devices.
<22:21>	MR_WR_RANK	R/W	0x0	0x0	Selects the DRAM rank for either MRW or MPR sequences.
<20:18>	MR_WR_SEL	R/W	0x0	0x0	Selects which MR to write with the MR write sequence. Which pins to drive and how to drive them is automatically controlled through the DDR3/4 mode setting. Bits<19:18> are also used to select the MPR page for an MPR sequence. A value of 0x7 selects an RCW write for both DDR4 and DDR3 MRW operations.
<17:0>	MR_WR_ADDR	R/W	0x0	0x0	Sets a value for A<17:0> for MR write operations. Note that many of these bits must be 0 for various MRs. Bits<7:0> are also used for write data on an MPR sequence write operation.

LMC MR Data Register 0 LMC(0)_MPR_DATA0

This register provides bits <63:0> of MPR data register. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	MPR_DATA	RO	0x0	—	MPR data bits<63:0>. Bits<7:0> represent the MPR data for the lowest-order ×4 device (×4 device 0); bits<15:8> represent ×4 device 1; ..., bits<63:56> are for ×4 device 7.

Cavium Confidential For Bowen Yang 02/2/2014

LMC MR Data Register 1
LMC(0)_MPR_DATA1

This register provides bits <127:64> of MPR data register. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	MPR_DATA	RO	0x0	—	MPR data bits<127:64>. Bits<7:0> represent the MPR data for ×4 device 8; bits<15:8> represent ×4 device 9; ...; bits<63:56> are for ×4 device 15.

LMC MR Data Register 2
LMC(0)_MPR_DATA2

This register provides bits <143:128> of MPR data register. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	—	RAZ	—	—	Reserved.
<63:0>	MPR_DATA	RO	0x0	—	MPR data bits<143:128>. Bits<7:0> represent the MPR data for ×4 device 16; bits<15:8> represent ×4 device 17.

LMC Dual Memory Configuration Register
LMC(0)_DUAL_MEMCFG

This register controls certain parameters of dual-memory configuration. It enables the design to have two, separate memory configurations, selected dynamically by the reference address. Note however, that both configurations share LMC(0)_CONTROL[XOR_BANK], LMC(0)_CONFIG[PBANK_LSB], LMC(0)_CONFIG[RANK_ENA], and all timing parameters.

In this description:

- **config0** refers to the normal memory configuration that is defined by the LMC(0)_CONFIG[ROW_LSB] parameter
- **config1** refers to the dual (or second) memory configuration that is defined by this register.

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:19>	—	RAZ	—	—	Reserved
<18:16>	ROW_LSB	R/W	0x3	—	Encoding used to determine which memory address bit position represents the low order DDR ROW address. Refer to LMC(0)_CONFIG[ROW_LSB].
<15:4>	—	RAZ	—	—	Reserved
<3:0>	CS_MASK	R/W	0x0	—	Chip-select mask. This mask corresponds to the eight chip-select signals for a memory configuration. Each reference address asserts one of the chip-select signals. If that chip-select signal has its corresponding CS_MASK bit set, then the config1 parameters are used, otherwise the config0 parameters are used. CS_MASK<3:2> must be cleared.

Cavium Confidential For Bowen Yang ALC 02/07/2014

LMC Nonexistent Memory Register LMC(0)_NXM

Following is the decoding for mem_msb/rank:

0000: mem_msb = mem_adr[25]
0001: mem_msb = mem_adr[26]
0010: mem_msb = mem_adr[27]
0011: mem_msb = mem_adr[28]
0100: mem_msb = mem_adr[29]
0101: mem_msb = mem_adr[30]
0110: mem_msb = mem_adr[31]
0111: mem_msb = mem_adr[32]
1000: mem_msb = mem_adr[33]
1001: mem_msb = mem_adr[34]
1010: mem_msb = mem_adr[35]
1011: mem_msb = mem_adr[36]
1100–1111 = reserved

For example, for a DIMM made of Samsung's K4B1G0846C-ZCF7 1Gb (16M × 8 bit × 8 bank) DDR3 parts, the column address width = 10; so with 10b of col, 3b of bus, 3b of bank, row_lsb = 16. Therefore, row = mem_adr[29:16] and mem_msb = 4.

Note also that addresses greater than the max defined space (pbank_msb) are also treated as NXM accesses.

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:24>	—	RAZ	—	—	Reserved.
<23:20>	MEM_MSB_D1_R1	R/W	0x0	—	Max row MSB for DIMM1, RANK1/DIMM1 in single ranked
<19:16>	MEM_MSB_D1_R0	R/W	0x0	—	Max row MSB for DIMM1, RANK0
<15:12>	MEM_MSB_D0_R1	R/W	0x0	—	Max row MSB for DIMM0, RANK1/DIMM0 in single ranked
<11:8>	MEM_MSB_D0_R0	R/W	0x0	—	Max Row MSB for DIMM0, RANK0
<7:4>	—	RAZ	—	—	Reserved.
<3:0>	CS_MASK	R/W	0x0	—	Chip select mask. Tis mask corresponds to the four chip selects for a memory configuration. If LMC(0..0)_CONFIG[RANK_ENA]=0, then this mask must be set in pairs because each reference address will assert a pair of chip selects. If the chip select(s) have a corresponding CS_MASK bit set, then the reference is to nonexistent memory (NXM). LMC will alias a NXM read reference to use the lowest, legal chip-select signal and return 0s for data. LMC normally discards NXM write operations, but will also alias them when LMC(0)_CONTROL[NXM_WRITE_EN] = 1.

Cavium Confidential For Bowen Yang 02/7/2014

LMC DIMM Parameters Registers 0
LMC(0)_DIMM(0..1)_DDR4_PARAMS0

This register contains values to be programmed into the extra DDR4 control words in the corresponding (registered) DIMM. These are control words RC1x through RC8x. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:56>	RC8X	R/W	0x0	0x0	RC8x.
<55:48>	RC7X	R/W	0x0	0x0	RC7x.
<47:40>	RC6X	R/W	0x0	0x0	RC6x.
<39:32>	RC5X	R/W	0x0	0x0	RC5x.
<31:24>	RC4X	R/W	0x0	0x0	RC4x.
<23:16>	RC3X	R/W	0x0	0x0	RC3x.
<15:8>	RC2X	R/W	0x0	0x0	RC2x.
<7:0>	RC1X	R/W	0x0	0x0	RC1x.

LMC BIST Control Registers
LMC(0)_BIST_CTL

This register has fields to control BIST operation. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved.
<3>	DLCRAM_BIST_STATUS	RO/H	0	0	Reserved.
<2>	DLCRAM_BIST_DONE	RO/H	0	—	Reserved.
<1>	START_BIST	R/W	0	0	Reserved.
<0>	CLEAR_BIST	R/W	0	0	Reserved.

LMC DIMM Parameters Registers 1
LMC(0)_DIMM(0..1)_DDR4_PARAMS1

This register contains values to be programmed into the extra DDR4 control words in the corresponding (registered) DIMM. These are control words RC9x through RCBx. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:24>	—	RAZ	—	—	Reserved.
<23:16>	RCBX	R/W	0x0	0x0	RCBx.
<15:8>	RCAX	R/W	0x0	0x0	RCAx.
<7:0>	RC9X	R/W	0x0	0x0	RC9x.

Cavium Confidential For Bowen Yang ALU 02/27/2014

LMC Reset Control Register
LMC(0)_RESET_CTL

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	DDR3PSV	R/W	0	0	Set by hardware to indicate DRAM contents were preserved during warm or soft reset. May be useful for system software to determine when the DDR3 contents have been preserved. Cleared by hardware during a cold reset. Never cleared by hardware during a warm/soft reset. Set by hardware during a warm/soft reset if the hardware automatically put the DDR3 DRAM into self-refresh during the reset sequence. Can also be written by software (to any value).
<2>	DDR3PSOFT	R/W	0	0	Set to enable preserve mode during soft reset. Enables preserve mode during a soft reset. When set, the DDR3 controller hardware automatically puts the attached DDR3 DRAM parts into self-refresh at the beginning of a soft reset sequence (see LMC(0)_SEQ_CTL[SEQ_SEL]), provided that the DDR3 controller is up. When clear, the DDR3 controller hardware does not put the attached DDR3 DRAM parts into self-refresh during a soft reset sequence. DDR3PSOFT is cleared on a cold reset. Warm and soft chip resets do not affect the DDR3PSOFT value. Outside of cold reset, only software CSR write operations change the DDR3PSOFT value.
<1>	DDR3PWARM	R/W	0	0	Set to enable preserve mode during warm reset. Enables preserve mode during a warm reset. When set, the DDR3 controller hardware automatically puts the attached DDR3 DRAM parts into self-refresh at the beginning of a warm reset sequence (see LMC(0)_SEQ_CTL[SEQ_SEL]), provided that the DDR3 controller is up. When clear, the DDR3 controller hardware does not put the attached DDR3 DRAM parts into self-refresh during a warm reset sequence. DDR3PWARM is cleared on a cold reset. Warm and soft chip resets do not affect the DDR3PWARM value. Outside of cold reset, only software CSR write operations change the DDR3PWARM value. Note that if a warm reset follows a soft reset, DDR3PWARM has no effect, as the DDR3 controller is no longer up after any cold/warm/soft reset sequence.
<0>	DDR3RST	R/W	0	0	Memory Reset. 0 = Reset asserted; 1 = Reset deasserted. DDR3 DRAM parts have a new RESET# pin that was not present in DDR2 parts. This field controls the assertion of the CN71XX pin that attaches to RESET#. When set, CN71XX asserts RESET#. When clear, CN71XX deasserts RESET#. DDR3RST is set on a cold reset. Warm and soft chip resets do not affect this value. Outside of cold reset, only software CSR write operations change the DDR3RST value.

Cavium Confidential For Bowyang AL U 02/27/2014

LMC Memory Configuration Register LMC(0)_CONFIG

This register controls certain parameters required for memory configuration. Note the following:

- Priority order for hardware write operations to LMC(0)_CONFIG/LMC(0)_FADR/LMC(0)_SCRAMBLED_FADR/LMC(0)_ECC_SYND: DED error ≥ NXM error > SEC error.
- The self-refresh entry sequence(s) power the DLL up/down (depending on LMC(0)_MODEREG_PARAMS0[DLL]) when LMC(0)_CONFIG[SREF_WITH_DLL] is set.
- Prior to the self-refresh exit sequence, LMC(0)_MODEREG_PARAMS0 should be reprogrammed (if needed) to the appropriate values.

See Section 4.8 for the LMC bring-up sequence. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	—	RAZ	—	—	Reserved
<62>	BG2_ENABLE	R/W	1	—	BG2 pin enable for DDR4 mode. Only is active when LMC(0)_DDR_PLL_CTL[DDR4_MODE] = 1. Typically cleared only for DDR4 ×16 devices, where there is no BG2 pin on the device.
<61>	MODE_X4DEV	RO	0	—	×4 device mode. Always reads as 0 for CN71XX devices, which does not have ×4 device support.
<60>	MODE32b	RO	1	1	Thirty-two-bit datapath mode. Always set to 1, only 32b mode is supported.
<59>	SCRZ	R/W1S/H	0	—	When set, hide LMC*_SCRAMBLE_CFG0 and LMC*_SCRAMBLE_CFG1.
<58>	EARLY_ UNLOAD_ D1_R1	R/W	0	—	When set, unload the PHY silo one cycle early for rank 3 read requests. The recommended EARLY_UNLOAD_D1_R1 value is 0.
<57>	EARLY_ UNLOAD_ D1_R0	R/W	0	—	When set, unload the PHY silo one cycle early for rank 2 read requests. The recommended EARLY_UNLOAD_D1_RO value is 0.
<56>	EARLY_ UNLOAD_ D0_R1	R/W	0	—	When set, unload the PHY silo one cycle early for rank 1 read requests. The recommended EARLY_UNLOAD_D0_R1 value is 0.
<55>	EARLY_ UNLOAD_ D0_R0	R/W	0	—	When set, unload the PHY silo one cycle early for rank 0 read requests. The recommended EARLY_UNLOAD_D0_R0 value is 0.
<54:51>	INIT_STATUS	R/W1	0x0	—	Indicates status of initialization. INIT_STATUS[n] = 1 implies rank n has been initialized. Software must set necessary RANKMASK bits before executing the initialization sequence using LMC(0)_SEQ_CTL. If the rank has been selected for initialization with the RANKMASK bits, the INIT_STATUS bits are set after successful initialization and after self-refresh exit. INIT_STATUS determines the chip-select signals that assert during refresh, ZQCS, precharge power-down entry/exit, and self-refresh entry SEQ_SELs.

Cavium Confidential For Bowen Yang ALD 02/17/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<50:47>	MIRRMASK	R/W	0x0	—	Mask determining which ranks are address-mirrored. MIRRMASK<*n*> = 1 means rank *n* addresses are mirrored for $0 \le n \le 3$. MIRRMASK<3:2> must be 0. When RANK_ENA = 0, MIRRMASK<1> MBZ. A mirrored read/write operation has the following differences: • DDR_BA<1> is swapped with DDR_BA<0>; • DDR_A<8> is swapped with DDR_A<7>; • DDR_A<6> is swapped with DDR_A<5>; • DDR_A<4> is swapped with DDR_A<3>.
<46:43>	RANKMASK	R/W	0x0	—	Mask to select rank to be leveled/initialized. To write-level/read-level/initialize rank *i*, set RANKMASK< *i*> as: **RANK_ENA = 1** / **RANK_ENA = 0** RANKMASK<0> = CS0 / CS0 RANKMASK<1> = CS1 / MBZ RANKMASK<2> = MBZ / MBZ RANKMASK<3> = MBZ / MBZ For read/write leveling, each rank has to be leveled separately, so RANKMASK should only have one bit set. RANKMASK is not used during self-refresh entry/exit and precharge power-down entry/exit instruction sequences. When RANK_ENA = 0, RANKMASK<3:1> must be 0s.
<42>	RANK_ENA	R/W	0	—	RANK enable (for use with dual-rank DIMMs). • For dual-rank DIMMs, RANK_ENA enables the drive of the DDR_CS*_L and DDR_ODT_<1:0> pins differently based on the (PBANK_LSB – 1) address bit. • Write 0 for single-ranked DIMMs.
<41>	SREF_WITH_DLL	R/W	0	0	Self-refresh entry/exit write-mode registers. When set, self-refresh entry sequence writes MR2 and MR1 (in this order, in all ranks), and self-refresh exit sequence writes MR1, MR0, MR2, and MR3 (in this order, for all ranks). The write operations occur before self-refresh entry, and after self-refresh exit. When clear, self-refresh entry and exit instruction sequences do not write any registers in the DDR3/4 parts.
<40>	EARLY_DQX	R/W	0	0	Set this bit to send DQ*x* signals one CK cycle earlier for the case when the shortest DQ*x* lines have a larger delay than the CK line.

Cavium Confidential For Bowen Yang AUG 02/2/2015

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<39:18>	REF_ZQCS_INT	R/W	0x1	—	Refresh & ZQCS intervals. Refresh interval is represented in number of 512 CK cycle increments. ZQCS interval is represented in a number of refresh intervals. A refresh sequence is triggered when bits<24:18> are equal to 0x0, and a ZQCS sequence is triggered when bits<39:18> are equal to 0x0. The ZQCS timer only decrements when the refresh timer is 0. Program bits<24:18> to RND-DN(T_{REFI}/clkPeriod/512). A value of 0 in bits<24:18> will effectively turn off refresh. Program bits<36:25> to (RND-DN(ZQCS_period/refresh_period)–1), where refresh_period is the effective period programmed in bits<24:18>. Note that this value should always be greater than 32, to account for resistor calibration delays. 000_00000000_00000000 = reserved Maximum refresh interval = 127 × 512= 65024 CK cycles Maximum ZQCS interval = 32768 × 127 × 512 = 2,130,706,432 CK cycles. If the refresh interval is programmed to ~8µs, the maximum ZQCS interval is ~262ms, or ~4 ZQCS operations per second. INIT_STATUS determines which ranks receive the REF / ZQCS. LMC does not send any refreshes / ZQCSs when INIT_STATUS = 0.
<17>	RESET	R/W	0	—	Reset one-shot pulse for LMC(0)_OPS_CNT, LMC(0)_IFB_CNT, and LMC(0)_DCLK_CNT. To cause the reset, software writes this to a 1, then rewrites it to a 0.
<16>	ECC_ADR	R/W	0	1	Include memory reference address in the ECC calculation. 0 = disabled, 1 = enabled.
<15:12>	FORCEWRITE	R/W	0x0	0x0	Force the oldest outstanding write to complete after having waited for $2^{FORCEWRITE}$ CK cycles. 0 = disabled.
<11:9>	IDLEPOWER	R/W	0x0	0x6	Enter precharge power-down mode after the memory controller has been idle for $2^{(2+IDLEPOWER)}$ CK cycles. 0 = disabled. **NOTE:** This field should only be programmed after initialization. LMC(0)_MODEREG_PARAMS0[PPD] determines whether the DRAM DLL is disabled during the precharge power-down.
<8:5>	PBANK_LSB	R/W	0x5	—	DIMM address bit select. Reverting to the explanation for ROW_LSB, PBANK_LSB would be: ROW_LSB bit + #rowbits + #rankbits Decoding for PBANK_LSB 0x0: DIMM = mem_adr<28>; if RANK_ENA=1, rank = mem_adr<27> 0x1: DIMM = mem_adr<29>; if RANK_ENA=1, rank = mem_adr<28> 0x2: DIMM = mem_adr<30>; if RANK_ENA=1, rank = mem_adr<29> 0x3: DIMM = mem_adr<31>; if RANK_ENA=1, rank = mem_adr<30> 0x4: DIMM = mem_adr<32>; if RANK_ENA=1, rank = mem_adr<31> 0x5: DIMM = mem_adr<33>; if RANK_ENA=1, rank = mem_adr<32> 0x6: DIMM = mem_adr<34>; if RANK_ENA=1, rank = mem_adr<33> 0x7: DIMM = mem_adr<35>; if RANK_ENA=1, rank = mem_adr<34> 0x8: DIMM = mem_adr<36>; if RANK_ENA=1, rank = mem_adr<35> 0x9: DIMM = 0; if RANK_ENA=1, rank = mem_adr<36> 0xA–0xF: reserved For example, for a DIMM made of Samsung's K4B1G0846C-F7 1Gb (16M × 8 bit × 8 bank) DDR3 parts, the column address width = 10, so with 10b of col, 3b of bus, 3b of bank, ROW_LSB = 16. So, row = mem_adr<29:16>. With RANK_ENA = 0, PBANK_LSB = 2. With RANK_ENA = 1, PBANK_LSB = 3.

Cavium Confidential For Bowei Yang 9/27/2021

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<4:2>	ROW_LSB	R/W	0x3	—	Row address bit select. Encoding used to determine which memory address bit position represents the low order DDR ROW address. The processor's memory address<34:7> needs to be translated to DRAM addresses (bnk,row,col,rank and DIMM) and that is a function of the following: **1.** Datapath width (64) **2.** # banks (8) **3.** # column bits of the memory part—specified indirectly by this register. **4.** # row bits of the memory part—specified indirectly by PBANK_LSB **5.** # ranks in a DIMM—specified by RANK_ENA **6.** # DIMMs in the system by the register below (PBANK_LSB). Col Address starts from mem_addr[2] for 32-bit (4 bytes) DQ width. ROW_LSB is mem_adr[14] for 32-bit mode. Therefore, the ROW_LSB parameter should be set to 0x0 (32b). Decoding for ROW_LSB: **Value** / **Mem address bit that is LSB**: 0x0 <14>; 0x1 <15>; 0x2 <16>; 0x3 <17>; **Value** / **Mem address bit that is LSB**: 0x4 <18>; 0x5 <19>; 0x6 <20>; 0x7 Reserved For example, for a DIMM made of Samsung's K4B1G0846C-F7 1GB (16M × 8 bit × 8 bank) DDR3 parts, the column address width = 10, so with 10b of col, 2b of bus, 3b of bank, ROW_LSB = 14. So, row = mem_adr<28:14>. Refer to Figure 4–1.
<1>	ECC_ENA	R/W	0	1	ECC enable. When set, enables the 8-bit ECC check/correct logic. Should be 1 when used with DIMMs with ECC; 0, otherwise. When this mode is turned on, DQ<71:64> on write operations contains the ECC code generated for the 64 bits of data which will be written in the memory. Later on read operations, will be used to check for single-bit error (which will be auto-corrected) and double-bit error (which will be reported). When not turned on, DQ<71:64> are driven to 0. Please refer to SEC_ERR, DED_ERR, LMC(0)_FADR, and LMC(0)_ECC_SYND for diagnostics information when there is an error.
<0>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang 02-27-2014

LMC Control Register
LMC(0)_CONTROL

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	SCRAMBLE_ENA	R/W	0	0	When set, enables the scramble/descramble logic.
<62:51>	THRCNT	RO/H	0x0	0x0	Fine count.
<50:43>	PERSUB	R/W	0x0	0x0	Offset for HFA rate-matching
<42:39>	THRMAX	R/W	0xF	0x2	Fine rate matching max bucket size. In conjunction with the coarse rate matching logic, the fine rate matching logic gives software the ability to prioritize HFA read requests over L2C write operations. Higher PERSUB values result in a lower HFA read bandwidth. 0x0 = Reserved
<38:34>	CRM_CNT	RO/H	0x0	0x0	Coarse count
<33:29>	CRM_THR	R/W	0x0	0x8	Coarse rate matching threshold
<28:24>	CRM_MAX	R/W	0x1F	0x1F	Coarse rate matching max bucket size. The coarse rate matching logic is used to control the bandwidth allocated to HFA read requests. CRM_MAX is subdivided into two regions with HFA read requests being preferred over LMC read/write operations when CRM_CNT < CRM_THR. CRM_CNT increments by 1 when an HFA read is slotted and by 2 when a LMC read/write is slotted, and rolls over when CRM_MAX is reached. 0x0 = Reserved
<23>	RODT_BPRCH	R/W	0	0	When set, the turn-off time for the ODT pin during a RD command is delayed an additional CK cycle.
<22>	WODT_BPRCH	R/W	0	0	When set, the turn-off time for the ODT pin during a WR command is delayed an additional CK cycle.
<21:20>	BPRCH	R/W	0x0	0x0	Back porch enable. When set, the turn-on time for the default DDR_DQ*/DDR_DQS_*_P/N drivers is delayed an additional BPRCH CK cycles. 00 = 0 CK cycles 01 = 1 CK cycles 10 = 2 CK cycles 11 = 3 CK cycles
<19>	EXT_ZQCS_DIS	R/W	0	0	Disable (external) auto-ZQCS calibration. When clear, LMC runs external ZQ calibration every LMC(0)_CONFIG[REF_ZQCS_INT] CK cycles.
<18>	INT_ZQCS_DIS	R/W	1	0	Disable (internal) auto-ZQCS calibration. When clear, LMC runs internal ZQ calibration every LMC(0)_CONFIG[REF_ZQCS_INT] CK cycles.
<17>	AUTO_DCLKDIS	R/W	0	1	When 1, LMC automatically shuts off its internal clock to conserve power when there is no traffic. Note that this has no effect on the DDR3 PHY and pads clocks.
<16>	XOR_BANK	R/W	0	1	When set to 1: bank[2:0] = address[9:7] ⊕ address[14:12] When clear: bank[2:0] = address[9:7]
<15:12>	MAX_WRITE _BATCH	R/W	0x8	0x8	Maximum number of consecutive write operations to service before forcing read operations to interrupt.

Cavium Confidential For Bowen Yang 12/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<11>	NXM_WRITE_EN	R/W	0	0	NXM write mode. When clear, LMC discards write operations to addresses that don't exist in the DRAM (as defined by LMC(0)_NXM configuration). When set, LMC completes write operations to addresses that don't exist in the DRAM at an aliased address.
<10>	ELEV_PRIO_DIS	R/W	0	0	Disable elevate priority logic. When set, write operations are sent in regardless of priority information from L2C.
<9>	INORDER_WR	R/W	0	0	Send write operations in order (regardless of priority).
<8>	INORDER_RD	R/W	0	0	Send read operations in order (regardless of priority).
<7>	THROTTLE_WR	R/W	0	0	When set, use at most one IFB for write operations.
<6>	THROTTLE_RD	R/W	0	0	When set, use at most one IFB for read operations.
<5:4>	FPRCH2	R/W	0x0	0x1	Front porch enable. When set, the turn-off time for the default DDR_DQ*/DDR_DQS_*_P/N drivers is FPRCH2 CKs earlier. 00 = 0 CK cycles 01 = 1 CK cycles 10 = 2 CK cycles 11 = Reserved
<3>	POCAS	R/W	0	0	Reserved; must be zero.
<2>	DDR2T	R/W	0	1	Turn on the DDR 2T mode. 2 CK-cycle window for CMD and address. This mode helps relieve setup time pressure on the address and command bus which nominally have a very large fanout. Please refer to Micron's tech note tn_47_01 titled *DDR2-533 Memory Design Guide for Two Dimm Unbuffered Systems* for physical details.
<1>	BWCNT	R/W	0	0	Bus utilization counter clear. Clears the LMC(0)_OPS_CNT, LMC(0)_IFB_CNT and LMC(0)_DCLK_CNT registers. To clear the CSRs, software should first write this field to a 1, then write this field to a 0.
<0>	RDIMM_ENA	R/W	0	—	Registered DIMM enable. When set allows the use of JEDEC Registered DIMMs which require address and control bits to be registered in the controller.

Cavium Confidential For Bowen Yang ALC 02/27/2014

LMC Timing Parameters Register 0
LMC(0)_TIMING_PARAMS0

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:44>	TCKSRE	R/W	0x5	0x5	Indicates T_{CKSRE} constraints. Set this field as follows: RNDUP[T_{CKSRE}(ns) / T_{CYC}(ns)] – 1 where T_{CKSRE} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = max(5nCK, 10 ns).
<43:39>	TRP	R/W	0x6	0x6	Indicates T_{RP} constraints. Set {TRP_EXT<0>, TRP<3:0>} as follows: RNDUP[T_{RP}(ns) / T_{CYC}(ns)] + ((RNDUP[T_{RTP}(ns) / T_{CYC}(ns)]) – 4) – 1 where T_{RP} and T_{RTP} are from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP T_{RP} = 10–15ns. TYP T_{RTP} = max(4nCK, 7.5 ns).
<38:35>	TZQINIT	R/W	0x2	0x2	Indicates T_{ZQINIT} constraints. Set this field as follows: RNDUP[T_{ZQINIT}(ns) / (256 × T_{CYC}(ns))] where T_{ZQINIT} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = 2 (equivalent to 512).
<34:31>	TDLLK	R/W	0x2	0x2	Indicates T_{DLLK} constraints. Set this field as follows: RNDUP[T_{DLLK}(ns) / (256 × T_{CYC}(ns))] where T_{DLLK} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = 2 (equivalent to 512). This parameter is used in self-refresh exit and assumed to be greater than T_{RFC}.
<30:26>	TMOD	R/W	0xC	0xC	Indicates tMOD constraints. Set this field as follows: RNDUP[T_{MOD}(ns) / T_{CYC}(ns)] – 1 where T_{MOD} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = max(12nCK, 15 ns).
<25:22>	TMRD	R/W	0x4	0x4	Indicates T_{MRD} constraints. Set this field as follows: RNDUP[T_{MRD}(ns) / T_{CYC}(ns)] – 1 where T_{MRD} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = 4nCK.
<21:16>	TXPR	R/W	0x5	0x5	Indicates T_{XPR} constraints. Set this field as follows: RNDUP[T_{XPR}(ns) / (16 × T_{CYC}(ns))] where T_{XPR} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = max(5nCK, T_{RFC}+10 ns).

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<15:12>	TCKE	R/W	0x3	0x3	Indicates T_{CKE} constraints. Set this field as follows: RNDUP[T_{CKE}(ns) / T_{CYC}(ns)] – 1 where T_{CKE} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = max(3nCK, 7.5/5.625/5.625/5 ns).
<11:8>	TZQCS	R/W	0x4	0x4	Indicates T_{ZQCS} constraints. Set this field as follows: RNDUP[T_{ZQCS}(ns) / (16 × T_{CYC}(ns))] where T_{ZQCS} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = 4 (equivalent to 64).
<7:0>	—	RAZ	—	—	Reserved

LMC Timing Parameters Register 1
LMC(0)_TIMING_PARAMS1

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:49>	—	RAZ	—	—	Reserved
<48:44>	TXPDLL	R/W	0x0	0xA	Indicates T_{XPDLL} constraints. Set this field as follows: RNDUP[T_{XPDLL}(ns) / T_{CYC}(ns)] – 1 where T_{XPDLL} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP=max(10nCK, 24 ns)
<43:39>	TFAW	R/W	0x0	0x9	Indicates T_{FAW} constraints. Set this field as follows: RNDUP[T_{FAW}(ns) / (4 × T_{CYC}(ns))] where T_{FAW} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = 30–40 ns
<38:35>	TWLDQSEN	R/W	0x7	0x7	Indicates $T_{WLDQSEN}$ constraints. Set this field as follows: RNDUP[$T_{WLDQSEN}$(ns) / (4 × T_{CYC}(ns))] where $T_{WLDQSEN}$ is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = max(25nCK)
<34:31>	TWLMRD	R/W	0xA	0xA	Indicates T_{WLMRD} constraints. Set this field as follows: RNDUP[T_{WLMRD}(ns) / (4 × T_{CYC}(ns))] where T_{WLMRD} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = max(40nCK)
<30:28>	TXP	R/W	0x3	0x3	Indicates T_{XP} constraints. Set this field as follows: RNDUP[T_{XP}(ns) / T_{CYC}(ns)] – 1 where T_{XP} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = max(3nCK, 7.5 ns)

Cavium Confidential For Bowei Yang 02/12/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<27:25>	TRRD	R/W	0x2	0x2	Indicates T_{RRD} constraints. Set this field as follows: RNDUP[T_{RRD}(ns) / T_{CYC}(ns)] – 1, where T_{RRD} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = max(4nCK, 10 ns) 0x0 = reserved 0x1 = 3 T_{CYC} ... 0x6 = 7 T_{CYC} 0x7 = 8 T_{CYC} For DDR4, this is the TRRD_S parameter.
<24:18>	TRFC	R/W	0x6	0x7	Indicates T_{RFC} constraints. Set this field as follows: RNDUP[T_{RFC}(ns) / (8 × T_{CYC}(ns))] where T_{RFC} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = 90–350 ns 0x00 = reserved 0x01 = 8 T_{CYC} 0x02 = 16 T_{CYC} 0x03 = 24 T_{CYC} 0x4 = 32 T_{CYC} ... 0x7E = 1008 T_{CYC} 0x7F = 1016 T_{CYC}
<17:14>	TWTR	R/W	0x2	0x3	Indicates T_{WTR} constraints. Set this field as follows: RNDUP[T_{WTR}(ns) / T_{CYC}(ns)] – 1 where T_{WTR} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = max(4nCK, 7.5 ns) 0x0 = reserved 0x1 = 2 ... 0x07 = 8 0x8–0xF = reserved
<13:10>	TRCD	R/W	0x4	0x4	Indicates T_{RCD} constraints. Set this field as follows: RNDUP[T_{RCD}(ns) / T_{CYC}(ns)] where T_{RCD} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = 10–15 ns 0x0 = reserved 0x1 = 2 (2 is the smallest value allowed) 0x2 = 2 0x3 = 3 ... 0xE = 14 0xF = reserved In 2T mode, set this register to TRCD – 1, not going below 2.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<9:4>	TRAS	R/W	0xC	0xC	Indicates T_{RAS} constraints. Set {TRAS_EXT<0>, TRAS<4:0>} (CSR fields) as follows: RNDUP[T_{RAS}(ns)/T_{CYC}(ns)] – 1, where T_{RAS} is from the DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = 35ns – 9 × T_{REFI} 0x0: reserved 0x1: 2 T_{CYC} 0x2: 3 T_{CYC} ... 0x3F: 64 T_{CYC}
<3:0>	TMPRR	R/W	0x1	0x1	Indicates T_{MPRR} constraints. Set this field as follows: RNDUP[T_{MPRR}(ns) / T_{CYC}(ns)] – 1 where T_{MPRR} is from the JEDEC DDR3 spec, and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = 1 nCK

Cavium Confidential For Bowen Yang ALU 02/27/2014

LMC Mode Register Parameters 0 Register LMC(0)_MODEREG_PARAMS0

These parameters are written into the DDR3 MR0, MR1, MR2 and MR3 registers. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved
<24>	PPD	R/W	0	0	DLL control for precharge powerdown. 0 = slow exit (DLL off) 1 = fast exit (DLL on) LMC writes this value to MR0[PPD] in the selected DDR3 parts during power-up/init and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START] and LMC(0)_CONFIG[RANKMASK]. This value must equal the MR0[PPD] value in all the DDR3 parts attached to all ranks during normal operation.
<23:21>	WRP	R/W	0x0	0x0	Write recovery for auto precharge. Should be programmed to be equal to or greater than RNDUP[T_{WR}(ns) / T_{cyc}(ns)]. 0x0 = 5 0x4 = 8 0x1 = 5 0x5 = 10 0x2 = 6 0x6 = 12 0x3 = 7 0x7 = 14 LMC writes this value to MR0[WR] in the selected DDR3 parts during power-up/init and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START] and LMC(0)_CONFIG[RANKMASK]. This value must equal the MR0[WR] value in all the DDR3 parts attached to all ranks during normal operation.
<20>	DLLR	R/W	0	0	DLL reset. LMC writes this value to MR0[DLL] in the selected DDR3 parts during power-up/init and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh exit instruction sequences. See LMC(0)_CONFIG[SEQUENCE,INIT_START,RANKMASK]. The MR0[DLL] value must be 0 in all the DDR3 parts attached to all ranks during normal operation.
<19>	TM	R/W	0	0	Test mode. LMC writes this value to MR0[TM] in the selected DDR3 parts during power-up/init and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START] and LMC(0)_CONFIG[RANKMASK]. The MR0[TM] value must be 0 in all the DDR3 parts attached to all ranks during normal operation.
<18>	RBT	RO	1	1	Read burst. Type 1 = interleaved (fixed). LMC writes this value to MR0[RBT] in the selected DDR3 parts during power-up/init and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START] and LMC(0)_CONFIG[RANKMASK]. The MR0[RBT] value must be 1 in all the DDR3 parts attached to all ranks during normal operation.

Cavium Confidential For Ravi Yarlagadda ALD 8/7/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<17:14>	CL	R/W	0x2	0x2	CAS latency. In DDR3 mode: 0x2 = 5 0x1 = 12 0x4 = 6 0x3 = 13 0x6 = 7 0x5 = 14 0x8 = 8 0x7 = 15 0xA = 9 0x9 = 16 0xC = 10 0x0, 0xB, 0xD, 0xF = reserved 0xE = 11 In DDR4 mode: 0x0 = 9 0x1 = 10 0x2 = 11 0x3 = 12 0x4 = 13 0x5 = 14 0x6 = 15 0x7 = 16 0x8 = 18 0x9 = 20 0xA = 22 0xB = 24 0xC, 0xD, 0xE, 0xF = reserved LMC writes this value to MR0[CAS Latency / CL] in the selected DDR3 parts during power-up/init and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START] and LMC(0)_CONFIG[RANKMASK]. This value must equal the MR0[CAS Latency / CL] value in all the DDR3/4 parts attached to all ranks during normal operation. t_{CL} must be programmed to greater than or equal to t_{CWL} for proper LMC operation.
<13:12>	BL	RO	0x1	0x1	Burst length: 0 = 8 (fixed), 1 = 4 or 8 (on-the-fly). LMC writes this value to MR0[BL] in the selected DDR3 parts during power-up/init and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START] and LMC(0)_CONFIG[RANKMASK]. The MR0[BL] value must be 0 in all the DDR3/4 parts attached to all ranks during normal operation.
<11>	QOFF	R/W	0	0	Qoff enable: 0 = enable, 1= disable. LMC writes this value to MR1[Qoff] in the DDR3 parts in the selected ranks during power-up/init, write-leveling, and if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK,INIT_STATUS] and LMC(0)_RESET_CTL[DDR3PWARM, DDR3PSOFT]. The MR1[Qoff] value must be 0 in all the DDR3 parts attached to all ranks during normal operation.
<10>	TDQS	R/W	0	0	TDQS enable: 1 = enable, 0 = disable. LMC writes this value to MR1[TDQS] in the DDR3 parts in the selected ranks during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK,INIT_STATUS] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT].

Cavium Confidential For Beijing Huawei

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<9>	WLEV	RO	0	0	Write leveling enable: 1 = enable, 0 = disable. LMC writes MR1[Level]=0 in the DDR3 parts in the selected ranks during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit sequences. (Write-leveling can only be initiated via the write-leveling instruction sequence.) See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK,INIT_STATUS] and LMC(0)_RESET_CTL[DDR3PWARM, DDR3PSOFT].
<8:7>	AL	R/W	0x0	0x0	Reserved; must be zero.
<6>	DLL	R/W	0	0	DLL Enable. 0 = enable; 1 = disable. LMC writes this value to MR1[DLL] in the selected DDR3 parts during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK,INIT_STATUS] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT]. This value must equal the MR1[DLL] value in all the DDR3 parts attached to all ranks during normal operation. In DLL-off mode, CL/CWL must be programmed equal to 6/6, respectively, as per the JEDEC DDR3 specifications.
<5>	MPR	R/W	0	0	MPR. LMC writes this value to MR3[MPR] in the selected DDR3 parts during power-up/init, read-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh exit instruction sequences. (LMC also writes MR3[MPR] = 1 at the beginning of the read-leveling instruction sequence. Read-leveling should only be initiated via the read-leveling instruction sequence.) See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START] and LMC(0)_CONFIG[RANKMASK]. The MR3[MPR] value must be 0 in all the DDR3 parts attached to all ranks during normal operation.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<4:3>	MPRLOC	R/W	0x0	0x0	MPR Location. LMC writes this value to MR3[MPRLoc] in the selected DDR3 parts during power-up/init, read-leveling, and, if LMC(0)_CONFIG [SREF_WITH_DLL] is set, self-refresh exit instruction sequences. (LMC also writes MR3[MPRLoc] = 0 at the beginning of the read-leveling instruction sequence.) See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START] and LMC(0)_CONFIG[RANKMASK]. The MR3[MPRLoc] value must be 0 in all the DDR3 parts attached to all ranks during normal operation.
<2:0>	CWL	R/W	0x0	0x0	CAS write latency In DDR3 mode: 0x0 = 5 0x4 = 9 0x1 = 6 0x5 = 10 0x2 = 7 0x6 = 11 0x3 = 8 0x7 = 12 In DDR4 mode: 0x0 = 9 0x4 = 10 0x1 = 11 0x5 = 12 0x2 = 13 0x6 = 16 0x3 = 18 0x7 = reserved LMC writes this value to MR2[CWL] in the selected DDR3 parts during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK] and LMC(0)_RESET_CTL[DDR3PWARM, DDR3PSOFT]. This value must equal the MR2[CWL] value in all the DDR3 parts attached to all ranks during normal operation. T_{CWL} must be programmed to less than or equal to T_{CL} for proper LMC operation.

Cavium Confidential For Bowen Yang ALU 02/27/2014

LMC Write OnDieTermination Mask Register LMC(0)_WODT_MASK

System designers may desire to terminate DQ/DQS lines for higher-frequency DDR operations, especially on a multirank system. DDR3 DQ/DQS I/Os have built-in termination resistors that can be turned on or off by the controller, after meeting T_{AOND} and T_{AOF} timing requirements. Each rank has its own ODT pin that fans out to all of the memory parts in that DIMM. System designers may prefer different combinations of ODT ONs for write operations into different ranks.

CN71XX supports full programmability by way of the mask register below. Each rank position has its own four-bit programmable field. When the controller writes to that rank, it sets the four ODT pins to the mask pins as shown in Table 4–8. For example, when doing a write into Rank0, a system designer may desire to terminate the lines with the resistor on DIMM0/Rank1. The mask WODT_D0_R0 would then be 0x2.

CN71XX drives the appropriate mask values on the ODT pins by default. If this feature is not required, write 0x0 in this register. When a given RANK is selected, the WODT mask for that RANK is used. The resulting WODT mask is driven to the DIMMs in the following manner:

Table 4–8 Mask-to-WODT Conversion

	RANK_ENA=1	RANK_ENA=0
Mask[3] →	MBZ	MBZ
Mask[2] →	MBZ	MBZ
Mask[1] →	DDR_ODT_1	MBZ
Mask[0] →	DDR_ODT_0	DDR_ODT_0

LMC always writes entire cache blocks and always writes them via two consecutive write CAS operations to the same rank+bank+row spaced exactly 4 CK cycles apart. When a WODT mask bit is set, LMC asserts the CN71XX ODT output pin(s) starting the same CK as the first write CAS operation. Then, CN71XX normally continues to assert the ODT output pin(s) for 9+LMC(0)_CONTROL[WODT_BPRCH] more CK cycles—for a total of 10+LMC(0)_CONTROL[WODT_BPRCH] CK cycles for the entire cache block write—through the second write CAS operation of the cache block, satisfying the 6 CK DDR3 ODTH8 requirements.

But it is possible for CN71XX to issue two cache-block write operations separated by as few as WtW = 8 or 9 (10 if LMC(0)_CONTROL[WODT_BPRCH] = 1) CK cycles. In that case, CN71XX asserts the ODT output pin(s) for the WODT mask of the first cache block write for WtW CK cycles, then asserts the ODT output pin(s) for the WODT mask of the second cache block write for 10+LMC(0)_CONTROL[WODT_BPRCH] CK cycles (or less if a third cache block write follows within 8 or 9 (or 10) CK cycles of this second cache block write). Note that it may be necessary to force LMC to space back-to-back cache-block write operations to different ranks apart by at least 10+LMC(0)_CONTROL[WODT_BPRCH] CK cycles to prevent DDR3 ODTH8 violations.

See Table 4–7 for the address.

Cavium Confidential For Beyond ARM Wang 5/21/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved
<11:8>	WODT_D0_R1	R/W	0xF	—	Write ODT mask DIMM0, RANK1/DIMM0 in SingleRanked. If RANK_ENA = 0, WODT_D0_R1<3:0> must be zero.
<7:4>	—	RAZ	—	—	Reserved
<3:0>	WODT_D0_R0	R/W	0xF	—	Write ODT mask DIMM0, RANK0. If RANK_ENA = 0, WODT_D0_R0<3,1> must be zero.

LMC Compensation Control Register LMC(0)_COMP_CTL2

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:50>	—	RAZ	—	—	Reserved
<49:45>	DDR__PTUNE	RO/H	—	—	DDR PCTL from compensation circuit. The encoded value provides debug information for the compensation impedance on P-pullup. Attribute: -exempt_name_double_underscore=True.
<44:40>	DDR__NTUNE	RO/H	—	—	DDR NCTL from compensation circuit. The encoded value provides debug information for the compensation impedance on N-pulldown. Attribute: -exempt_name_double_underscore=True.
<39:36>	PTUNE_OFFSET	R/W	0	—	Ptune Offset value.
<35:32>	NTUNE_OFFSET	R/W	0	—	Ntune Offset value.
<31>	M180	R/W	0	—	Reserved; must be zero.
<30>	BYP	R/W	0	—	Bypass mode. When set, PTUNE and NTUNE are the compensation setting. When clear, DDR__PTUNE and DDR__NTUNE are the compensation setting.
<29:25>	PTUNE	R/W	0x0	—	PCTL impedance control in bypass mode
<24:20>	NTUNE	R/W	0x0	—	NCTL impedance control in bypass mode
<19:16>	RODT_CTL	R/W	0x0	—	NCTL RODT impedance control bits. This field controls ODT values during a memory read on the CN71XX side. In DDR3 mode: 0x0 = No ODT 0x1 = 20 Ω 0x2 = 30 Ω 0x3 = 40 Ω 0x4 = 60 Ω 0x5 = 120 Ω 0x6–0xF = reserved In DDR4 mode: 0x0 = No ODT 0x1 = 40 Ω 0x2 = 60 Ω 0x3 = 80 Ω 0x4 = 120 Ω 0x5 = 240 Ω 0x6 = 34 Ω 0x7 = 48 Ω 0x8–0xF = reserved

Cavium Confidential For Bowen Yang ALD 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<15:12>	CONTROL_CTL	R/W	0x4	—	Drive strength control for ODT, etc. drivers.. In DDR3 mode: 0x1 = 24 Ω 0x5 = 40 Ω 0x2 = 26.67 Ω 0x6 = 48 Ω 0x3 = 30 Ω 0x7 = 60 Ω 0x4 = 34.3 Ω 0x0, 0x8–0xF = reserved In DDR4 mode: 0x0 = reserved 0x4 = 34 Ω 0x1 = reserved 0x5 = 40 Ω 0x2 = 26 Ω 0x6 = 48 Ω 0x3 = 30 Ω 0x7 = 68 Ω 0x8–0xF = reserved
<11:8>	CMD_CTL	R/W	0x4	—	Drive strength control for CMD/A/RESET_L drivers. In DDR3 mode: 0x1 = 24 Ω 0x5 = 40 Ω 0x2 = 26.67 Ω 0x6 = 48 Ω 0x3 = 30 Ω 0x7 = 60 Ω 0x4 = 34.3 Ω 0x0,0x8–0xF = reserved In DDR4 mode: 0x0 = reserved 0x4 = 34 Ω 0x1 = reserved 0x5 = 40 Ω 0x2 = 26 Ω 0x6 = 48 Ω 0x3 = 30 Ω 0x7 = 68 Ω 0x8–0xF = reserved
<7:4>	CK_CTL	R/W	0x4	—	Drive strength control for DDR_CK_*_P/DDR_DIMM*_CS*_L/ DDR_DIMM*_ODT_*/DDR#_DIMM*_CKE* drivers. In DDR3 mode: 0x1 = 24 Ω 0x5 = 40 Ω 0x2 = 26.67 Ω 0x6 = 48 Ω 0x3 = 30 Ω 0x7 = 60 Ω 0x4 = 34.3 Ω 0x0,0x8–0xF = reserved In DDR4 mode: 0x0 = reserved 0x4 = 34 Ω 0x1 = reserved 0x5 = 40 Ω 0x2 = 26 Ω 0x6 = 48 Ω 0x3 = 30 Ω 0x7 = 68 Ω 0x8–0xF = reserved
<3:0>	DQX_CTL	R/W	0x4	—	Drive strength control for DDR_DQ*/DDR_DQS_*_P/N drivers. 0x1 = 24 Ω 0x5 = 40 Ω 0x2 = 26.67 Ω 0x6 = 48 Ω 0x3 = 30 Ω 0x7 = 60 Ω 0x4 = 34.3 Ω 0x0,0x8–0xF = reserved

Cavium Confidential For Bowen Yang ALU 02/27/2014

LMC DLL Control/System-Memory-Clock-Reset Register LMC(0)_DLL_CTL2

See Section 4.8 for the initialization sequence. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:17>	—	RAZ	—	—	Reserved
<16>	INTF_EN	R/W	0	—	Interface enable.
<15>	DLL_BRINGUP	R/W	0	0	DLL bring up.
<14>	DRESET	R/W	1	0	System-memory-clock domain reset. The reset signal that is used by the system-memory-clock domain is (DRESET -OR- core-clock reset).
<13>	QUAD_DLL_ENA	R/W	0	0	DLL enable
<12:9>	BYP_SEL	R/W	0x0	0x0	Reserved; must be zero.
<8:0>	BYP_SETTING	R/W	0x0	0x0	Reserved; must be zero.

LMC IFB Performance Counter Register LMC(0)_IFB_CNT

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	IFBCNT	RO/H	0x0	—	Performance counter. 64-bit counter that increments every CK cycle that there is something in the in-flight buffer.

LMC OPS Performance Counter Register LMC(0)_OPS_CNT

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	OPSCNT	RO/H	0x0	—	Performance counter. A 64-bit counter that increments when the DDR3 data bus is being used. Bus utilization = $\frac{\text{OPSCNT}}{\text{LMC(0)_DCLK_CNT[DCLKCNT]}}$

LMC System-Memory-Clock Counter Register LMC(0)_DCLK_CNT

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	DCLKCNT	RO/H	0x0	—	Performance counter. A 64-bit counter that increments every CK cycle.

Cavium Confidential For Bowen Yang ALU 02/27/2014

LMC Interrupt Enable Register LMC(0)_INT_EN

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved.
<5>	DDR_ERROR_ ALERT_ENA	R/W	0	—	Reserved.
<4>	DLCRAM_DED_ENA	R/W	0	1	Reserved.
<3>	DLCRAM_SEC_ENA	R/W	0	1	Reserved.
<2>	INTR_DED_ENA	R/W	0	1	ECC double error detect (DED) interrupt enable bit. When set, the memory controller raises a processor interrupt on detecting an uncorrectable double-bit ECC error.
<1>	INTR_SEC_ENA	R/W	0	1	ECC single error correct (SEC) interrupt enable bit. When set, the memory controller raises a processor interrupt on detecting a correctable single-bit ECC error.
<0>	INTR_NXM_WR_ENA	R/W	0	1	Nonwrite error interrupt enable bit. When set, the memory controller raises a processor interrupt on detecting an nonexistent memory write.

LMC Interrupt Register LMC(0)_INT

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved
<11>	DDR_ERR	R/W1C/H	0	0	Reserved.
<10>	DLCRAM_DED_ ERR	R/W1C/H	0	0	Reserved.
<9>	DLCRAM_SEC_ ERR	R/W1C/H	0	0	Reserved.
<8:5>	DED_ERR	R/W1C/H	0x0	0x0	Double error detected (DED) of Rd Data. Each bit corresponds to two phases: <5> corresponds to {DQ[31:0]_c0_p1, DQ[31:0]_c0_p0} <6> corresponds to {DQ[31:0]_c1_p1, DQ[31:0]_c1_p0} <7> corresponds to {DQ[31:0]_c2_p1, DQ[31:0]_c2_p0} <8> corresponds to {DQ[31:0]_c3_p1, DQ[31:0]_c3_p0} where _cC_pP denotes cycle C and phase P. Write of 1 clears the corresponding error bit.
<4:1>	SEC_ERR	R/W1C/H	0x0	0x0	Single error (corrected) of Rd Data. Each bit corresponds to two phases: <1> corresponds to {DQ[31:0]_c0_p1, DQ[31:0]_c0_p0} <2> corresponds to {DQ[31:0]_c1_p1, DQ[31:0]_c1_p0} <3> corresponds to {DQ[31:0]_c2_p1, DQ[31:0]_c2_p0} <4> corresponds to {DQ[31:0]_c3_p1, DQ[31:0]_c3_p0} where _cC_pP denotes cycle C and phase P. Write of 1 clears the corresponding error bit.
<0>	NXM_WR_ERR	R/W1C/H	0	0	Write to nonexistent memory. Write of 1 clears the corresponding error bit.

Cavium Confidential Bowen Yang ALU 3/2/2014

LMC Slot Control0 Register
LMC(0)_SLOT_CTL0

This register is an assortment of control fields needed by the memory controller. If software has not previously written to this register (since the last DRESET), hardware updates the fields in this register to the minimum allowed value when any of LMC(0)_RLEVEL_RANK(0..3), LMC(0)_WLEVEL_RANK(0..3), LMC(0)_CONTROL and LMC(0)_MODEREG_PARAMS0 registers change. Ideally, only read this register after LMC has been initialized and LMC(0)_RLEVEL_RANK(0..3), LMC(0)_WLEVEL_RANK(0..3) have valid data.

The interpretation of the fields in this register depends on LMC(0)_CONFIG[DDR2T]:

- If LMC(0)_CONFIG[DDR2T] = 1, (FieldValue + 4) is the minimum CK cycles between when the DRAM part registers CAS commands of the 1st and 2nd types from different cache blocks.
- If LMC(0)_CONFIG[DDR2T] = 0, (FieldValue + 3) is the minimum CK cycles between when the DRAM part registers CAS commands of the 1st and 2nd types from different cache blocks. FieldValue = 0 is always illegal in this case.

The hardware-calculated minimums for these fields are shown in Table 4–9.

Table 4–9 LMC(0)_SLOT_CTL0 Hardware-Calculated Minimums

Field Name	Minimum
R2R_S_INIT	4
R2W_S_INIT	8 + (RL[1] + MaxRdSkew[2]) – (WL[3] + MinWrSkew[4]) + LMC(0)_CONTROL[BPRCH]
W2R_S_INIT	5 + LMC(0)_TIMING_PARAMS1[TWTR] + WL
W2W_S_INIT	4
R2R_L_INIT	LMC(0)_MODEREG_PARAMS3[TCCD_L] (decoded
R2W_L_INIT	8 + (RL[1] + MaxRdSkew[2]) (WL[3] + MinWrSkew[4]) + LMC*_CONTROL[BPRCH]
W2R_L_INIT	5 + LMC(0)_TIMING_PARAMS2[TWTR_L] + WL
W2W_L_INIT	LMC(0)_MODEREG_PARAMS3[TCCD_L] (decoded

1. RL = CL + AL (LMC(0)_MODEREG_PARAMS0[CL] selects CL, LMC(0)_MODEREG_PARAMS0[AL] selects AL)
2. MaxRdSkew = max(LMC(0)_RLEVEL_RANKi[BYTEj] ÷ 4) + 1 (max is across all ranks i (0..3) and bytes j (0..8))
3. WL = CWL + AL (LMC(0)_MODEREG_PARAMS0[CWL] selects CWL)
4. MinWrSkew = min(LMC(0)_WLEVEL_RANKi[BYTEj] ÷ 8) – LMC(0)_CONFIG[EARLY_DQX] (min is across all ranks i (0..3) and bytes j (0..8))

R2W_INIT has one CK cycle built in for CN71XX-internal ODT settling/channel turnaround time. See Table 4–7 for the address.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:42>	W2W_L_INIT	R/W/H	0x5	—	Write-to-write spacing control for back-to-back write followed by write cache block accesses to the same rank and DIMM, and same BG for DDR4.
<41:36>	W2R_L_INIT	R/W/H	0x10	—	Write-to-read spacing control for back-to-back write followed by read cache block accesses to the same rank and DIMM, and same BG for DDR4.
<35:30>	R2W_L_INIT	R/W/H	0x9	—	Read-to-write spacing control for back-to-back read followed by write cache block accesses to the same rank and DIMM, and same BG for DDR4.
<29:24>	R2R_L_INIT	R/W/H	0x5	—	Read-to-read spacing control for back-to-back read followed by read cache block accesses to the same rank and DIMM, and same BG for DDR4.
<23:18>	W2W_INIT	R/W/H	0x4	—	Write-to-write spacing control for back-to-back write followed by write cache block accesses to the same rank and DIMM.
<17:12>	W2R_INIT	R/W/H	0xC	—	Write-to-read spacing control for back-to-back write followed by read cache block accesses to the same rank and DIMM.
<11:6>	R2W_INIT	R/W/H	0x9	—	Read-to-write spacing control for back-to-back read followed by write cache block accesses to the same rank and DIMM.
<5:0>	R2R_INIT	R/W/H	0x4	—	Read-to-read spacing control for back-to-back read followed by read cache block accesses to the same rank and DIMM.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

LMC Slot Control1 Register
LMC(0)_SLOT_CTL1

This register is an assortment of control fields needed by the memory controller. If software has not previously written to this register (since the last DRESET), hardware updates the fields in this register to the minimum allowed value when any of LMC(0)_RLEVEL_RANK*n*, LMC(0)_WLEVEL_RANK*n*, LMC(0)_CONTROL and LMC(0)_MODEREG_PARAMS0 CSRs change. Ideally, only read this register after LMC has been initialized and LMC(0)_RLEVEL_RANK*n*, LMC(0)_WLEVEL_RANK*n* have valid data. The field value is the minimum CK cycles between when the DRAM part registers CAS commands of the first and second types from different cache blocks.

The hardware-calculated minimums for these fields are shown in Table 4–10.

Table 4–10 LMC(0)_SLOT_CTL1 Hardware-Calculated Minimums

Field Name	Minimum
R2R_XRANK_INIT	5 + MaxRdSkew[1] – MinRdSkew[2]+ LMC(0)_CONTROL[RODT_BPRCH]
R2W_XRANK_INIT	8 + (RL[3] + MaxRdSkew[1]) – (WL[4] + MinWrSkew[5]) + LMC(0)_CONTROL[BPRCH]
W2R_XRANK_INIT	6 + MaxWrSkew[6] + LMC(0)_CONTROL[FPRCH2]
W2W_XRANK_INIT	7 + MaxWrSkew[6] - MinWrSkew[5]

1. MaxRdSkew = max(LMC(0)_RLEVEL_RANKi[BYTEj] ÷ 4) + 1 (max is across all ranks i (0..3) and bytes j (0..8))
2. MinRdSkew = min(LMC(0)_RLEVEL_RANKi[BYTEj] ÷ 4) (min is across all ranks i (0..3) and bytes j (0..8))
3. RL = CL + AL (LMC(0)_MODEREG_PARAMS0[CL] selects CL, LMC(0)_MODEREG_PARAMS0[AL] selects AL)
4. WL = CWL + AL (LMC(0)_MODEREG_PARAMS0[CWL] selects CWL)
5. MinWrSkew = min(LMC(0)_WLEVEL_RANKi[BYTEj] ÷ 8) – LMC(0)_CONFIG[EARLY_DQX] (min is across all ranks i (0..3) and bytes j (0..8))
6. MaxWrSkew = max(LMC(0)_WLEVEL_RANKi[BYTEj] ÷ 8) – LMC(0)_CONFIG[EARLY_DQX] + 1 (max is across all ranks i (0..3) and bytes j (0..8))

R2W_XRANK_INIT has one extra CK cycle built in for CN71XX-internal ODT settling/channel turnaround time. W2R_XRANK_INIT has one extra CK cycle built in for channel turnaround time. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:24>	—	RAZ	—	—	Reserved
<23:18>	W2W_XRANK_INIT	R/W/H	0x8	—	Write-to-write spacing control for back-to-back write followed by write cache block accesses across ranks of the same DIMM.
<17:12>	W2R_XRANK_INIT	R/W/H	0x7	—	Write-to-read spacing control for back-to-back write followed by read cache block accesses across ranks of the same DIMM.
<11:6>	R2W_XRANK_INIT	R/W/H	0x9	—	Read-to-write spacing control for back-to-back read followed by write cache block accesses across ranks of the same DIMM.
<5:0>	R2R_XRANK_INIT	R/W/H	0x6	—	Read-to-read spacing control for back-to-back read followed by read cache block accesses across ranks of the same DIMM.

Cavium Confidential For Bowen Yang ALC 07/2014

LMC Slot Control2 Register
LMC(0)_SLOT_CTL2

This register is an assortment of control fields needed by the memory controller. If software has not previously written to this register (since the last DRESET), hardware updates the fields in this register to the minimum allowed value when any of LMC(0)_RLEVEL_RANK*n*, LMC(0)_WLEVEL_RANK*n*, LMC(0)_CONTROL and LMC(0)_MODEREG_PARAMS0 CSRs change. Ideally, only read this register after LMC has been initialized and LMC(0)_RLEVEL_RANK*n*, LMC(0)_WLEVEL_RANK*n* have valid data. The field value is the minimum CK cycles between when the DRAM part registers CAS commands of the first and second types from different cache blocks.

The hardware-calculated minimums for these fields are shown in Table 4–11.

Table 4–11 LMC(0)_SLOT_CTL2 Hardware-Calculated Minimums

Field Name	Minimum
R2R_XDIMM_INIT	6 + MaxRdSkew[1] – MinRdSkew[2]+ LMC(0)_CONTROL[RODT_BPRCH]
R2W_XDIMM_INIT	9 + (RL[3] + MaxRdSkew[1]) – (WL[4] + MinWrSkew[5]) + LMC(0)_CONTROL[BPRCH]
W2R_XDIMM_INIT	6 + MaxWrSkew[6] + LMC(0)_CONTROL[FPRCH2]
W2W_XDIMM_INIT	8 + MaxWrSkew[6] - MinWrSkew[5]

1. MaxRdSkew = max(LMC(0)_RLEVEL_RANKi[BYTEj] ÷ 4) + 1 (max is across all ranks i (0..3) and bytes j (0..8))
2. MinRdSkew = min(LMC(0)_RLEVEL_RANKi[BYTEj] ÷ 4) (min is across all ranks i (0..3) and bytes j (0..8))
3. RL = CL + AL (LMC(0)_MODEREG_PARAMS0[CL] selects CL, LMC(0)_MODEREG_PARAMS0[AL] selects AL)
4. WL = CWL + AL (LMC(0)_MODEREG_PARAMS0[CWL] selects CWL)
5. MinWrSkew = min(LMC(0)_WLEVEL_RANKi[BYTEj] ÷ 8) – LMC(0)_CONFIG[EARLY_DQX] (min is across all ranks i (0..3) and bytes j (0..8))
6. MaxWrSkew = max(LMC(0)_WLEVEL_RANKi[BYTEj] ÷ 8) – LMC(0)_CONFIG[EARLY_DQX] + 1 (max is across all ranks i (0..3) and bytes j (0..8))

R2W_XDIMM_INIT has 2 extra CK cycles built in for CN71XX-internal ODT settling/channel turnaround time. R2R_XDIMM_INIT, W2R_XRANK_INIT, W2W_XDIMM_INIT have 1 extra CK cycle built in for channel turnaround time. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:24>	—	RAZ	—	—	Reserved
<23:18>	W2W_XDIMM_INIT	R/W/H	0x9	—	Write-to-write spacing control for back-to-back write followed by write cache block accesses across DIMMs.
<17:12>	W2R_XDIMM_INIT	R/W/H	0x7	—	Write-to-read spacing control for back-to-back write followed by read cache block accesses across DIMMs.
<11:6>	R2W_XDIMM_INIT	R/W/H	0xA	—	Read-to-write spacing control for back-to-back read followed by write cache block accesses across DIMMs.
<5:0>	R2R_XDIMM_INIT	R/W/H	0x7	—	Read-to-read spacing control for back-to-back read followed by read cache block accesses across DIMMs.

Cavium Confidential For Bowen Yang AL 2/27/2014

LMC PHY Control Register
LMC(0)_PHY_CTL

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:51>	—	RAZ	—	—	Reserved
<50>	PHY_RESET	WO	0	0	Reserved.
<49>	DSK_DBG_RD_COMPLETE	RO/H	0	—	Reserved.
<48:39>	DSK_DBG_RD_DATA	RO/H	0x0	—	Reserved.
<38>	DSK_DBG_RD_START	WO/H	0	0	Reserved.
<37:36>	DSK_DBG_CLK_SCALER	R/W	0x1	0x1	Reserved.
<35:34>	DSK_DBG_OFFSET	R/W	0x0	0x0	Reserved.
<33>	DSK_DBG_NUM_BITS_SEL	R/W	1	1	Reserved.
<32:29>	DSK_DBG_BYTE_SEL	R/W	0x0	0x0	Reserved.
<28:25>	DSK_DBG_BIT_SEL	R/W	0x0	0x0	Reserved.
<24>	DBI_MODE_ENA	RO	0	—	Enable DBI mode for PHY, must be 0. DBI mode is not supported in CN71XX.
<23>	DDR_ERROR_N_ENA	R/W	0	0	Reserved.
<22>	REF_PIN_ON	R/W	0	0	Reserved.
<21>	DAC_ON	R/W	0	0	Reserved.
<20>	INT_PAD_LOOPBACK_ENA	R/W	0	—	DDR pad loopback enable.
<19>	INT_PHY_LOOPBACK_ENA	R/W	0	—	Internal PHY loopback enable.
<18>	PHY_DSK_RESET	R/W	0	—	PHY deskew reset. When set, the deskew reset signal goes active if the vrefint/deskew training sequence is in the idle state.
<17>	PHY_DSK_BYP	R/W	0	—	PHY deskew bypass.
<16>	PHY_PWR_SAVE_DISABLE	R/W	0	—	DDR PHY power save disable.
<15>	TEN	R/W	0	—	DDR PHY test-enable pin.
<14>	RX_ALWAYS_ON	R/W	0	—	Reserved; must be zero.
<13>	LV_MODE	R/W	0	—	Reserved; must be zero.
<12>	CK_TUNE1	R/W	0	—	Reserved; must be zero.
<11:8>	CK_DLYOUT1	R/W	0x0	—	Reserved; must be zero.
<7>	CK_TUNE0	R/W	0	—	Reserved; must be zero.
<6:3>	CK_DLYOUT0	R/W	0x0	—	Reserved; must be zero.
<2>	LOOPBACK	R/W	0	—	Reserved; must be zero.

Cavium Confidential For Bowen Yang At U 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<1>	LOOPBACK_POS	R/W	0	—	Reserved; must be zero.
<0>	TS_STAGGER	R/W	0	—	TS stagger mode. This mode configures output drivers with two-stage drive strength to avoid undershoot issues on the bus when strong drivers are suddenly turned on. When this mode is asserted, CN71XX configures output drivers to be weak drivers (60Ω output impedance) at the first CK cycle, and change drivers to the designated drive strengths specified in LMC(0)_COMP_CTL2 [CMD_CTL/CK_CTL/DQX_CTL] starting at the following cycle.

LMC DLL Control/System-Memory-Clock Reset Register LMC(0)_DLL_CTL3

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:44>	—	RAZ	—	—	Reserved
<43>	DCLK90_FWD	W0	0	0	Reserved; must be zero.
<42>	DDR_90_DLY_BYP	R/W	0	0	Reserved; must be zero.
<41>	DCLK90_RECAL_DIS	R/W	1	1	Disable periodic recalibration of DDR90 delay line in.
<40>	DCLK90_BYP_SEL	R/W	0	0	Bypass setting select for DDR90 delay line
<39:31>	DCLK90_BYP_SETTING	R/W	0x0	0x0	Bypass setting for DDR90 delay line
<30>	DLL_FAST	RO/H	—	—	Reserved; must be zero.
<29:21>	DLL90_SETTING	RO/H	—	—	Reserved; must be zero.
<20>	FINE_TUNE_MODE	R/W	0	1	DLL fine tune mode. 0 = disabled; 1 = enable. When enabled, calibrate internal PHY DLL every LMC(0)_CONFIG[REF_ZQCS_INT] CK cycles.
<19>	DLL_MODE	R/W	0	0	Reserved; must be zero.
<18:15>	DLL90_BYTE_SEL	R/W	0x1	0x1	Observe DLL settings for selected byte. 0x3 = byte 4 0x4 = byte 3 0x5 = byte 2 0x6 = byte 1 0x7 = byte 0 0x0–0x2,0x8–0xF: Reserved
<14>	OFFSET_ENA	R/W	0	0	Reserved; must be zero.
<13>	LOAD_OFFSET	WR0	0	0	Reserved; must be zero.
<12:11>	MODE_SEL	R/W	0x0	0x0	Reserved; must be zero.
<10:7>	BYTE_SEL	R/W	0x0	0x0	Reserved; must be zero.
<6:0>	OFFSET	R/W	0x0	0x0	Reserved; must be zero.

Cavium Confidential For Bowei Yang 2/27/2014

LMC DDR PLL Control Register
LMC(0)_DDR_PLL_CTL

This register controls the DDR_CK frequency. For details, refer to Section 4.13. See Section 4.8 for the initialization sequence. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:31>	—	RAZ	—	—	Reserved
<30>	PHY_DCOK	R/W	0	1	Set to power up PHY logic after setting LMC(0)_DDR_PLL_CTL[DDR4_MODE].
<29>	DDR4_MODE	R/W	0	—	DDR4 mode select: 1 = DDR4, 0 = DDR3.
<28>	PLL_FBSLIP	RO/H	0	—	PLL FBSLIP indication.
<27>	PLL_LOCK	RO/H	0	—	PLL LOCK indication.
<26>	PLL_RXSLIP	RO/H	0	—	PLL RXSLIP indication.
<25:24>	CLKR	R/W	0x0	—	PLL post-divider control.
<23>	JTG_TEST_MODE	R/W	0	0	Reserved; must be zero.
<22>	DDR_DIV_RESET	R/W	1	0	DDR postscalar-divider reset.
<21:18>	DDR_PS_EN	R/W	0x2	—	DDR postscalar divide ratio. Determines the LMC CK speed. 0x0 = divide LMC PLL by 1 0x1 = divide LMC PLL by 2 0x2 = divide LMC PLL by 3 0x3 = divide LMC PLL by 4 0x4 = divide LMC PLL by 5 0x5 = divide LMC PLL by 6 0x6 = divide LMC PLL by 7 0x7 = divide LMC PLL by 8 0x8 = divide LMC PLL by 10 0x9 = divide LMC PLL by 12 0xA–0xF = reserved DDR_PS_EN is not used when DDR_DIV_RESET = 1
<17:8>	—	RAZ	—	—	Reserved
<7>	RESET_N	R/W	0	1	PLL reset
<6:0>	CLKF	R/W	0x30	—	Multiply reference by CLKF. 32 ≤ CLKF ≤ 64. LMC PLL frequency = 50 × CLKF. min = 1.0 GHz, max = 2.4 GHz.

Cavium Confidential For Boliena Yang ALU 02/27/2014

LMC Mode Register Parameters 1 Register LMC(0)_MODEREG_PARAMS1

These parameters are written into the DDR3 MR0, MR1, MR2 and MR3 registers. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:45>	RTT_NOM_11	R/W	0x0	0x0	Reserved.
<44:43>	DIC_11	R/W	0x0	0x0	Reserved.
<42:41>	RTT_WR_11	R/W	0x0	0x0	Reserved.
<40>	SRT_11	R/W	0	0	Reserved.
<39>	ASR_11	R/W	0	0	Reserved.
<38:36>	PASR_11	R/W	0x0	0x0	Reserved.
<35:33>	RTT_NOM_10	R/W	0x0	0x0	Reserved.
<32:31>	DIC_10	R/W	0x0	0x0	Reserved.
<30:29>	RTT_WR_10	R/W	0x0	0x0	Reserved.
<28>	SRT_10	R/W	0	0	Reserved.
<27>	ASR_10	R/W	0	0	Reserved.
<26:24>	PASR_10	R/W	0x0	0x0	Reserved.
<23:21>	RTT_NOM_01	R/W	0x0	0x0	RTT_NOM rank 1. LMC writes this value to MR1[RTT_NOM] in the rank 1 (i.e. DIMM0_CS1) DDR3 parts when selected during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT]. Per JEDEC DDR3 specifications, if RTT_NOM is used during write operations, only values MR1[RTT_NOM] = 1 (RZQ/4), 2 (RZQ/2), or 3 (RZQ/6) are allowed. Otherwise, values MR1[RTT_NOM] = 4 (RZQ/12) and 5 (RZQ/8) are also allowed.
<20:19>	DIC_01	R/W	0x0	0x0	Output driver impedance control rank 1. LMC writes this value to MR1[D.I.C.] in the rank 1 (i.e. DIMM0_CS1) DDR3 parts when selected during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT].
<18:17>	RTT_WR_01	R/W	0x0	0x0	RTT_WR rank 1. LMC writes this value to MR2[RTT_WR] in the rank 1 (i.e. DIMM0_CS1) DDR3 parts when selected during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT].
<16>	SRT_01	R/W	0	0	Self-refresh temperature range rank 1. LMC writes this value to MR2[SRT] in the rank 1 (i.e. DIMM0_CS1) DDR3 parts when selected during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT].

Cavium Confidential For Board Layout

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<15>	ASR_01	R/W	0	0	Auto self-refresh rank 1. LMC writes this value to MR2[ASR] in the rank 1 (i.e. DIMM0_CS1) DDR3 parts when selected during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT].
<14:12>	PASR_01	R/W	0x0	0x0	Partial array self-refresh rank 1. LMC writes this value to MR2[PASR] in the rank 1 (i.e. DIMM0_CS1) DDR3 parts when selected during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT].
<11:9>	RTT_NOM_00	R/W	0x0	0x0	RTT_NOM rank 0. LMC writes this value to MR1[RTT_NOM] in the rank 0 (i.e. DIMM0_CS0) DDR3 parts when selected during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT]. Per JEDEC DDR3 specifications, if RTT_NOM is used during write operations, only values MR1[RTT_NOM] = 1 (RZQ/4), 2 (RZQ/2), or 3 (RZQ/6) are allowed. Otherwise, values MR1[RTT_NOM] = 4 (RZQ/12) and 5 (RZQ/8) are also allowed.
<8:7>	DIC_00	R/W	0x0	0x0	Output driver impedance control rank 0. LMC writes this value to MR1[D.I.C.] in the rank 0 (i.e. DIMM0_CS0) DDR3 parts when selected during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT].
<6:5>	RTT_WR_00	R/W	0x0	0x0	RTT_WR rank 0. LMC writes this value to MR2[RTT_WR] in the rank 0 (i.e. DIMM0_CS0) DDR3 parts when selected during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT].
<4>	SRT_00	R/W	0	0	Self-refresh temperature range rank 0. LMC writes this value to MR2[SRT] in the rank 0 (i.e. DIMM0_CS0) DDR3 parts when selected during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT].

Cavium Confidential For Beta Review Only 2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<3>	ASR_00	R/W	0	0	Auto self-refresh rank 0. LMC writes this value to MR2[ASR] in the rank 0 (i.e. DIMM0_CS0) DDR3 parts when selected during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT].
<2:0>	PASR_00	R/W	0x0	0x0	Partial array self-refresh rank 0. LMC writes this value to MR2[PASR] in the rank 0 (i.e. DIMM0_CS0) DDR3 parts when selected during power-up/init, write-leveling, and, if LMC(0)_CONFIG[SREF_WITH_DLL] is set, self-refresh entry and exit instruction sequences. See LMC(0)_SEQ_CTL[SEQ_SEL, INIT_START], LMC(0)_CONFIG[RANKMASK] and LMC(0)_RESET_CTL[DDR3PWARM,DDR3PSOFT].

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

LMC Read On-DieTermination Mask Register LMC(0)_RODT_MASK

System designers may desire to terminate DQ/DQS lines for higher frequency DDR operations, especially on a multirank system. DDR3 DQ/DQS I/Os have built-in termination resistors that can be turned on or off by the controller, after meeting T_{AOND} and T_{AOF} timing requirements. Each rank has its own ODT pin that fans out to all the memory parts in that DIMM. System designers may prefer different combinations of ODT ONs for read operations into different ranks. CN71XX supports full programmability by way of the mask register below. Each rank position has its own 8-bit programmable field. When the controller does a read to that rank, it sets the two ODT pins to the MASK pins below. For example, when doing a read from rank 0, a system designer may desire to terminate the lines with the resistor on rank 1. The mask RODT_D0_R0 would then be {00000010}.

CN71XX drives the appropriate mask values on the ODT pins by default. If this feature is not required, write 0x0 in this register. Note that, as per the JEDEC DDR3 specifications, the ODT pin for the rank that is being read should always be 0x0.

When a given RANK is selected, the RODT mask for that rank is used. The resulting RODT mask is driven to the DIMM, as shown in Table 4–12.

Table 4–12 Mask-to-RODT Conversion

	RANK_ENA=1	RANK_ENA=0
Mask[3] →	MBZ	MBZ
Mask[2] →	MBZ	MBZ
Mask[1] →	DDR_ODT_1	MBZ
Mask[0] →	DDR_ODT_0	DDR_ODT_0

LMC always reads entire cache blocks and always reads them via two consecutive read CAS operations to the same rank+bank+row spaced exactly 4 CK cycles apart. When a RODT mask bit is set, LMC asserts the CN71XX ODT output pin(s) starting (CL – CWL) CK cycles after the first read CAS operation. Then, CN71XX normally continues to assert the ODT output pin(s) for 9+LMC(0)_CONTROL[RODT_BPRCH] more CK cycles—for a total of 10+LMC(0)_CONTROL[RODT_BPRCH] CK cycles for the entire cache block read—through the second read CAS operation of the cache block, satisfying the 6 CK DDR3 ODTH8 requirements.

But it is possible for CN71XX to issue two cache-block read operations separated by as few as RtR = 8 or 9 (10 if LMC(0)_CONTROL[RODT_BPRCH]=1) CK cycles. In that case, CN71XX asserts the ODT output pin(s) for the RODT mask of the first cache block read for RtR CK cycles, then asserts the ODT output pin(s) for the RODT mask of the second cache block read for 10+LMC(0)_CONTROL[RODT_BPRCH] CK cycles (or less if a third cache block read follows within 8 or 9 (or 10) CK cycles of this second cache block read). Note that it may be necessary to force LMC to space back-to-back cache-block read operations to different ranks apart by at least 10+LMC(0)_CONTROL[RODT_BPRCH] CK cycles to prevent DDR3 ODTH8 violations. See Table 4–7 for the address.

Cavium Confidential For Robert Yang 07/27/2015

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:28>	—	RAZ	0x0	—	Reserved; must be zero.
<27:24>	RODT_D1_R1	RAZ	0x0	—	Reserved; must be zero.
<23:20>	—	RAZ	0x0	—	Reserved; must be zero.
<19:16>	RODT_D1_R0	RAZ	0x0	—	Reserved; must be zero.
<15:12>	—	RAZ	0x0	—	Reserved; must be zero.
<11:8>	RODT_D0_R1	R/W	0x0	—	Read ODT mask DIMM0, RANK1/DIMM0 in SingleRanked. If RANK_ENA = 1, RODT_D0_R1<1> must be zero. Otherwise, RODT_D0_R1<3:0> is not used and must be zero.
<7:4>	—	RAZ	0x0	—	Reserved; must be zero.
<3:0>	RODT_D0_R0	R/W	0x0	—	Read ODT mask DIMM0, RANK0. If RANK_ENA = 1, RODT_D0_RO<0> must be zero. Otherwise, RODT_D0_RO<1:0,3> must be zero.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

LMC DIMM Parameters Register
LMC(0)_DIMM(0..1)_PARAMS

This register contains values to be programmed into each control word in the corresponding (registered) DIMM. The control words allow optimization of the device properties for different raw card designs. Note that LMC only uses this CSR when LMC(0)_CONTROL[RDIMM_ENA]=1. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:60>	RC15	R/W	0x0	0x0	RC15, Reserved
<59:56>	RC14	R/W	0x0	0x0	RC14, Reserved
<55:52>	RC13	R/W	0x0	0x0	RC13, Reserved
<51:48>	RC12	R/W	0x0	0x0	RC12, Reserved
<47:44>	RC11	R/W	0x0	0x0	RC11, Encoding for RDIMM operating VDD
<43:40>	RC10	R/W	0x0	0x0	RC10, Encoding for RDIMM operating speed
<39:36>	RC9	R/W	0x0	0x0	RC9, Power savings settings control word
<35:32>	RC8	R/W	0x0	0x0	RC8, Additional IBT settings control word
<31:28>	RC7	R/W	0x0	0x0	RC7, Reserved
<27:24>	RC6	R/W	0x0	0x0	RC6, Reserved
<23:20>	RC5	R/W	0x0	0x0	RC5, CK driver characteristics control word
<19:16>	RC4	R/W	0x0	0x0	RC4, Control signals driver characteristics control word
<15:12>	RC3	R/W	0x0	0x0	RC3, CA signals driver characteristics control word
<11:8>	RC2	R/W	0x0	0x0	RC2, Timing control word
<7:4>	RC1	R/W	0x0	0x0	RC1, Clock driver enable control word
<3:0>	RC0	R/W	0x0	0x0	RC0, Global features control word

Cavium Confidential For Bowen Yang ALU 02/27/2014

LMC Read Level Rank Register LMC(0)_RLEVEL_RANK(0..3)

Four of these CSRs exist per LMC, one for each rank. Read level setting is measured in units of ¼ CK, so the BYTE*n* values can range over 16 CK cycles. Each CSR is written by hardware during a read-leveling sequence for the rank. (Hardware sets STATUS to 3 after hardware read-leveling completes for the rank.)

If hardware is unable to find a match per LMC(0)_RLEVEL_CTL[OFFSET_EN] and LMC(0)_RLEVEL_CTL[OFFSET], then hardware sets LMC(0)_RLEVEL_RANK*i*[BYTE*n*<5:0>] to 0x0.

Each CSR may also be written by software, but not while a read-leveling sequence is in progress. (Hardware sets STATUS to 1 after a CSR write.) Software initiates a hardware read-leveling sequence by programming LMC(0)_RLEVEL_CTL and writing INIT_START = 1 with SEQUENCE = 1. See LMC(0)_RLEVEL_CTL.

LMC(0)_RLEVEL_RANK*i* values for ranks *i* without attached DRAM should be set such that they do not increase the range of possible BYTE values for any byte lane. The easiest way to do this is to set LMC(0)_RLEVEL_RANK*i* = LMC(0)_RLEVEL_RANK*j*, where j is some rank with attached DRAM whose LMC(0)_RLEVEL_RANK*j* is already fully initialized. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:56>	—	RAZ	—	—	Reserved
<55:54>	STATUS	RO/H	0x0	—	Indicates status of the read-leveling and where the BYTE*n* programmings in <53:0> came from: 0x0 = BYTE*n* values are their reset value 0x1 = BYTE*n* values were set via a CSR write to this register 0x2 = read-leveling sequence currently in progress (BYTE*n* values are unpredictable) 0x3 = BYTE*n* values came from a complete read-leveling sequence
<53:48>	BYTE8	R/W/H	0x0	—	Read level setting. BYTE8 is not present on the CN71XX. Software should set LMC(0)_RLEVEL_RANK*[BYTE8] = LMC(0)_RLEVEL_RANK*[BYTE0], using the final BYTE0 value, or set BYTE8 to a value that does not increase the range of possible BYTE* values.
<47:42>	BYTE7	R/W/H	0x0	—	Read level setting.
<41:36>	BYTE6	R/W/H	0x0	—	Read level setting.
<35:30>	BYTE5	R/W/H	0x0	—	Read level setting.
<29:24>	BYTE4	R/W/H	0x0	—	Read level setting.
<23:18>	BYTE3	R/W/H	0x0	—	Read level setting.
<17:12>	BYTE2	R/W/H	0x0	—	Read level setting.
<11:6>	BYTE1	R/W/H	0x0	—	Read level setting.
<5:0>	BYTE0	R/W/H	0x0	—	Read level setting.

Cavium Confidential For Boren Yang ARU 02/27/2014

LMC Read Level Control Register LMC(0)_RLEVEL_CTL

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:24>	PATTERN	R/W	0x55	0x55	Sets the data pattern used to match in read-leveling operations.
<23:22>	—	RAZ	—	—	Reserved
<21>	DELAY_UNLOAD_3	R/W	0	1	When set, unload the PHY silo one cycle later during read-leveling if LMC(0)_RLEVEL_RANK*i*[BYTE*<1:0>] = 0x3. DELAY_UNLOAD_3 should normally be set, particularly at higher speeds.
<20>	DELAY_UNLOAD_2	R/W	0	1	When set, unload the PHY silo one cycle later during read-leveling if LMC(0)_RLEVEL_RANK*i*[BYTE*<1:0>] = 0x2. DELAY_UNLOAD_2 should normally be set.
<19>	DELAY_UNLOAD_1	R/W	0	1	When set, unload the PHY silo one cycle later during read-leveling if LMC(0)_RLEVEL_RANK*i*[BYTE*<1:0>] = 0x1. DELAY_UNLOAD_1 should normally be set.
<18>	DELAY_UNLOAD_0	R/W	0	1	When set, unload the PHY silo one cycle later during read-leveling if LMC(0)_RLEVEL_RANKi[BYTE*<1:0>] = 0x0. DELAY_UNLOAD_0 should normally be set.
<17:10>	BITMASK	R/W	0x0	0x0	Mask to select bit lanes on which read-leveling feedback is returned when OR_DIS is set to 1.
<9>	OR_DIS	R/W	0	0	Disable ORing of bits in a byte lane when computing the read-leveling bitmask. This bit should normally not be set.
<8>	OFFSET_EN	R/W	1	1	When set, LMC attempts to select the read-leveling setting that is LMC(0)_RLEVEL_CTL[OFFSET] settings earlier than the last passing read-leveling setting in the largest contiguous sequence of passing settings. When clear, or if the setting selected by LMC(0)_RLEVEL_CTL[OFFSET] did not pass, LMC selects the middle setting in the largest contiguous sequence of passing settings, rounding earlier when necessary.
<7:4>	OFFSET	R/W	0x2	0x2	The offset used when LMC(0)_RLEVEL_CTL[OFFSET] is set.
<3:0>	BYTE	R/W	0x0	—	0 ≤ BYTE ≤ 8. Byte index for which bitmask results are saved in LMC(0)_RLEVEL_DBG.

Cavium Confidential Bowen Liu 2/27/2014

LMC Read Level Debug Register LMC(0)_RLEVEL_DBG

A given read request to this register returns the read-leveling pass/fail results for all possible delay settings (i.e. the BITMASK) for only one byte in the last rank that the hardware ran read-leveling on. LMC(0)_RLEVEL_CTL[BYTE] selects the particular byte. To get these pass/fail results for a different rank, you must run the hardware read-leveling again. For example, it is possible to get the BITMASK results for every byte of every rank if you run read-leveling separately for each rank, probing LMC(0)_RLEVEL_DBG between each read-leveling. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	BITMASK	RO/H	0x0	0x0	Bitmask generated during read level settings sweep. BITMASK[n] = 0 means read level setting n failed; BITMASK[n] = 1 means read level setting n passed for $0 \le n \le 63$.

LMC Write Level Rank Registers LMC(0)_WLEVEL_RANK(0..3)

Four of these registers exist per LMC, one for each rank. Read level setting is measured in units of 1/8 CK, so the below BYTEn values can range over four CK cycles. Assuming LMC(0)_WLEVEL_CTL[SSET] = 0, the BYTEn<2:0> values are not used during write-leveling, and they are overwritten by the hardware as part of the write-leveling sequence. (Hardware sets STATUS to 3 after hardware write-leveling completes for the rank). Software needs to set BYTEn<4:3> bits.

Each CSR may also be written by software, but not while a write-leveling sequence is in progress. (Hardware sets STATUS to 1 after a CSR write.) Software initiates a hardware write-leveling sequence by programming LMC(0)_WLEVEL_CTL and writing RANKMASK and INIT_START = 1 with SEQUENCE = 6 in LMC(0)_CONFIG.

LMC will then step through and accumulate write leveling results for eight unique delay settings (twice), starting at a delay of LMC(0)_WLEVEL_RANK(0..3)[BYTEn<4:3>]× 8 CK increasing by 1/8 CK each setting. Hardware will then set LMC(0)_WLEVEL_RANKi[BYTEn<2:0>] to indicate the first write-leveling result of 1 that followed a result of 0 during the sequence by searching for a 1100 pattern in the generated bitmask, except that LMC will always write LMC(0)_WLEVEL_RANKi[BYTEn<0>] = 0. If hardware is unable to find a match for a 1100 pattern, then hardware sets LMC(0)_WLEVEL_RANKi[BYTEn<2:0>] to 0x4. See LMC(0)_WLEVEL_CTL.

LMC(0)_WLEVEL_RANKi values for ranks i without attached DRAM should be set such that they do not increase the range of possible BYTE values for any byte lane. The easiest way to do this is to set LMC(0)_WLEVEL_RANKi = LMC(0)_WLEVEL_RANKj, where j is some rank with attached DRAM whose LMC(0)_WLEVEL_RANKj is already fully initialized. See Table 4–7 for the address.

Cavium Confidential For Boyen Yang 09/28/14

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:47>	—	RAZ	—	—	Reserved
<46:45>	STATUS	RO/H	0x0	—	Indicates status of the write-leveling and where the BYTE* programmings in <44:0> came from: 0 = BYTE* values are their reset value. 1 = BYTE* values were set via a register-write operation to this register. 2 = write-leveling sequence currently in progress (BYTE* values are unpredictable). 3 = BYTE* values came from a complete write-leveling sequence, irrespective of which lanes are masked via LMC(0)_WLEVEL_CTL[LANEMASK].
<44:40>	BYTE8	R/W/H	0x0	—	Write level setting. BYTE8 is not present on the CN71XX. Software should set BYTE8 = LMC(0)_RLEVEL_RANK(0..3)[BYTE0], using the final BYTE0 value, or set BYTE8 to a value that does not increase the range of possible BYTE* values.
<39:35>	BYTE7	R/W/H	0x0	—	Write level setting. BYTE7 is not present on the CN71XX. Software should set BYTE7 = LMC(0)_RLEVEL_RANK(0..3)[BYTE0], using the final BYTE0 value, or set BYTE7 to a value that does not increase the range of possible BYTE* values.
<34:30>	BYTE6	R/W/H	0x0	—	Write level setting. Bit 0 of BYTE6 must be zero during normal operation.
<29:25>	BYTE5	R/W/H	0x0	—	Write level setting. Bit 0 of BYTE5 must be zero during normal operation.
<24:20>	BYTE4	R/W/H	0x0	—	Write level setting. Bit 0 of BYTE4 must be zero during normal operation.
<19:15>	BYTE3	R/W/H	0x0	—	Write level setting. Bit 0 of BYTE3 must be zero during normal operation.
<14:10>	BYTE2	R/W/H	0x0	—	Write level setting. Bit 0 of BYTE2 must be zero during normal operation.
<9:5>	BYTE1	R/W/H	0x0	—	Write level setting. Bit 0 of BYTE1 must be zero during normal operation.
<4:0>	BYTE0	R/W/H	0x0	—	Write level setting. Bit 0 of BYTE0 must be zero during normal operation.

Cavium Confidential For Bowen Yang ALU 02/27/2014

LMC Write Level Control Register LMC(0)_WLEVEL_CTL

See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:22>	—	RAZ	—	—	Reserved
<21:19>	RTT_NOM	R/W	0x0	0x0	LMC writes a decoded value to MR1[Rtt_Nom] of the rank during write leveling. Per JEDEC DDR3 specifications, only values MR1[Rtt_Nom] = 1 (RZQ/4), 2 (RZQ/2), or 3 (RZQ/6) are allowed during write leveling with output buffer enabled. 0x0 = LMC writes 001 (RZQ/4) to MR1[Rtt_Nom] 0x1 = LMC writes 010 (RZQ/2) to MR1[Rtt_Nom] 0x2 = LMC writes 011 (RZQ/6) to MR1[Rtt_Nom] 0x3 = LMC writes 100 (RZQ/12) to MR1[Rtt_Nom] 0x4 = LMC writes 101 (RZQ/8) to MR1[Rtt_Nom] 0x5 = LMC writes 110 (Rsvd) to MR1[Rtt_Nom] 0x6 = LMC writes 111 (Rsvd) to MR1[Rtt_Nom] 0x7 = LMC writes 000 (Disabled) to MR1[Rtt_Nom]
<18:11>	BITMASK	R/W	0x0	—	Mask to select bit lanes on which write-leveling feedback is returned when OR_DIS is set to 1.
<10>	OR_DIS	R/W	0	—	Disable ORing of bits in a byte lane when computing the write-leveling bit mask.
<9>	SSET	R/W	0	—	Run write-leveling on the current setting only.
<8:0>	LANEMASK	R/W	0x0	—	One-shot mask to select byte lane to be leveled by the write-leveling sequence. Used with ×16 parts where the upper and lower byte lanes need to be leveled independently.

LMC Write Level Debug Register LMC(0)_WLEVEL_DBG

A given write of this register returns the write-leveling pass/fail results for all possible delay settings (i.e. the BITMASK) for only one byte in the last rank that the hardware write-leveled. LMC(0)_WLEVEL_DBG[BYTE] selects the particular byte.

To get these pass/fail results for a different rank, run the hardware write-leveling again. For example, it is possible to get the BITMASK results for every byte of every rank if you run write-leveling separately for each rank, probing this register between each write-leveling. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved
<11:4>	BITMASK	RO/H	0x0	0x0	Bitmask generated during write level settings sweep. If LMC(0)_WLEVEL_CTL[SSET] = 0, BITMASK<*n*>=0 means write level setting *n* failed; BITMASK<*n*> = 1 means write level setting *n* passed for 0 ≤ *n* ≤ 7. BITMASK contains the first eight results of the total 16 collected by LMC during the write-leveling sequence. If LMC(0)_WLEVEL_CTL[SSET] = 1, BITMASK<0> = 0 means curr write level setting failed; BITMASK<0> = 1 means curr write level setting passed.
<3:0>	BYTE	R/W	0x0	0x0	0 ≤ BYTE ≤ 4.

Cavium Confidential For Bowen Yang @2016

LMC DIMM Control Register
LMC(0)_DIMM_CTL

Note that this register is only used when LMC(0)_CONTROL[RDIMM_ENA] = 1. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:46>	—	RAZ	—	—	Reserved
<45>	PARITY	R/W	0	0	Parity. The Par_In input of a registered DIMM should be tied off. LMC adjusts the value of the DDR_WE_L (DWE#) pin during DDR3 register part control word writes to ensure the parity is observed correctly by the receiving SSTE32882 register part. When Par_In is grounded, PARITY should be cleared to 0.
<44:32>	TCWS	R/W	0x4E0	0x4E0	LMC waits for this time period before and after a RDIMM control word access during a power-up/init SEQUENCE. TCWS is in multiples of 8 CK cycles. Set TCWS (CSR field) = RNDUP[T_{CWS}(ns)/(8 × T_{CYC}(ns))], where T_{CWS} is the desired time (ns), and T_{CYC}(ns) is the DDR clock frequency (not data rate). TYP = 0x4E0 (equivalent to 15µs) when changing clock timing (RC2.DBA1, RC6.DA4, RC10.DA3, RC10.DA4, RC11.DA3, and RC11.DA4) TYP = 0x8, otherwise 0x0 = Reserved
<31:16>	DIMM1_WMASK	R/W	0xFFFF	0xFFFF	Reserved.
<15:0>	DIMM0_WMASK	R/W	0xFFFF	0xFFFF	DIMM0 write mask. If (DIMM0_WMASK[*n*] = 1), write DIMM0.RC*n*.

LMC Scramble Configuration Register 0
LMC(0)_SCRAMBLE_CFG0

Note that address scrambling usually maps addresses into the same rank. Exceptions are when LMC(0)_NXM[CS_MASK] requires aliasing that uses the lowest, legal chip select(s). See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	KEY	R/W/H	0x0	—	Scramble key for data

LMC Scramble Configuration Register 1
LMC(0)_SCRAMBLE_CFG1

Note that address scrambling usually maps addresses into the same rank. Exceptions are when LMC(0)_NXM[CS_MASK] requires aliasing that uses the lowest, legal chip select(s). See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	KEY	R/W/H	0x0	—	Scramble key for addresses

Cavium Confidential For Bowen Yang ALU 02/27/2014

LMC Scrambled Failing Address Register (SEC/DED/NXM) LMC(0)_SCRAMBLED_FADR

This register captures the first transaction with ECC/NXM errors. A DED/NXM error can over-write this register with its failing addresses if the first error was a SEC. If you write LMC(0)_CONFIG→SEC_ERR/DED_ERR/NXM_ERR, it clears the error bits and capture the next failing address. If FDIMM is two, the error is in the higher bits DIMM.

LMC(0)_FADR captures the failing pre-scrambled address location (split into DIMM, bunk, bank, etc.). If scrambling is off, LMC(0)_FADR also captures the failing physical location in the DRAM parts. This register captures the actual failing address location in the physical DRAM parts.

- If scrambling is on, this register contains the failing physical location in the DRAM parts (split into DIMM, bunk, bank, etc);
- If scrambling is off, the pre-scramble and post-scramble addresses are the same; so the contents of this register match the contents of LMC(0)_FADR.

See Table 4–7 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:40>	—	RAZ	—	—	Reserved
<39:38>	FILL_ORDER	RO/H	0x0	0x0	Fill order for failing transaction.
<37>	FDIMM	RO/H	0	0	Failing DIMM number.
<36>	FBUNK	RO/H	0	0	Failing rank.
<35:32>	FBANK	RO/H	0x0	0x0	Failing bank<2:0>.
<31:14>	FROW	RO/H	0x0	0x0	Failing row address<15:0>.
<13:0>	FCOL	RO/H	0x0	0x0	Failing column address<13:0>. Technically, represents the address of the 128-bit data that had an ECC error, i.e., FCOL<0> is always 0. It can be used in conjunction with LMC(0)_CONFIG[DED_ERR] to isolate the 64-bit chunk of data in error.

Cavium Confidential For Bowen Yang ALB 02/27/2014

LMC DIMM Control Register
LMC(0)_DDR4_DIMM_CTL

This register is used only when LMC(0)_CONTROL[RDIMM_ENA] = 1. See Table 4–7 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:22>	—	RAZ	—	—	Reserved
<21:11>	DDR4_DIMM1_WMASK	R/W	0x7FF	0x7FF	Reserved.
<10:0>	DDR4_DIMM0_WMASK	R/W	0x7FF	0x7FF	DIMM0 write mask. If (DIMM0_WMASK[n] = 1), write DIMM0.RC*n*.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Chapter 5

I/O Interconnect (IOI), I/O Bridge (IOB) and Fetch and Add Unit (FAU)

This chapter describes the following topics:

- I/O Interconnect
- IOB Architecture Overview
- Don't-Write-Back Engine
- Fetch and Add Unit (FAU)
- Fetch-and-Add Operations
- I/O Bridge Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

5.1 I/O Interconnect

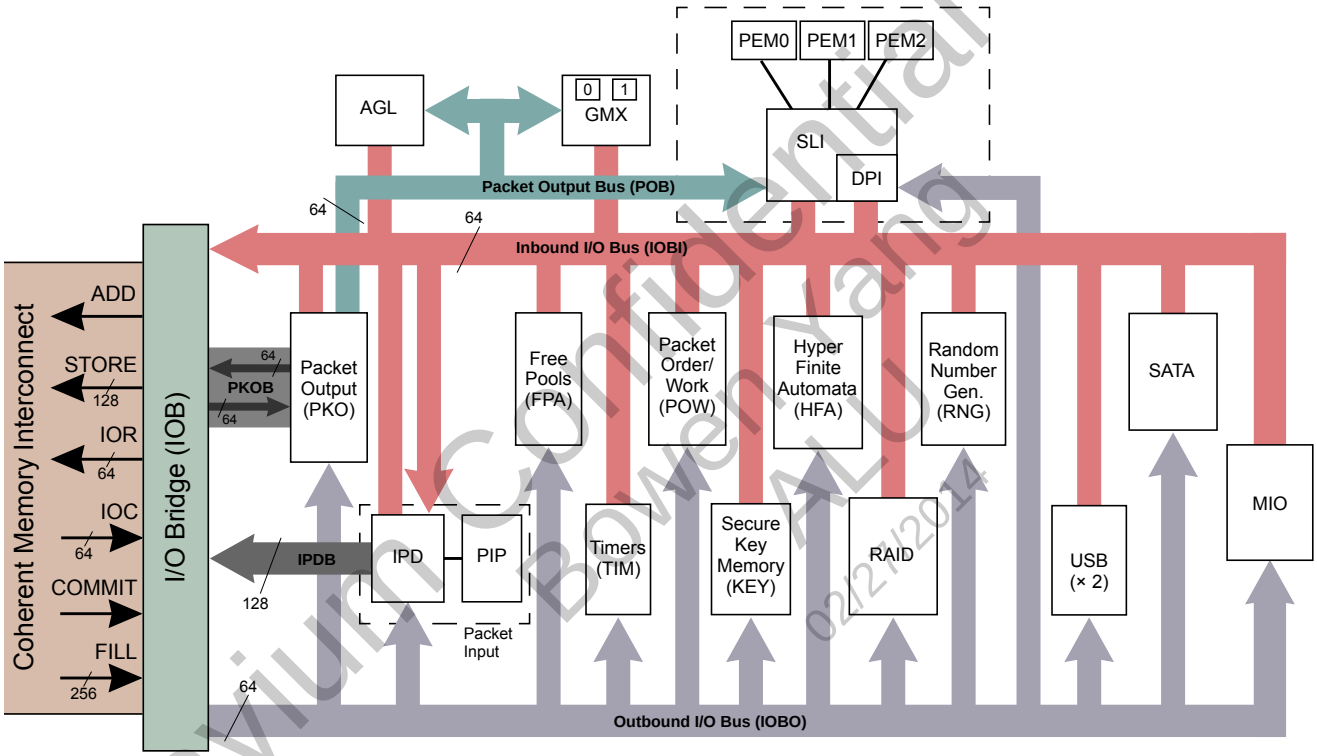
5.1.1 I/O Interconnect Overview

Figure 5–1 shows the I/O and coprocessor components and their physical connections—the I/O interconnect (IOI). The IOI includes the following buses and their associated controls:

- a 64-bit inbound bus (IOBI)
- a 64-bit outbound bus (IOBO)
- a 64-bit packet-outbound bus (POB)
- a 64-bit PKO-specific bus (PKOB)
- a 128-bit input-packet data bus (IPDB).

All the buses run at the coprocessor-clock frequency. I/O and coprocessor devices use these buses to communicate among themselves and, bridged through the I/O Bridge (IOB), to communicate with the L2 Cache (L2C) and cores on the coherent memory interconnect (CMI).

Figure 5–1 IOI Block Diagram



Cavium Confidential FOR Bowen Yang ALU 02/24/2014

5.1.2 IOI Flow Examples

The following figures are examples of transactions on the internal I/O buses to illustrate their usages.

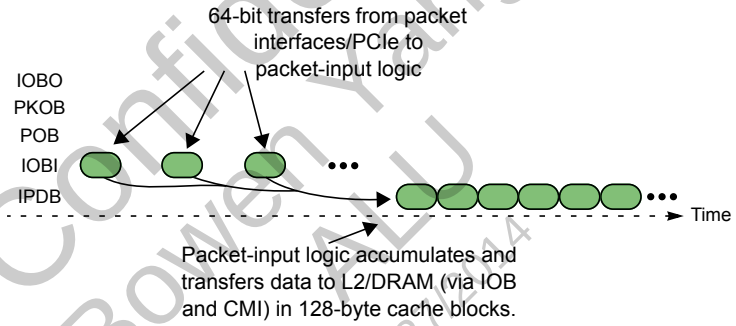
Figure 5–2 shows IOI flows for a packet-data input example. The packet data first arrives at one of the packet interfaces and is captured by either the SGMII/QSGMII (GMX), RGMII (AGL), RXAUI (GMX), or DPI input interfaces, respectively. This is not shown in the figure.

Figure 5–2 does, however, show the packet-input data pieces (each of up to 64-bits) eventually placed by one of the I/O interfaces onto the internal IOBI bus.

The I/O interface also had to previously arbitrate for the IOBI bus cycles, as do all other IOBI drivers. The central-packet input hardware (IPD) latches the packet-data pieces directly from the IOBI bus for processing. There is much processing that occurs, but one particular aspect regarding this example is that the IPD hardware accumulates the 64-bit packet-data pieces from the IOBI bus into full 128-byte cache blocks. IPD then forwards the cache-block write requests to the IOB on the IPDB bus, and the IOB later forwards the cache-block write request onto the CMI.

NOTE: A full 64/128-bit (per coprocessor-clock cycle) data path, using only the IOBI and IPDB buses, exists internally to store input-packet data into L2/DRAM. This gives a raw hardware packet-input processing capability in excess of 30 Gb/s.

Figure 5–2 IOI Flows for a Packet-Data Input



Cavium Confidential FNF
Bowen Yang
ALU
02/21/2014

Figure 5–3 shows IOI flows for packet-data output. The IOB first fills data using the CMI (not shown). An individual fetch operation is up to 128 bytes (the cache-block size). The IOB returns the filled data to the centralized-packet-output hardware (PKO) using the PKOB bus. PKO buffers the PKOB data and eventually forwards it onto the packet output bus (POB), though PKO has large buffers (because it supports in-line TCP/UDP checksum generation), so there may be considerable time delay between the PKOB and POB buses.

NOTE: A full 64-bit (per coprocessor-clock cycle) data path, using only the PKOB and POB buses, exists internally to send packet-output data off-chip. This gives a raw hardware packet-output processing capability in excess of 30 Gb/s. This bandwidth is available independently of the packet-input data flow. This is for internal busing only. External interfaces may limit the sustained bandwidth.

Figure 5–3 IOI Flows for Packet Data Output

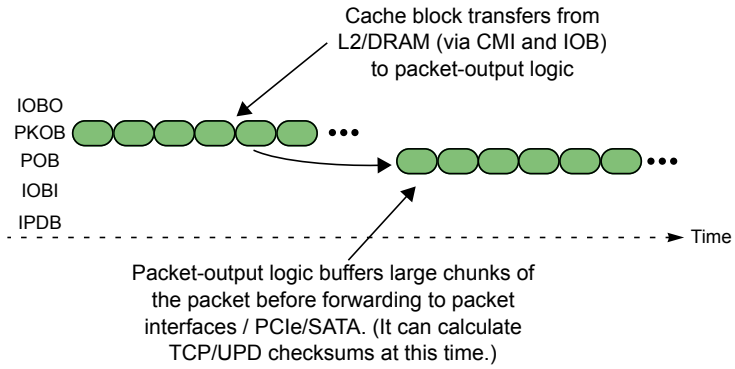


Figure 5–4 has an IOI-flow example that uses the hardware free pools (FPA). This example shows transactions resulting from an allocation request to a free pool. First (not shown), the core issues a load/IOBDMA operation to an address corresponding to FPA, and the address passes through the CMI. Later (shown), the IOB bridges the command and address onto the IOBO bus (in green) and the FPA hardware recognizes that it is the destination for the address. The FPA hardware processes the command, returning the result, (i.e. a pointer to the available memory), later on the IOBI bus (also in green). Figure 5–4 also shows an FPA DMA read access that (infrequently) may be required to replenish the pool of pointers contained in FPA. First, the FPA hardware places the L2/DRAM address onto the IOBI bus (shown in gold). The IOB bridges this address onto the CMI (not shown) and eventually returns the DMA read data containing the additional pointers on the IOBO bus (shown in gold).

Figure 5–4 IOI Flow: Memory Allocate

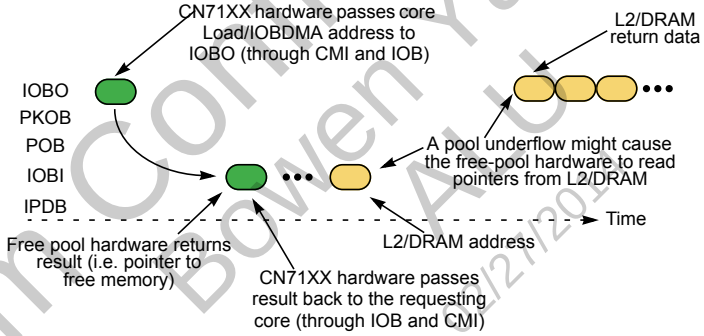
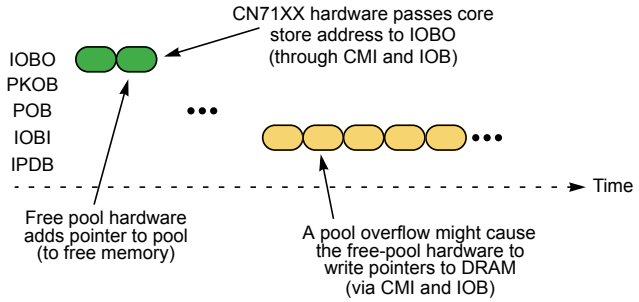


Figure 5–5 has a similar IOI flow example pertaining to the FPA, but this example is of a memory free rather than a memory allocate. The core freeing the memory executes a store to an address and the hardware passes the store through the CMI (not shown). (This example also does not show the Don't-Write-Back (DWB) CMI commands that IOB hardware may create as a result of the memory free command.) The IOB bridges the address/data pair onto the IOBO bus (shown in green), the FPA hardware recognizes it, and buffers the pointer to the available memory in the pool within the FPA block. Figure 5–5 also shows an FPA DMA write access that (infrequently) may be required to free up space in the pool within the FPA block. The FPA hardware places the DMA address and data onto the IOBI bus (shown), which the IOB bridges onto the CMI (not shown).

Cavium Confidential
Bowen Yang
12/21/201

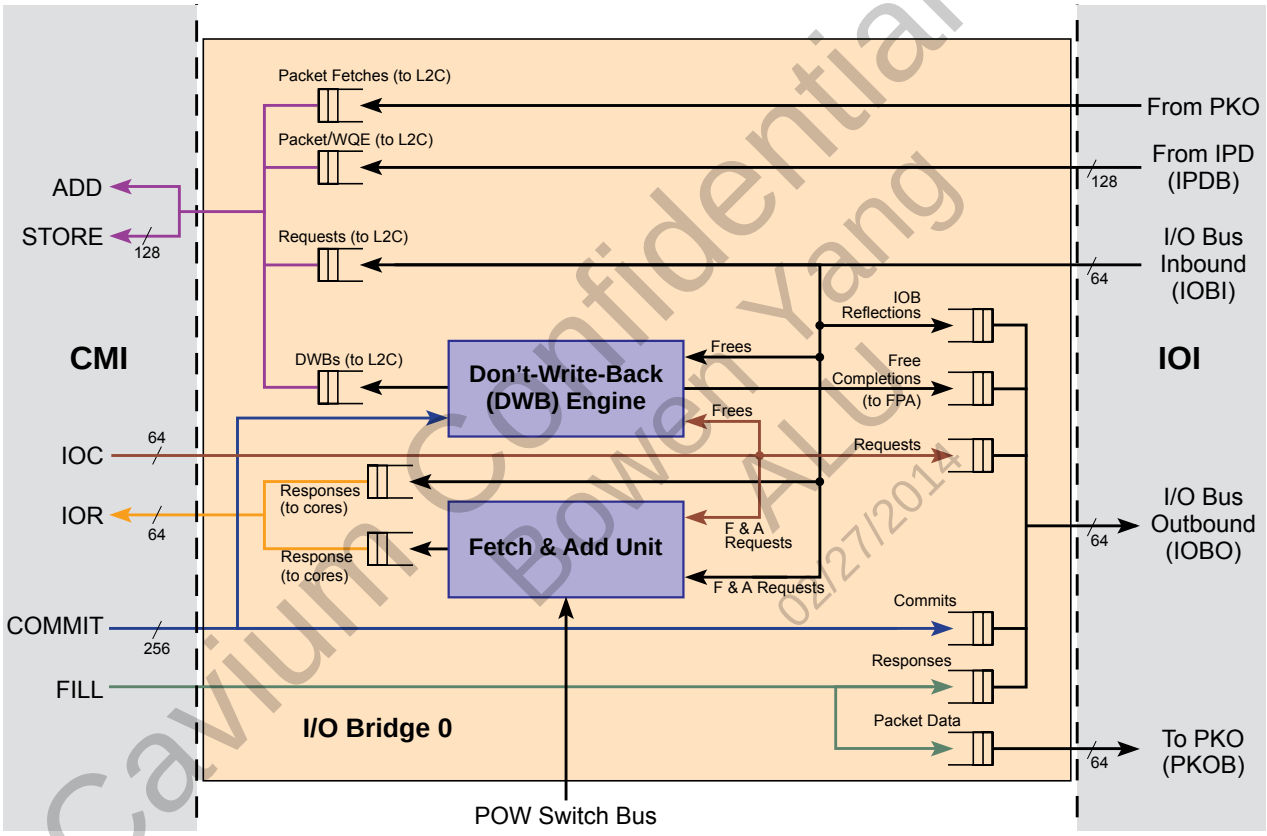
Figure 5–5 IOI Flow: Memory Free



5.2 IOB Architecture Overview

Figure 5–6 shows a block diagram of the IOB internal architecture. The CMI is on the left side of the figure and the I/O buses are in the right, the top, and the bottom of the figure.

Figure 5–6 I/O Bridge Block Diagram



Cavium Confidential For Bowen Yang ALU 02/27/2014

The primary function of the IOB is a bridge, so the largest components of the IOB architecture are the queues to hold information transferred on the different buses. There are six queues arbitrating to transfer on the ADD/STORE/IOR buses of the CMI and five queues arbitrating to transfer on the IOBO bus. The multiple queues are required for a number of reasons: deadlock-avoidance, different data sources, and performance.

5.3 Don't-Write-Back Engine

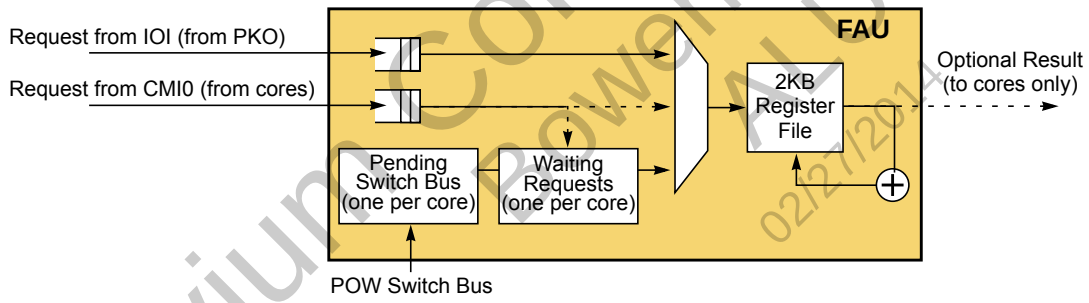
Another important component within the IOB is the DWB engine. This engine intercepts memory free requests destined to the free-pool hardware (FPA). The memory free requests include a hint indicating the number of DWB CMI transactions that the IOB hardware can issue. The memory has just been freed and it will not be used until it is reallocated, so it would be wasteful for the hardware to write the cache blocks from the level-2 cache back to the DRAM. The DWB commands causes the dirty bit for the selected blocks to be cleared, thus avoiding these wasteful write-backs.

The IOB can intercept these memory free commands arriving from either the cores (via the CMI IOC bus) or from other hardware units (via the IOBI bus). The IOB can buffer a limited number of the memory free commands inside DWB. If buffering is available, the IOB intercepts the memory free request until it has finished issuing the CMI DWB commands for the request, and then forwards the request onto the FPA hardware (via the IOBO bus). If the DWB buffering is not available, the IOB DWB unit does not intercept the memory free command, and the command instead goes to the FPA hardware without the FPA issuing any DWB commands. This implementation is possible since it is optional for the hardware to issue the DWBs.

5.4 Fetch and Add Unit (FAU)

Figure 5–7 expands on the final important component of the IOB architecture—the Fetch and Add Unit (FAU). The FAU is a 2KB register file supporting read, write, atomic fetch-and-add, and atomic-update operations. The unit can be accessed from both the cores and the centralized packet-output unit (PKO). The cores use the FAU for general synchronization purposes, including applications like assigning unique positions for inserting items into shared queues.

Figure 5–7 FAU Block Diagram



Cavium Confidential For Bowen Yang ALU 02/21/2014

FAU registers are used particularly to maintain lengths of the output queues (up to 256) to transfer packets off-chip (via PKO). This is the reason that the PKO submits commands to the FAU. Core software increments counters in the FAU when it adds a packet to a queue. Under software command, the PKO decrements the same counter after it removes the packet from a queue and sends it off-chip. This way, the register continually maintains the size of the output queue (in packets and/or bytes). The core software will require the current queue size if it must implement quality-of-service (QOS) algorithms like Random Early Discard (RED).

Another feature of the FAU is its tight integration with the tag switch synchronization provided by the Packet Order / Work Unit (POW). This tight integration can minimize critical section latencies when both tag switch and FAU synchronization is required.

The FAU tracks the pending tag switch state for each core. The FAU considers a tag switch to be pending from when the POW tag switch store (that uses the POW DID of 0x60) issues until the switch completion indication returns via the POW switch bus. This FAU pending switch logic is, in general, a conservative estimator of the pending switch state held at both the cores and POW. For switches that are not descheduled, it tracks exactly.

Each core can specify the “tag wait” feature for up to one outstanding FAU request at any time. (The number of outstanding FAU requests that do not specify the tag wait feature is never restricted by the hardware.) When this option is enabled, the FAU attempts to delay the request until the prior tag switch is no longer pending. While the hardware delays the request, it buffers it in a store that contains one entry per core. Thus, the restriction on only one outstanding tag-wait request from each core.

The FAU processes requests without the tag-wait feature enabled immediately. It also processes requests with the preferred tag wait requests immediately when (it thinks) there is not a pending switch. If the hardware buffers a tag-wait request for too long, it removes the request from the buffer and returns a failure indication to the requestor without performing the operation specified in the request. The tag wait time-out length is variable based on configuration, in multiples of 256 internal clock cycles.

Figure 5–7 shows all the components of the fetch-and-add hardware. The hardware arbitrates between the three different sources of requests; the cores, the PKO, and the buffered core tag-wait requests. The hardware completes the requests by reading and writing the 2 KB register file. Some core-generated FAU requests require responses to be reflected back through the CMI.

Chapter 10 describes the software interface to the PKO.

Cavium Confidential For Bowen Yang ALU 02/7/2014

5.5 Fetch-and-Add Operations

Core operations to the FAU issue through Load, Store, and IOBDMA instructions.

For load and store operations, the FAU operation size matches the size specified by the instruction. For example, a core byte-load instruction (i.e. LB or LBU) initiates a byte operation on the register file. This means that the register value can change, and the load result is returned, only for the particular byte in the register file referenced by the LB/LBU instruction. All the core load/store instruction sizes can be used to operate on the fetch-and-add register file:

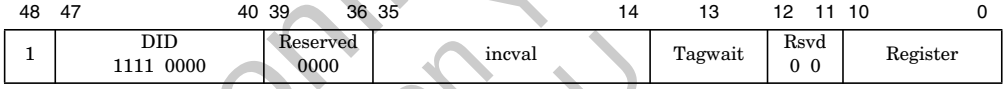
- 8-bit operation (LB/LBU/SB)
- 16-bit operation (LH/LHU/SH)
- 32-bit operation (LW/LWU/SW)
- 64-bit operation (LD/SD)

IOBDMA operations destined for the FAU always return a 64-bit result to the core, but the actual operation performed by the hardware, and the effective result returned, may be either 8-bit, 16-bit, 32-bit, or 64-bit.

The configuration of the particular core that executes the load/store selects the endian-ness of the 8-bit, 16-bit, and 32-bit loads/stores. The FAU has a IOB_CTL_STATUS[FAU_END] bit that can specify either little-endian or big-endian addressing for IOBDMA operations. This IOBDMA endian configuration value is common to all cores.

5.5.1 Load Operations

Load Physical Address for FAU Operations



- **incval** - The value to add to the register after returning the load result. This is a 22-bit signed value, of which only the bottom 8-bits are used for 8-bit operations and the bottom 16-bits for 16-bit operations.
- **Tagwait** - If set, the hardware attempts to delay servicing the request until after the prior tag switch completes.
- **Register** - Selects a particular FAU register.
  - **<10:3>** selects the register
  - **<2:0>** selects a byte offset into that register

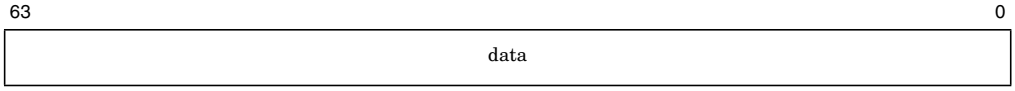
NOTE: Register field must be naturally aligned to the size. This is required by the core.

NOTE: The value returned for a load or IOBDMA is the old value from the previous add operation.

Cavium Confidential For Bowen Yang ALU 02/27/2014

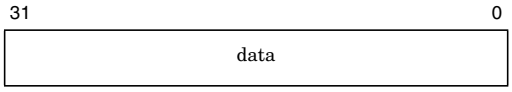
Load Operation Result In Cases Where Tagwait = 0:

64-bit operation result for a LD



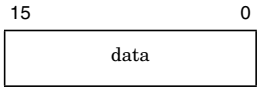
- **data** - data is current value of register location.

32-bit operation result for a LW/LWU



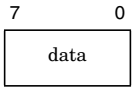
- **data** - data is current value of register location.

16-bit operation result for a LH/LHU



- **data** - data is current value of register location.

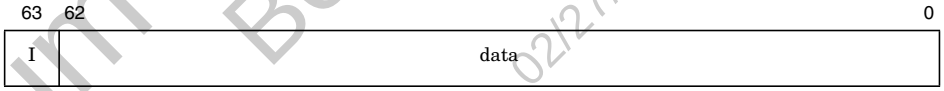
8-bit operation result for a LB/LBU



- **data** - data is current value of register location.

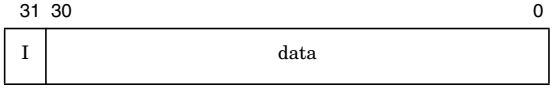
Load Operation Result In Cases Where Tagwait = 1:

64-bit operation result for a LD



- **I** - Operation did not complete.
- **data** - Unpredictable if I is set. Otherwise, data is current value of the lower 63-bits of the register location.

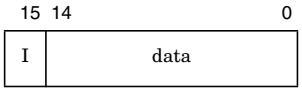
32-bit operation result for a LW/LWU



- **I** - Operation did not complete.
- **data** - unpredictable if I is set. Otherwise, data is current value of the lower 31-bits of the register location.

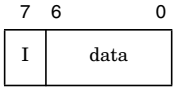
Cavium Confidential For
Bowen Yang
ALU
02/27/2014

16-bit operation result for a LH/LHU



- **I** - Operation did not complete.
- **data** - unpredictable if I is set. Otherwise, data is current value of the lower 15-bits of the register location.

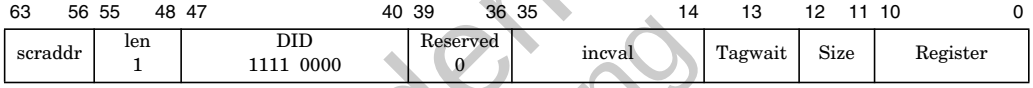
8-bit operation result for a LB/LBU



- **I** - Operation did not complete.
- **data** - unpredictable if I is set. Otherwise, data is current value of the lower 7 bits of the register location.

5.5.2 IOBDMA Operations

IOBDMA Store Data for FAU Operations



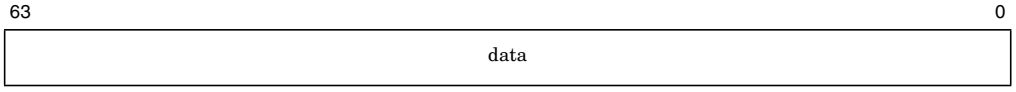
- **scraddr** - Refer to Section 2.7.
- **len** - Must be 1. Refer to Section 2.7.
- **incval** - The value to add to the register. This is a 22-bit signed value, of which only the bottom 8 bits are used for 8-bit operations and the bottom 16-bits for 16-bit operations.
- **Tagwait** - If set, the hardware will attempt to delay servicing the request until after the prior tag switch completes.
- **Size** - Indicates the size of the op

  0x0 = 8-bit
  0x1 = 16-bit
  0x2 = 32-bit
  0x3 = 64-bit

- **Register** - Selects a particular FAU register.

NOTE: Register field must be naturally aligned to the size.

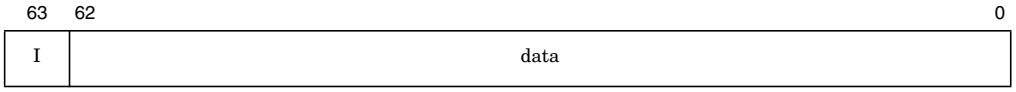
64-bit operation (tagwait=0) result for an IOBDMA (in CVMSEG_LM)



- **data** - data is current value of register location.

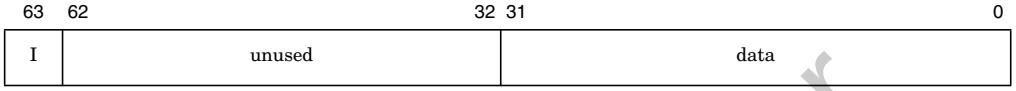
Cavium Confidential For Bowen Yang ALU 02/27/2014

64-bit operation (tagwait=1) result for an IOBDMA (in CVMSEG_LM)



- **I** - Incomplete. When I = 1, the operation did not complete and the data is unpredictable.
- **data** - Unpredictable when I = 1. Otherwise, data is current value of the lower 63 bits of the register location.

32-bit operation result for an IOBDMA (in CVMSEG_LM)

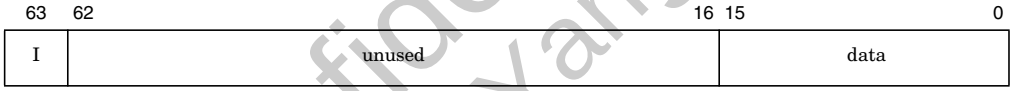


- **I** - Incomplete. The hardware always forces I = 0 when tagwait = 0. When tagwait = 1, then I may be 1 or 0.

  I = 1, the operation did not complete and the data is unpredictable.
  I = 0, the operation completed.

- **unused** - Unpredictable when I = 1; otherwise all 0s.
- **data** - Unpredictable when I = 1. Otherwise, data is the current value of the 32-bit register.

16-bit operation result for an IOBDMA (in CVMSEG_LM)

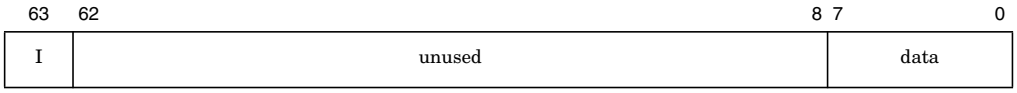


- **I** - Incomplete. The hardware always forces I = 0 when tagwait = 0. When tagwait = 1, then I may be 1 or 0.

  I = 1, the operation did not complete and the data is unpredictable.
  I = 0, the operation completed.

- **unused** - Unpredictable when I = 1; otherwise all 0s.
- **data** - Unpredictable when I = 1. Otherwise, data is the current value of the 16-bit register.

8-bit operation result for an IOBDMA (in CVMSEG_LM)



- **I** - Incomplete. The hardware always forces I = 0 when tagwait = 0. When tagwait = 1, then I may be 1 or 0.

  I = 1, the operation did not complete and the data is unpredictable.
  I = 0, the operation completed.

- **unused** - Unpredictable when I = 1; otherwise all 0s.
- **data** - Unpredictable when I = 1. Otherwise, data is the current value of the 8-bit register.

Cavium Confidential For Bowen Yang ALU 02/27/2014

5.5.3 Store Operations

Store Physical Address for FAU Operations

48	47 40	39 14	13	12 11	10 0
1	DID 1111 0000	Reserved 0	noadd	Rsvd 0	Register

- **noadd** - If clear, do an (atomic) update, else just store
- **Register** - Selects a particular FAU register.

NOTE: Register field must be naturally aligned to the size. This is required by the core.

The store address above can be used for any of the core 8-bit (SB), 16-bit (SH), 32-bit (SW), or 64-bit (SD) store instructions to produce FAU operations of the corresponding size. The no add bit selects whether to add the store data to the previous register value or to simply over-write the prior register value.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

5.6 I/O Bridge Registers

The I/O Bridge registers are listed in Table 5–1.

Table 5–1 IOB Registers

Register	Address	CSR Type[1]	Detailed Description
IOB_FAU_TIMEOUT	0x00011800F0000000	RSL	See page 354
IOB_N2C_RSP_PRI_CNT	0x00011800F0000008	RSL	See page 354
IOB_I2C_PRI_CNT	0x00011800F0000010	RSL	See page 354
IOB_P2C_REQ_PRI_CNT	0x00011800F0000018	RSL	See page 355
IOB_N2C_L2C_PRI_CNT	0x00011800F0000020	RSL	See page 355
IOB_DWB_PRI_CNT	0x00011800F0000028	RSL	See page 355
IOB_OUTB_REQ_PRI_CNT	0x00011800F0000038	RSL	See page 355
IOB_OUTB_COM_PRI_CNT	0x00011800F0000040	RSL	See page 356
IOB_OUTB_FPA_PRI_CNT	0x00011800F0000048	RSL	See page 356
IOB_CTL_STATUS	0x00011800F0000050	RSL	See page 356
IOB_INT_SUM	0x00011800F0000058	RSL	See page 357
IOB_INT_ENB	0x00011800F0000060	RSL	See page 357
IOB_PKT_ERR	0x00011800F0000068	RSL	See page 358
IOB_INB_DATA_MATCH	0x00011800F0000070	RSL	See page 358
IOB_INB_CONTROL_MATCH	0x00011800F0000078	RSL	See page 358
IOB_INB_DATA_MATCH_ENB	0x00011800F0000080	RSL	See page 358
IOB_INB_CONTROL_MATCH_ENB	0x00011800F0000088	RSL	See page 359
IOB_OUTB_DATA_MATCH	0x00011800F0000090	RSL	See page 359
IOB_OUTB_CONTROL_MATCH	0x00011800F0000098	RSL	See page 359
IOB_OUTB_DATA_MATCH_ENB	0x00011800F00000A0	RSL	See page 359
IOB_OUTB_CONTROL_MATCH_ENB	0x00011800F00000A8	RSL	See page 360
IOB_TO_CMB_CREDITS	0x00011800F00000B0	RSL	See page 360
IOB_PP_BIST_STATUS	0x00011800F0000700	RSL	See page 360
IOB_BIST_STATUS	0x00011800F00007F8	RSL	See page 361
IOB_CHIP_GLB_PWR_THROTTLE	0x00011800F0000808	RSL	See page 361
IOB_CHIP_PWR_OUT	0x00011800F0000818	RSL	See page 362
IOB_CHIP_CUR_PWR	0x00011800F0000828	RSL	See page 363

1. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Zang at 9.27.21

Fetch and Add Unit Tag-Switch Time-Out Register IOB_FAU_TIMEOUT

Enables and supplies the number if clock ticks the FAU unit will wait for a tag-switch before timing out. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	—	—	Reserved
<12>	TOUT_ENB	R/W	1	0	Enable FAU time-out: 1 = enable the time-out, 0 = disable.
<11:0>	TOUT_VAL	R/W	0x4	0x4	When a tag request arrives from the core, a timer is started associated with that core. The timer, which increments every 256 coprocessor-clock cycles, is compared to TOUT_VAL. When the two are equal, the IOB flags the tag request to complete as a time-out tag operation. The 256-count timer used to increment the core-associated timer is always running, so the first increment of the core-associated timer may occur anywhere within the first 256 coprocessor-clock cycles. **NOTE:** Zero is an illegal value.

IOI To CMI Response Priority Counter IOB_N2C_RSP_PRI_CNT

Enables and supplies the time-out count for raising the priority of IOI responses access to the coherent memory interconnect (CMI). See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	CNT_ENB	R/W	0	0	Access-priority enable. Enables the raising of CMI-access priority when CNT_VAL is reached.
<14:0>	CNT_VAL	R/W	0x0	0x0	Count value. The number of coprocessor-clock cycles to wait before raising the priority for access to CMI.

IPD To CMI Store Priority Counter IOB_I2C_PRI_CNT

Enables and supplies the time-out count for raising the priority of IPD store access to the CMI. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	CNT_ENB	R/W	0	0	Access-priority enable. Enables the raising of CMI-access priority when CNT_VAL is reached.
<14:0>	CNT_VAL	R/W	0x0	0x0	Count value. The number of coprocessor-clock cycles to wait before raising the priority for access to CMI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PKO To CMI Response Priority Counter
IOB_P2C_REQ_PRI_CNT

Enables and supplies the time-out count for raising the priority of PKO load access to the CMI. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	CNT_ENB	R/W	0	0	Access-priority enable. Enables the raising of CMI-access priority when CNT_VAL is reached.
<14:0>	CNT_VAL	R/W	0x0	0x0	Count value. The number of coprocessor-clock cycles to wait before raising the priority for access to CMI.

IOI To CMI L2C Priority Counter
IOB_N2C_L2C_PRI_CNT

Enables and supplies the time-out count for raising the priority of IOI store/load access to the CMI. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	CNT_ENB	R/W	0	0	Access-priority enable. Enables the raising of CMI-access priority when CNT_VAL is reached.
<14:0>	CNT_VAL	R/W	0x0	0x0	Count value. The number of coprocessor-clock cycles to wait before raising the priority for access to CMI.

Don’t-Write-Back to CMI Priority Counter
IOB_DWB_PRI_CNT

Enables and supplies the time-out count for raising the priority of don't-write-back (DWB) requests to the L2C. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	CNT_ENB	R/W	0	0	Access-priority enable. Enables the raising of CMI-access priority when CNT_VAL is reached.
<14:0>	CNT_VAL	R/W	0x0	0x0	Count value. The number of coprocessor-clock cycles to wait before raising the priority for access to CMI.

Request to IOI Priority Counter
IOB_OUTB_REQ_PRI_CNT

Enables and supplies the time-out count for raising the priority of request transfers to the outbound IOI. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	CNT_ENB	R/W	0	0	Access-priority enable. Enables the raising of CMI-access priority when CNT_VAL is reached.
<14:0>	CNT_VAL	R/W	0x0	0x0	Count value. The number of coprocessor-clock cycles to wait before raising the priority for access to CMI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Commit to IOI Priority Counter
IOB_OUTB_COM_PRI_CNT

Enables and supplies the time-out count for raising the priority of Commit request to the outbound IOI. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	CNT_ENB	R/W	0	0	Access-priority enable. Enables the raising of CMI-access priority when CNT_VAL is reached.
<14:0>	CNT_VAL	R/W	0x0	0x0	Count value. The number of coprocessor-clock cycles to wait before raising the priority for access to CMI.

FPA Returns to IOI Priority Counter
IOB_OUTB_FPA_PRI_CNT

Enables and supplies the time-out count for raising the priority of FPA return page request to the outbound IOI. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	CNT_ENB	R/W	0	0	Access-priority enable. Enables the raising of CMI-access priority when CNT_VAL is reached.
<14:0>	CNT_VAL	R/W	0x0	0x0	Count value. The number of coprocessor-clock cycles to wait before raising the priority for access to CMI.

I/O Bridge Control and Status Register
IOB_CTL_STATUS

This register provides control and status information for IOB functions. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved
<9:6>	XMC_PER	R/W	0x0	0x0	IBC XMC PUSH EARLY
<5>	RR_MODE	R/W	0	0	When set to 1, enables round-robin mode of next transaction that could arbitrate for the address/command bus (ADD).
<4:3>	—	RAZ	—	—	Reserved
<2>	PKO_ENB	R/W	0	0	Toggles the endian style of the FAU for the PKO: 0 = big-endian, 1 = little-endian.
<1>	DWB_ENB	R/W	1	1	Enables the don't-write-back function of the IOB.
<0>	FAU_END	R/W	0	0	Toggles the endian style of the FAU: 0 = big-endian, 1 = little-endian.

Cavium Confidential For Bowen Yang ALU 02/27/2014

I/O Bridge Interrupt-Summary Register IOB_INT_SUM

This register provides the interrupt-summary bits of the IOB. Note that only the first detected error associated with bits [5:0] is set. Another bit can be set only after the previously reported bit has been cleared. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7>	OUTB_MAT	R/W1C/H	0	0	This bit is set to 1 when there was a match on the outbound bus to the outbound pattern matchers.
<6>	INB_MAT	R/W1C/H	0	0	This bit is set to 1 when there was a match on the outbound bus to the inbound pattern matchers.
<5>	P_DAT	R/W1C/H	0	0	This bit is set to 1 when a data arrives before a SOP for the same port for a pass-through packet.
<4>	NP_DAT	R/W1C/H	0	0	This bit is set to 1 when a data arrives before a SOP for the same port for a non-pass-through packet.
<3>	P_EOP	R/W1C/H	0	0	Pass-through EOP. This bit is set to 1 when an EOP is followed by an EOP for the same port for a pass-through packet.
<2>	P_SOP	R/W1C/H	0	0	Pass-through SOP. This bit is set to 1 when an SOP is followed by an SOP for the same port for a pass-through packet.
<1>	NP_EOP	R/W1C/H	0	0	Non-pass-through EOP. This bit is set to 1 when an EOP is followed by an EOP for the same port for a non-pass-through packet.
<0>	NP_SOP	R/W1C/H	0	0	Non-pass-through SOP. This bit is set to 1 when an SOP is followed by an SOP for the same port for a non-pass-through packet.

I/O Bridge Interrupt-Enable Register IOB_INT_ENB

This register provides the interrupt-enable bits of the IOB. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7>	OUTB_MAT	R/W	0	0	When set to 1 and IOB_INT_SUM[OUTB_MAT] is asserted, the IOB generates an interrupt.
<6>	INB_MAT	R/W	0	0	When set to 1 and IOB_INT_SUM[INB_MAT] is asserted, the IOB generates an interrupt.
<5>	P_DAT	R/W	0	0	When set to 1 and IOB_INT_SUM[P_DAT] is asserted, the IOB generates an interrupt.
<4>	NP_DAT	R/W	0	0	When set to 1 and IOB_INT_SUM[NP_DAT] is asserted, the IOB generates an interrupt.
<3>	P_EOP	R/W	0	0	Pass-through EOP enable. When set to 1 and IOB_INT_SUM[P_EOP] is asserted, IOB generates an interrupt.
<2>	P_SOP	R/W	0	0	Pass-through SOP enable. When set to 1 and IOB_INT_SUM[S_EOP] is asserted, IOB generates an interrupt.
<1>	NP_EOP	R/W	0	0	Non-pass-through EOP enable. When set to 1 and IOB_INT_SUM[NP_EOP] is asserted, IOB generates an interrupt.
<0>	NP_SOP	R/W	0	0	Non-pass-through SOP enable. When set to 1 and IOB_INT_SUM[NP_SOP] is asserted, IOB generates an interrupt.

Cavium Confidential For Bowen Yang ALU 12/27/2014

I/O Bridge Packet-Error Register
IOB_PKT_ERR

This register provides port status about the failing packet-receive error. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved
<11:6>	VPORT	RO/H	0x0	—	When one of the IOB_INT_SUM[3:0] is set, this field latches the failing vport associated with the error.
<5:0>	PORT	RO/H	0x0	—	Failing port. When one of IOB_INT_SUM[3:0] is set, this field latches the failing port associated with the error.

I/O Bridge Inbound Data-Match Register
IOB_INB_DATA_MATCH

This register provides the match pattern for inbound IOI data. When inbound data matches the enabled bits of this pattern (see IOB_INB_DATA_MATCH_ENB), IOB_CTL_STATUS[INB_MAT] is set to 1. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	DATA	R/W/H	0x0	—	Pattern to match on the inbound IOI.

I/O Bridge Inbound Control-Match Register
IOB_INB_CONTROL_MATCH

This register provides the match pattern for inbound IOI control bits. When inbound control matches the enabled bits of this pattern (see IOB_INB_CONTROL_MATCH_ENB), IOB_CTL_STATUS[INB_MAT] is set to 1. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:29>	—	RAZ	—	—	Reserved
<28:21>	MASK	R/W/H	0x0	—	Pattern to match on the inbound IOI.
<20:17>	OPC	R/W/H	0x0	—	Pattern to match on the inbound IOI.
<16:8>	DST	R/W/H	0x0	—	Pattern to match on the inbound IOI.
<7:0>	SRC	R/W/H	0x0	—	Pattern to match on the inbound IOI.

I/O Bridge Inbound Data-Match-Enable Register
IOB_INB_DATA_MATCH_ENB

This register enables the match of the corresponding bit in IOB_INB_DATA_MATCH. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	DATA	R/W/H	0x0	—	When a bit is set to 1 in this register, it enables the corresponding bit in IOB_INB_DATA_MATCH to be matched.

Cavium Confidential For Bowen Yang ALU 02/12/14

I/O Bridge Inbound Control-Match-Enable Register IOB_INB_CONTROL_MATCH_ENB

This register enables the match of the corresponding bit in IOB_INB_CONTROL_MATCH. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:29>	—	RAZ	—	—	Reserved
<28:21>	MASK	R/W/H	0x0	—	When a bit is set to 1 in this field, it enables the corresponding bit in IOB_INB_CONTROL_MATCH[MASK] to be matched.
<20:17>	OPC	R/W/H	0x0	—	When a bit is set to 1 in this field, it enables the corresponding bit in IOB_INB_CONTROL_MATCH[OPC] to be matched.
<16:8>	DST	R/W/H	0x0	—	When a bit is set to 1 in this field, it enables the corresponding bit in IOB_INB_CONTROL_MATCH[DST] to be matched.
<7:0>	SRC	R/W/H	0x0	—	When a bit is set to 1 in this field, it enables the corresponding bit in IOB_INB_CONTROL_MATCH[SRC] to be matched.

I/O Bridge Outbound Data-Match Register IOB_OUTB_DATA_MATCH

This register provides the match pattern for outbound IOI data. When outbound data matches the enabled bits of this pattern (see IOB_OUTB_DATA_MATCH_ENB), IOB_CTL_STATUS[OUTB_MAT] is set to 1. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	DATA	R/W/H	0x0	—	Pattern to match on the outbound IOI.

I/O Bridge Outbound Control-Match Register IOB_OUTB_CONTROL_MATCH

This register provides the match pattern for outbound IOI control bits. When outbound control matches the enabled bits of this pattern (see IOB_OUTB_CONTROL_MATCH_ENB), IOB_CTL_STATUS[OUTB_MAT] is set to 1. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:26>	—	RAZ	—	—	Reserved
<25:18>	MASK	R/W/H	0x0	—	Pattern to match on the outbound IOI.
<17>	EOT	R/W/H	0	—	Pattern to match on the outbound IOI.
<16:9>	DST	R/W/H	0x0	—	Pattern to match on the outbound IOI.
<8:0>	SRC	R/W/H	0x0	—	Pattern to match on the outbound IOI.

I/O Bridge Outbound Data-Match-Enable Register IOB_OUTB_DATA_MATCH_ENB

This register enables the match of the corresponding bit in IOB_OUTB_DATA_MATCH. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	DATA	R/W/H	0x0	—	When a bit is set to 1 in this register, it enables the corresponding bit in IOB_OUTB_DATA_MATCH to be matched.

Cavium Confidential For Bowen Yang ALU 02/27/2014

I/O Bridge Outbound Control-Match-Enable Register IOB_OUTB_CONTROL_MATCH_ENB

This register enables the match of the corresponding bit in IOB_OUTB_CONTROL_MATCH. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:26>	—	RAZ	—	—	Reserved
<25:18>	MASK	R/W/H	0x0	—	When a bit is set to 1 in this field, it enables the corresponding bit in IOB_OUTB_CONTROL_MATCH[MASK] to be matched.
<17>	EOT	R/W/H	0	—	When a bit is set to 1 in this field, it enables the corresponding bit in IOB_OUTB_CONTROL_MATCH[EOT] to be matched.
<16:9>	DST	R/W/H	0x0	—	When a bit is set to 1 in this field, it enables the corresponding bit in IOB_OUTB_CONTROL_MATCH[DST] to be matched.
<8:0>	SRC	R/W/H	0x0	—	When a bit is set to 1 in this field, it enables the corresponding bit in IOB_OUTB_CONTROL_MATCH[SRC] to be matched.

I/O Bridge-to-CMI Credits Register IOB_TO_CMB_CREDITS

This register controls the number of read and write operations that may be outstanding to the L2C (via the CMI). Possible values are 1–8, with 0x0 setting 8 credits. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8:6>	PKO_RD	R/W/H	0x0	—	Number of PKO read operations that can be out to L2C.
<5:3>	NCB_RD	R/W/H	0x0	—	Number of IOBI read operations that can be out to L2C.
<2:0>	NCB_WR	R/W/H	0x0	—	Number of IOBI / PIP/IPD write operations that can be out to L2C.

IOB BIST Cores Status Register IOB_PP_BIST_STATUS

This register controls the number of read and write operations that may be outstanding to the L2C (via the CMI). Possible values are 1–8, with 0x0 setting 8 credits. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	PP_BSTAT	RO	0x0	0x0	BIST Status of the cores. Bit vector position is the number of the core (i.e. core 0 = PP_BSTAT[0]).

Cavium Confidential For Bowen Yang ALU 02/27/2014

IOB Memories BIST Status Register
IOB_BIST_STATUS

This register shows the result of the BIST run on the IOB memories. See Table 5–1 for address. A 1 in any of the bits indicates a BIST error. A 0 indicates the BIST passed or never ran.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:23>	—	RAZ	—	—	Reserved
<22>	XMDFIF	RO	0	0	xmdfif_bist_status
<21>	XMCFIF	RO	0	0	xmcfif_bist_status
<20>	IORFIF	RO	0	0	iorfif_bist_status
<19>	RSDFIF	RO	0	0	rsdfif_bist_status
<18>	IOCFIF	RO	0	0	iocfif_bist_status
<17>	ICNRCB	RO	0	0	icnr_cb_reg_fifo_bist_status
<16>	ICR0	RO	0	0	icr_bist_req_fifo0_status
<15>	ICR1	RO	0	0	icr_bist_req_fifo1_status
<14>	ICNR1	RO	0	0	Reserved
<13>	ICNR0	RO	0	0	icnr_reg_mem0_bist_status
<12>	IBDR0	RO	0	0	ibdr_bist_req_fifo0_status
<11>	IBDR1	RO	0	0	Reserved
<10>	IBR0	RO	0	0	ibr_bist_rsp_fifo0_status
<9>	IBR1	RO	0	0	ibr_bist_rsp_fifo1_status
<8>	ICNRT	RO	0	0	icnr_tag_cb_reg_fifo_bist_status
<7>	IBRQ0	RO	0	0	ibrq_bist_req_fifo0_status
<6>	IBRQ1	RO	0	0	ibrq_bist_req_fifo1_status
<5>	ICRN0	RO	0	0	icr_ncb_bist_mem0_status
<4>	ICRN1	RO	0	0	icr_ncb_bist_mem1_status
<3>	ICRP0	RO	0	0	icr_pko_bist_mem0_status
<2>	ICRP1	RO	0	0	icr_pko_bist_mem1_status
<1>	IBD	RO	0	0	ibd_bist_mem0_status
<0>	ICD	RO	0	0	icd_ncb_fifo_bist_status

IOB Chip Global Power Throttle Register
IOB_CHIP_GLB_PWR_THROTTLE

This register controls the min/max power settings. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:34>	—	RAZ	—	—	Reserved
<33:32>	PWR_BW	R/W	0x0	0x0	Configures the reaction time of the closed-loop feedback control system for the AVG_CHIP_POWER power approximation. Higher numbers decrease bandwidth, reducing response time, which could lead to greater tracking error, but reduce ringing.
<31:24>	PWR_MAX	R/W	0xFF	0xFF	Maximum allowed CURRENT_POWER_SETTING value. PWR_MAX must be ≥ PWR_MIN.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<23:16>	PWR_MIN	R/W	0x0	—	Minimum allowed CURRENT_POWER_SETTING value. PWR_MIN must be ≤ PWR_MAX. Cavium recommends a PWR_MIN value larger than zero to set a minimum performance level in case PWR_SETTING is set to an unreachable goal. See the cores CP0 register PowThrottle on page 157. PWR_MIN = 50% of PowThrottle[MAXPOW] could be a good choice, for example.
<15:0>	PWR_SETTING	R/W	0xFFFF	—	A power limiter for the chip. A limiter of the power consumption of the chip. This power limiting is implemented by a closed-loop feedback control system for the AVG_CHIP_POWER power approximation. The direct output of the PWR_SETTING feedback control system is the CURRENT_POWER_SETTING value. The power consumed by the chip (estimated currently by the AVG_CHIP_POWER value) is an indirect output of the PWR_SETTING feedback control system. PWR_SETTING is not used by the hardware when PWR_MIN equals PWR_MAX. PWR_MIN and PWR_MAX threshold requirements always supersede PWR_SETTING limits. (For maximum PWR_SETTING feedback control freedom, set PWR_MIN=0 and PWR_MAX=0xFF.) PWR_SETTING equal to 0 forces the chip to consume near minimum power. Increasing PWR_SETTING value from 0 to 0xFFFF increases the power that the chip is allowed to consume linearly (roughly) from minimum to maximum.

IOB Chip Power Numbers Register IOB_CHIP_PWR_OUT

This register contains power numbers from the various partitions on the chip. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:48>	CPU_PWR	RO/H	0x0	—	An estimate of the current CPU core complex power consumption. The CPU core complex includes the caches and DRAM controller(s), as well as all CPU cores. Linearly larger values indicate linearly higher power consumption. This power consumption estimate is energy per core clock.
<47:32>	CHIP_POWER	RO/H	0x0	—	An estimate of the current total power consumption by the chip. Linearly larger values indicate linearly higher power consumption. CHIP_POWER is the sum of CPU_POWER and COPROC_POWER.
<31:16>	COPROC_POWER	RO/H	0x0	—	An estimate of the current coprocessor power consumption. Linearly larger values indicate linearly higher power consumption. This estimate is energy per core clock, and will generally decrease as the ratio of core to coprocessor clock speed increases.
<15:0>	AVG_CHIP_POWER	RO/H	0x0	—	An average of CHIP_POWER, calculated using an IIR filter with an average weight of 16K core clocks.

Cavium Confidential for Bowen Yang 8/27/2014

IOB Chip Current Power Register
IOB_CHIP_CUR_PWR

This register contains the current power setting. See Table 5–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	CURRENT_POWER_SETTING	RO/H	0x0	—	Global throttling value currently being used. Throttling can force units (CPU cores, in particular) idle for a portion of time, which will reduce power consumption. When CURRENT_POWER_SETTING is equal to zero, the unit is idle most of the time and consumes minimum power. When CURRENT_POWER_SETTING is equal to 0xFF, units are never idled to reduce power. The hardware generally uses a CURRENT_POWER_SETTING value that is as large as possible (in order to maximize performance) subject to the following constraints (in priority order): • PWR_MIN ≤ CURRENT_POWER_SETTING ≤ PWR_MAX • Power limits from the PWR_SETTING feedback control system In the case of the CPU cores, CURRENT_POWER_SETTING effectively limits the CP0 PowThrottle[POWLIM] value: effective POWLIM = MINIMUM(CURRENT_POWER_SETTING,PowThrottle[POWLIM])

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 6

On-Chip Logic Analyzer Unit (OCLA)

This chapter contains the following subjects:

- Overview
- Features
- Architecture
- Enumerations
- OCLA Registers
- DTX Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

This chapter discusses the on-chip logic analyzer unit (OCLA). The OCLA provides capabilities to trace internal transactions for software debugging, performance tuning, and hardware diagnostics.

6.1 Features

CN71XX's OCLA complex has the following major features:

- In-block storage for 4096 36-bit captures.
- Optional overflow to DDR/L2 cache.
- Optional idle and duplicate data compression.
- One 72-bit or two 36-bit capture data streams.
- Four data matcher and counter blocks.
- Two programmable FSMs.
- Three multicore debug (MCD) triggers.

6.2 Architecture

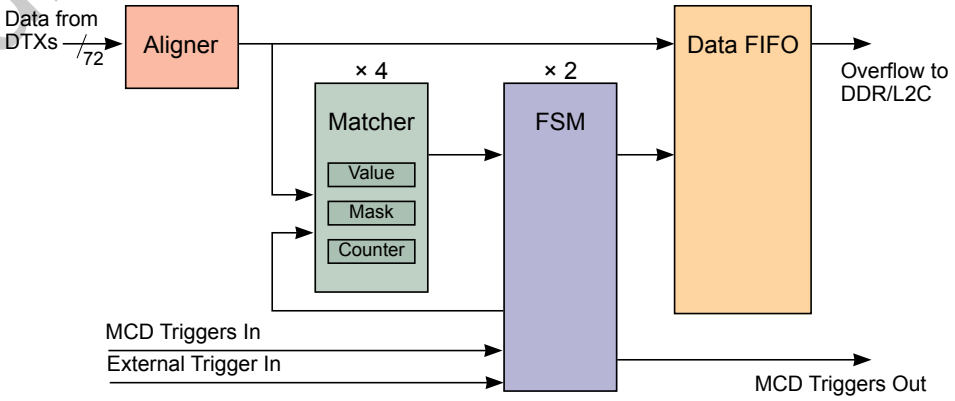
There is one OCLA complex in CN71XX, listed in Table 6–1.

Table 6–1 OCLA Complexes

OCLA Number	Location	Clock Domain	Maximum Size (OCLA(0)_CONST[DAT_SIZE])
OCLA(0)	L2 cache interface	core-clock	4096

Referencing Figure 6–1, each OCLA receives 72 bits of data from coprocessors or capture points via Debug Bus Transmitters (DTXs). This raw data is visible in OCLA(0)_RAW(0..1). Each bit of data is aligned by a programmable shifter, then feeds four 72-bit wide programmable ternary matchers (MAT). Each matcher also includes a 32-bit counter. Each matcher provides data to two parallel FSMs, which combine the match information to determine when to trigger and capture data. When the FSM requests a capture, the data is stored in the Data FIFO, and optionally overflows to DDR/L2 cache.

Figure 6–1 OCLA Complex Architecture



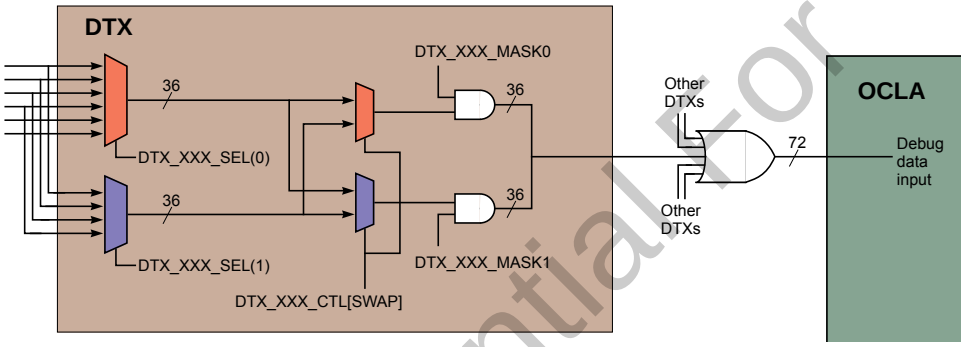
Cavium Confidential For Boyan Yang ALU 02/27/2014

6.2.1 Debug Bus Transmitters (DTX)

Feeding each OCLA is a series of Debug Bus Transmitters (DTXs) distributed in the coprocessors and other critical points outside of OCLA. The internals of one DTX and some of the logic feeding it is shown in Figure 6–2. Each DTX selects the bit-mask of 72 bits it will drive to OCLA; only one DTX may drive any given bit or the results will be ORed together. Each DTX also selects what data is interesting to drive onto the bus (debug mux).

NOTE: Any read or write to a DTX register must be followed by a read (which completes) to SLI_BIST_STATUS.

Figure 6–2 DTX Architecture



6.2.1.1 DTX Broadcast

As many DTXs share the debug bus, it is often convenient for software to disable all debug bus drivers. DTX therefore supports broadcast to allow writing every DTX with a single operation. Before doing the first broadcast, software must set DTX_LMC(0)_BCST_RSP[ENA]. Once done, a write to DTX_BROADCAST_CTL has the same effect as writing DTX_*xxx*_CTL for every *xxx*. Likewise for DTX_BROADCAST_SEL(0..1) or DTX_BROADCAST_ENA(0..1).

Cavium Confidential For Bowen Ying ALC 02/27/2014

6.2.1.2 DTX Setup

Typically before setting up an OCLA scenario, software would:

1. Initialize broadcast by setting DTX_LMC(0)_BCST_RSP[ENA], as described above.
2. Write DTX_BROADCAST_CTL, DTX_BROADCAST_ENA(0..1), DTX_BROADCAST_SEL(0..1) with zero to clear out any previous configurations.
3. Configure DTX_*xxx*_CTL and DTX_*xxx*_ENA(0), and set DTX_*xxx*_ENA(0)[ENA]=all-ones for the *xxx* of interest to be routed to the low half of the 72-bit data bus.
4. Configure DTX_*yyy*_CTL and DTX_*yyy*_ENA(1), and set DTX_*yyy*_ENA(1)[ENA]=all-ones for the *yyy* of interest to be routed to the high half of the 72-bit data bus.

6.2.2 Aligner

Each bit of debug data first enters a programmable aligner. This delays each bit by 0–7 cycles as specified in OCLA(0)_STAGE(0..71). This corrects for the variable amount of propagation delay between signals from different blocks, and also allows aligning request signals with data that may follow in subsequent cycles.

6.2.3 Matchers

The 72 bits of aligned data enters 4 parallel matchers.

Each matcher contains 72-bits to compare with the aligned data in OCLA(0)_MAT(0..3)_VALUE(0..1), using the bit-mask OCLA(0)_MAT(0..3)_MASK(0..1). The FSMs may overwrite the value to compare against by rotating the captured data by OCLA(0)_MAT(0..3)_CTL[SHIFT]. This allows a transaction identifier to be captured from one transaction and used to match against a future transaction.

Each matcher contains a 32-bit counter, OCLA(0)_MAT(0..3)_COUNT. If OCLA(0)_MAT(0..3)_CTL[INC_MATCH] is set, this counter will increment on each match. Otherwise it will only increment based on FSM requests. This counter may be used directly as a performance counter, or may be used to collect samples of capture data (sample every Nth match, or match after every Nth cycle.)

When the counter reaches OCLA(0)_MAT(0..3)_THRESH an interrupt is generated in OCLA(0)_STATE_INT[OVFL]. The FSMs may reset the counter, as may software with OCLA(0)_STATE_INT.

If OCLA(0)_MAT(0..3)_CTL[FSM_CTR] is clear, the match result is sent to the FSMs. If set, the FSM receives if the counter is greater than or equal to the threshold.

Cavium Confidential For Beiwen Yang 02-27-2014 Only

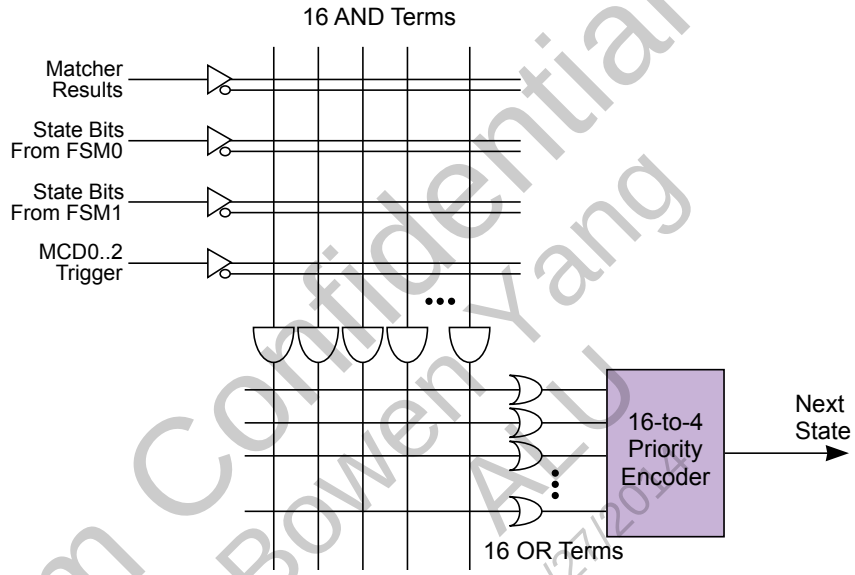
6.2.4 FSMs

Each matcher feeds two FSMs. The FSMs may be used to capture two separate streams of data (on the upper 36-bits and lower 36-bits of data), or used together for more complicated expressions and to double the state space.

Each FSM is implemented as a programmable logic array (PLA) style organization, with 15 inputs and 4 outputs. The PLA is shown in Figure 6–3. The PLA 15 inputs consist of the 4 outputs from the 4 matchers, 4 bits from the previous state of FSM0, 4 bits of previous state from FSM1. In addition there are three Multi-core Debug wires that may be generated or received from the cores or other OCLAs; see Section 2.22.1.

Each input or its inverse can be ANDed into one of 16 AND terms, based on OCLA(0)_FSM(0..1)_AND(0..15)_I(0..1). The 16 AND terms are then reduced to 16 OR terms based on OCLA(0)_FSM(0..1)_OR(0..15). Each OR term corresponds to a one-hot state number; i.e. OR0 means switch-to-state-0. The 16 OR terms are priority encoded to form the 4-bit state number. If no ORs are asserted the state number remains the same as in the previous cycle.

Figure 6–3 FSM PLA



Cavium Confidential For Bowen Yang ALU 02/27/2013

6.2.5 State Outputs

Each FSM can be in one of 16 states as generated by the FSM described in Section 6.2.4. Each FSM's state number then determines the output signals from the FSM using OCLA(0)_FSM(0..1)_STATE(0..15):

- If OCLA(0)_FSM(0..1)_STATE(0..15)[SET_INT], set OCLA(0)_STATE_INT[FSM(0..1)_INT] to signal an interrupt to software.
- If [CAP], possibly enable capturing as described in Section 6.2.6.
- If [SET_MCD], set OCLA(0)_STATE_SET[MCD] and drive the Multichip Debug (MCD) wire to the cores and other OCLA units. MCD wires are broadcast across the chipor driven to GPIO pins for logic analyzer triggering (see GPIO_OUTSEL_E).
- If [SET_TRIG], set OCLA(0)_STATE_SET[TRIG] and if the bit transitions from zero to one, start the capture counter as described in Section 6.2.6.
- If [SET_VAL<0..3>], for the matchers corresponding to the set bits, rotate the 72-bits of potential capture data by OCLA(0)_MAT(0..3)_CTL[SHIFT] and load it into OCLA(0)_MAT(0..3)_VALUE(0..1). This operation takes 4 cycles to compete, therefore at least 4 cycles must be between the reception of the post-aligned value being loaded and the post-aligned value to compare against.
- If [CLR_CNT<0..3>], for the matchers corresponding to the set bits, clear OCLA(0)_MAT(0..3)_COUNT.
- If [INC_CNT<0..3>], for the matchers corresponding to the set bits, increment OCLA(0)_MAT(0..3)_COUNT.

6.2.6 Capturing FIFOs

The 72-bits of potential capture data is split into two 36-bit halves. Capture of each half can be controlled by either a single FSM or by a combination of both FSM outputs, as configured by OCLA(0)_CDH(0..1)_CTL[CAP_CTL].

Captured data is in the form of 38-bit-wide capture entries.

OCLA Capture Data Structure OCLA_CAP_DAT_S

Data packets are indicated by an entry's bit 37 being clear, and described by the OCLA_CAP_DAT_S structure:

Bit Pos	Field Name	Field Description
<37>	CTL	Indicates a control word. Always clear for data structures.
<36>	HI	Set to indicate a sample of high data, clear for a sample of low data.
<35:0>	DATA	Captured trace data.

OCLA Capture Control Structure OCLA_CAP_CTL_S

Control packets are indicated by an entry's bit 37 being set, and described by the OCLA_CAP_CTL_S structure:

Bit Pos	Field Name	Field Description
<37>	CTL	Indicates a control word. Always set for control structures.

Cavium Confidential For Bowen Wang ALU 02/27/2014

Bit Pos	Field Name	Field Description
<36>	—	Reserved
<35>	EOT1	End of duplicated capture for high data. Symmetric with EOT0 description; see [EOT0].
<34>	EOT0	End of duplicated capture for low data. When set, CYCLE indicates the cycle at which the previous entry of low data stopped being replicated. This may be set along with SOT0 to indicate a repeat followed by new sequence.
<33>	SOT1	Start transition from no-capture to capture or duplicated data stopped while capturing for high data.
<32>	SOT0	Start transition from no-capture to capture or duplicated data stopped while capturing for low data. When set, CYCLE indicates the cycle number of the next new low data, minus one cycle.
<31:0>	CYCLE	Cycle at which this control entry was written, from OCLA(0)_TIME register.

Each cycle, the OCLA can capture:

- Nothing.
- A low or high data entry.
- A low and high data entry.
- A control entry. If needed, the control entry may be shared between both low and high; there are never two control entries inserted in the same cycle.
- A control entry and low or high data entry.

Control packets are used to optionally indicate that the data sequence is discontinuous. If OCLA(0)_CDH(0..1)_CTL[DUP] is clear, duplicate data samples are compressed with control packets. If OCLA(0)_CDH(0..1)_CTL[DIS_STAMP] is clear, non-captured times are indicated with control packets. When time rolls over, OCLA will always insert a control entry with CYCLE = 0 so wrapping may be detected unless there is a data entry captured that cycle (and thus software can intuit the rollover by counting data entries from the last control entry inserted).

A SOT control packet is inserted when OCLA(0)_CDH(0..1)_CTL[DIS_STAMP] is clear and the data stream transitions from no-capture to valid data or the data stream transitions from duplicated data (when so enabled) to different data.

A EOT control packet is inserted when OCLA(0)_CDH(0..1)_CTL[DIS_STAMP] is clear and the data stream transitions from duplicated data (when so enabled) to either different data or no capture.

The presence of an EOT control packet indicates duplicated data. When an EOT control packet is present:

```
Repeat count = (EOT's CYCLE count) - (SOT's CYCLE count)
                  - (number of distinct data values)
```

where (SOT's CYCLE count) is the cycle count in the last control packet (which is always an SOT control packet) and (number of distinct data values) is the number of distinct data values between two control packets.

Table 6–2 illustrates this process with an example.

Table 6–2 Example Capture Sequence

Time	Precaptured Data	Entries Created
0	No-capture	—
1	No-capture	CTL: CYCLE=1, SOT
2	0x10	DATA: 0x10
3	0x20	DATA: 0x20

Cavium Confidential For Bowen Yang ALI 02/21/2014

Table 6–2 Example Capture Sequence (Continued)

Time	Precaptured Data	Entries Created
4	No-capture	—
5	No-capture	—
6	No-capture	CTL: CYCLE=6, SOT
7	0x30	DATA: 0x30
8	0x30	—
9	0x30	—
10	0x30	CTL: CYCLE=10, EOT, SOT
11	0x40	DATA: 0x40
12	0x40	—
13	0x40	CTL: CYCLE=13, EOT
14	No-capture	—

Captured data is retrieved by reading OCLA(0)_DAT_POP. Data may be read while capturing is in progress, however the FIFO may wrap around while being read, unless it is configured to stop on FIFO full.

6.2.7 Capture to DDR / L2 Cache

To reduce loss of captured data, captures may be configured to overflow the OCLA FIFO into memory. When OCLA(0)_FIFO_DEPTH is greater than OCLA(0)_FIFO_LIMIT[DDR] entries, FIFO entries will be removed, packed into a cache line, and overflowed to DDR/L2.

Each cache line contains 26 entries bit-packed into a 128-byte line of memory, packed as follows: .

Bit Index	Field
<31:0>	Timestamp. Value of OCLA(0)_TIME at which the entry was written. Note this will be later than the time point at which the entry was captured.
<35:32>	Reserved.
<73:36>	Entry 0.
<111:74>	Entry 1.
<149:112>	Entry 2.
. . .	. . .
<1023:986>	Entry 25.

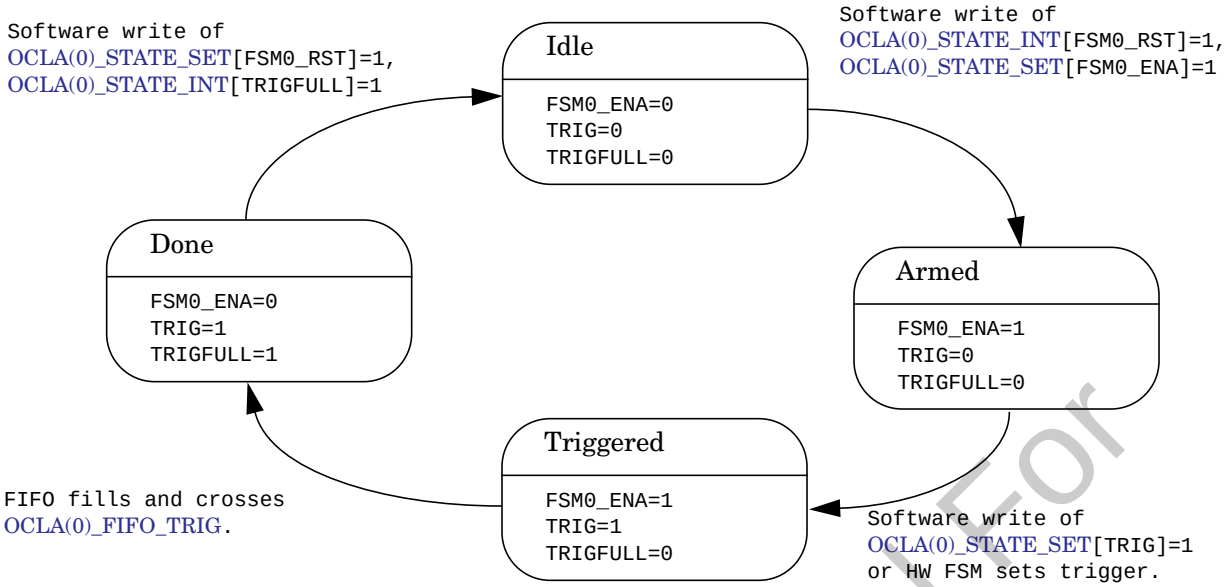
If the captures are extremely frequent, and the memory subsystem very busy, the capture buffer may fill (indicated by OCLA(0)_FIFO_DEPTH == OCLA(0)_CONST[DAT_SIZE]). OCLA will optionally stop capture if OCLA(0)_FIFO_LIMIT[OVERFULL] is set to just less than OCLA(0)_FIFO_DEPTH.

6.2.8 Capture Transitions

OCLA may be in several overall states as indicated by OCLA(0)_STATE_INT and OCLA(0)_STATE_SET. Figure 6–4 shows how these fields relate in typical usage.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 6–4 Overall State Transition Example



1. Names shown correspond to fields in OCLA(0)_STATE_INT and OCLA(0)_STATE_SET.
2. For clarity, only FSM0 is shown; similar fields and transitions exist for FSM1.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

6.3 Enumerations

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

6.4 OCLA Registers

The OCLA registers are listed in Table 6–3.

Table 6–3 OCLA Registers

Register	Base Address[1]	CSR Type	Detailed Description
OCLA(0)_CONST	0x00011800A8000000 | α<<24	RSL	See page 376
OCLA(0)_SFT_RST	0x00011800A8000020 | α<<24	RSL	See page 376
OCLA(0)_BIST_RESULT	0x00011800A8000040 | α<<24	RSL	See page 376
OCLA(0)_GEN_CTL	0x00011800A8000060 | α<<24	RSL	See page 376
OCLA(0)_STATE_INT	0x00011800A8000080 | α<<24	RSL	See page 377
OCLA(0)_STATE_SET	0x00011800A80000A0 | α<<24	RSL	See page 376
OCLA(0)_TIME	0x00011800A80000C0 | α<<24	RSL	See page 378
OCLA(0)_RAW(0..1)	0x00011800A8000100 | α<<24 | β<<3	RSL	See page 378
OCLA(0)_FIFO_DEPTH	0x00011800A8000200 | α<<24	RSL	See page 379
OCLA(0)_FIFO_LIMIT	0x00011800A8000240 | α<<24	RSL	See page 379
OCLA(0)_FIFO_TAIL	0x00011800A8000260 | α<<24	RSL	See page 379
OCLA(0)_FIFO_WRAP	0x00011800A8000280 | α<<24	RSL	See page 379
OCLA(0)_FIFO_TRIG	0x00011800A80002A0 | α<<24	RSL	See page 380
OCLA(0)_STACK_BASE	0x00011800A8000400 | α<<24	RSL	See page 380
OCLA(0)_STACK_TOP	0x00011800A8000420 | α<<24	RSL	See page 380
OCLA(0)_STACK_WRAP	0x00011800A8000440 | α<<24	RSL	See page 381
OCLA(0)_STACK_STORE_CNT	0x00011800A8000460 | α<<24	RSL	See page 381
OCLA(0)_STACK_CUR	0x00011800A8000480 | α<<24	RSL	See page 381
OCLA(0)_CDH(0..1)_CTL	0x00011800A8000600 | α<<24 | β<<3	RSL	See page 384
OCLA(0)_DAT_POP	0x00011800A8000800 | α<<24	RSL	See page 385
OCLA(0)_STAGE(0..71)	0x00011800A8100000 | α<<24 | β<<3	RSL	See page 380
OCLA(0)_MAT(0..3)_CTL	0x00011800A8200000 | α<<24 | β<<12	RSL	See page 382
OCLA(0)_MAT(0..3)_VALUE(0..1)	0x00011800A8210000 | α<<24 | β<<12 | χ<<3	RSL	See page 382
OCLA(0)_MAT(0..3)_MASK(0..1)	0x00011800A8220000 | α<<24 | β<<12 | χ<<3	RSL	See page 382
OCLA(0)_MAT(0..3)_COUNT	0x00011800A8230000 | α<<24 | β<<12	RSL	See page 383
OCLA(0)_MAT(0..3)_THRESH	0x00011800A8240000 | α<<24 | β<<12	RSL	See page 383
OCLA(0)_FSM(0..1)_AND(0..15)_I(0..1)	0x00011800A8300000 | α<<24 | β<<12 | χ<<4 | δ<<3	RSL	See page 383
OCLA(0)_FSM(0..1)_OR(0..15)	0x00011800A8310000 | α<<24 | β<<12 | χ<<3	RSL	See page 383
OCLA(0)_FSM(0..1)_STATE(0..15)	0x00011800A8320000 | α<<24 | β<<12 | χ<<3	RSL	See page 384
OCLA(0)_DAT(0..8191)	0x00011800A8400000 | α<<24 | β<<3	RSL	See page 385

1. For registers with multiple versions, α represents the values within the specified range, β the second range, χ the third, and δ the fourth.

Cavium Confidential For Bowen Yang 6/2/2014

6.4.1 General Registers

OCLA Constants Registers
OCLA(0)_CONST

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved.
<15:0>	DAT_SIZE	RO	—	—	Size of data RAM in units of 36-bit entries. This value is subject to change between chip passes, and software should thus use this value rather than a hard coded constant.

OCLA Reset Registers
OCLA(0)_SFT_RST

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	RESET	R/W	0	0	Reset. When set, causes a block reset, except RSL.

OCLA BIST Result Registers
OCLA(0)_BIST_RESULT

This register provides access to the internal BIST results. Each bit is the BIST result of an individual memory (per bit, 0 = pass and 1 = fail). See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	DAT	RO	X	0x0	BIST result of the DAT memory.

OCLA General Control Registers
OCLA(0)_GEN_CTL

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	EXTEN	R/W	0	0	Enable external triggering. 0 = External triggering ignored. 1 = When the external trigger pin selected with GPIO_OCLA_EXTEN_TRIG is high it will cause triggerring and set OCLA(0)_STATE_SET[TRIG]. The external device must de-assert the signal to release the trigger (it is not edge sensitive.)
<2>	DEN	R/W	0	1	Enable data bus and counter clocking. When set, the OCLA inbound data bus may be used and counters may increment. When clear, the bus is always zero and internal flops may be clock gated off to save power. Must be set for normal operation.
<1>	STT	R/W	0	—	Store to DDR directly, bypassing L2 cache.
<0>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang ALU 02/27/2014

OCLA State and Interrupt Registers
OCLA(0)_STATE_INT

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:60>	FSM1_STATE	RO/H	0x0	—	FSM1 current state.
<59:56>	FSM0_STATE	RO/H	0x0	—	FSM0 current state.
<55:36>	—	RAZ	—	—	Reserved
<35>	FSM1_RST	R/W1C	0	0	FSM1 hold in state zero. Writing one to OCLA(0)_STATE_SET[FSM1_RST] sets this bit and holds FSM1 in state zero, writing one to OCLA(0)_STATE_INT[FSM1_RST] removes the hold.
<34>	FSM0_RST	R/W1C	0	0	FSM0 hold in state zero. Writing one to OCLA(0)_STATE_SET[FSM0_RST] sets this bit and holds FSM0 in state zero, writing one to OCLA(0)_STATE_INT[FSM0_RST] removes the hold.
<33>	FSM1_ENA	R/W1C/H	0	0	FSM1 sequencing enabled.
<32>	FSM0_ENA	R/W1C/H	0	0	FSM0 sequencing enabled.
<31:19>	—	RAZ	—	—	Reserved
<18>	DDRFULL	R/W1C/H	0	0	DDR buffer wrapped. Asserted when OCLA(0)_STACK_CUR has wrapped and been re-initialized to OCLA(0)_STACK_BASE.
<17>	WMARK	R/W1C/H	0	0	Internal buffer watermark reached. Asserted when OCLA(0)_FIFO_DEPTH > OCLA(0)_FIFO_LIMIT[WMARK].
<16>	OVERFULL	R/W1C/H	0	0	Capture ended due to FIFO overflow. Asserted when OCLA(0)_FIFO_DEPTH > OCLA(0)_FIFO_LIMIT[OVERFULL].
<15>	TRIGFULL	R/W1C/H	0	0	Capture ended due to buffer full. Asserted when OCLA(0)_FIFO_TRIG[LIMIT] ≥ OCLA(0)_FIFO_TRIG[COUNT].
<14>	CAPTURED	R/W1C/H	0	0	Capture started. Asserted when the first capture is made. Informational only; often masked.
<13>	FSM1_INT	R/W1C/H	0	0	FSM1 interrupt requested.
<12>	FSM0_INT	R/W1C/H	0	0	FSM0 interrupt requested.
<11:9>	MCD	R/W1C/H	0x0	0x0	Multichip Debug (MCD0..2) set. Asserted on MCD received from another coprocessor or code, or FSM MCD request or W1S to OCLA(0)_STATE_SET[MCD].
<8>	TRIG	R/W1C/H	0	0	Internal trigger set. Asserted on FSM internal trigger request or W1S to OCLA(0)_STATE_SET[TRIG]
<7:4>	—	RAZ	—	—	Reserved
<3:0>	OVFL	R/W1C/H	0x0	0x0	Match counter has overflowed. Asserted when OCLA(0)_MAT(0..3)_COUNT ≥ OCLA(0)_MAT(0..3)_THRESH. Informational only; often masked. Writing 1 clears the counter, not just the interrupt.

Cavium Confidential For Beta Version Only 02-21-2014

OCLA State Set Registers
OCLA(0)_STATE_SET

This register reads identically to OCLA(0)_STATE_INT, but allows R/W1S instead of R/W1C access. See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:60>	FSM1_STATE	RO/H	0x0	—	See OCLA(0)_STATE_INT[FSM1_STATE].
<59:56>	FSM0_STATE	RO/H	0x0	—	See OCLA(0)_STATE_INT[FSM0_STATE].
<55:36>	—	RAZ	—	—	Reserved
<35>	FSM1_RST	R/W1S	0	0	See OCLA(0)_STATE_INT[FSM1_RST].
<34>	FSM0_RST	R/W1S	0	0	See OCLA(0)_STATE_INT[FSM0_RST].
<33>	FSM1_ENA	R/W1S	0	1	See OCLA(0)_STATE_INT[FSM1_ENA].
<32>	FSM0_ENA	R/W1S	0	1	See OCLA(0)_STATE_INT[FSM0_ENA].
<31:19>	—	RAZ	—	—	Reserved
<18>	DDRFULL	R/W1S/H	0	0	See OCLA(0)_STATE_INT[DDRFULL].
<17>	WMARK	R/W1S/H	0	0	See OCLA(0)_STATE_INT[WMARK].
<16>	OVERFULL	R/W1S/H	0	0	See OCLA(0)_STATE_INT[OVERFULL].
<15>	TRIGFULL	R/W1S/H	0	0	See OCLA(0)_STATE_INT[TRIGFULL].
<14>	CAPTURED	R/W1S/H	0	0	See OCLA(0)_STATE_INT[CAPTURED].
<13>	FSM1_INT	R/W1S/H	0	0	See OCLA(0)_STATE_INT[FSM1_INT].
<12>	FSM0_INT	R/W1S/H	0	0	See OCLA(0)_STATE_INT[FSM0_INT].
<11:9>	MCD	R/W1S/H	0x0	0x0	See OCLA(0)_STATE_INT[MCD].
<8>	TRIG	R/W1S/H	0	0	See OCLA(0)_STATE_INT[TRIG].
<7:4>	—	RAZ	—	—	Reserved
<3:0>	OVFL	R/W1S/H	0x0	0x0	See OCLA(0)_STATE_INT[OVFL].

OCLA Current Time Registers
OCLA(0)_TIME

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CYCLE	R/W/H	0x0	—	Current time as free running counter. Loaded into captured control packets.

OCLA Raw Input Registers
OCLA(0)_RAW(0..1)

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:36>	—	RAZ	—	—	Reserved
<35:0>	RAW	RO/H	—	—	Raw value of debug bus input signals into OCLA.

Cavium Confidential For Bowen Yang ALU 02/27/2014

OCLA Capture FIFO Depth Registers
OCLA(0)_FIFO_DEPTH

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	DEPTH	RO/H	0x0	0x0	Current capture FIFO depth in 36-bit words.

OCLA Capture FIFO Limit Registers
OCLA(0)_FIFO_LIMIT

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	OVERFULL	R/W	all-ones	—	Stop level. When OCLA(0)_FIFO_DEPTH > OVERFULL, stop capturing and set OCLA(0)_STATE_INT[OVERFULL]. This should be set to no more than OCLA(0)_CONST[DAT_SIZE] minus 26 when using DDR capture to insure that overflow can be detected.
<47:32>	DDR	R/W	all-ones	—	DDR level. When OCLA(0)_FIFO_DEPTH > DDR, FIFO entries will be removed, packed into a cache line, and overflowed to DDR/L2. All-ones disables overflow to DDR/L2. If non-zero must be at least 28.
<31:16>	BP	R/W	all-ones	—	Backpressure level. When OCLA(0)_FIFO_DEPTH > BP, OCLA will signal backpressure to coprocessors. All-ones disables indicating backpressure.
<15:0>	WMARK	R/W	all-ones	—	Interrupt watermark level. When OCLA(0)_FIFO_DEPTH > WMARK OCLA will set OCLA(0)_STATE_INT[WMARK] interrupt. All-ones disables setting the interrupt.

OCLA Capture FIFO Tail Registers
OCLA(0)_FIFO_TAIL

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TAIL	RO/H	0x0	0x0	Address last written into entry FIFO.

OCLA Capture FIFO Wrap Counter Registers
OCLA(0)_FIFO_WRAP

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	WRAPS	R/W/H	0x0	0x0	Number of times FIFO has wrapped since trigger. Cleared when OCLA(0)_STATE_INT[TRIG] clear.

Cavium Confidential For Bowen Yang ALU 02/27/2014

OCLA Capture FIFO Trigger Level Registers
OCLA(0)_FIFO_TRIG

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:16>	LIMIT	R/W	0x0	0x1	Post-trigger number of entries to collect before stopping collection. If zero, collection will never stop, which may be desirable when overflowing to DDR/L2. Must be < OCLA(0)_CONST[DAT_SIZE] - 5.
<15:0>	CNT	R/W/H	0x0	0x0	Number of entries collected since trigger. Cleared when OCLA(0)_STATE_INT[TRIG] clear.

OCLA Input Staging Registers
OCLA(0)_STAGE(0..71)

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved
<2:0>	DLY	R/W	0x0	0x0	Cycles of delay staging to apply to corresponding input bit.

6.4.2 DDR/L2 Overflow Stack

OCLA Stack Base Registers
OCLA(0)_STACK_BASE

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:42>	—	RAZ	—	—	Reserved
<41:7>	PTR	R/W	0x0	—	Memory address for base of overflow stack.
<6:0>	—	RAZ	—	—	Reserved. For cache line alignment.

OCLA Stack Top Address Registers
OCLA(0)_STACK_TOP

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:42>	—	RAZ	—	—	Reserved
<41:7>	PTR	R/W	0x0	—	Memory address for top of overflow stack plus one.
<6:0>	—	RAZ	—	—	Reserved. For cache line alignment.

Cavium Confidential For Bowen Yang ALU 02/17/2014

OCLA Stack Current Registers
OCLA(0)_STACK_CUR

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:42>	—	RAZ	—	—	Reserved
<41:7>	PTR	R/W/H	0x0	—	Next address to write for overflow stack. During initialization this must be between OCLA(0)_STACK_BASE and OCLA(0)_STACK_TOP.
<6:0>	—	RAZ	—	—	Reserved. For cache line alignment.

OCLA Stack Wrap Counter Registers
OCLA(0)_STACK_WRAP

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	WRAPS	R/W/H	0x0	0x0	Number of times stack has been reset to OCLA(0)_STACK_BASE since trigger. Cleared when OCLA(0)_STATE_INT[TRIG] clear.

OCLA Stack Stores Performance Counter Registers
OCLA(0)_STACK_STORE_CNT

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	STORES	R/W/H	0x0	0x0	Number of cache line stores sent to memory subsystem. Not cleared by hardware.

Cavium Confidential For Bowen Yang ALU 02/27/2014

6.4.3 Per-Matcher Registers

OCLA Matcher Control Registers
OCLA(0)_MAT(0..3)_CTL

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7>	FSM_CTR	R/W	0	1	What output matcher provides to FSM: 0 = FSM receives raw match signal, asserting only in those cycles with matches. 1 = FSM receives OCLA(0)_MAT(0..3)_COUNT >= OCLA(0)_MAT(0..3)_THRESH.
<6>	INC_MATCH	R/W	0	1	Increment OCLA(0)_MAT(0..3)_COUNT counter automatically on each match.
<5:0>	SHIFT	R/W	0x0	—	Right rotation amount to apply to data loaded into OCLA(0)_MAT(0..3)_VALUE(0..1) register when FSM requests a value load.

OCLA Matcher Compare Value Registers
OCLA(0)_MAT(0..3)_VALUE(0..1)

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:36>	—	RAZ	—	—	Reserved
<35:0>	MASK	R/W/H	0x0	—	Data value to compare against when corresponding bits of OCLA(0)_MAT(0..3)_MASK(0..1) are set. Value may be updated with OCLA(0)_FSM(0..1)_STATE(0..15)[SET_VAL].

OCLA Matcher Compare Mask Registers
OCLA(0)_MAT(0..3)_MASK(0..1)

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:36>	—	RAZ	—	—	Reserved
<35:0>	MASK	R/W	0x0	—	Bitmask of which bits in OCLA(0)_MAT(0..3)_VALUE(0..1) are to be compared. 0 = Do not compare bit. 1 = Compare bit.

Cavium Confidential For Bowen Yang ALU 02/27/2014

OCLA Matcher Count Registers
OCLA(0)_MAT(0..3)_COUNT

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	COUNT	R/W/H	0x0	—	Current counter value. Note software must reset this to zero (or the appropriate count) before starting capture.

OCLA Matcher Count Threshold Registers
OCLA(0)_MAT(0..3)_THRESH

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	THRESH	R/W	0x0	—	Counter threshold value. Compared against OCLA(0)_MAT(0..3)_COUNT to assert matcher output, and set OCLA(0)_STATE_INT[OVFL].

6.4.4 Per-FSM Registers

OCLA FSM PLA AND Tree Registers
OCLA(0)_FSM(0..1)_AND(0..15)_I(0..1)

Values for PLA-AND plane. AND(0..15) represents the 15 allowed AND terms. I(0..1) for I=0 indicates the term non-inverted, for I=1 indicates the term inverted. Any AND tree may be disabled by setting the same bit in both _I(0) and _I(1), as "((1) & !(1))" is always false.

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:15>	—	RAZ	—	—	Reserved
<14:12>	MCD	R/W	0x0	—	Multichip Debug (MCD) 0..2 inputs.
<11:8>	MATCH	R/W	0x0	—	Matcher 0..3 input.
<7:4>	FSM1_STATE	R/W	0x0	—	FSM 1 last state.
<3:0>	FSM0_STATE	R/W	0x0	—	FSM 0 last state.

OCLA FSM PLA AND Tree Registers
OCLA(0)_FSM(0..1)_OR(0..15)

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	OR_STATE	R/W	0x0	—	Column to drive on PLA OR-plane.

Cavium Confidential For Bowen Kang ALU 02/27/2014

OCLA FSM State Registers
OCLA(0)_FSM(0..1)_STATE(0..15)

See Section 6.2.5 for more details on each of these actions. See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:30>	—	RAZ	—	—	Reserved
<29>	SET_INT	R/W	0x0	0x0	In this state set interrupt.
<28>	CAP	R/W	0	1	In this state request capture this cycle.
<27:25>	SET_MCD	R/W	0x0	0x0	In this state set MCD.
<24>	SET_TRIG	R/W	0	0	In this state set internal trigger indication.
<23:20>	—	RAZ	—	—	Reserved
<19:16>	SET_VAL	R/W	0x0	0x0	In this state store match value into matcher 0..3.
<15:12>	—	RAZ	—	—	Reserved
<11:8>	CLR_CNT	R/W	0x0	0x0	In this state clear match counter.
<7:4>	—	RAZ	—	—	Reserved
<3:0>	INC_CNT	R/W	0x0	0x0	In this state increment match counter. See Section 6.2.5 for more information on this or any of the bits above.

6.4.5 Per-Capture Half Registers

OCLA Capture Data Half Control Registers
OCLA(0)_CDH(0..1)_CTL

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5>	DUP	R/W	0	0	Retain duplicates in the data stream.
<4>	DIS_STAMP	R/W	0	0	Remove time stamps from data stream.
<3:0>	CAP_CTL	R/W	0x2	0x2	Minterms that will cause data to be captured. These minterms are the four inputs to a 4-1 mux selected by PLA1 and 0. The output is thus calculated from the equation: fsmcap0 = OCLA(0)_FSM(0)_STATE[state0][CAP]. fsmcap1 = OCLA(0)_FSM(1)_STATE[state1][CAP]. out = ((<3> & fsmcap0 & fsmcap0) || (<2> & fsmcap1 & !fsmcap0) || (<1> & !fsmcap1 & fsmcap0) || (<0> & !fsmcap1 & !fsmcap0)). Common examples: 0x0 = No capture. 0x2 = Capture when fsmcap0 requests capture. 0x4 = Capture when fsmcap1 requests capture. 0x6 = Capture on fsmcap0 | fsmcap1. 0x8 = Capture on fsmcap0 & fsmcap1. 0xF = Always capture.

Cavium Confidential For Bowen Yang ALU 6247954

6.4.6 Data Registers

OCLA Data Registers
OCLA(0)_DAT(0..8191)

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:38>	—	RAZ	—	—	Reserved
<37:0>	ENTRY	RO/H	0x0	—	Captured entry. Data is in the format described by OCLA_CAP_DAT_S or OCLA_CAP_CTL_S.

OCLA Data Pop Registers
OCLA(0)_DAT_POP

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	VALID	RC/H	0	1	Valid entry. Indicates the FIFO contains data, and equivalent to OCLA(0)_FIFO_DEPTH[DEPTH] != 0.
<62>	TRIG	RO/H	0	0	Internal trigger set. Equivalent to OCLA(0)_STATE_INT[TRIG].
<61>	WMARK	RO/H	0	0	Internal buffer watermark reached. Equivalent to OCLA(0)_STATE_INT[WMARK].
<60:38>	—	RAZ	—	—	Reserved
<37:0>	ENTRY	RC/H	0x0	—	Captured entry. If VALID is set, has read side effect of unloading data by decrementing OCLA(0)_FIFO_DEPTH[DEPTH]. Data is in the format described by OCLA_CAP_DAT_S or OCLA_CAP_CTL_S. Note that unloading data will cause that data not to be sent to memory, therefore OCLA(0)_DAT_POP should not be read when OCLA(0)_FIFO_LIMIT[DDR] != all-ones.

Cavium Confidential For Bowen Yang ALU 02/27/2014

6.5 DTX Registers

DTX registers exist for many coprocessors on the chip; see Section 6.2.1. The generic form of the DTX block's registers are listed in Table 6–3.

NOTE: Any read or write operation to a DTX register must be followed by a read operation (which completes) to SLI_BIST_STATUS.

Table 6–4 DTX Registers

Register[1]	Base Address[2 3]	CSR Type	Detailed Description
DTX_XXX_SEL(0..1)	0x00011800FE*xxx*000 | α<<3	RSL	See page 386
DTX_XXX_ENA(0..1)	0x00011800FE*xxx*020 | α<<3	RSL	See page 386
DTX_XXX_DAT(0..1)	0x00011800FE*xxx*040 | α<<3	RSL	See page 386
DTX_XXX_CTL	0x00011800FE*xxx*060	RSL	See page 387
DTX_XXX_BCST_RSP	0x00011800FE*xxx*080	RSL	See page 387

1. This table indicates the general pattern of DTX register names. *XXX* is one of AGL, BROADCAST, DFA, DPI, FPA, GMX, IOB, IPD, L2C_CBC, L2C_MCI, L2C_TAD, LMC, MIO, PCS, PEM, PIP, PKO, POW, RST, SATA, SLI, TIM, USBDRD; see the SDK for an exhaustive list.
2. This table indicates the general pattern of DTX addresses. *XXX* indicates the specific DTX's address; see the SDK for an exhaustive list.
3. For registers with multiple versions, α represents the values within the specified range, β the second range, χ the third, and δ the fourth.

DTX Select Register DTX_XXX_SEL(0..1)

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:24>	—	RAZ	0x0	0x0	Reserved.
<23:0>	VALUE	R/W	0x0	0x0	Debug select. Selects which signals to drive onto low/high 36-bit debug buses.

DTX Enable Register DTX_XXX_ENA(0..1)

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:36>	—	RAZ	0x0	0x0	Reserved.
<35:0>	ENA	R/W	0x0	0x0	Output enable vector of which bits to drive onto the low/high 36-bit debug buses. Normally only one block will drive each bit.

DTX Raw Data Register DTX_XXX_DAT(0..1)

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:36>	—	RAZ	0x0	0x0	Reserved.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<35:0>	RAW	R/W	0x0	0x0	Raw debug data captured by the DTX before the ENA is applied. This gives the ability to peek into blocks during an OCLA capture without OCLA reconfiguration.

DTX Control Register
DTX_XXX_CTL

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	0x0	0x0	Reserved.
<4>	ACTIVE	R/W	0	0	Force the block's gated clocks on, so that the state of idle signals may be captured.
<3:2>	—	RAZ	0x0	0x0	Reserved.
<1>	ECHOEN	R/W	0	0	Drive debug bus with the value in DTX_XXX_ENA(0..1) instead of the normal block debug data. For diagnostic use only.
<0>	SWAP	R/W	0	0	Swap the high and low 36-bit debug bus outputs.

DTX Broadcast Response Register
DTX_XXX_BCST_RSP

See Table 6–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	0x0	0x0	Reserved.
<0>	ENA	R/W	0	0	Enable this DTX instance as the responder to DTX Broadcast reads/ writes.

Cavium Confidential For Bowen Yang ALU 02/27/2014

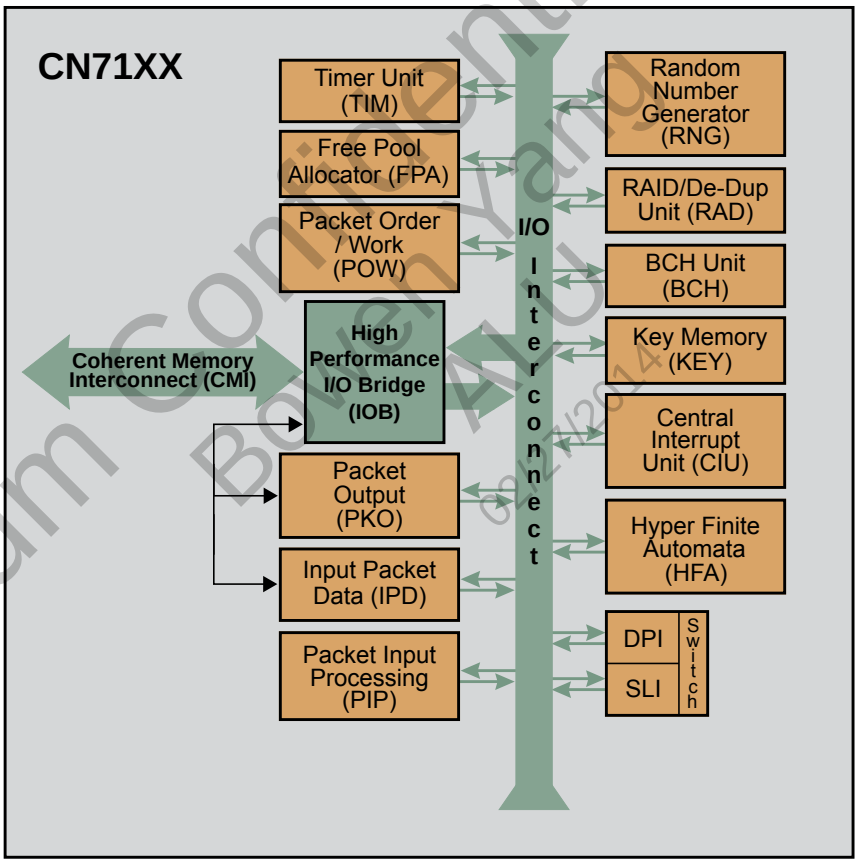
Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Section 2: OCTEON III Coprocessors

This section contains chapters that describe the various coprocessors that make up the CN71XX (refer to Figure S2-1). They are the following:

- Packet Input Processing/Input Packet Data (PIP/IPD)
- Packet Output Data (PKO)
- Packet Order / Work (POW)
- Free Pool Allocator (FPA)
- Hyper Finite Automata unit (HFA)
- DPI/SLI unit
- Timer unit (TIM)
- RAID unit (RAD)
- BCH unit (BCH)
- Key Memory unit (KEY)
- Random-Number Generator (RNG), Random-Number Memory (RNM)

Figure S2-1CN71XX Coprocessors



Cavium Confidential For Baiwen Yang ALU 09/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 7

Packet Order / Work Unit (POW)

This chapter contains the following subjects:

- Overview
- POW Work Flow, Operations, and Ordering
- Example Software Architecture Using POW Ordering and Synchronization
- POW Internal Architecture
- Work-Queue Entry Format
- Core and Fetch-and-Add Pending Switch Bits
- POW Interrupts
- POW QOS Features
- POW Debug Visibility
- POW Performance Considerations
- Forward Progress Constraints
- POW Operations
- POW ECC Codes
- POW Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The Packet Order/Work (POW) unit is a coprocessor that provides the following important functions:

- work queueing
- work scheduling/descheduling
- ordering and synchronization of work

Work Queueing

Work is described by an associated work-queue entry and may be created by either hardware units or core software. The CN71XX centralized packet-input hardware creates a work-queue entry and submits work for each packet arrival. Core software can create work-queue entries and submit work as desired. The packet output (PKO), PCIe, Timer (TIM), and Hyper Finite Automata (HFA) hardware units can also submit work-queue entries created by core software after completing an operation/ instruction.

POW implements eight input work queues. The different work queues can be used to provide different service levels. The POW hardware implements static and weighted-round-robin priorities programmably on a core-by-core basis.

The input work queues can be infinitely large, overflowing to DRAM when necessary.

Work Scheduling / Descheduling

Core software requests work from POW. POW selects (i.e. schedules) work for the core and returns a pointer to the work-queue entry that describes the work to core software. This off-loads much overhead from the cores and the coherent memory bus.

All work is not equal since the POW hardware supports (16) groups. Each piece of work has an associated group identifier. A configuration variable for each core specifies the groups that the associated core will accept when it requests work. This configuration variable is a 16-bit bitmask, one bit per group, so core software can specify all possible combinations of groups. The POW hardware does not schedule a piece of work if the core doesn't accept the group associated with the work.

Groups provide a means to execute different functions on different cores, though all cores share the same POW hardware. For example, packet processing may be pipelined from one group of cores to another group of cores, with the first group performing the first stage of work and the next group performing the next stage of work.

A core can deschedule a piece of work. Deschedule means that the software running on this core will not complete the work at this time, and the POW hardware should reschedule it later. The POW hardware reschedules previously descheduled work at higher priority than it schedules new work from an input queue. Deschedule can be useful in a number of circumstances:

- It can transfer work from one core group to another. This is one mechanism to implement “work pipelining.”
- It can avoid consuming a core for work that requires a large synchronization delay.
- It can make work interruptible.

Cavium Confidential For Bowen Yang, A-U 07/27/2015

Ordering and Synchronization of Work

The POW hardware associates a 32-bit tag value and a tag type with each piece of work. The work-queue entry and the request to add work to an input work queue contain the initial tag value. (This initial tag value may be created by either the centralized input packet processing hardware or by core software.) Core software can also later switch the tag/type as the work progresses through different phases of the application. If the same tag value sequences are used by two packets, the packets are ordered.

There are three different tag types:

- **ORDERED** - Ordering is guaranteed with this tag type. (Atomicity is not.)
- **ATOMIC** - Ordering and atomicity are guaranteed with this tag type. Two pieces of work holding the same ATOMIC tag cannot be scheduled simultaneously.
- **NULL** - No ordering is guaranteed with this tag type, and work cannot be in-flight (with respect to the POW hardware) with this tag type.

The POW hardware, in combination with core software, uses these tag/type values to order and synchronize related work, and allow unrelated work to be unordered and unsynchronized. This is essential for efficient multicore execution. Two pieces of work may be/are related and will be ordered and synchronized when they share the same tag value and tag type. Two pieces of work may be unrelated and will execute entirely in parallel when they have different tag or tag type values.

For example, the tag value may be a hash of the standard TCP five-tuple (IP source address, IP destination address, IP protocol, TCP source port, TCP destination port) defining a “flow”. The same flow will have the same tag value, so it may be ordered and synchronized. Different flows will likely have different tag values, so will likely not be ordered and synchronized, and can be executed completely in parallel on different cores.

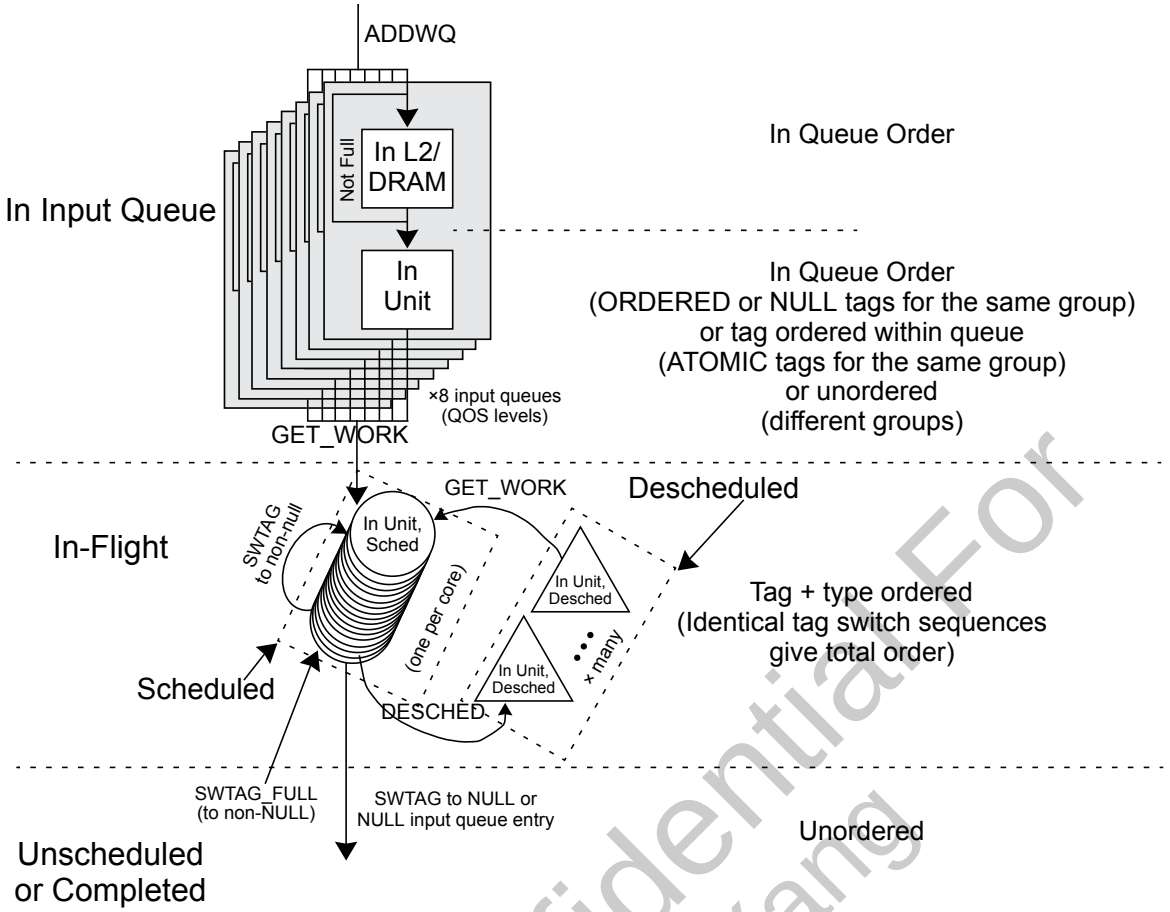
At different code phases, core software can change the tag value via a tag switch transaction with separated switch request and switch completion wait operations. The POW hardware completes a requested switch when the required ordering and atomicity constraints for the work are met. This separated switch transaction allows software to overlap the switch request latency with other profitable work and also allows software the option to deschedule the work while a tag switch is pending, thus avoiding long synchronization delays.

7.1 POW Work Flow, Operations, and Ordering

Figure 7–1 shows an abstracted view of the POW unit, focusing on the states of work as it flows through the POW Unit, and touching on the most important operations that cores can execute to transform work. Often, work flows through the states in the figure from top to bottom—work is first in the input queues, then in-flight, and finally unscheduled or completed.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 7–1 States of Work Flowing Through the POW



At any given time, only one piece of work can be scheduled to a particular core. This is shown in the center of Figure 7–1 (see “in unit, sched”). Clearly, the number of scheduled items is limited to the number of cores (up to four). A core is not scheduled if it is executing unscheduled work or if it completes scheduled work without requesting new work. Scheduled work is a subset of the POW in-flight work. Any core may deschedule a scheduled item at any point (see “in unit, desched” in Figure 7–1). Descheduled work remains in-flight, and will be rescheduled later, but is not currently executing on a core. Work that is scheduled remains scheduled after the completion of a tag switch transaction (SWTAG), unless the switch has a next tag state of NULL. A tag switch with a next tag switch to NULL causes work to immediately become unscheduled on the core. Figure 7–1 abstracts SWTAG transactions by showing them as a single arrow; in reality, a deschedule operation can occur after SWTAG transaction starts, but before it completes.

Work typically enters the POW unit through one of the input queues. The top of Figure 7–1 shows the eight POW input work queues. The POW unit internal entries are shared by in-flight work and work in input queues. Both software and hardware can add input-queue entries (ADDWQ). Though the POW unit size is limited, the POW hardware maintains the illusion of an infinite input work queue. When space is not available in the POW unit, the POW hardware adds the input-queue entries to an L2/DRAM list maintained by hardware. If space is available in the POW unit when work is added, the POW hardware buffers the work internally immediately and avoids the overhead of the memory list. If the POW hardware puts work in a memory

Cavium Confidential For Bowen Yang 04/22/14

list, it later automatically (and in the background) moves the work from L2/DRAM into the unit as soon as space becomes available in the unit, in the order that the work was originally added.

Work is typically scheduled to a core when core software executes a GET_WORK transaction to request new work. The POW hardware can schedule in-unit input-queue entries to cores in response to these requests. The POW hardware can also schedule descheduled work to cores in response to GET_WORKs. The POW hardware prioritizes descheduled work above input queue work. The POW scheduler never schedules a descheduled item that has a pending tag switch, and never schedules an input queue entry with an ATOMIC tag unless it can immediately have the tag. In other words, POW only schedules work when it can make forward progress. Input work-queue entries with the NULL tag type are a special case. The POW hardware immediately unschedules NULL type input queue work returned for a GET_WORK.

Work also enters the POW unit when an unscheduled (from the perspective of the POW hardware) core executes a SWTAG_FULL transaction. This is shown by the upward arrow in the bottom of Figure 7–1. Work that enters the POW hardware unit this way is immediately scheduled, and is then not distinguishable from other scheduled work.

Figure 7–1 also shows another interesting aspect of the POW hardware on the right side. The ordering guarantees for work as it flows through the POW unit.

First, if work is in an input queue in memory (at the top in Figure 7–1), POW keeps it strictly in-order on a per-queue basis.

Second, when work is in-flight (either scheduled or descheduled), ordering is strictly based on tag and tag type values. POW does not force any ordering nor synchronize in-flight work that uses different tag values or different tag type values. This in-flight work freely executes in parallel.

NOTE: The group identifier of work does not affect the ordering of in-flight work, it only affects the cores to which a descheduled item can be rescheduled.

Third, when work is both in an input queue and in the POW unit (i.e. between the memory input queues and the in-flight work), the work ordering guarantees are mixed. The POW hardware work scheduler skips past in-unit input-queue entries that cannot be immediately scheduled when it searches for schedulable work. The POW scheduler never skips past ORDERED and NULL input queue work, so the POW scheduler schedules work with these types (and the same group) strictly in per input queue order. The POW scheduler skips input queue work with the ATOMIC tag type and a tag that cannot immediately be scheduled, and so only guarantees tag order for ATOMIC input queue work (that has the same group). The POW work scheduler skips over input queue work that is not in the desired group, so no ordering is implied between two input-queue entries in different groups. Finally, at the bottom of Figure 7–1, unscheduled work is not synchronized by POW hardware and so is completely unordered.

The POW hardware maintains order across tag switches. Any in-flight work that executes the identical series of tag switches (each with the same tag/type values) while in-flight will be ordered identically through each switch. With proper configuration and software support, the POW hardware can totally order the processing of all packets in a flow. CN71XX provides total per-flow work ordering support for input packets (perhaps all the way to output) as long as the following

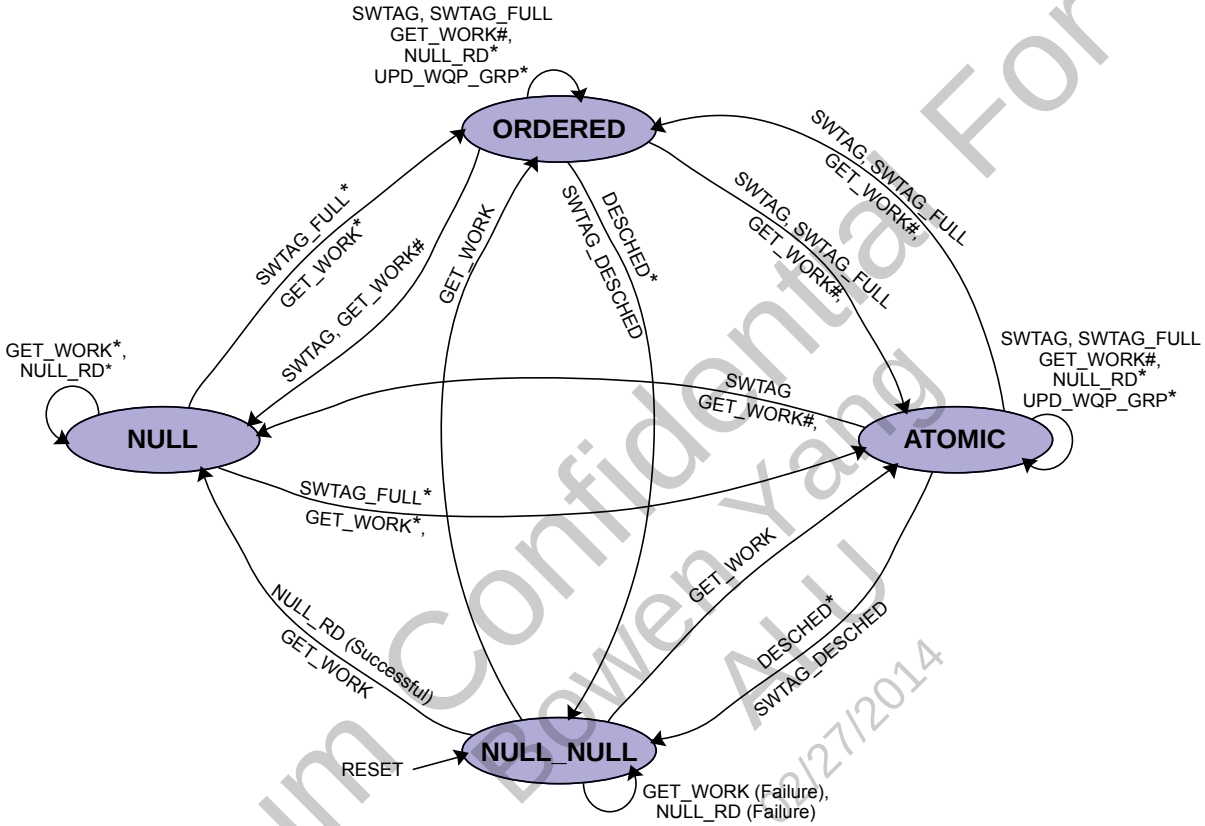
Cavium Confidential For Bowen Yang ALU 2/12/2014

conditions are true: (1) All packets from the same flow enter POW via the same input queue with the same initial tag value and group, and (2) The software processes packets from the same flow with the same sequence of non-NULL tag switches.

Figure 7–2 depicts an abstracted view of the POW core state (i.e. the POW states visible to a core) and the operations that affect it, focusing on the legal major operations in each state. A state has an arc tagged with a particular operation when it is legal to issue the operation in the state.

NOTE: The POW ADDWQ and NOP commands do not affect POW core state and are legal at any time. The POW CLR_NSCHED command does not affect POW core state and has its own issue rules.

Figure 7–2 The POW States Visible to a Core



*** = can start while a SWTAG or SWTAG_FULL is pending to the state**
 # = hardware inserts an implicit switch to NULL prior to the GET_WORK

Cavium Confidential For Bowen Yang ALU 12/27/2014

The abstracted states in Figure 7–2 closely mirror the tag types available in the tag switch operation. A new work request that receives work with an ORDERED, ATOMIC, or NULL tag puts the core into the ORDERED, ATOMIC, or NULL POW core state, respectively. A tag switch to an ORDERED, ATOMIC, or NULL tag type puts the core in the ORDERED, ATOMIC or NULL POW core state, respectively. These operations are depicted by arcs entering these states.

Figure 7–2 introduces the NULL_NULL state and the NULL_RD transaction. NULL_NULL is a special state entered only after a deschedule or reset. NULL_NULL and NULL_RD are required because a deschedule operation detaches an internal POW entry from a core and there may not be another entry available (see

Section 7.3, "POW Internal Architecture" on page 404 for more description of the POW hardware internals). NULL_NULL is similar to NULL, with the clear difference that it is illegal to SWTAG when in the NULL_NULL state. NULL_RD causes the POW hardware to attempt to convert the state to NULL when it is in the NULL_NULL state. (NULL_RD will fail when there are no more internal POW entries - see Section 7.10, "Forward Progress Constraints" on page 418 for forward-progress implications.)

The GET_WORK arcs exiting the ORDERED and ATOMIC states are special and marked with "#". These transactions implement multiple functions to release the prior work and schedule new work for this core. The POW hardware actually executes an implicit switch to NULL before executing the GET_WORK in these two cases. This implicit switch to NULL releases the prior work, so that the hardware always starts a GET_WORK from the NULL or NULL_NULL states. Note the implication that a GET_WORK from ORDERED or ATOMIC that does not successfully return work will change to the NULL state. A GET_WORK transaction from NULL_NULL that does not successfully return work will stay in the NULL_NULL state.

As in Figure 7–1, Figure 7–2 abstracts SWTAG, GET_WORK, and NULL_RD transactions as a single arc, though all these operations can have separate request and completion times. This is because the initial request solely determines the legal operations that can follow. The only question is whether the next legal transaction can start before the POW hardware completes the previous transaction. A tag switch transaction has explicitly-separated request and completion operations, but the get work and NULL_RD transactions are separated only with core IOBDMA operations. IOBDMA operations are described in Section 2.7. Here are the rules regarding transaction start time for the POW transactions that affect POW core state:

- SWTAG_DESCHED, DESCHED, UPD_WQP_GRP, and NOP transactions do not have separated start and completion times, so can be followed immediately by any legal command.
- The transactions marked "*" can start before the prior SWTAG is complete.
- In all other cases in Figure 7–2, A following transaction must not start before the prior transaction is complete.

The hardware behavior is unpredictable when the rules evident in Figure 7–2 are violated by core software. Note some specific restrictions:

- It IS NOT LEGAL to initiate a deschedule from the NULL or NULL_NULL POW core state.
- It IS NOT LEGAL to initiate any tag switch from the NULL_NULL state.
- It IS NOT LEGAL to initiate a tag switch with tag type of NULL from the NULL POW core state.
- It IS NOT LEGAL to issue any tag switch or get work operation while there is a pending switch with an ORDERED or ATOMIC tag type.
- It IS NOT LEGAL to initiate any transaction while a get work transaction is pending.
- It IS NOT LEGAL to initiate any transaction while a NULL_RD operation is pending.
- It IS NOT LEGAL to initiate a SWTAG_FULL or SWTAG_DESCHED transaction with tag type of NULL.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 7–1 details the available POW operations:

Table 7–1 POW Operations

Operation	Description
ADDWQ(tag_type, tag, wqp, grp, qos)	This adds work to the input queue selected by the QOS. Tag_type can legally be ATOMIC, ORDERED, or NULL. QOS is a 3-bit value, grp is a 4-bit value, and tag is a 32-bit value. The work-queue pointer (wqp) must be a 64-bit aligned pointer into L2/DRAM and must point to a legal work-queue entry (see Section 7.4, "Work-Queue Entry Format" on page 407). Furthermore, the work-queue entry group, tag type, and tag fields in the work-queue entry in L2/DRAM must exactly match the corresponding values supplied with the ADDWQ or the POW hardware may produce unpredictable results. The input work-queues are infinite, so this transaction never fails.
GET_WORK(wait)	This transaction attempts to get work for the requesting core. The value of POW_PP_GRP_MSK(0..3)[GRP_MSK] for the core at the time of the GET_WORK specifies the groups that are acceptable. The wait option causes the POW hardware to delay responding to the request until either work becomes available or the request times out. In any case, the POW hardware returns a failure response if it was unable to find work for the core, or a pointer to the work-queue entry if it successfully found work for the core. **NOTE:** It is possible, though unlikely, for a timeout to occur when the wait bit is clear, as well as when the wait bit is set, if the work search takes too long. The POW_PP_GRP_MSK that specifies the acceptable groups for a core must not be written between the start and completion of the GET_WORK, or unpredictable results may occur. Otherwise, the CSR can be written at any time. POW_NW_TIM[NW_TIM] specifies the configurable time-out counter interval that controls a single counter used for all cores. The POW hardware times out a GET_WORK request after two interval timer expirations, so the effective time-out interval varies between one and two times the configured interval. There are two new-work-request timeout cases: • WAIT bit clear. The new work request can time out if the timer expires before the prefetch engine has reached the end of all work queues. This can occur if the executable work queue entry is deep in the queue and the prefetch engine is subject to many resets (i.e. high switch, deschedule, or new work load from other cores). Thus, it is possible for a core to receive a work response with the NO_WORK bit set even though there was at least one executable entry in the work queues. The other (and typical) scenario for receiving a NO_WORK response with the WAIT bit clear is that the prefetch engine has reached the end of all work queues without finding executable work. • WAIT bit set. The new work request can time out if the timer expires before the prefetch engine has found executable work. In this case, the only scenario where the core will receive a work response with the NO_WORK bit set is if the timer expires. Note: it is still possible for a core to receive a NO_WORK response even though there was at least one executable entry in the work queues. In either case, it is important to note that switches and deschedules are higher priority operations that can cause the prefetch engine to reset. Thus in a system with many switches or deschedules occurring, it is possible for the new work timer to expire (resulting in NO_WORK responses) before the prefetch engine is able to get very deep into the work queues.

Cavium Confidential Draft Baichen Yang 2012-10-10

Table 7–1 POW Operations (Continued)

Operation	Description
NULL_RD	This transaction attempts to change to the NULL state when in the NULL_NULL state. If the core POW state is not NULL_NULL prior to the NULL_RD, POW will not change the core POW state. If the core POW state is NULL_NULL before the NULL_RD, but there are no available internal POW entries, POW fails the NULL_RD and does not change the core POW state. In all cases, POW returns the core state after processing the NULL_RD. When this returned state is NULL_NULL, the NULL_RD failed.
SWTAG(new_tag_type, new_tag)	This starts a tag switch transaction. The POW hardware completes the tag switch transaction later when it clears the pending switch bit for the core (refer to Section 7.5 for more information on POW tag switch-pending indications). An exception is a SWTAG to NULL, whose completion POW hardware never transmits. A SWTAG must not be used when switching from the NULL state. • A SWTAG from an ATOMIC tag releases the ATOMIC tag immediately once the tag switch transaction starts, perhaps long before the SWTAG transaction completes. • A SWTAG to an ATOMIC tag completes when the work acquires the new ATOMIC tag. At most one piece of work holds an ATOMIC tag at any time. The FIFO order is the acquisition order for the tag. • A SWTAG from an ORDERED tag cannot complete until all work ordered earlier in the old tag's FIFO start a SWTAG transaction. • A SWTAG to an ORDERED tag occurs immediately when switching from an ATOMIC or NULL tag, and occurs once the ordering constraints of the old tag are met when switching from an ORDERED tag.
SWTAG_FULL(new_tag_type, new_tag, new_wqp, new_grp)	This is identical to SWTAG, except that the transaction additionally updates the work-queue pointer (new_wqp) and 4-bit group identifier (new_grp) for the work that is held in the POW. SWTAG_FULL must be used for all switches from the NULL state. The POW hardware never interprets or uses the work-queue pointer supplied in this transaction, but it may deliver it to software later to complete a GET_WORK. The POW hardware stores <35:3> of the work-queue pointer. SWTAG_FULL must not be used for switches to NULL.
DESCHED(nosched)	This executes a deschedule transaction. When the nosched bit is set on DESCHED (or SWTAG_DESCHED) operations, the POW hardware does not schedule the packet to a core until a subsequent CLR_NSCHED operation clears the nosched bit for the POW entry. The POW entry can be determined with a POW status load with get_cur = 1 prior to the DESCHED (refer to Section 7.11.1). The index field in <50:40> identifies the POW ID. Note that it is recommended that the core be in ATOMIC state rather than ORDERED state at the time of the DESCHED (see Section 7.9). If nosched is set in the DESCHED, and if software somehow informs other cores that the POW entry is now on a deschedule list, software should first wait for the DESCHED to complete before informing the other cores. (Without this rule, it is possible for a CLR_NSCHED initiated by a different core to complete without seeing the POW entry on a deschedule list and without seeing the nosched bit set in the POW entry, even though the other core initiated the CLR_NSCHED after being explicitly informed that the POW entry was descheduled.) Software can determine when the DESCHED is complete via the pend_desched bit from a get_cur==0 POW status load (refer to Sections 7.8 and 7.11.1 for more details on POW status loads).
SWTAG_DESCHED(new_tag_type, new_tag, new_grp, nosched)	This is identical to a SWTAG followed by a DESCHED, except that it also updates the group. It must follow all the same rules as a separate SWTAG/DESCHED combo would. It must follow the same start rules as SWTAG (shown in Figure 7–2). The new_tag_type parameter cannot be NULL. SWTAG_DESCHED is well-suited for transferring work from one group to another - work pipelining.

Cavium Confidential For Boner Yang 02/27/2014

Table 7–1 POW Operations (Continued)

Operation	Description
UPD_WQP_GRP (new_wqp, new_grp)	Update the work-queue pointer (new_wqp) and group identifier (new_grp) for the work that is held in the POW.
CLR_NSCHED (wqp, index)	Clears the nosched bit for the POW entry selected by index. CLR_NSCHED is a NOP under any the following conditions: • the nosched bit is not set in the POW entry selected by index, or • the wqp in the POW entry does not match the supplied wqp Before initiating a CLR_NSCHED operation, software must guarantee that all *DESCHEDs and CLR_NSCHEDs are complete. software can read the pend_desched and pend_nosched_clr bits via POW status loads to determine when these conditions are true. (Refer to Sections 7.8 and 7.11.1 for more details on POW status loads.) It is not legal for a core to initiate a CLR_NSCHED while that core has either a GET_WORK or a NULL_RD operation pending. After a CLR_NSCHED operation, software must guarantee that the CLR_NSCHED is complete before issuing any subsequent POW operations. It can do this by checking the pend_nosched_clr via POW status read operations. Note also that index will typically be determined by POW status loads prior to the *DESCHED that set the nosched bit. A POW status load with get_cur = 1 returns the index field in <50:40>.
NOP	No Operation

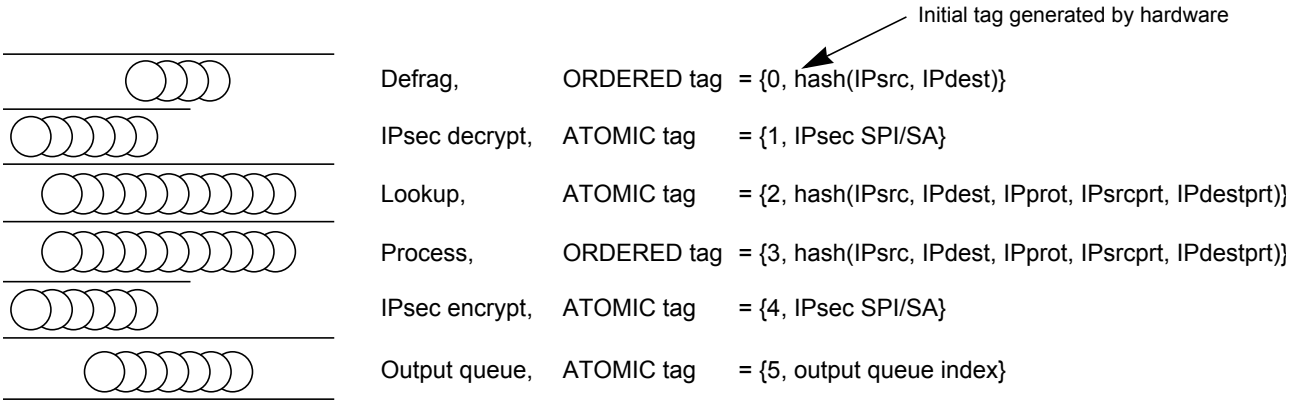
7.2 Example Software Architecture Using POW Ordering and Synchronization

This section describes one specific example of software usage of some of the POW hardware features. We describe this example only to illustrate a possible usage of some of the POW hardware features, not to advocate any of the particular techniques. The POW hardware synchronization support is very flexible and can be used in a huge variety of ways.

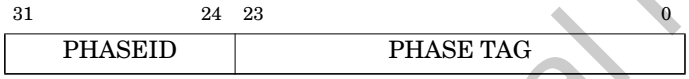
Figure 7–3 shows the simple synchronization architecture for the Firewall/VPN packet-processing example. This simple architecture assumes that there are at most six application phases during the processing of each individual packet: defrag, IPsec decrypt, Lookup, Process, IPsec encrypt, and output queue. It also assumes IPv4, but could be modified/generalized to IPv6.

Cavium Confidential For Bowen Yang ALC 6/2/2014

Figure 7–3 Architecture for the Firewall/VPN



The example software architecture in Figure 7–3 also uses some of the possible 32 tag bits to force unique tags in the different processing phases. For example, software could choose to partition the 32 tag bits in the following way:



Phase ID - Unique identifier for the phase
- 0x0 = Defrag
- 0x1 = IPsec Decrypt
- 0x2 = Lookup,
- 0x3 = Process
- 0x4 = IPsec encrypt
- 0x5 = Output queue

Phase TAG - tag value for within the phase

This phase tagging is not required by the hardware. It guarantees that different software processing phases execute in parallel in a pipelined fashion. The different PHASE ID values guarantee different tag values, so packets execute simultaneously with any other packet that is in a different phase. Of course, multiple packets can also execute freely within the same flow if they have different PHASE TAG values.

As is always possible with the POW hardware, the different PHASE TAGs architected in Figure 7–3 guarantee processing order among all packets in the same flow (and the same direction). Some phases also have ATOMIC tags to serialize access to shared data structures in some of the phases. Generally speaking, the phase tag value for each phase in the figure is either:

a. A hash result using the maximum number of packet fields to differentiate flows as much as possible, but still guarantee that two packets from the same flow produce the same hash value, or

b. An index into a critical data structure, or

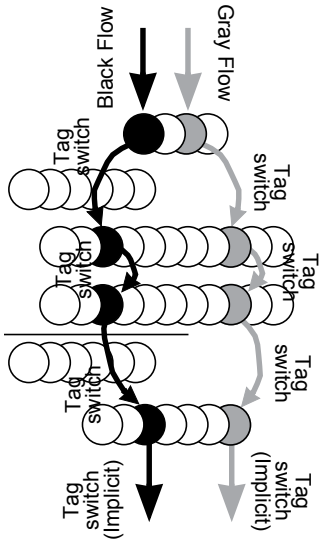
c. BOTH of the above.

The goal is to expose many different tag values to the POW hardware so it can parallelize as much work as possible, yet still guarantee strict in-order processing for packets in the same flow, and still guarantee that access to critical data structures is appropriately synchronized.

Cavium Confidential For Bolin Yang AU 02/27/2021

Figure 7–4 depicts the execution sequence using this software architecture for packets from two different flows: the gray flow and the black flow that pass through to different output queues without any IPsec processing by CN71XX. The Figure 7–4 example execution shows that the two different flows select different PHASE TAG values in every phase. Thus, the POW hardware will schedule the packets from the two flows entirely in parallel.

Figure 7–4 Software Execution Sequence



Initial tag generated by hardware

Defrag,	ORDERED tag	= {0, hash(IPsrc, IPdest)}
IPsec decrypt,	ATOMIC tag	= {1, IPsec SPI/SA}
Lookup,	ATOMIC tag	= {2, hash(IPsrc, IPdest, IPprot, IPsrcprt, IPdestprt)}
Process,	ORDERED tag	= {3, hash(IPsrc, IPdest, IPprot, IPsrcprt, IPdestprt)}
IPsec encrypt,	ATOMIC tag	= {4, IPsec SPI/SA}
Output queue,	ATOMIC tag	= {5, output queue index}

A tag switch defines the transition from one phase to the next in Figure 7–4, though the cores can work on the packet during a tag switch transaction and make the boundaries fuzzy. The packets from the gray and black flows execute four different phases to complete a packet, and four different tag switches occur for each packet.

The software need not issue a tag switch transaction for the first phase since the hardware creates the proper initial tag (Chapter 9, "Packet Input Processing/Input Packet Data Unit (PIP/IPD)" describes the programmability of the centralized packet-input processing and input-packet-data unit (PIP/IPD) that generates the initial tags), and the POW hardware only schedules the work after it has the initial tag. Though the last switch (to NULL that removes the work from POW) must execute inside POW, software need not specifically issue it since it can be an implicit part of the get work request for the next packet. So this architecture requires exactly three tag switches to execute the example pass-through packets.

Note also that in the example execution in Figure 7–4, the software simply skips the IPsec phases since no IPsec is required. As long as all packets in the same flow (and the same direction) skip the same IPsec phases, the switches in the IPsec phases can be avoided and the packets in the flow can still be kept in order by the POW hardware. Other flows may require IPsec processing, so may require up to two additional tag switches.

To explain the assumed ordering and synchronization requirements of the simple example architecture of Figure 7–4 in more detail, in the remainder of this section we briefly describe the functions assumed for each phase, and how the architected tags for each phase give the desired synchronization and ordering behavior.

Cavium Confidential For Edward Yang 02/21/2014

7.2.1 Defragmentation

This stage defragments input packets. All input packets enter this phase, some of which may be fragmented, and fewer defragmented packets exit.

The POW ORDERED tag type architected for this phase does not serialize the work within the defragmentation phase in any way, so some other means of memory synchronization (e.g. load-linked/store-conditional) would likely be required by software to correctly access the defragmentation data structure.

Any inbound packets in the same flow, fragmented or not, must have identical values for the IP source and destination address fields, and so will have the same PHASE TAG and will be ordered. Both unfragmented and defragmented packets from the same flow will enter the next stage in order.

The fragments that enter but do not exit this stage immediately switch to NULL and do not execute in any other phases. When the last fragment creates a completely defragmented packet, the defragmented packet is processed in subsequent phases in place of the input fragment, so the defragmented packet assumes the work order of the input fragment. This gives the defragmented packet the ideal ordering; as if it instantly appeared with the last input fragment, even though the software processing time needed to create the defragmented packet may be large.

7.2.2 IPsec Decryption

This stage performs IPsec decryption for packets that need it. If not needed, this phase is skipped entirely.

In general, inbound IPsec processing requires some synchronization between different packets that use the same IPsec tunnel for shared structures, like the anti-replay window. The architecture specifies the simplest possible solution for this synchronization; a single ATOMIC tag that covers the IPsec SA for the entire packet decryption. The PHASE TAG value uniquely identifies the SA, so after the inbound tag switch completes, the core can freely read and write the tunnel data structure until the next tag switch.

All packets from the same flow will have the same IPsec SPI/SA value and so will remain ordered as they exit this phase. Packets from other flows that use the same tunnel are also serialized through this phase, and will exit this phase in order, but will likely have a different tag in the next phase so will probably be unordered going into the next phase.

7.2.3 Lookup

This example assumes a stateful firewall, and this stage finds the flow (identified by a 5-tuple in this example) records and updates the state. On the first packet of a flow, a fast-path flow record will not exist for the flow, so the software will need to validate the flow and cache it for the fast-path flow records for later packets. The PHASE TAG architected for this phase selects a particular hash bucket. Thus, the ATOMIC tag serializes accesses to a bucket, and no further synchronization on the hash data structure may be required. No packets from a flow will observe (or modify) the flow state before the previous packets from the flow have updated the fast-path flow record.

With this cached architecture, it will likely take much longer to initially validate and cache the flow state than it will take to process subsequent packets. Thus, some packets may need to wait for the validation to complete before their lookup can continue. With some protocols, like TCP, these waits will not happen within the same

Cavium Confidential For Bowen Yang 02/27/2018

flow, but long waits are still possible with other protocols or with packets from a different flow that, unluckily, happen to collide in the same hash bucket. The deschedule operation provided by the POW hardware can be useful in these long-wait circumstances. The software can deschedule the current work and execute other work, and the POW hardware will reschedule the work later when the ATOMIC tag is available.

7.2.4 Process

This stage is for some processing of the packet that may be required before sending the packet out. For example, the packet may require NAT translation or TCP sequence number adjustment.

This stage uses an ORDERED tag with a hashed flow identifier for the PHASE TAG. This will force order for packets in the same flow as they exit this stage, as precisely as possible.

Of course, the firewall software may choose to drop a packet in either the lookup or process stages without executing any further stages.

7.2.5 IPsec Encryption

This stage performs IPsec encryption for packets that need it. If not needed, this phase is skipped entirely.

This stage's tag usage is very similar to the IPsec decryption stage.

7.2.6 Output Queue

This stage places the packet into an output queue to be sent out. The tag is ATOMIC to synchronize critical data structures needed for the enqueue. The PHASE TAG identifies the exact output queue, so only references that use the same output queue are serialized.

The firewall software may choose to perform quality-of-service (QOS) calculations (e.g. RED) at this stage to determine whether to drop the packet based upon queue sizes.

7.3 POW Internal Architecture

NOTE: For illustration purposes, Figure 7–5 shows five hypothetical cores. CN71XX has four cores and would perform the tasks differently.

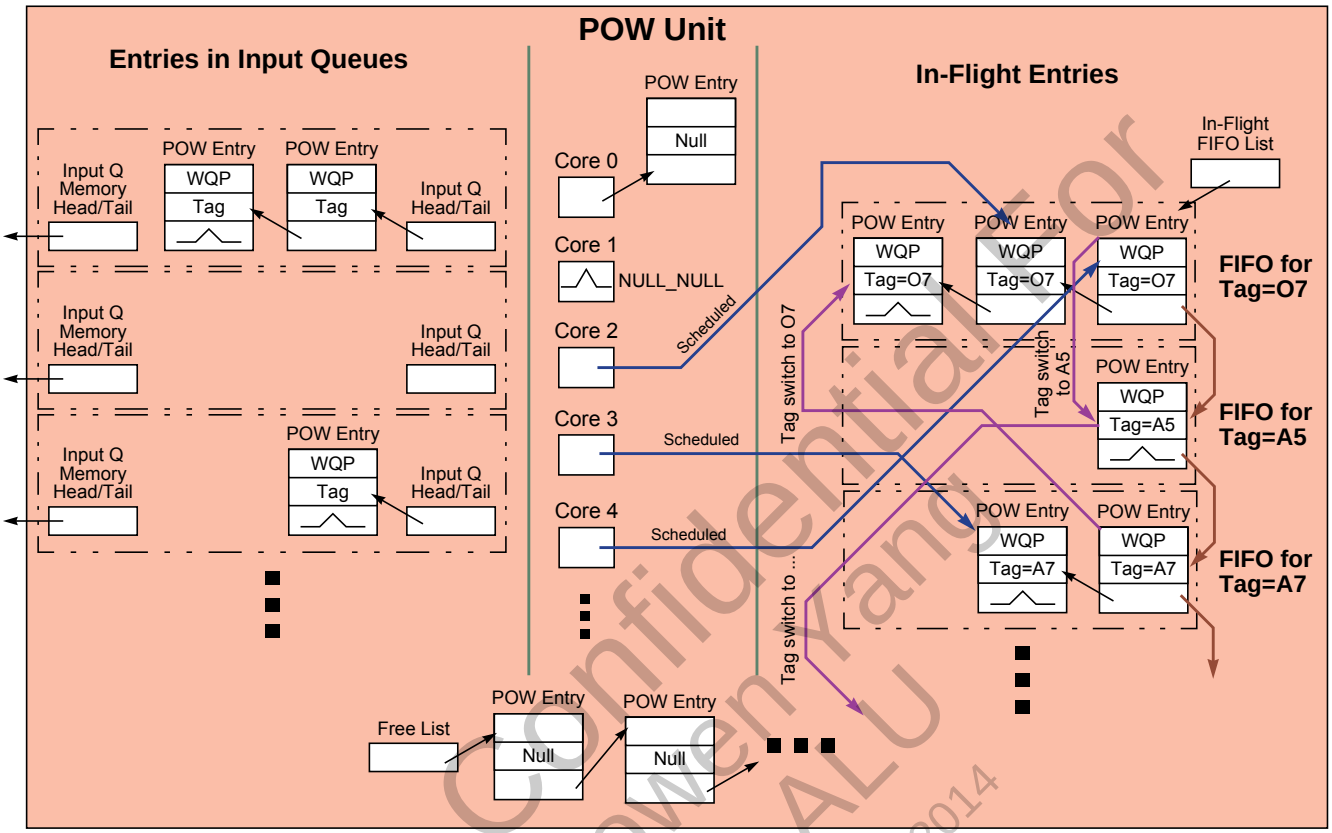
Figure 7–5 has a conceptual picture of the internal POW architecture. The actual implementation differs for performance reasons. The largest component of the POW unit is the 512 internal POW entries. As the figure shows, POW entries are either in input queues, in-flight, attached to a core that is in the NULL state, or in the free list. Each POW entry contains at least the following information:

- A pointer to the work-queue entry in L2/DRAM (WQP).
- The current tag and tag type
- The current group.
- Pointers to link the entry into various lists.

Cavium Confidential For Bowen Yang ALU 02/27/2014

The left side of Figure 7–5 shows the input queues. There are eight input queues. For each input queue, there is both a memory list and an in-unit list. As the POW hardware adds new work into the unit, either directly from an ADDWQ or from the head of the associated memory queue, it allocates an internal POW entry and fills it with the necessary information. Once the POW hardware allocates an internal POW entry for a piece of work, the work remains in the unit while it is in the input queue or in-flight. The POW hardware cannot overflow to L2/DRAM after that point. Core operations only cause the POW entry to attach/detach to/from particular cores, and move between lists.

Figure 7–5 Internal POW Architecture Components



Cavium Confidential - For Bowen Yang ALU 2/27/2014

NOTE: Once the POW unit loads work into the unit, it no longer reads or writes the work-queue entry locations in L2/DRAM. The work-queue entry in L2/DRAM is only used while work is in an input queue, but never while the work is in the unit. The POW hardware must carry the work-queue pointer along at all points when it is inside the unit, since the work-queue entry pointer is what (indirectly) describes the actual work that needs to be performed.

The right side of Figure 7–5 shows the in-flight POW entries. Conceptually, in-flight POW entries are always organized in FIFOs, with one FIFO associated with each unique in-flight tag and tag type value combination. For example, Figure 7–5 shows an O7 (tag = 7, type = ORDERED), A5 (tag = 5, type = ATOMIC), and A7 (tag = 7, type = ATOMIC) FIFO. Work first enters a FIFO when the POW hardware either schedules work from an input queue or switches from the NULL core state. SWTAG then causes POW entries to move from FIFO to FIFO. Note that there are no NULL FIFOs; work is no longer in-flight after a switch to NULL.

The FIFO orders in Figure 7–5 imply all ordering and synchronization constraints. For an ORDERED tag, the FIFO order indicates the order of the work holding the tag. For an ATOMIC tag, the FIFO order indicates the order that the work will switch to the ATOMIC tag, and the head of the list is the only work that has successfully switched to the ATOMIC tag. The POW hardware must move the head of one FIFO to the tail of another FIFO as part of any non-NULL tag switch. This FIFO move completes a switch to an ORDERED tag, but for an ATOMIC tag the work must further become the head of the new FIFO before the switch is complete. The work at the head of a FIFO never has a pending switch, or else the POW hardware would immediately execute the FIFO move required by the switch.

For example, in Figure 7–5, since the O7 FIFO is ORDERED, each of the three pieces of work has successfully switched to the O7 tag. The head of the O7 FIFO cannot have a pending switch, but the remaining work can have a pending switch to a different non-NULL tag. If non-head work has a pending FIFO move (because of a pending non-NULL SWTAG), the move cannot occur until the head moves. The A5 FIFO has only one entry so there can be no pending SWTAGs. The one piece of work holds the ATOMIC tag. In the A7 FIFO, the switch to A7 is complete for the head entry in the FIFO, but is pending for the other work. No work in any ATOMIC FIFO can have a pending SWTAG to a different tag.

The POW hardware implements the in-flight FIFOs as pure first-in, first-out lists, with one exception. The POW hardware can, in some cases, remove work from an ORDERED list out-of-order when the work has a pending SWTAG to NULL.

The middle of Figure 7–5 also depicts the POW core states. A core is either scheduled to in-flight work (i.e. in the ORDERED or ATOMIC states), in the NULL state with an attached POW entry, or in the NULL_NULL state. For example, Core 0 is in the NULL state; it has an attached POW entry, but the entry is not scheduled. Core 1 is in the NULL_NULL state; it neither has an attached POW entry nor is scheduled. Core 2 may currently be in the ORDERED state, or may have a pending switch to another state. Core 3 has a pending switch to the ATOMIC state. Core 4 is in the ORDERED state and has no pending switches.

The final POW architectural component shown in Figure 7–5 is the free list. The POW hardware uses the available POW entries to create in-unit input work-queue entries and to successfully complete core NULL_RD requests. If there are no free POW entries, the POW hardware can neither create in-unit work entries nor successfully complete core NULL_RD requests for cores in the NULL_NULL state. If there are fewer than eight free POW entries available, the POW hardware cannot create in-unit input work-queue entries, but can still successfully complete NULL_RD requests. This reservation of the last eight free-list entries makes it less likely that NULL_RD requests will fail. The POW hardware frees any POW entry previously attached to a core when it successfully completes a GET_WORK for the core. (The core would always have been in the NULL state when a POW entry frees, either because the GET_WORK initiated from the NULL state, or because of a switch to NULL implicitly inserted by the hardware prior to a GET_WORK transaction from ORDERED or ATOMIC.)

Cavium Confidential For Robert Yang AMD 08/2014

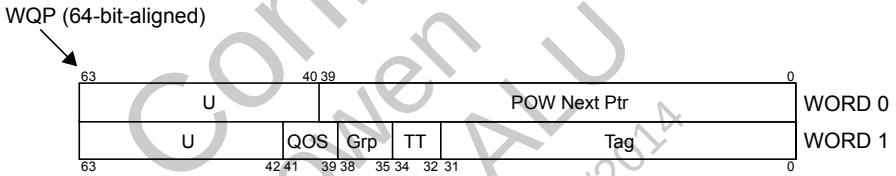
7.4 Work-Queue Entry Format

The work-queue entry in L2/DRAM is the primary descriptor that describes work. Though POW hardware does much to assist work queueing, scheduling, and synchronization, it doesn't dictate much about the format of the work-queue entry. POW hardware only uses the L2/DRAM work-queue entry in specific circumstances, and it uses few fields in those cases. The central Packet-Input Processing and Input-Packet Data Unit (PIP/IPD) may be the primary hardware component that constrains the work-queue entry format. (Chapter 9 describes the work-queue entries created by the PIP/IPD.) This section strictly discusses the requirements of the POW unit hardware.

First, it is only possible for the POW hardware to read/write to the L2/DRAM locations containing the work-queue entry at two times: (1) when processing the ADDWQ, and (2) when moving work from a memory input queue to a POW entry in the unit. At all other times the WQP held in a POW entry need only be a 64-bit aligned pointer (with at least <35:3> stored). The POW hardware stores the pointer and delivers the pointer to the work when a core executes a GET_WORK transaction.

Second, Figure 7–6 depicts the format requirements for the work-queue entry when it is used by the POW hardware. The POW Next Ptr field (in WORD 0) is used by hardware to link work in a memory input queue, so must not be used by software when work is created with ADDWQ. The POW hardware always reads the value of the QOS field in WORD 1 from L2/DRAM when a CN71XX hardware unit submits an ADDWQ, and so must be set appropriately by software in only that case. The POW hardware may also read the Grp, TT, and Tag fields from L2/DRAM for any ADDWQ. Software must guarantee that these L2/DRAM work-queue entry fields exactly match the same fields also supplied when software submits the ADDWQ transaction, or unpredictable results may occur.

Figure 7–6 Format Requirements for the Work-Queue Entry



U – Never used by POW hardware

POW Next Ptr – Written and read by POW hardware to implement memory input queue

QOS – Selects the input queue for the work. Read from L2/DRAM by the POW hardware when an OCTEON hardware unit initiates an ADDWQ.

Grp, TT, Tag – Select the group, tag type, and tag for the work. Read from L2/DRAM by the POW hardware either when an OCTEON hardware unit initiates an ADDWQ, or when moving work from a memory input queue into the unit.

Cavium Confidential For Bowen Yang ALU 12/27/2014

7.5 Core and Fetch-and-Add Pending Switch Bits

The core/fetch-and-add (FAU)/POW hardware SWTAG implementation includes switch pending bits held local to a core and inside the Fetch-and-Add unit inside the I/O bridge (IOB) block. The pending switch bits inside a core get set when the core starts a switch transaction by executing a store to a particular (I/O space) physical address destined for the POW hardware (refer to Section 7.11.3). The FAU hardware also sets its pending switch bit for a core when it sees the same store.

The SWTAG implementation also includes a dedicated bus that indicates switch completion. The pending switch bits in the core and in the FAU hardware get cleared by the dedicated switch completion bus.

The FAU hardware uses its pending switch bit to delay selected requests until after a tag switch completes. The core software uses the pending switch bit to determine when the switch completes.

It can locally sample its pending switch bit with the

```
RDHWR rt, $30
```

MIPS instruction, which returns a one when the pending switch bit is clear, and a zero when the pending switch bit is set. Core software must use this instruction to determine that the prior switch transaction is complete.

NOTE: A core can have, at most, one outstanding (non-NULL) switch, so only one pending switch bit is required.

The cores start SWTAG transactions by executing I/O store operations (<48> of the physical address set) with DID values of 0x60, 0x61, or 0x63 (in <47:40> of the physical address) selecting the POW unit. I/O store operations with these DID values are POW unit stores.

Here are the precise rules regarding the pending switch bit in a core:

- POW unit store operations with DID = 0x60 set the bit
- POW unit store operations with DID = 0x63 clear the bit
- A tag switch completion from POW clears the bit

Here are the rules regarding the pending switch bits in the Fetch-and-Add logic:

- POW unit store operations with DID = 0x60 sets the bit corresponding to the core
- A tag switch completion from POW clears the bit corresponding to the core

A SWTAG transaction starts with a POW unit store operation to DID = 0x60 that sets the pending switch bits in both the core and the FAU. Normally, the tag switch completes with the tag switch completion from POW on the special switch bus, which clears all the pending switch bits. Here are the exception cases:

- The POW hardware never sends a switch completion for a SWTAG to NULL.
- The POW hardware will not send a switch completion for a piece of work after the core executes a DESCHED transaction to deschedule and detach the work from the core.

Cavium Confidential For Bowen Yang ANU 2/27/2014

Given this pending switch bit behavior, software should use the following rules for POW unit I/O store operations:

- DID = 0x60 should be used to start non-NULL SWTAG and SWTAG_FULL transactions. This sets the pending switch bit in the core and in the Fetch-and-Add unit.
- DID = 0x63 should be used to start all DESCHED and SWTAG_DESCHED transactions. This clears the pending switch bit in the core, ensuring that it is not stale for any subsequent work.
- DID = 0x61 should be used to start ADDWQ, NULL SWTAG, UPD_WQP_GRP, CLR_NSCHED, and NOP transactions. These will affect no pending switch bit.

Core software should not initiate any DESCHED transaction while there is an in-flight Fetch-and-Add request that is waiting for a tag switch completion. (However, it is legal to issue a DESCHED after a Fetch-and-Add request fails due to a timeout on a tag switch wait.) Note also that the Fetch-and-Add pending switch bit may conservatively remain set after a DESCHED, so core software should only use the Fetch-and-Add tagwait feature when a tag switch is known to be outstanding.

7.6 POW Interrupts

There are two categories of POW interrupts: per-packet/work-receipt interrupts and input-queue threshold interrupts.

7.6.1 Per-Packet/Work Receipt Interrupts

The highest-performance CN71XX core software avoids the high cost of interrupts on packet receipt. Instead, this software will poll the POW hardware scheduler to find work. However, the POW hardware does support coalescing interrupts for those applications that require them. The interrupts can occur based on either work counts or time.

The goal of the POW per-packet/work interrupt hardware is to produce interrupts when work is present and can be executed by a core. All work is not considered equal with respect to the interrupts generated by the POW hardware. Rather, the POW hardware accounts for and controls interrupts on a per-group basis. Software can configure and accept interrupts separately for each core.

There are two distinct ways a piece of work in a particular group is “executable” in the POW unit:

- IQ executable
- DS executable.

IQ Executable

The work is present in one of the eight POW input work queues and is held in one of the 512 entries inside the POW unit. Note specifically that work held in one of the eight input queues, but not yet held inside the POW unit (i.e. resides in L2/DRAM), is NOT considered “executable.” This is because the POW hardware cannot schedule work to a core if it is not currently in the POW unit.

Note that though work is IQ executable and an interrupt is generated, it is still possible for a subsequent interrupt handler request for work to fail to return work from the interrupting group. Here are the ways this is possible:

Cavium Confidential For Bowen Yang ALU 02/27/2014

1. Of course, if the interrupt handler did not include the interrupting group as an acceptable one (in POW_PP_GRP_MSK(0..3)) before the request, the hardware cannot return work for the interrupting group. Also, if the interrupt handler requests work from more than just the interrupting group, the POW hardware may return work from a different group.
2. This core (or any other core using the same group) may have been given new work from the group while the hardware was generating the interrupt, but before the interrupt handler requests work.
3. IQ work with an atomic tag may be constantly "IQ executable," though have a tag conflict with some other piece of in-flight work that constantly prevents the POW hardware from scheduling it.
4. The POW hardware may time out if it is forced to search too deep from among the DS or IQ executable items to supply work for a core.

   This can happen while attempting to acquire a particular "IQ Executable" piece of work if there is:

   - too many work items for different groups "earlier" in the in-unit input queue(s), or
   - too many work items for the same group, but with an atomic tag conflict "earlier" in the in-unit input queue(s).

The next section describes the POW hardware scheduling search priority/order and will explain the concept of "earlier" in the input queue(s).

DS Executable

The work was previously descheduled, and is now ready to be rescheduled again.

DS executable work is generally higher priority than IQ executable work since the POW hardware schedules previously descheduled work before it schedules new input queue work. Also, tag conflicts with descheduled (or other in-flight) work can prevent work (in this or any other group) from executing/proceeding, unlike IQ executable work.

Note that a descheduled item with the nosched bit set is not considered to be executable, as it cannot be scheduled to a core.

7.6.1.1 Work-queue Interrupt Logic

Here is the work-queue interrupt logic inside POW that is replicated for each of the 16 groups:

- A counter of the number of work-queue entries that are IQ executable (POW_WQ_INT_CNT(0..15)[IQ_CNT]) and an associated interrupt threshold (POW_WQ_INT_THR(0..15)[IQ_THR]).
- A temporary IQ executable interrupt threshold disable bit (POW_WQ_INT[IQ_DIS]). With the IQ executable interrupt threshold disable bit, software can prevent continual interrupts when the IQ executable count exceeds its threshold, but cannot be reduced due to cases (3) or (4) outlined above.

  While the disable count bit is set, the threshold check on the IQ executable count cannot generate an interrupt until the next time expiration. This bit can be written to one by software, and is cleared by hardware. The hardware clears the IQ executable interrupt threshold disable bit under these conditions:

  a. The IQ executable count is zero for the group, or
  b. The hardware decrements the time counter for this group to zero.

Cavium Confidential For Bowen Yang AL U 02/27/2014

- A counter of the number of work-queue entries that are DS executable (POW_WQ_INT_CNT(0..15)[DS_CNT]) and an associated interrupt threshold (POW_WQ_INT_THR(0..15)[DS_THR]).
- A time counter (POW_WQ_INT_CNT(0..15)[TC_CNT]), an associated time threshold (POW_WQ_INT_THR(0..15)[TC_THR]), and a time threshold interrupt enable bit (POW_WQ_INT_THR(0..15)[TC_EN]).

  The time counter decrements whenever either the IQ executable count is non-zero or the DQ executable count is non-zero for the group.

  The hardware sets the time counter to the threshold value whenever:

  a. both the IQ and DS counts are zero, or
  b. the interrupt bit for this group is written with a 1 (to clear), or
  c. the time threshold value is written, or
  d. the temporary IQ executable interrupt threshold disable bit is written to a one (to set it), or
  e. the time counter is zero

- An interrupt bit (POW_WQ_INT[WQ_INT<*>]).

  This bit is set by hardware and can be cleared by software. The POW hardware sets this bit under any/all of the following three conditions:

  a. The IQ executable count is ≥ its threshold and the IQ threshold check is enabled (the threshold check is disabled when the threshold is zero or when the temporary IQ executable interrupt threshold disable bit is set).
  b. The DS executable count is ≥ its threshold and the DS threshold check is enabled (the threshold check is disabled when the threshold is zero).
  c. It decrements the time counter to zero and the time count check is enabled.

The POW hardware decrements the per-group time counters at a frequency selected by a single counter (POW_WQ_INT_PC[PC]) and associated period (POW_WQ_INT_PC[PC_THR]) that is shared by all groups. The POW hardware loads the counter with POW_WQ_INT_PC[PC_THR] × 256 + 255 initially and whenever POW_WQ_INT_PC[PC_THR] is written, and decrements the counter once every cycle otherwise. When the counter reaches zero, the hardware reloads the shared counter and decrements each of the per-group time counters by one if the associated IQ executable or DS executable counts are non-zero, whether the time threshold interrupt is enabled or not.

As touched upon earlier in this section, software must handle any discrepancies that occur from the fact that the IQ executable count for a group may not match the number of successful work requests for the group that follow.

Cavium Confidential For Bowen Yang 02/27/2014

Here are some possible ideas, not requirements, to alleviate possible interrupt forward-progress issues if they arise:

- Use the temporary IQ executable interrupt threshold disable bit to prevent continual interrupts from cases (3) or (4) above.
- Use only the time threshold interrupts and not the executable count threshold interrupts to avoid continual interrupts from cases (3) or (4).
- Use dedicated input queues containing only interrupt traffic and/or dedicated input queues containing only specific groups to reduce the probability of problems with (4) above.
- Don't use atomic tags for the interrupt word in the IQ if (3) above is a problem.

These POW work-queue interrupts eventually become CIU_INT*n*_SUM0/4[WORKQ], which CIU distributes to processor cores and PCIe. Refer to Chapter 13.

7.6.2 Input-Queue Threshold Interrupts

The second category of POW interrupts are input-queue threshold interrupts. These interrupts can be used to manage the total number of work-queue entries in an input queue. The total number of work-queue entries in input queue *n*, which includes work-queue entries that are in-unit and in L2/DRAM, is POW_IQ_CNT*n*[IQ_CNT].

There is one threshold interrupt, corresponding interrupt enable, and corresponding interrupt threshold for each POW input queue:

- The interrupt for input queue *n* is POW_IQ_INT[IQ_INT<*n*>].
- The enable for input queue *n* is POW_IQ_INT_EN[INT_EN<*n*>].
- The threshold for input queue n is POW_IQ_THR*n*[IQ_THR].

The POW hardware sets the interrupt bit (POW_IQ_INT[IQ_INT<*n*>]) whenever the total number of work-queue entries in the input queue (POW_IQ_CNT*n*[IQ_CNT]) changes value to equal the threshold for the input queue (POW_IQ_THR*n*[IQ_THR]). The hardware does not generate these interrupts when the number of work-queue entries is not changing. Software can clear the interrupt bit by writing a one to it. Conceptually, it is not possible for the number of work-queue entries to change by more than one any time that it changes. This means that the threshold is a barrier: if the number of work-queue entries is currently below the threshold, an interrupt will always be asserted if the number later meets or exceeds the threshold, and if the number of work-queue entries is currently above the threshold, an interrupt will always be asserted if the number later meets or falls below the threshold.

It is possible for software to generate both high and low watermark interrupts, even though the hardware implements only a single threshold. Start with the threshold set at the high watermark. Once the number of work-queue entries in an input queue rises to meet the high watermark, the POW hardware will generate an interrupt. At that point, the software interrupt handler can switch the threshold to the low watermark and clear the interrupt. Then, once the number of work-queue entries in an input queue falls to meet the low watermark, the POW hardware will generate an interrupt, and the software interrupt handler can change the threshold to the high watermark and clear the interrupt.

Whenever any enabled POW input queue threshold interrupt asserts, CIU_INT*n*_SUM0/4[POWIQ] asserts. CIU can distribute this interrupt to processor cores and PCIe. Refer to Chapter 13.

Cavium Confidential For Boren Yang 02/27/2012

7.7 POW QOS Features

The POW hardware contains a number of quality-of-service (QOS) features to prioritize input queues differently and keep work flowing through the unit smoothly. The main features are thresholds and scheduling.

7.7.1 Thresholds

Having thresholds is a QOS feature to limit the number of in-unit POW entries consumed by individual input queues (i.e. to limit POW_QOS_THR*n*[BUF_CNT]). In-unit POW entries are consumed by work in input queues, in-flight work. In-flight work may be descheduled, and POW_QOS_THR*n*[DES_CNT] indicates the total number of descheduled work items in all groups. POW_QOS_THR*n*[FREE_CNT] indicates the number of available POW entries — a total number available for all groups.

The POW hardware implements the following two thresholds separately for each input queue:

- maximum number of in-unit work-queue entries: POW_QOS_THR*n*[MAX_THR] + 1.

  POW_QOS_THR*n*[BUF_CNT] does not exceed this threshold. If there are more than POW_QOS_THR*n*[MAX_THR] + 1 entries in the queue, those entries must reside in L2/DRAM, not in the POW unit.

- minimum number of free in-unit work-queue entries: POW_QOS_THR*n*[MIN_THR]

  If the number of available in-unit entries (i.e. POW_QOS_THR*n*[FREE_CNT]) is smaller than POW_QOS_THR*n*[MIN_THR] for the queue, no more entries from the queue are loaded into the unit.

7.7.2 Scheduling

The second QOS feature is the mechanisms to control the input-queue traversal. The POW hardware traverses all input queues in parallel when it is searching for new input-queue work to schedule to the cores. The POW hardware scheduling can implement, on a core-by-core basis, mixtures of static priority (based on the input queue) and weighted round-robin priority. This can programmably prevent cores from receiving work from particular input queues.

When scheduling work to a core, POW hardware first scans the list of descheduled work. If a previously descheduled, currently schedulable item exists for a group that the core resides in, the POW hardware schedules this work to the core. If such descheduled items are not available, the POW hardware attempts to schedule work from an input queue to the core in the following manner.

The input-queue search proceeds in configurable rounds. The software can configure the 32 rounds via the POW_QOS_RND*n*[RND_P*] fields. Each configured round is an 8-bit bit-mask indicating the input queues that participate in the round. In general, the more rounds that a queue can participate in (as selected by the POW_QOS_RND*n*[RND*] configuration), the higher is the priority of the queue.

The following pseudo-code conceptually describes the scheduling algorithm that the POW hardware implements for a GET_WORK operation when schedulable input-queue work is selected for the core:

Cavium Confidential For Bowen Yang AL 03/27/2014

```
get_work(core) {

for(iq = 0; iq < 8; iq++) {
   size[iq] = POW_QOS_THR[iq].BUF_CNT;         // # POW entries IQ has in unit
   pri[iq] = POW_PP_GRP_MSK[core].QOS<iq>_PRI; // priority of this IQ for this core
   if(pri[iq] == 0xF) size[iq] = 0;            // IQ may be disabled for the core
   if(size[iq]) wqe[iq] = first WQE in iq;
}
found_pri = 0xF;
round = current_round;
iq = current_iq;
grp_msk = POW_PP_GRP_MSK[core].GRP_MSK;

do {
   // check if this iq participates in this round
   mask = (POW_QOS_RND[round / 4] >> ((round & 0x3) * 8)) & 0xFF;
   if((mask & (1 << iq)) && size[iq]) {
     // check if this is the best candidate for this core, or:
     //   - the core is in the selected group, and
     //   - there are no (atomic) conflicts with in-flight work, and
     //   - we have not already found work of the same or higher priority
     if((grp_msk & (1 << wqe[iq].grp)) &&
       ((wqe[iq].type != ATOMIC) || no_matching_inflight_atomic_tag(wqe[iq].tag)) &&
        (pri[iq] < found_pri)) {
       found_wqe = wqe[iq];
       found_pri = pri[iq];
     }

     wqe[iq] = next WQE in iq;
     size[iq] --;
   }

   // advance to the next round/iq
   if(iq == 0) {
     iq = 7;
     round = (round + 1) % 32;
   }
   else
     iq--;
   // we are done once we find a work-queue entry
   // at the highest possible priority
   for(i = 0; i < 8; i++) {
     if(size[i] && (pri[i] < found_pri))
       break;
   }
   found = (i == 8) && (found_pri < 0xF);
} while(!found);

current_round = round;
current_iq = iq;
return(found_wqe);
}
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

The POW hardware scans the input queues starting at a current round and the current input queue. The remaining queues that participate in the current round are sequentially scanned for work first. After that, the rounds are scanned circularly. Each input queue is scanned in order, and different input queues are scanned in the order indicated by the round configuration.

The POW hardware selects the highest-priority schedulable input queue work for the core. Whenever the POW hardware schedules work from an input queue to a core, it updates the current queue/round so that the next work request starts from the queue/round that sequentially follows the position where the last schedulable work was found. The current queue/round is shared by all cores.

These are some notable aspects of the input queue scheduling algorithm:

- The POW hardware can only schedule in-unit work-queue entries, whose count for a given input queue is indicated by POW_QOS_THR*n*[BUF_CNT].
- Each core can reside in any combination of groups, indicated by POW_PP_GRP_MSK*n*[GRP_MSK].
- Each core can establish its own individual priority for each input queue, indicated by POW_PP_GRP_MSK*n*[QOS*_PRI]. The hardware can find work more quickly when the core uses fewer priorities, as the higher-priority queues must be traversed in entirety before lower-priority work can be scheduled.
- POW_QOS_RND*n* configuration always indicates the order that the input queues are traversed, regardless of the selected priorities.
- Weighted-round-robin is used between queues of the same priority. The POW_QOS_RND*n* configuration establishes the weighting.

  For example, the highest-weight queue might participate in all 32 rounds, while a low-weight queue might participate in only one of the 32 rounds. Assuming there are schedulable items in both these equal-priority-example queues, the POW hardware schedules 32 items from the high-weight queue and then schedules one item from the low-weight queue.

7.8 POW Debug Visibility

The POW hardware provides complete visibility into its current state through POW load operations. This visibility can be used for POW debug dumps and for probing current status. Section 7.11 describes the details of the POW status loads, POW memory loads, and POW index/pointer loads that return the POW state. This section describes them qualitatively.

Figure 7–5 figuratively indicates much of the POW state. The more detailed list is:

- 512 POW entries, linked in various in-unit queues and tag lists (i.e. the FIFO for a given tag)
- eight in-unit input queues, one per QOS level
- 16 external (i.e., overflow) input queues, two per QOS level
- 16 in-unit descheduled head queues, one per group
- a nosched descheduled head queue
- four core states, one per core
- a free list for POW entries
- a 16-entry[1] reserved free queue[2] for servicing NULL_RD requests from cores.
- a prefetched entry[2] used for ADDWQs.
- four prefetched entries[2] used for moving input queue entries from L2/DRAM into the unit.

POW status loads return the current state for individual cores. The POW core state includes:

1. This number is derived by taking the next power of two at or above the number of cores.
2. These nine entries (next power of two at or above the number of cores, plus 5) are not directly visible to software. All other POW entries can be accounted for by the in-unit input queues, free list, core-headed tag lists, or tag lists associated with the descheduled heads, though. The prefetched and reserved entries reside among the remaining unaccounted-for POW entries.

Cavium Confidential For Bowen Yang ALU 02/27/2014

- State bits indicating the operations that the POW is currently attempting to process. For example, there is a bit indicating that this core has a pending switch and has not left the original tag list.
- The current state / tag type of the core, as depicted in Figure 7–2.
- The current work-queue pointer, group, and tag for the POW entry attached to the core (if any POW entry is attached).
- The next/pending work-queue pointer, group, tag type, and tag (if there are pending operations).
- Head and tail bits, forward and reverse links to maintain the tag list that the POW entry is in (if any POW entry is attached).

Note that when a POW entry is attached to a core, the POW core state supersedes the POW entry status. (When the POW hardware attaches a POW entry to a core, it loads the POW entry state into the POW core state and does not update the POW entry state until it detaches it from the core.)

POW Memory loads return the current state of an individual POW entry through POW memory loads. The state of a POW entry includes:

- The current work-queue pointer, group, tag type, and tag of the POW entry.
- A bit indicating that the POW entry has a pending switch and has not left its original tag list, together with the tag type and tag for the new tag list.
- A forward link for the input, free, or descheduled head list that the POW entry is currently on (if any).
- A tail bit and forward link for the tag list that the POW entry is currently on (if any).
- A nosched bit.

POW index/pointer loads return the heads and tails for the following lists:

- the 8 in-unit input queues.
- the 16 external input queues, and which is the current head for reloading. (The hardware alternates between the two memory queues associated with a given input queue when it reloads. This is a higher-performance implementation than would be a single input-memory list per QOS level, as it allows the hardware to traverse the two input-memory lists in parallel.)
- the 16 descheduled head queues.
- the nosched descheduled head queue
- the free list.

To get a consistent snapshot of the POW state, it is best to capture it while all POW activity and operations have ceased. If these types of captures are not possible, then capture the POW state as quickly as possible to get a view that is as consistent as possible. Do this by first reading the core state, then the heads/tails of the various lists of POW entries, and then traversing the lists.

Note that for the POW hardware to correctly service the visibility (and CSR) read operations, software must restrict itself to at most one outstanding load/IOBDMA operation inflight from each individual core. This is most simply done by only using load operations to read the visibility information, and never using IOBDMA operations.

Cavium Confidential For Bowen Yang AMD 02/27/2014

7.9 POW Performance Considerations

The features provided by the POW hardware are so integral to the CN71XX design that its use can have important performance effects. This section describes how the POW performance interacts with, influences, and actually determines how other on-chip components will work.

First, the POW hardware does not ideally reschedule descheduled work with an ORDERED tag. The POW hardware can reschedule only the head of an ORDERED tag FIFO in some cases, though the ORDERED tag semantics allow it to schedule all the work in the FIFO. Core software is recommended to always use ATOMIC tags when descheduling. If ORDERED behavior was desired, core software can immediately switch to an ORDERED tag after it receives the rescheduled work.

Second, some notes on the raw CN71XX hardware latencies, as it is desirable for software to overlap the latencies with other useful operations:

- CN71XX requires approximately 40 cycles from the start of a SWTAG until it completes.
- CN71XX requires approximately 60 cycles from the start of a GET_WORK until it completes.

Third, some notes on the POW hardware behavior for the performance-critical SWTAG and GET_WORK transactions:

- SWTAG transactions are highest priority and always complete in near minimum latency (excluding bus contentions transferring the request to the unit) if there are no synchronization/ordering conflicts.
- The POW hardware prefetches work for all cores during cycles that would otherwise be idle. Thus, GET_WORK transactions can complete in near minimum latency (excluding bus contentions that transfer the load/IOBDMA to/from the unit) in many cases. However, changes to POW_PP_GRP_MSK*n* CSRs and SWTAG transactions both invalidate the prefetch.
- GET_WORK operations can be slower for cores that cannot accept all groups. When a core exists in fewer groups, the POW scheduler must be more selective, skipping over more work to find work that is schedulable for a core. It may be desirable to restrict work for particular groups to particular input queues, since this can ensure that there is some work for all/more groups at or near the head of an input queue, and can make work for all or more of the acceptable groups more easily visible to the prefetcher.
- GET_WORK operations can be slower for cores that use more priorities. When a core uses more input queue priorities, lower-priority work cannot be scheduled until all higher-priority in-unit queues have been scanned for schedulable work.
- GET_WORK prefetches are slower when the prefetcher must skip over a number of input-queue entries that have conflicting ATOMIC tags. Some applications may find it advantageous to avoid ATOMIC tags in the input queue.

Cavium Confidential For Bowen Yang 02/27/2014

7.10 Forward Progress Constraints

Care is necessary to guarantee that core software makes forward progress when it uses POW synchronization features. This section gives examples of some pitfalls and suggests some rules for core software to use that can avoid forward progress problems. These may not cover all circumstances, but touches on all the sources that can limit forward progress.

First, multi-constraint synchronization deadlocks must be avoided by core software. The POW hardware alone cannot be a source of multi-constraint synchronization deadlocks since each piece of work, and each core, has at most one active tag with a well-defined set of ordering / synchronization constraints. Of course, it is always possible for core software to create deadlock conditions if it combines POW synchronization constraints with other synchronization/locking constraints. (In CN71XX, core software can synchronize through means other than provided by the POW hardware, like load-linked/store-conditional sequences and Fetch-and-Add hardware sequences.) For example, if work that holds a POW atomic tag critical section cannot proceed until it enters another critical section, but the other critical section is held by work that is waiting for a tag switch to the same POW atomic lock, neither piece of work can make forward progress.

Second, when descheduling is used by any work using a particular tag value, all work using that particular tag value may need to use descheduling also. For example, a deadlock is present if a descheduled piece of work holds an atomic tag, and all cores are waiting for a pending switch to the same atomic tag, and none of the cores eventually deschedule. The work scheduled in the cores cannot make forward progress because of the descheduled work, and the descheduled work can never be scheduled because the cores never become available. It may be advantageous for software to partition the available tag/type values into deschedulable ones and non-deschedulable ones by convention. Non-deschedulable tag switches/waits can be faster for shortly-held and/or non-conflicting tags since they need not be encumbered with any code support for descheduling. Deschedulable tag switches may require higher overhead in the non-conflicting case, but still may be faster, for long-held conflicting tags.

Third, livelock and/or infinite loops must be avoided by core software. Core software needs to eventually make forward progress on scheduled work. It is always possible for core software to create a continual stream of tag switches or deschedules so that work never completes. This may be due to either an infinite loop or livelock condition. The POW scheduler assists somewhat in this task since it never reschedules work that has a pending tag switch, and never schedules an input queue entry that has an atomic tag conflict.

Fourth, in-unit input-queue entries and descheduled items must eventually schedule to core software. The POW scheduler's capability to search and find acceptable input work is limited by the number of entries in the unit, the group value, and tag conflicts. The most obvious way to find this pitfall is with groups or the nosched bit. The POW scheduler will never schedule work that has the nosched bit set or is to a group number that is not acceptable by cores, and work that cannot be scheduled due to this constraint will consume an entry in the POW unit. Of course, work can also schedule more freely when it has fewer tag synchronization constraints, so it is better to use more tags rather than fewer. A more subtle issue, which may be more of a performance problem than a hard forward-progress problem, can occur when input queue work requests a highly-contended atomic tag value. The POW scheduler will not schedule this input queue item whenever there is any in-flight work holding or switching to the same atomic tag. Another more subtle issue may occur when some input queues have entries in the POW unit, and remaining input queues do not have

Cavium Confidential For Bowen Zhang at U 6/8/2014

any entries in the unit because space is not available. Core software may eventually need to service the in-unit input-queue entries before the POW scheduler can select the in-memory input-queue entries.

Fifth, the in-flight full condition. This occurs when there are so many in-flight entries that the POW hardware cannot place any input-queue entries in the unit, and furthermore may not be able to successfully allocate a POW entry for a NULL_RD command. The POW hardware can, however, schedule descheduled items and service tag switches for existing POW entries when in an in-flight full condition. Core software must eventually accept and complete descheduled work to relieve an in-flight full condition. Deadlock could occur, for example, if all cores simply stall for successful NULL_RD commands following a deschedule or prior to a switch to NULL. The NULL_RDs may always fail due to the in-flight full condition, and the in-flight full condition may never clear out since no POW entries are freed when work completes. More simply, core software must not require that POW schedule an input queue entry before it can accept descheduled work.

7.11 POW Operations

This section shows the detailed bit formats and codes for the various transactions.

7.11.1 Load Operations

NOTE:
- Unused and unaddressed fields in the address are reserved and must be set to 0s by the software.
- Unused fields in the results are actually 0s.
- The POW load result is unpredictable when anything but a 64-bit load (i.e. anything other than an LD) is used.

GET_WORK Load Operations

Physical Address for a GET_WORK Load

48	47	40	39	4	3	2	0
1	DID 0110 0000		Reserved 0000 0000 0000 0000 0000 0000 0000 0000 0000		Wait	000	

- **Wait** - If set, don't return load result until work is available or time-out

NOTE: The POW hardware behavior is unpredictable when anything but a 64-bit load (i.e. anything other than an LD) is used for work request loads or NULL_RD loads.

Result for a GET_WORK Load

63	62	40	39	0
no_work	Reserved 000 0000 0000 0000 0000 0000		addr	

- **no_work** - Set when no new work-queue entry was returned
- **addr** - The work-queue pointer. Must be aligned on a 64-bit boundary. Unpredictable when no_work is set.

Cavium Confidential For Bowen Yang ALU 02/27/2014

POW Status Load Operations

Physical Address for a POW Status Load

48	47 40	39 10	9 6	5	4	3	2 0
1	DID 0110 0001	Reserved 0000 0000 0000 0000 0000 0000 0000 00	coreid	get_rev	get_cur	get_wqp	000

- **get_rev** - If set and **get_cur** is set, return reverse tag-list pointer rather than forward tag-list pointer
- **get_cur** - If set, return current status rather than pending status
- **get_wqp** - If set, get the work-queue pointer rather than tag/type

Result for a POW Status Load (when get_cur==0 and get_wqp==0)

63 62	61	60	59	58	57	56	55
unused 00	pend_switch	pend_switch _full	pend_switch _null	pend_desched	pend_desched _switch	pend_nosched	pend_new_work

54	53	52	51	50 40	39 36	35 34	33 32	31 0
pend_new_ work_wait	pend_null_rd	pend_nosched _clr	unused	pend_index	pend_grp	unused 00	pend_type	pend_tag

- **pend_switch** - set when there is a pending non-NULL SWTAG or SWTAG_FULL, and the POW entry has not left the list for the original tag.
- **pend_switch_full** - set when SWTAG_FULL and **pend_switch** is set.
- **pend_switch_null** - set when there is a pending NULL SWTAG, or an implicit switch to NULL.
- **pend_desched** - set when there is a pending DESCHED or SWTAG_DESCHED.
- **pend_desched_switch** - set when there is a pending SWTAG_DESCHED and **pend_desched** is set.
- **pend_nosched** - set when nosched is desired and **pend_desched** is set.
- **pend_new_work** - set when there is a pending GET_WORK.
- **pend_new_work_wait** - when **pend_new_work** is set, this bit indicates that the wait bit was set.
- **pend_null_rd** - set when there is a pending NULL_RD.
- **pend_nosched_clr** - set when there is a pending CLR_NSCHED.
- **pend_index** - this is the index when **pend_nosched_clr** is set.
- **pend_grp** - this is the **new_grp** when (**pend_desched** AND **pend_desched_switch**) is set.
- **pend_type** - this is the tag type when **pend_switch** or (**pend_desched** AND **pend_desched_switch**) are set.
- **pend_tag** - this is the tag when **pend_switch** or (**pend_desched** AND **pend_desched_switch**) are set.

Result for a POW Status Load (when get_cur==0 and get_wqp==1)

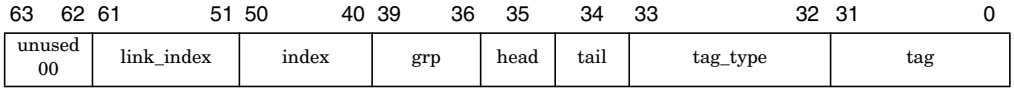
63 62	61	60	59	58	57	56	55
00	pend_switch	pend_switch _full	pend_switch _null	pend_desched	pend_desched _switch	pend_nosched	pend_new_work

54	53	52	51	50 40	39 36	35 0
pend_new_work_wait	pend_null_rd	pend_nosched_clr	unused	pend_index	pend_grp	pend_wqp

- **pend_wqp** - this is the **wqp** when **pend_nosched_clr** is set.

Cavium Confidential For Bowen Yang 02/17/2014

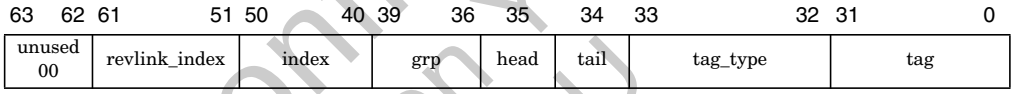
Result for a POW Status Load (when get_cur==1, get_wqp==0, and get_rev==0)



NOTE: The **link_index**, **grp**, and **tag** fields are unpredictable when the core's state is NULL or NULL_NULL.
The **index** field is unpredictable when the core's state is NULL_NULL.

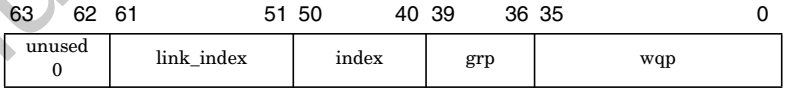
- **link_index** - points to the next POW entry in the tag list when **tail** == 0 (and **tag_type** is not NULL or NULL_NULL),.
- **index** - the POW entry attached to the core.
- **grp** - the group attached to the core (updated when new tag list entered on SWTAG_FULL).
- **head** - set when this POW entry is at the head of its tag list (also set when in the NULL or NULL_NULL state).
- **tail** - set when this POW entry is at the tail of its tag list (also set when in the NULL or NULL_NULL state).
- **tag_type** - the tag type attached to the core (updated when new tag list entered on SWTAG, SWTAG_FULL, or SWTAG_DESCHED).
- **tag** - the tag attached to the core (updated when new tag list entered on SWTAG, SWTAG_FULL, or SWTAG_DESCHED).

Result for a POW Status Load (when get_cur==1, get_wqp==0, and get_rev==1)



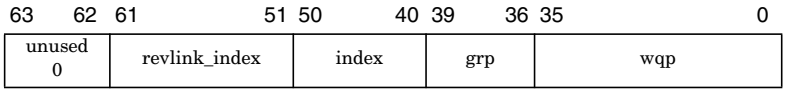
- **revlink_index** - points to the prior POW entry in the tag list when **head** == 0 (and **tag_type** is not NULL or NULL_NULL). This field is unpredictable when the core's state is NULL or NULL_NULL.

Result for a POW Status Load (when get_cur==1, get_wqp==1, and get_rev==0)



- **wqp** - the wqp attached to the core (updated when new tag list entered on SWTAG_FULL).

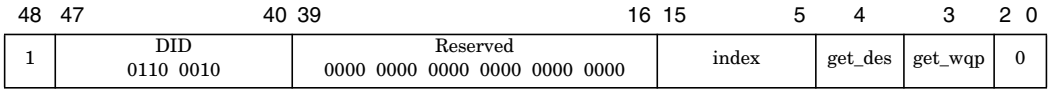
Result for a POW Status Load (when get_cur==1, get_wqp==1, and get_rev==1)



Cavium Confidential For Bowen Yang 08/27/2014

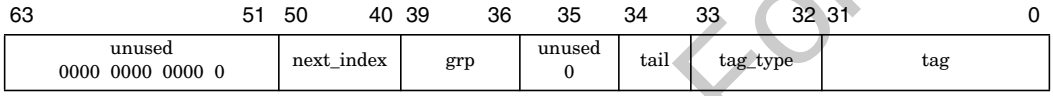
POW Memory Load Operations

Physical Address for a POW Memory Load



- **get_des** - If set, return deschedule information rather than the standard response for work-queue index (invalid if the work-queue entry is not on the deschedule list).
- **get_wqp** - If set, get the work-queue pointer rather than tag/type (no effect when **get_des** set).

Result For POW MemoryLoad (get_des == 0 and get_wqp == 0)



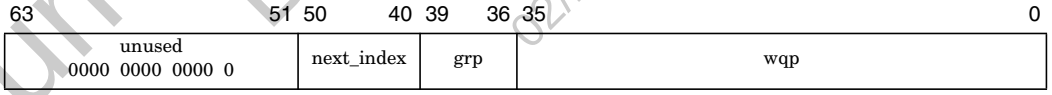
NOTE: All of these fields are unpredictable when the POW entry is currently scheduled to a core.

The **grp, tail, tag_type**, and **tag** fields are unpredictable when the POW entry is on a free list.

The **next_index** field is unpredictable when the POW entry is on the reserved free list.

- **next_index** - the next entry in the input, free, descheduled_head list (unpredictable if entry is the tail of the list).
- **grp** - the group of the POW entry.
- **tag_type** - the tag type of the POW entry.
- **tag** - the tag of the POW entry.

Result For POW Memory Load (get_des == 0 and get_wqp == 1)



NOTE: All of these fields are unpredictable when the POW entry is currently scheduled to a core.

The **grp** and **wqp** fields are unpredictable when the POW entry is on a free list.

The **next_index** field is unpredictable when the POW entry is on the reserved free list.

- **wqp** - the WQP held in the POW entry.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Result For POW MemoryLoad (get_des == 1)

63..51	50..40	39..36	35	34	33..32	31..0
unused 0000 0000 0000 0	fwd_index	grp	nosched	pend_switch	pend_type	pend_tag

NOTE: These fields are unpredictable when the POW entry is not descheduled (i.e., when the POW entry is currently scheduled to a core, in an input queue, or on a free list).

- **fwd_index** - the next entry in the tag list connected to the descheduled head.
- **pend_tag** - the next tag for the new tag list when **pend_switch** is set.
- **nosched** - the nosched bit for the POW entry.
- **pend_type** - the next tag type for the new tag list when **pend_switch** is set.
- **pend_tag** - the next tag for the new tag list when **pend_switch** is set.

POW Index/Pointer Load Operations

Physical Address for a POW Index/Pointer Load

48	47..40	39..9	8..5	4	3	2..0
1	DID 0110 0011	Reserved 0000 0000 0000 0000 0000 0000 0000 000	qosgrp	get_des_get_tail	get_rmt	000

NOTE: The POW hardware behavior is unpredictable when anything but a 64-bit load (i.e. anything other than an LD) is used for work request loads or NULL_RD loads.

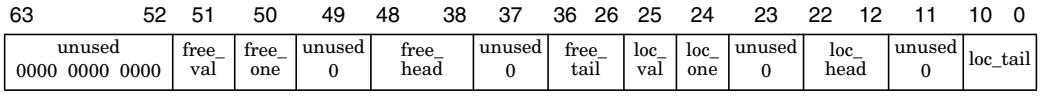
- **qosgrp** -
  - when {**get_rmt** ==0 AND **get_des_get_tail** == 0}, this field selects one of eight POW internal-input queues (0-7), one per QOS level; values 8-15 are illegal in this case;
  - when {**get_rmt** ==0 AND **get_des_get_tail** == 1}, this field selects one of 16 deschedule lists (per group);
  - when **get_rmt** ==1, this field selects one of 16 memory-input queue lists. The two memory-input queue lists associated with each QOS level are:

    qosgrp = 0, qosgrp = 8: QOS0
    qosgrp = 1, qosgrp = 9: QOS1
    qosgrp = 2, qosgrp = 10: QOS2
    qosgrp = 3, qosgrp = 11: QOS3
    qosgrp = 4, qosgrp = 12: QOS4
    qosgrp = 5, qosgrp = 13: QOS5
    qosgrp = 6, qosgrp = 14: QOS6
    qosgrp = 7, qosgrp = 15: QOS7

- **get_des_get_tail** - if set and **get_rmt** is clear, return deschedule list indexes rather than indexes for the specified qos level; if set and **get_rmt** is set, return the tail pointer rather than the head pointer for the specified qos level.
- **get_rmt -** if set, return remote pointers rather than the local indexes for the specified qos level.

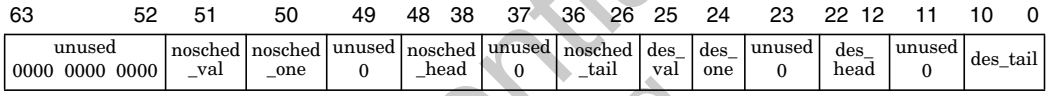
Cavium Confidential For Bowen Yang ALU 02/27/2014

Result For POW Index/Pointer Load (get_rmt == 0/get_des_get_tail == 0)



- **free_val** - set when there is one or more POW entries on the free list.
- **free_one** - set when there is exactly one POW entry on the free list.
- **free_head** - when **free_val** is set, indicates the first entry on the free list.
- **free_tail** - when **free_val** is set, indicates the last entry on the free list.
- **loc_val** - set when there is one or more POW entries on the input Q list selected by **qosgrp**.
- **loc_one** - set when there is exactly one POW entry on the input Q list selected by **qosgrp**.
- **loc_head** - when **loc_val** is set, indicates the first entry on the input Q list selected by **qosgrp**.
- **loc_tail** - when **loc_val** is set, indicates the last entry on the input Q list selected by **qosgrp**.

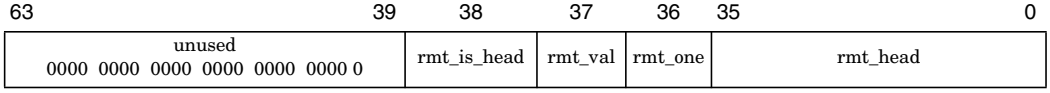
Result For POW Index/Pointer Load (get_rmt == 0/get_des_get_tail == 1)



- **nosched_val** - set when there is one or more POW entries on the nosched list.
- **nosched_one** - set when there is exactly one POW entry on the nosched list.
- **nosched_head** - when **nosched_val** is set, indicates the first entry on the nosched list.
- **nosched_tail** - when **nosched_val** is set, indicates the last entry on the nosched list.
- **des_val** - set when there is one or more descheduled heads on the descheduled list selected by **qosgrp**.
- **des_one** - set when there is exactly one descheduled head on the descheduled list selected by **qosgrp**.
- **des_head** - when **des_val** is set, indicates the first descheduled head on the descheduled list selected by **qosgrp**.
- **des_tail** - when **des_val** is set, indicates the last descheduled head on the descheduled list selected by **qosgrp**.

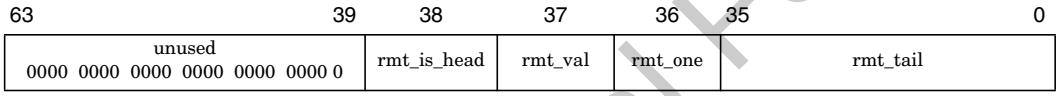
Cavium Confidential For Bowen Yang ALU 02/27/2014

Result For POW Index/Pointer Load (get_rmt == 1/get_des_get_tail == 0)



- **rmt_is_head** - set when this DRAM list is the current head (i.e. is the next to be reloaded when the POW hardware reloads a POW entry from DRAM). The POW hardware alternates between the two DRAM lists associated with a QOS level when it reloads work from DRAM into the POW unit.
- **rmt_val** - set when the DRAM portion of the input Q list selected by **qosgrp** contains one or more pieces of work.
- **rmt_one** - set when the DRAM portion of the input Q list selected by **qosgrp** contains exactly one piece of work.
- **rmt_head** - when **rmt_val** is set, indicates the first piece of work on the DRAM input Q list selected by **qosgrp**.

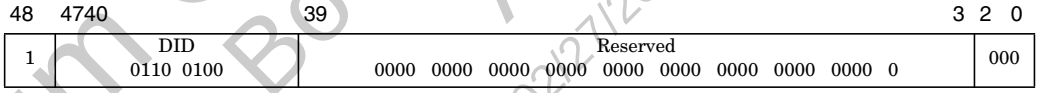
Result For POW Index/Pointer Load (get_rmt == 1/get_des_get_tail == 1)



- **rmt_is_head** - set when this DRAM list is the current head (i.e. is the next to be reloaded when the POW hardware reloads a POW entry from DRAM). The POW hardware alternates between the two DRAM lists associated with a QOS level when it reloads work from DRAM into the POW unit.
- **rmt_val** - set when the DRAM portion of the input Q list selected by **qosgrp** contains one or more pieces of work.
- **rmt_one** - set when the DRAM portion of the input Q list selected by **qosgrp** contains exactly one piece of work.
- **rmt_tail** - when **rmt_val** is set, indicates the last piece of work on the DRAM input Q list selected by **qosgrp**.

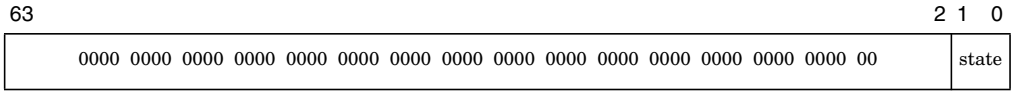
NULL_RD Load Operations

Physical Address for a NULL_RD Load



NOTE: The POW hardware behavior is unpredictable when anything but a 64-bit load (i.e. anything other than an LD) is used for work request loads or NULL_RD loads.

Result For NULL_RD Load



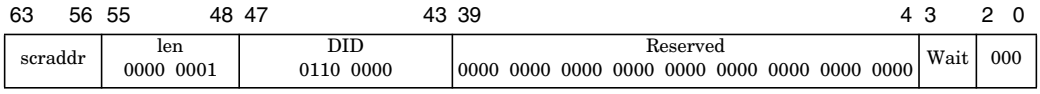
- **state** - The current POW state for the core after the NULL_RD operation.
  0x0 = ORDERED 0x2 = NULL
  0x1 = ATOMIC 0x3 = NULL_NULL

Cavium Confidential For Bowen Yang ALU 02/27/2014

7.11.2 IOBDMA Operations

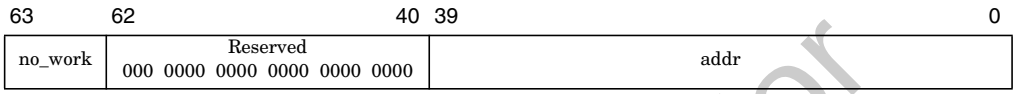
GET_WORK IOBDMA Operations

IOBDMA GET_WORK Addressing



- **scraddr** - Defined in "cnMIPS III Cores" on page 61.
- **Wait** - If set, don't return load Result until work is available or time-out

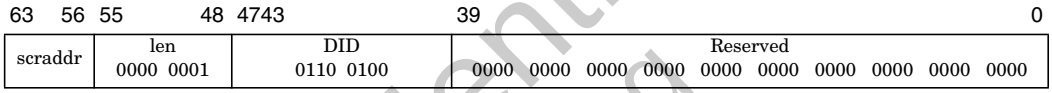
Result for an IOBDMA GET_WORK



- **no_work** - Set when no new work-queue entry was returned
- **addr** - The work-queue pointer. Must be aligned on a 64-bit boundary. Unpredictable when no_work is set.

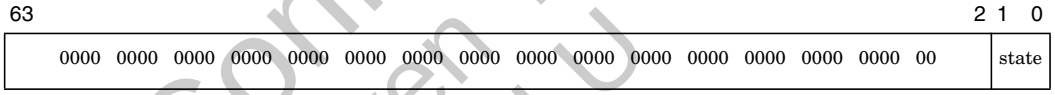
NULL_RD IOBDMA Operations

IOBDMA NULL_RD Addressing



- **scraddr** - Defined in "cnMIPS III Cores" on page 61.

Result For NULL_RD IOBDMA

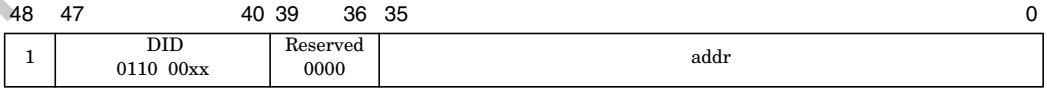


- **state** - NULL_NULL on failure

  0x0 = ORDERED 0x2 = NULL
  0x1 = ATOMIC 0x3 = NULL_NULL

7.11.3 Store Operations

Physical Address to Store to POW

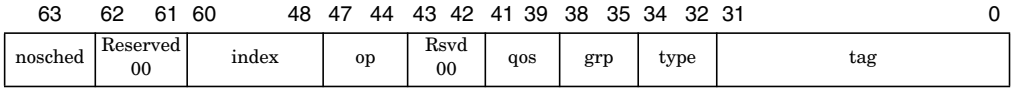


- **DID** = 0x60 for SWTAG_FULL, SWTAGs not switching to NULL
  DID = 0x63 for DESCHED, SWTAG_DESCHED
  DID = 0x61 otherwise
- **addr** - The address of the work-queue entry. Not used for SWTAG, SWTAG_DESCHED, DESCHED, NOP, addr<2:0> must be zero.

NOTE: The POW hardware behavior is unpredictable when anything but a 64-bit store (i.e. anything other than an SD) is used.

Cavium Confidential For Bowen Yang ALU 02/24/2014

Store Data on a Store to POW



- **nosched** - Don't reschedule this entry. nosched is used for SWTAG_DESCH and DESCHED
- **index** - Contains index of entry for a CLR_NSCHED
- **op** - The operation to perform

  The following different op tag types are used:

  0x0 = SWTAG
  0x1 = SWTAG_FULL
  0x2 = SWTAG_DESCHED
  0x3 = DESCHED
  0x4 = ADDWQ
  0x5 = UPD_WQP_GRP
  0x7 = CLR_NSCHED
  0xF = NOP

- **qos** - The QOS level for the packet. qos is only used for ADDWQ.
- **grp** - The group that the work-queue entry will be scheduled to. grp is used for ADDWQ, SWTAG_FULL, SWTAG_DESCH, and UPD_WQP_GRP
- **type** - The type of the tag. type is used for everything except DESCH, UPD_WQP_GRP, NOP, and CLR_NSCHED

  The following different tag types are used:

  0x0 = ORDERED
  0x1 = ATOMIC
  0x2 = NULL
  0x3 = NULL_NULL

- **tag** - The actual tag. tag is used for everything except DESCHED, UPD_WQP_GRP, NOP, and CLR_NSCHED

7.12 POW ECC Codes

Table 7–2 shows the POW 11-bit ECC code.

Table 7–2 POW 11-Bit ECC Code

CCCCC 4:0	10:8	7:4	3:0	
00001	100	1011	0111	0x4B7
00010	101	0101	1011	0x55B
00100	110	0110	1101	0x66D
01000	111	1000	1110	0x78E
10000	111	1111	0000	0x7F0
000000	000	0000	0000	
xxxxxx	xxx	xxxx	xxxx	
210000	111	1111	0000	← Syndromes
008421	FCA	9653	EDB7	

Cavium Confidential For Bowen Yang ALU 02/27/2014

7.13 POW Registers

The POW registers are listed in Table 7–3.

Table 7–3 POW Registers

Register	Address[1]	CSR Type[2]	Detailed Description
POW_PP_GRP_MSK(0..3)	0x1670000000000 + α*0x8	NCB	See page 429
POW_WQ_INT_THR(0..15)	0x1670000000080 + α*0x8	NCB	See page 429
POW_WQ_INT_CNT(0..15)	0x1670000000100 + α*0x8	NCB	See page 430
POW_QOS_THR(0..7)	0x1670000000180 + α*0x8	NCB	See page 431
POW_QOS_RND(0..7)	0x16700000001C0 + α*0x8	NCB	See page 431
POW_WQ_INT	0x1670000000200	NCB	See page 432
POW_WQ_INT_PC	0x1670000000208	NCB	See page 432
POW_NW_TIM	0x1670000000210	NCB	See page 433
POW_ECC_ERR	0x1670000000218	NCB	See page 434
POW_NOS_CNT	0x1670000000228	NCB	See page 435
POW_PF_RST_MSK	0x1670000000230	NCB	See page 435
POW_IQ_INT	0x1670000000238	NCB	See page 435
POW_IQ_INT_EN	0x1670000000240	NCB	See page 435
POW_WS_PC(0..15)	0x1670000000280 + α*0x8	NCB	See page 436
POW_WA_PC(0..7)	0x1670000000300 + α*0x8	NCB	See page 436
POW_IQ_CNT(0...7)	0x1670000000340 + α*0x8	NCB	See page 436
POW_WA_COM_PC	0x1670000000380	NCB	See page 436
POW_IQ_COM_CNT	0x1670000000388	NCB	See page 436
POW_TS_PC	0x1670000000390	NCB	See page 437
POW_DS_PC	0x1670000000398	NCB	See page 437
POW_IQ_THR(0..7)	0x16700000003A0 + α*0x8	NCB	See page 437
POW_BIST_STAT	0x16700000003F8	NCB	See page 437

1. For registers with multiple versions, α represents the values within the specified range.
2. NCB-type registers are accessed directly across the IOI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

POW Core Group Mask Registers
POW_PP_GRP_MSK(0..3)

These registers select which groups a core belongs to (one per core). A 1 in any bit position sets the core's membership in the corresponding group. A value of 0x0 prevents the core from receiving new work.

NOTE: Disabled or nonexistent cores should have this field set to 0xFFFF (the reset value) to maximize POW performance.

This register also contains the QOS level priorities for each core:

- 0x0 is highest priority.
- 0x7 is the lowest priority.
- 0xF prevents that core from receiving work from that QOS level.
- 0x8 – 0xE are reserved and should not be used.

For a given core, priorities should begin at 0x0 and remain contiguous throughout the range. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:44>	QOS7_PRI	R/W	0x0	—	Core(0..3) priority for QOS level 7.
<43:40>	QOS6_PRI	R/W	0x0	—	Core(0..3) priority for QOS level 6.
<39:36>	QOS5_PRI	R/W	0x0	—	Core(0..3) priority for QOS level 5.
<35:32>	QOS4_PRI	R/W	0x0	—	Core(0..3) priority for QOS level 4.
<31:28>	QOS3_PRI	R/W	0x0	—	Core(0..3) priority for QOS level 3.
<27:24>	QOS2_PRI	R/W	0x0	—	Core(0..3) priority for QOS level 2.
<23:20>	QOS1_PRI	R/W	0x0	—	Core(0..3) priority for QOS level 1.
<19:16>	QOS0_PRI	R/W	0x0	—	Core(0..3) priority for QOS level 0.
<15:0>	GRP_MSK	R/W	0xFFFF	0xFFFF	Core(0..3) group mask

POW Work-Queue Interrupt Threshold Registers
POW_WQ_INT_THR(0..15)

These registers contain the thresholds for enabling and setting work-queue interrupts (one per group). For more information on this register, refer to Section 7.6.

NOTE: Up to four of the POW's internal storage buffers can be allocated for hardware use and are therefore not available for incoming work-queue entries. Additionally, any core that is not in the NULL_NULL state consumes a buffer. Thus in a four-core system, it is not advisable to set either IQ_THR or DS_THR to greater than 512 – 4 – 4 = 504. Doing so could prevent the interrupt from ever triggering.

See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:29>	—	RAZ	—	—	Reserved
<28>	TC_EN	R/W	0x0	—	Time counter interrupt enable for group(0..15). This field must be 0 when [TC_THR] is 0.
<27:24>	TC_THR	R/W	0x0	—	Time counter interrupt threshold for group(0..15). When this field is equal to 0, POW_WQ_INT_CNT(0..15)[TC_CNT] is zero.
<23:21>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<20:12>	DS_THR	R/W	0x0	—	Deschedule count threshold for group(0..15). When this field is 0, the threshold interrupt is disabled.
<11:9>	—	RAZ	—	—	Reserved
<8:0>	IQ_THR	R/W	0x0	—	Input queue count threshold for group(0..15). When this field is 0, the threshold interrupt is disabled.

POW Work-Queue Interrupt Count Registers
POW_WQ_INT_CNT(0..15)

These registers contain read-only copies of the counts used to trigger work-queue interrupts (one per group). For more information on this register, refer to Section 7.6. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:28>	—	RAZ	—	—	Reserved
<27:24>	TC_CNT	RO	0x0	—	Time counter current value for group(0..15). Hardware sets this field to the value of POW_WQ_INT_THR(0..15)[TC_THR] whenever: • the corresponding IQ_CNT is equal to 0 and the corresponding DS_CNT is equal to 0. • the corresponding POW_WQ_INT[WQ_INT<*>] is written with a 1 by software. • the corresponding POW_WQ_INT[IQ_DIS<*>] is written with a 1 by software. • the corresponding POW_WQ_INT_THR(0..15) is written by software. • TC_CNT is equal to 1 and periodic counter POW_WQ_INT_PC[PC] is equal to 0. Otherwise, hardware decrements this field whenever the periodic counter POW_WQ_INT_PC[PC] is equal to 0. This field is 0 whenever POW_WQ_INT_THR(0..15)[TC_THR]is equal to 0.
<23:22>	—	RAZ	—	—	Reserved
<21:12>	DS_CNT	RO	0x0	—	Deschedule executable count for group(0..15)
<11:10>	—	RAZ	—	—	Reserved
<9:0>	IQ_CNT	RO	0x0	—	Input queue executable count for group(0..15)

Cavium Confidential For Edward Yang ALC 02/27/2014

POW QOS Threshold Registers
POW_QOS_THR(0..7)

These registers contain the thresholds for allocating POW internal storage buffers (one per QOS level). If the number of remaining free buffers drops below the minimum threshold (MIN_THR) or the number of allocated buffers for this QOS level rise above the maximum threshold (MAX_THR), future incoming work-queue entries are buffered externally rather than internally.

This register also contains a read-only count of the current number of free buffers (FREE_CNT), the number of internal buffers currently allocated to this QOS level (BUF_CNT), and the total number of buffers on the deschedule list (DES_CNT) (which is not the same as the total number of descheduled buffers). See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:58>	—	RAZ	—	—	Reserved
<57:48>	DES_CNT	RO	0x0	—	Deschedule list count. Number of buffers on the deschedule list.
<47:46>	—	RAZ	—	—	Reserved
<45:36>	BUF_CNT	RO	0x0	—	Buffer count. Number of internal buffers allocated to QOS level (0..7)
<35:34>	—	RAZ	—	—	Reserved
<33:24>	FREE_CNT	RO	0x1F7	—	Free buffer count. Number of total free buffers
<23:21>	—	RAZ	—	—	Reserved
<20:12>	MAX_THR	R/W	0x1FF	—	Max threshold for QOS level(0..7)
<11:9>	—	RAZ	—	—	Reserved
<8:0>	MIN_THR	R/W	0x0	—	Min threshold for QOS level (0..7)

POW QOS Issue Round Registers
POW_QOS_RND(0..7)

These registers contain the round definitions for issuing new work. Each round consists of eight bits, with each bit corresponding to a QOS level. There are four rounds contained in each of the eight registers for a total of 32 rounds. The issue logic traverses through the rounds sequentially (lowest round to highest round) in an attempt to find new work for each core. Within each round, the issue logic traverses through the QOS levels sequentially (highest QOS to lowest QOS) skipping over each QOS level with a clear bit in the round mask.

NOTE: Setting a QOS level to all 0s in all issue round registers prevents work from being issued from that QOS level.

See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:24>	RND_P3	R/W	0xFF	—	Round mask for (round # × 4 + 3)
<23:16>	RND_P2	R/W	0xFF	—	Round mask for (round # × 4 + 2)
<15:8>	RND_P1	R/W	0xFF	—	Round mask for (round # × 4 + 1)
<7:0>	RND	R/W	0xFF	—	Round mask for (round # × 4)

Cavium Confidential For Bowen Yang 02/27/2014

POW Work-Queue Interrupt Register
POW_WQ_INT

This register contains the bits (one per group) that set work-queue interrupts and are used to clear these interrupts. Also contains the input queue interrupt temporary disable bits (one per group). For more information on this register, refer to Section 7.6. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:16>	IQ_DIS	R/W1S	0x0	—	Input queue interrupt temporary disable mask. The corresponding [WQ_INT<*>] bit cannot be set due to IQ_CNT/IQ_THR check when this bit is set. The corresponding [IQ_DIS] bit is cleared by hardware whenever: • POW_WQ_INT_CNT(0..15)[IQ_CNT] = 0, or • POW_WQ_INT_CNT(0..15)[TC_CNT] = 1 when periodic counter POW_WQ_INT_PC[PC] = 0.
<15:0>	WQ_INT	R/W1C	0x0	—	Work-queue interrupt bits. The corresponding WQ_INT bit is set by hardware whenever: • POW_WQ_INT_CNT(0..15)[IQ_CNT] ≥ POW_WQ_INT_THR(0..15)[IQ_THR] and the threshold interrupt is not disabled. When IQ_DIS<*> = 1, the interrupt is disabled. When POW_WQ_INT_THR(0..15)[IQ_THR] = 0, the interrupt is disabled. • POW_WQ_INT_CNT(0..15)[DS_CNT] ≥ POW_WQ_INT_THR(0..15)[DS_THR] and the threshold interrupt is not disabled. When POW_WQ_INT_THR(0..15)[DS_THR] = 0, the interrupt is disabled. • POW_WQ_INT_CNT(0..15)[TC_CNT] = 1 when both periodic counter POW_WQ_INT_PC[PC] = 0 and POW_WQ_INT_THR(0..15)[TC_EN] = 1, and one of either POW_WQ_INT_CNT(0..15)[IQ_CNT] > 0 or POW_WQ_INT_CNT(0..15)[DS_CNT] > 0.

POW Work-Queue Interrupt Periodic Counter Register
POW_WQ_INT_PC

This register contains the threshold value for the work-queue interrupt periodic counter and also a read-only copy of the periodic counter. For more information on this register, refer to Section 7.6. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:60>	—	RAZ	—	—	Reserved
<59:32>	PC	RO	0x0	—	Work-queue interrupt periodic counter
<31:28>	—	RAZ	—	—	Reserved
<27:8>	PC_THR	R/W	0x0	—	Work-queue interrupt periodic counter threshold
<7:0>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Wang ALU 10/7/2014

POW New-Work Timer Period Register
POW_NW_TIM

This register sets the minimum period for a new-work-request timeout. The period is specified in n–1 notation, with the increment value of 1024 clock cycles.

NOTE: The maximum period for a new work request timeout is 2 × the minimum period.

The new-work-request timeout counter is reset when this register is written.

There are two new-work-request timeout cases:

- **WAIT bit clear**.

  The new-work request can timeout if the timer expires before the prefetch engine has reached the end of all work queues. This can occur if the executable work-queue entry is deep in the queue and the prefetch engine is subject to many resets (i.e. high switch, deschedule, or new-work load from other cores).

  Thus, it is possible for a core to receive a work response with the NO_WORK bit set even though there was at least one executable entry in the work queues. The other (and typical) scenario for receiving a NO_WORK response with the WAIT bit clear is that the prefetch engine has reached the end of all work queues without finding executable work.

- **WAIT bit set**.

  The new-work request can timeout if the timer expires before the prefetch engine has found executable work. In this case, the only scenario where the core will receive a work response with the NO_WORK bit set is if the timer expires.

  Note that it is still possible for a core to receive a NO_WORK response even though there was at least one executable entry in the work queues.

In either case, it is important to note that switches and deschedules are higher priority operations that can cause the prefetch engine to reset. Thus in a system with many switches or deschedules occurring, it is possible for the new-work timer to expire (resulting in NO_WORK responses) before the prefetch engine is able to get very deep into the work queues.

See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved
<9:0>	NW_TIM	R/W	0x0	0x4	New-work-timer period. 0x0 = 1024 clock cycles 0x1 = 2048 clock cycles 0x2 = 3072 clock cycles ... etc.

Cavium Confidential Pre
Bowen Yang
ALS
02/27/2014

POW ECC Error Register
POW_ECC_ERR

This register contains the single- and double-error bits and the corresponding interrupt-enable bits for the ECC-protected POW index memory, plus the syndrome value in the event of an ECC error.

It also contains the remote-pointer-error bit and its interrupt-enable bit. RPE is set when POW detects corruption on one or more of the input-queue lists in L2/DRAM (i.e. POW's local copy of the tail pointer for the L2/DRAM input queue did not match the last entry on the list). This is caused by L2/DRAM corruption, and is generally a fatal error because it likely caused POW to load bad work-queue entries.

Finally, it contains the following illegal-operation indicators and corresponding interrupt-enable bits:

IOP <28:16>	IOP_IE <44:32>	Illegal Operation
<28>	<44>	Received CSR load from core with CSR load pending.
<27>	<43>	Received DBG load from core with DBG load pending
<26>	<42>	Received ADDWQ with tag specified as NULL_NULL
<25>	<41>	Received illegal opcode
<24>	<40>	Received SWTAG/SWTAG_FULL/SWTAG_DESCH/DESCH/ UPD_WQP_GRP/GET_WORK/NULL_RD from core with CLR_N-SCHED pending
<23>	<39>	Received CLR_NSCHED from core with SWTAG_DESCH/DESCH/ CLR_NSCHED pending
<22>	<38>	Received SWTAG/SWTAG_FULL/SWTAG_DESCH/DESCH/ UPD_WQP_GRP/GET_WORK/NULL_RD from core with NULL_RD pending
<21>	<37>	Received SWTAG/SWTAG_FULL/SWTAG_DESCH/DESCH/ UPD_WQP_GRP/GET_WORK/NULL_RD from core with GET_WORK pending
<20>	<36>	Received SWTAG_FULL/SWTAG_DESCH from core with tag specified as NULL
<19>	<35>	Received SWTAG/SWTAG_FULL/SWTAG_DESCH from core with tag specified as NULL_NULL
<18>	<34>	Received SWTAG/SWTAG_FULL/SWTAG_DESCH/GET_WORK from core with pending tag switch to ORDERED or ATOMIC
<17>	<33>	Received SWTAG/SWTAG_DESCH/DESCH/UPD_WQP_GRP from core in NULL state
<16>	<32>	Received SWTAG/SWTAG_FULL/SWTAG_DESCH/DESCH/ UPD_WQP_GRP from core in NULL_NULL state

See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:45>	—	RAZ	—	—	Reserved
<44:32>	IOP_IE	R/W	0x0	—	Illegal-operation interrupt-enable bits.
<31:29>	—	RAZ	—	—	Reserved
<28:16>	IOP	R/W1C	0x0	0x0	Illegal-operation errors.
<15:14>	—	RAZ	—	—	Reserved
<13>	RPE_IE	R/W	0	—	Remote-pointer-error interrupt-enable bit.
<12>	RPE	R/W1C	0	0	Remote-pointer error.
<11:9>	—	RAZ	—	—	Reserved

Cavium Confidential For Power_Yang 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<8:4>	SYN	RO	—	—	Syndrome value (only valid when DBE or SBE is set).
<3>	DBE_IE	R/W	0	—	Double-bit-error interrupt enable.
<2>	SBE_IE	R/W	0	—	Single-bit-error interrupt enable.
<1>	DBE	R/W1C	0	0	Double-bit error.
<0>	SBE	R/W1C	0	0	Single-bit error.

POW Noschedule Count Register
POW_NOS_CNT

This register contains the number of work-queue entries on the noschedule list. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved
<9:0>	NOS_CNT	RO	0x0	—	Number of work-queue entries on the no-schedule list.

POW Prefetch Reset Mask Register
POW_PF_RST_MSK

This register resets the work prefetch engine when work is stored in an internal buffer (either when the add work arrives or when the work is reloaded from an external buffer) for an enabled QOS level (1 bit per QOS level). See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	RST_MSK	R/W	0x0	—	Prefetch engine reset mask.

POW Input-Queue Interrupt Register
POW_IQ_INT

This register contains the bits (one per QOS level) that can trigger the input queue interrupt. If POW_IQ_CNT#QOS# changes, an IQ_INT bit is set and the resulting value is equal to POW_IQ_THR#QOS#. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	IQ_INT	R/W1C	0x0	—	Input-queue interrupt bits.

POW Input-Queue Interrupt-Enable Register
POW_IQ_INT_EN

This register contains the bits (one per QOS level) that enable the input-queue interrupt. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	INT_EN	R/W	0x0	—	Input-queue interrupt enable bits.

Cavium Confidential For Bowen Yang ALU 02/20/14

POW Work-Schedule Performance Counter Registers
POW_WS_PC(0..15)

One per group; counts the number of work schedules for each group. Write to clear. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	WS_PC	R/W1C	0x0	—	Work-schedule performance counter for group number.

POW Work-Add Performance Counter Registers
POW_WA_PC(0..7)

One per QOS level; counts the number of add new work requests for each QOS level. Write to clear. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	WA_PC	R/W1C	0x0	—	Work-add performance counter for QOS level number.

POW Input-Queue Count Registers
POW_IQ_CNT(0...7)

This register contains a read-only count of the number of work-queue entries for each QOS level (one register per QOS level). See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	IQ_CNT	RO	0x0	—	Input-queue count for QOS level number.

POW Work-Add Combined Performance Counter Register
POW_WA_COM_PC

This register counts the number of add new work requests for all QOS levels. Write to clear. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	WA_PC	R/W1C	0x0	—	Work add combined performance counter

POW Input-Queue Combined-Count Register
POW_IQ_COM_CNT

This register contains a read-only count of the total number of work-queue entries in all QOS levels. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	IQ_CNT	RO	0x0	—	Input-queue combined count

Cavium Confidential For Bowen Yang ALU 02/27/2014

POW Tag-Switch Performance Counter Register
POW_TS_PC

This register counts the number of tag switch requests. Write to clear. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	TS_PC	R/W1C	0x0	—	Tag-switch performance counter.

POW Deschedule Performance Counter Register
POW_DS_PC

This register counts the number of deschedule requests. Write to clear. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	DS_PC	R/W1C	0x0	—	Deschedule performance counter.

POW Input Queue Threshold Register
POW_IQ_THR(0..7)

This register contains the threshold value for triggering input-queue interrupts (one per QOS level). See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	IQ_THR	R/W	0xFFFFFFFF	—	Input-queue threshold for QOS level (0..7).

POW BIST Status Register
POW_BIST_STAT

This register contains the BIST status for the POW memories: 0 = pass, 1 = fail. See Table 7–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved.
<11>	CAM	RO	0	0	POW CAM BIST status.
<10>	—	RAZ	—	—	Reserved.
<10:8>	NBR	RO	0x0	0x0	IOI receiver memory BIST status.
<7:8>	—	RAZ	—	—	Reserved.
<5:4>	NBT	RO	0x0	0x0	IOI transmitter memory BIST status.
<3>	IDX	RO	0	0	Index memory BIST status.
<2>	FIDX	RO	0	0	Forward index memory BIST status.
<1>	PEND	RO	0	0	Pending switch memory BIST status.
<0>	ADR	RO	0	0	Address memory BIST status.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 8

Free Pool Unit (FPA)

This chapter contains the following information about the Free Pool Unit (FPA).

- Overview
- Free Pool Unit Operations
- FPA Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

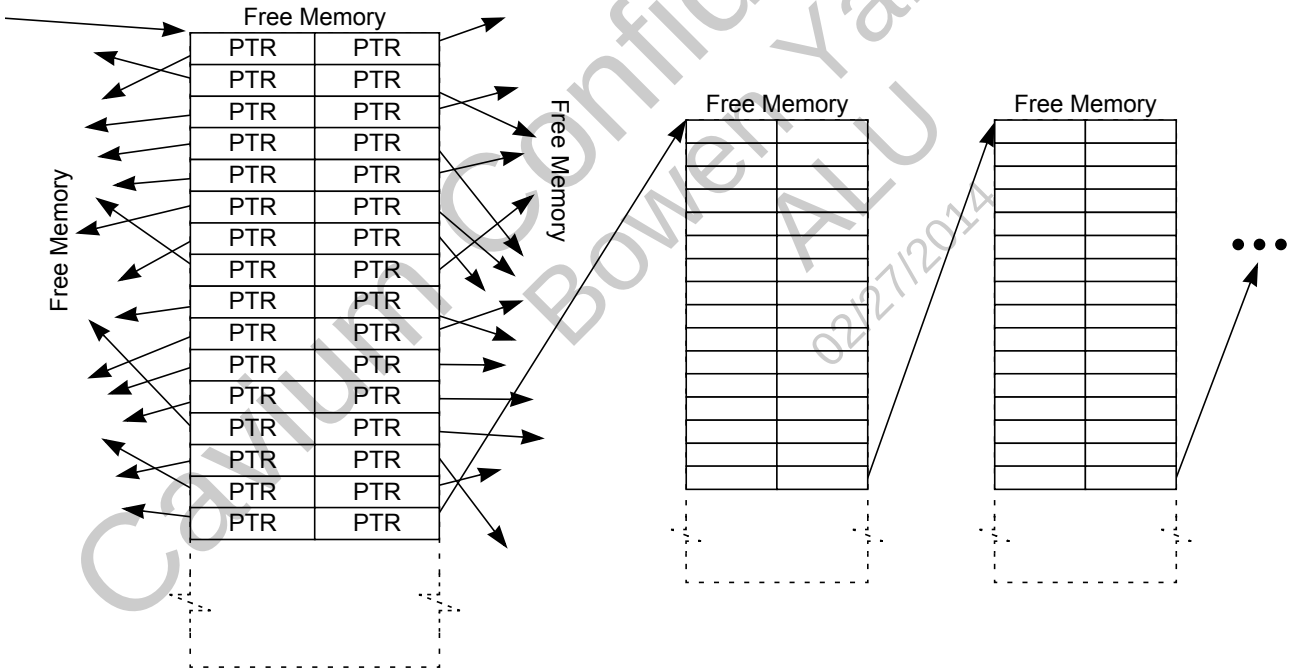
The FPA is a CN71XX unit that maintains eight infinite-size pools of pointers to free L2/DRAM memory. Both core software and other CN71XX hardware units allocate and free pointers from/to the pools. Core software and the centralized input-packet-processing hardware (see Chapter 9) units allocate memory from the pools. Core software, the CN71XX centralized output packet processing (see Chapter 10), DMA Packet Interface (DPI) (see Chapter 12), Hyper Finite Automata (HFA) (see Chapter 11), and Timer (TIM) (see Chapter 16) hardware units free memory to the pools.

The FPA hardware implements a data structure that approximates a logical stack/LIFO for each free pointer pool. The FPA hardware unit stores/caches the top of the stacks in the unit at any time. When a pool is too large to fit in the in-unit store, the FPA builds a tree/list data structure in L2/DRAM, using the freed memory in the pool, to store the extra pointers. Each pool's size is unlimited due to this technique.

The only constraint required by the FPA hardware is that pointers submitted to the free pools must be aligned on a 128-byte boundary and the free memory must be 128 bytes or more. The free memory size can be different in different pools and, in fact, can also be different within the same pool.

Figure 8–1 depicts the data structure the FPA hardware builds in memory when the in-unit stores overflow. Thirty-one out of every 32 pieces of free memory are unmodified by the FPA hardware. Their pointers are held in other free memory. One out of every 32 pieces of memory contains 32 pointers. Thirty-one of the 32 pointers point only at available memory, and the last pointer points at both free memory and 32 more pointers.

Figure 8–1 FPA Free Memory Pointers



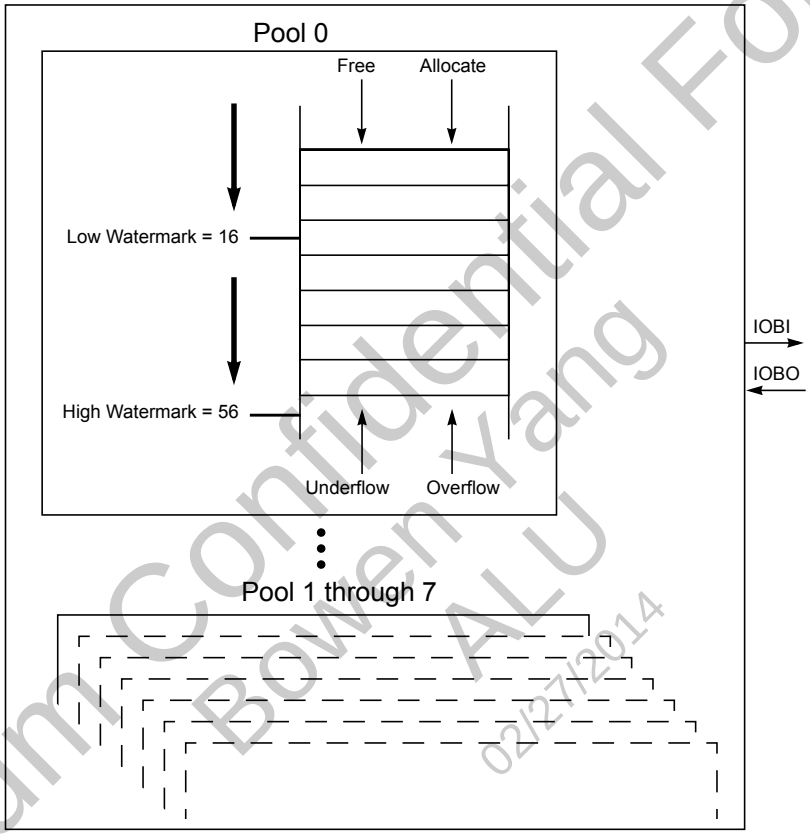
The FPA hardware unit can internally hold up to 2048 pointers, with 256 entries for all pools. (See FPA_FPF*_SIZE).

Cavium Confidential For Bowen Yang ALU 02/27/2014

Each pool has high and low watermarks for its in-unit stores. (See Figure 8–2 and FPA_FPF*_MARKS). CN71XX hardware overflows and underflows the pointers to/from L2/DRAM based on these watermarks. When the pool has more pointers in the unit than the high watermark for the pool allows, CN71XX writes 32 pointers from the pool's in-unit store into L2/DRAM. When the pool has fewer pointers in the unit than the low watermark for the pool allows, CN71XX reads 32 pointers from L2/DRAM into the pool's in-unit store.

CN71XX hardware can also automatically issue "don't-write-back" (DWB) commands on the coherent memory interconnect (CMI) for free memory blocks to ensure that DRAM bandwidth is not unnecessarily wasted writing cache blocks back to memory. This chapter describes operations that include fields to indicate the number of DWBs to execute, but the I/O bridge (IOB) unit implements the DWB functionality (refer to Section 5.3).

Figure 8–2 FPA Free Memory Pointers



Cavium Confidential For Bowen Yang ALU 02/27/2014

When a core requests a pointer from a pool that does not have enough pointers to satisfy the request, the FPA hardware returns a NULL pointer (all 0s) to the core.

The eight available DIDs select the pool for a particular operation. Pool 0 is special since the centralized input packet processing hardware uses storage allocated from it to store packet data. The centralized input packet processing hardware also allocates work queue entries from a programmable pool. Whenever pool 0 or the work queue entry pool are empty, the centralized input packet processing hardware cannot receive packets.

The FPA hardware checks for corruption of the free list in DRAM. These checks can detect ordinary memory corruption, and can also detect pointer duplications in some circumstances. The FPA hardware sets one of FPA_INT_SUM[Q*n*_COFF,Q*n*_PERR,Q*n*_UND] when it detects these errors. In addition to the 32 pointers written to L2/DRAM, CN71XX writes out the page index for the pool (FPA_QUE*n*_PAGE_INDEX) and the pool number. If the values mismatch when it reads pointers back, the FPA hardware sets FPA_INT_SUM[Q*n*_PERR] and loads FPA_QUE_EXP/FPA_QUE_ACT.

Core software can read the FPA_QUE*n*_AVAILABLE[QUE_SIZ] register to find the number of buffers that are currently free in a pool.

Load and IOBDMA operations from the cores allocate memory. (The result is a byte pointer to the free memory.) Stores free memory.

8.1 Free Pool Unit Operations

This section shows the detailed bit formats and codes for the various transactions.

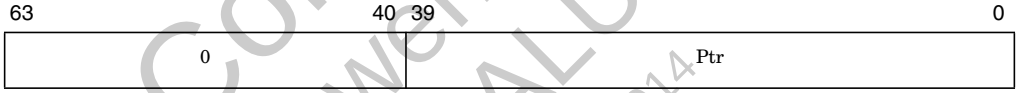
8.1.1 Load Operations

Physical Address for Load

- **DID** - pool number: 0x28 = pool 0, 0x29 = pool 1, ... 0x2F = pool 7.

NOTE: The FPA hardware behavior is unpredictable when anything but a 64-bit load (i.e. anything other than an LD) is used.

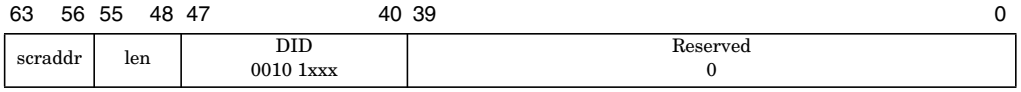
Result for a Load



- **Ptr** - Pointer to free memory. If all zeroes, the selected pool is empty.
  - **Ptr** must be aligned on a 128 byte boundary, thus Ptr <6:0> will always be zero.

8.1.2 IOBDMA Operations

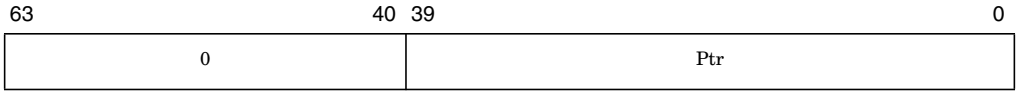
IOBDMA Addressing



- **scraddr** - scratchpad result address, defined in “IOBDMA Operations” on page 84.
- **len** - can legally be from 1 to 16
- **DID** - pool number: 0x28 = pool 0, 0x29 = pool 1, ... 0x2F = pool 7.

Cavium Confidential For Bowen Yang ALU 02/7/2014

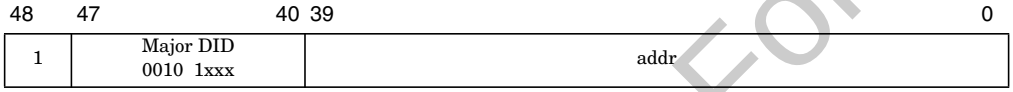
Result for an IOBDMA Request



- **Ptr** - Pointer to free memory. If all zeroes, the selected pool is empty.
  - **Ptr** must be aligned on a 128 byte boundary, thus Ptr <6:0> will always be 0x0.
  - If len is larger than the number of available pointers in the selected pool (i.e. if len > FPA_QUE*n*_AVAILABLE[QUE_SIZ]), then all (len) **Ptr**s returned for the IOBDMA operation are all 0s, indicating that the pool does not have an adequate number of pointers to satisfy the IOBDMA.

8.1.3 Store Operations

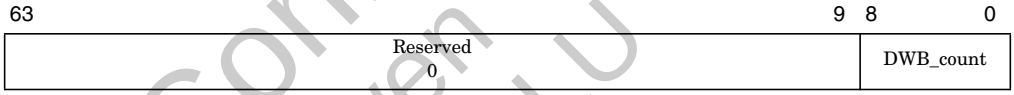
Physical Address to Store to FPA



- **DID** - pool number: 0x28 = pool 0, 0x29 = pool 1, ... 0x2F = pool 7.
- **addr** - pointer to available memory
  - Pointer must be aligned on 128 byte boundary (or unpredictable results)
  - Pointer must point to at least 128 bytes of free memory (or FPA hardware may over-write some useful data with pointers)

NOTE: The FPA hardware behavior is unpredictable when anything but a 64-bit store (i.e. anything other than an SD) is used.

Store Data on a Store to FPA



- The lowest 9 bits contain the **DWB_count**. DWB_count is the number of cache lines for which the hardware (in IOB) should try to execute “don’t-write-back” commands. The hardware starts from the beginning of the free memory (i.e. to where the address points) and marches forward linearly. As the DWB command can modify the value of memory locations, software must ensure that DWB_count × 128 does not exceed the number of available bytes. As the DWB commands consume CMI bandwidth, software should keep the DWB_count low to cover only those cache blocks that may have been modified.

Cavium Confidential For Bowen Yang ALU 09/16/14

8.2 FPA Registers

The FPA CSRs are listed in Table 8–1.

Table 8–1 FPA Registers

Register	Address	CSR Type[1]	Detailed Description
FPA_FPF0_MARKS	0x0001180028000000	RSL	See page 445
FPA_FPF(1..7)_MARKS	0x0001180028000000 + α*0x8	RSL	See page 445
FPA_INT_SUM	0x0001180028000040	RSL	See page 446
FPA_INT_ENB	0x0001180028000048	RSL	See page 448
FPA_CTL_STATUS	0x0001180028000050	RSL	See page 450
FPA_FPF0_SIZE	0x0001180028000058	RSL	See page 450
FPA_FPF(1..7)_SIZE	0x0001180028000058 + α*0x8	RSL	See page 450
FPA_QUE(0..7)_AVAILABLE	0x0001180028000098 + α*0x8	RSL	See page 451
FPA_BIST_STATUS	0x00011800280000E8	RSL	See page 451
FPA_QUE(0..7)_PAGE_INDEX	0x00011800280000F0 + α*0x8	RSL	See page 452
FPA_QUE_EXP	0x0001180028000130	RSL	See page 452
FPA_QUE_ACT	0x0001180028000138	RSL	See page 452
FPA_POOL(0..7)_THRESHOLD	0x0001180028000140 + α*0x8	RSL	See page 452
FPA_POOL(0..7)_START_ADDR	0x0001180028000258 + α*0x8	RSL	See page 453
FPA_POOL(0..7)_END_ADDR	0x0001180028000358 + α*0x8	RSL	See page 453
FPA_ADDR_RANGE_ERROR	0x0001180028000458	RSL	See page 453
FPA_PACKET_THRESHOLD	0x0001180028000460	RSL	See page 453
FPA_WQE_THRESHOLD	0x0001180028000468	RSL	See page 454

1. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

FPA Queue 0 Free Page FIFO Read Write Marks Register FPA_FPF0_MARKS

This is the high- and low-watermark register that determines when we write and read free pages from L2C for queue 0. The value of FPF_RD and FPF_WR should have a difference of at least 33. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:24>	—	RAZ	—	—	Reserved
<23:12>	FPF_WR	R/W	0xC4	0x0	When the number of free-page pointers in a queue exceeds this value, the FPA writes 32-page pointers of that queue to DRAM. The recommended value for this field is FPA_FPF0_SIZE[FPF_SIZ] × .75. The maximum value is FPA_FPF0_SIZE[FPF_SIZ] – 2.
<11:0>	FPF_RD	R/W	0x40	0x0	When the number of free-page pointers in a queue drops below this value and there are free-page pointers in DRAM, the FPA reads one page (32 pointers) from DRAM. The recommended value for this field is FPA_FPF0_SIZE[FPF_SIZ] × .25.The maximum value is FPA_FPF0_SIZE[FPF_SIZ] – 34; the minimum value is 16.

FPA Queue 1–7 Free Page FIFO Read Write Marks Registers FPA_FPF(1..7)_MARKS

This is the high- and low-watermark register that determines when we write and read free pages from L2C for queues 1–7. The value of FPF_RD and FPF_WR should have a difference of at least 33. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:22>	—	RAZ	—	—	Reserved
<21:11>	FPF_WR	R/W	0xC4	0x0	When the number of free-page pointers in a queue exceeds this value, the FPA writes 32-page pointers of that queue to DRAM. The recommended value for this field is FPA_FPF*n*_SIZE[FPF_SIZ] × .75. The maximum value is FPA_FPF*n*_SIZE[FPF_SIZ] – 2.
<10:0>	FPF_RD	R/W	0x40	0x0	When the number of free-page pointers in a queue drops below this value and there are free-page pointers in DRAM, the FPA reads one page (32 pointers) from DRAM. The recommended value for this field is FPA_FPF*n*_SIZE[FPF_SIZ] × .25. The maximum value is FPA_FPF*n*_SIZE[FPF_SIZ] – 34; the minimum value is 16.

Cavium Confidential For Bowen Yang 02/27/2014

FPA Interrupt Summary Register
FPA_INT_SUM

This register contains the different interrupt summary bits of the FPA. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:50>	—	RAZ	—	—	Reserved
<49>	PADDR_E	R/W1C/H	0	0	Set when a pointer address does not fall in the address range for a pool specified by FPA_POOL(0..8)_START_ADDR and FPA_POOL(0..8)_END_ADDR.
<48:44>	—	RAZ	—	—	Reserved
<43>	FREE7	R/W1C/H	0	0	Set when a pointer for POOL7 is freed.
<42>	FREE6	R/W1C/H	0	0	Set when a pointer for POOL6 is freed.
<41>	FREE5	R/W1C/H	0	0	Set when a pointer for POOL5 is freed.
<40>	FREE4	R/W1C/H	0	0	Set when a pointer for POOL4 is freed.
<39>	FREE3	R/W1C/H	0	0	Set when a pointer for POOL3 is freed.
<38>	FREE2	R/W1C/H	0	0	Set when a pointer for POOL2 is freed.
<37>	FREE1	R/W1C/H	0	0	Set when a pointer for POOL1 is freed.
<36>	FREE0	R/W1C/H	0	0	Set when a pointer for POOL0 is freed.
<35>	POOL7TH	R/W1C/H	0	0	Set when FPA_QUE7_AVAILABLE is equal to FPA_POOL7_THRESHOLD[THRESH] and a pointer is allocated or deallocated.
<34>	POOL6TH	R/W1C/H	0	0	Set when FPA_QUE6_AVAILABLE is equal to FPA_POOL6_THRESHOLD[THRESH] and a pointer is allocated or deallocated.
<33>	POOL5TH	R/W1C/H	0	0	Set when FPA_QUE5_AVAILABLE is equal to FPA_POOL5_THRESHOLD[THRESH] and a pointer is allocated or deallocated.
<32>	POOL4TH	R/W1C/H	0	0	Set when FPA_QUE4_AVAILABLE is equal to FPA_POOL4_THRESHOLD[THRESH] and a pointer is allocated or deallocated.
<31>	POOL3TH	R/W1C/H	0	0	Set when FPA_QUE3_AVAILABLE is equal to FPA_POOL3_THRESHOLD[THRESH] and a pointer is allocated or deallocated.
<30>	POOL2TH	R/W1C/H	0	0	Set when FPA_QUE2_AVAILABLE is equal to FPA_POOL2_THRESHOLD[THRESH] and a pointer is allocated or deallocated.
<29>	POOL1TH	R/W1C/H	0	0	Set when FPA_QUE1_AVAILABLE is equal to FPA_POOL1_THRESHOLD[THRESH] and a pointer is allocated or deallocated.
<28>	POOL0TH	R/W1C/H	0	0	Set when FPA_QUE0_AVAILABLE is equal to FPA_POOL0_THRESHOLD[THRESH] and a pointer is allocated or deallocated.
<27>	Q7_PERR	R/W1C/H	0	0	Set when a queue7 pointer read from the stack in the L2C does not have the FPA ownership bit set.
<26>	Q7_COFF	R/W1C/H	0	0	Set when a queue7 stack end tag is present and the count available is greater than pointers present in the FPA.
<25>	Q7_UND	R/W1C/H	0	0	Set when a queue7 page count available goes negative.
<24>	Q6_PERR	R/W1C/H	0	0	Set when a queue6 pointer read from the stack in the L2C does not have the FPA ownership bit set.
<23>	Q6_COFF	R/W1C/H	0	0	Set when a queue6 stack end tag is present and the count available is greater than pointers present in the FPA.
<22>	Q6_UND	R/W1C/H	0	0	Set when a queue6 page count available goes negative.

Cavium Confidential For Review ALL 9/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<21>	Q5_PERR	R/W1C/H	0	0	Set when a queue5 pointer read from the stack in the L2C does not have the FPA ownership bit set.
<20>	Q5_COFF	R/W1C/H	0	0	Set when a queue5 stack end tag is present and the count available is greater than pointers present in the FPA.
<19>	Q5_UND	R/W1C/H	0	0	Set when a queue5 page count available goes negative.
<18>	Q4_PERR	R/W1C/H	0	0	Set when a queue4 pointer read from the stack in the L2C does not have the FPA ownership bit set.
<17>	Q4_COFF	R/W1C/H	0	0	Set when a queue4 stack end tag is present and the count available is greater than pointers present in the FPA.
<16>	Q4_UND	R/W1C/H	0	0	Set when a queue4 page count available goes negative.
<15>	Q3_PERR	R/W1C/H	0	0	Set when a queue3 pointer read from the stack in the L2C does not have the FPA ownership bit set.
<14>	Q3_COFF	R/W1C/H	0	0	Set when a queue3 stack end tag is present and the count available is greater than pointers present in the FPA.
<13>	Q3_UND	R/W1C/H	0	0	Set when a queue3 page count available goes negative.
<12>	Q2_PERR	R/W1C/H	0	0	Set when a queue2 pointer read from the stack in the L2C does not have the FPA ownership bit set.
<11>	Q2_COFF	R/W1C/H	0	0	Set when a queue2 stack end tag is present and the count available is greater than pointers present in the FPA.
<10>	Q2_UND	R/W1C/H	0	0	Set when a queue2 page count available goes negative.
<9>	Q1_PERR	R/W1C/H	0	0	Set when a queue1 pointer read from the stack in the L2C does not have the FPA ownership bit set.
<8>	Q1_COFF	R/W1C/H	0	0	Set when a queue1 stack end tag is present and the count available is greater than pointers present in the FPA.
<7>	Q1_UND	R/W1C/H	0	0	Set when a queue1 page count available goes negative.
<6>	Q0_PERR	R/W1C/H	0	0	Set when a queue0 pointer read from the stack in the L2C does not have the FPA ownership bit set.
<5>	Q0_COFF	R/W1C/H	0	0	Set when a queue0 stack end tag is present and the count available is greater than pointers present in the FPA.
<4>	Q0_UND	R/W1C/H	0	0	Set when a queue0 page count available goes negative.
<3>	FED1_DBE	R/W1C/H	0	0	Set when a single-bit error is detected in FPF1.
<2>	FED1_SBE	R/W1C/H	0	0	Set when a double-bit error is detected in FPF1.
<1>	FED0_DBE	R/W1C/H	0	0	Set when a single-bit error is detected in FPF0.
<0>	FED0_SBE	R/W1C/H	0	0	Set when a double-bit error is detected in FPF0.

Cavium Confidential For Bowen Yang ATU 02/27/2014

FPA Interrupt Enable Register
FPA_INT_ENB

The FPA interrupt-enable register. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:50>	—	RAZ	—	—	Reserved
<49>	PADDR_E	R/W	0	0	When set to 1 and FPA_INT_SUM[PADDR_E] is asserted, the FPA asserts an interrupt.
<48:44>	—	RAZ	—	—	Reserved
<43>	FREE7	R/W	0	0	When set to 1 and FPA_INT_SUM[FREE7] is asserted, the FPA asserts an interrupt.
<42>	FREE6	R/W	0	0	When set to 1 and FPA_INT_SUM[FREE6] is asserted, the FPA asserts an interrupt.
<41>	FREE5	R/W	0	0	When set to 1 and FPA_INT_SUM[FREE5] is asserted, the FPA asserts an interrupt.
<40>	FREE4	R/W	0	0	When set to 1 and FPA_INT_SUM[FREE4] is asserted, the FPA asserts an interrupt.
<39>	FREE3	R/W	0	0	When set to 1 and FPA_INT_SUM[FREE3] is asserted, the FPA asserts an interrupt.
<38>	FREE2	R/W	0	0	When set to 1 and FPA_INT_SUM[FREE2] is asserted, the FPA asserts an interrupt.
<37>	FREE1	R/W	0	0	When set to 1 and FPA_INT_SUM[FREE1] is asserted, the FPA asserts an interrupt.
<36>	FREE0	R/W	0	0	When set to 1 and FPA_INT_SUM[FREE0] is asserted, the FPA asserts an interrupt.
<35>	POOL7TH	R/W	0	0	When set to 1 and FPA_INT_SUM[POOL7TH] is asserted, the FPA asserts an interrupt.
<34>	POOL6TH	R/W	0	0	When set to 1 and FPA_INT_SUM[POOL6TH] is asserted, the FPA asserts an interrupt.
<33>	POOL5TH	R/W	0	0	When set to 1 and FPA_INT_SUM[POOL5TH] is asserted, the FPA asserts an interrupt.
<32>	POOL4TH	R/W	0	0	When set to 1 and FPA_INT_SUM[POOL4TH] is asserted, the FPA asserts an interrupt.
<31>	POOL3TH	R/W	0	0	When set to 1 and FPA_INT_SUM[POOL3TH] is asserted, the FPA asserts an interrupt.
<30>	POOL2TH	R/W	0	0	When set to 1 and FPA_INT_SUM[POOL2TH] is asserted, the FPA asserts an interrupt.
<29>	POOL1TH	R/W	0	0	When set to 1 and FPA_INT_SUM[POOL1TH] is asserted, the FPA asserts an interrupt.
<28>	POOL0TH	R/W	0	0	When set to 1 and FPA_INT_SUM[POOL0TH] is asserted, the FPA asserts an interrupt.
<27>	Q7_PERR	R/W	0	0	When set to 1 and FPA_INT_SUM[Q7_PERR] is asserted, the FPA asserts an interrupt.
<26>	Q7_COFF	R/W	0	0	When set to 1 and FPA_INT_SUM[Q7_COFF] is asserted, the FPA asserts an interrupt.
<25>	Q7_UND	R/W	0	0	When set to 1 and FPA_INT_SUM[Q7_UND] is asserted, the FPA asserts an interrupt.
<24>	Q6_PERR	R/W	0	0	When set to 1 and FPA_INT_SUM[Q6_PERR] is asserted, the FPA asserts an interrupt.
<23>	Q6_COFF	R/W	0	0	When set to 1 and FPA_INT_SUM[Q6_COFF] is asserted, the FPA asserts an interrupt.
<22>	Q6_UND	R/W	0	0	When set to 1 and FPA_INT_SUM[Q6_UND] is asserted, the FPA asserts an interrupt.

Cavium Confidential For Bowen Yang 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<21>	Q5_PERR	R/W	0	0	When set to 1 and FPA_INT_SUM[Q5_PERR] is asserted, the FPA asserts an interrupt.
<20>	Q5_COFF	R/W	0	0	When set to 1 and FPA_INT_SUM[Q5_COFF] is asserted, the FPA asserts an interrupt.
<19>	Q5_UND	R/W	0	0	When set to 1 and FPA_INT_SUM[Q5_UND] is asserted, the FPA asserts an interrupt.
<18>	Q4_PERR	R/W	0	0	When set to 1 and FPA_INT_SUM[Q4_PERR] is asserted, the FPA asserts an interrupt.
<17>	Q4_COFF	R/W	0	0	When set to 1 and FPA_INT_SUM[Q4_COFF] is asserted, the FPA asserts an interrupt.
<16>	Q4_UND	R/W	0	0	When set to 1 and FPA_INT_SUM[Q4_UND] is asserted, the FPA asserts an interrupt.
<15>	Q3_PERR	R/W	0	0	When set to 1 and FPA_INT_SUM[Q3_PERR] is asserted, the FPA asserts an interrupt.
<14>	Q3_COFF	R/W	0	0	When set to 1 and FPA_INT_SUM[Q3_COFF] is asserted, the FPA asserts an interrupt.
<13>	Q3_UND	R/W	0	0	When set to 1 and FPA_INT_SUM[Q3_UND] is asserted, the FPA asserts an interrupt.
<12>	Q2_PERR	R/W	0	0	When set to 1 and FPA_INT_SUM[Q2_PERR] is asserted, the FPA asserts an interrupt.
<11>	Q2_COFF	R/W	0	0	When set to 1 and FPA_INT_SUM[Q2_COFF] is asserted, the FPA asserts an interrupt.
<10>	Q2_UND	R/W	0	0	When set to 1 and FPA_INT_SUM[Q2_UND] is asserted, the FPA asserts an interrupt.
<9>	Q1_PERR	R/W	0	0	When set to 1 and FPA_INT_SUM[Q1_PERR] is asserted, the FPA asserts an interrupt.
<8>	Q1_COFF	R/W	0	0	When set to 1 and FPA_INT_SUM[Q1_COFF] is asserted, the FPA asserts an interrupt.
<7>	Q1_UND	R/W	0	0	When set to 1 and FPA_INT_SUM[Q1_UND] is asserted, the FPA asserts an interrupt.
<6>	Q0_PERR	R/W	0	0	When set to 1 and FPA_INT_SUM[Q0_PERR] is asserted, the FPA asserts an interrupt.
<5>	Q0_COFF	R/W	0	0	When set to 1 and FPA_INT_SUM[Q0_COFF] is asserted, the FPA asserts an interrupt.
<4>	Q0_UND	R/W	0	0	When set to 1 and FPA_INT_SUM[Q0_UND] is asserted, the FPA asserts an interrupt.
<3>	FED1_DBE	R/W	0	0	When set to 1 and FPA_INT_SUM[FED1_DBE] is asserted, the FPA asserts an interrupt.
<2>	FED1_SBE	R/W	0	0	When set to 1 and FPA_INT_SUM[FED1_SBE] is asserted, the FPA asserts an interrupt.
<1>	FED0_DBE	R/W	0	0	When set to 1 and FPA_INT_SUM[FED0_DBE] is asserted, the FPA asserts an interrupt.
<0>	FED0_SBE	R/W	0	0	When set to 1 and FPA_INT_SUM[FED0_SBE] is asserted, the FPA asserts an interrupt.

Cavium Confidential For Ruilen Yang ALD 02/7/2014

FPA Control and Status Register
FPA_CTL_STATUS

This register provides FPA control and status information. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:21>	—	RAZ	—	—	Reserved
<20>	FREE_EN	R/W	0	0	Enables the setting of the INT_SUM_[FREE*] bits.
<19>	RET_OFF	R/W	0	0	When set, IOI devices returning pointers are stalled.
<18>	REQ_OFF	R/W	0	0	When set, IOI devices requesting pointers are stalled.
<17>	RESET	R/W	0	0	When set, causes a reset of the FPA with the exception of the RSL.
<16>	USE_LDT	R/W	0	0	When cleared to 0, the FPA uses LDT to load pointers from the L2C.
<15>	USE_STT	R/W	0	0	When cleared to 0, the FPA uses STT to store pointers to the L2C.
<14>	ENB	R/W	0	0	Enable. Must be set to 1 after writing all configuration registers and 10 cycles have passed. If any of the configuration registers are written after writing this bit, the FPA may begin to operate incorrectly.
<13:7>	MEM1_ERR	R/W	0x0	0x0	Causes a flip of the ECC bit associated [38:32] respective to bit [6:0] of this field, for FPF FIFO1.
<6:0>	MEM0_ERR	R/W	0x0	0x0	Causes a flip of the ECC bit associated [38:32] respective to bit [6:0] of this field, for FPF FIFO0.

FPA Queue 0 Free Page FIFO Size Register
FPA_FPF0_SIZE

This register specifies the number of page pointers that are kept local to the FPA for this queue. FPA queues are assigned in order from queue0 to queue7, though only queue0 through queue*x* can be used. The sum of the eight (0–7) FPA_FPF*n*_SIZE registers must be limited to 2048 bytes. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved
<11:0>	FPF_SIZ	R/W/H	0x100	0x0	The number of entries assigned in the FPA FIFO (used to hold page-pointers) for this queue. The value of this register must divisible by 2, and the FPA ignores bit[0] of this register. The total of the FPF_SIZ fields of the eight FPA_FPF*n*_SIZE registers must not exceed 2048 entries. After writing this field, the FPA needs 10 coprocessor-clock cycles to be ready for operation. The location assignment in the FPA FIFO must start with queue 0, then 1, 2, etc. The number of usable entries is FPF_SIZ – 2.

Cavium Confidential For Chen Yang 02/01/2012

FPA Queue 1–7 Free Page FIFO Size Registers
FPA_FPF(1..7)_SIZE

These registers specify the number of page pointers that are kept local to the FPA for this queue. FPA queues are assigned in order from queue0 to queue7, though only queue0 through queue*x* can be used. The sum of the eight (0–7) FPA_FPF*n*_SIZE registers must be limited to 2048 bytes. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:11>	—	RAZ	—	—	Reserved
<10:0>	FPF_SIZ	R/W/H	0x100	0x0	The number of entries assigned in the FPA FIFO (used to hold page-pointers) for this queue. The value of this register must divisible by 2, and the FPA ignores bit[0] of this register. The total of the FPF_SIZ fields of the eight FPA_FPF*n*_SIZE registers must not exceed 2048 entries. After writing this field, the FPA needs 10 coprocessor-clock cycles to be ready for operation. The location assignment in the FPA FIFO must start with queue 0, then 1, 2, etc. The number of usable entries is FPF_SIZ – 2.

FPA Queue 0–7 Free Page Available Registers
FPA_QUE(0..7)_AVAILABLE

These registers specify the number of page pointers that are available in the FPA and local DRAM. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:29>	—	RAZ	—	—	Reserved
<28:0>	QUE_SIZ	RO/H	0x0	0x0	The number of free pages available in this queue.

BIST Status of FPA Memories Register
FPA_BIST_STATUS

This register provides the result of the BIST run on the FPA memories. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved
<4>	FRD	RO	0	0	FPA_FRD memory BIST status
<3>	FPF0	RO	0	0	FPA_FPF0 memory BIST status
<2>	FPF1	RO	0	0	FPA_FPF1 memory BIST status
<1>	FFR	RO	0	0	FPA_FFR memory BIST status
<0>	FDR	RO	0	0	FPA_FDR memory BIST status

Cavium Confidential For Bowen Yang ALU 3/27/2014

FPA Queue 0–7 Page Index Registers
FPA_QUE(0..7)_PAGE_INDEX

These registers provide the present index page for each of queues 0–7. Each PG_NUM field reflects the number of pages of pointers that have been written to memory for the respective queue. See Table 8–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved
<24:0>	PG_NUM	RO/H	0x0	—	Page number

FPA Queue Expected Value Register
FPA_QUE_EXP

When an FPA_INT_SUM[Q*n*_PERR] occurs, this register is latched with the expected value. Once this register is latched with the first error, it is not latched again until all errors have been cleared. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:29>	—	RO/H	0x0	0x7	Reserved
<28:26>	EXP_QUE	RO/H	0x0	—	Expected FPA queue number read from memory.
<25:0>	EXP_INDX	RO/H	0x0	—	Expected page number read from memory.

FPA Queue Actual Value Register
FPA_QUE_ACT

When an FPA_INT_SUM[Q*n*_PERR] occurs, this register is latched with the actual value read from L2C. Once this register is latched with the first error, it is not latched again until all errors have been cleared. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:29>	—	RO/H	0x0	0x7	Reserved
<28:26>	ACT_QUE	RO/H	0x0	—	FPA queue number read from memory.
<25:0>	ACT_INDX	RO/H	0x0	—	Page number read from memory.

FPA Pool 0–7 Threshold Register
FPA_POOL(0..7)_THRESHOLD

When the value of FPA_QUE(0..7)_AVAILABLE is equal to FPA_POOL(0..7)_THRESHOLD[THRESH] when a pointer is allocated or deallocated, set interrupt FPA_INT_SUM[POOL*TH]. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:29>	—	RAZ	—	—	Reserved
<28:0>	THRESH	R/W/H	0x1FFFFFFF	0x0	Threshold

Cavium Confidential For Bowen Kang ALU 01/27/2014

FPA Pool 0–7 Starting Address Registers
FPA_POOL(0..7)_START_ADDR

Pointers sent to this pool must be equal to or less than this address. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:33>	—	RAZ	—	—	Reserved
<32:0>	ADDR	R/W	0x0	—	Address<39:7>.

FPA Pool 0–7 Ending Address Registers
FPA_POOL(0..7)_END_ADDR

Pointers sent to this pool must be equal to or less than this address. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:33>	—	RAZ	—	—	Reserved
<32:0>	ADDR	R/W	0x1FFFFFFFF	—	Address<39:7>.

FPA Pool Address Range Error Information Register
FPA_ADDR_RANGE_ERROR

When an address is sent to a pool that does not fall in the start and end address specified by FPA_POOL(0..7)_START_ADDR and FPA_POOL(0..7)_END_ADDR, the information related to the failure is captured here. In addition, FPA_INT_SUM[PADDR_E] is set and this register is not updated again until FPA_INT_SUM[PADDR_E] is cleared. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:38>	—	RAZ	—	—	Reserved.
<37:33>	POOL	RO/H	0x0	—	Pool that address was sent to.
<32:0>	ADDR	RO/H	0x0	—	Failing address.

FPA Packet Threshold Register
FPA_PACKET_THRESHOLD

When the value of FPA_QUE0_AVAILABLE[QUE_SIZ] is less than the value of this register, a low pool count signal is sent to the PCIe packet instruction engine (to make it stop reading instructions) and to the Packet-Arbiter informing it to not give grants to packets MAC with the exception of the PCIe MAC. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved.
<31:0>	THRESH	R/W/H	0x0	0x0	Packet threshold.

Cavium Confidential For Bowen Yang ALU 02/27/2014

FPA WQE Threshold Register
FPA_WQE_THRESHOLD

When the value of FPA_QUE(0..7)_AVAILABLE[QUE_SIZ] (which queue number is determined by the value of IPD_WQE_FPA_QUEUE) is less than the value of this register, a low pool count signal is sent to the PCIe packet instruction engine (to make it stop reading instructions) and to the Packet-Arbiter informing it to not give grants to packets MAC with the exception of the PCIe MAC. See Table 8–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved.
<31:0>	THRESH	R/W/H	0x0	0x0	Work-queue entry threshold.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Chapter 9

Packet Input Processing/Input Packet Data Unit (PIP/IPD)

This chapter contains the following subjects:

- Overview
- Input Ports
- Input Packet Formats and Pre-IP Parsing
- Packet Buffering
- Packet Scheduling
- Work-Queue Entry
- Input Packet Data Unit (IPD) Quality of Service
- PIP/IPD Per-QOS Admission Control
- Per-Port and QOS Threshold Interrupts
- PIP Registers
- IPD Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

This chapter discusses the CN71XX centralized packet input processing and input packet data unit (PIP/IPD). It receives packet input data from any/all of the SGMII/1000BASE-X, TBD, or DPI interfaces or from the PKO loop-back port. It can have a combined total of up to TBD input ports for receiving packets between all of these sources. The packets arriving on the different ports share the same PIP/IPD hardware resources, but logically the PIP/IPD hardware treats the different in-flight packets independently.

The PIP/IPD units allocate and write packet data into buffers in a format that is convenient to higher-layer software. The unit supports a programmable buffer size (with pads at the top and bottom for software use), and can distribute packet data across multiple buffers to support large packet input sizes.

The PIP/IPD also creates and allocates (if necessary) a work-queue entry for each packet. This work-queue entry contains a pointer to the buffered packet, hardware parsing results, and packet error checks. The unit performs many L2–L4 checks, including the TCP/UDP checksum check. The unit can skip over a programmable amount of user-defined input data before parsing the input packet. This is useful for user-defined headers passed with the packets. The unit also implements packet-instruction headers, which allow packet scheduling and decode information to be more closely controlled, and has highly configurable automatic tuple and/or mask packet tag generation. The unit supports both Broadcom HiGig* and Marvell DSA packet formats.

9.1 Input Ports

The PIP/IPD hardware accepts packets from input ports numbered as follows:

- PIP/IPD Ports 0–3 = SGMII/QSGMII/RXAUI packet interface 0 ports 0–3
- PIP/IPD Ports 16–19 = SGMII/QSGMII packet interface 1 ports 0–3
- PIP/IPD Port 24 = RGMII packet
- PIP/IPD Ports 32–35 = DPI interface ports 0–3
- PIP/IPD Ports 36–39 = internal loopback ports 0–3

The PIP/IPD hardware treats all the packet-interface ports nearly identically, though each can be configured differently. The exceptions are that there is no CRC removal on the DPI port, and that packet-instruction headers are handled specially on DPI input ports.

Packet interface 0 is either in RXAUI or SGMII/1000BASE-X modes. In the RXAUI case, only port 0 (i.e. PIP/IPD port 0) is ever used, and ports 1–3 (i.e. PIP/IPD ports 1–3) are not used. Packet interface 1 is always in SGMII/1000BASE-X mode and uses ports 16–19. Chapters 23 (SGMII/QSGMII), 21 (RXAUI), and 25 (RGMII) describe the CN71XX packet interfaces.

Section 12.1 describes the CN71XX DPI input ports. Note that each DPI packet input port has eight instruction input rings attached, and there are 32 total rings.

The CN71XX loopback ports provide a loopback path from L2/DRAM, through PKO, into PIP/IPD, and on to L2/DRAM and POW, like packets that arrive via the packet interfaces and DPI input ports. PIP/IPD ports 36–39 correspond to PKO output ports of the same number. Chapter 10 describes PKO loopback.

Cavium Confidential For Bowen Yang ALU 1/27/2014

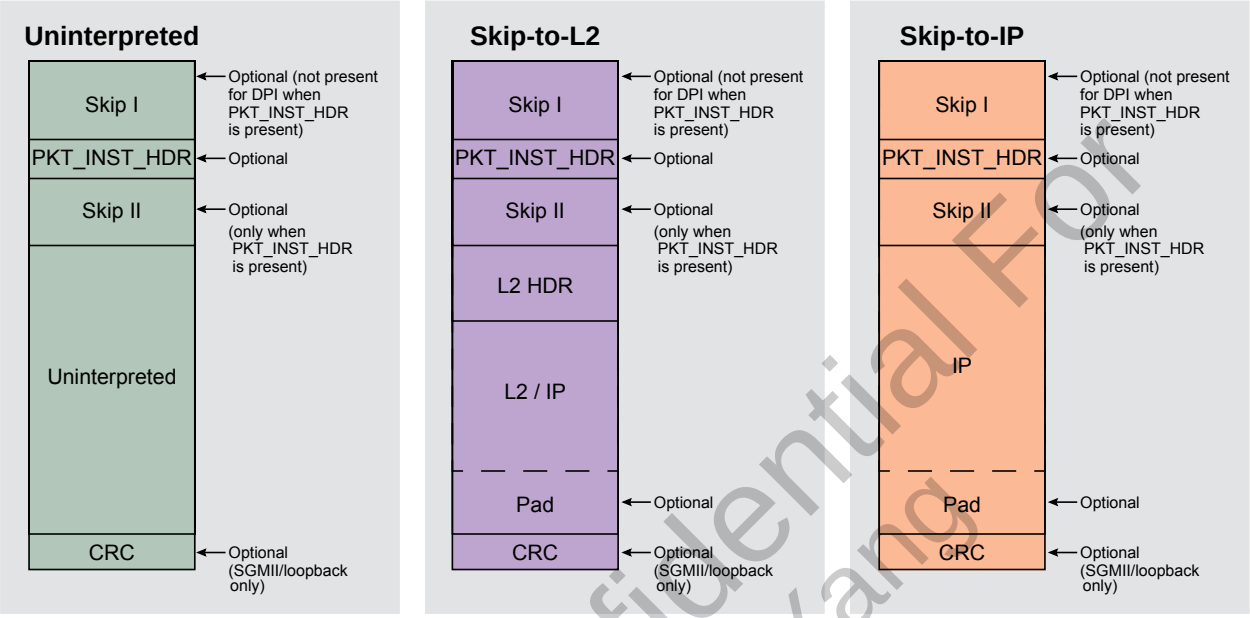
9.2 Input Packet Formats and Pre-IP Parsing

PIP/IDP supports the three different packet-input parse modes.

- uninterpreted
- skip-to-L2
- skip-to-IP

Figure 9–1 shows the packet formats supported by each parse mode.

Figure 9–1 Packet Input Format Modes



When a packet does not have an instruction header, the PIP/IPD hardware creates the packet scheduling and packet-decode information based solely on the packet data and configuration. The optional packet instruction header (PKT_INST_HDR) allows an external device to more directly control the packet scheduling and decoding on a packet-by-packet basis (via RAWFULL and RAWSCH packets, defined in Section 9.2.1).

With the skip fields (and pad field), an application can attach some information in addition to the normal packet. The skip-to-L2 mode parses various ethernet-like L2 header formats and can determine whether IP is present in the packet. The skip-to-IP mode directs PIP/IPD to directly parse the contained IP packet.

Much PIP/IPD parsing is disabled for uninterpreted packets. (A raw hardware checksum is still generated, the packet length is still checked against the skip amount, and CRC may still be discarded.)

The Skip I field may be present in all cases, except for DPI ports when a PKT_INST_HDR is present. Skip is not allowed prior to a PKT_INST_HDR when a packet arrives via a DPI input port. The number of Skip I bytes is configured separately for each port that a packet arrives on and is nominally a fixed value for the port (PIP_PRT_CFG*n*[SKIP]). Alternatively, the number of Skip I bytes in packets on the port can vary when PIP_PRT_CFGB*n*[ALT_SKP_EN]=1. With this, one or two bits at arbitrary locations in the first 64 bits of the packet can select from

Cavium Confidential For Bowen Yang ALU 02/27/2014

up to 4 skip lengths (PIP_PRT_CFG*n*[SKIP] or PIP_ALT_SKIP_CFG*[SKIP1,SKIP2,SKIP3]) for the Skip I byte length. It can be any byte amount from 0 (where it is not present) to its maximum.

The Skip II field can only be present when a PKT_INST_HDR is present. The Skip II field can be any byte amount from 0 (where it is not present) up to its maximum.

The total SKIP is the total number of bytes in the Skip I, PKT_INST_HDR, and Skip II fields shown in Figure 9–1. The maximum allowed SKIP is discussed in Section 9.2.8.

The optional pad field is only relevant to the PIP/IPD hardware when the packet is L2 or IP. For example, a 40-byte IP packet coming in via the packet interface may have been padded out to the minimum defined packet size of 64 bytes. If any input packet contains a pad beyond the end of the IP packet, the PIP/IPD hardware receives the pad and buffers it like all other packet data. It does not get stripped, so the pad is eventually passed on to core software with the rest of the packet.

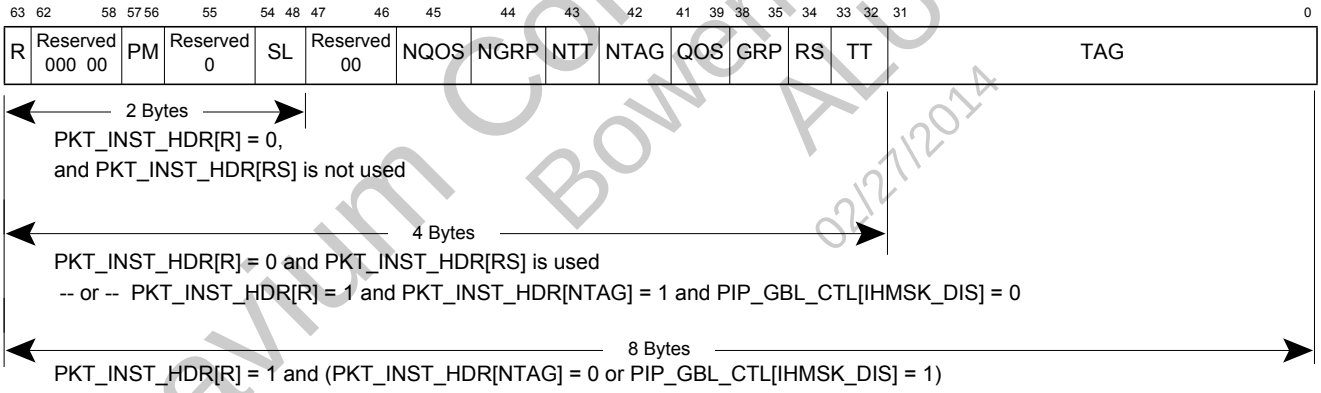
The optional CRC/FCS field protects the packet from transmission errors. Section 9.2.6 covers CRC in more detail.

9.2.1 Packet Instruction Header

The packet instruction header contains information to control packet scheduling and decode. Figure 9–2 shows the packet instruction header that is optionally included at the beginning of the packet. As shown, the full PKT_INST_HDR is eight bytes. All eight bytes are not required in all cases, however.

The packet-instruction-header size for a given packet is effectively selected by PKT_INST_HDR[SL], as the value in PKT_INST_HDR[SL] includes both the number of PKT_INST_HDR bytes and Skip II bytes. Refer to the PKT_INST_HDR[SL] description on page 460.

Figure 9–2 Packet Instruction Header Format (PKT_INST_HDR)



The packet instruction header format (PKT_INST_HDR) is similar to the DPI instruction header format (DPI_INST_HDR). Section 12.1.5.1 shows the DPI instruction format. (Section 9.2.2 shows the transformation from DPI_INST_HDR to PKT_INST_HDR in some detail.)

The packet instruction header format is also similar to the format of the packet scheduling information in WORD1 of the resultant work-queue entry. Refer to Section 9.5 for more description of the work-queue entry format created on packet arrival.

Cavium Confidential For Bowen Yang ALU 02/27/2014

For the packet-interface and loopback ports (ports 0–3, 16–19, 36–39), the presence or absence of an instruction header is selected on a per-port basis (PIP_PRT_CFG(0..3, 16–19, 36..39)[INST_HDR]). A PKT_INST_HDR must be present for the corresponding port when PIP_PRT_CFG(0, 16–19, 36..39)[INST_HDR] is set.

All DPI instructions (refer to Section 12.1.1) include a DPI_INST_HDR, but the resultant packets that enter via the DPI interface ports (ports 32–35) only include a PKT_INST_HDR in the following cases:

- the DPI instruction selects raw scheduling or decoding (i.e. DPI_INST_HDR[R] is set, or the packet is RAWFULL or RAWSCH), or
- all packets arriving via the DPI port are forced to include PKT_INST_HDRs (i.e. SLI_PKT(0..31)_INSTR_HEADER[USE_IHDR] is set).

When a packet has an instruction header, it may be one of two special forms:

- RAWFULL (PKT_INST_HDR[R] is set and PKT_INST_HDR[PM] is uninterpreted)

  PKT_INST_HDR directly specifies the packet scheduling information, and PIP/IPD do not generate the decode information by decoding a RAWFULL packet. (The decode information comes from a configuration register (PIP_RAW_WORD) in this case.)
- RAWSCH (PKT_INST_HDR[R] is set and PKT_INST_HDR[PM] is skip-to-L2 or skip-to-IP)

  PKT_INST_HDR directly specifies the packet-scheduling information, and PIP/IPD generate the packet decode information from RAWSCH packets.

RAWFULL and RAWSCH packets require either an eight-byte or four-byte PKT_INST_HDR. Other PKT_INST_HDRs may be either two or four bytes, depending on whether PKT_INST_HDR[RS] is used.

The following descriptions refer to Figure 9–2:

PKT_INST_HDR[R] If PKT_INST_HDR[R] = 1 and PKT_INST_HDR[PM] = 0 (i.e. if the packet is a RAWFULL packet), then, PIP/IPD uses PKT_INST_HDR[QOS,GRP,TT,TAG] for scheduling information (WORD 1), depending on PKT_INST_HDR[NQOS,NGRP,NTT,NTAG] and PIP_GBL_CTL[IHMSK_DIS], and PIP_RAW_WORD[WORD] for decode information (WORD 2) in the work-queue entry.

If PKT_INST_HDR[R] = 1 and PKT_INST_HDR[PM] ≠ 0 (i.e. if the packet is a RAWSCH packet), then PIP/IPD uses PKT_INST_HDR[QOS,GRP,TT,TAG] for scheduling information (WORD 1), and parses the packet for decode information (WORD 2) in the work-queue entry.

If PKT_INST_HDR[R] = 1, then PKT_INST_HDRs are eight bytes if PKT_INST_HDR[NTAG] = 0 or if PIP_GBL_CTL[IHMSK_DIS] = 1; else four bytes.

Cavium Confidential For Bowen Yang 5/27/2014

PKT_INST_HDR[PM] The mode used to parse the packet:

0 = uninterpreted
1 = skip-to-L2 mode
2 = skip-to-IP mode
3 = reserved

Whenever PKT_INST_HDRs are present, PKT_INST_HDR[R,PM] overrides the default PIP_PRT_CFG*[PM] parsing selection. See Table 9–1.

Table 9–1 PKT_INST_HDR Parsing Selection Override

PKT_INST_HDR[R]	PKT_INST_HDR[PM]	Comment
1	0	**RAWFULL**
1	≠0	**RAWSCHED** – PKT_INST_HDR[PM] selects parse mode.
0	X	**Ordinary (non-RAW) packet instruction header** – PKT_INST_HDR[PM] selects parse mode.

PKT_INST_HDR[SL] The number of bytes in the PKT_INST_HDR and Skip II fields (refer to Figure 9–1) in the packet.

The Skip II field could have no bytes (i.e. it may not exist), but the PKT_INST_HDR must exist and has minimum size constraints. The following bullets describe the minimum PKT_INST_HDR size and, thus, the minimum SL values for a packet:

- When PKT_INST_HDR[R] = 1, the PKT_INST_HDR must be four bytes or more. SL is eight or more if PKT_INST_HDR[NTAG] = 0 or if PIP_GBL_CTL[IHMSK_DIS] = 1; else four or more
- When PKT_INST_HDR[R] = 0, the PKT_INST_HDR may be as small as two bytes, so SL should be two or more. If PKT_INST_HDR[RS] is needed, SL should be four or more bytes rather than two. (The PKT_INST_HDR[RS] description below explains when SL might be used.)

The total SKIP length (sum of Skip I, PKT_INST_HDR, and Skip II) is discussed in Section 9.2.8.

PKT_INST_HDR[NQOS] Must be 0 when PKT_INST_HDR[R] = 0. When set to 1, NQOS prevents PIP from directly using PKT_INST_HDR[QOS] for the QOS value in the work-queue entry. NQOS may commonly be clear so that PKT_INST_HDR[QOS] is used directly by the PIP hardware. If PIP_GBL_CTL[IHMSK_DIS] = 1, CN71XX does not use NQOS.

PKT_INST_HDR[NGRP] Must be 0 when PKT_INST_HDR[R] = 0. When set to 1, NGRP prevents PIP from directly using PKT_INST_HDR[GRP] for the GRP value in the work-queue entry. NGRP may commonly be clear so that PKT_INST_HDR[GRP] is used directly by the PIP hardware. If PIP_GBL_CTL[IHMSK_DIS] = 1, CN71XX does not use NGRP.

PKT_INST_HDR[NTT] Must be 0 when PKT_INST_HDR[R] = 0. When set to 1, NTT prevents PIP from directly using PKT_INST_HDR[TT] for the TT value in the work-queue entry. NTT may commonly be clear so that PKT_INST_HDR[TT] is used directly by the PIP hardware. If PIP_GBL_CTL[IHMSK_DIS] = 1, CN71XX does not use NTT.

PKT_INST_HDR[NTAG] Must be 0 when PKT_INST_HDR[R] = 0. When set to 1, NTAG prevents PIP from directly using PKT_INST_HDR[TAG] for the TAG value in the work-queue entry. NTAG may commonly be clear so that PKT_INST_HDR[TAG] is used directly by the PIP hardware. If PIP_GBL_CTL[IHMSK_DIS] = 1, CN71XX does not use NTAG.

Cavium Confidential For Beiwen Yang on 4_U 6/17/2014

PKT_INST_HDR[QOS, GRP,TT,TAG] When PKT_INST_HDR[R] = 1 and the corresponding PKT_INST_HDR[NQOS,NGRP,NTT,NTAG] = 0, or PIP_GBL_CTL[IHMSK_DIS] = 1, PKT_INST_HDR[QOS,GRP,TT,TAG] are the selected scheduling parameters that become WORD1[QOS,GRP,TT,TAG] of the work-queue entry. Refer to Section 9.5.

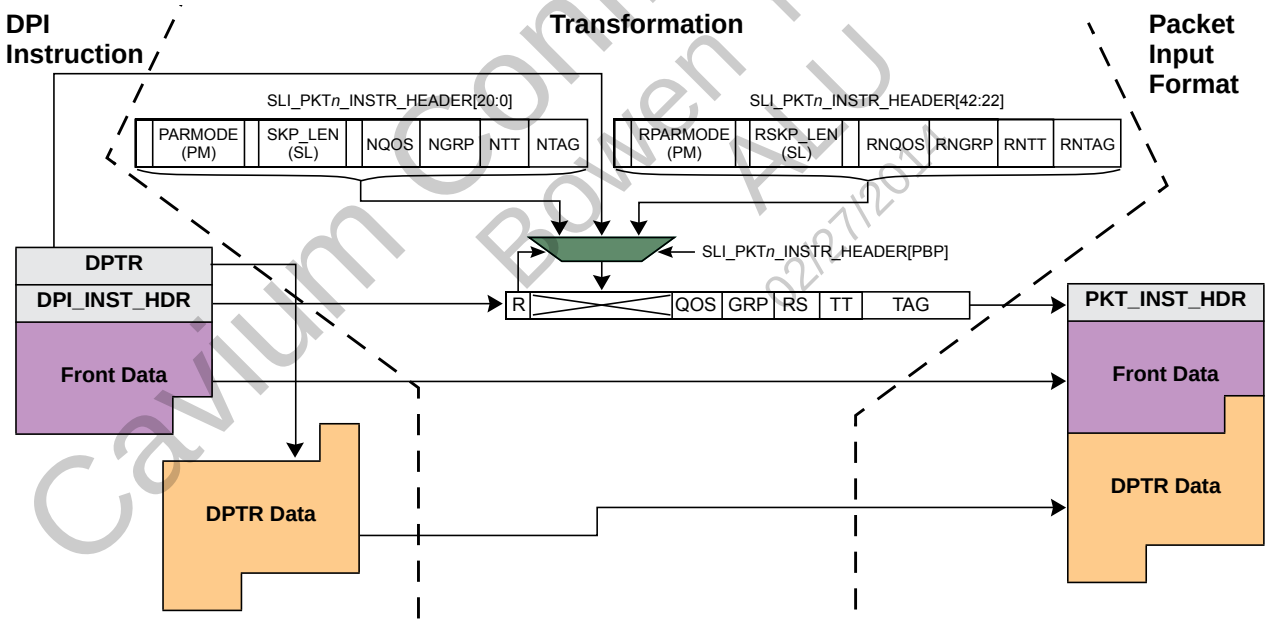
PKT_INST_HDR[RS] When PKT_INST_HDR[RS] = 1, it enables the packet to be buffered solely in the work-queue entry, and not in L2/DRAM. Note the following points about PKT_INST_HDR[RS]:

- PKT_INST_HDR[RS] is not used if any of the following are true:
  - PIP_PRT_CFG*n*[DYN_RS] = 1 for the given port, or
  - PIP_GBL_CTL[IGNRS] = 1 and the port is not a DPI port, or
  - the packet is not dynamic short. (i.e. It doesn't fit entirely in the work-queue entry. Refer to Section 9.5 for more details regarding dynamic-short packets.)
- If PIP_GBL_CTL[IGNRS] = 1 and the port is not a DPI port, PKT_INST_HDR[RS] is effectively forced to 0.
- If PIP_PRT_CFG*n*[DYN_RS] = 1, the packet is always enabled to be buffered solely in the work-queue entry, regardless of the PKT_INST_HDR[RS] value.

9.2.2 DPI Instruction-to-Packet Conversion

Figure 9–3 shows the hardware translation from a DPI instruction into a packet, including the creation of a PKT_INST_HDR from a DPI_INST_HDR. Whenever a PKT_INST_HDR is prepended to a DPI input packet, the packet size and buffering are correspondingly increased to include the PKT_INST_HDR. A packet instruction header is present on a DPI packet when DPI_INST_HDR[R] is set, or when SLI_PKT(0..31)_INSTR_HEADER[USE_IHDR] is set for the ring the packet arrived on.

Figure 9–3 DPI Instruction to Packet Transformation



The PKT_INST_HDR comes directly from the DPI_INST_HDR in the DPI instruction, with the exception of bits <62:42>. PKT_INST_HDR[PM, SL, NQOS, NGRP, NTT, NTAG] are the only bits within <62:42> that are not reserved.

Cavium Confidential For Bowen Yang ALU 02/27/2015

When SLI_PKT(0..31)_INSTR_HEADER[PBP] is set:

- SLI_PKT*n*_INSTR_HEADER[USE_IHDR] must be set so that DPI always creates a PKT_INST_HDR.
- PKT_INST_HDR[PM, SL] come from the individual DPI instruction.
- When DPI_INST_HDR[R] is set, PKT_INST_HDR[NQOS, NGRP, NTT, NTAG] come from SLI_PKT*n*_INSTR_HEADER[RNQOS, RNGRP, RNTT, RNTAG].
- When DPI_INST_HDR[R] is clear, PKT_INST_HDR[NQOS, NGRP, NTT, NTAG] come from SLI_PKT*n*_INSTR_HEADER[NQOS, NGRP, NTT, NTAG].

When SLI_PKT*n*_INSTR_HEADER[PBP] is clear:

- When DPI_INST_HDR[R] is set, PKT_INST_HDR[PM, SL, NQOS, NGRP, NTT, NTAG] come from SLI_PKT*n*_INSTR_HEADER[RPARMODE, RSKP_LEN, RNQOS, RNGRP, RNTT, RNTAG].
- When DPI_INST_HDR[R] is clear and SLI_PKT*n*_INSTR_HEADER[USE_IHDR] is set for the port, PKT_INST_HDR[PM, SL, NQOS, NGRP, NTT, NTAG] come from SLI_PKT*n*_INSTR_HEADER[PAR_MODE, SKP_LEN, NQOS, NGRP, NTT, NTAG].

Note that a PKT_INST_HDR should not be present when a port is in DSA mode (i.e. when PIP_PRT_CFG*n*[DSA_EN] = 1). This implies that both DPI_INST_HDR[R] and the corresponding SLI_PKT*n*_INSTR_HEADER[USE_IHDR] should be clear.

Refer to Section 12.1.1 for more detailed description of a DPI instruction.

9.2.3 Parse Mode and Skip Length Selection

The parse mode and skip lengths are selected by a combination of per-port configuration and packet-by-packet information included in the optional PKT_INST_HDR (see Figure 9–1).

For parse mode:

- When a PKT_INST_HDR is not included in a packet, per-port configuration (i.e. PIP_PRT_CFG*n*[MODE]) selects the parse mode used for the packet.
- When a PKT_INST_HDR is included in a packet, PKT_INST_HDR[R,PM] select the parse mode used for the packet, and determine whether the packet is RAWFULL or RAWSCHED.

For skip lengths:

- Per-port configuration (i.e. PIP_PRT_CFG*n*[SKIP]) selects the number of bytes in the Skip I field, except in the following three cases:
  - When the alternate skip mechanism is enabled for a port (i.e. when PIP_PRT_CFGB*n*[ALT_SKP_EN]=1), up to two bits in the first 64 bits of the received packet can select from amongst four different Skip I lengths. PIP_PRT_CFGB*n*[ALT_SKP_SEL] selects one of the four possible PIP_ALT_SKIP_CFG(0..3) configurations. PIP_ALT_SKIP_CFG*[LEN,BIT1,BIT0] selects where the bits are in the packet. PIP_PRT_CFG*n*[SKIP] and PIP_ALT_SKIP_CFG*[SKIP3,SKIP2,SKIP1] are the four possible Skip I lengths.
  - When a PKT_INST_HDR is present in a packet arriving via a DPI port (the Skip I field is not present and PIP_PRT_CFG*n*[SKIP] is ignored in this case).

Cavium Confidential For Bowen Yang ALU 02/27/2014

– When the port is in DSA mode (i.e. when PIP_PRT_CFG*n*[DSA_EN] = 1). (Neither Skip I, PKT_INST_HDR, nor Skip II should be present in this case, and PIP_PRT_CFG*n*[SKIP] indicates the DSA tag size.)

Note that when the port is in HiGig* mode (i.e. PIP_PRT_CFG*n*[HIGIG_EN] = 1), PIP_PRT_CFG*n*[SKIP] is the Skip I size, and must always equal either 12 or 16.

- Whenever a PKT_INST_HDR is included with a packet, PKT_INST_HDR[SL] indicates the combined number of bytes in both the PKT_INST_HDR and Skip II fields.

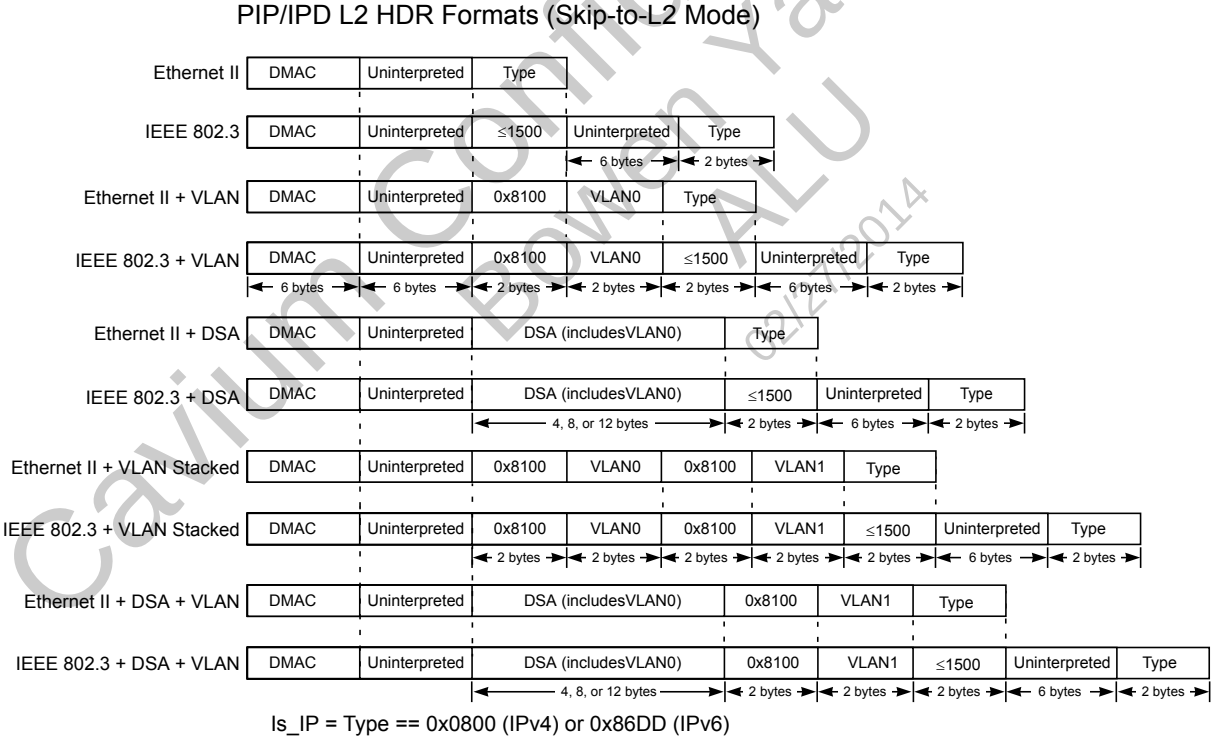
Note that, as described in Section 9.2.2, PKT_INST_HDR[PM,SL] (when present) are derived from per-port configuration rather than packet-by-packet information for the DPI ports (ports 32–35) if SLI_PKT*n*_INSTR_HEADER[PBP] is clear.

9.2.4 PIP/IPD L2 Parsing and Is_IP Determination

In the Skip-to-L2 mode, the PIP/IPD hardware parses the L2 HDR field to determine its length and whether an IP packet follows. Figure 9–4 shows the supported L2 HDR types, which are Ethernet II and IEEE 802.3 with zero, one, or two additional DSA/VLAN fields. Regardless of the classification, the packet is considered to be IP only when the last two bytes of a valid L2 HDR field (i.e. the TYPE field) equals 0x0800 (IPv4) or 0x86DD (IPv6).

In the skip-to-IP case, there is no L2 HDR field, but the PIP/IPD hardware acts much the same as the case when the hardware considers the packet to be IP in the Skip-to-L2 mode. Note that the skip field could be anything, including an L2 header field that does not need to be interpreted by the PIP/IPD hardware.

Figure 9–4 Supported L2 HDR Types in Skip-to-L2 Mode



Cavium Confidential For Bowen Yang ALU 02/27/2014

9.2.5 Pre-IP Parsing Summary

The following pseudo-code describes the Pre-IP parsing (except for the bit-select extractor, which is explained in Section 9.4.2) in detail:

```
// port is the input port the packet arrived on (0,32..39)
if(PIP_PRT_CFG[port][DSA_EN]) { // if port is in DSA mode
  SKIP = 0;
  pkt_inst_hdr_pres = false;
}
else { // port is not in DSA mode - SKIP is skip, look for instruction header
  SKIP = PIP_PRT_CFG[port][SKIP];
  if(PIP_PRT_CFGB[port][ALT_SKP_EN]) {
    sel = PIP_PRT_CFGB[port][ALT_SKP_SEL];
    word = first 64-bits in the packet;
    index = word<PIP_ALT_SKIP_CFG[sel][BIT0]>;
    if(PIP_ALT_SKIP_CFG[sel][LEN])
      index = index | ((word<PIP_ALT_SKIP_CFG[sel][BIT1]>) << 1);
    if(index != 0)
      SKIP = PIP_ALT_SKIP_CFG[sel][SKIP[index]];
  }
  if((port >= 32) && (port <= 35)) { // if a DPI port
    if(PCI_INST_HDR[R] or SLI_PKT(0..31)_INSTR_HEADER[USE_IHDR]) {
      PKT_INST_HDR = PCI_INST_HDR;
      pkt_inst_hdr_pres = true;
      SKIP = 0;
      if(PCI_INST_HDR[R])
        PKT_INST_HDR<62:42> = SLI_PKT[port]_INSTR_HEADER<42:22>; // RPARMODE,RSKIP_LEN
      else
        PKT_INST_HDR<62:42> = SLI_PKT[port]_INSTR_HEADER<20:0>; // PAR_MODE, SKIP_LEN
      if(SLI_PKT[port]_INSTR_HEADER[PBP])
        PKT_INST_HDR[PM,SL] = from individual DPI instruction;
      // CN71XX DPI logic includes PKT_INST_HDR as first 8 bytes
    }
    else {
      skip SKIP bytes;
      pkt_inst_hdr_pres = false;
    }
  }
  else { // if a SGMII, 1000Base-X, or loopback port
    if(PIP_PRT_CFG[port][HIGIG_EN]) {
      if(SKIP == 12) { // HG/HG+ - 12B
        skip 1 byte;
        BYTE = next byte;
        HG_PRI<5> = 0;
        HG_PRI<4> = BYTE<5>;     // CNG<1> at DW0_1<5>
        skip 5 bytes;
        BYTE = next byte;
        HG_PRI<2:0> = BYTE<2:0>; // CoS/PRIORITY at DW1_2<2:0>
        skip 1 byte;
        BYTE = next byte;
        HG_PRI<3> = BYTE<5>;     // CNG<0> at DW1_3<5>
        skip 5 bytes;
      }
      else { // HG2 - must be 16B
        skip 1 byte;
        BYTE = next byte;
        HG_PRI<3:0> = BYTE<3:0>; // TC at DW0_1<3:0>
        skip 6 bytes;
        BYTE = next byte;
        HG_PRI<5:4> = BYTE<7:6>; // DP at DW1_3<7:6>
        skip 9 bytes;
      }
    }
    else
      skip SKIP bytes;
    if(PIP_PRT_CFG(0,32..39)[INST_HDR]) {
      pkt_inst_hdr_pres = true;
      PKT_INST_HDR = next 8 bytes; // 2, 4, or 8 bytes may be used
    }
    else
      pkt_inst_hdr_pres = false;
  }
}
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

```
if(pkt_inst_hdr_pres) {
  PKT_PARSE_MODE = PKT_INST_HDR[PM];
  SKIPSL = PKT_INST_HDR[SL]; // note that PKT_INST_HDR[SL] should account for
                             // PKT_INST_HDR size (minimum two to eight bytes)
  LAST_SKIP = SKIPSL;
  ISRAWFULL = PKT_INST_HDR[R] && (PKT_PARSE_MODE is uninterpreted);
  ISRAWSCH = PKT_INST_HDR[R] && (PKT_PARSE_MODE is skip-to-L2 or skip-to-IP);
  skip SKIPSL bytes;
  SKIP = SKIP + SKIPSL;
}
else {
  LAST_SKIP = SKIP;
  PKT_PARSE_MODE = PIP_PRT_CFG[port][MODE];
  ISRAWFULL = false;
  ISRAWSCH = false;
}
L2_Size = 0;
VV = VS = false;
VLAN0 = VLAN1 = 0;
ethertype_val = 0;
if(PKT_PARSE_MODE is skip-to-L2) {
  DMAC = next 6 bytes;
  skip 12 bytes;
  ethertype_val = 1;
  L2_Size = L2_Size + 14;
  dsa_tag_present = PIP_PRT_CFG[port][DSA_EN] && (DMAC != 01-80-C2-00-00-01);
  if(dsa_tag_present) { // DSA mode
    VV = true;
    WORD0 = next 4 bytes;
    skip 4 bytes;
    VLAN0 = WORD0<15:0>;
    VLAN0<12> = WORD0<16>; // becomes WORD2[VC] in WQE
    TAG_CMD = WORD0<31:30>;
    SRC_ID = 0;
    if(PIP_PRT_CFG[port][SKIP] > 4) {
      WORD1 = next 4 bytes;
      skip PIP_PRT_CFG[port][SKIP]-4 bytes (i.e. skip either 4 or 8 bytes);
      if(TAG_CMD==0)
        VLAN0<12> = WORD1<30>; // becomes WORD2[VC] in WQE
      SRC_ID = WORD1<24:20>;
    }
    L2_Size = L2_Size + PIP_PRT_CFG[port][SKIP];
    ethertype = next two bytes;
    skip 2 bytes
  }
  else {
    ethertype = next two bytes;
    skip 2 bytes;
    if(ethertype == 0x8100) { // VLAN
      VV = true;
      VLAN0 = next two bytes;
      skip 2 bytes;
      ethertype = next two bytes;
      skip 2 bytes;
      L2_Size = L2_Size + 4;
    }
  }
  if(ethertype == 0x8100) { // STACKED VLAN
    VS = true;
    VLAN1 = next two bytes;
    skip 2 bytes
    ethertype = next two bytes;
    skip 2 bytes;
    L2_Size = L2_Size + 4;
  }
  limit = PIP_GBL_CFG[MAX_L2] ? 1535 : 1500;
  l2_length = ethertype;
  if(l2_length <= limit) {
    skip 6 bytes;
    L2_Size = L2_Size + 8;
    ethertype = next two bytes;
    skip 2 bytes;
  }
  Is_IP = (ethertype == 0x0800) || (ethertype == 0x86DD);
}
else if(PKT_PARSE_MODE == skip-to-IP) {
  Is_IP = true;
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

```
  if (LAST_SKIP >= 8) {
    ethertype = prior two bytes;
    ethertype_val = 1;
  }
}
else
  Is_IP = false;
```

9.2.6 Packet Input CRC

PIP/IPD can remove CRC from a packet, but never checks it. The packet interfaces check CRC of inbound packets. Refer to Chapter 23 and Chapter 24.

Note that the PIP/IPD hardware can never remove CRC for a packet that arrives via the DPI interface (ports 32–35).

For packets arriving by a packet interface (ports 0–3, 16–19), PIP/IPD can optionally either remove or retain the CRC field for later processing. When IPD_SUB_PORT_FCS[PORT_BIT<*port*>] is set for the port a packet arrives on, PIP/IPD removes the CRC before buffering the packet.
IPD_SUB_PORT_FCS[PORT_BIT<*port*>] should only be set when both CRC is present and should be removed.

9.2.7 Packet Length Checks

PIP/IDP is capable of two classes of length checks. The first class of checks (MAXERR, MINERR, and PUNY) ensures that the packet is within a legal packet size. The second class (LENERR) is able to validate the L2 length field when an L2 header is present.

Most of these exceptions can be enabled by setting the appropriate bit in PIP_PRT_CFG*n* for each port in which the checker is desired. If the bit is set, the exception is logged in PIP_INT_REG. In addition, packets sent into CN71XX arrive with a receive error (work-queue entry WORD2[RE] = 1) and the opcode (WORD2[Opcode]) is set to the exception in the work queue entry.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 9–2 lists the length exceptions, their causes, and how they are handled.

Table 9–2 Receive Errors/Exceptions

Exception	Cause	Notification
MAXERR	A packet was received with length > PIP_FRM_LEN_CHK0[MAXLEN] bytes. For tagged frames, MAXLEN increases by the DSA TAG size (PIP_PRT_CFG[SKIP] bytes) or the VLAN TAG size (four bytes) for each DSA/VLAN found up to a maximum of two DSA/VLANs. When PIP_PRT_CFG[DSA_EN] = 0, the max size is MAXLEN + 8 bytes. When PIP_PRT_CFG[DSA_EN] = 1, the max size is MAXLEN+PIP_PRT_CFG[SKIP]+4 bytes. **NOTE:** If GMX0_RX*n*_FRM_CTL[PTP_MODE] is set, the eight-byte timestamp is prepended to the packet. MAXLEN should be increased by 8 to compensate for the additional timestamp field.	PIP_PRT_CFG*n*[MAXERR_EN] enables the check for port *n*. If enabled, PIP_INT_REG[MAXERR] is set to 1 and work-queue entry WORD2[OPCODE] is set to 0x3 (if packet has bad FCS) or 0x4 (if packet has good FCS).
MINERR	A packet was received with length < PIP_FRM_LEN_CHK0/1[MINLEN] bytes or a pause packet was received with length < (GMX0_RX_FRM_CHK[FCSERR] ? 64 : 60). **NOTE:** If GMX0_RX*n*_FRM_CTL[PTP_MODE] is set, the eight-byte timestamp is prepended to the packet. MINLEN should be increased by 8 to compensate for the additional timestamp field.	PIP_PRT_CFG*n*[MINERR_EN] enables the check for port *n*. If enabled, PIP_INT_REG[MINERR] is set to 1 and work-queue entry WORD2[OPCODE] is set to 0x6 (if packet has bad FCS) or 0x8 (if packet has good FCS). GMX0_RX*n*_FRM_CHK[MINERR] enables the pause frame check for port n. If enabled, GMX0_RX_INT_REG[MINERR] is set to 1 and work-queue entry WORD2[OPCODE] is set to 0x6 (if packet has bad FCS) or 0x8 (if packet has good FCS).
LENERR	A packet's received length does not match the length field extracted from the L2 header within the packet. The check is only valid for packets PARSE_MODE = skip-to-L2 with a valid length field (64–1500 bytes). Refer to Figure 9–5 for the length-check algorithm. **NOTE:** If GMX0_RX*n*_FRM_CTL[PTP_MODE] is set, the eight-byte timestamp is prepended to the packet. Normally, PIP_PRT_CFG[SKIP] is increased by 8 to compensate for the additional timestamp field. Since the algorithm already factors in the SKIP, no additional changes are necessary for proper operation.	PIP_PRT_CFG*n*[LENERR_EN] enables the check for port *n*. When PIP_PRT_CFG*n*[VLAN_LEN] = 1, the check is suppressed for DSA/VLAN packets. When PIP_PRT_CFG*n*[PAD_LEN] = 1, the check is disabled for packets with padding in the data. If enabled, PIP_INT_REG[LENERR] is set to 1 and work-queue entry WORD2[OPCODE] is set to 0xA.
PUNY	FCS stripping is enabled (i.e., IPD_SUB_PORT_FCS[PORT_BIT<*port*>] is set for the port that the packet arrived on) and a packet arrived that was four bytes or less.	This check is always on when IPD_SUB_PORT_FCS[PORT_BIT<*port*>] is set for the port that the packet arrived on. PIP/IPD disables FCS strip for the packet, sets PIP_INT_REG[PUNYERR] to one, and sets the work-queue entry WORD2[OPCODE] to 47. The resultant packet indicated by the work-queue entry is always four bytes or less.

Cavium Confidential For Power Xaml ARB 02/27/2014

Figure 9–5 Length-Check Algorithm

```
#define DMAC_SZ 6
#define SMAC_SZ 6
#define LENGTH_SZ 2
#define FCS_SZ 4
#define ETHERNET_HEADER_SIZE (DMAC_SZ+SMAC_SZ+LENGTH_SZ+FCS_SZ)
//
// parameters
//  port       port that the packet arrived on
//  DMAC       48-bit destination MAC address from the ethernet packet
//  l2_length  16 bit field as extracted from the ethernet packet (aka ethertype)
//  pkt_size   total size for the received packet including any FCS bytes
//  SKIP       the sum of Skip I + PKT_INST_HDR + Skip II
//  vv         Set when one or more VLAN/DSA is present in the packet
//  vs         Set when stacked VLAN/DSA is present
// return value
//  false when the HW will compute a LENERR
// notes
//  algorithm assumes that FCS is in the packet. if FCS is not present,
//  disable the check via PIP_PRT_CFG[port][LENERR_EN].
//
bool l2_length_ok(int port, uint48 DMAC, uint16 l2_length, uint16 pkt_size, uint16 SKIP, bool vv,
                  bool vs) {

 if (!PIP_PRT_CFG[port][LENERR_EN]
     || (PKT_PARSE_MODE != skip-to-L2)
     || (PIP_PRT_CFG[port][VLAN_LEN] && vv))
   return true;

 if (l2_length > (PIP_GBL_CFG[MAX_L2] ? 1535 : 1500)) // type, not length, so no check
   return true;

  dsa_tag_present = PIP_PRT_CFG[port][DSA_EN] && (DMAC != 01-80-C2-00-00-01);

  vlan0_size = dsa_tag_present           ? PIP_PRT_CFG[port][SKIP] : 4;
  SKIP       = PIP_PRT_CFG[port][DSA_EN] ? 0 : SKIP; // SKIP must be zero for DSA mode

  uint16 apply_pad_then_vlan = (l2_length < 46) ? 46 : l2_length;
  apply_pad_then_vlan += vlan0_size*vv + 4*vs;
  apply_pad_then_vlan += ETHERNET_HEADER_SIZE + SKIP;

  uint16 apply_vlan_then_pad = l2_length + vlan0_size*vv + 4*vs;
  apply_vlan_then_pad = (apply_vlan_then_pad < 46) ? 46 : apply_vlan_then_pad;
  apply_vlan_then_pad += ETHERNET_HEADER_SIZE + SKIP;

  if (pkt_size < apply_vlan_then_pad)
      return false;

  if (!PIP_PRT_CFG[port][PAD_LEN]
          && (pkt_size != apply_pad_then_vlan)
          && (pkt_size != apply_vlan_then_pad))
      return false;

   return true;
}
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

9.2.8 Legal SKIP Values

The total SKIP (total number of bytes in the Skip I, PKT_INST_HDR, and Skip II fields shown in Figure 9–1) must not be too large.

Note that when DSA mode is enabled (i.e. when PIP_PRT_CFG*n*[DSA_EN] = 1), the total SKIP must be zero, and PIP_PRT_CFG*n*[SKIP] indicates the Marvell DSA tag size (4-byte, 8-byte, or 12-byte), not the Skip I. The remainder of this section assumes that DSA mode is not enabled.

Note also that when HiGig* mode is enabled (i.e. when PIP_PRT_CFG*n*[HIGIG_EN] = 1), the Skip I (i.e. PIP_PRT_CFG*n*[SKIP]) must equal either 0xC or 0x10, the HiGig* header size.

SKIP should always be less than the number of bytes in the packet. Otherwise WORD2[RE] will be set in the work-queue entry and PIP_INT_REG[SKPRUNT] will be set. WORD2[Opcode] will also be 12 or 17 if the packet was not already tagged with a receive error that has a smaller WORD2[Opcode] code. (The packet may commonly already be tagged with a WORD2[Opcode] = 6 or 8 (i.e. MINERR) receive error, so WORD2[Opcode] = 12 or 17 may not be often seen.) The number of bytes in the packet is before the optional CRC strip.

When the maximums specified in Table 9–3 are adhered to, PIP/IPD hardware is fully functional, with no caveats.

Table 9–3 SKIP Maximum Values (No Caveats Needed)

Packet Type	Maximum SKIP (No Caveats Needed)
Uninterpreted, RAWFULL, or RAWSCH	127
Skip-to-L2, Is_IP=0	130 – L2_Size
Skip-to-L2, Is_IP=1, IPv4	104 – L2_Size
Skip-to-L2, Is_IP=1, IPv6	84 – L2_Size
Skip-to-IP, IPv4	104
Skip-to-IP, IPv6	84

PIP/IPD hardware can also function with some caveats at larger SKIP values.

- For example, PIP/IPD hardware supports packets and SKIP values in the ranges specified in Table 9–4 with modest caveats.

Table 9–4 SKIP Value Ranges (QOS L4 Watcher Caveat)

Packet Type	SKIP Range (QOS L4 Watcher Caveat)
Skip-to-L2, Is_IP=1, IPv6	(85 – L2_Size) to (98 – L2_Size)
Skip-to-IP, IPv4	105 to 116
Skip-to-IP, IPv6	85 to 98

CN71XX is fully functional for these IPv4/IPV6 packets in these SKIP ranges when L4 QOS watchers are not enabled.

When L4 QOS watchers are enabled for these packets with a SKIP in this range, the hardware may disable the L4 watchers when it generates the QOS value used for admission control. This means that the QOS that the hardware uses for admission control may differ from the QOS used for the work-queue entry. (Sections 9.4.3 and 9.5 define the QOS value used for the work-queue entry, and Section 9.7 defines the QOS generated for admission control.)

Cavium Confidential For Bowen Yang ALU 5/27/2014

L4 QOS watchers are enabled for a packet when there exists an ***i*** such that PIP_PRT_CFG*n*[QOS_WAT<*i*>] (for the port the packet arrives on) is set, while PIP_QOS_WATCH[*i*][TYPE] is also TCP or UDP.

- For example, PIP/IPD hardware supports packets and SKIP values in the ranges shown in Table 9–5 with larger, but still modest caveats.

Table 9–5 SKIP Value Ranges (QOS DiffServ/Watcher Caveat)

Packet Type	SKIP Range (QOS DiffServ/Watcher Caveat)
Skip-to-L2, Is_IP=1, IPv6	(99 – L2_Size) to (104 – L2_Size)
Skip-to-IP, IPv6	99 to 112

CN71XX is fully functional for these IPv6 packets in these SKIP ranges when neither QOS watchers nor QOS DiffServ are enabled.

When either is enabled for these IPv6 packets in these SKIP ranges, the hardware may disable QOS watchers and/or DiffServ when it generates the QOS value used for admission control, so the QOS used for admission control may differ from the QOS used for the work-queue entry.

QOS watchers are enabled for a packet when PIP_PRT_CFG*n*[QOS_WAT] (for the port the packet arrives on) is not 0, and QOS DiffServ is enabled for a packet when PIP_PRT_CFG*n*[QOS_DIFF] (for the port the packet arrives on) is set.

- Generally speaking, the PIP/IPD hardware does not support SKIP values larger than the values specified in Table 9–6.

Table 9–6 SKIP Maximum Values (All Caveats)

Packet Type	Maximum SKIP (All Caveats)
Uninterpreted, RAWFULL, or RAWSCH	127
Skip-to-L2, Is_IP=0	130 – L2_Size
Skip-to-L2, Is_IP=1, IPv4	104 – L2_Size
Skip-to-L2, Is_IP=1, IPv6	104 – L2_Size
Skip-to-IP, IPv4	116
Skip-to-IP, IPv6	112

9.3 Packet Buffering

PIP/IPD normally writes a copy of the entire packet into L2/DRAM buffers, as well as a portion or all the packet (depending on the packet size and configuration) into WORD4–WORD15 of the work-queue entry.

There are several reasons why a packet may not be buffered in ordinary L2/DRAM buffers in CN71XX:

- All bytes of some short packets can completely fit in WORD4–WORD15 of the work-queue entry for some configurations, so the ordinary L2/DRAM packet buffering is redundant and avoided. Refer to Section 9.5 for more details on this dynamic short case and WORD4–WORD15 the work-queue entry.
- Packets may selectively be dropped by PIP/IPD. Refer to Section 9.7. In this case, the packet is buffered neither in L2/DRAM, nor in the work-queue entry.
- Packets may be dropped due to buffer overflow outside of PIP/IPD. In this case also, the packet is buffered neither in L2/DRAM nor in the work-queue entry. Note that these external-buffer overflows can also result in partial packets (i.e. WORD2[RE] is set and WORD2[OPCODE] is set to 1 (partial error) in this case. See Section 9.5.), which are buffered in L2/DRAM.

Cavium Confidential For Bowen Yang ALU 04/27/2014

The remainder of this section assumes that the packet is buffered in L2/DRAM.

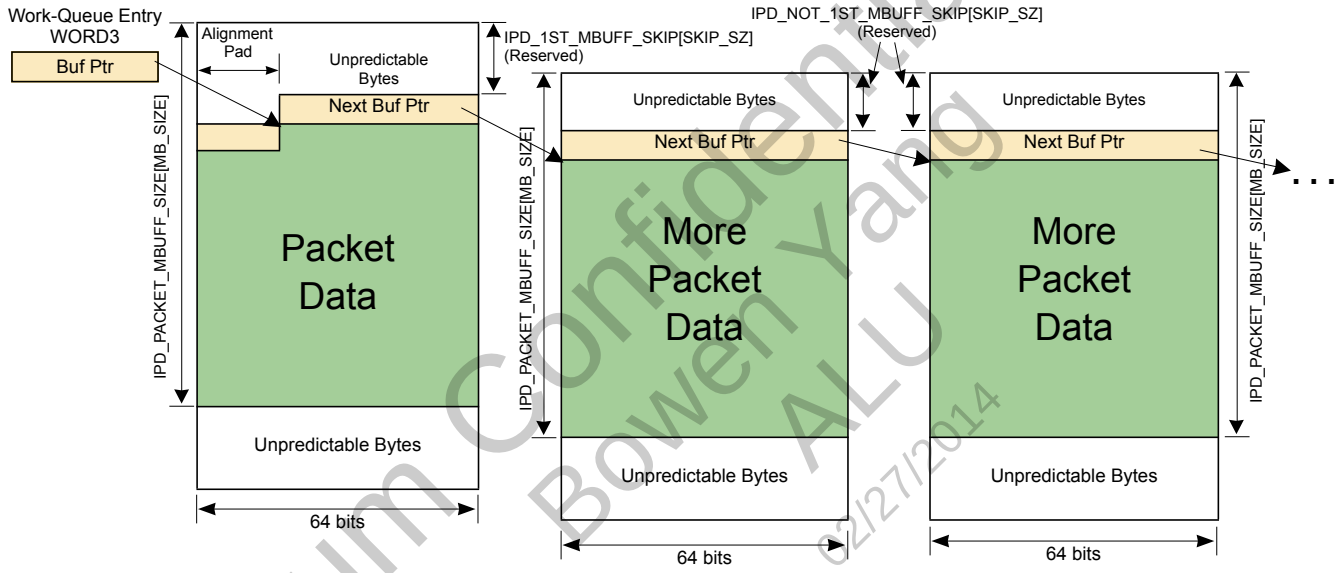
The PIP/IPD hardware both allocates buffers for a packet and writes packet data into the buffers allocated for a packet. PIP/IPD gets the packet buffers it needs from free pool 0, which is a hardware free pool managed by the FPA hardware unit (see Chapter 8.) PIP/IPD prequeues a number of pointers to packet buffers that it receives from FPA. The number of packet-buffer pointers currently held by IPD is

IPD_PTR_COUNT[PKT_PCNT] + IPD_PTR_COUNT[PFIF_CNT] +
IPD_PTR_COUNT[PKTV_CNT] + IPD_PRC_PORT_PTR_FIFO_CTL[MAX_PKT] +
(IPD_CTL_STATUS[NO_WPTR] × IPD_PTR_COUNT[WQEV_CNT])

IPD holds no pointers if it has not yet been enabled.

Figure 9–6 depicts the way that the PIP/IPD hardware writes a large packet into L2/DRAM. This example is a case where the packet is too large to fit in a single buffer (in fact, it is too large to fit into three buffers), so the PIP/IPD hardware must allocate multiple buffers, write the packet data into them, and link them together with pointers. Ultimately, the PIP/IPD hardware places a pointer to the first buffer holding the packet data in the work-queue entry that it creates to signal the arrival of a packet. This work-queue entry also indicates the number of buffers that were used (WORD2[Bufs]) and the total packet length in bytes (Len). Refer to Section 9.5 for more description of the work-queue entry and WORD2.

Figure 9–6 PIP/IPD Hardware Allocating Multiple Buffers



The PIP/IPD stores packets into buffers contiguously, but leaves space at the top and bottom of a buffer if desired. Software can configure the number of 64-bit words at the front of a buffer that are guaranteed not to contain packet data. There are two configuration values for this reserved distance at the top of a buffer:

- one for the first buffer (IPD_1ST_MBUFF_SKIP[SKIP_SZ]) used by the packet. (PIP/IPD writes the work-queue entry associated with each packet in this space when it is in packet-buffer-dependent work-queue entry allocation mode (i.e. when IPD_CTL_STATUS[NO_WPTR] = 1). Refer to Section 9.5.)
- the other for all subsequent buffers (IPD_NOT_1ST_MBUFF_SKIP[SKIP_SZ]) used by the packet.

Cavium Confidential
Bowen Yang
ALU
02/27/2014

These reserved field sizes can be as small as zero and as large as 32 64-bit words. Software can also configure the buffer size (IPD_PACKET_MBUFF_SIZE[MB_SIZE]), again in 64-bit words, defined as the distance from the start of the buffer to the first word that the PIP/IPD hardware cannot write. IPD_PACKET_MBUFF_SIZE[MB_SIZE] must always be at least 18 64-bit words larger than IPD_1ST_MBUFF_SKIP[SKIP_SZ], and at least 16 64-bit words larger than IPD_NOT_1ST_MBUFF_SKIP[SKIP_SZ]. Provided there is enough packet data, the PIP/IPD hardware always fills the buffer with the maximum amount of packet data allowed by these configuration values (and the alignment pad in the case of the first buffer). The bytes in the buffer that do not contain packet data are unpredictable.

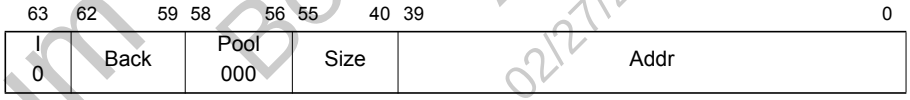
The PIP/IPD always writes packet data in complete (128-byte) cache blocks, including when it writes the first and last data. This means that if a buffer ends within the same cache block as the last buffer write allowed by IPD_PACKET_MBUFF_SIZE[MB_SIZE], but not at the end of a cache block, the memory immediately following the end of the buffer will be corrupted.

PIP/IPD has several modes to write packet data to L2/DRAM, selected by IPD_CTL_STATUS[OPC_MODE]:

- Write all packet data into the L2 cache.
- Write all packet data, but don't allocate space for it in the L2 cache.
- Write the first cache block of packet data into the L2 cache, and write any remaining packet data without allocating space for it in the L2 cache.
- Write the first two cache blocks of packet data into the L2 cache.
- Write any remaining packet data without allocating space for it in the L2 cache.

The PIP/IPD hardware links the packet segments in Figure 9–6 together with buffer pointers. It writes the first buffer pointer in the work-queue entry created for the packet, and writes all subsequent buffer pointers exactly eight bytes before the first byte of packet data. Figure 9–7 shows the format of each 64-bit buffer pointer. Note that the buffer-pointer field in the last buffer contains unpredictable data that should never be used by core software. All other buffer pointers are valid ones. The PIP/IPD hardware allocates space in every buffer for a buffer pointer prior to the packet data in the buffer, though the buffer-pointer field in the last buffer is invalid.

Figure 9–7 Format Of 64-bit Buffer Pointer



Addr	Byte address of start-of-packet in the buffer.
Size	Size of the buffer in bytes.
Pool	Always 0x0.
Back	Distance from Addr to beginning of buffer (loaded from the configuration register).
I	Always 0.

The **Addr** field in a valid-buffer pointer is always the physical L2/DRAM address of the first byte of packet data in the buffer. The Addr field in the buffer pointer in the work-queue entry might not be aligned on a 64-bit boundary, corresponding to the start of the packet data in the first buffer, but the buffer pointers in all subsequent buffers are naturally aligned on a 64-bit boundary.

Cavium Confidential For Bowen Yang ALU 02/27/2013

Except for the last valid-buffer pointer, the **Size** field in Figure 9–7 contains exactly the number of bytes of packet data held in the buffer. The Size field in the last valid-buffer pointer indicates the distance from Addr to the end of the buffer, just like the Size field in all other valid-buffer pointers. However, the Size field in the last valid-buffer pointer is larger than the number of bytes of packet data in the last buffer unless the packet ends exactly on the buffer boundary. When there is only one buffer, the last valid-buffer pointer is the one in the work-queue entry. Otherwise, the last valid-buffer pointer is the one in the second-to-last buffer.

CN38XX/CN36XX pass-2 versions have an erratum in which the Size field is too large by 8. The current CN71XX version is compatible with this specification (in which the Size field is correct) only when IPD_CTL_STATUS[LEN_M8] = 1. When IPD_CTL_STATUS[LEN_M8] = 0 (default value), the Size field is too large by 8, as it is in pass-2 CN38XX/CN36XX parts.

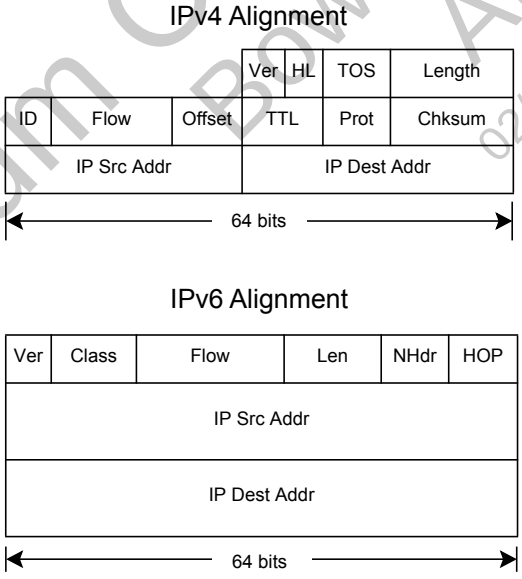
The value in the **Back** field in Figure 9–7 can indicate the distance from Addr to the beginning of the buffer (in cache blocks). PIP/IPD hardware creates it from one of two configuration variables:

- IPD_1ST_NEXT_PTR_BACK[BACK]
- IPD_2ND_NEXT_PTR_BACK[BACK]

one value for the first buffer pointer (i.e. the buffer pointer in the work-queue entry), and another for all other buffer pointers (i.e. the buffer pointers in the buffers). Software can precalculate these Back field values to match the reserved field sizes in the two cases. Sections 10.4, 10.7, 12.3.2, and 12.3.7 describe usages of the Back field, which generally specifies the number of 128-byte cache blocks.

The alignment-pad field in the first buffer in Figure 9–6 can be between 0 and 7 bytes. It places the packet data at a convenient alignment for core software processing. For example, Figure 9–8 shows the alignment for IPv4 and IPv6 packets. (The PIP hardware aligns a packet that it considers to be IP, but that has neither 4 nor 6 in the IP version field, as if it were an IPv4 packet.) In the non-IP cases, configuration variables (PIP_GBL_CFG[NIP_SHF] and PIP_GBL_CFG[RAW_SHF]) select the alignment pad value.

Figure 9–8 Packet Alignment for IPv4 and IPv6 Packets



Cavium Confidential For Bowen Yang ALU 02/27/2014

The following pseudo code describes the number of alignment pad bytes added by the PIP/IPD hardware:

```
if (packet had an L1/L2 receive error) {
  // WORD2[RE] is set in this case
  if (pkt_len <= 128)
    alignment pad = PIP_GBL_CFG[NIP_SHF];
  else
    alignment pad = unpredictable; // but in the range 0..7
}
else if (ISRAWFULL)
  alignment pad = PIP_GBL_CFG[RAW_SHF];
else if (Is_IP) {
  // WORD2[NI] is clear in this case
  if (IP.ver == 6)
     // WORD2[V6] is set
    alignment pad = (8 - (SKIP + L2_size)) & 0x7;
  else
     // WORD2[V6] is clear
    alignment pad = (4 - (SKIP + L2_size)) & 0x7;
}
else
  // WORD2<9> is set in this case
  alignment pad = PIP_GBL_CFG[NIP_SHF];
```

This code uses the following definitions from the pseudo-code in Section 9.2.5:

- **L2_size** is the number of L2 HDR bytes. L2_size is zero when in skip-to-IP mode. Figure 9–6 shows the L2_size values for the skip-to-L2 mode
- **SKIP** is the total number of bytes in any Skip I, PKT_INST_HDR, and Skip II fields, if present.
- **Is_IP** is set for an IP packet.
- **ISRAWFULL** is set for a RAWFULL packet.

Note that PIP_GBL_CFG[NIP_SHF] and PIP_GBL_CFG[RAW_SHF] are common for all packets from all input ports.

Figures 9–6 and 9–8 depict the packet-data format when PIP/IPD is in big-endian mode. The unit is in big-endian mode for the packet data in the buffers when IPD_CTL_STATUS[PKT_LEND] is clear. In big-endian mode, the unit places sequential packet bytes from most-significant to least-significant within a 64-bit word in an L2/DRAM buffer, and stores the next buf pointers in big-endian format.

When IPD_CTL_STATUS[PKT_LEND] is set, the unit is in little-endian mode, and it instead places sequential packet bytes from least significant to most significant within a 64-bit word in an L2/DRAM buffer, and stores the next pointers in little-endian format. Note that the number of alignment-pad bytes is identical in both big and little endian modes, but the most-significant bytes are the pad bytes in big-endian mode, while the least-significant bytes are the pad bytes in little-endian mode. Note also that IPD_CTL_STATUS[PKT_LEND] does not affect the format of the packet data in the work-queue entry.

Note that PIP/IPD can support a maximum of 255 buffers for a packet and a maximum packet size of 65535 bytes.

Cavium Confidential For Bowen Yang ARU 5/27/2014

9.4 Packet Scheduling

Unless it drops an inbound packet, PIP/IPD creates a work-queue entry for each packet (refer to Sections 9.6 and 9.7 for PIP/IPD drop conditions). The work-queue entry contains the following fields that are important for packet scheduling:

- QOS: Selects one of the POW input queues (one of eight).
- GRP: Selects the group attached to the work (one of sixteen).
- TT: Tag type: NULL, ORDERED, or ATOMIC.
- Tag: 32-bit tag value.

The remainder of this section qualitatively describes how these fields are generated by PIP/IPD. Section 9.5 describes the fields in more detail. Refer to Chapter 7 for more information on work-queue entries and how these fields affect work scheduling.

9.4.1 RAWFULL and RAWSCHED Packets

For RAWFULL and RAWSCHED packets, when the corresponding PKT_INST_HDR[NQOS,NGRP,NTT,NTAG] = 0 or PIP_GBL_CTL[IHMSK_DIS] = 1, the QOS, GRP, TT and Tag generated by PIP/IPD are directly specified in fields in the packet instruction header (PKT_INST_HDR[QOS,GRP,TT,TAG]).

For RAWFULL and RAWSCHED packets, when the corresponding PKT_INST_HDR[NQOS,NGRP,NTT,NTAG] = 1 and PIP_GBL_CTL[IHMSK_DIS] = 0, PIP can calculate QOS, GRP, TT and Tag values as described in Sections 9.4.3 through 9.4.6.

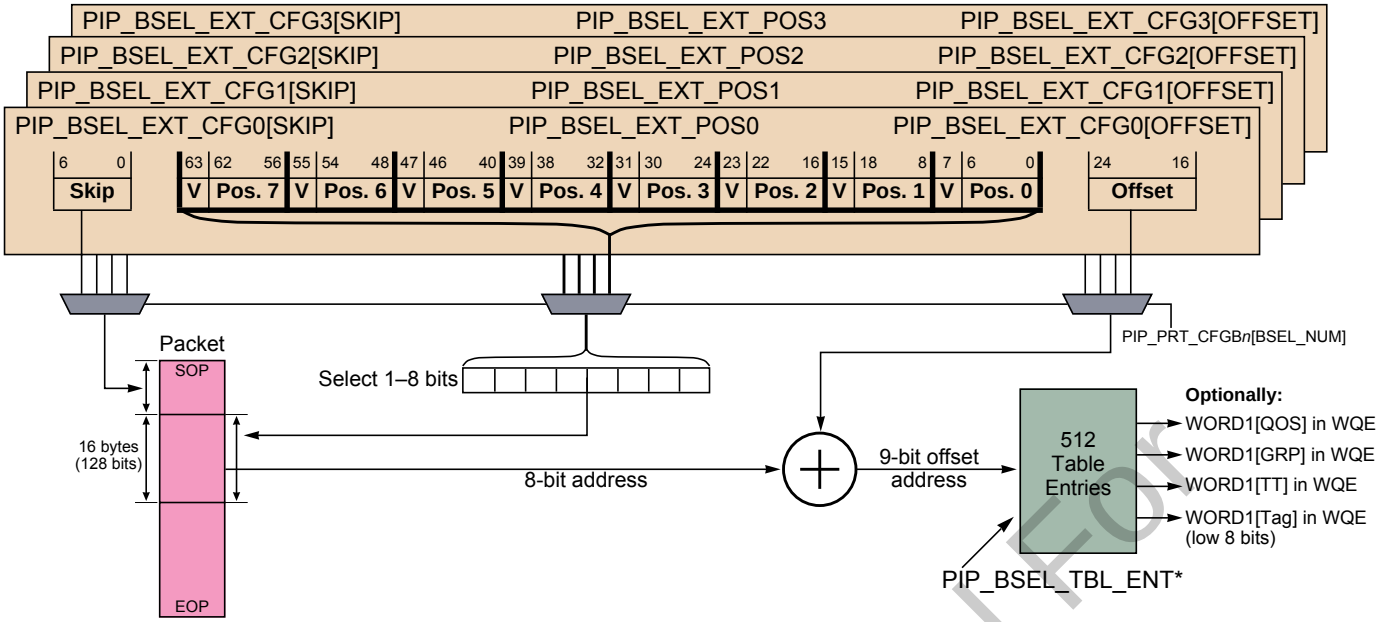
9.4.2 Bit-Select Extractor

While PIP provides numerous options for packet-scheduling determination, there can be times when the bits used to determine scheduling may be out of the range that PIP parses. For example, PIP does not parse bits that are in packet regions that PIP treats as uninterpreted (either using the UNINTERPRETED parse mode or residing in SKIP I or SKIP II region). In these cases, software can configure PIP to select the bits of the packet in order to lookup scheduling parameters using the bit-select extractor feature (packet parsing algorithm). This gives precise scheduling control for these packets.

Figure 9–9 shows how one of the four bit-extractors uses the packet contents to index into a table in order to optionally determine the QOS, GRP, TT, and TAG for the packet. The table contents determine which of the fields to use.

Cavium Confidential For Bowen Yang AT&T 12/27/2014

Figure 9–9 Bit-Select Extractor



PIP_PRT_CFGB*n*[BSEL_EN] enables the bit extractor for the port, and PIP_PRT_CFGB*n*[BSEL_NUM] selects one of the four extractors. PIP_BSEL_EXT_CFG*n*[SKIP] specifies an offset in bytes from SOP into the packet. PIP_BSEL_EXT_POS*n* specifies which eight (or fewer) bits are used to build the address that is added to the nine-bit offset specified by PIP_BSEL_EXT_CFG*n*[OFFSET].

9.4.3 QOS

The QOS value can be calculated by PIP/IPD in a variety of ways for non-RAWFULL and non-RAWSCHED packets:

- The default QOS value for a packet may be used. This default value can be configured separately for each input port.
- The DSA/VLAN priority field can override the default QOS when DSA/VLAN is present. This can be enabled on a per-port basis, and there is an eight-entry table to convert the three-bit DSA/VLAN priority field into a QOS value. VLAN stacking is supported, and either VLAN priority field may be used when VLAN stacking is used.
- HiGig* priority fields can optionally override the default QOS when in HiGig* mode (i.e. when PIP_PRT_CFG*n*[HIGIG_EN] = 1). This can be enabled on a per-port basis, and there is a 64-entry table to convert the (up to 6-bit) HiGig* priority field into a QOS value.
- The IP DiffServ field can override the QOS value for IP packets. This can be enabled on a per-port basis, and there is a 64-entry table to convert the DiffServ field (i.e. IP.TOS/CLASS<7:2>) into a QOS value.
- One of the eight “QOS/GRP Watchers” can override the QOS value for IP packets that match a protocol/next_header value or TCP/UDP port or ethertype. Each QOS watcher is individually enabled on a per-port basis.
- The bit-select extractor can override the QOS value (refer to Section 9.4.2). The bit-select extractor is enabled on a per-port basis.

Cavium Confidential For Bowen Yang ALU 02-7-2014

9.4.4 GRP

The GRP value can be calculated by PIP/IPD in a number of ways for non-RAWFULL and non-RAWSCHED packets:

- The default GRP for the packet may be used. This default value can be configured separately for each port.
- Low-order tag bits can override the GRP value. This can be enabled on the per-port basis, and the tag width and base GRP value are also programmable on a per-port basis.
- In DSA mode, the low-order SRC_ID or VLAN ID can be converted into a GRP value.
- One of eight “QOS/GRP Watchers” can override the GRP value for IP packets that match a protocol/next_header value or TCP/UDP port or ethertype. Each GRP watcher is individually enabled on a per-port basis.
- The bit-select extractor can override the GRP value (refer to Section 9.4.2). The bit-select extractor is enabled on a per-port basis.

9.4.5 TT

For IP packets that are neither RAWFULL nor RAWSCHED, there are four per-port configured TT values used by PIP/IPD:

- An IPv4 TCP TT value
- An IPv4 non-TCP value
- An IPv6 TCP TT value
- An IPv6 non-TCP value

For non-IP packets that are neither RAWFULL nor RAWSCHED, there is a per-port configured TT value used by PIP/IPD.

For packets, the bit-select extractor can override the TT value (refer to Section 9.4.2). The bit-select extractor is enabled on a per-port basis.

9.4.6 Tag

There are two major tag-generation algorithms that are used for packets that are neither RAWFULL nor RAWSCHED: tuple and mask.

In many cases, the goal of the PIP/IPD tag-generation algorithms is to generate different tags for different flows, so CN71XX can schedule packets for different flows freely to different cores. Tuple-tag generation can create a hash of up to a 7-tuple. Tuple-tag generation is well-suited to IP TCP 5-tuple hash generation, for example. The mask tag is useful for non-IP fixed headers, and also for including additional information from user-defined headers.

The tuple and hash algorithms can be combined in the following ways based on a per-port configured tag mode:

- Always use the tuple tag.
- Always use the mask tag.
- Use the tuple tag for IP packets, and the mask tag for non-IP packets.
- Exclusive-OR (XOR) the tuple and the mask tags.

Cavium Confidential For Bowen Yang ALU 3/2/2014

For an IP packet, the tuple tag is an XOR of a source and destination hash value. This XOR can give the same result for both directions of a TCP flow, for example. The tuple-tag generation scheme also includes a secret configured value that makes it difficult or impossible for potential denial-of-service attackers to predict tag/flow collisions.

The PIP/IPD source tuple-hash value is a CRC of any combination of these two fields, each selected by per-port configuration:

- IP source address
- IP TCP/UDP source port

PIP/IPD zeroes the source tuple-hash value for an initiating TCP SYN packet if selected on a per-port basis.

The PIP/IPD destination tuple-hash value is a CRC of any combination of these fields, each selected by per-port configuration:

- DSA/VLAN ID
  - Either or both in the stacked DSA/VLAN case
- IP Destination Address
- IP Protocol/next header value
- IP TCP/UDP destination port

The mask tag is a CRC of selected bytes from the first 128 bytes in the packet. The bytes are selected by four 128-bit masks. Each bit in each mask represents one of the first 128-bytes in the packet. (Note specifically that mask-tag generation is not affected by any SKIP value.) Each port can be configured to use any one of the four 128-bit masks.

The resultant tuple/mask tag is up to 16 bits. If desired, fewer bits can be used, generating fewer unique tag values. If desired, the port number can be included in the tag generated by PIP/IPD.

The Tag value can be overridden by the bit-select extractor (refer to Section 9.4.2). The bit-select extractor is enabled on a per-port basis.

9.5 Work-Queue Entry

The PIP/IPD hardware creates a work-queue entry for each packet that arrives and that it does not decide to drop. (Sections 9.6 and 9.7 cover the PIP/IPD drop conditions.) CN71XX supports two different work-queue entry allocation modes:

- packet-buffer-independent (IPD_CTL_STATUS[NO_WPTR] = 0)
- packet-buffer-dependent (IPD_CTL_STATUS[NO_WPTR] = 1).

When IPD_CTL_STATUS[NO_WPTR] = 0, PIP/IPD is in packet-buffer-independent mode. In this mode, the hardware allocates the work-queue entry from a configured hardware free pool (IPD_WQE_FPA_QUEUE[WQE_QUE]), and PIP/IPD prequeues a number of the pointers to work-queue entries that it receives from FPA. The number of prequeued work-queue-entry buffers in this mode is:

IPD_PTR_COUNT[WQE_PCNT] + IPD_PTR_COUNT[WQEV_CNT].

When IPD_CTL_STATUS[NO_WPTR] = 1, PIP/IPD is in packet-buffer-dependent mode. In this mode, PIP/IPD does not prequeue any work-queue-entry buffers and the hardware writes the work-queue entry into the space (size selected by

Cavium Confidential For Bowen Yang ALU 2/27/2014

IPD_1ST_MBUFF_SKIP[SKIP_SZ]) prior to the packet data in the buffer (refer to Figure 9–6). IPD_1ST_MBUFF_SKIP[SKIP_SZ] must be at least 16 (64-bit words) in this mode to leave enough space in the first buffer for the work-queue entry. Note that packet buffers always begin on 128-byte boundaries.

There are two possibilities in packet-buffer-dependent mode:

- If IPD_1ST_MBUFF_SKIP[SKIP_SZ] is between 16 and 31, PIP/IPD writes the work-queue entry for each packet in the first naturally-aligned 128-byte block at the front of the first buffer for the packet.
- If IPD_1ST_MBUFF_SKIP[SKIP_SZ] = 32, PIP/IPD writes the work-queue entry for each packet in the second naturally-aligned 128-byte block at the front of the first buffer for the packet.

IPD holds no pointers if it has not yet been enabled.

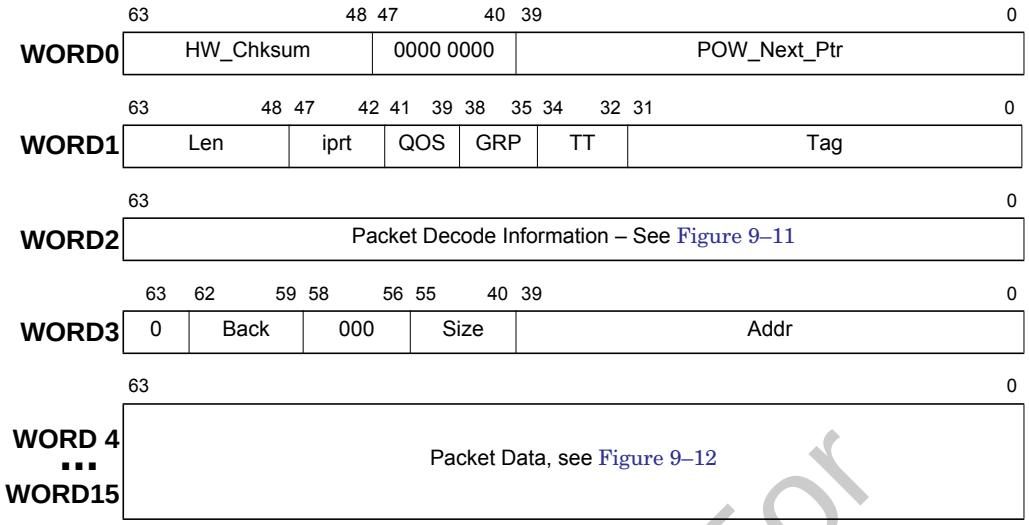
Figure 9–10 shows the work-queue entry created by the PIP/IPD hardware for input packets. The work-queue entry is 128 bytes, or 16 64-bit words. WORD0 and WORD1 contain scheduling information and match the format in Figure 7–6, with some additional information added into fields that are not used by the POW unit hardware. WORD2 contains decode information that is described in Figure 9–11. WORD3 contains a pointer to the packet data in the first buffer.

Words 4–15 contain packet data, in different formats as defined in Figure 9–12. When the unit considers the packet to be an IP packet, it places the IP at an offset defined by PIP_IP_OFFSET[OFFSET]. With this flexibility, the particular configuration defines the position of the IP packet in the work-queue entry, and the amount of (pre-IP) packet header data in the work-queue entry.

Note that Figures 9–10, 9–11, and 9–12 contain a simple description of the fields in the figure. More-detailed descriptions of the fields are provided in the text following the figures.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Figure 9–10 PIP/IPD Hardware Work-Queue Entry

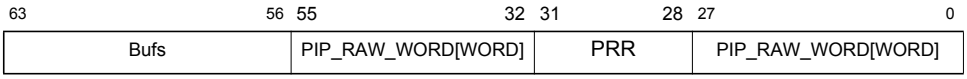


POW_Next_Ptr	Reserved for POW hardware.
HW_Chksum	Hardware-calculated checksum of packet bytes.
Len	Total packet length in bytes.
iprt	Input port number.
QOS	POW input queue used for the work.
GRP	Core group number used for the work.
TT	Tag type used for the work (Atomic, Ordered, or Null)
Tag	Tag used for the work.
Addr	Byte address of start-of-packet in the first buffer.
Size	Size of the first buffer in bytes.
Back	Distance from Addr to beginning of the first buffer (loaded from the configuration register).

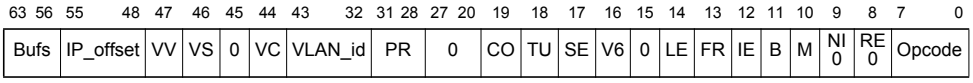
Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 9–11 Work-Queue Entry Format; Word 2 Cases

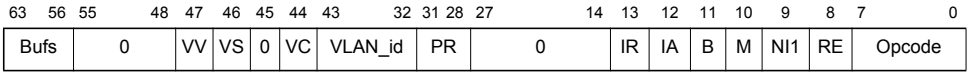
WORD2 (RAWFULL and (No L2/L1 Error))



WORD2 (Is_IP and (No L2/L1 Error))



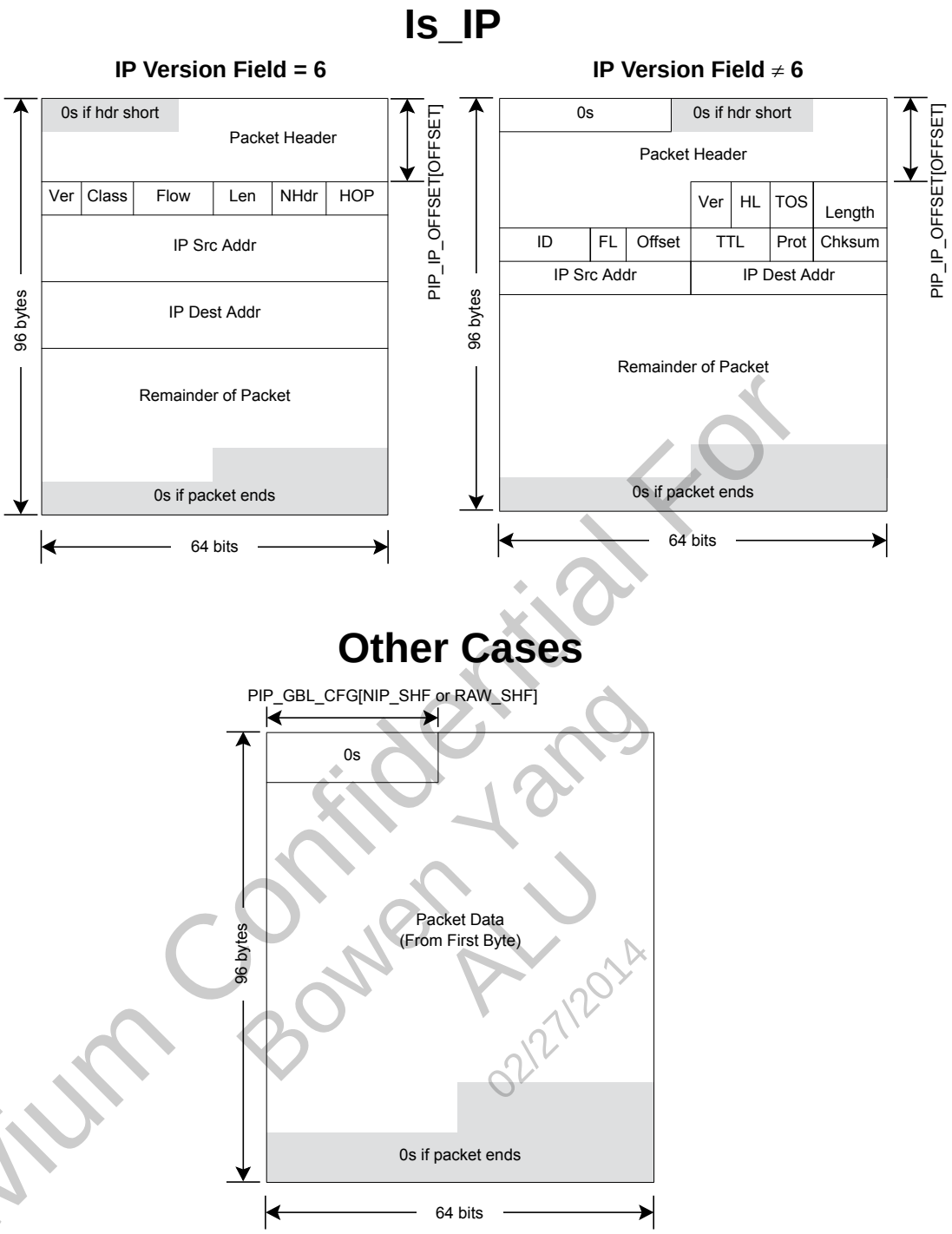
WORD2 (All Other Cases)



Bufs	Number of buffers used to store the packet.
IP_Offset	Number of bytes from start-of-packet to the start of IP.
VV	Set to 1 when DSA/VLAN is found in L2 HDR.
VS	Set to 1 when stacked DSA/VLAN is present.
VC	When VV = 1, the DSA/VLAN CFI bit; otherwise 0.
VLAN_id	When VV = 1, the DSA/VLAN ID field; otherwise 0.
PR	Ring Identifier (if DPI)
PRR	Ring Identifier for RAWFULL (if DPI) or PIP_RAW_WORD[WORD]
CO	Set to 1 when IP packet is IPCOMP.
TU	Set to 1 when IP packet is TCP or UDP.
SE	Set to 1 when IP may packet require IPSEC decryption.
V6	IP version 6.
LE	IP packet has TCP/UDP error.
FR	IP packet is a fragment.
IE	IP packet error.
B	Packet's DMAC is broadcast.
M	Packet's DMAC is multicast.
NI	Not an IP packet.
RE	Packet has L2/L1 error.
IR	Packet's Type indicates RARP.
IA	Packet's Type indicates ARP.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 9–12 Work-Queue Entry Format; Word 4–15 Cases



Words 4–15 are unpredictable when WORD2[RE] = 1 and the packet is not RAWFULL.

Figure 9–12 shows the packet data format in Words 4–15 when the PIP/IPD places the packet data into the work-queue entry in big-endian format, as is the case when IPD_CTL_STATUS[WQE_LEND] is clear. When IPD_CTL_STATUS[WQE_LEND] is set, the unit instead inserts the packet data bytes in Words 4–15 of the work-queue entry in little-endian format, from least-significant to most-significant within each word rather than most-significant to least-significant. In other words, the unit byte-swaps Words 4–15 when IPD_CTL_STATUS[WQE_LEND] is set.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Whenever the work-queue entry contains valid packet data, the address of the first byte of packet data in the work-queue entry can be calculated as follows:

```
WQEbytesAddr = WQE_ptr + 32; // get to WORD4
if(Is_Rawfull)
   WQEbytesAddr += PIP_GBL_CFG[RAW_SHF];   // RAWFULL case
else if(WORD2[NI])
   WQEbytesAddr += PIP_GBL_CFG[NIP_SHF];   // non-IP case
else
   WQEbytesAddr += PIP_IP_OFFSET[OFFSET]*8 // IP case
                 + (WORD2[V6] ^ 1)*4
                 - MIN(WORD2[IP_offset],PIP_IP_OFFSET[OFFSET]*8);
```

Is_Rawfull should correctly indicate when the packet was RAWFULL. This can be calculated via a number of means, including PIP_RAW_WORD[WORD] values that guarantee unique WORD2 fields for RAWFULL packets. The work-queue entry contains valid packet data whenever WORD2[RE] = 0[1].

PIP/IPD analyzes the packet length, work-queue-entry alignment configuration, and dynamic alignment to determine if the packet will fit into Words 4–15 of the work-queue-entry or if it requires buffering in L2/DRAM. A packet that meets such criteria is said to be “dynamic short.” Basically, dynamic short means that all packet bytes reside in Words 4–12 of the work-queue entry, so the ordinary L2/DRAM copy of the packet might be considered redundant. In order for a packet to be dynamic short in the IP case, all of the header prior to the IP packet must fit into PIP_IP_OFFSET[OFFSET] words, and all of the IP packet and pad must fit into the remaining Words 4–15 of the work-queue entry. Refer to Figure 9–1 for the definition of pad.

Note that any packet that has an L2/L1 receive error is never dynamic short. Note also that the packet includes any inbound CRC bytes that are not stripped (see Section 9.2.6). Note specifically that for an IPv4 packet, the first four bytes in Word 4 of the work-queue entry can never buffer packet data, so the effective work-queue buffer size is 92 bytes rather than the possible 96 bytes for IPv4, as Figure 9–12 shows.

When PIP/IPD classifies a packet as dynamic short, it will only (redundantly) buffer the packet in the L2/DRAM buffer if it is not enabled to avoid the ordinary L2/DRAM copy. When PIP/IPD does not write the ordinary copy, WORD2[Bufs] is set to 0 to indicate that no buffers have been allocated for the packet (outside the work-queue entry, that is).

PIP/IPD is enabled to avoid the ordinary L2/DRAM copy of dynamic short packets whenever one of the following occurs:

- PIP_PRT_CFG*n*[DYN_RS] = 1, or
- the packet has a PKT_INST_HDR and both of the following are true:
  - PKT_INST_HDR[RS] =1, and
  - either PIP_GBL_CTL[IGNRS] = 0, or the port is a DPI port.

Cavium Confidential For Bowen Yang ALU 02/27/2014

1. Technically, WORD2[RE] doesn’t exist for RAWFULL packets. But Cavium recommends that PIP_RAW_WORD[WORD<8>] = 0, so the system software can act as if WORD2[RE] is always present for work-queue entries created by PIP/IPD.

Note that the PIP/IPD hardware ignores both PIP_PRT_CFG*n*[DYN_RS] and PKT_INST_HDR[RS] for packets that the hardware does not classify dynamic short, and always creates ordinary L2/DRAM copies of these packets. Thus, it is legal for PKT_INST_HDR[RS] to be set for large packets. For example, it can be set for all packets, and PIP/IPD will buffer minimally.

The remainder of this section is a description of each field in WORD0 – WORD3 of the work-queue entry. Section 9.2.5 describes the Pre-IP parsing that occurs prior to the pseudo-code fragments included below, and defines ethertype, ethertype_val, Is_IP, ISRAWFULL, and ISRAWSCH.

WORD0[HW_Chksum] Refer to Figure 9–10. A hardware-generated 16-bit one's complement sum of the packet data.

WORD0[HW_Chksum] is the 1's complement sum calculated over the packet from [SKIP + 20B] until the end of packet (including any optional CRC bytes). The packet is padded with a zero octet at the end (if necessary) to make it a multiple of two octets.

Software should not strip the FCS (i.e. IPD_SUB_PORT_FCS[PORT_BIT<*n*>] should be clear, refer to Section 9.2.6) from ports for which WORD0[HW_Chksum] may be used. This is because the CRC bytes are included in the WORD0[HW_Chksum] value, and software will likely need to reference the calculated CRC value to effectively use the WORD0[HW_Chksum] value.

The starting checksum value is 0. If the packet is less than the start of the checksum, then the checksum will simply be 0.

WORD0[POW_Next_Ptr] Refer to Figure 9–10. Reserved for POW Unit. See Chapter 7.

WORD1[Len] Refer to Figure 9–10. The total number of bytes of packet data, from 1 to 65535. The total packet length includes all the packet bytes shown in the formats in Figure 9–1, except that the CRC field is discarded when IPD_SUB_PORT_FCS[PORT_BIT<n>] = 1. (If IPD_SUB_PORT_FCS[PORT_BIT<n>] = 0, both the Len value and the stored packet (both the work-queue-entry copy and the ordinary copy) include the CRC bytes. See Section 9.2.6 for more details on CRC processing.)

WORD1[Len] exactly matches the number of packet bytes written to the ordinary L2/DRAM copy of the packet, except for dynamic short packets when PIP/IPD avoids the ordinary L2/DRAM copy.

WORD1[iprt] Refer to Figure 9–10. The input port that the packet arrived on, as described in Section 9.1.

Cavium Confidential For Bowen Yang ALS 02/27/2014

WORD1[QOS] Refer to Figure 9–10. The POW-unit input queue used for the work. For RAWFULL and RAWSCH packets, QOS can be PKT_INST_HDR[QOS]. Otherwise, PIP/IPD selects this (differently for each port) based on a number of parameters, including:

- a default QOS value for each port (PIP_PRT_CFG*n*[QOS]).
- Two eight-entry tables (PIP_QOS_VLAN*n*[VLAN.priority][QOS,QOS1]) to convert a DSA/VLAN priority (from the most-significant three bits of the VLAN field or from the DSA tag in Figure 9–4) to a QOS value. Per-port configuration (PIP_PRT_CFG*n*[QOS_VSEL]) selects which table.
- a VLAN enable bit for each port (PIP_PRT_CFG*n*[QOS_VLAN]) that causes the calculated DSA/VLAN QOS value to be used when the packet has a DSA/VLAN field (as shown in Figure 9–4) in the L2 HDR.
- A 64-entry table (PIP_HG_PRI_QOS gives access to the table) that optionally (PIP_PRT_CFG*n*[HG_QOS]) converts priority bits extracted from the HiGig* header into QOS values then in HiGig* mode. (The pseudo-code in Section 9.2.5 indicates which bits (HG_PRI<5:0>) are extracted from the HiGig* header.
- a VLAN selection bit, used for all ports, that selects which DSA/VLAN priority to use when VLAN stacking is detected. (In the pseudo-code below, VLAN0.priority is the first in network order, and VLAN1.priority is the second in network order.)
- a 64-entry table (PIP_QOS_DIFF*n*[IP.TOS/CLASS<7:2>][QOS]) to convert a diffserv DSCP field (the most-significant six bits of IPv4 TOS / IPv6 CLASS) to a QOS value.
- a diffserv-enable bit for each port (PIP_PRT_CFG*n*[QOS_DIFF]) that causes the calculated diffserv QOS value to be used when the packet is IP.
- eight QOS watchers (PIP_QOS_WATCH*n*[i]), including a QOS value, that can match IP protocol/next_header values, TCP/UDP destination port numbers, or ethertype. Note that the PKT_PARSE_MODE must be either SKIP-TO-L2 or SKIP-TO-IP to extract the fields that the watchers use to match.
- an enable bit for each watcher for each port (PIP_PRT_CFG*n*[QOS_WAT<i>]) that causes the QOS watcher QOS value to be used when the QOS watcher matches for the packet.
- the bit-select extractor configuration (refer to Section 9.4.2). The bit-select extractor is enabled on a per-port basis.

Cavium Confidential For Bowen Yang ALU 02/27/2014

The following pseudo code details the PIP/IPD work-queue QOS selection for a given packet:

```
QOS = PIP_PRT_CFG[port][QOS];
if (there is an L1/L2 receive error) {
  // WORD2[RE] will be set in this case, as will WORD2[NI]
}
else if((ISRAWFULL || ISRAWSCH) && (!PKT_INST_HDR[NQOS] ||PIP_GBL_CTL[IHMSK_DIS])) {
  QOS = PKT_INST_HDR[QOS];
}
else if(PIP_PRT_CFGB[port][BSEL_EN] && PIP_BSEL_TBL_ENT[bsel_tbl_idx][QOS_EN]) {
  // bsel_tbl_idx = select + PIP_BSEL_EXT_CFG[bsel_num][OFFSET], where
  // bsel_num = PIP_PRT_CFGB[port][BSEL_NUM],
  // select is the bits selected from the packet (using PIP_BSEL_EXT_CFG[bsel_num][SKIP]
  // and PIP_BSEL_EXT_POS[bsel_num])
  QOS = PIP_BSEL_TBL_ENT[bsel_tbl_idx][QOS];
}
else {
  if(PIP_PRT_CFG[port][HG_QOS]) {
    QOS = HG_QOS_TABLE[HG_PRI<5:0>]; // HG_QOS_TABLE has the PIP_HG_PRI_QOS[QOS] values
  }
  if(VV && PIP_PRT_CFG[port][QOS_VLAN]) {
    VLAN = (VS && PIP_GBL_CFG[VS_QOS]) ? VLAN1 : VLAN0;
    if(PIP_PRT_CFG[port][QOS_VSEL])
      QOS = PIP_QOS_VLAN[VLAN<15:13>][QOS1]; // VLAN<15:13> is priority
    else
      QOS = PIP_QOS_VLAN[VLAN<15:13>][QOS ]; // VLAN<15:13> is priority
  }
  if(Is_IP && !WORD2[IE] && !WORD2[LE] &&
      PIP_PRT_CFG[port][QOS_DIFF] && !(VV && PIP_PRT_CFG[port][QOS_VLAN] &&
      PIP_PRT_CFG[port][QOS_VOD]))
    QOS = PIP_QOS_DIFF[IP.TOS/CLASS<7:2>][QOS]; // the 6 msbs of TOS/CLASS
  for (i=0; i<8; i++) {
    if (PIP_PRT_CFG[port][QOS_WAT<i>] and PIP_QOS_WATCH[i][TYPE]!=0) {
      mask16 = ~PIP_QOS_WATCH[i][MASK] & 0xffff;
      mask8 = ~PIP_QOS_WATCH[i][MASK] & 0xff;
      match16 = PIP_QOS_WATCH[i][MATCH] & mask16;
      match8 = PIP_QOS_WATCH[i][MATCH] & mask8;
      if(Is_IP && !WORD2[IE] && !WORD2[LE] &&
            PIP_QOS_WATCH[i][TYPE] is protocol/next_header and
            match8 equals (IP protocol/next header field)&mask8) {
         QOS = PIP_QOS_WATCH[i][WATCHER];
         break;
      }
      else if(Is_IP && !WORD2[IE] && !WORD2[LE] &&
            PIP_QOS_WATCH[i][TYPE] is TCP and
            IP protocol/next header field is TCP and
            match16 equals (TCP destination port field)&mask16 and
            !WORD2[FR] and
            (WORD2[V6] || (ipv4.HL==5))) {
         QOS = PIP_QOS_WATCH[i][WATCHER];
         break;
      }
      else if(Is_IP && !WORD2[IE] && !WORD2[LE] &&
            PIP_QOS_WATCH[i][TYPE] is UDP and
            IP protocol/next header field is UDP and
            match16 equals (UDP destination port field)&mask16 and
            !WORD2[FR] and
            (WORD2[V6] || (ipv4.HL==5))) {
         QOS = PIP_QOS_WATCH[i][WATCHER];
         break;
      }
      else if(ethertype_val and
            PIP_QOS_WATCH[i][TYPE] is ETHER_TYPE and
            match16 equals (ethertype)&mask16) {
         QOS = PIP_QOS_WATCH[i][WATCHER];
         break;
```

Cavium Confidential For Power Ng ALU 6/17/2014

```
      }
    }
  }
}
```

Note that the QOS watchers will resolve in watcher order if more than one watcher matches the packet with different PIP_QOS_WATCH*n*[WATCHER] values. Note also that for IPv6, the IP next-header field is the initial next-header field.

WORD1[GRP] Refer to Figure 9–10. The core group number used for the work. For RAWFULL and RAWSCH packets, GRP can be PKT_INST_HDR[GRP].

- There is a default GRP value for each port (PIP_PRT_TAG[port][GRP] below).
- There is an enable bit per port (PIP_PRT_TAG[port][GRPTAG]) that causes PIP/IPD to set WORD1[GRP] to the least-significant bits of WORD1[Tag] (translated).
- When DSA mode is enabled (i.e. when PIP_PRT_CFG*n*[DSA_EN] = 1), the low-order 4-bits of SRC_ID or VLAN ID fields from the DSA tag can be converted into a 4-bit GRP value via 16-entry tables (PIP_DSA_SRC_GRP[MAP<SRC_ID<3:0>>] and PIP_DSA_VID_GRP[MAP<VLAN0<3:0>>], respectively). Three mode bits (PIP_GBL_CTL[DSA_GRP_SID,DSA_GRP_SCMD,DSA_GRP_TVID]) control which is selected, as well as the DSA TAG_CMD field. Section 9.2.5 defines the DSA tag SRC_ID, VLAN0, and TAG_CMD fields.
- There are eight GRP watchers (PIP_QOS_WATCH*n*[*i*] below), including a GRP value, that can match IP protocol/next_header values, TCP/UDP destination port numbers, or ethertype. Note that the PKT_PARSE_MODE must be either SKIP-TO-L2 or SKIP-TO-IP to extract the fields that the watchers use to match.
- There is an enable bit for each watcher for each port (PIP_PRT_CFG*n*[GRP_WAT<*i*>]) that causes the GRP watcher GRP value to be used when the GRP watcher matches for the packet.
- the bit-select extractor configuration (refer to Section 9.4.2). The bit-select extractor is enabled on a per-port basis.

```
GRP = PIP_PRT_TAG[port][GRP];
if (there is an L1/L2 receive error) {
  // WORD2[RE] will be set in this case, as will WORD2[NI]
}
else if((ISRAWFULL || ISRAWSCH) && (!PKT_INST_HDR[NGRP] || PIP_GBL_CTL[IHMSK_DIS])) {
  GRP = PKT_INST_HDR[GRP];
}
else if(PIP_PRT_CFGB[port][BSEL_EN] && PIP_BSEL_TBL_ENT[bsel_tbl_idx][GRP_EN]) {
  // bsel_tbl_idx = select + PIP_BSEL_EXT_CFG[bsel_num][OFFSET], where
  // bsel_num = PIP_PRT_CFGB[port][BSEL_NUM],
  // select is the bits selected from the packet (using PIP_BSEL_EXT_CFG[bsel_num][SKIP]
  // and PIP_BSEL_EXT_POS[bsel_num])
  GRP = PIP_BSEL_TBL_ENT[bsel_tbl_idx][GRP];
}
else {
  if((Is_IP || PIP_PRT_TAG[port][GRPTAG_MSKIP]) && PIP_PRT_TAG[port][GRPTAG]) {
    GRP = ((WORD1[Tag] & ~PIP_PRT_TAG[port][GRPTAGMASK])
         + PIP_PRT_TAG[port][GRPTAGBASE]) & 0xF;
  }
  if(PIP_PRT_CFG[port][DSA_EN]) {
    if(PIP_GBL_CTL[DSA_GRP_SID]
       || (PIP_GBL_CTL[DSA_GRP_SCMD] && (TAG_CMD == 0)))
      GRP = PIP_DSA_SRC_GRP[MAP<SRC_ID<3:0>>];
    else if(PIP_GBL_CTL[DSA_GRP_TVID])
      GRP = PIP_DSA_VID_GRP[MAP<VLAN0<3:0>>];
  }
  for (i=0; i<8; i++) {
```

Cavium Confidential For Bowen Yang ALU 02/2/2014

```
    if (PIP_PRT_CFG[port][GRP_WAT<i>] and PIP_QOS_WATCH[i][TYPE]!=0) {
      mask16 = ~PIP_QOS_WATCH[i][MASK] & 0xffff;
      mask8 = ~PIP_QOS_WATCH[i][MASK] & 0xff;
      match16 = PIP_QOS_WATCH[i][MATCH] & mask16;
      match8 = PIP_QOS_WATCH[i][MATCH] & mask8;
      if(!WORD2[IE] && !WORD2[LE] &&
            PIP_QOS_WATCH[i][TYPE] is protocol/next_header and
            match8 equals (IP protocol/next header field)&mask8) {
         GRP = PIP_QOS_WATCH[i][GRP];
         break;
    }
    else if(!WORD2[IE] && !WORD2[LE] &&
            PIP_QOS_WATCH[i][TYPE] is TCP and
            IP protocol/next header field is TCP and
            match16 equals (TCP destination port field)&mask16 and
            !WORD2[FR] and
            (WORD2[V6] || (ipv4.HL==5))) {
       GRP = PIP_QOS_WATCH[i][GRP];
       break;
    }
    else if(!WORD2[IE] && !WORD2[LE] &&
            PIP_QOS_WATCH[i][TYPE] is UDP and
            IP protocol/next header field is UDP and
            match16 equals (UDP destination port field)&mask16 and
            !WORD2[FR] and
            (WORD2[V6] || (ipv4.HL==5))) {
       GRP = PIP_QOS_WATCH[i][GRP];
       break;
    }
    else if(ethertype_val and
            PIP_QOS_WATCH[i][TYPE] is ETHER_TYPE and
            match16 equals (ethertype)&mask16) {
       GRP = PIP_QOS_WATCH[i][GRP];
       break;
      }
    }
  }
}
```

Note that the GRP watchers resolve in watcher order if more than one watcher matches the packet with different PIP_QOS_WATCH*n*[GRP] values. Note also that for IPv6, the IP next header field is the initial next header field.

WORD1[TT] Refer to Figure 9–10. The initial tag type, (NULL, ORDERED, or ATOMIC), for the work-queue entry.

For both RAWFULL and RAWSCH packets, TT can be PKT_INST_HDR[TT].

For other packets, there are per-port configuration variables (PIP_PRT_TAG*n* below) that control the tag type. The tag type can be:

- different for IP and non-IP packets
- different for IPv4 and IPv6 packets
- different for TCP and non-TCP packets

For all packets, the bit-select extractor can override the TT value (refer to Section 9.4.2. The bit-select extractor is enabled on a per-port basis.

Cavium Confidential For Bowen Yang AU 02/27/2015

The following pseudo code details the hardware behavior:

```
// zero-extend the two-bit fields to make the three-bit TT field
TT = PIP_PRT_TAG[port][NON_TAG]
if(there is a L2/L1 receive error) {
  // WORD2[RE] will be set, as will WORD2[NI]
}
else if((ISRAWFULL || ISRAWSCH) && (!PKT_INST_HDR[NTT] || PIP_GBL_CTL[IHMSK_DIS])) {
  TT = PKT_INST_HDR[TT]
}
else if(PIP_PRT_CFGB[port][BSEL_EN] && PIP_BSEL_TBL_ENT[bsel_tbl_idx][TT_EN]) {
  // bsel_tbl_idx = select + PIP_BSEL_EXT_CFG[bsel_num][OFFSET], where
  // bsel_num = PIP_PRT_CFGB[port][BSEL_NUM],
  // select is the bits selected from the packet (using PIP_BSEL_EXT_CFG[bsel_num][SKIP]
  // and PIP_BSEL_EXT_POS[bsel_num])
  TT = PIP_BSEL_TBL_ENT[bsel_tbl_idx][TT];
}
else if(Is_IP) {
  if(WORD2[V6])
    proto_nh = IPv6.next_header; // the initial next header
  else
    proto_nh = IPv4.protocol;
  // WORD2[NI] will always be clear here
  if(!WORD2[IE] and (proto_nh==6)) {
    if(WORD2[V6])
      TT = PIP_PRT_TAG[port][TCP6_TAG]
    else
      TT = PIP_PRT_TAG[port][TCP4_TAG]
  }
  else {
    if(WORD2[V6])
      TT = PIP_PRT_TAG[port][IP6_TAG]
    else
      TT = PIP_PRT_TAG[port][IP4_TAG]
  }
}
```

WORD1[Tag] Refer to Figure 9–10. The initial tag for the work-queue entry.

For RAWFULL and RAWSCH packets, Tag can be PKT_INST_HDR[TAG].

For other packets:

- Tag<31:24> is always 0x0.
- Tag<23:16> is either the port number that the packet arrived on or all 1s.
- Tag<15:0> is a hash of specific fields within the packet.

PIP_PRT_TAG*n*[TAG_MODE] picks from among various combinations of the two different tag generation algorithms.

The first tag generation algorithm is the tuple hash. The tuple hash optionally includes the IP source and destination addresses, the IP protocol/next_header value, the TCP/UDP source and destination ports, and DSA/VLAN ID. There are per-port configuration variables (PIP_PRT_TAG) that control these options. The IPv4 and IPv6 configuration variables are separate. The tuple hash calculation also includes a secret value. The tuple hash results and, most importantly, potential collisions between different tuple hash results change with different secret values.

The second tag generation algorithm is the mask hash. The mask hash can optionally include any user-defined range of bytes within the first 128B of a packet. This is accomplished with four 128-bit masks (PIP_TAG_INC*n*). Each bit in each mask represents one of the first 128 bytes in the packet. (Note specifically that the mask hash is NOT affected by any SKIP values — each 128-bit mask refers strictly to the

Cavium Confidential For Bowen Yang ALU 02/27/2014

first 128 bytes in the packet.) Each port can be configured to use any of the 4 masks with PIP_PRT_CFG*n*[TAG_INC]. PIP_PRT_TAG*n*[TAG_MODE] picks the tuple hash, the mask hash, or a combination of the two hashes.

For all packets, the bit-select extractor can override the Tag value (refer to Section 9.4.2. The bit-select extractor is enabled on a per-port basis.

The following C-like pseudo code details precisely how PIP/IPD creates the Tag field:

```
Tag = 0;
if(there is a L2/L1 receive error) {
  // WORD2[RE] will be set, as will WORD2[NI]
  inc_port = true;
}
else if((ISRAWFULL || ISRAWSCH) && (!PKT_INST_HDR[NTAG] || PIP_GBL_CTL[IHMSK_DIS])) {
  inc_port = false;
  Tag = PKT_INST_HDR[TAG];
}
else if(PIP_PRT_CFGB[port][BSEL_EN] && PIP_BSEL_TBL_ENT[bsel_tbl_idx][TAG_EN]) {
 // bsel_tbl_idx = select + PIP_BSEL_EXT_CFG[bsel_num][OFFSET], where
 // bsel_num = PIP_PRT_CFGB[port][BSEL_NUM],
 // select is the bits selected from the packet (using PIP_BSEL_EXT_CFG[bsel_num][SKIP]
 // and PIP_BSEL_EXT_POS[bsel_num])
 TAG<7:0> = PIP_BSEL_TBL_ENT[bsel_tbl_idx][TAG];
 TAG<15:8> = PIP_BSEL_EXT_CFG[bsel_num][TAG];
 inc_port = PIP_PRT_TAG[port][INC_PRT];
 if(!inc_port) TAG<31:16> = PIP_BSEL_EXT_CFG[bsel_num][UPPER_TAG];
}
else {
  inc_port = true;
  if(PIP_PRT_TAG[port][TAG_MODE] == 0)        // always tuple tag
    Tag<15:0> = hw_tuple_tag();
  else if(PIP_PRT_TAG[port][TAG_MODE] == 1)   // always mask tag
    Tag<15:0> = hw_mask_tag();
  else if(PIP_PRT_TAG[port][TAG_MODE] == 2) { // Is_IP ? tuple tag : mask tag
    if(Is_IP) // i.e. is WORD2[9] clear
       Tag<15:0> = hw_tuple_tag();
    else
       Tag<15:0> = hw_mask_tag();
  }
  else                                        // mask XOR tuple
    Tag<15:0> = hw_mask_tag() ^ hw_tuple_tag();
}
if(inc_port) {
  if(PIP_PRT_TAG[port][INC_PRT]) Tag<23:16> = port;
  else Tag<23:16> = 0xFF;
}

uint32 hw_tuple_tag() {
  Tag = 0;
  if(Is_IP) {
    // WORD2[NI] will always be clear here
    if(WORD2[V6])
      Tag = hw_ipv6_hash(port, WORD2, IP source address, IP destination address,
              initial IP next_header, tcp/udp source port, tcp/udp destination port,
               TCP_flag_SYN, TCP_flag_ACK, VLAN0, VLAN1);
    else
      Tag = hw_ipv4_hash(port, WORD2, IP source address, IP destination address,
               IP protocol, tcp/udp source port, tcp/udp destination port,
               TCP_flag_SYN, TCP_flag_ACK, VLAN0, VLAN1);
  }
  return(Tag);
}

uint16 crc16(uint16 iv, uint64 p) {
  int i;
  int j;
  uint   b;
  uint16 poly = 0x1021;

  for (i=7; i>=0; i--) {
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

```
    b = (p >> (i*8)) & 0xff;
    for (j=7; j>=0; j--) {
      iv = ((iv << 1) ^ ((((iv >> 15)&1) ^ ((b >> j)&1)) ? poly : 0))
           & 0xffff;
    }
  }
  return iv;
}

uint16 hw_ipv4_hash(uint8 port, uint64 WORD2, uint32 IPsrc, uint32 IPdest,
                    uint8 protocol, uint16 tcp_udp_sprt, uint16 tcp_udp_dprt,
                  bool TCP_flag_SYN, TCP_flag_ACK,
                  uint16 VLAN0, uint16 VLAN1)
{
  uint16 src_crc  = 0xffff;
  uint16 dst_crc  = 0xffff;
  uint16 prot_crc = 0xffff;
  uint16 result   = 0;

  uint16 src_secret16 = (PIP_TAG_SECRET[SRC]);
  uint32 src_secret32 = (((uint32)src_secret16) << 16) | src_secret16;

  uint16 dst_secret16 = (PIP_TAG_SECRET[DST]);
  uint32 dst_secret32 = (((uint32)dst_secret16) << 16) | dst_secret16;

  // WORD2[NI] will always be clear here
  if(!WORD2[IE]) {
    if(PIP_PRT_TAG[port][IP4_SRC])
      src_crc = crc16(src_crc, ((uint64)IPsrc+src_secret32)<<32);
    if(PIP_PRT_TAG[port][IP4_DST])
      dst_crc = crc16(dst_crc, ((uint64)IPdest+dst_secret32)<<32);
    if (   PIP_PRT_TAG[port][IP4_PCTL]
        || (WORD2[VV] && !WORD2[VS] && PIP_PRT_TAG[port][INC_VLAN])
        || (WORD2[VS] &&               PIP_PRT_TAG[port][INC_VS]  )) {
      uint64 pctl_vlan_dat = 0;
      if(PIP_PRT_TAG[port][IP4_PCTL])
        pctl_vlan_dat |= (((uint64)protocol)<<56);
      if (   (WORD2[VV] && !WORD2[VS] && PIP_PRT_TAG[port][INC_VLAN] )
          || (WORD2[VS] &&               (PIP_PRT_TAG[port][INC_VS]&1))) {
        vlan0 = 0x8000 | VLAN0<11:0>;
        pctl_vlan_dat |= (((uint64)vlan0)<<32);
      }
      if (WORD2[VS] && (PIP_PRT_TAG[port][INC_VS]&2)) {
        vlan1 = 0x8000 | VLAN1<11:0>;
        pctl_vlan_dat |= (((uint64)vlan1)<<16);
      }
      prot_crc = crc16(prot_crc, pctl_vlan_dat);
    }
    if(WORD2[TU] && !WORD2[FR] && !WORD2[LE] && (IPv4.HL==5)) {
      if(PIP_PRT_TAG[port][IP4_SPRT])
        src_crc = crc16(src_crc, ((uint64)tcp_udp_sprt+src_secret16)<<48);
      if(PIP_PRT_TAG[port][IP4_DPRT])
        dst_crc = crc16(dst_crc, ((uint64)tcp_udp_dprt+dst_secret16)<<48);
      if(PIP_GBL_CFG[TAG_SYN] && protocol==TCP
              && TCP_flag_SYN && !TCP_flag_ACK)
        src_crc = 0;
    }
    result = (src_crc ^ dst_crc ^ prot_crc) & ~PIP_TAG_MASK[MASK];
  }
  return(result);
}

uint32 hw_ipv6_hash(uint8 port, uint64 WORD2, uint64 IPsrc[2], uint64 IPdest[2],
                    uint8 next_header, uint16 tcp_udp_sprt, uint16 tcp_udp_dprt,
                    bool TCP_flag_SYN, TCP_flag_ACK,
                    uint16 VLAN0, uint16 VLAN1)
{
  uint16 src_crc  = 0xffff;
  uint16 dst_crc  = 0xffff;
  uint16 prot_crc = 0xffff;
  uint16 result   = 0;

  uint16 src_secret16 = (PIP_TAG_SECRET[SRC]);
```

Cavium Confidential For Steven Yang ALU 02/27/2014

```
  uint32 src_secret32 = (((uint32)src_secret16) << 16) | src_secret16;

  uint16 dst_secret16 = (PIP_TAG_SECRET[DST]);
  uint32 dst_secret32 = ()(uint32)dst_secret16) << 16) | dst_secret16;

  // WORD2[NI] will always be clear here
  if(!WORD2[IE]) {
    if(PIP_PRT_TAG[port][IP6_SRC]) {
      src_crc = crc16(src_crc,    ((IPsrc[0]+src_secret32)&0xffffffff)
                                | (((IPsrc[0]>>32)+src_secret32)<<32));
      src_crc = crc16(src_crc,    ((IPsrc[1]+src_secret32)&0xffffffff)
                                | (((IPsrc[1]>>32)+src_secret32)<<32));
    }
    if(PIP_PRT_TAG[port][IP6_DST]) {
      dst_crc = crc16(dst_crc,    ((IPdest[0]+dst_secret32)&0xffffffff)
                                | (((IPdest[0]>>32)+dst_secret32)<<32));
      dst_crc = crc16(dst_crc,    ((IPdest[1]+dst_secret32)&0xffffffff)
                                | (((IPdest[1]>>32)+dst_secret32)<<32));
    }
    if (   PIP_PRT_TAG[port][IP6_NXTH]
        || (WORD2[VV] && !WORD2[VS] && PIP_PRT_TAG[port][INC_VLAN])
        || (WORD2[VS] &&               PIP_PRT_TAG[port][INC_VS] )) {
      uint64 nxth_vlan_dat = 0;
      if(PIP_PRT_TAG[port][IP6_NXTH])
        nxth_vlan_dat |= (((uint64)next_header)<<56);
      if (   (WORD2[VV] && !WORD2[VS] && PIP_PRT_TAG[port][INC_VLAN] )
          || (WORD2[VS] &&              (PIP_PRT_TAG[port][INC_VS]&1))) {
        vlan0 = 0x8000 | VLAN0<11:0>;
        nxth_vlan_dat |= (((uint64)vlan0)<<32);
      }
      if (WORD2[VS] && (PIP_PRT_TAG[port][INC_VS]&2)) {
        vlan1 = 0x8000 | VLAN1<11:0>;
        nxth_vlan_dat |= (((uint64)vlan1)<<16);
      }
      prot_crc = crc16(prot_crc, nxth_vlan_dat);
    }
    if(WORD2[TU] && !WORD2[FR] && !WORD2[LE]) {
      if(PIP_PRT_TAG[port][IP6_SPRT]) src_crc =
        crc16(src_crc, ((uint64)tcp_udp_sprt+src_secret16)<<48);
      if(PIP_PRT_TAG[port][IP6_DPRT]) dst_crc =
        crc16(dst_crc, ((uint64)tcp_udp_dprt+dst_secret16)<<48);
      if(PIP_GBL_CFG[TAG_SYN] && next_header==TCP
              && TCP_flags_SYN && !TCP_flags_ACK)
        crc_src = 0;
    }
    result = (src_crc ^ dst_crc ^ prot_crc) & ~PIP_TAG_MASK[MASK];
  }
  return(result);
}

// NOTE: pkt_dat is from the first byte - no SKIP is applied
uint16 hw_mask_tag(uint16 pkt_len, uint8* pkt_dat, uint8 port)
{
  uint16 crc    = 0xffff;
  uint16 result = 0;
  int i;
  int j;
  uint16 pkt_cnt = 0;
  for (i=0; i<16; i++) {
    uint8 tag_inc = PIP_TAG_INC[i + (PIP_PRT_CFG[port][TAG_INC]*16)];
    uint8 mod_inc = 0;
    if ((pktLen & 7) == 0) {
      if (pktLen > (i*8)) {
        mod_inc = 0xff;
      }
    }
    else {
      if ( (pktLen>>3) == i )
        mod_inc = (0xff << (8-(pktLen&7)));
      else if ( (pktLen>>3) > i)
        mod_inc = 0xff;
    }
    if((tag_inc & mod_inc) != 0) {
```

Cavium Confidential For Bowen Song ALU 02/27/2014

```
      uint64 dat = 0;
      for (j=7; j>=0; j--, pkt_cnt++)
        dat |= (((tag_inc&mod_inc)>>j)&1) ?
                    ((uint64)pkt_dat[pkt_cnt] << (j*8)) : 0;
      crc16(crc, dat);
    }
    else
      pkt_cnt += 8;
  }
  result = crc & ~PIP_TAG_MASK[MASK];

  return(result);
}
```

WORD3[Back, Size, Addr] Refer to Figure 9–10. Fields in the first buffer pointer. Figure 9–7 details these fields. Back, Size, and Addr are 0 when there is no ordinary DRAM copy of the packet, which happens when the packet is dynamic short and PIP/IPD is enabled to avoid the ordinary L2/DRAM copy.

WORD2[Bufs] Refer to Figure 9–11. The number of buffers used to store the packet data. Any value for this field is possible. The value 0 happens (only) for dynamic short packets when PIP/IPD is enabled to avoid the ordinary L2/DRAM copy, and it indicates that PIP/IPD created no ordinary L2/DRAM of the packet.

WORD2[IP_offset] Note that WORD2[IP_offset] is not present when there is an L2/L1 receive error, nor when the packet is RAWFULL, nor when the packet is not IP.

Refer to Figure 9–11. The number of bytes from the first byte of packet data to the first byte of the IP packet.

WORD2[VV] Note that WORD2[VV] is not present for a RAWFULL packet.

Refer to Figure 9–11. Indicates that one or more DSA/VLAN fields were found in the L2 HDR for the packet. This bit asserts only when in skip-to-L2 mode and the L2 HDR type is Ethernet II + VLAN (possibly stacked) or IEEE 802.3 + VLAN (possibly stacked) or DSA (possibly stacked).

WORD2[VV] will always be set when PIP_PRT_CFG*n*DSA_EN] = 1 and the DMAC is not the MAC CTL address (i.e. is not 01-80-C2-00-00-01).

Also see the VV definition in the pseudo-code in Section 9.2.5 and Figure 9–4. WORD2[VV] is unpredictable when WORD2[RE] = 1.

WORD2[VS] Note that WORD2[VS] is not present for a RAWFULL packet. WORD2[VS] is never set when WORD2[VV] is present and clear.

Refer to Figure 9–11. Indicates that multiple VLAN fields were found in the L2 HDR for the packet. This bit asserts only when in skip-to-L2 mode and the L2 HDR type is Ethernet II + VLAN Stacked or IEEE 802.3 + VLAN Stacked or DSA + VLAN.

Also see the VS definition in the pseudo-code in Section 9.2.5 and Figure 9–4. WORD2[VS] is unpredictable when WORD2[RE] = 1 (and the packet is not RAWFULL).

Cavium Confidential For Bowen Feng ALI 5/27/2014

WORD2[VC] Note that WORD2[VC] is not present for a RAWFULL packet.

Refer to the DSA/VLAN CFI bit (i.e. VLAN<12>) in Figure 9–11.

If WORD2[VS] = 1, PIP_GBL_CTL[VS_WQE] selects which of the two DSA/VLAN fields (VLAN0 or VLAN1, as described in Section 9.2.5 and Figure 9–4) the CFI comes from.

- if PIP_GBL_CTL[VS_WQE] = 0, VLAN0 is selected.
- if PIP_GBL_CTL[VS_WQE] = 1, VLAN1 is selected.

WORD2[VC] is set to the CFI field in the DSA tag when PIP_PRT_CFG*n*[DSA_EN] = 1 and the DMAC is not the MAC CTL address (i.e. is not 01-80-C2-00-00-01) and (WORD2[VS] = 0 or PIP_GBL_CTL[VS_WQE] = 0). Refer to the VLAN0/VLAN1<12> definition in the pseudo-code in Section 9.2.5.

WORD2[VC] is 0 when WORD2[VV] = 0, and is unpredictable when WORD2[RE] = 1 (and the packet is not RAWFULL).

WORD2[VLAN_id] Note that WORD2[VLAN_id] is not present for a RAWFULL packet.

Refer to the VLAN ID field (i.e. VLAN<11:0>) in Figure 9–11.

If WORD2[VS] = 1, PIP_GBL_CTL[VS_WQE] selects which of the two DSA/VLAN fields (VLAN0 or VLAN1, as described in Section 9.2.5 and Figure 9–4) is present here.

- if PIP_GBL_CTL[VS_WQE] = 0, VLAN0 is selected.
- if PIP_GBL_CTL[VS_WQE] = 1, VLAN1 is selected.

WORD2[VLAN_id] is set to the VID field in the DSA tag when PIP_PRT_CFG*n*[DSA_EN] = 1 and the DMAC is not the MAC CTL address (i.e. is not 01-80-C2-00-00-01) and (WORD2[VS] = 0 or PIP_GBL_CTL[VS_WQE] = 0). Refer to the VLAN0/VLAN1<11:0> definition in the pseudo-code in Section 9.2.5.

This field is 0x0 when WORD2[VV] is clear, and is unpredictable when WORD2[RE] is set (and the packet is not RAWFULL).

WORD2[PR] Note that WORD2[PR] is not present for a RAWFULL packet.

Refer to Figure 9–11. The ring identifier for DPI packets when PIP_GBL_CTL[RING_EN] = 1.

When a packet arrives via a DPI input port (i.e. WORD1[iprt] is in the range 32..35) and DPI ring information is enabled in the work-queue entry (i.e. PIP_GBL_CTL[RING_EN] = 1, then WORD2[PR] contains the relative position of the ring in the port. Refer to Section 9.4.

If the packet did not arrive via a DPI input port or PIP_GBL_CTL[RING_EN] = 0, then WORD2[PR] is always zero.

WORD2[PRR] Note that WORD2[PRR] is not present for packets that are not RAWFULL or that have an L2/L1 error.

Refer to Figure 9–11 and to WORD2[PR] description above. The ring identifier for DPI packets when PIP_GBL_CTL[RING_EN] = 1.

If the packet did not arrive via a DPI input port or PIP_GBL_CTL[RING_EN] = 0, then WORD2[PRR] is always PIP_RAW_WORD[WORD<31:28>].

Cavium Confidential For Bowen Yang AU 02/27/2011

WORD2[CO] Note that WORD2[CO] is not present when there is an L2/L1 receive error, nor when the packet is RAWFULL, nor when the packet is not IP.

Refer to Figure 9–11. Set when the IP packet is IPCOMP (i.e. when the IPv4 protocol value or the initial IPv6 next header equals 108). This bit is always clear when WORD2[IE] = 1, and when WORD2[V6] and WORD2[FR] are both set.

WORD2[TU] Note that WORD2[TU] is not present when there is an L2/L1 receive error, nor when the packet is RAWFULL, nor when the packet is not IP.

Refer to Figure 9–11. Set when the IP packet is TCP or UDP (i.e. when the IPv4 protocol value or the initial IPv6 next header equals 6 (TCP) or 17 (UDP)). This bit is always clear when WORD2[IE] = 1, and when WORD2[V6] and WORD2[FR] are both set.

WORD2[SE] Note that WORD2[SE] is not present when there is an L2/L1 receive error, nor when the packet is RAWFULL, nor when the packet is not IP.

Refer to Figure 9–11. Set when the IP packet may require IPSEC decryption. This bit gets set in a number of ways:

- The packet is IPSEC ESP (i.e. the IPv4 protocol value or the initial IPv6 next header equals 50).
- The packet is IPSEC AH (i.e. the IPv4 protocol value or the initial IPv6 next header equals 51).
- The packet is TCP (i.e. the IPv4 protocol value or the initial IPv6 next header equals 6) and the TCP destination port matches one of four possible programmed values and (WORD2[V6] || (IPv4.HL==5)).
- The packet is UDP (i.e. the IPv4 protocol value or the initial IPv6 next header equals 17) and the UDP destination port matches one of four possible programmed values and (WORD2[V6] || (IPv4.HL==5)).

There are four programmable destination port values (PIP_DEC_IPSEC[0..3]) shared by TCP and UDP. Each programmed port can match TCP and/or UDP. WORD2[SE] = 0 whenever any of WORD2[IE], WORD2[FR], or WORD2[LE] are set.

WORD2[V6] Note that WORD2[V6] is not present when there is an L2/L1 receive error, nor when the packet is RAWFULL, nor when the packet is not IP.

Refer to Figure 9–11. This bit is set when the IP version number field is 6.

WORD2[LE] Note that WORD2[LE] is not present when there is an L2/L1 receive error, nor when the packet is RAWFULL, nor when the packet is not IP.

Refer to Figure 9–11. This bit is set when WORD2[TU] is set and the PIP/IPD hardware found an error in the TCP/UDP header and/or data. When this bit is set, WORD2[Opcode] contains one of the following possible (non-zero) codes:

Cavium Confidential For Bowen Yang ALC 02/27/2014

1 = Malformed L4:

IPv4/TCP

- ipv4_total_length < IP_header(20B) + (TCP_data_offset × 4)

  or

- TCP_data_offset < 5

IPv6/TCP

- ipv6_payload_length < TCP_data_offset × 4

  or

- TCP_data_offset < 5

IPv4/UDP

- ipv4_total_length < IP_header(20B) + UDP_header(8B)

IPv6/UDP

- ipv6_payload_length < UDP_header(8B)

2 = L4 checksum error: the L4 checksum value is bad. Note that the hardware checksum calculation complies with section 5 of RFC 1624.

NOTE: For a UDP packet that has a UDP length (specified in the UDP L4 header) that indicates a larger packet than the IP length indicates (specified in the IP L3 header), either a UDP L4 checksum error (WORD2[Opcode] = 2) or a UDP length error (WORD2[Opcode] = 3) is possible.

3 = UDP length error: The UDP length field would make the UDP data longer than what remains in the IP packet (as defined by the IP header length field).

IP/TCP

- Will not assert

IPv4/UDP

- ipv4_total_length < IP_header(20B) + UDP_length

IPv6/UDP

- ipv6_payload_length < UDP_length

NOTE: The hardware does not consider it an error when the UDP length field makes the UDP data shorter than the IP packet.

4 = Bad L4 Port: either the source or destination TCP/UDP port is 0.
8 = TCP FIN Only: the packet is TCP and only the FIN flag set.
9 = TCP No Flags: the packet is TCP and no flags are set.
10 = TCP FIN RST: the packet is TCP and both FIN and RST are set.
11 = TCP SYN URG: the packet is TCP and both SYN and URG are set.
12 = TCP SYN RST: the packet is TCP and both SYN and RST are set.
13 = TCP SYN FIN: the packet is TCP and both SYN and FIN are set.

NOTE: WORD2[LE] is never set when WORD2[IE] or WORD2[FR] are set, and is never set for IPv4 packets when (IPv4.HL ≠ 5).

WORD2[FR] Note that WORD2[FR] is not present when there is an L2/L1 receive error, nor when the packet is RAWFULL, nor when the packet is not IP.

Refer to Figure 9–11. Set when the packet is a fragment.

For IPv4, this bit is set when either the MF bit is set or the offset field is non-zero.

For IPv6, this bit is set when the initial next header value is fragmentation (i.e. 44). WORD2[FR] is never set when WORD2[IE] is set.

Cavium Confidential For Bowen Yang ALU 2/21/2014

WORD2[IE] Note that WORD2[IE] is not present when there is an L2/L1 receive error, nor when the packet is RAWFULL, nor when the packet is not IP.

Refer to Figure 9–11. Set when the packet has an IP exceptional condition. When this bit is set, WORD2[Opcode] contains one of the following possible (non-zero) codes:

1 = Neither IPv4 nor IPv6: the IP version field is neither 4 nor 6.

2 = IPv4 header checksum error: the IPv4 header has a checksum violation. Note that the hardware checksum calculation complies with section 5 of RFC 1624. Also note that the CN71XX hardware does not check IPv4-header checksums for IPv4 packets with options (i.e. HLEN > 5.

3 = IP malformed header: the packet is not long enough to contain the IP header.

IPv6 packet

- received packet length – crc < SKIP+L2_Size+IPv6_header(40 bytes)

not IPv6 packet (IPv4 packet)

- received packet length – crc < SKIP+L2_Size+IPv4_header(20 bytes)

  or

- ipv4_header_length < 5

  or

- ipv4_total_length < 20

Note that CN71XX's IPv4 malformed-header check assumes that the IPv4 header is 20 bytes. This means that for IPv4 packets with options (i.e. HLEN > 5), the hardware may not signal an IP malformed error even though the header size is larger than the IP packet size. Thus, as part of the IPv4 options handler, software should check that HLEN × 4 does not exceed the IP packet size.

4 = IP malformed: the packet is not long enough to contain the bytes indicated by the IP header. Pad is allowed.

NOTE: The IP length can legally indicate fewer bytes than are in the packet.

IPv6 packet

- received packet length – crc < SKIP+L2_Size+IPv6_header(40)+ipv6_payload_length

not IPv6 packet (IPv4 packet)

- received packet length – crc < SKIP+L2_Size+IPv4_total_length

5 = IP TTL hop: the IPv4 TTL field or the IPv6 hop count field are zero.

6 = IP options:

For IPv4, the packet has options (i.e. the IP HLEN field is not 5. Note that the case where HLEN < 5 is flagged in the IP malformed-header check). Also note that the CN71XX hardware does not check header checksums of IPv4 packets with options and only partially performs an IP malformed-header check.

Upon receipt of an IPv4 packet with an IP-options exception, software should perform an IPv4-header checksum check, and check that HLEN × 4 does not exceed the IP length indicated in the IPv4 header.

Cavium Confidential For Bowen Yang ALU 02/27/2014

For IPv6, the packet has early extension headers.
When PIP_GBL_CTL[IP6_EEXT<1>] = 1, this violation occurs when the (initial) next header field in the IPv6 header does not contain one of the following values:

- 6 (TCP)
- 17 (UDP)
- 44 (fragmentation)
- 58 (ICMP)
- 50 (ESP)
- 51 (AH)
- 108 (IPCMP)

When PIP_GBL_CTL[IP6_EEXT<1>] = 0 and PIP_GBL_CTL[IP6_EEXT<0>] = 1, this violation occurs when the (initial) next header field in the IPv6 header contains one of the following values:

- 0 (hop-by-hop)
- 60 (destination)
- 43 (routing)

WORD2[B] Note that WORD2[B] is not present when there is not an L2/L1 receive error and the packet is RAWFULL.

Refer to Figure 9–11. Set when the packet's destination MAC address field in the L2 HDR is the broadcast address (i.e. all 1s). This bit is unpredictable when WORD2[RE] is set, and always clear when not in skip-to-L2 mode.

WORD2[M] Note that WORD2[M] is not present when there is not an L2/L1 receive error and the packet is RAWFULL.

Refer to Figure 9–11. Set when the packet's destination MAC address field in the L2 HDR is a multicast address (i.e. the group bit is set, and at least one of the remaining bits is a zero). This bit is unpredictable when WORD2[RE] is set, and always clear when not in skip-to-L2 mode.

WORD2[NI] Note that WORD2[NI] is not present when the packet is RAWFULL.

Refer to Figure 9–11. Set when the packet is not an IP packet or has L2/L1 errors.

WORD2[RE] Note that WORD2[RE] is not present when there is not an L2/L1 receive error and the packet is RAWFULL.

Refer to Figure 9–11. Set when the packet had an L2/L1 receive error. These errors are generated by PIP/IPD (refer to Section 9.2.7) or directly by the packet interfaces (GMX—refer to Chapter 23, RGMII—refer to Chapter 25), PCIe interfaces (Chapter 22), or PKO internal loopback.

When WORD2[RE] is set, WORD2[Opcode] contains one of the following possible (non-zero) codes.

1 = partial error: the packet was partially received. If the packet arrived via a packet interface, internal buffering/bandwidth was not adequate to receive the entire packet. If the packet arrived via PKO internal loopback, the packet was aborted by a higher priority packet. Not applicable when the packet arrived via the DPI interface.

2 = jabber error: the packet was too large and is truncated. Applicable only to packet interface ports.

Cavium Confidential For Bowen Yang ALU 02/27/2014

3 = overrun error: the packet is longer than allowed and had an FCS error.

4 = oversize error: the packet is longer than allowed.

5 = alignment error: the RGMII packet is not an integer number of bytes and had an FCS error (100M and 10M only).

6 = fragment error: the packet is shorter than allowed and had an FCS error.

7 = FCS error: if the packet arrived via a packet interface, the packet had an FCS error. If the packet arrived via PKO internal loopback, the packet had one or more parity errors. Not applicable when the packet arrived via the DPI interface.

8 = undersize error: the packet is shorter than allowed.

9 = terminate error: the packet was terminated with an idle cycle instead of a terminate cycle. Applicable only to packet interface ports in XAUI mode.

10 = length mismatch error: the packet had a length that did not match the length field in the L2 HDR.

11 = GMX/AGL RX error: the packet had one or more data reception errors in which a control byte was detected in the frame. Applicable only to GMX/AGL ports.

12 = GMX/AGL skip error: the packet was not large enough to cover the skipped bytes. Applicable only to packet interface ports.

13 = RGMII nibble error: A RGMII packet had a stutter error (data not repeated, 10/100M only).

15 = DMA packet error: the packet experienced one or more errors during the DMA read operation.

17 = PIP Skip error: a packet was not large enough to cover the skipped bytes.

18 = L2 header malformed: the packet is not long enough to contain the L2 header.

47 = PUNY error: the packet was 4B or less when FCS stripping is enabled (i.e. when IPD_SUB_PORT_FCS[PORT_BIT<port>] is set for the port that the packet arrived on). PIP/IPD disables the FCS strip for the PUNY packet, so WORD1[Len] will always be between one and four in this case.

WORD2[Opcode] Note that WORD2[Opcode] is not present when there is not an L2/L1 receive error and the packet is RAWFULL.

Normally zero, but contains a (non-zero) exception opcode when any one of WORD2[RE], WORD2[IE], or WORD2[LE] are set. For exception opcode values, refer to WORD2[RE], WORD2[IE], and WORD2[LE].

WORD2[IR] Note that WORD2[IR] is not present when there is not an L2/L1 receive error and the packet is RAWFULL or IP.

Refer to Figure 9–11. Set when the packet's L2 HDR TYPE field indicates RARP (i.e. equals 0x0835). This bit is unpredictable when WORD2[RE] is set, and always clear when not in skip-to-L2 mode.

WORD2[IA] Note that WORD2[IA] is not present when there is not an L2/L1 receive error and the packet is RAWFULL or IP.

Refer to Figure 9–11. Set when the packet's L2 HDR TYPE field indicates ARP (i.e. equals 0x0806). This bit is unpredictable when WORD2[RE] is set, and always clear when not in skip-to-L2 mode.

Cavium Confidential For Bowen Yang ALU 02/27/2014

9.6 Input Packet Data Unit (IPD) Quality of Service

There are many facets to the packet input quality of service capabilities of CN71XX in general. This section and the next focus on CN71XX's backpressure and packet drop capabilities. The PIP/IPD unit implements important components of CN71XX's backpressure and packet drop capabilities.

First, when there are no buffers available for PIP/IPD from the hardware pool 0 (i.e. when IPD_QUE0_FREE_PAGE_CNT and/or FPA_QUE(0..7)_AVAILABLE are almost zero) or from the FPA pool selected by IPD_WQE_FPA_QUEUE (i.e. the configured pool that PIP/IPD allocates work-queue entries from), the PIP/IPD hardware stops accepting input packet data from any of SGMII or DPI sources, until buffers become available in the hardware free pool. This may cause the packet-interface logic to drop packets indiscriminately and/or backpressure the ports on the interfaces, and will cause the DPI interface hardware to stop fetching instructions and input packet data for the four DPI input ports.

In many systems, it will be desirable to avoid this buffer exhaustion situation. There are multiple CN71XX mechanisms that can keep the input packets from exhausting the available buffer space.

One mechanism is per-port backpressure. This can prevent the packet from arriving at PIP/IPD, and, thus, may prevent CN71XX from dropping packets. The PIP/IPD hardware maintains a counter per port that PIP/IPD may increment when PIP/IPD buffers packets. The counters are the following:

- IPD_PORT_BP_COUNTERS_PAIR(0..3,16..19,24,32..35)
- IPD_PORT_BP_COUNTERS2_PAIR(36..39)

These three registers are referred to as IPD_PORT_BP_COUNTERS*_PAIR*n* collectively.

When a packet arrives and PIP/IPD creates a work-queue entry for the packet, PIP/IPD adds to the counter for a port according to the following table:

IPD_CTL_STATUS		
[NADDBUF]	[ADDPKT]	Add to Counter
0	0	WORD2[Bufs]
0	1	WORD2[Bufs] + 1
1	0	Undefined
1	1	1

Thus, the per-port counter can simply track FPA pool 0 buffers used, packets arrived, or the total of both. Note that when IPD_CTL_STATUS[ADDPKT] is clear, IPD_PORT_BP_COUNTERS*_PAIR*n* counts only buffers and not packets, and PIP/IPD does not change IPD_PORT_BP_COUNTERS*_PAIR*n* for dynamic short packets that are solely buffered in the work-queue entry.

When IPD_PORT_BP_COUNTERS*_PAIR*n* exceeds the threshold for the port (IPD_PORT(0..3,16..19,24,32..35)_BP_PAGE_CNT[PAGE_CNT] or IPD_PORT(36..39)_BP_PAGE_CNT2[PAGE_CNT] is the threshold, collectively referred to as IPD_PORT*n*_BP_PAGE_CNT*[PAGE_CNT]), the threshold check is active, and PIP/IPD can backpressure the port when backpressure is enabled for the port. A 0→1 transition on IPD_CTL_STATUS[PBP_EN] is necessary to enable per-port backpressure. Each port has an individual backpressure enable – IPD_PORT*n*_BP_PAGE_CNT*[BP_ENB].

Cavium Confidential For Bowen Yang ALU 02/27/2014

When backpressure is both enabled and active for a port, the hardware attempts to prevent packet arrival on the port. If the port is a DPI port, CN71XX prevents its DPI interface hardware from fetching packet data for the port. (Note that CN71XX also contains logic to backpressure individual DPI rings. Refer to Section 12.1.4.)

In addition to or alternative to per-port backpressure, the PIP/IPD unit can also drop packets whenever the counter for the port (i.e. IPD_PORT_BP_COUNTERS*_PAIR*n*) exceeds the threshold for the port. (IPD_PORT*n*_BP_PAGE_CNT*[PAGE_CNT] is also the counter threshold for this packet drop mechanism.) Though the per-port backpressure and this per-port drop mechanism share a counter and a threshold, the per-port drop mechanism can be enabled independently of per-port backpressure. A 0→1 transition on IPD_CTL_STATUS[PBP_EN] is required to enable per-port packet drop, as it is for per-port backpressure, but there is a separate per-port enable for per-port packet drop (IPD_BP_PRT_RED_END[PRT_ENB*n*]).

Note that when PIP_PRT_CFG*n*[RAWDRP] is clear, RAWFULL and RAWSCH packets are never dropped by this per-port drop mechanism, but that when PIP_PRT_CFG*n*[RAWDRP] is set, RAWFULL and RAWSCH packets are treated as other packets.

For the per-port backpressure and/or per-port packet drop mechanisms (that use IPD_PORT_BP_COUNTERS*_PAIR*n* and the IPD_PORT*n*_BP_PAGE_CNT*[PAGE_CNT] threshold) to work correctly when enabled, the core software must eventually decrement IPD_PORT_BP_COUNTERS*_PAIR*n* after a packet and/or its buffers should no longer be assigned to the port, and the port is free to receive more packets. When core software writes the IPD_SUB_PORT_BP_PAGE_CNT register with the appropriate values (2's complement of the amount to decrement), PIP/IPD decrements IPD_PORT_BP_COUNTERS*_PAIR*n* for a port. The simplest scheme would be to decrement the counter for a port once the core software first gets the work that PIP/IPD creates for the packet, but this is not required and many other options are possible. Each decrement can be millions, so core software is free to decrement the port backpressure counter for many input packets using only a single CSR write.

The final mechanism that CN71XX provides to keep from exhausting the pool 0 buffers is IPD's per-QOS admission control. If desired, when the FPA pool 0 buffers get low (i.e. when IPD_QUE0_FREE_PAGE_CNT and/or FPA_QUE(0..7)_AVAILABLE get low), PIP/IPD can drop packets. The different QOS levels can be configured with different thresholds on the FPA pool 0 buffer count — packets with some QOS values can drop sooner than other packets with different QOS values. PIP/IPD hardware does not allocate buffers for, nor writes any data into, buffers for packets that it chooses to drop by either the per-QOS admission control or by the per-port packet drop mechanism. The details of the per-QOS admission control algorithm are described in the next section.

Cavium Confidential For Beijen Yang 2014-5-4

9.7 PIP/IPD Per-QOS Admission Control

As Section 9.5 describes, PIP/IPD uses input port, configuration variables, and packet parsing to select one of eight QOS levels. It then tags the work-queue entry for the packet with the QOS level (assuming the packet is not dropped) before sending the work to POW. PIP/IPD normally uses the exact same calculation to produce a QOS value to differentiate and prioritize packets for admission control, but the admission control QOS can be different than the value specified for the work-queue entry when the packet has one of the following:

- an L2/L1 receive error (WORD2[RE] = 1)
- an IP error (WORD2[IE] = 1)
- a TCP/UDP error (WORD2[LE] = 1)

The two QOS calculations are slightly different since PIP/IPD performs the admission-control check before the entire packet is received, and checksums and other errors cannot be checked before the hardware completes the admission control calculations. The admission control QOS logic assumes that these error conditions are not present when it calculates the QOS value.

The following pseudo-code details the PIP/IPD admission control QOS selection for a given packet:

```
bool not_present_or_in_last_pip_tick(uint _skip, uint _pkt_len, uint _byte_num) {
  uint pkt_accumulate = ((_skip % 8) ? (8-(_skip % 8)) : 0);
  uint last_tick = (pkt_accumulate + ((_pkt_len % 8) ? (_pkt_len % 8) : 8));
  return (_pkt_len - last_tick) < _byte_num;
}

QOS = PIP_PRT_CFG[port][QOS];
if (there is an L1/L2 receive error and the packet is <= 136 bytes) {
  ;
}
else if((ISRAWFULL || ISRAWSCH) && (!PKT_INST_HDR[NQOS] ||PIP_GBL_CTL[IHMSK_DIS])) {
  QOS = PKT_INST_HDR[QOS];
}
else if(PIP_PRT_CFGB[port][BSEL_EN] && PIP_BSEL_TBL_ENT[bsel_tbl_idx][QOS_EN]) {
 // bsel_tbl_idx = select + PIP_BSEL_EXT_CFG[bsel_num][OFFSET], where
 // bsel_num = PIP_PRT_CFGB[port][BSEL_NUM],
 // select is the bits selected from the packet (using PIP_BSEL_EXT_CFG[bsel_num][SKIP]
 // and PIP_BSEL_EXT_POS[bsel_num]) - selected bits are 0 when not present in the packet
 QOS = PIP_BSEL_TBL_ENT[bsel_tbl_idx][QOS];
}
else {
  if(PIP_PRT_CFG[port][HG_QOS]) {
   QOS = HG_QOS_TABLE[HG_PRI<5:0>]; // HG_QOS_TABLE has the PIP_HG_PRI_QOS[QOS] values
  }
  if(VV && PIP_PRT_CFG[port][QOS_VLAN]) {
    VLAN = (VS && PIP_GBL_CFG[VS_QOS]) ? VLAN1 : VLAN0;
    if(PIP_PRT_CFG[port][QOS_VSEL])
      QOS = PIP_QOS_VLAN[VLAN<15:13>][QOS1]; // VLAN<15:13> is priority
    else
      QOS = PIP_QOS_VLAN[VLAN<15:13>][QOS ]; // VLAN<15:13> is priority
  }
  if(Is_IP && ((IP.version==4) || (IP.version==6)) &&
     (IP.version is present in packet)) {
    if(PIP_PRT_CFG[port][QOS_DIFF] && !(VV && PIP_PRT_CFG[port][QOS_VLAN]
      && PIP_PRT_CFG[port][QOS_VOD]))) {
      use_diffserv = !not_present_or_in_last_pip_tick(l2_skip, pkt_len,
                      IP.TOS/CLASS<7:2>);
      tos_class = use_diffserv ? IP.TOS/CLASS<7:2> : 0;
      QOS = PIP_QOS_DIFF[tos_class][QOS]; // the 6 msbs of TOS/CLASS }
    }
  }
  for (i=0; i<8; i++) {
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

```
    if (PIP_PRT_CFG[port][QOS_WAT<i>] and PIP_QOS_WATCH[i][TYPE]!=0) {
      mask16 = ~PIP_QOS_WATCH[i][MASK] & 0xffff;
      mask8 = ~PIP_QOS_WATCH[i][MASK] & 0xff;
      match16 = PIP_QOS_WATCH[i][MATCH] & mask16;
      match8 = PIP_QOS_WATCH[i][MATCH] & mask8;
      protnh = IP protocol/next header field is present in packet ?
                    IP protocol/next header field : 0;
      if(ethertype_val and
           PIP_QOS_WATCH[i][TYPE] is ETHER_TYPE and
           match16 equals (ethertype)&mask16 and
           ethertype field is present in packet) {
        QOS = PIP_QOS_WATCH[i][WATCHER];
        break;
      }
      else if (packet is IP malformed and
          not_present_or_in_last_pip_tick(l2_skip, pkt_len, PROT/NH) and
          PIP_QOS_WATCH[i][TYPE] is protocol/next_header) {
        QOS = UNPREDICTABLE;
        break;
      }
      else if(PIP_QOS_WATCH[i][TYPE] is protocol/next_header and
           match8 equals protnh&mask8) {
        QOS = PIP_QOS_WATCH[i][WATCHER];
        break;
      }
      else if(PIP_QOS_WATCH[i][TYPE] is TCP and protnh is TCP and
           match16 equals (TCP destination port field)&mask16 and
           ((packet is IPv6) or ((IPv4.HL==5) && !IPv4.MF &&
           !IPv4.offset)) and
           TCP destination port field is present in packet) {
        QOS = PIP_QOS_WATCH[i][WATCHER];
        break;
      }
      else if(PIP_QOS_WATCH[i][TYPE] is UDP and protnh is UDP and
           match16 equals (UDP destination port field)&mask16 and
           ((packet is IPv6) or ((IPv4.HL==5) && !IPv4.MF &&
           !IPv4.offset)) and
           UDP destination port field is present in packet) {
        QOS = PIP_QOS_WATCH[i][WATCHER];
        break;
      }
    }
  }
}
```

Each of the eight QOS levels has:

- Independent pass and drop marks for RED-like probabilistic drop scheme
- Independent mixing to calculate a moving average of IPD_QUE0_FREE_PAGE_CNT (i.e. the number of buffers available in FPA pool 0).

Figure 9–13 shows how PIP/IPD calculates the packet-drop probability for a QOS level using the configured values IPD_QOS(0..7)_RED_MARKS[PASS] and IPD_QOS*n*_RED_MARKS[DROP]. (IPD_RED_QUE(0..7)_PARAM[PRB_CON] must also be set properly for correct operation.) The actual hardware calculates the drop probability for the eight QOS levels only periodically and at a granularity of 1/256. PIP/IPD recalculates the drop probability once every 8 × (IPD_RED_PORT_ENABLE[PRB_DLY] + 68) clock cycles, using the moving average of IPD_QUE0_FREE_PAGE_CNT for the QOS level at that time. IPD_RED_PORT_ENABLE[PRB_DLY] should typically be set to 0x0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Upon packet arrival, whenever the current moving average for the QOS level is between the pass and drop marks, PIP/IPD compares the precalculated drop probability to a pseudo-random number to determine whether to drop a packet due to per-QOS admission control. The precalculated drop probability is not used by PIP/IPD whenever the current moving average is below the drop or above the pass mark.

PIP/IPD also recalculates the moving average of IPD_QUE0_FREE_PAGE_CNT (equivalently, FPA_QUE*n*_AVAILABLE) once every 8 × (IPD_RED_PORT_ENABLE[AVG_DLY]+10) cycles:

avg = ((avg × IPD_RED_QUE*n*_PARAM[AVG_CON]) + (IPD_QUE0_FREE_PAGE_CNT × IPD_RED_QUE*n*_PARAM[NEW_CON])) / 256

where:

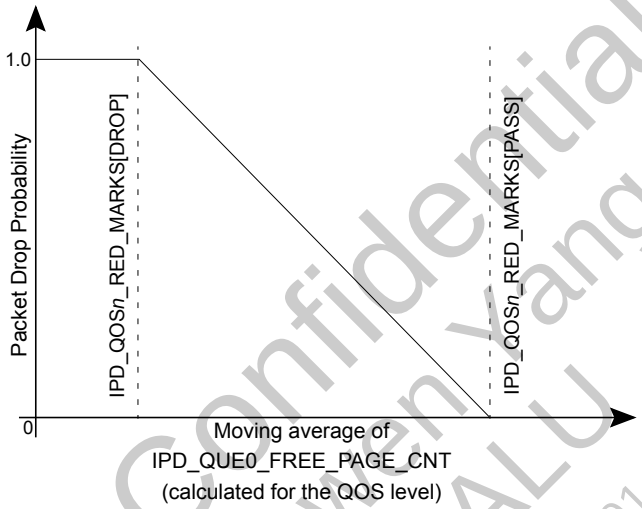
avg = moving average of IPD_QUE0_FREE_PAGE_CNT for the QOS level

IPD_RED_QUE*n*_PARAM[AVG_CON] = AVG_CON config for the QOS level

IPD_RED_QUE*n*_PARAM[NEW_CON] = NEW_CON config for the QOS level

IPD_QUE0_FREE_PAGE_CNT = number of buffers in FPA pool 0.

Figure 9–13 PIP/IPD Packet-Drop Probability



The moving average for a QOS level tracks changes to IPD_QUE0_FREE_PAGE_CNT more closely when AVG_CON for the QOS level is smaller, and less closely when AVG_CON is larger. (AVG_CON and NEW_CON must total to 256.) Note that the fractional part of **avg** is ignored, and this can lead to an inherent error of as much as 256/(256-AVG_CON) in the **avg** calculation.

Note that there are two methods to make the moving average track the actual average less closely:

- Larger IPD_RED_PORT_ENABLE[AVG_DLY] values cause the moving averages of ALL QOS levels to track less closely
- Larger AVG_CON values cause the moving averages of individual QOS levels to track less closely

The **avg** equation above assumes that IPD_RED_QUE(0..7)_PARAM[USE_PCNT] = 0.

Cavium Confidential For Bowen Yang ALU 9/27/2014

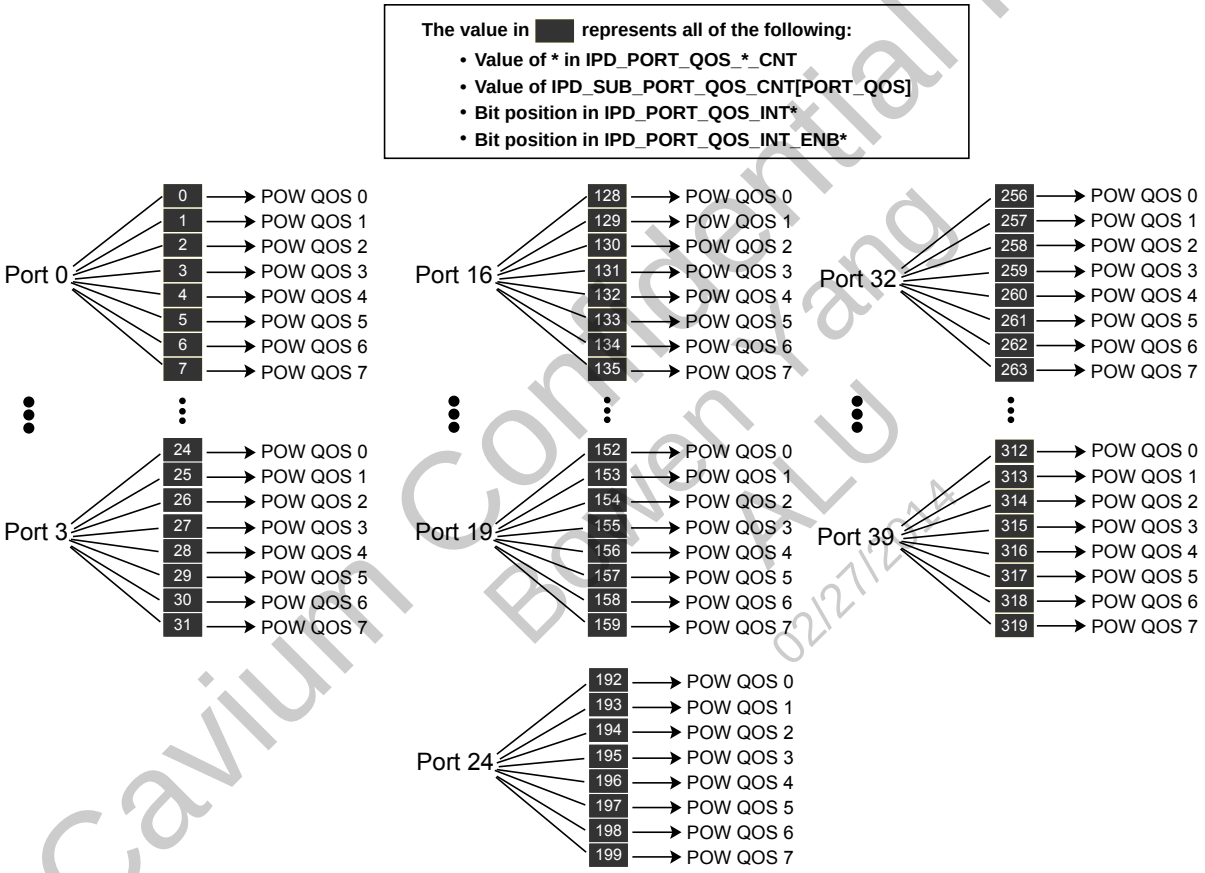
If IPD_RED_QUE*n*_PARAM[USE_PCNT] = 1, **avg** for the selected QOS level is simply a periodic sample of IPD_QUE0_FREE_PAGE_CNT (assuming AVG_CON and NEW_CON total to 256).

RAWFULL and RAWSCH packets are never dropped due to per-QOS admission control when PIP_PRT_CFG*n*[RAWDRP] is clear. Otherwise, admission control is enabled on an individual port basis via IPD_RED_PORT_ENABLE[PRT_ENB<*n*>]/ IPD_RED_PORT_ENABLE2[PRT_ENB<*n*>].

9.8 Per-Port and QOS Threshold Interrupts

The PIP/IPD hardware maintains a counter per port/QOS (IPD_PORT_QOS_(0..31,128..159,192..199,256..319)_CNT that PIP/IPD increments when it sends a packet to POW. The port/QOS value is a combination of the port number (up to 16) and POW QOS level (eight levels). PIP/IPD maintains a total of 128 counters, corresponding interrupt bits, corresponding interrupt enables, and corresponding watermarks, one for each POW QOS level and port combination. Each port/QOS combination is assigned a value as defined in Figure 9–14.

Figure 9–14 Port/QOS Combination Values



Cavium Confidential For Broadcom and RLU 02/27/2014

The PIP/IPD hardware increments the corresponding counter once per POW work-queue-entry creation. PIP/IPD adds to the counter for a port/QOS according to the following table:

IPD_CTL_STATUS		
[PQ_NABUF]	[PQ_APKT]	Add to Counter
0	0	WORD2[Bufs]
0	1	WORD2[Bufs] + 1
1	0	Undefined
1	1	1

When hardware increments the counter, the corresponding interrupt bit is set when IPD_PORT_QOS_(0..31,128..159,192..199,256..319)_CNT[WMARK] is less than or equal to the new counter value (after the add) and greater than the old counter value. The software nominally decrements a counter by writing the IPD_SUB_PORT_QOS_CNT CSR. The corresponding interrupt bit is set when IPD_PORT_QOS_(0..31,128..159,192..199,256..319)_CNT[WMARK] is greater than or equal to the new counter value (after the “subtract”) and less than the old counter value. Software may change the threshold to switch between high watermark and low watermark. (A threshold value update does not cause the interrupt bit to be set.)

For example, when the hardware increments a port/QOS counter to equal or exceed the watermark threshold, the corresponding interrupt bit is set (IPD_PORT_QOS_INT(0,2,3,4). Software may decide to reduce the watermark at that point while it changes processing so that the port/QOS counter drains. When the software later decrements the port/QOS counter below the watermark, the corresponding interrupt bit is set, which may cause software to increase the watermark to its original value and continue normal processing.

Table 9–7 shows a mapping of the Port/QOS combination values to the bit position for IPD_PORT_QOS_INT(0,2,3,4)[INT] and IPD_PORT_QOS_INT_ENB(0,2,3,4)[ENB].

Table 9–7 Port/QOS Value Mapping to [INT] and [ENB] Bits

Register	Port #	Port/QOS Bits	[INT]/[ENB] Bits
IPD_PORT_QOS_INT0	0	0–7	0–7
	1	8–15	8–15
	2	16–23	16–23
	3	24–31	24–31
IPD_PORT_QOS_INT2	16	128–135	0–7
	17	136–143	8–15
	18	144–151	16–23
	19	152–159	24–31
IPD_PORT_QOS_INT3	24	192–199	0–7
IPD_PORT_QOS_INT4	32	256–263	0–7
	33	264–271	8–15
	34	272–279	16–23
	35	280–287	24–31
	36	288–295	32–39
	37	296–303	40–47
	38	304–311	48–55
	39	312–319	56–63

Cavium Confidential For Bowen Yang 02/7/2014

Note that PIP/IPD increments the counter for a packet at the point when it starts writing the work-queue entry to L2/DRAM, which is before the point when it adds the work-queue entry to POW. It is possible, though perhaps unlikely, for the software handler for the PIP/IPD per-port and QOS interrupt to execute before the interrupt-causing work-queue entry is present in POW. System software may need to account for this.

Software can read the current values of the counters (IPD_PORT_QOS_(0..31,128..159,192..199,256..319)_CNT) at any point.

PIP/IPD asserts an interrupt wire whenever any interrupt bit and its corresponding enable bit is set. The interrupt wire generated is sent from PIP/IPD to CIU. Refer to the following fields in registers in Chapter 13:

- CIU_INT(0..7,32)_SUM0[IPDPPTHR]
- CIU_INT(0..3)_SUM4[IPDPPTHR]
- CIU_INT(0..7,32)_EN0[IPDPPTHR]
- CIU_INT(0..3)_EN4_0[IPDPPTHR]

Cavium Confidential For Bowen Yang ALU 02/27/2014

9.9 PIP Registers

All the PIP registers are 64-bits long (eight bytes). The PIP registers are shown in Table 9–8.

Table 9–8 PIP Registers

Register	Address[1]	CSR Type[2]	Detailed Description
PIP_BIST_STATUS	0x00011800A0000000	RSL	See page 509
PIP_INT_REG	0x00011800A0000008	RSL	See page 509
PIP_INT_EN	0x00011800A0000010	RSL	See page 510
PIP_STAT_CTL	0x00011800A0000018	RSL	See page 510
PIP_GBL_CTL	0x00011800A0000020	RSL	See page 511
PIP_GBL_CFG	0x00011800A0000028	RSL	See page 513
PIP_SFT_RST	0x00011800A0000030	RSL	See page 513
PIP_BCK_PRS	0x00011800A0000038	RSL	See page 514
PIP_CLKEN	0x00011800A0000040	RSL	See page 514
PIP_IP_OFFSET	0x00011800A0000060	RSL	See page 514
PIP_TAG_SECRET	0x00011800A0000068	RSL	See page 515
PIP_TAG_MASK	0x00011800A0000070	RSL	See page 515
PIP_TODO_ENTRY	0x00011800A0000078	RSL	See page 516
PIP_DEC_IPSEC(0..3)	0x00011800A0000080 + α*0x8	RSL	See page 516
PIP_RAW_WORD	0x00011800A00000B0	RSL	See page 516
PIP_QOS_VLAN(0..7)	0x00011800A00000C0 + α*0x8	RSL	See page 516
PIP_QOS_WATCH(0..7)	0x00011800A0000100 + α*0x8	RSL	See page 517
PIP_FRM_LEN_CHK0	0x00011800A0000180	RSL	See page 517
PIP_DSA_SRC_GRP	0x00011800A0000190	RSL	See page 518
PIP_DSA_VID_GRP	0x00011800A0000198	RSL	See page 518
PIP_HG_PRI_QOS	0x00011800A00001A0	RSL	See page 519
PIP_VLAN_ETYPES(0..1)	0x00011800A00001C0 + α*0x8	RSL	See page 519
PIP_PRT_CFG(0..3,16..19,24,32..39)	0x00011800A0000200 + α*0x8	RSL	See page 520
PIP_PRT_TAG(0..3,16..19,24,32..39)	0x00011800A0000400 + α*0x8	RSL	See page 522
PIP_QOS_DIFF(0..63)	0x00011800A0000600 + α*0x8	RSL	See page 523
PIP_STAT0_PRT(0..3,16..19,24,32..39)	0x00011800A0000800 + α*0x50	RSL	See page 528
PIP_STAT1_PRT(0..3,16..19,24,32..39)	0x00011800A0000808 + α*0x50	RSL	See page 528
PIP_STAT2_PRT(0..3,16..19,24,32..39)	0x00011800A0000810 + α*0x50	RSL	See page 528
PIP_STAT3_PRT(0..3,16..19,24,32..39)	0x00011800A0000818 + α*0x50	RSL	See page 529
PIP_STAT4_PRT(0..3,16..19,24,32..39)	0x00011800A0000820 + α*0x50	RSL	See page 529
PIP_STAT5_PRT(0..3,16..19,24,32..39)	0x00011800A0000828 + α*0x50	RSL	See page 529
PIP_STAT6_PRT(0..3,16..19,24,32..39)	0x00011800A0000830 + α*0x50	RSL	See page 529
PIP_STAT7_PRT(0..3,16..19,24,32..39)	0x00011800A0000838 + α*0x50	RSL	See page 529
PIP_STAT8_PRT(0..3,16..19,24,32..39)	0x00011800A0000840 + α*0x50	RSL	See page 530
PIP_STAT9_PRT(0..3,16..19,24,32..39)	0x00011800A0000848 + α*0x50	RSL	See page 530
PIP_STAT10_PRT(0..3,16..19,24,32..39)	0x00011800A0001480 + α*0x10	RSL	See page 530
PIP_STAT11_PRT(0..3,16..19,24,32..39)	0x00011800A0001488 + α*0x10	RSL	See page 531
PIP_TAG_INC(0..63)	0x00011800A0001800 + α*0x8	RSL	See page 523
PIP_STAT_INB_PKTS(0..3,16..19,24,32..39)	0x00011800A0001A00 + α*0x20	RSL	See page 532
PIP_STAT_INB_OCTS(0..3,16..19,24,32..39)	0x00011800A0001A08 + α*0x20	RSL	See page 532
PIP_STAT_INB_ERRS(0..3,16..19,24,32..39)	0x00011800A0001A10 + α*0x20	RSL	See page 532
PIP_BSEL_EXT_CFG(0..3)	0x00011800A0002800 + α*0x10	RSL	See page 524
PIP_BSEL_EXT_POS(0..3)	0x00011800A0002808 + α*0x10	RSL	See page 525
PIP_ALT_SKIP_CFG(0..3)	0x00011800A0002A00 + α*0x8	RSL	See page 526
PIP_BSEL_TBL_ENT(0..511)	0x00011800A0003000 + α*0x8	RSL	See page 526
PIP_PRT_CFGB(0..3,16..19,24,32..39)	0x00011800A0008000 + α*0x8	RSL	See page 527

1. For registers with multiple versions, α represents the values within the specified range.
2. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential PDF
Erwin Yang 6/27/2014

PIP BIST Results Register
PIP_BIST_STATUS

BIST status register. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:20>	—	RAZ	—	—	Reserved
<19:0>	BIST	RO	0x0	0x0	BIST results. Hardware sets a bit in BIST for memory that fails BIST.

PIP Interrupt Register
PIP_INT_REG

This register captures any exception event that occurs. It sets the exception bit regardless of the value of PIP_INT_EN. PIP_INT_EN only controls if an interrupt is raised to software. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	—	—	Reserved
<12>	PUNYERR	R/W1C/H	0	0	Frame was received with length ≤ 4B when CRC stripping in IPD is enabled.
<11>	LENERR	R/W1C/H	0	0	Frame was received with length error.
<10>	MAXERR	R/W1C/H	0	0	Frame was received with length > max_length.
<9>	MINERR	R/W1C/H	0	0	Frame was received with length < min_length.
<8>	BEPERR	R/W1C/H	0	0	Parity error in back-end memory
<7>	FEPERR	R/W1C/H	0	0	Parity error in front-end memory
<6>	TODOOVR	R/W1C/H	0	0	To-do list overflow.
<5>	SKPRUNT	R/W1C/H	0	0	Skip runt packets. Packet was engulfed by skipper. This interrupt can occur with received PARTIAL packets that are truncated to SKIP bytes or smaller.
<4>	BADTAG	R/W1C/H	0	0	A bad tag was sent from IPD.
<3>	PRTNXA	R/W1C/H	0	0	Nonexistent port
<2>	BCKPRS	R/W1C/H	0	0	PIP asserted backpressure.
<1>	CRCERR	R/W1C/H	0	0	PIP calculated bad CRC.
<0>	PKTDRP	R/W1C/H	0	0	Packet dropped due to QOS.

Notes:

- TODOOVR: The PIP to-do list stores packets that have been received and require work-queue-entry generation. PIP normally asserts backpressure when the list fills up such that any error normally is a result of programming PIP_BCK_PRS[HIWATER] incorrectly. PIP itself can handle 29M packets/s @ 500 MHz or 15 Gb/s @ 64B packets.
- SKPRUNT: If a [packet size] < [amount programmed in the per-port skippers], there is nothing to parse and the entire packet is skipped over. This is probably not the desired effect, so there is an indication to software.
- BADTAG: A tag is considered bad when it is resumed by a new packet before it was released by PIP. PIP considers a tag released by one of two methods.
  - QOS dropped so that it is released over the **pip_ipd_release** bus.
  - Work-queue entry is validated by the **pip_ipd_done** signal.
- PRTNXA: CN71XX supports ports 0–3, 16–19, 24, 32–39. If PIP receives a packet that is not in the range, the address processed is mapped into the valid address space (the mapping is currently unpredictable) and the bit is set.

Cavium Confidential For Bowen Yang ALU 02/27/2014

For packet ports (0–31), PRTNXA is asserted for packets received on ports 2–15, 20–23, 25–31 in SGMII mode or on ports 1–15, 17–23, 25–31 in RXAUI mode.

For upper (DPI and loopback) ports (32–63), PRTNXA is asserted for packets received on ports 40–47 and 52–63.

- BCKPRS: PIP can assert backpressure to the receive logic when the to-do list exceeds a high-water mark (see PIP_BCK_PRS for more details). When this occurs, PIP can raise an interrupt to software.
- CRCERR: CN71XX can compute CRC in two places. Each GMX port computes its own CRC, but PIP can provide an additional check or check loopback or PCIe ports. If PIP computes a bad CRC, PIP raises an interrupt.
- PKTDRP: PIP can drop packets based on QOS results received from IPD. If the QOS algorithm decides to drop a packet, PIP asserts an interrupt.

PIP Interrupt Enable Register
PIP_INT_EN

This register determines if hardware should raise an interrupt to software when an exception event occurs.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	—	—	Reserved
<12>	PUNYERR	R/W	0	0	Frame was received with length ≤ 4B when CRC stripping in IPD is enabled.
<11>	LENERR	R/W	0	0	Frame was received with length error.
<10>	MAXERR	R/W	0	0	Frame was received with length > max_length.
<9>	MINERR	R/W	0	0	Frame was received with length < min_length.
<8>	BEPERR	R/W	0	0	Parity error in back end memory
<7>	FEPERR	R/W	0	0	Parity error in front end memory
<6>	TODOOVR	R/W	0	0	To-do list overflow.
<5>	SKPRUNT	R/W	0	0	Packet was engulfed by skipper.
<4>	BADTAG	R/W	0	0	A bad tag was sent from IPD.
<3>	PRTNXA	R/W	0	0	Nonexistent port
<2>	BCKPRS	R/W	0	0	PIP asserted backpressure.
<1>	CRCERR	R/W	0	0	PIP calculated bad CRC.
<0>	PKTDRP	R/W	0	0	Packet dropped due to QOS.

PIP Stat Control Register
PIP_STAT_CTL

This register controls how the PIP statistics counters are handled. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
63:1>	—	RAZ	—	—	Reserved
<0>	RDCLR	R/W	1	1	Stat registers are read and clear: 0 = stat registers hold value when read 1 = stat registers are cleared when read

Cavium Confidential For Brown Yang ALU 02/27/2014

PIP Global Control Register
PIP_GBL_CTL

This register contains global control information. These are the enable signals global-checker for IPv4/IPv6 and TCP/UDP parsing. The enable signals affect all ports. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:28>	—	RAZ	—	—	Reserved
<27>	IHMSK_DIS	R/W	0	0	Instruction Header Mask Disable. 0 = Allow NTAG, NTT, NGRP, NQOS bits in the instruction header to control which fields from the instruction header are used for work-queue entry WORD2. 1 = Ignore the NTAG,NTT,NGRP,NQOS bits in the instruction header and act as if these fields were zero. Thus always use the TAG,TT,GRP,QOS (depending on the instruction header length) from the instruction header for the work-queue entry WORD2.
<26>	DSA_GRP_TVID	R/W	0	0	DSA group algorithm. Use the DSA source ID to compute GRP.
<25>	DSA_GRP_SCMD	R/W	0	0	DSA group algorithm. Use the DSA source ID to compute GRP when the DSA tag command to TO_CPU.
<24>	DSA_GRP_SID	R/W	0	0	DSA group algorithm. Use the DSA VLAN ID to compute GRP.
<23:21>	—	RAZ	—	—	Reserved
<20>	RING_EN	R/W	0	1	Enable DPI ring information in the work-queue entry.
<19:17>	SPARE	R/W	0x0	0x0	Spare bits.
<16>	IGNRS	R/W	0	0	Ignore RS bit. When set to 1, ignore PKT_INST_HDR[RS]. Does not apply to DPI ports (32–35). When using 2-byte instruction header words, either PIP_PRT_CFG[DYN_RS] or IGNRS should be set.
<15>	VS_WQE	R/W	0	0	Indicates which DSA/VLAN CFI and ID to use when VLAN stacking: 0 = use 1st VLAN (network order), 1 = use 2nd VLAN (network order)
<14>	VS_QOS	R/W	0	0	Indicates which DSA/VLAN priority to use when VLAN stacking: 0 = use 1st VLAN (network order), 1 = use 2nd VLAN (network order)
<13>	L2_MAL	R/W	1	1	Enable L2 malformed packet check.
<12>	TCP_FLAG	R/W	1	1	Enable TCP flags checks.
<11>	L4_LEN	R/W	1	1	Enable TCP/UDP length check.
<10>	L4_CHK	R/W	1	1	Enable TCP/UDP checksum check.
<9>	L4_PRT	R/W	1	1	Enable TCP/UDP illegal port check.
<8>	L4_MAL	R/W	1	1	Enable TCP/UDP malformed packet check.
<7:6>	—	RAZ	—	—	Reserved
<5:4>	IP6_EEXT	R/W	0x1	0x3	Enable IPv6 early extension headers.
<3>	IP4_OPTS	R/W	1	1	Enable IPv4 options check.
<2>	IP_HOP	R/W	1	1	Enable TTL (IPv4) / hop (IPv6) check.
<1>	IP_MAL	R/W	1	1	Enable malformed check.
<0>	IP_CHK	R/W	1	1	Enable IPv4 header checksum check.

The following text describes the conditions in which each checker asserts and flags an exception. When the checker is disabled, the exception is not flagged and the packet is parsed as best it can be parsed.

Cavium Confidential For Boren Wu Kang 2017-2019

NOTE: By disabling conditions, packets can be parsed incorrectly (i.e. IP_MAL and L4_MAL could cause bits to be seen in the wrong place. IP_CHK and L4_CHK mean that the packet was corrupted).

- TCP_FLAG: Indicates any of the following conditions:
  - {URG, ACK, PSH, RST, SYN, FIN}: tcp_flag
  - 6'b000001 = (FIN only)
  - 6'b000000 = (0)
  - 6'bxxx1x1 = (RST+FIN+*)
  - 6'b1xxx1x = (URG+SYN+*)
  - 6'bxxx11x = (RST+SYN+*)
  - 6'bxxxx11 = (SYN+FIN+*)
- L4_LEN: Indicates that the TCP or UDP length does not match the IP length.
- L4_CHK: Indicates that a packet classified as either TCP or UDP contains an L4 checksum failure.
- L4_PRT: Indicates that a TCP or UDP packet has an illegal port number – either the source or destination port is 0.
- L4_MAL: Indicates that a TCP or UDP packet is not long enough to cover the TCP or UDP header.
- IP6_EEXT: Indicates the presence of IPv6 early extension headers. These bits only apply to packets classified as IPv6. Bit 0 flags early extensions when next_header is any one of the following:
  - Hop-by-hop (0)
  - Destination (60)
  - Routing (43)

  Bit 1 flags early extensions when next_header is not any of the following:
  - TCP (6)
  - UDP (17)
  - Fragmentation (44)
  - ICMP (58)
  - IPSEC ESP (50)
  - IPSEC AH (51)
  - IPCOMP
- IP4_OPTS: Indicates the presence of IPv4 options. It is set when the length ≠ 5. This only applies to packets classified as IPv4.
- IP_HOP: Indicates that the IPv4 TTL field or IPv6 HOP field is 0.
- IP_MAL: Indicates that the packet was malformed. Malformed packets are defined as packets that are not long enough to cover the IP header or not long enough to cover the length in the IP header.
- IP_CHK: Indicates that an IPv4 packet contained an IPv4 header checksum violations. Only applies to packets classified as IPv4.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PIP Global Config Register
PIP_GBL_CFG

This register provides global configuration information that applies to all ports. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:19>	—	RAZ	—	—	Reserved
<18>	TAG_SYN	R/W	0	0	Do not include src_crc for TCP/SYN&!ACK packets 0 = include src_crc 1 = tag hash is dst_crc for TCP/SYN&!ACK packets
<17>	IP6_UDP	R/W	1	1	IPv6/UDP checksum is not optional 0 = Allow optional checksum code 1 = Do not allow optional checksum code
<16>	MAX_L2	R/W	0	0	Config bit to choose the largest L2 frame size Chooses the value of the L2 Type/Length field to classify the frame as length. 0 = 1500 / 0x5DC 1 = 1535 / 0x5FF
<15:11>	—	RAZ	—	—	Reserved
<10:8>	RAW_SHF	R/W	0x0	0x0	RAW packet shift amount number of bytes to pad a RAW packet.
<7:3>	—	RAZ	—	—	Reserved
<2:0>	NIP_SHF	R/W	0x0	0x0	Non-IP shift amount. Number of bytes to pad a packet that has been classified as not IP.

Note:

- IP6_UDP: IPv4 allows an optional UDP checksum by sending the all-0s patterns. IPv6 outlaws this and the spec says to always check UDP checksum. This mode bit allows the user to treat IPv6 as IPv4, meaning that the all-0s pattern will cause a UDP checksum pass.

PIP Soft Reset Register
PIP_SFT_RST

Allows soft reset. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	RST	R/W	0	0	Reset. When set to 1 by software, PIP gets a short reset pulse (three cycles in duration).

This bit resets much of PIP's internal state. The following CSRs are not reset:

- PIP_BIST_STATUS
- PIP_STAT*n*_PRT*m*
- PIP_STAT_INB_PKTS*n*
- PIP_STAT_INB_OCTS*n*
- PIP_STAT_INB_ERRS*n*
- PIP_TAG_INC*n*

Cavium Confidential For Bowen Yang ALU 6/27/2014

PIP Back Pressure Register
PIP_BCK_PRS

This register indicates when PIP is asserting backpressure. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	BCKPRS	RO	0	0	PIP is currently asserting backpressure to IOB. Backpressure from PIP asserts when the entries to the todo list exceed HIWATER. Backpressure is held until the todo list ≤ LOWATER.
<62:13>	—	RAZ	0	—	Reserved
<12:8>	HIWATER	R/W	0x18	0x18	Water mark in the todo list to assert backpressure. **NOTE:** This field must not be modified from its reset value.
<7:5>	—	RAZ	0	—	Reserved
<4:0>	LOWATER	R/W	0x4	0x4	Water mark in the todo list to release backpressure. **NOTE:** This field must not be modified from its reset value.

PIP Clock Enable Register
PIP_CLKEN

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	0	—	Reserved
<0>	CLKEN	R/W	0	0	Controls the conditional clocking within PIP 0 = Allow hardware to control the clocks. 1 = Force the clocks to be always on.

PIP IP Offset Into the Work-Queue-Entry Register
PIP_IP_OFFSET

This register specifies the eight-byte offset to find the start of the IP header in the data portion of IP work-queue entries. OFFSET is restricted in that the entire IP and TCP/UDP header must be buffered by hardware. In general, OFFSET must be set in the 0–4 range. If the system restricts all IPv6 packets, the full range of 0–7 can be used if desired. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<2:0>	OFFSET	R/W	0x0	0x0	Byte offset. The number of eight-byte ticks to include in the work-queue entry prior to IP data. 0x0 = 0 bytes (IP data starts at WORD4 of work-queue entry) 0x1 = 8 bytes (IP data starts at WORD5 of work-queue entry) 0x2 = 16 bytes (IP data starts at WORD6 of work-queue entry) 0x3 = 24 bytes (IP data starts at WORD7 of work-queue entry) 0x4 = 32 bytes (IP data starts at WORD8 of work-queue entry) 0x5 = 40 bytes (IP data starts at WORD9 of work-queue entry) 0x6 = 48 bytes (IP data starts at WORD10 of work-queue entry) 0x7 = 56 bytes (IP data starts at WORD11 of work-queue entry) In normal configurations, OFFSET must be set in the 0–4 range to allow the entire IP and TCP/UDP headers to be buffered in hardware and calculate the L4 checksum for TCP/UDP packets. The maximum value of OFFSET is determined by the types of packets that can be sent to PIP as follows: **Packet Type — MAX OFFSET** IPv4/TCP/UDP — 7 IPv6/TCP/UDP — 5 IPv6/without L4 parsing — 6 If the L4 can be ignored, the maximum value of OFFSET for IPv6 packets can increase to 6. Here are the following programming restrictions for IPv6 packets and OFFSET = 6: PIP_GBL_CTL[TCP_FLAG] = 0 PIP_GBL_CTL[L4_LEN] = 0 PIP_GBL_CTL[L4_CHK] = 0 PIP_GBL_CTL[L4_PRT] = 0 PIP_GBL_CTL[L4_MAL] = 0 PIP_DEC_IPSEC*n*[TCP] = 0 PIP_DEC_IPSEC*n*[UDP] = 0 PIP_PRT_TAG*n*[IP6_DPRT] = 0 PIP_PRT_TAG*n*[IP6_SPRT] = 0 PIP_GBL_CTL[TCP6_TAG] = 0 PIP_PRT_TAG*n*[TAG_SYN] = 0

PIP Initial-Value Register
PIP_TAG_SECRET

The source and destination initial values (IVs) in tag generation provide a mechanism for each CN71XX to be unique. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:16>	DST	R/W	0x0	0x0	Secret for the destination tuple tag CRC calculation.
<15:0>	SRC	R/W	0x0	0x0	Secret for the source tuple tag CRC calculation.

PIP Mask-Bit Register
PIP_TAG_MASK

Provides the mask-bit field in tag generation. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	MASK	R/W	0x0	0x0	When set, clears the individual bits of the lower 16 bits of the computed tag. Does not affect RAW or INST_HDR packets.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PIP To-Do List Entry Register
PIP_TODO_ENTRY

Head entry of the to-do list (debug only). Summary of the current packet that has completed and is waiting to be processed. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	VAL	RO/H	—	—	Entry is valid.
<62>	—	RAZ	—	—	Reserved
<61:0>	ENTRY	RO/H	—	—	To-do list entry summary.

PIP UDP/TCP Ports to Watch for DEC IPSEC Registers
PIP_DEC_IPSEC(0..3)

PIP sets the dec_ipsec based on TCP or UDP destination port. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:18>	—	RAZ	—	—	Reserved
<17>	TCP	R/W	0	0	This DPRT should be used for TCP packets.
<16>	UDP	R/W	0	0	This DPRT should be used for UDP packets.
<15:0>	DPRT	R/W	0x0	0x0	UDP or TCP destination port to match on.

PIP RAW Word2 of the Work-Queue-Entry Register
PIP_RAW_WORD

The RAW Word2 to be inserted into the work-queue entry of RAWFULL packets. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:56>	—	RAZ	—	—	Reserved
<55:0>	WORD	R/W	0x0	0x0	Word2 of the workQ entry. The 8-bit bufs field is still set by hardware (IPD)

PIP QOS VLAN Tables Registers
PIP_QOS_VLAN(0..7)

If the PIP indentifies that a packet is DSA/VLAN tagged, the QOS can be set based on the DSA/VLAN user priority. These eight registers comprise the QOS values for all DSA/VLAN user priority values. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved
<6:4>	QOS1	R/W	0x0	0x0	DSA/VLAN QOS level. Selected when PIP_PRT_CFG*n*[QOS_VSEL] = 1.
<3>	—	RAZ	—	—	Reserved
<2:0>	QOS	R/W	0x0	0x0	DSA/VLAN QOS value. Selected when PIP_PRT_CFG*n*[QOS_VSEL] = 0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PIP QOS Watcher-Table Registers
PIP_QOS_WATCH(0..7)

Sets up the configuration CSRs for the four QOS watchers. Each watcher can be set to look for a specific protocol, Ethertype, or TCP/UDP destination port to override the default QOS value. Note that the watchers have overlapping criteria. See Table 9–8 for address

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:32>	MASK	R/W	0x0	0x0	Mask off a range of values.
<31:28>	—	RAZ	—	—	Reserved
<27:24>	GRP	R/W	0x0	0x0	The group number of the watcher.
<23>	—	RAZ	—	—	Reserved
<22:20>	WATCHER	R/W	0x0	0x0	The QOS level of the watcher.
<19>	—	RAZ	—	—	Reserved
<18:16>	TYPE	R/W	0x0	0x0	The field for the watcher match against: 0x0 = disable across all ports 0x1 = protocol (IPv4) or next_header (IPv6) 0x2 = TCP destination port 0x3 = UDP destination port 0x4 = Ethertype 0x5–7 = Reserved
<15:0>	MATCH	R/W	0x0	0x0	The value to watch for.

PIP Frame Length Check Register
PIP_FRM_LEN_CHK0

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
63:32>	—	RAZ	—	—	Reserved
<31:16>	MAXLEN	R/W	0x600	0x600	Byte count for max-sized frame check. PIP_PRT_CFG*n*[MAXERR_EN] enables the check for port *n*. If enabled, failing packets set the MAXERR interrupt and work-queue entry WORD2[opcode] is set to OVER_FCS (0x3, if packet has bad FCS) or OVER_ERR (0x4, if packet has good FCS). The effective MAXLEN used by hardware is as follows: If PIP_PRT_CFG*n*[DSA_EN] is 0, PIP_FRM_LEN_CHK*n*[MAXLEN] + (4 × VV) + (4 × VS) If PIP_PRT_CFG*n*[DSA_EN] is 1, PIP_FRM_LEN_CHK*n*[MAXLEN] + PIP_PRT_CFG*n*[SKIP] + (4 × VS) If PTP_MODE, the 8-byte timestamp is prepended to the packet. MAXLEN should be increased by 8 to compensate for the additional timestamp field.
<15:0>	MINLEN	R/W	0x40	0x40	Byte count for min-sized frame check. PIP_PRT_CFG*n*[MINERR_EN] enables the check for port *n*. If enabled, failing packets set the MINERR interrupt and work-queue entry WORD2[opcode] is set to UNDER_FCS (0x6, if packet has bad FCS) or UNDER_ERR (0x8, if packet has good FCS). If PTP_MODE, the 8-byte timestamp is prepended to the packet. MINLEN should be increased by 8 to compensate for the additional timestamp field.

Cavium Confidential For Bowen Yang

PIP DSA Source Group Register
PIP_DSA_SRC_GRP

Use the DSA source ID to compute GRP. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:60>	MAP15	R/W	0x0	0x0	DSA group algorithm.
<59:56>	MAP14	R/W	0x0	0x0	DSA group algorithm.
<55:52>	MAP13	R/W	0x0	0x0	DSA group algorithm.
<51:48>	MAP12	R/W	0x0	0x0	DSA group algorithm.
<47:44>	MAP11	R/W	0x0	0x0	DSA group algorithm.
<43:40>	MAP10	R/W	0x0	0x0	DSA group algorithm.
<39:36>	MAP9	R/W	0x0	0x0	DSA group algorithm.
<35:32>	MAP8	R/W	0x0	0x0	DSA group algorithm.
<31:28>	MAP7	R/W	0x0	0x0	DSA group algorithm.
<27:24>	MAP6	R/W	0x0	0x0	DSA group algorithm.
<23:20>	MAP5	R/W	0x0	0x0	DSA group algorithm.
<19:16>	MAP4	R/W	0x0	0x0	DSA group algorithm.
<15:12>	MAP3	R/W	0x0	0x0	DSA group algorithm.
<11:8>	MAP2	R/W	0x0	0x0	DSA group algorithm.
<7:4>	MAP1	R/W	0x0	0x0	DSA group algorithm.
<3:0>	MAP0	R/W	0x0	0x0	DSA group algorithm.

PIP DSA VID Group Register
PIP_DSA_VID_GRP

Use the DSA source ID to compute GRP. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:60>	MAP15	R/W	0x0	0x0	DSA group algorithm.
<59:56>	MAP14	R/W	0x0	0x0	DSA group algorithm.
<55:52>	MAP13	R/W	0x0	0x0	DSA group algorithm.
<51:48>	MAP12	R/W	0x0	0x0	DSA group algorithm.
<47:44>	MAP11	R/W	0x0	0x0	DSA group algorithm.
<43:40>	MAP10	R/W	0x0	0x0	DSA group algorithm.
<39:36>	MAP9	R/W	0x0	0x0	DSA group algorithm.
<35:32>	MAP8	R/W	0x0	0x0	DSA group algorithm.
<31:28>	MAP7	R/W	0x0	0x0	DSA group algorithm.
<27:24>	MAP6	R/W	0x0	0x0	DSA group algorithm.
<23:20>	MAP5	R/W	0x0	0x0	DSA group algorithm.
<19:16>	MAP4	R/W	0x0	0x0	DSA group algorithm.
<15:12>	MAP3	R/W	0x0	0x0	DSA group algorithm.
<11:8>	MAP2	R/W	0x0	0x0	DSA group algorithm.
<7:4>	MAP1	R/W	0x0	0x0	DSA group algorithm.
<3:0>	MAP0	R/W	0x0	0x0	DSA group algorithm.

Cavium Confidential For Baoqun Yang ALU 02/27/2014

PIP HiGig Priority QOS Register
PIP_HG_PRI_QOS

This register controls accesses to the HG_QOS_TABLE. To write an entry of the table, write PIP_HG_PRI_QOS with [PRI] as the table address, [QOS] as the priority level, and [UP_QOS] = 1. To read an entry of the table, write PIP_HG_PRI_QOS with [PRI] as the table address, QOS as “don’t-care” priority level, [UP_QOS] = 0, and then read PIP_HG_PRI_QOS. The table data will be in PIP_HG_PRI_QOS[QOS]. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
63:13>	—	RAZ	—	—	Reserved
<12>	UP_QOS	RAZ	0	—	When set, updates the entry in the HG_QOS_TABLE as specified by PIP_HG_PRI_QOS[PRI] to a value of QOS as follows: HG_QOS_TABLE[PRI] = QOS.
<11>	—	RAZ	—	—	Reserved
<10:8>	QOS	R/W/H	0x0	—	QOS Map level to priority.
<7:6>	—	RAZ	—	—	Reserved
<5:0>	PRI	R/W	0x0	—	The priority level from HiGig header: HiGig/HiGig+ PRI = {1'b0, CNG[1:0], COS[2:0]} HiGig2 PRI = {DP[1:0], TC[3:0]}

PIP VLAN Ethertypes Registers
PIP_VLAN_ETYPES(0..1)

These registers specify ethertypes that will be parsed as containing VLAN information. Each TYPE is orthogonal; if all eight are not required, set multiple TYPEs to the same value (as in the 0x8100 default value). See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	TYPE3	R/W	0x8100	0x8100	VLAN Ethertype (type 3)
<47:32>	TYPE2	R/W	0x8100	0x8100	VLAN Ethertype (type 2)
<31:16>	TYPE1	R/W	0x8100	0x8100	VLAN Ethertype (type 1)
<15:0>	TYPE0	R/W	0x8100	0x8100	VLAN Ethertype (type 0)

Cavium Confidential For Bowen Yang ALU 02/27/2014

PIP Per-Port Configuration Registers
PIP_PRT_CFG(0..3,16..19,24,32..39)

This register contains per-port configuration information. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:53>	—	RAZ	—	—	Reserved
<52>	PAD_LEN	R/W	0	0	When set, disables the length check for packets with padding in the client data.
<51>	VLAN_LEN	R/W	0	0	When set, disables the length check for DSA/VLAN packets.
<50>	LENERR_EN	R/W	1	1	L2 length error check enable. Frame was received with length error. This check is typically not enabled for incoming packets on the DPI ports because the CRC bytes may not normally be present.
<49>	MAXERR_EN	R/W	1	1	Max frame error check enable. Frame was received with length > max_length. max_length is defined by PIP_FRM_LEN_CHK0[MAXLEN].
<48>	MINERR_EN	R/W	1	1	Min frame error check enable. Frame was received with length < min_length. This check is typically not enabled for incoming packets on the DPI ports because the CRC bytes may not normally be present. min_length is defined by PIP_FRM_LEN_CHK0[MINLEN].
<47:44>	GRP_WAT_47	R/W	0x0	0x0	GRP watcher enable. Provides an enable bit for each watcher for each of four ports. When set to 1, GRP number in PIP_QOS_WATCH(0..7)[GRP] is used when the GRP watcher matches for the packet. (watchers 4–7)
<43:40>	QOS_WAT_47	R/W	0x0	0x0	QOS watcher enable. Provides an enable bit for each of four watchers for each port. When set to 1, QOS level in PIP_QOS_WATCH(0..7)[WATCHER] is used when the QOS watcher matches for the packet.(watchers 4–7)
<39:37>	—	RAZ	—	—	Reserved
<36>	RAWDRP	R/W	0	0	Allow RAW packet drop. 0 = never drop packets that PIP indicates are RAW 1 = allow the IPD to drop RAW packets based on RED algorithm
<35:34>	TAG_INC	R/W	0x0	0x0	Tag include. Specifies which of the four groups of PIP_TAG_INC(0..63) registers to use when calculating the mask tag hash 0x0 = registers 0–15 0x1 = registers 16–31 0x2 =registers 32–47 0x3 = registers 48–63
<33>	DYN_RS	R/W	0	0	Dynamic RS. Dynamically calculate RS based on packet size and configuration. If DYN_RS is set, PKT_INST_HDR[RS] is not used. When using 2-byte instruction header words, either DYN_RS or PIP_GBL_CTL[IGNRS] should be set.
<32>	INST_HDR	R/W	0	0	INST header. When set, the eight-byte INST_HDR is present on all packets. Internally set for RAWFULL/RAWSCHED packets on the DPI ports (32–35). Internally cleared for all other packets on the DPI ports (32–35). Must be 0 in DSA mode.
<31:28>	GRP_WAT	R/W	0x0	0x0	GRP watcher enable. Provides an enable bit for each watcher for each of four ports. When set to 1, GRP number in PIP_QOS_WATCH(0..7)[GRP] is used when the GRP watcher matches for the packet. (watchers 0–3)
<27>	HG_QOS	R/W	0	0	When set, uses the HiGig priority bits as a lookup into the HG_QOS_TABLE (PIP_HG_PRI_QOS) to determine the QOS value. This field must not be set when HIGIG_EN = 0.
<26:24>	QOS	R/W	0x0	0x0	Default QOS level of the port

Cavium Confidential For Power Yang ALC 8/5/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<23:20>	QOS_WAT	R/W	0x0	0x0	QOS watcher enable. Provides an enable bit for each of four watchers for each port. When set to 1, QOS level in PIP_QOS_WATCH(0..7)[WATCHER] is used when the QOS watcher matches for the packet. (watchers 0–3)
<19>	QOS_VSEL	R/W	0	0	Which QOS in PIP_QOS_VLAN to use. 0 = PIP_QOS_VLAN(0..7)[QOS] 1 = PIP_QOS_VLAN(0..7)[QOS1]
<18>	QOS_VOD	R/W	0	0	QOS VLAN over Diffserv. If DSA/VLAN exists, it is used, else, if IP exists, Diffserv is used, else, the per-port default is used. Watchers are still highest priority.
<17>	QOS_DIFF	R/W	0	0	QOS Diffserv
<16>	QOS_VLAN	R/W	0	0	QOS VLAN
<15:13>	—	RAZ	—	—	Reserved
<12>	CRC_EN	RO	0	0	CRC Checking enabled.
<11>	HIGIG_EN	R/W	0	0	Enable HiGig parsing. This field should not be set for DPI ports (ports 32–35). This field should not be set for ports in which PTP_MODE = 1. When this field = 1: • DSA_EN must be 0. • PIP_PRT_CFGB(0..3,16..19,24,32..39)[ALT_SKP_EN] must be 0. • SKIP is both Skip I size and the size of the HiGig* header (12 or 16 bytes).
<10>	DSA_EN	R/W	0	0	Enable DSA tag parsing. This field should not be set for ports in which PTP_MODE = 1. When DSA_EN = 1: • HIGIG_EN must be 0. • SKIP is the size of DSA tag (4, 8, or 12 bytes) rather than the size of Skip I. Total SKIP (Skip I + header + Skip II must be zero. • INST_HDR must be 0 (non-DPI ports). • PIP_PRT_CFGB(0..3,16..19,24,32..39)[ALT_SKP_EN] must be 0. • For DPI ports, SLI_PKT*_INSTR_HEADER[USE_IHDR] and DPI_INST_HDR[R] should be clear. • MODE must be “skip to L2.”
<9:8>	MODE	R/W	0x0	0x0	Parse mode 0x0 = no packet inspection 0x1 = L2 parsing/skip-to-L2 0x2 = IP parsing / skip-to-L3 0x3 = Illegal Must be 0x2 (“skip to L2”) when in DSA mode.
<7>	—	RAZ	—	—	Reserved
<6:0>	SKIP	R/W	0x0	0x0	Optional SKIP I amount for packets. Hardware forces SKIP to 0x0 for packets on DPI ports (32–35) when a PKT_INST_HDR is present. See PIP_PRT_CFGB(0..3,16..19,24,32..39)[ALT_SKP*] and PIP_ALT_SKIP_CFG(0..3). See Section 9.2.3, Parse Mode and Skip Selection and Section 9.2.8, Legal SKIP Values for further details. In DSA mode, indicates the DSA header length, not SKIP I size. (Must be 4, 8, or 12). In HIGIG mode, indicates both the SKIP I size and the HiGig header size (must be 12 or 16). If PTP_MODE, the 8-byte timestamp is prepended to the packet. SKIP should be increased by 8 to compensate for the additional timestamp field.

Cavium Confidential For Bowen Yang 02-27-2014

PIP Per-Port Tag Configuration Registers
PIP_PRT_TAG(0..3,16..19,24,32..39)

This register contains per-port tag configuration information. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:40>	—	RAZ	—	—	Reserved
<39:36>	GRPTAGBASE	R/W	0x0	0x0	When GRPTAG = 1, specifies the offset to use to compute the work-queue entry group from tag bits.
<35:32>	GRPTAGMASK	R/W	0x0	0x0	When GRPTAG = 1, specifies which of the least-significant bits of the work-queue entry Tag field to exclude from the computation.
<31>	GRPTAG	R/W	0	0	Use work-queue entry tag. Enables the use of the least-significant bits of the work-queue entry Tag field to determine the work-queue entry group. group = (WORD2[Tag<3:0>] AND $\overline{\text{GRPTAGMASK}}$) + GRPTAGBASE For more information on the work-queue entry, refer to Section 9.5.
<30>	GRPTAG_MSKIP	R/W	0	0	When set, GRPTAG is used regardless if the packet IS_IP.
<29:28>	TAG_MODE	R/W	0x0	0x0	Specifies the tag algorithm to use 0x0 = always use tuple tag algorithm 0x1 = always use mask tag algorithm 0x2 = use tuple if IP, else use mask 0x3 = tuple XOR mask
<27:26>	INC_VS	R/W	0x0	0x0	Specifies the DSA/VLAN ID (VID) to be included in the tuple tag generation when VLAN stacking is detected 0x0 = do not include VID 0x1 = include VID (VLAN0) in hash 0x2 = include VID (VLAN1) in hash 0x3 = include VID ({VLAN0,VLAN1)} in hash
<25>	INC_VLAN	R/W	0	0	Specifies whether the DSA/VID is included in the tuple tag generation when VLAN stacking is not detected: 0 = do not include VID, 1 = include VID in hash
<24>	INC_PRT	R/W	0	0	Indicates whether the port is included in tag.
<23>	IP6_DPRT	R/W	0	0	Indicates whether the TCP/UDP dst port is included in tuple tag for IPv6 packets.
<22>	IP4_DPRT	R/W	0	0	Indicates whether the TCP/UDP dst port is included in tuple tag for IPv4 packets.
<21>	IP6_SPRT	R/W	0	0	Indicates whether the TCP/UDP src port is included in tuple tag for IPv6 packets
<20>	IP4_SPRT	R/W	0	0	Indicates whether the TCP/UDP src port is included in tuple tag for IPv4 packets.
<19>	IP6_NXTH	R/W	0	0	Indicates whether IPv6 includes next header in tuple tag hash
<18>	IP4_PCTL	R/W	0	0	Indicates whether IPv4 includes protocol in tuple tag hash
<17>	IP6_DST	R/W	0	0	Indicates whether IPv6 includes dst address in tuple tag hash
<16>	IP4_DST	R/W	0	0	Indicates whether IPv4 includes dst address in tuple tag hash
<15>	IP6_SRC	R/W	0	0	Indicates whether IPv6 includes src address in tuple tag hash
<14>	IP4_SRC	R/W	0	0	Indicates whether IPv4 includes src address in tuple tag hash
<13:12>	TCP6_TAG	R/W	0x0	0x0	Sets the tag_type of a TCP packet (IPv6) 0x0 = ordered tags 0x1 = atomic tags 0x2 = null tags 0x3 = reserved

Cavium Confidential For Review Only 02/7/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<11:10>	TCP4_TAG	R/W	0x0	0x0	Sets the tag_type of a TCP packet (IPv4) 0x0 = ordered tags 0x1 = atomic tags 0x2 = null tags 0x3 = reserved
<9:8>	IP6_TAG	R/W	0x0	0x0	Sets whether IPv6 packet tag type 0x0 = ordered tags 0x1 = atomic tags 0x2 = null tags 0x3 = reserved
<7:6>	IP4_TAG	R/W	0x0	0x0	Sets whether IPv4 packet tag type 0x0 = ordered tags 0x1 = atomic tags 0x2 = null tags 0x3 = reserved
<5:4>	NON_TAG	R/W	0x0	0x0	Sets whether non-IP packet tag type 0x0 = ordered tags 0x1 = atomic tags 0x2 = null tags 0x3 = reserved
<3:0>	GRP	R/W	0x0	0x0	Core group number. Specifies the core group (0–15) to use for the work.

PIP QOS Diffserv Tables Registers
PIP_QOS_DIFF(0..63)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved
<2:0>	QOS	R/W	0x0	0x0	Diffserv QOS level. Specifies the Diffserv QOS level.

PIP Include Registers
PIP_TAG_INC(0..63)

Specifies which bytes to include in the new tag hash algorithm. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	EN	R/W	0x0	0x0	Indicates which bytes to include in the mask tag algorithm. It is broken into four 16-entry masks to cover 128 bytes. The PIP_PRT_CFG(0..3,16..19,24,32..39)[TAG_INC] field selects which of the four: if PIP_PRT_CFG*n*[TAG_INC] = 0x0, use registers 0–15 if PIP_PRT_CFG*n*[TAG_INC] = 0x1, use registers 16–31 if PIP_PRT_CFG*n*[TAG_INC] = 0x2, use registers 32–47 if PIP_PRT_CFG*n*[TAG_INC] = 0x3, use registers 48–63 Bit[7] corresponds to the MSB and bit[0] corresponds to the LSB of the 8-byte word. If PTP_MODE, the 8-byte timestamp is prepended to the packet. The EN byte masks should be adjusted to compensate for the additional timestamp field.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PIP Bit-Select Extractor Configuration Registers
PIP_BSEL_EXT_CFG(0..3)

These registers set the tag, offset, and skip values to be used when using the corresponding bit-select extractor (packet parsing algorithm). See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:56>	—	RAZ	—	—	Reserved
<55:40>	UPPER_TAG	R/W	0x0	—	Extra tag bits to be added to tag field from the table. Only included when PIP_PRT_TAG[INC_PRT]=0. WORD2[TAG<31:16>].
<39:32>	TAG	R/W	0x0	—	Extra tag bits to be added to the tag field from the table. WORD2[TAG<15:8>].
<31:25>	—	RAZ	—	—	Reserved
<24:16>	OFFSET	R/W	0x0	—	Indicates offset to add to extractor memory address to get final address to the lookup table.
<15:7>	—	RAZ	—	—	Reserved
<6:0>	SKIP	R/W	0x0	—	Indicates the number of bytes to skip from the start of packet (0–64).

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

PIP Bit-Select Extractor Bit Positions/Valids Registers
PIP_BSEL_EXT_POS(0..3)

These registers contain the 8 bit positions and valids to be used when using the corresponding extractor.

Examples on bit positioning:

MSB of the 3rd byte	PIP_BSEL_EXT_CFG*[SKIP]=1 POS*n*=15 (decimal) or PIP_BSEL_EXT_CFG*[SKIP]=0 POS*n*=23 (decimal)
LSB of the 5th byte	PIP_BSEL_EXT_CFG*[SKIP]=4 POS*n*=0
2nd LSB of the 1st byte	PIP_BSEL_EXT_CFG*[SKIP]=0 POS*n*=1

POS*n*_VAL and POS*n* correspond to <*n*> in the resultant index into PIP_BSEL_TBL_ENT. When only *x* bits ($0 < x < 7$) are to be extracted, POS[7:*x*] should normally be clear. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	POS7_VAL	R/W	0	—	Valid bit for bit position 7
<62:56>	POS7	R/W	0x0	—	Bit position for the 8th bit from 128-bit segment of the packet that is defined by the SKIP field of PIP_BSEL_EXT_CFG.
<55>	POS6_VAL	R/W	0	—	Valid bit for bit position 6
<54:48>	POS6	R/W	0x0	—	Bit position for the 7th bit from 128-bit segment of the packet that is defined by the SKIP field of PIP_BSEL_EXT_CFG.
<47>	POS5_VAL	R/W	0	—	Valid bit for bit position 5
<46:40>	POS5	R/W	0x0	—	Bit position for the 6th bit from 128-bit segment of the packet that is defined by the SKIP field of PIP_BSEL_EXT_CFG.
<39>	POS4_VAL	R/W	0	—	Valid bit for bit position 4
<38:32>	POS4	R/W	0x0	—	Bit position for the 5th bit from 128-bit segment of the packet that is defined by the SKIP field of PIP_BSEL_EXT_CFG.
<31>	POS3_VAL	R/W	0	—	Valid bit for bit position 3
<30:24>	POS3	R/W	0x0	—	Bit position for the 4th bit from 128-bit segment of the packet that is defined by the SKIP field of PIP_BSEL_EXT_CFG.
<23>	POS2_VAL	R/W	0	—	Valid bit for bit position 2
<22:16>	POS2	R/W	0x0	—	Bit position for the 3rd bit from 128-bit segment of the packet that is defined by the SKIP field of PIP_BSEL_EXT_CFG.
<15>	POS1_VAL	R/W	0	—	Valid bit for bit position 1
<14:8>	POS1	R/W	0x0	—	Bit position for the 2nd bit from 128-bit segment of the packet that is defined by the SKIP field of PIP_BSEL_EXT_CFG.
<7>	POS0_VAL	R/W	0	—	Valid bit for bit position 0
<6:0>	POS0	R/W	0x0	—	Bit position for the 1st bit from 128-bit segment of the packet that is defined by the SKIP field of PIP_BSEL_EXT_CFG.

Cavium Confidential For Bowei Yang ALL 09/27/2014

PIP Alternate SKIP Configuration Registers
PIP_ALT_SKIP_CFG(0..3)

The actual SKIP I determined by hardware is based on the packet contents. BIT0 and BIT1 make up a two-bit value that the selects the skip value as follows.

lookup_value = LEN ? (packet_in_bits[BIT1], packet_in_bits[BIT0]) : (0, packet_in_bits[BIT0]);

SKIP I = lookup_value == 3 ? SKIP3 :
lookup_value == 2 ? SKIP2 :
lookup_value == 1 ? SKIP1 :
PIP_PRT_CFG<*port*>[SKIP];

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:57>	—	RAZ	—	—	Reserved
<56>	LEN	R/W	0	—	Indicates the length of the selection field. 0 = 0, BIT0 1 = BIT1, BIT0
<55:46>	—	RAZ	—	—	Reserved
<45:40>	BIT1	R/W	0x0	—	Indicates the bit location in the first word of the packet to use to select the skip amount. BIT1 must be present in the packet.
<39:38>	—	RAZ	—	—	Reserved
<37:32>	BIT0	R/W	0x0	—	Indicates the bit location in the first word of the packet to use to select the skip amount. BIT0 must be present in the packet.
<31:23>	—	RAZ	—	—	Reserved
<22:16>	SKIP3	R/W	0x0	—	Indicates number of bytes to skip from start of packet 0–64.
<15>	—	RAZ	—	—	Reserved
<14:8>	SKIP2	R/W	0x0	—	Indicates number of bytes to skip from start of packet 0–64.
<7>	—	RAZ	—	—	Reserved
<6:0>	SKIP1	R/W	0x0	—	Indicates number of bytes to skip from start of packet 0–64.

PIP Bit-Select Extractor Table Entry Registers
PIP_BSEL_TBL_ENT(0..511)

These registers contain the extractor table entries. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	TAG_EN	R/W	0	—	Enables the use of the TAG field
<62>	GRP_EN	R/W	0	—	Enables the use of the GRP field
<61>	TT_EN	R/W	0	—	Enables the use of the TT field
<60>	QOS_EN	R/W	0	—	Enables the use of the QOS field
<59:40>	—	RAZ	—	—	Reserved
<39:32>	TAG	R/W	0x0	—	TAG bits to be used if TAG_EN is set.
<31:20>	—	RAZ	—	—	Reserved
<19:16>	GRP	R/W	0x0	—	GRP field to be used if GRP_EN is set.
<15:10>	—	RAZ	—	—	Reserved
<9:8>	TT	R/W	0x0	—	TT field to be used if TT_EN is set.
<7:3>	—	RAZ	—	—	Reserved
<2:0>	QOS	R/W	0x0	—	QOS field to be used if QOS_EN is set.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PIP Per-port Configuration B Registers
PIP_PRT_CFGB(0..3,16..19,24,32..39)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:39>	—	RAZ	—	—	Reserved
<38:37>	ALT_SKP_SEL	R/W	0x0	—	Alternate skip selector. When enabled (ALT_SKP_EN), selects which of the four PIP_ALT_SKIP_CFG*x* to use with the packets arriving on the port.
<36>	ALT_SKP_EN	R/W	0	—	Enable the alternate skip selector. When enabled, the hardware is able to recompute the SKIP I value based on the packet contents. Up to two of the initial 64 bits of the header are used along with four PIP_ALT_SKIP_CFG*x* to determine the updated SKIP I value. The bits of the packet used should be present in all packets. PIP_PRT_CFG(0..3,16..19,24,32..39)[DSA_EN,HIGIG_EN] must be disabled when ALT_SKP_EN is set. ALT_SKP_EN must not be set for DPI ports (32–35) when a PKT_INST_HDR is present. ALT_SKP_EN should not be enabled for ports which have GMX_RX_FRM_CTL[PTP_MODE] set as the timestamp will be prepended onto the initial 64 bits of the packet.
<35>	—	RAZ	—	—	Reserved
<34:33>	BSEL_NUM	R/W	0x0	0x0	Which of the 4 bit select extractors to use. (Alias to PIP_PRT_CFG)
<32>	BSEL_EN	R/W	0	0	Enable to turn on/off use of bit select extractor. (Alias to PIP_PRT_CFG)
<31:0>	—	RAZ	—	—	Reserved

Cavium Confidential R-0.0
Bowen Yang
ALU
02/27/2014

9.9.1 PIP Statistics Counters

These counters work with packet ports (0–3, 16–19), AGL port (24), DPI ports (32–35), and internal loopback ports (36–39).

Special Statistics Counter Behavior

1. Read and write operations must arbitrate for the statistics resources along with the packet engines that are incrementing the counters. In order to not drop packet information, the packet hardware is always a higher priority and the CSR requests are only satisfied when there are idle cycles. This can potentially cause long delays if the system becomes full.
2. Stat counters can be cleared in two ways.
   a. If PIP_STAT_CTL[RDCLR] is set, all read accesses clear the register.
   b. Any write operation to a stats register also resets the register to 0x0.

   Note that the clearing operations must obey rule #1 above.
3. All counters are wrapping. Software must ensure they are read periodically.
4. The counters accumulate statistics for packets that are sent to PKI. If PTP_-MODE is enabled, the eight-byte timestamp is prepended to the packet. This additional eight bytes of data is captured in the octet counts.

PIP Port Statistics 0 Registers
PIP_STAT0_PRT(0..3,16..19,24,32..39)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	DRP_PKTS	R/W/H	0x0	—	Inbound packets dropped by the IPD QOS widget per port
<31:0>	DRP_OCTS	R/W/H	0x0	—	Inbound octets dropped by the IPD QOS widget per port

PIP Port Statistics 1 Registers
PIP_STAT1_PRT(0..3,16..19,24,32..39)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	OCTS	R/W/H	0x0	—	Number of octets received by PIP (good and bad)

PIP Port Statistics 2 Registers
PIP_STAT2_PRT(0..3,16..19,24,32..39)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
63:32>	PKTS	R/W/H	0x0	—	Number of packets processed by PIP.
<31:0>	RAW	R/W/H	0x0	—	RAWFULL and RAWSCH packets without an L1/L2 error received by PIP per port.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PIP Port Statistics 3 Registers
PIP_STAT3_PRT(0..3,16..19,24,32..39)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	BCST	R/W/H	0x0	—	Number of identified L2 broadcast packets. Does not include multicast packets. Only includes packets whose parse mode is SKIP_TO_L2.
<31:0>	MCST	R/W/H	0x0	—	Number of identified L2 multicast packets. Does not include broadcast packets. Only includes packets whose parse mode is SKIP_TO_L2.

PIP Port Statistics 4 Registers
PIP_STAT4_PRT(0..3,16..19,24,32..39)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	H65TO127	R/W/H	0x0	—	Number of 65-to-127-byte packets.
<31:0>	H64	R/W/H	0x0	—	Number of 64-byte packets.

PIP Port Statistics 5 Registers
PIP_STAT5_PRT(0..3,16..19,24,32..39)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	H256TO511	R/W/H	0x0	—	Number of 256-to-511-byte packets.
<31:0>	H128TO255	R/W/H	0x0	—	Number of 128-to-255-byte packets.

PIP Port Statistics 6 Registers
PIP_STAT6_PRT(0..3,16..19,24,32..39)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	H1024TO1518	R/W/H	0x0	—	Number of 1024-to-1518-byte packets
<31:0>	H512TO1023	R/W/H	0x0	—	Number of 512-to-1023-byte packets

PIP Port Statistics 7 Registers
PIP_STAT7_PRT(0..3,16..19,24,32..39)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	FCS	R/W/H	0x0	—	Number of packets with FCS or Align opcode errors
<31:0>	H1519	R/W/H	0x0	—	Number of 1519-to-max packets

NOTE: DPI does not check FCS; therefore FCS never increments on DPI ports 32–35.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PIP Port Statistics 8 Registers
PIP_STAT8_PRT(0..3,16..19,24,32..39)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	FRAG	R/W/H	0x0	—	Number of packets with length < minimum and FCS error
<31:0>	UNDERSZ	R/W/H	0x0	—	Number of packets with length < minimum

NOTE: DPI does not check FCS; therefore FRAG never increments on DPI ports 32–35.

PIP Port Statistics 9 Registers
PIP_STAT9_PRT(0..3,16..19,24,32..39)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	JABBER	R/W/H	0x0	—	Number of packets with length > maximum and FCS error
<31:0>	OVERSZ	R/W/H	0x0	—	Number of packets with length > maximum

NOTE: DPI does not check FCS; therefore JABBER never increments on DPI ports 32–35.

PIP Port Statistics 10 Registers
PIP_STAT10_PRT(0..3,16..19,24,32..39)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	BCAST	R/W/H	0x0	—	Number of packets with L2 Broadcast DMAC that were dropped due to RED. The hardware considers a packet to be an L2 broadcast packet when the 48-bit DMAC is all ones. Only applies when the parse mode for the packet is SKIP-TO-L2.
<31:0>	MCAST	R/W/H	0x0	—	Number of packets with L2 Multicast DMAC that were dropped due to RED. The hardware considers a packet to be an L2 multicast packet when the least-significant bit of the first byte of the DMAC is set and the packet is not an L2 broadcast packet. Only applies when the parse mode for the packet is SKIP-TO-L2.

Cavium Confidential For Bowen Ang 02/27/2013

PIP Port Statistics 11 Registers
PIP_STAT11_PRT(0..3,16..19,24,32..39)

See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	BCAST	R/W/H	0x0	—	Number of packets with L3 Broadcast Dest Address that were dropped due to RED. The hardware considers an IPv4 packet to be broadcast when all bits are set in the MSB of the destination address. IPv6 does not have the concept of broadcast packets. Only applies when the parse mode for the packet is SKIP-TO-L2 and the packet is IP or the parse mode for the packet is SKIP-TO-IP.
<31:0>	MCAST	R/W/H	0x0	—	Number of packets with L3 Multicast Dest Address that were dropped due to RED. The hardware considers an IPv4 packet to be multicast when the most-significant nibble of the 32-bit destination address is 0xE (i.e. it is a class D address). The hardware considers an IPv6 packet to be multicast when the most-significant byte of the 128-bit destination address is all ones. Only applies when the parse mode for the packet is SKIP-TO-L2 and the packet is IP or the parse mode for the packet is SKIP-TO-IP.

Cavium Confidential Bowen Yang ALU 02/27/2014

9.9.2 PIP Inbound Statistics Registers

The inbound statistics registers collect all data sent to PIP from all the packet interfaces, providing the raw counts of everything that comes into the block. The counts reflect all error packets and packets dropped by the PKI RED engine. PIP_STAT_INB_OCTS(0..3,16..19,24,32..39)[OCTS] includes the bytes from timestamp fields in PTP_MODE.

These counts are intended for system debug, but could convey useful information in production systems.

PIP Statistic Inbound Packets Registers
PIP_STAT_INB_PKTS(0..3,16..19,24,32..39)

Inbound packets received by PIP per port. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	PKTS	R/W/H	0x0	—	Number of packets without errors received by PIP.

PIP Statistic Inbound Octets Registers
PIP_STAT_INB_OCTS(0..3,16..19,24,32..39)

Inbound octets received by PIP per port. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	OCTS	R/W/H	0x0	—	Total number of octets from all packets received by PIP.

PIP Statistic Inbound Error Registers
PIP_STAT_INB_ERRS(0..3,16..19,24,32..39)

Inbound error packets received by PIP per port. See Table 9–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	ERRS	R/W/H	0x0	—	Number of packets with errors received by PIP.

Cavium Confidential For Bowen Yang ALU 2/27/2014

9.10 IPD Registers

The IPD registers are shown in See Table 9–9.

Table 9–9 IPD Registers

Register	Address[1]	CSR Type[2]	Detailed Description
IPD_1ST_MBUFF_SKIP	0x00014F0000000000	NCB	See page 534
IPD_NOT_1ST_MBUFF_SKIP	0x00014F0000000008	NCB	See page 534
IPD_PACKET_MBUFF_SIZE	0x00014F0000000010	NCB	See page 534
IPD_CTL_STATUS	0x00014F0000000018	NCB	See page 535
IPD_WQE_FPA_QUEUE	0x00014F0000000020	NCB	See page 536
IPD_PORT(0..3,16..19,24,32..35)_BP_PAGE_CNT	0x00014F0000000028 + α*0x8	NCB	See page 536
IPD_SUB_PORT_BP_PAGE_CNT	0x00014F0000000148	NCB	See page 537
IPD_1ST_NEXT_PTR_BACK	0x00014F0000000150	NCB	See page 537
IPD_2ND_NEXT_PTR_BACK	0x00014F0000000158	NCB	See page 537
IPD_INT_ENB	0x00014F0000000160	NCB	See page 538
IPD_INT_SUM	0x00014F0000000168	NCB	See page 538
IPD_SUB_PORT_FCS	0x00014F0000000170	NCB	See page 539
IPD_QOS(0..7)_RED_MARKS	0x00014F0000000178 + α*0x8	NCB	See page 539
IPD_PORT_BP_COUNTERS_PAIR(0..3,16..19,24,32..35)	0x00014F00000001B8 + α*0x8	NCB	See page 539
IPD_PORT(36..39)_BP_PAGE_CNT2	0x00014F0000000248 + α*0x8	NCB	See page 539
IPD_PORT_BP_COUNTERS2_PAIR(36..39)	0x00014F0000000268 + α*0x8	NCB	See page 540
IPD_PORT_BP_COUNTERS3_PAIR(40..43)	0x00014F0000000270 + α*0x8	NCB	See page 540
IPD_PORT(40..47)_BP_PAGE_CNT3	0x00014F0000000290 + α*0x8	NCB	See page 540
IPD_PORT_BP_COUNTERS4_PAIR(44..47)	0x00014F00000002B0 + α*0x8	NCB	See page 540
IPD_RED_PORT_ENABLE	0x00014F00000002D8	NCB	See page 541
IPD_RED_QUE(0..7)_PARAM	0x00014F00000002E0 + α*0x8	NCB	See page 541
IPD_PTR_COUNT	0x00014F0000000320	NCB	See page 542
IPD_BP_PRT_RED_END	0x00014F0000000328	NCB	See page 542
IPD_QUE0_FREE_PAGE_CNT	0x00014F0000000330	NCB	See page 542
IPD_CLK_COUNT	0x00014F0000000338	NCB	See page 543
IPD_PWP_PTR_FIFO_CTL	0x00014F0000000340	NCB	See page 543
IPD_PRC_HOLD_PTR_FIFO_CTL	0x00014F0000000348	NCB	See page 543
IPD_PRC_PORT_PTR_FIFO_CTL	0x00014F0000000350	NCB	See page 544
IPD_PKT_PTR_VALID	0x00014F0000000358	NCB	See page 544
IPD_WQE_PTR_VALID	0x00014F0000000360	NCB	See page 544
IPD_RED_PORT_ENABLE2	0x00014F00000003A8	NCB	See page 545
IPD_BIST_STATUS	0x00014F00000007F8	NCB	See page 545
IPD_SUB_PORT_QOS_CNT	0x00014F0000000800	NCB	See page 545
IPD_PORT_QOS_INT(0,2,3,4)	0x00014F0000000808 + α*0x8	NCB	See page 546
IPD_PORT_QOS_INT_ENB(0,2,3,4)	0x00014F0000000848 + α*0x8	NCB	See page 546
IPD_PORT_QOS_(0..31,128..159,192..199,256..319)_CNT	0x00014F0000000888 + α*0x8	NCB	See page 546

1. For registers with multiple versions, α represents the values within the specified range.
2. NCB-type registers are accessed directly across the IOI.

Cavium Confidential For Bowen Ying ALD 04/7/2014

IPD First Memory-Buffer Word Skip Size Register
IPD_1ST_MBUFF_SKIP

The number of words that the IPD skips when writing the first memory buffer (MBUF). See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	SKIP_SZ	R/W	0x0	0x0	The number of eight-byte words from the top of the first MBUF that the IPD stores the next pointer. Legal values for this field are 0 to 32, but the SKIP_SZ+18 ≤ IPD_PACKET_MBUFF_SIZE[MB_SIZE]. This field must be at least 16 when IPD_CTL_STATUS[NO_WPTR] = 1.

IPD Not First MBUF Word-Skip Size Register
IPD_NOT_1ST_MBUFF_SKIP

The number of words that the IPD skips when writing any MBUF that is not the first. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	SKIP_SZ	R/W	0x0	0x0	The number of eight-byte words from the top of any MBUF that is not the first MBUF that the IPD writes the next-pointer. Legal values are 0 to IPD_PACKET_MBUFF_SIZE[MB_SIZE] – 16 (to a maximum of 32).

IPD PACKET MBUF Size In Words Register
IPD_PACKET_MBUFF_SIZE

The number of words in a MBUF used for packet data store. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved
<11:0>	MB_SIZE	R/W	0x20	0x20	The number of eight-byte words in an MBUF. This must be a number in the range of 32 to 2048. This is also the size of the FPA's queue 0 free page.

Cavium Confidential For Bowen Yang ALU 02/27/2014

IPD Control Status Register
IPD_CTL_STATUS

The number of words in an MBUF used for packet data store. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:18>	—	RAZ	—	—	Reserved
<17>	USE_SOP	R/W	0	0	When set, the SOP sent by the MAC will be used in place of the SOP generated by the IPD.
<16>	RST_DONE	RO/H	1	0	When 0, IPD has finished reset. No access except the reading of this bit should occur to the IPD until this is asserted. Or a 1000 core clock cycles has passed after the de-assertion of reset.
<15>	CLKEN	R/W	0	0	Controls the conditional clocking within IPD. 0 = Allow hardware to control the clocks 1 = Force the clocks to always be on
<14>	NO_WPTR	R/W	0	0	When set, the work-queue-entry pointers are not used and the work-queue entry is located at the front of the packet. When set: • IPD_WQE_FPA_QUEUE[WQE_QUE] is not used; • IPD_1ST_MBUFF_SKIP[SKIP_SZ] must be at least 16; • If 16 ≤ IPD_1ST_MBUFF_SKIP[SKIP_SZ] ≤ 31, the work-queue entry is written into the first 128-byte cache block in the first buffer that contains the packet. • If IPD_1ST_MBUFF_SKIP[SKIP_SZ] = 32, the work-queue entry is written into the second 128-byte cache block in the first buffer that contains the packet.
<13>	PQ_APKT	R/W	0	0	When set, IPD_PORT_QOS_*n*_CNT is incremented by one for each work queue entry that is sent to POW.
<12>	PQ_NABUF	R/W	0	0	When set, IPD_PORT_QOS_*n*_CNT is not incremented when IPD allocates a buffer for a packet.
<11>	IPD_FULL	R/W	0	0	IPD full. When this bit is set to 1, the IPD drives the IPD_BUFF_FULL line to the IOB arbiter, telling it to not give grants to IOI devices sending packet data; when it is clear to 0, the IPD acts normally.
<10>	PKT_OFF	R/W	0	0	Packet buffer off. When this bit is set to 1, the IPD does not buffer the received packet data; when it is clear to 0, the IPD works normally, buffering the received packet data.
<9>	LEN_M8	R/W	0	1	Data length minus 8. When this bit is set to 1, 8 is subtracted from the data-length field in the header written to the POW and the top of an MBUF. The CN71XX generates a length that includes the length of the data plus 8 for the header field. By setting this bit, the 8 for the Instr field is not included in the Length field of the header. **NOTE:** When this bit is set, the CN71XX is compliant with the IPD specification.
<8>	RESET	R/W	0	0	Reset. When set, causes a reset of the IPD, except RSL.
<7>	ADDPKT	R/W	0	0	Additional packet. When set to 1, IPD_PORT_BP_COUNTERS_PAIR*n*[CNT_VAL], and IPD_PORT_BP_COUNTERS*m*_PAIR*n*[CNT_VAL] are incremented by one for every work-queue entry that is sent to POW.
<6>	NADDBUF	R/W	0	0	When set to 1, IPD_PORT_BP_COUNTERS_PAIR*n*[CNT_VAL], and IPD_PORT_BP_COUNTERS*m*_PAIR*n*[CNT_VAL] are not incremented when IPD allocates a buffer for a packet on the port.
<5>	PKT_LEND	R/W	0	0	Packet little-endian. Changes PKT to little-endian write operations to L2C.
<4>	WQE_LEND	R/W	0	0	Work-queue entry little-endian. Changes work-queue entry to little-endian write operations to L2C.

Cavium Confidential For Beta Use Only 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<3>	PBP_EN	R/W	0	0	Port backpressure enable. When set, enables the sending of port-level backpressure to the OCTEON III input ports. **NOTE:** The application should not de-assert this bit after asserting it. The receivers of this bit may have been put into backpressure mode and can only be released by IPD informing them that the backpressure has been released.
<2:1>	OPC_MODE	R/W	0x0	0x0	Select the style of write to the L2C. 0x0 = all packet data and next-buffer pointers are written through to memory. 0x1 = all packet data and next-buffer pointers are written into the cache. 0x2 = the first aligned cache block holding the packet data and initial next-buffer pointer is written to the L2 cache. All remaining cache blocks are not written to the L2 cache. 0x3 = the first two aligned cache blocks holding the packet data and initial next-buffer pointer is written to the L2 cache. All remaining cache blocks are not written to the L2 cache.
<0>	IPD_EN	R/W	0	0	IPD enable. When set to 1, enables the operation of the IPD. When clear to 0, the IPD appears to the IOB arbiter to be applying backpressure, which causes the IOB arbiter to not send grants to IOI devices requesting to send packet data to the IPD.

IPD Work-Queue Entry FPA Page Size Register
IPD_WQE_FPA_QUEUE

Specifies from which FPA queue (0–7) to fetch page pointers for work-queue entries. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved
<2:0>	WQE_QUE	R/W	1	1	Specifies the FPA queue from which to fetch page-pointers for work-queue entries. This field is not used when IPD_CTL_STATUS[NO_WPTR] is set.

IPD Port Backpressure Page-Count Registers
IPD_PORT(0..3,16..19,24,32..35)_BP_PAGE_CNT

This register specifies the maximum number of pages the port can use before backpressure (BP) is applied to the port. BP is applied once the maximum is exceeded. See also IPD_PORT(36..39)_BP_PAGE_CNT2 and IPD_PORT(40..47)_BP_PAGE_CNT3. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:18>	—	RAZ	—	—	Reserved
<17>	BP_ENB	R/W	0	0	Backpressure enable: 1 = BP is applied, 0 = BP is not applied to port.
<16:0>	PAGE_CNT	R/W	0x0	0x0	The maximum number of page pointers assigned to the port. When exceeded, BP is applied to the port. This value is in 256 page-pointer increments, (i.e. 0x0 = 0 page pointers, 0x1 = 256 page pointers, 0x2 = 512 page pointers, etc.)

Cavium Confidential For Bowen Yang ALU 5/7/2014

IPD Subtract Port Backpressure Page-Count Register IPD_SUB_PORT_BP_PAGE_CNT

This register supplies a 2's-complement value that is added to the indicated port's page-count register, with the net result of the page count lowered. The value should be the 2's complement of the value that needs to be subtracted. Users add 2's complement values to IPD_PORT(0..3,16..19,24,32..35)_BP_PAGE_CNT to return MBUFs (i.e. lower the count) to the counter in order to avoid port-level backpressure being applied to the port. Backpressure is applied when the MBUF-used count of a port exceeds the value in IPD_PORT*n*_BP_PAGE_CNT, IPD_PORT*n*_BP_PAGE_CNT2, and IPD_PORT*n*_BP_PAGE_CNT3. See Table 9–9 for address.

NOTE: This register cannot be written from the DPI via a window write operation.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:31>	—	RAZ	—	—	Reserved
<30:25>	PORT	R/W	—	0x0	The port whose page-count register receives the PAGE_CNT field.
<24:0>	PAGE_CNT	R/W	—	0x0	The number of page pointers to add to the port counter pointed to by the PORT field.

IPD First Next Pointer Back Values Register IPD_1ST_NEXT_PTR_BACK

This register contains the BACK field used to create the next pointer header for the first MBUF. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	BACK	R/W	0x0	0	Used to find head of buffer from the next pointer header.

IPD Second Next Pointer Back Value Register IPD_2ND_NEXT_PTR_BACK

This register contains the BACK field for use in creating the next pointer header for the first MBUF. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	BACK	R/W	0x0	0	Used to find head of buffer from the next pointer header.

Cavium Confidential For Bowen Yang ALU 02/27/2014

IPD Interrupt Enable Register
IPD_INT_ENB

This register is used to enable the various interrupting conditions of IPD. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved
<11>	PQ_SUB	R/W	0	0	Enables an interrupt when a port-QOS does a subtract to the count that causes the counter to wrap.
<10>	PQ_ADD	R/W	0	0	Enables an interrupt when a port-QOS does an add to the count that causes the counter to wrap.
<9>	BC_OVR	R/W	0	0	Enables interrupt when the byte-count to the IOB overflows.
<8>	D_COLL	R/W	0	0	Enables interrupt when the packet/work-queue-entry data to IOB collides.
<7>	C_COLL	R/W	0	0	Enables interrupt when the packet/work-queue-entry commands to IOB collide.
<6>	CC_OVR	R/W	0	0	Enables interrupt when the command credits to the IOB overflow.
<5>	DC_OVR	R/W	0	0	Enables interrupt when the data credits to the IOB overflow.
<4>	BP_SUB	R/W	0	0	Enables interrupts when a backpressure subtract has an illegal value.
<3>	PRC_PAR3	R/W	0	0	Enables parity error interrupts for bits [127:96] of the PBM memory.
<2>	PRC_PAR2	R/W	0	0	Enables parity error interrupts for bits [95:64] of the PBM memory.
<1>	PRC_PAR1	R/W	0	0	Enables parity error interrupts for bits [63:32] of the PBM memory.
<0>	PRC_PAR0	R/W	0	0	Enables parity error interrupts for bits [31:0] of the PBM memory.

IPD Interrupt Summary Register
IPD_INT_SUM

The fields in this register are set when an interrupt condition occurs. Write 1 to clear. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved
<11>	PQ_SUB	R/W1C/H	0	0	Set when a port-QOS does a subtract to the count that causes the counter to wrap.
<10>	PQ_ADD	R/W1C/H	0	0	Set when a port-QOS does an add to the count that causes the counter to wrap.
<9>	BC_OVR	R/W1C/H	0	0	Set when the byte-count to the IOB overflows.
<8>	D_COLL	R/W1C/H	0	0	Set when the packet/work-queue-entry data to IOB collides.
<7>	C_COLL	R/W1C/H	0	0	Set when the packet/work-queue-entry commands to IOB collide.
<6>	CC_OVR	R/W1C/H	0	0	Set when the command credits to the IOB overflow.
<5>	DC_OVR	R/W1C/H	0	0	Set when the data credits to the IOB overflow.
<4>	BP_SUB	R/W1C/H	0	0	Set when a backpressure subtract is done with a supplied illegal value.
<3>	PRC_PAR3	R/W1C/H	0	0	Set when a parity error is detected for bits [127:96] of the PBM memory.
<2>	PRC_PAR2	R/W1C/H	0	0	Set when a parity error is detected for bits [95:64] of the PBM memory.
<1>	PRC_PAR1	R/W1C/H	0	0	Set when a parity error is detected for bits [63:32] of the PBM memory.
<0>	PRC_PAR0	R/W1C/H	0	0	Set when a parity error is detected for bits [31:0] of the PBM memory.

Cavium Confidential For Bowen Yang ALU 08/27/2014

IPD Subtract Ports FCS Register
IPD_SUB_PORT_FCS

When a bit is set, the port corresponding to the set bit position subtracts four bytes from the end of the packet. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:40>	—	RAZ	—	—	Reserved
<39:36>	PORT_BIT2	R/W	0xF	0xF	When a bit is set, the port corresponding to the bit position set subtracts the FCS for packets on that port.
<35:32>	—	RAZ	—	—	Reserved
<31:0>	PORT_BIT	R/W	0xFFFFFFFF	0xFFFFFFFF	When a bit is set, the port corresponding to the bit position set subtracts the FCS for packets on that port.

IPD QOS (0..7) Marks Red High/Low Registers
IPD_QOS(0..7)_RED_MARKS

Set the pass-drop marks for QOS level. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	DROP	R/W	0x0	—	Packet will be dropped if the average queue size is equal to or less than this value.
<31:0>	PASS	R/W	0x0	—	Packet will be passed if the average queue size is larger than this value.

IPD Memory-Buffer Counters Registers
IPD_PORT_BP_COUNTERS_PAIR(0..3,16..19,24,32..35)

See also IPD_PORT_BP_COUNTERS2_PAIR*n*, IPD_PORT_BP_COUNTERS3_PAIR*n*, and IPD_PORT_BP_COUNTERS4_PAIR*n*. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved
<24:0>	CNT_VAL	RO/H	0x0	—	Number of MBUFs being used by data on this port.

IPD Loopback Port Backpressure Page-Count 2 Registers
IPD_PORT(36..39)_BP_PAGE_CNT2

This register specifies the maximum number of pages the loopback port can use before backpressure (BP) is applied to the port. BP is applied once the maximum is exceeded. See also IPD_PORT*n*_BP_PAGE_CNT and IPD_PORT*n*_BP_PAGE_CNT3. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:18>	—	RAZ	—	—	Reserved
<17>	BP_ENB	R/W	0	0	Backpressure enable: 1 = BP will be applied, 0 = BP will not be applied to port.
<16:0>	PAGE_CNT	R/W	0x0	0x0	The maximum number of page pointers assigned to the port, which, when exceeded, will cause BP to be applied to the port. This value is in 256 page-pointer increments, (i.e. 0x0 = 0 page pointers, 0x1 = 256 page pointers, 0x2 = 512 page pointers, etc.)

Cavium Confidential For Bowen Yang ALU 02/27/2014

IPD Memory-Buffer Counters 2 Loopback Port Registers
IPD_PORT_BP_COUNTERS2_PAIR(36..39)

Ports used to generate backpressure per-loopback port. See also IPD_PORT_BP_COUNTERS_PAIR*n* and IPD_PORT_BP_COUNTERS3_PAIR*n*. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved
<24:0>	CNT_VAL	RO/H	0x0	—	Number of MBUFs being used by data on this port.

IPD Memory-Buffer Counters 3 Loopback Port Registers
IPD_PORT_BP_COUNTERS3_PAIR(40..43)

Ports used to generate backpressure per-loopback port. See also IPD_PORT_BP_COUNTERS_PAIR*n* and IPD_PORT_BP_COUNTERS3_PAIR*n*. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved
<24:0>	CNT_VAL	RO/H	0x0	—	Number of MBUFs being used by data on this port.

IPD Loopback Port Backpressure Page-Count 3 Registers
IPD_PORT(40..47)_BP_PAGE_CNT3

This register specifies the maximum number of pages the loopback port can use before backpressure (BP) is applied to the port. BP is applied once the maximum is exceeded. See also IPD_PORT*n*_BP_PAGE_CNT and IPD_PORT*n*_BP_PAGE_CNT3. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:18>	—	RAZ	—	—	Reserved
<17>	BP_ENB	R/W	0	0	Backpressure enable: 1 = BP will be applied, 0 = BP will not be applied to port.
<16:0>	PAGE_CNT	R/W	0x0	0x0	The maximum number of page pointers assigned to the port, which, when exceeded, will cause BP to be applied to the port. This value is in 256 page-pointer increments, (i.e. 0x0 = 0 page pointers, 0x1 = 256 page pointers, 0x2 = 512 page pointers, etc.)

IPD Memory-Buffer Counters 4 Loopback Port Registers
IPD_PORT_BP_COUNTERS4_PAIR(44..47)

Ports used to generate backpressure per-loopback port. See also IPD_PORT_BP_COUNTERS_PAIR*n* and IPD_PORT_BP_COUNTERS3_PAIR*n*. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved
<24:0>	CNT_VAL	RO/H	0x0	—	Number of MBUFs being used by data on this port.

Cavium Confidential For Bowen Yang ALU 07/21/2014

IPD RED Port Enable Register
IPD_RED_PORT_ENABLE

Set the pass-drop marks for QOS level. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:50>	PRB_DLY	R/W	0x0	0x0	Pass-drop probability delay. The number of core-clock cycles to wait ([PRB_DLY + 68] × 8) before calculating the new packet-drop probability for each QOS level.
<49:36>	AVG_DLY	R/W	0x0	—	Average-queue-size delay. The number of core-clock cycles to wait ([AVG_DLY + 10] × 8) before calculating the moving average for each QOS level. Larger AVG_DLY values cause the moving averages of all QOS levels to track changes in the actual free space more slowly. Smaller IPD_RED_QUE*n*_PARAM[NEW_CON] (and larger IPD_RED_QUE*n*_PARAM[AVG_CON]) values can have a similar effect, but only affect an individual QOS level, rather than all.
<35:0>	PRT_ENB	R/W	0x0	0x0	Port enable. Any bit that is set enables the corresponding port's ability to have packets dropped by RED probability.

IPD RED Queue(0..7) Parameters Registers
IPD_RED_QUE(0..7)_PARAM

The parameter values that control the passing and dropping of packets by the RED engine for QOS level *n*. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:49>	—	RAZ	—	—	Reserved
<48>	USE_PCNT	R/W	0	0	Use packet/page count. When this bit is set, RED uses the actual packet-page count in place of the average for RED calculations.
<47:40>	NEW_CON	R/W	0x0	—	This value is used to control how much of the present actual queue size is used to calculate the new average queue size. The value is a number from 0 to 256, which represents NEW_CON/256 of the actual queue size that is used in the calculation. The number in this field plus the value of AVG_CON must be equal to 256. Larger IPD_RED_PORT_ENABLE[AVG_DLY] values cause the moving averages of all QOS levels to track changes in the actual free space more slowly. Smaller NEW_CON (and larger AVG_CON) values can have a similar effect, but only affect an individual QOS level, rather than all.
<39:32>	AVG_CON	R/W	0x0	—	This value is used to control how much of the present average queue size is used to calculate the new average queue size. The value is a number from 0 to 256, which represents AVG_CON/256 of the average queue size that will be used in the calculation. The number in this field plus the value of NEW_CON must be equal to 256. Larger IPD_RED_PORT_ENABLE[AVG_DLY] values cause the moving averages of all QOS levels to track changes in the actual free space more slowly. Smaller NEW_CON (and larger AVG_CON) values can have a similar effect, but only affect an individual QOS level, rather than all.
<31:0>	PRB_CON	R/W	0x0	—	Used in computing the probability of a packet being passed or drop by the WRED engine. The field is calculated to be $(255 \times 2^{24})/(\text{PASS-DROP})$ where PASS and DROP are the field from the IPD_QOS*n*_RED_MARKS register.

Cavium Confidential For
Shiwei Yang
07/7/2014

IPD Page Pointer Count Register
IPD_PTR_COUNT

Shows the number of work-queue entries and packet page pointers stored in the IPD. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:19>	—	RAZ	—	—	Reserved
<18>	PKTV_CNT	RO/H	0	0	Packet pointer valid.
<17>	WQEV_CNT	RO/H	0	0	Work-queue-entry pointer valid. This value is 1 when a work-queue entry is being held for use by the IPD. The value of this field should be added to the value of WQE_PCNT for a total count of the work-queue-entry page pointers being held by IPD. When IPD_CTL_STATUS[NO_WPTR] is set, this field represents a packet pointer, not a work-queue-entry pointer.
<16:14>	PFIF_CNT	RO/H	0x0	0x0	PFIF count. PKT_PCNT + PFIF_CNT + IPD_PRC_PORT_PTR_FIFO_CTL[MAX_PKT] is the number of packet page pointers in IPD.
<13:7>	PKT_PCNT	RO/H	0x0	0x0	Packet count. PKT_PCNT + PFIF_CNT + IPD_PRC_PORT_PTR_FIFO_CTL[MAX_PKT] is the number of packet page pointers in IPD.
<6:0>	WQE_PCNT	RO/H	0x0	0x0	Number of page pointers for work-queue-entry storage that are buffered in the IPD. The total count is the value of this buffer, plus WQEV_CNT.

IPD Backpressure Port RED-Enable Register
IPD_BP_PRT_RED_END

When IPD applies backpressure to a port and the corresponding enable bit in this register is set, the RED unit will drop packets for that port. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	PRT_ENB	R/W	0x0	0x0	Port enable. The ports corresponding to set bit positions in this field drop all NON-RAW packets to that port when port-level backpressure is applied to that port. The applying of port-level backpressure for this dropping does not take into consideration the value of IPD_PORT(0..3,16..19,24,32..35)_BP_PAGE_CNT[BP_ENB], nor IPD_RED_PORT_ENABLE[PRT_ENB].

IPD Queue0 Free Page Count Register
IPD_QUE0_FREE_PAGE_CNT

The number of free-page pointers that are available for use in the FPA for Queue 0. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	Q0_PCNT	RO/H	0x0	0x0	Queue 0 page-pointer count. The number of Queue 0 page pointers available.

Cavium Confidential For Baiwen Yang @2012/04

IPD Clock Count Register
IPD_CLK_COUNT

This register counts the number of core-clock cycles since the deassertion of reset. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	CLK_CNT	RO/HS	0x0	0x0	Clock count. This counter is cleared to 0x0 when reset is applied and increments on every rising edge of the core clock.

IPD PWP Pointer FIFO Control Register
IPD_PWP_PTR_FIFO_CTL

This register allows reading of the page pointers stored in the IPD PWP FIFO. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:61>	—	RAZ	—	—	Reserved
<60:54>	MAX_CNTS	RO/H	0x40	0x40	Maximum count. Maximum number of packet pointers or work-queue-entry pointers that could be in the FIFO. When IPD_CTL_STATUS[NO_WPTR] is set, this field only represents the maximum number of packet pointers; work-queue-entry pointers are not used in this mode.
<53:46>	WRADDR	RO/H	—	—	Work-queue-entry read address. The current FIFO work-queue-entry read address.
<45:38>	PRADDR	RO/H	—	—	Packet read address. The current FIFO packet read address.
<37:9>	PTR	RO	—	—	The output of the PWP_FIFO.
<8>	CENA	R/W	1	1	Low-active chip enable to the read port of the PWP_FIFO. This field also controls the multiplexer select that steers RADDR to the PWP_FIFO. **NOTE:** Setting this field to 0 allows reading of the memories through the PTR field, but causes unpredictable operation of the IPD under normal operation.
<7:0>	RADDR	R/W	0x0	0x0	Sets the address to read from in the PWP_FIFO. Addresses 0 through 63 contain packet pointers while addresses 64 through 127 contain work-queue-entry pointers.

IPD PRC Holding Pointer FIFO Control Register
IPD_PRC_HOLD_PTR_FIFO_CTL

This register allows reading of the page pointers stored in the IPD PRC holding FIFO. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:39>	—	RAZ	—	—	Reserved
<38:36>	MAX_PKT	RO	0x5	0x5	Maximum packet count. Maximum number of packet pointers that could be in the FIFO.
<35:33>	PRADDR	RO/H	—	—	Packet read address. The current packet pointer read address.
<32:4>	PTR	RO/H	—	—	The output of the PRC holding FIFO.
<3>	CENA	R/W	1	1	Low-active chip enable that controls the multiplexer select that steers RADDR to the holding FIFO. **NOTE:** Setting this field to 0 allows reading of the memories through the PTR field, but causes unpredictable operation of the IPD under normal operation.

Cavium Confidential For Bowen Land ALD 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<2:0>	RADDR	R/W	0x0	0x0	Sets the address to read from in the holding FIFO in the PRC. This FIFO holds packet pointers to be used for packet-data storage.

IPD PRC Port Pointer FIFO Control Register
IPD_PRC_PORT_PTR_FIFO_CTL

This register allows reading of the page pointers stored in the IPD PRC port FIFO. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:44>	—	RAZ	—	—	Reserved
<43:37>	MAX_PKT	RO	0x40	0x40	Maximum packet. Maximum number of packet pointers that are in the FIFO.
<36:8>	PTR	RO	—	—	The output of the PRC port pointer FIFO.
<7>	CENA	R/W	1	1	Low-active chip enable to the read port of the PWP_FIFO. This field also controls the multiplexer select that steers RADDR to the PWP_FIFO. **NOTE:** Setting this field to 0 allows reading of the memories through the PTR field, but causes unpredictable operation of the IPD under normal operation.
<6:0>	RADDR	R/W	0x0	0x0	Sets the address to read from in the port FIFO in the PRC. This FIFO holds packet pointers to be used for packet-data storage.

IPD Packet Pointer Valid Register
IPD_PKT_PTR_VALID

This register contains the value of the fetched packet pointer. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:29>	—	RAZ	—	—	Reserved
<28:0>	PTR	RO/H	—	—	Packet pointer value.

IPD Work-Queue Entry Pointer Valid Register
IPD_WQE_PTR_VALID

This register contains the value of the fetched work-queue-entry pointer. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:29>	—	RAZ	—	—	Reserved
<28:0>	PTR	RO/H	—	—	Work-queue-entry pointer value. When IPD_CTL_STATUS[NO_WPTR] is set, this field represents a packet pointer, not a work-queue-entry pointer.

Cavium Confidential For Bowen Yang ALU 02/27/2014

IPD RED Loopback Port Enable Register
IPD_RED_PORT_ENABLE2

This register sets the pass-drop marks for QOS level. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved
<11:0>	PRT_ENB	R/W	0x0	0x0	Port enable for loopback ports 36–47. Any bit that is set enables the corresponding port's ability to have packets dropped by RED probability.

IPD BIST STATUS Register
IPD_BIST_STATUS

This register is the BIST status for IPD memories. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:18>	—	RAZ	—	—	Reserved
<17>	CSR_MEM	RO	0	0	CSR Register Memory BIST Status.
<16>	CSR_NCMD	RO	0	0	CSR IOB Commands Memory BIST Status.
<15>	PWQ_WQED	RO	0	0	PWQ PIP work-queue entry DONE memory BIST status.
<14>	PWQ_WP1	RO	0	0	PWQ work-queue entry PAGE1 PTR memory BIST status.
<13>	PWQ_POW	RO	0	0	PWQ POW MEM memory BIST status.
<12>	IPQ_PBE1	RO	0	0	IPQ PBE1 memory BIST status.
<11>	IPQ_PBE0	RO	0	0	IPQ PBE0 memory BIST status.
<10>	PBM3	RO	0	0	PBM3 memory BIST status.
<9>	PBM2	RO	0	0	PBM2 memory BIST status.
<8>	PBM1	RO	0	0	PBM1 memory BIST status.
<7>	PBM0	RO	0	0	PBM0 memory BIST status.
<6>	PBM_WORD	RO	0	0	PBM_WORD memory BIST status.
<5>	PWQ1	RO	0	0	PWQ1 memory BIST status.
<4>	PWQ0	RO	0	0	PWQ0 memory BIST status.
<3>	PRC_OFF	RO	0	0	PRC_OFF memory BIST status.
<2>	IPD_OLD	RO	0	0	IPD_OLD memory BIST status.
<1>	IPD_NEW	RO	0	0	IPD_NEW memory BIST status.
<0>	PWP	RO	0	0	PWP memory BIST status.

IPD Subtract Port QOS Count Register
IPD_SUB_PORT_QOS_CNT

Supplies a 2s-complement value [CNT] that is added to the indicated port/QOS-count register, with the net result of the page count lowered. The value added is indicated in the PORT_QOS field and must be the 2's complement of the value that needs to be subtracted. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:41>	—	RAZ	—	—	Reserved
<40:32>	PORT_QOS	R/W	—	0x0	The port/QOS to add the CNT field to.
<31:0>	CNT	R/W	—	0x0	The value to be added to the register selected in the PORT_QOS field.

Cavium Confidential For Bowen Yang ARU 02/27/2014

IPD Port-QOS Interrupt Register 0
IPD_PORT_QOS_INT(0,2,3,4)

The bit corresponding to the port/QOS is set in this register when the per-port/QOS counter exceeds its set corresponding threshold as defined in IPD_PORT_QOS_(0..31,128..159,192..199,256..319)_CNT[WMARK]. The ports and associated registers are shown in Table 9–10.

Table 9–10 Port QOS Interrupt Registers

Register	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15:8>	<7:0>
INT0	reserved				Port 3	Port 2	Port 1	Port 0
INT1	reserved							
INT2	reserved				Port 19	Port 18	Port 17	Port 16
INT3	reserved							Port 24
INT4	Port 39	Port 38	Port 37	Port 36	Port 35	Port 34	Port 33	Port 32
INT5	Port 47	Port 46	Port 45	Port 44	Port 43	Port 42	Port 41	Port 40

See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	INTR	R/W1C	0x0	0x0	Interrupt bits.

IPD Port-QOS Interrupt Enable Register 0
IPD_PORT_QOS_INT_ENB(0,2,3,4)

This register provides the interrupt-enable bits for port/QOS interrupts; when the enable bit is set and the corresponding bit in IPD_PORT_QOS_INT(0,2,3,4) is set, an interrupt is generated. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	ENB	R/W	0x0	all-ones	Enable bits.

IPD Port QOS Count Register
IPD_PORT_QOS_(0..31,128..159,192..199,256..319)_CNT

Each of these registers provides a counter per port/QOS. See Section 9.8 for a detailed description of the port/QOS counter. The counters are organized in sequence where the first eight counter (0–7) belong to port 0, QOS 0–7 respectively, followed by port 1 at (8–15), etc. See Table 9–9 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	WMARK	R/W	all-ones	—	When the field CNT, after being modified, is equal to or crosses this value (i.e. value was greater than, then becomes less than, or the value was less than and becomes greater than) the corresponding bit in IPD_PORT_QOS_INT(0,2,3,4) is set.
<31:0>	CNT	RO/H	0x0	—	The packet-related count that is incremented as specified by IPD_SUB_PORT_QOS_CNT.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Chapter 10

Packet-Output Processing Unit (PKO)

This chapter contains the following subjects:

- Overview
- Output Ports
- Output Packet Format and TCP/UDP Checksum Insertion
- PKO Output Queue
- PKO Commands
- PKO Queue-Arbitration Algorithm
- PKO-Internal Port-to-Engine Arbitration Algorithm
- PKO Don't-Write-Back (DWB) Calculation
- PKO-External Port-Level Backpressure
- PKO-Internal Port Rate-Limiting
- PKO Performance
- PKO Operations
- PKO Registers

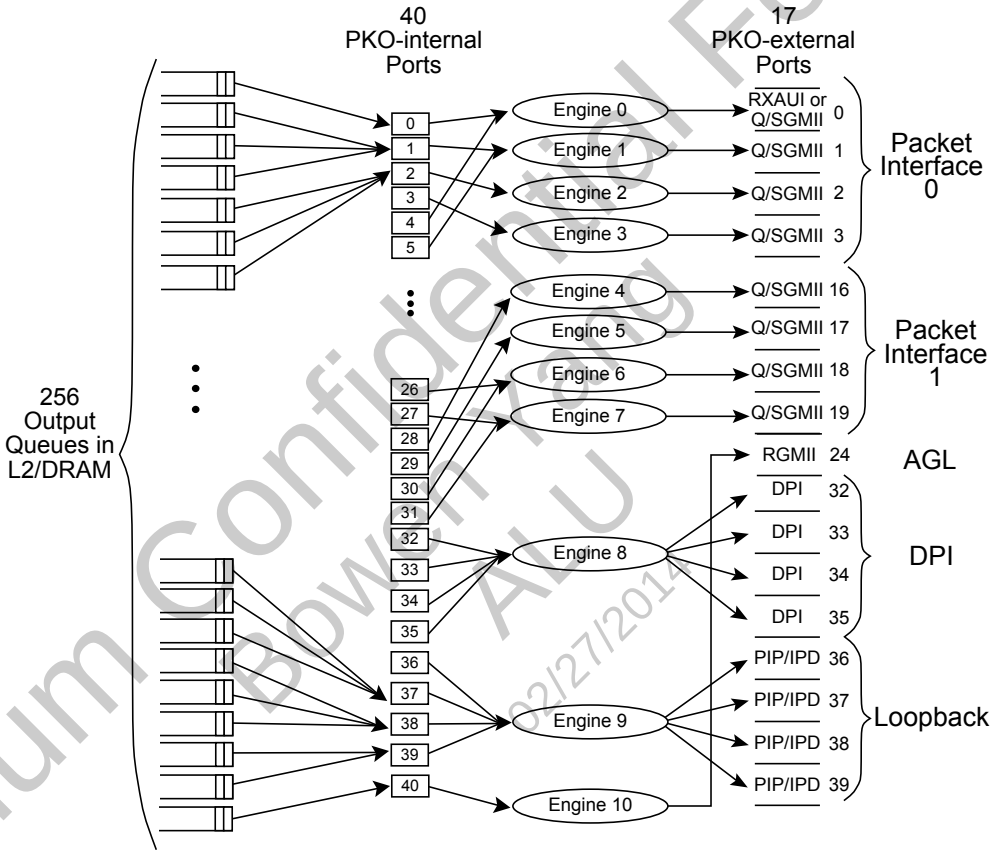
Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

This chapter discusses the centralized packet-output processing (PKO) unit. It gathers packet data from L2/DRAM and sends it out on any/all of the SGMII/ QSGMII, RXAUI, RGMII, or DPI interfaces. It can also loopback packets to the CN71XX packet input. PKO can send packets to a combined total of up to 17 different ports/destinations. Each individual port/destination receives a linear stream of packets from PKO.

Figure 10–1 shows the PKO conceptual overall architecture. The packet-output queues are on the left. Core software inserts packets into one of the 256 output queues when it needs to send a packet. The 17 PKO packet ports/destinations are on the right. PKO hardware implements a programmable multi-level arbitration algorithm to assign packets to one of 11 available PKO DMA engines, including 41 PKO-internal rate-limited ports.

Figure 10–1 PKO Conceptual Architecture



The PKO hardware supports up to 256 output queues that buffer packets to be sent out to the 17 available PKO-external ports. Each output queue is a chunked list of commands in L2/DRAM. Each packet transfer is one command in an output queue. The core software writes a queue doorbell register inside the PKO unit after it adds commands to a queue. The PKO hardware reads commands from L2/DRAM and can cache four (or more) commands per queue inside the unit. The queue commands can

Cavium Confidential For Bowen Yang ALU 02/27/2014

optionally direct PKO to perform up to two fetch-and-add (FAU) register decrements after sending a packet. They can also direct PKO to either submit a work-queue entry or write an L2/DRAM byte to zero after sending a packet.

The PKO hardware has both a linked mode and a true-gather mode that can construct full packets from multiple packet segments in L2/DRAM, and can optionally (on a per-segment basis) free up the buffers containing the packet data and/or gather list after sending a packet. The PKO hardware is bi-endian and supports L2 cache bypass on a packet-by-packet basis. It also frees command queue chunks/buffers after it reads all the command words from the chunk.

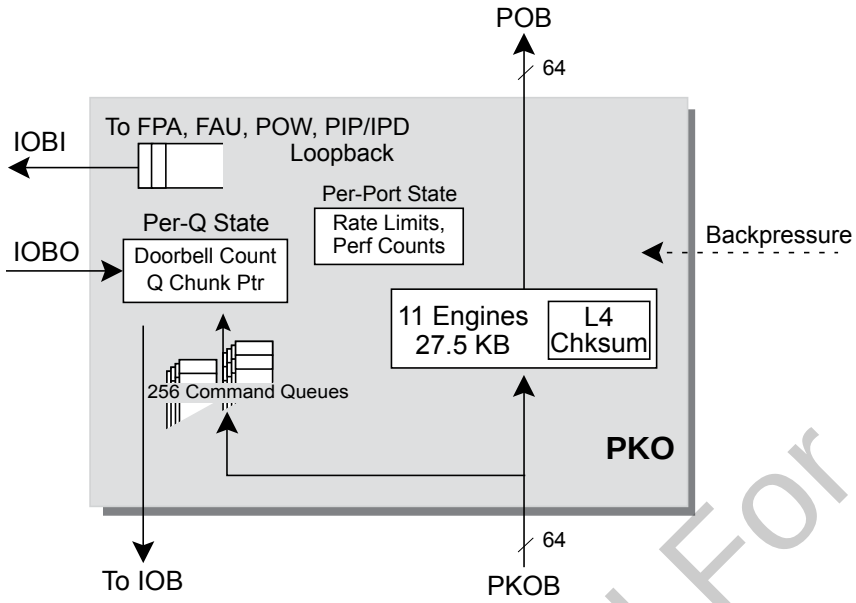
Each PKO-internal port can have a variable number of contiguous output queues attached to it, up to a maximum of 16. Section 10.5 describes the PKO arbitration algorithm that selects packets from the output queues for the internal ports. The algorithm includes a mixture of static priority and weighted round-robin algorithms on a packet-by-packet basis. PKO contains byte and packet-performance counters per internal port.

Each PKO DMA engine fetches up to four in-flight packets, distributing the packets amongst the PKO-external ports attached to the DMA engine. Each engine contains hardware to calculate the L4 checksum for a flexibly-positioned IP TCP/UDP packet. When this checksum option is selected for a packet, the PKO unit can often buffer an entire packet in its internal store before inserting the checksum as it sends the packet out the packet-out bus (POB). Each engine has 2.5KB of storage. A GMX packet port has a minimum of 2.5KB, which can be changed with PKO_REG_GMX_PORT_MODE. An engine only reads the packet data from L2/DRAM once to send out a packet, unless it is directed to calculate the checksum for a TCP/UDP packet that is too large to fit in its available buffering. The packets sent through the different DMA engines share some of the same PKO hardware resources, but logically the PKO unit treats the different in-flight packets independently. Each DMA engine can have a programmable number of PKO-internal ports attached to it. Section 10.5 describes the arbitration algorithm that selects packets from the PKO-internal ports for the PKO DMA engines, which includes port rate-limiting as well as a second level of static priority and weighted round-robin arbitration.

Figure 10–2 shows the PKO unit internal architecture in more detail. The packet-data and queue commands enter the unit from L2/DRAM via the PKO bus (PKOB). The PKO hardware transfers the packet data to the SGMII/QSGMII/RXAUI or DPI output ports via the POB. PKO transfers internal loopback packet data to the PIP/IPD via the inbound I/O bus (IOBI). All gather/packet/chunk buffer frees, FAU register decrements, work-queue entry submits, or L2/DRAM zero-byte write operations also exit PKO by the IOBI. The output-queue doorbell write operations arrive at the PKO via the outbound I/O bus (IOBO).

Cavium Confidential For Broadcom Use Only

Figure 10–2 PKO Internal Architecture



10.1 Output Ports

As Figure 10–1 indicates, the PKO hardware sends packets to up to 17 PKO-external output ports/destinations numbered as follows:

- PKO ports 0–3 = Packet interface 0 ports 0–3
- PKO ports 16–19 = Packet interface 1 ports 0–3
- PKO port 24 = AGL port
- PKO ports 32–35 = DPI ports 0–3
- PKO ports 36–39 = internal loopback ports 0–3

When a packet interface is in XAUI mode, it only uses the first port assigned to it, so only PKO port 0 exists, and the remainder (i.e. PKO ports 1–3) are not used.

Chapter 23, Chapter 21 and Chapter 25 describe the CN71XX packet interfaces.

Section 12.2 describes the CN71XX DPI output ports. Note that the CN71XX DPI logic distributes packets from a DPI output port across multiple DPI packet-output rings. It uses the relative position of a PKO output queue attached to a PKO-internal port to do this. Section 12.2 describes the details.

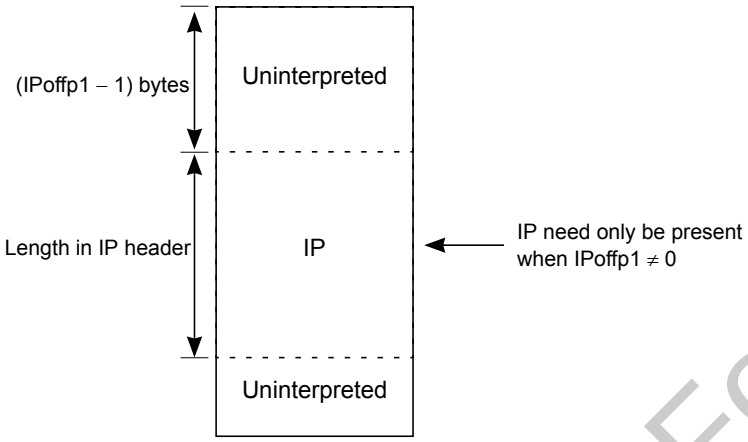
The CN71XX loopback ports provide a loopback path from L2/DRAM, through PKO, into PIP/IPD, and on to L2/DRAM and POW, like packets that arrive via the packet interfaces and DPI input ports. PIP/IPD ports 36–39 correspond to PKO output ports of the same number. Chapter 9 describes PIP/IPD loopback.

Cavium Confidential For Bowen Yang ALU 6/27/2014

10.2 Output Packet Format and TCP/UDP Checksum Insertion

Figure 10–3 shows the format requirements for packets entering and exiting PKO.

Figure 10–3 Format Requirements for Packets



If the command for a packet does not direct PKO to calculate a TCP/UDP checksum, there is no format requirement. In that case, PKO simply interprets the output packet data as a byte stream. The packet may actually contain an L2 header, an IP packet, and other fields, but the PKO unit does not interpret them.

If the command for a packet directs the PKO unit to calculate a TCP/UDP checksum, the command also contains a field (IPoffp1) that indicates the first byte of the IP packet. The following are the requirements to use the PKO unit TCP/UDP L4 checksum generation:

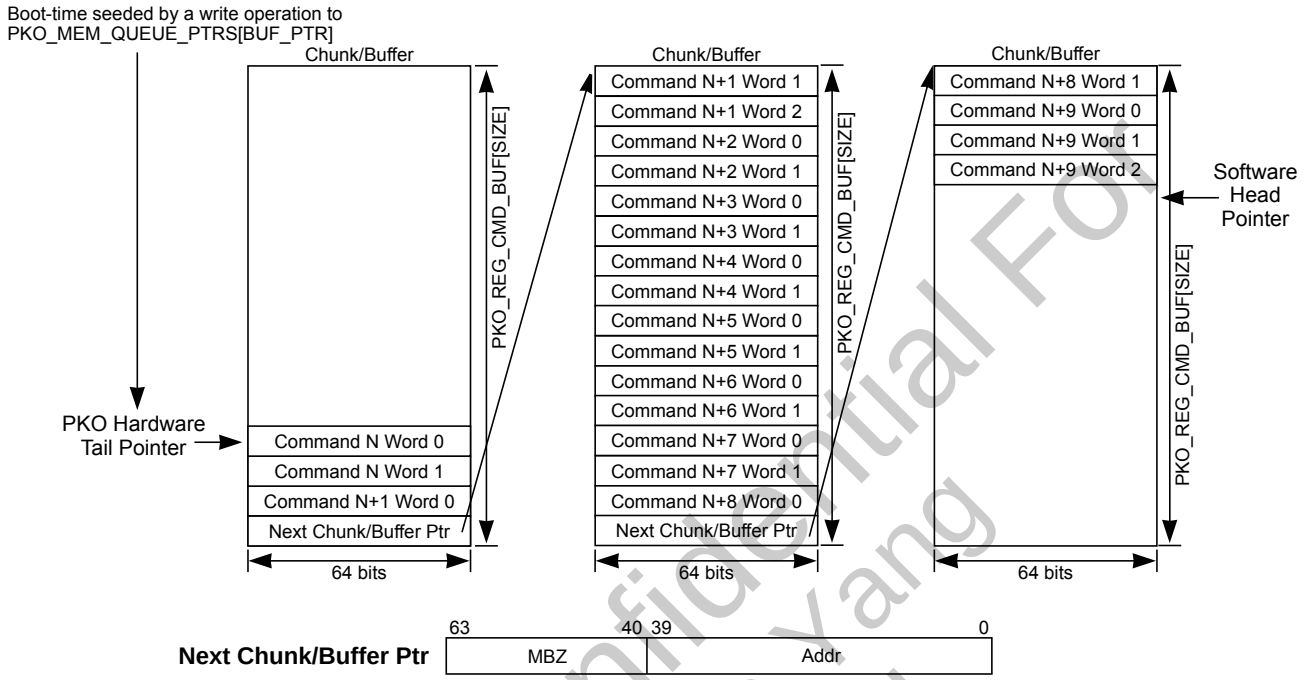
- The IP must be TCP or UDP (i.e. the protocol/next_header field must be either TCP (6) or UDP (17)) and must entirely reside in this packet. The IP can be followed by a trailer.
- If IPv4, the packet must not have options and must not be a fragment (i.e. HLEN must equal 5 and both MF and offset must be zero).
- If IPv6, the packet must not have any extension headers prior to the TCP/UDP header (i.e. the next_header field in the IPv6 header must be either TCP (6) or UDP (17)).
- In the UDP case, the PKO unit uses the UDP header length field to calculate the checksum, so the UDP length value must be legal (e.g. it must not indicate that the UDP data is longer than the IP length allows).
- For IPv4/IPv6 and UDP/TCP, there are the following limits to the indicated packet length:
  - for IPv6, in both the UDP and TCP cases, the indicated packet length must be less than 65495
  - for IPv4, in the UDP case, the indicated packet length must be less than 65515

Cavium Confidential For Bowen Yang ALU 2/27/2014

10.3 PKO Output Queue

Figure 10–4 shows the structure of each output queue. Each output queue is a linked list of chunks, or buffers. Software allocates these chunks and the PKO hardware frees them. The PKO hardware reads words from the output queue (starting at the tail), and traverses the next chunk buffer pointer to the next chunk when it reaches the last word of a chunk. When the PKO hardware jumps chunks like this, it frees the earlier chunk/buffer to the FPA hardware-managed pool selected by the PKO_REG_CMD_BUF[POOL] field.

Figure 10–4 Structure of Output Queue



The PKO_REG_CMD_BUF[SIZE] field indicates the size of a chunk. Figure 10–4 shows an example where the chunk size is 16 64-bit words – the minimum legal value. The final word in a chunk is a next chunk/buffer ptr, but all other words are valid command words. Each command has the information the PKO hardware needs to send out a single packet. This example shows ten commands, or ten packets, in the output queue. The commands are numbered from *N* to *N*+9, with smaller numbers for the older commands. The packets for the older commands are sent first. Every command is either two or three words. Note that the words of an individual command may straddle a chunk boundary. This happens for commands one and eight in Figure 10–4.

The PKO hardware maintains the tail pointer for the queue and the core software maintains the head pointer. To insert a packet into a queue, software must first write the command words for the packet into the queue, allocating chunks if necessary, and then write the PKO doorbell address for the queue with the number of words added to the queue. (“PKO Operations” on page 570 describes the store operations to the PKO unit that are a doorbell write operation.) Software must guarantee that L2/DRAM contains the number of words indicated in the doorbell write operation before the doorbell write operation reaches the PKO.

Cavium Confidential For Bowen Yang 6/8/2014

The distance between the head pointer and the tail pointer is both the size of the output queue and the outstanding doorbell count. The size of the output queue is limited only by the available memory and the 20-bit outstanding doorbell counter for a queue. Core software must guarantee that each queue is smaller than 2^{20} words.

Note that the PKO hardware may read the next chunk buffer ptr as soon as the doorbell count indicates that the next-to-last word in a chunk contains a valid command word. This implies that software must allocate the next chunk buffer, and set the next chunk buffer ptr in the prior chunk to point to it, as soon as it writes the next-to-last chunk word.

Software can “ring the doorbell” with any number of command words. Software can issue a doorbell write operation for each individual command word, or can issue exactly one doorbell write operation for each command, or can accumulate the words from multiple commands into a single doorbell write operation. The only requirements are that the number of valid command words in the queue must be at least as large as the doorbell count, and the next chunk buffer pointers interspersed among the words must also be set up properly. Section 10.11 describes the operations needed to ring the queue doorbells.

Figure 10–4 shows only a single output queue. The PKO hardware can manage either 256, 128, or 64 possible output queues in different modes selected by the PKO_REG_QUEUE_MODE register. An application does not need to use all possible queues in any of the three modes. The advantage of the modes with fewer queues is that the PKO unit has a fixed-size internal memory for command caching, and it caches more commands for each queue when there are fewer configured queues. The command cache is 8, 16, or 32 (64-bit) words per queue for the 256, 128, and 64 queue modes, respectively.

At boot time, software must configure each queue with the original next chunk buffer pointer (i.e. the starting tail pointer) and must attach a queue to a port. This is done with write operations to the PKO_MEM_QUEUE_PTRS register. This configuration can attach from one to 16 queues to a given output port. All queues attached to a single port must have contiguous queue IDs, but otherwise there are no queue/port attachment constraints. Software must supply both the queue ID and its attached port ID with each doorbell write operation.

Figure 10–4 also shows the next chunk buffer pointer format. The primary component is the Addr field that selects a legal L2/DRAM byte location. Though Addr is a byte address, it must be naturally aligned on a 128-byte cache block boundary, so its least-significant seven bits must be all 0s.

10.4 PKO Commands

PKO reads packets from L2/DRAM memory and passes them out a port whenever one of the queues attached to the port has valid commands. Each command can individually specify whether the PKO hardware should calculate and insert the IP TCP/UDP checksum.

Each command can also specify either linked or gather mode to construct the packet. In either case, the PKO hardware reads the packet segments from L2/DRAM, and can also optionally free the buffers containing the segments to one of the eight available hardware-managed free pools. (See “Free Pool Unit (FPA)” on page 439.) The free pool selection and the decision whether to free the buffer containing a segment can be on a segment-by-segment basis. A command can also specify whether to free buffers on a command-by-command basis.

Cavium Confidential For Power Lang ALU 02/26/2014

Figure 10–5 shows an example of usage of the linked packet construction mode. This example shows a packet consisting of three segments. A segment is simply contiguous packet bytes which, except for the last segment, must be preceded by a 64-bit pointer to the next segment. In the example, the entire packet is the concatenation of all of segment 0, all of segment 1, and the first bytes of segment 2.

Figure 10–5 Usage of the Linked Packet Construction Mode

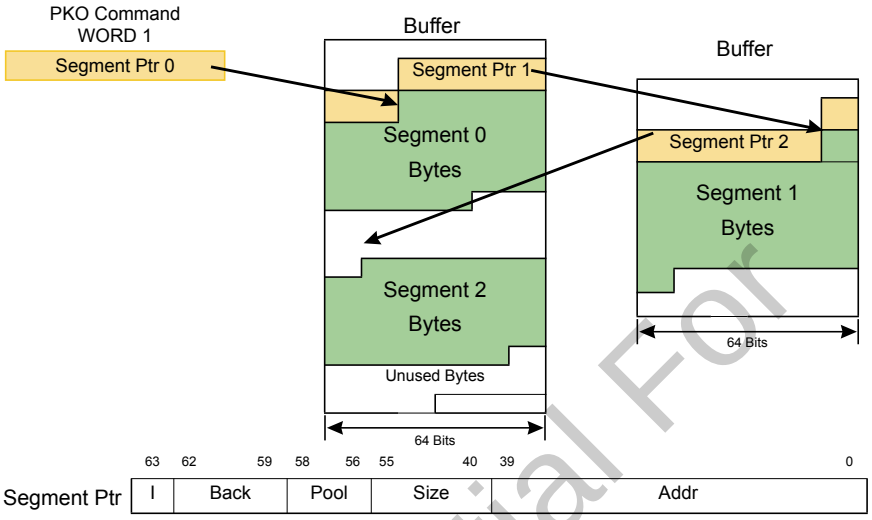
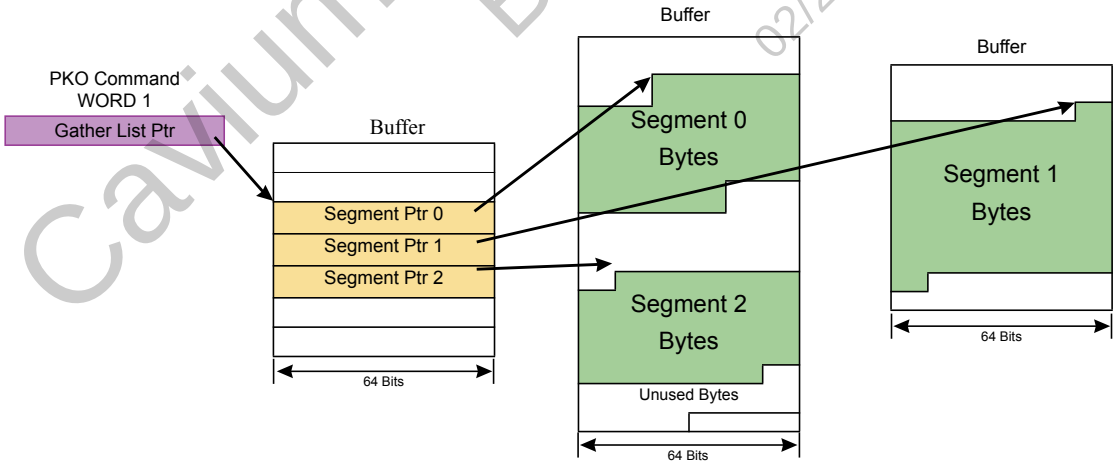


Figure 10–5 also shows the format of the 64-bit segment pointer. It has the byte address (Addr), the size in bytes (size), the hardware free pool (Pool), the distance from the start of the segment to the start of the buffer (Back), and free selection bit (I) for the next segment. This segment pointer format matches the format of WORD 1 in the command and is described below.

In linked mode, the segment pointer in WORD 1 of the command must be naturally-aligned on a 64-bit boundary. All other segment pointers precede the start of the segment data by exactly eight bytes and so are not aligned on a 64-bit boundary when the segment start is not aligned on a 64-bit boundary.

Figure 10–6 shows the same packet example as Figure 10–5, but it shows the packet segments connected with gather mode rather than linked mode.

Figure 10–6 Packet Segments Connected with Gather Mode



Cavium Confidential For Bolven Kang ALU 02/27/2014

Gather mode is different from linked mode in the following three ways:

- There need not be any segment pointers preceding any segment data.
- Instead, all segment pointers are contained in a gather list. The gather list (i.e. all segment pointers) must be aligned on a 64-bit boundary. The buffer containing a gather list can be freed just like the buffer containing any segment can be freed.
- WORD 1 of the command points indirectly to the gather list rather than directly to the first segment. The format of this gather list pointer is similar to a segment pointer.

Note the distinction between a segment and a buffer in Figures 10–5 and 10–6. Buffers that the PKO hardware frees must start on a 128-byte boundary and be at least 128 bytes. A segment can be as small as one byte and can start on any byte alignment. Multiple segments may reside in a single buffer. The first and last segment in Figure 10–5 share the same buffer, and the second segment resides in a different buffer.

Segments and gather lists can share buffers freely, but when they do, software must guarantee that the PKO hardware does not free a buffer before reading all segments and gather-list entries contained in the buffer. When the command/buffer pointer directs the PKO hardware to free the buffer containing a segment or gather list, PKO may free the buffer immediately after it finishes reading the segment or gather list. The consequence is that software must follow the following rules regarding PKO buffer frees:

- When multiple segments share a buffer, only the last segment in the list can direct the PKO to free the buffer. This rule applies to both linked- and gather-mode packets.
- With a gather-mode packet, whenever a gather list shares a buffer with one or more segments from the packet, only the last segment in a packet can direct PKO to free the buffer. If the last segment of the packet is not one of the segments sharing this buffer in this case, PKO cannot safely free this buffer as part of the packet send.

The last segment is special in both linked and gather modes. First, in linked mode, a segment pointer need not precede the last segment. The field does not need to exist, and the PKO hardware will never read it. Second, in both linked and gather mode, the last segment may be larger than needed to contain the packet. The total bytes field in the command determines the number of bytes in the packet. The PKO hardware uses this field to determine when to stop taking bytes from the last segment. This total bytes field must be larger than the sum of the sizes of the segments other than the last, and less than or equal to the sum of the sizes of all segments including the last. In other words, the last segment must hold at least one byte of the packet.

Figures 10–5 and 10–6 show PKO in the default big-endian mode. PKO also supports full little-endian mode on a packet-by-packet basis. In little-endian mode, the following apply:

- The output queue and commands, segment pointers, and gather-list formats are unchanged.
- The Addr field of a segment pointer is a little-endian byte pointer rather than a big-endian one.
- Segment bytes are fetched from L2/DRAM in little-endian format rather than big-endian format.

Cavium Confidential For Bowen Yang 8/2/2014

- Unaligned segment pointers (in linked-packet mode) must be in correct little-endian format, eight bytes prior to the segment bytes (i.e. the least-significant eight bytes of the pointer are first, the next least-significant bytes are next, etc.).

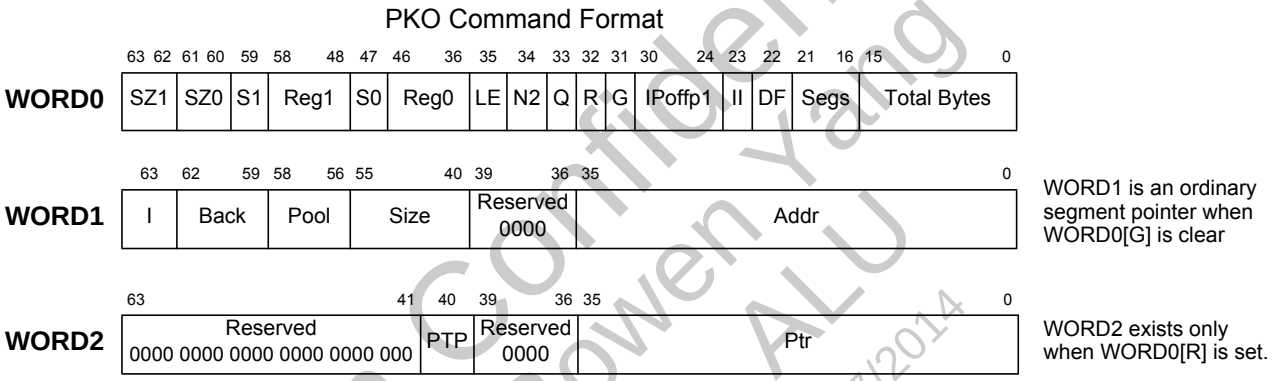
Each command may individually specify that the PKO hardware complete up to three additional operations after it fetches all pointers and packet data indicated by the command:

- Decrement up to two Fetch-and-Add (FAU) registers. The particular register, the size of the decrement operation (64, 32, 16, or 8-bits), and the decrement amount (either one or the packet (i.e. WORD0[Total Bytes]) size in bytes) can be specified individually for each decrement. With these operations, software can maintain accurate queue sizes (bytes and/or packets) for each queue in selected FAU registers.
- Either:
  - write an L2/DRAM byte to zero, or
  - submit a work-queue entry

With the L2/DRAM write, software can poll until the PKO hardware sends a packet. The work-queue entry is pre-constructed by software, so can cause the software to perform arbitrary tasks.

Figure 10–7 shows the format of commands in the PKO output queues. Every command is either two or three 64-bit words. The last word only exists when WORD0[R] is set.

Figure 10–7 Format of Commands in the PKO Output Queues



The WORD1 format matches the segment pointer format shown in Figure 10–5, though WORD1 is a gather list pointer rather than a segment pointer when WORD0[G] is set. The description of the WORD1 fields describes all their uses.

The remainder of this section provides descriptions of each field in the command.

WORD0[SZ1] Size of the WORD0[Reg1] FAU subtract (0 = 8-bit, 1 = 16-bit, 2 = 32-bit, 3 = 64-bit) that can follow a packet send. Must be 0x0 when WORD0[Reg1] is 0x0.

WORD0[SZ0] Size of the WORD0[Reg0] FAU subtract (0 = 8-bit, 1 = 16-bit, 2 = 32-bit, 3 = 64-bit) that can follow a packet send. Must be 0x0 when WORD0[Reg0] is 0x0.

WORD0[S1] If set, the WORD0[Reg1] FAU subtract value is 1, else the WORD0[Reg1] subtract value is WORD0[Total Bytes]. Must be 0x0 when WORD0[Reg1] is 0x0. Note that WORD0[Total Bytes] does not account for CRC bytes that may optionally be added to the packet.

Cavium Confidential For Bowen Yang ALU 02/27/2014

WORD0[Reg1] If non-zero, PKO subtracts from the WORD0[Reg1] FAU register after it sends the packet. If WORD0[Reg1] is zero, PKO does not subtract from WORD0[Reg1] after it sends the packet. WORD0[Reg1] is a byte address in the FAU register file, and must be naturally-aligned based on SZ1. (i.e. The WORD0[SZ1] least-significant bits of WORD0[Reg1] must be zero.)

When IOB_CTL_STATUS[PKO_ENB] is clear, WORD0[Reg1] is a big-endian pointer; when it is set, WORD0[Reg1] is a little-endian pointer

WORD0[S0] If set, the WORD0[Reg0] FAU subtract value is 1, else the WORD0[Reg0] subtract value is WORD0[Total Bytes]. Must be 0 when WORD0[Reg0] is 0x0. Note that WORD0[Total Bytes] does not account for CRC bytes that may optionally be added to the packet.

WORD0[Reg0] If non-zero, PKO subtracts from the WORD0[Reg0] FAU register after it sends the packet. If WORD0[Reg0] is zero, PKO does not subtract from WORD0[Reg0] after it sends the packet. WORD0[Reg0] is a byte address in the FAU register file, and must be naturally-aligned based on SZ0. (i.e. The WORD0[SZ0] least-significant bits of WORD0[Reg0] must be 0x0.)

When IOB_CTL_STATUS[PKO_ENB] is clear, WORD0[Reg0] is a big-endian pointer; when it is set, WORD0[Reg0] is a little-endian pointer

WORD0[LE] Determines PKO's endian mode. When clear (default), PKO operates in big-endian mode, as the diagrams in this section show. When set, all segment pointers and segment bytes are interpreted in little-endian order.

WORD0[N2] Determines PKO's L2 cache allocation. When clear, PKO allocates all load data into the L2 cache. When set, PKO does not allocate blocks containing segment bytes into the L2 cache.

NOTE: PKO always allocates command chunks and gather lists into the L2 cache.

Also, PKO always allocates packet data into the cache on the first packet read of a two-pass TCP/UDP checksum calculation. Refer to Section 10.10 for more performance details regarding this calculation.

WORD0[N2] affects performance, but otherwise does not affect CN71XX behavior. It may be advantageous to set WORD0[N2] if packet data will not be used after PKO sends the packet, which will allow the L2 cache to retain other more useful information.

WORD0[Q] When WORD0[R] is set, WORD0[Q] indicates whether WORD2[Ptr] is a WQE pointer or an L2/DRAM pointer. WORD0[Q] must be zero when WORD0[R] is clear. When WORD0[Q] is set, WORD2[Ptr] points to a WQE that PKO will submit to POW after sending the packet. (See Chapter 7 for a description of POW WQEs.) When WORD0[Q] is clear (and WORD0[R] is set), WORD2[Ptr] points to an L2/DRAM location that will be written by PKO after sending the packet.

When both WORD0[Q,R] are set:

- If WORD2[PTP] is clear, PKO submits the WQE to POW after it sends the packet.
- If WORD2[PTP] is set, PKO first timestamps the outgoing packet, then writes the 64-bit timestamp (at offset PKO_REG_TIMESTAMP[WQE_WORD]) into the WQE, then submits the WQE to POW after it sends the packet.

Cavium Confidential For Power Yang ATU 02/17/2014

When WORD0[Q] is clear while WORD0[R] is set:

- If WORD2[PTP] is clear, PKO writes the L2/DRAM 8-bit byte to zero after sending the packet.
- If WORD2[PTP] is set, PKO first timestamps the outgoing packet, then writes the L2/DRAM 64-bit word to the 64-bit timestamp after it sends the packet.

WORD0[R] Indicates whether WORD2 exists. If WORD0[R] is set, WORD2 exists and the PKO command is three words total. If WORD0[R] is clear, WORD2 does not exist and the PKO command is two words total.

The WORD0[Q] description above describes how WORD2 is used.

WORD0[G] Determines whether the PKO hardware operates in linked or gather mode for the packet. If set, PKO is in gather mode and WORD1 points to a gather list. If clear, PKO is in linked mode and WORD1 points to the first segment.

WORD0[IPoffp1] Determines whether the PKO hardware calculates the TCP/UDP checksum for a packet and specifies the location of the first byte of the IP packet. If WORD0[IPoffp1] is zero, PKO does not insert a TCP/UDP checksum into the packet. If WORD0[IPoffp1] is non-zero, PKO hardware will generate and insert the TCP or UDP checksum for an IP packet. The IP packet must be exactly WORD0[IPoffp1]–1 bytes from the beginning of the packet. Section 10.2 discusses requirements for and restrictions for TCP/UDP checksum generation.

WORD0[II] If set, ignore the I bit (effectively, force them all to zero) in all segment and gather pointers.

WORD0[DF] If set, by default buffers should not be freed for all segments (and any gather list, if present). If clear, by default buffers should be freed for any segments (or any gather list, if present). If WORD0[II] is set, WORD0[DF] is more than the default – it completely controls whether a buffer is freed by the PKO hardware. If WORD0[II] is clear, the I bit in the segment/gather pointer can invert the default behavior (i.e. If WORD0[II] is clear, the buffer containing the segment is freed when {WORD0[DF] $\oplus$ (segment/gather pointer I bit)} = 0.)

WORD0[Segs] The number of segments. If WORD0[G] is clear, WORD0[Segs] indicates the number of linked segments. If WORD0[G] is set, WORD0[Segs] is also the number of entries in the gather list.

WORD0[Total Bytes] The number of bytes in the packet. All packet bytes must reside in exactly WORD0[Segs] segments. Must be greater than 0 and not exceed 2^{16} – 8 bytes (65528 bytes).

I (WORD1 and all segment pointers) Invert bit of the segment/gather list pointer. Used to change the default freeing behavior for the buffer containing the segment or gather list. I exists both in WORD1 and in all segment pointers. When WORD0[II] is clear, I inverts the free behavior described by the WORD0[DF] bit. I is never used by the PKO hardware when WORD0[II] is set.

Back (WORD1 and all segment pointers) Back field of the segment/gather list pointer. When freeing the buffer containing this segment or gather list, the PKO hardware finds the start address of the buffer using the Back and Addr fields:

```
Buffer Start Address = ((Addr >> 7) - Back) << 7
```

NOTE: The Back value can specify a Buffer Start Address that is from 0–2047 bytes prior to Addr, and the Buffer Start Address must always be aligned on a 128-byte cache block boundary.

Cavium Confidential For Bowen Yang AMD 02/27/2017

Pool (WORD1 and all segment pointers) Pool field of the segment/gather list pointer. The PKO hardware frees to this (FPA hardware-managed) free pool when it frees the buffer containing this segment or gather list.

Size (WORD1 and all segment pointers) Size field of the segment/gather list pointer.

- If WORD0[G] is clear or the pointer does not reside in WORD1 of the command, Size is the number of bytes in the segment pointed at by Addr.
- If WORD0[G] is set, WORD1[Size] is the number of segments and must exactly equal WORD0[Segs].

Note that for all segment pointers other than the last, Size is also the number of packet bytes pointed at by Addr, but that the last segment may contain fewer than Size packet bytes. Size must not exceed $2^{16} - 8$ bytes (65528 bytes).

Addr (WORD1 and all segment pointers) Addr field of the segment/gather list pointer. Addr is always a physical memory byte pointer. If WORD0[G] is clear or the pointer does not reside in WORD1 of the command, Addr points to the start of the packet data for this segment and can be any byte alignment. If WORD0[G] is set, Addr points to the gather list — an array of segment pointers — and must be 64-bit aligned (i.e. the least-significant three bits must be clear). If WORD0[LE] is set, Addr is a little-endian byte pointer. Otherwise, Addr is a big-endian byte pointer.

WORD2[PTP] When WORD0[R] is set, WORD2 exists and WORD2[PTP] indicates whether PKO takes a timestamp for the packet. When WORD0[R] is clear, WORD2[PTP] does not exist. When WORD2[PTP] exists and is set, PKO samples MIO_PTP_CLOCK_HI while it sends the packet, when required by the IEEE 1588 standards. The WORD0[Q] description above describes the two different ways that PKO can return the timestamp to software in this case.

WORD2[Ptr] A physical L2/DRAM byte pointer PKO uses after sending the packet. WORD2 does not exist when WORD0[R] is clear.

If both WORD0[Q,R] are set:

- WORD2[Ptr] points to a POW WQE and must be aligned on a 64-bit boundary.

If WORD0[Q] is clear while WORD0[R] is set:

- WORD2[Ptr] points to an L2/DRAM location that PKO will write after sending the packet.
- If WORD2[PTP] is clear, PKO will write one L2/DRAM 8-bit byte at WORD2[Ptr] to zero after sending the packet. In this case, WORD2[Ptr] is a big-endian pointer when PKO_REG_FLAGS[STORE_BE] is set, else a little-endian pointer, and may be any byte alignment.
- If WORD2[PTP] is set, PKO will write one L2/DRAM 64-bit word with a timestamp after sending the packet. WORD2[Ptr] must be aligned on a 64-bit boundary in this case.

The WORD0[Q] description above describes more details about PKO WORD2[Ptr] usage.

Cavium Confidential For Buwen Yang ALU 04/27/2014

10.5 PKO Queue-Arbitration Algorithm

When PKO needs to send a packet through a port, the PKO hardware uses a mixed-priority/weighted-fair algorithm to select from among the queues attached to the PKO-internal port (a variable number, up to 16 queues).

Write operations to PKO_MEM_QUEUE_PTRS configure the mapping of PKO output queues to PKO-internal ports. All queues that map to a single port must reside in a contiguous range of queue identifiers (QIDs). Boot-time mapping of queue QID to PKO-internal port PID via PKO_MEM_QUEUE_PTRS[QID*,PID,IDX*,TAIL] also establishes the position of the queue in the range for the port.

- IDX must be the relative position of QID in the range (i.e. QID – min(QID)).
- TAIL must be set only for the last queue in the range for the port, i.e. only for the one with the largest QID.

This output-queue-to-PKO-internal port configuration should be done only once at boot time before any output packets flow on the queues/ports.

Each queue has an associated QOS_MASK field. Software must configure QOS_MASK for each used queue at boot time via PKO_MEM_QUEUE_PTRS[QOS_MASK]. During normal operation, software can also optionally update the QOS_MASK fields for the queues via PKO_MEM_QUEUE_QOS[QOS_MASK]. QOS_MASK controls scheduling of the queue. When QOS_MASK is set to zero, the queue is disabled, and can no longer send packets through the PKO-internal port.

Backpressure from DPI output rings can zero the effective QOS_MASK for a DPI output queue. This prevents PKO from scheduling packets to the DPI output ring. When CN71XX DPI hardware backpressures PKO for a DPI ring, PKO zeroes the QOS_MASK of all PKO output queues that place packets in the DPI ring. Note that CN71XX DPI logic can attach multiple DPI output rings to a given PKO-external DPI port, and uses the relative position of the queue on the PKO-internal port (i.e. IDX) to select among the DPI output rings attached to a port. Section 12.2 describes the details.

The priority algorithm allows a configurable number of queues to be designated (static) high priority. When present, these priority queues must be the lowest-index queues attached to the PKO-internal port. The lower the queue index, the (statically) higher the priority of the queue. PKO visits these priority queues in order prior to visiting the remaining weighted-fair queues. All priority queues are higher-priority than the weighted-fair queues. The programmed QOS_MASK value for a priority queue must be either all 1s (0xFF) or all 0s.

All queues that are not priority queues are weighted-fair queues. The weighted-fair algorithm has eight rounds, where each weighted-fair queue can optionally participate in each round. With weighted-fair queues, QOS_MASK is a bit-mask that determines the rounds that a queue participates in. The fewer rounds that a queue participates in, i.e. the fewer bits set in QOS_MASK, the lower priority the queue is. Of course, all disabled queues (QOS_MASK is all 0s) never participate in any round, and any enabled queue must participate in at least one of the eight rounds.

Cavium Confidential For Power Point Only ALU 02/7/2014

To use static queues within a port:

- The static-priority queues must be the first queues attached to the PKO-internal port (highest priority first).
- PKO_MEM_QUEUE_PTRS[STATIC_P] must be set for all queues attached to the PKO-internal port.
- PKO_MEM_QUEUE_PTRS[STATIC_Q] must be set for the queues attached to the PKO-internal port that are static-priority queues.
- PKO_MEM_QUEUE_PTRS[S_TAIL] must be set for the last static-priority queue attached to the PKO-internal port.
- PKO_MEM_QUEUE_PTRS[QOS_MASK] for a static queue must be either all 1s (to enable the queue) or all 0s (to disable the queue).

In the pseudo-code below:

- **current_round** and **queue_within_round** are the current weighted round-robin state of the PKO-internal port.
- **max_queue_idx** is the number of queues attached to the PKO-internal port minus one.
- **queue_base** is the first queue attached to the PKO-internal port.
- **num_static_priority** is the number of static-priority queues attached to the PKO-internal port.
- **QOS_MASK** is the effective QOS_MASK for the queue. For queues that do not map to DPI, and for queues that do map to DPI but are currently not backpressured, it is the programmed PKO_MEM_QUEUE_PTRS[QOS_MASK]/ PKO_MEM_QUEUE_QOS[QOS_MASK]. For queues that map to DPI and are currently being backpressured, QOS_MASK is zero. (Section 12.2 describes the mapping of PKO output queues to DPI rings and when a ring backpressures.)
- **doorbell_count** is the doorbell count for the queue (Sections 10.3 and 10.11 explain the doorbell).

Cavium Confidential For Bowen Yang ALU 02/27/2014

The following pseudo-code shows the details of the algorithm:

```
struct {
   int current_round, queue_within_round;
   int queue_base;   int max_queue_idx;
   int num_static_priority;
} port_state[40];

struct {
   int QOS_MASK;
   int doorbell_count;
} queue_state[128];

// port_queue_arb() only runs when the attached PKO-internal port is
// selected for its engine and at least one queue attached
// to the selected port has a non-zero doorbell count and has an effective
// QOS_MASK that is non-zero.

// port_queue_arb() returns the queue to select from.
int port_queue_arb(int port) {
   bool found = false;
   int queue;
   int loops = 0;
   do {
     int rel_q;
     bool is_static = (loops < port_state[port].num_static_priority);
     if(is_static)
       rel_q = loops;
     else
       rel_q = port_state[port].queue_within_round;
     queue = port_state[port].queue_base + rel_q;
     bool participate = (queue_state[queue].QOS_MASK >> port_state[port].current_round) & 1;
     if(participate && queue_state[queue].doorbell_count)
       found = true;
     if(!is_static) {
       if(port_state[port].queue_within_round == port_state[port].max_queue_idx) {
          port_state[port].current_round = (port_state[port].current_round + 1) & 7;
          port_state[port].queue_within_round = port_state[port].num_static_priority;
       }
       else
          port_state[port].queue_within_round ++;
     }
     loops++;
   } while(!found);
   return(queue);
 }
```

For example, for a four-queue configuration with these requirements:

- one static-priority queue (highest priority)
- one (generally, but not statically) higher-priority queue
- two low-priority queues

One reasonable configuration might be:

- queue 0: static-priority queue, QOS_MASK = 0xFF
- queue 1: weighted-fair queue, QOS_MASK = 0xFF
- queue 2: weighted-fair queue, QOS_MASK = 0x80
- queue 3: weighted-fair queue, QOS_MASK = 0x08

Example 10–1 Four-Queue Configuration Example, After Queue 0 is Drained

		Rounds							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Queues	**1**	1	3	4	5	6	8	9	10
	2	2	–	–	–	–	–	–	–
	3	–	–	–	–	7	–	–	–

If all four queues had packets to send with this configuration, first all queue 0 packets would be sent. Then, after eight rounds, eight packets would be sent through queue 1, and one packet would be sent through each of queue 2 and queue 3. Also, as configured, four packets would flow through queue 1 between interruptions in the flow from queue 2 or queue 3. Example 10–1 indicates the order of packet transition after queue 0 is drained.

Cavium Confidential For Boven Yang 02/27/2014

10.6 PKO-Internal Port-to-Engine Arbitration Algorithm

When PKO needs to send a packet through an engine, the PKO hardware uses a mixed priority/weighted-fair algorithm to select from among the PKO-internal ports attached to the engine. Any combination of PKO-internal ports can map to an engine. The PKO-internal ports attached to an engine need not be contiguous.

A write operation to PKO_MEM_PORT_PTRS[PID,EID] maps PKO-internal port PID to engine EID. An engine can service any combination of PKO-internal ports. This configuration should be done only once at boot time before any output packets flow on the ports/engines. EID values 0–10 map to engines 0–10 and EID value 15 leaves the PID unmapped. EID values 11–14 are illegal.

The reset/default mapping from internal ports to engines shown in Table 10–1 is as shown in Figure 10–1:

Table 10–1 Reset/Default Internal Ports-to-Engines Mapping

PKO-Internal Ports	Map to Engine
0, 4, 8, 12	0
1, 5, 9, 13	1
2, 6, 10, 14	2
3, 7, 11, 15	3
16, 20, 24, 28	4
17, 21, 25, 29	5
18, 22, 26, 30	6
19, 23, 27, 31	7
32, 33, 34, 35	8
36, 37, 38, 39	9
40	10

NOTE: There is one important constraint on the mapping of PKO-internal ports to PKO-external ports. As Figure 10–1 indicates, engine 8 sends to all of the PKO-external ports destined for DPI and engine 9 sends to all of the PKO-external ports destined for loopback. For PKO-external ports destined for DPI and loopback, the lower two bits of the PKO-internal port ID (PID) selects the PKO-external port, which results in the mapping as shown in Table 10–2.

Table 10–2 PKO-Internal Port to PKO-External Port Mapping for DPI and Loopback

PKO-External Ports	Serviced Only by PKO-Internal Ports
32 and 36	0, 4, 8, ... 32, 36
33 and 37	1, 5, 9, ... 33, 37
34 and 38	2, 6, 10, ... 34, 38
35 and 39	3, 7, 11, ... 35, 39

Each PKO-internal port has an associated QOS_MASK field. Software must configure QOS_MASK for each used PKO-internal port at boot time via PKO_MEM_PORT_PTRS[QOS_MASK]. During normal operation, software can also optionally update the QOS_MASK fields for the ports via PKO_MEM_PORT_QOS[QOS_MASK]. QOS_MASK controls scheduling of the port to the attached engine. When QOS_MASK is set to zero, the port is disabled, and can no longer send packets through the attached engine.

Cavium Confidential For Bowen Yang ALU 02/27/2014

The priority algorithm allows a configurable number of the PKO-internal ports that are attached to an engine to be designated (static) high priority. When present for an engine, these priority ports must be the lowest-index ports attached to the engine. Note that the high-priority ports attached to a single engine need not be at contiguous PIDs. The same is true for multiple engines: the high-priority ports need not be contiguous. The only requirement is that all high-priority ports attached to a single engine are lower PID than all the remaining non-priority ports attached to the same engine. The lower the PID, the (statically) higher the priority of the port. PKO visits these priority ports in order prior to visiting the remaining weighted-fair ports. All priority ports are higher-priority than the weighted-fair ports. The QOS_MASK value for a priority port must be either all 1s (0xFF) or all 0s. A PKO-internal port PID is static priority when PKO_MEM_PORT_PTRS[STATIC_P] is set for the port during configuration.

All PKO-internal ports that are not priority ports are weighted-fair ports. The weighted-fair algorithm has eight rounds, like the queue arbitration algorithm, where each weighted-fair port can optionally participate in each round. With weighted-fair ports, QOS_MASK is a bit-mask that determines the rounds that a PKO-internal port participates in. The fewer rounds that a port participates in, i.e. the fewer bits set in QOS_MASK, the lower priority the port is. Note that this per-port QOS affects PKO scheduling of packets, regardless of the size of the packets. Of course, all disabled ports (QOS_MASK is all 0s) never participate in any round, and any enabled port must participate in at least one of the eight rounds.

The reset/default state has all PKO-internal ports as weighted-fair ports with QOS_MASK=0xFF.

The PKO can also rate-limit and receive backpressure for PKO-internal ports. Refer to Section 10.8 and Section 10.9 below.

The following pseudo-code shows the details of the algorithm that selects the PKO-internal port for the engine. Note that the priority and weighted-fair scheduling is completely separate for each engine:

```
struct {
   int current_port;
   int current_round;
} engine_state[11];

struct {
   int doorbell_sum; // sum of the doorbell counts for all queues
                     // that are attached to this port and enabled
   bool STATIC_P;
   int QOS_MASK;
   int EID;
   bool rate_limited; // Refer to Section 10.9
   bool backpressure; // Refer to Section 10.8
} port_state[41];

// engine_port_arb() only runs when there is at least one
// attached port that:
//  - has a QOS_MASK that is non-zero (i.e. is enabled), and
//  - is not currently rate-limited, and
//  - is not currently backpressured, and
//  - has at least one queue attached that has a non-zero
//    doorbell count and is enabled

int engine_port_arb(int eng) {
   bool found = false;
   int port;
   int first_non_static_port;
   for(port = 0; port < 41; port++) {
      if(port_state[port].EID == eng) {
         if(port_state[port].STATIC_P) {
            first_non_static_port = port;
```

Cavium Confidential For Bowen Yang ALU 02/21/2014

```
            break;
         }
         else if(port_state[port].doorbell_sum
                 && !port_state[port].rate_limited
                 && !port_state[port].backpressure) {
            found = true;
            break;
         }
      }
   }
   while(!found) {
      port = engine_state[eng].current_port;
      if(port_state[port].EID == eng) {
         int QOS_MASK = port_state[port].QOS_MASK;
         int current_round = engine_state
         if(((QOS_MASK >> current_round) & 0x1)
                  && port_state[port].doorbell_sum
                  && !port_state[port].rate_limited
                  && !port_state[port].backpressure) {
            found = true;
         }
      }
      if(port == 41-1) {
         engine_state[eng].current_port = first_non_static_port;
         engine_state[eng].current_round =
            (engine_state[eng].current_round + 1) & 7;
      }
      else
         engine_state[eng].current_port = port + 1;
   }
   return(port);
}
```

After selecting a PKO-internal port for the engine as in this pseudo-code, PKO must further select a particular output queue that is attached to the PKO-internal port. Refer to Section 10.5.

10.7 PKO Don't-Write-Back (DWB) Calculation

A hardware pool free command includes a don't-write-back (DWB) argument that specifies the maximum number of don't-write-back commands to execute on the coherent memory bus. When PKO_REG_FLAGS[ENA_DWB] is clear, these DWB counts will always be zero.

If PKO_REG_FLAGS[ENA_DWB] is set, the DWB count for output queue chunks is:

((PKO_REG_CMD_BUF[SIZE] × 8) + 127) >> 7

If PKO_REG_FLAGS[ENA_DWB] is set, the DWB count for the buffer containing a segment that is freed is:

(((**Addr** & 127) + **Size** + 127) >> 7) + **Back**

which may result in a coherent memory bus DWB command for all cache blocks from the start of the buffer up to the block that includes the last byte of the segment.

- **Addr** and **Back** here come directly from the segment pointer.
- **Size** is always the number of bytes in the segment or gather list that are used by PKO.
  - For all gather lists and all segments except the last one for a packet, **Size** is the number of bytes in the segment/gather list.
  - For the last segment of a packet, **Size** may be less than the number of bytes in the segment since the number of bytes used in the segment can be limited by the total number of bytes in the packet.

Cavium Confidential For Bowen Yang ALI 02/27/2014

10.8 PKO-External Port-Level Backpressure

PKO receives backpressure from up to 52 PKO-external backpressure sources (16 from packet interface 0, 16 from packet interface 1, 32 from DPI output rings, and 4 from PIP/IPD (loopback)). PKO interprets most of these backpressures solely as port-level backpressures—PKO stops sending new packets from selected PKO-internal ports upon receipt of the port-level backpressure.

Note that backpressure only causes PKO to stop sending on packet boundaries, never in the middle of packets. (This is true for both queue-level and port-level backpressure.)

Each PKO-internal port (PID) optionally obeys backpressure from one external backpressure source, selected by PKO_MEM_PORT_PTRS[BP_PORT]. Table 10–3 indicates the selectable external backpressure sources:

Table 10–3 Selectable External Backpressure Sources

PKO_MEM_PORT_PTRS [BP_PORT]	PKO-External Source
0–15	Packet Interface 0, logical link port backpressure. Backpressure asserts for BP_PORT *i* when FINAL_XOF<i> is set for packet interface 0. Refer to Sections 22.3.2.2 (PFC) and 22.3.2.3.2 (HiGig2 messages).
16–31	Illegal
32	DPI port 32. Port-level backpressure occurs when all DPI packet output rings attached to PKO-external port 32 backpressure. Refer to Section 12.2.
33	DPI port 33. Port-level backpressure occurs when all DPI packet output rings attached to PKO-external port 33 backpressure. Refer to Section 12.2.
34	DPI port 34. Port-level backpressure occurs when all DPI packet output rings attached to PKO-eternal port 34 backpressure. Refer to Section 12.2.
35	DPI port 35. Port-level backpressure occurs when all DPI packet output rings attached to PKO-external port 35 backpressure. Refer to Section 12.2.
36	Loopback port 36 (PIP/IPD). Refer to per-port backpressure in Section 9.6.
37	Loopback port 37 (PIP/IPD). Refer to per-port backpressure in Section 9.6.
38	Loopback port 38 (PIP/IPD). Refer to per-port backpressure in Section 9.6.
39	Loopback port 39 (PIP/IPD). Refer to per-port backpressure in Section 9.6.
40–62	Illegal
63	None—PKO-internal port is never backpressured.

The reset/default setting is BP_PORT = PID for all PKO-internal ports. Though PKO_MEM_PORT_PTRS[BP_PORT] programming provides many backpressure options, PKO_MEM_PORT_PTRS[BP_PORT] values should be constrained. The following bullets describe the values that should be used.

Cavium Confidential For Boyer Yang 02/27/2011 ALL

- BP_PORT values 0–15 correspond to RXAUI packet interface 0. Note that interface-level backpressure is handled separately. Applications can assert finer-grained backpressure on individual PKO-internal ports using the GMX*n*_TX_OVR_BP[TX_PRT_BP] fields. It is also possible to receive logical-link backpressure from an attached HiGig2 or PFC device. A PKO-internal port that is destined for a packet interface should normally select BP_PORT values corresponding to the interface or select no backpressure. That is, if PKO_MEM_PORT_PTRS[EID] is 0..3, PKO_MEM_PORT_PTRS[BP_PORT] should be either 0–15 or 63.
- The DPI packet output ring backpressures are exceptional because the supplied backpressures are primarily for individual PKO output queues, not ports. Section 10.5 describes how PKO stops scheduling packets from individual PKO output queues when a destination DPI ring backpressures. Section 12.2 describes which DPI rings are the destination for PKO output queues that are destined to a PKO-external DPI port, and when an individual DPI ring backpressures. PKO considers a PKO-external DPI port to be backpressured (BP_PORT values 32..35) when all attached DPI packet output rings backpressure. A PKO-internal port PID that is destined for a PKO-external DPI port should always select a BP_PORT value corresponding to the PKO-external port. That is, if PKO_MEM_PORT_PTRS[EID] = 0x8, then PKO_MEM_PORT_PTRS[BP_PORT] should always equal 32 + PID<1:0>.
- BP_PORT values 36–39 correspond to PIP/IPD per-port backpressure for the corresponding port. A PKO-internal port (PID) that is destined for a loopback port should normally select a BP_PORT value corresponding to the loopback port or select no backpressure. That is, if PKO_MEM_PORT_PTRS[EID] = 9, then PKO_MEM_PORT_PTRS[BP_PORT] should always equal 36 + PID<1:0> or 63.

10.9 PKO-Internal Port Rate-Limiting

PKO optionally allows the rate at which PKO-internal ports send packet data to be limited. The solution includes both a fixed cost per eight bytes (PKO_MEM_PORT_RATE0[RATE_WORD]) and a fixed cost per packet (PKO_MEM_PORT_RATE0[RATE_PKT]), as well as an accumulated limit (PKO_MEM_PORT_RATE1[RATE_LIM]). Note that this per-port rate-limiting only causes PKO to stop sending on packet boundaries, never in the middle of packets.

PKO includes a 32-bit accumulator per PKO-internal port. The accumulator sums up the overheads of the previously-sent packets. Think of <3:0> of the accumulator as indicating the fraction of cycles and <31:4> as indicating cycles. Conceptually, PKO decrements <31:4> of the accumulator by one every cycle that it is larger than zero. This emulates a constant packet transmit rate.

On a packet start, PKO increments the accumulator for the PKO-internal port by

$$(256 \times \text{PKO_MEM_PORT_RATE0[RATE_PKT]}) + (N \times \text{PKO_MEM_PORT_RATE0[RATE_WORD]}),$$

where N = (len+7)/8 and len is the number of bytes in the packet that PKO transfers. This accumulates the fixed-cost per-packet at a granularity of 16 coprocessor-clock cycles and the per-byte cost at a granularity of 1/16th a coprocessor-clock cycle per eight bytes.

PKO stops new packet transmissions on a PKO-internal port due to rate limiting whenever <31:8> of the accumulator is larger than PKO_MEM_PORT_RATE1[RATE_LIM].

Cavium Confidential For Bowen Yang ATD 8/12/2014

The default/reset zero values for PKO_MEM_PORT_RATE0[RATE_PKT,RATE_WORD] and PKO_MEM_PORT_RATE1[RATE_LIM] (all zero) never rate limits the internal port. Software can change the PKO_MEM_PORT_RATE0[RATE_PKT,RATE_WORD] and PKO_MEM_PORT_RATE1[RATE_LIM] values dynamically during operation.

To a large extent, PKO rate-limiting works like a leaky bucket. The bucket leaks "water" at the constant rate of 16 units per coprocessor-clock cycle. Each packet adds a programmable amount of "water" to the bucket, so a unit can correspond to either a higher or lower constant data-transfer rate. CN71XX PKO stops adding new "water" (i.e. stops sending new packets) when the "water" is near the top of the bucket. PKO_MEM_PORT_RATE0[RATE_WORD,RATE_PKT] indicate the volume of "water" to add for a packet. PKO_MEM_PORT_RATE1[RATE_LIM] indicates the volume of the bucket.

Suppose the coprocessor-clock speed is 1 GHz. The following would be a typical configuration to rate-limit a PKO-internal port to 1 Gbit/second, for example:

First, program the cost per 8 bytes of the packet. 8 bytes = 64 bits, which transfers in Cycper8 = 64 cycles in this configuration. A RATE_WORD value of 1 is 1/16-cycle overhead per 8 bytes, so program:

PKO_MEM_PORT_RATE0[RATE_WORD] = Cycper8×16 = 64×16 = 1024

Second, program the overhead per packet. In a typical 1 Gbit/s configuration, the CN71XX SGMII / 1000BASE-X logic prepends an 8-byte PREAMBLE/SFD, appends a 4B FCS, and enforces a 12 byte inter-packet gap (IPG). (Refer to Chapter 23.) This totals 24 bytes, or 192 bits, which transfers in OverCyc = 192 cycles in this configuration. A RATE_PKT value of one adds a 16-cycle overhead, so program:

PKO_MEM_PORT_RATE0[RATE_PKT] = OverCyc/16 = 192/16 = 12

Finally, program the accumulator limit. The desired accumulator limit may vary for each situation, so it is difficult to predict an appropriate value. Suppose ~4KB of packet data is the appropriate value. 4KB is 32 Kbits, which transfers in LimCyc = 32 K cycles in this configuration. A RATE_LIM value of one sets the limit to 16 cycles, so program:

PKO_MEM_PORT_RATE1[RATE_LIM] = LimCyc/16 = 32K/16 = 2048

Scale the numbers down with a slower coprocessor clock or a faster data rate. Scale PKO_MEM_PORT_RATE1[RATE_LIM] for the desired accumulated limit.

Cavium Confidential For Bowen Yang ALU 6/12/2014

10.10 PKO Performance

PKO can send 25 million or more small packets per second when many engines are used. When only a single engine is used, PKO can send 13–25 million or more small packets per second through the single engine. PKO performance is better when the output (command) queue, gather lists, and packet data reside in the L2 cache rather than in DRAM.

The amount of internal buffering available for each engine affects the PKO behavior when it inserts a TCP/UDP checksum. Table 10–4 shows the internal buffering available for each engine based on the number of PKO-external ports that exist in the interface, as selected by PKO_REG_GMX_PORT_MODE[MODE*n*]. If one of the two packet interfaces is not used, the packet interface that is used gets a total of 20.0 KB for buffering.

Table 10–4 Internal Buffer Space

Packet Interface Ports	Per-Engine
4	2.5 KB
2	5.0 KB
1	10.0 KB

If the command for a packet directs the PKO engine to calculate a TCP/UDP checksum, and the entire packet fits inside the internal buffer space allocated to the engine, then the PKO engine reads the packet once, buffers it internally and calculates the checksum. It then inserts the checksum as it reads the packet from its internal buffer and sends it out a PKO-external port. The PKO engine cannot send any part of the packet out the POB until it reads all packet data from L2/DRAM.

If the command for a packet directs the PKO engine to calculate a TCP/UDP checksum, but the entire packet does not fit inside the internal buffer space allocated to the engine, then the PKO engine reads the packet twice from L2/DRAM. It calculates the TCP/UDP checksum during the first read. The data from the second read flows through the internal buffering and immediately out the POB, and the checksum logic inserts the checksum as it is sent out. In this case, the PKO engine starts sending the packet out the POB before it finishes reading the packet the second time.

If the command for a packet does not direct the PKO engine to calculate a TCP/UDP checksum, the PKO engine only reads the packet data from L2/DRAM once and the data flows through the internal buffering to the POB. In this case, the PKO engine starts sending the packet out the POB before it finishes reading the packet.

PKO has two performance counters for each engine: one that counts the number of packets sent out the engine, and another that counts the number of bytes sent out the engine. The packet counts can be read through PKO_MEM_COUNT0 (after setting up PKO_REG_READ_IDX), and the byte counts can be read through PKO_MEM_COUNT1 (again, after setting up PKO_REG_READ_IDX).

Cavium Confidential For Bowen Yang ALU 02/27/2014

10.11 PKO Operations

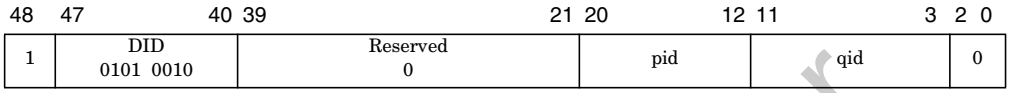
This section shows the detailed bit formats and codes for the various PKO transactions.

NOTE: Only 64-bit store operations are allowed (i.e. only SDs are allowed); no load or IOBDMA operations.

10.11.1 Store Operations

Doorbell Write Operations

Physical Address to Store to PKO



- **qid** - queue identifier (0–255 legal)
- **pid** - PKO-internal port identifier (0–40 legal). The PKO-internal port identifier must match the PKO-internal port previously attached to the selected queue. (The PKO-internal port was attached to a queue by a previous write operation for the queue to PKO_MEM_QUEUE_PTRS.)

Store Data on a Store to PKO

- **wdc** - contains the number of words to add to the doorbell count

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

10.12 PKO Registers

The PKO registers are listed in Table 10–5.

Table 10–5 PKO Registers

Register	Address	CSR Type[1]	Detailed Description
PKO_REG_FLAGS	0x0001180050000000	RSL	See page 572
PKO_REG_READ_IDX	0x0001180050000008	RSL	See page 572
PKO_REG_CMD_BUF	0x0001180050000010	RSL	See page 572
PKO_REG_GMX_PORT_MODE	0x0001180050000018	RSL	See page 573
PKO_REG_QUEUE_MODE	0x0001180050000048	RSL	See page 574
PKO_REG_ENGINE_INFLIGHT	0x0001180050000050	RSL	See page 574
PKO_REG_ENGINE_THRESH	0x0001180050000058	RSL	See page 575
PKO_REG_TIMESTAMP	0x0001180050000060	RSL	See page 575
PKO_REG_BIST_RESULT	0x0001180050000080	RSL	See page 576
PKO_REG_ERROR	0x0001180050000088	RSL	See page 576
PKO_REG_INT_MASK	0x0001180050000090	RSL	See page 576
PKO_REG_DEBUG(0..3)	0x0001180050000098 + α*0x8	RSL	See page 577
PKO_REG_QUEUE_PTRS1	0x0001180050000100	RSL	See page 577
PKO_REG_QUEUE_PREEMPT	0x0001180050000108	RSL	See page 578
PKO_REG_PREEMPT	0x0001180050000110	RSL	See page 578
PKO_MEM_QUEUE_PTRS	0x0001180050001000	RSL	See page 579
PKO_MEM_QUEUE_QOS	0x0001180050001008	RSL	See page 580
PKO_MEM_PORT_PTRS	0x0001180050001010	RSL	See page 580
PKO_MEM_PORT_QOS	0x0001180050001018	RSL	See page 582
PKO_MEM_PORT_RATE0	0x0001180050001020	RSL	See page 583
PKO_MEM_PORT_RATE1	0x0001180050001028	RSL	See page 583
PKO_MEM_COUNT0	0x0001180050001080	RSL	See page 583
PKO_MEM_COUNT1	0x0001180050001088	RSL	See page 584
PKO_MEM_DEBUG0	0x0001180050001100	RSL	See page 584
PKO_MEM_DEBUG1	0x0001180050001108	RSL	See page 584
PKO_MEM_DEBUG2	0x0001180050001110	RSL	See page 585
PKO_MEM_DEBUG3	0x0001180050001118	RSL	See page 585
PKO_MEM_DEBUG4	0x0001180050001120	RSL	See page 585
PKO_MEM_DEBUG5	0x0001180050001128	RSL	See page 586
PKO_MEM_DEBUG6	0x0001180050001130	RSL	See page 586
PKO_MEM_DEBUG7	0x0001180050001138	RSL	See page 587
PKO_MEM_DEBUG8	0x0001180050001140	RSL	See page 587
PKO_MEM_DEBUG9	0x0001180050001148	RSL	See page 588
PKO_MEM_DEBUG10	0x0001180050001150	RSL	See page 588
PKO_MEM_DEBUG11	0x0001180050001158	RSL	See page 588
PKO_MEM_DEBUG12	0x0001180050001160	RSL	See page 589
PKO_MEM_DEBUG13	0x0001180050001168	RSL	See page 589
PKO_MEM_DEBUG14	0x0001180050001170	RSL	See page 589

1. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PKO Flags Register
PKO_REG_FLAGS

This register contains flags to control the PKO unit. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8>	DIS_PERF3	R/W	0	0	Set to disable inactive queue QOS skipping.
<7>	DIS_PERF2	R/W	0	0	Set to disable inactive queue skipping
<6:4>	—	RAZ	—	—	Reserved
<3>	RESET	RAZ	0	0	Reset PKO. When set to 1, causes a four-cycle reset pulse to the entire PKO unit.
<2>	STORE_BE	R/W	0	0	Force to big endian. When set to 1, inverts bits[2:0] of all zero-byte write addresses.
<1>	ENA_DWB	R/W	0	0	Enable don't-write-backs. When set to 1, enables the use of Don't-Write-Backs during the buffer freeing operations.
<0>	ENA_PKO	R/W	0	0	Enable the PKO picker. When set to 1, enables the PKO picker and places the PKO in normal operation.

PKO Read Index Register
PKO_REG_READ_IDX

This register provides the read index during a CSR read operation to any of the CSRs that are physically stored as memories (i.e. the CSRs that begin with the prefix PKO_MEM_). Bits IDX[7:0] are the read index. INC[7:0] is an increment that is added to IDX[7:0] after any CSR read. The intended use is to initially write this CSR such that IDX = 0x0 and INC = 0x1. Then, the entire contents of a CSR memory can be read with consecutive CSR read commands. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:8>	INC	R/W	0x0	0x0	Increment to add to current index for next index.
<7:0>	IDX	R/W	0x0	0x0	Index to use for next memory CSR read.

PKO Command Buffer Parameters Register
PKO_REG_CMD_BUF

This register sets the command buffer parameters. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:23>	—	RAZ	—	—	Reserved
<22:20>	POOL	R/W	0x0	0x0	Pool number. Specifies the free list used to free command-buffer segments
<19:13>	—	R/W	0x0	0x0	Reserved. Must be zero.
<12:0>	SIZE	R/W	0x0	0x0	Size. Specifies the number of uint64s per command-buffer segment

Cavium Confidential For Bowen Yang ALU 02/27/2014

PKO GMX Port Mode Register
PKO_REG_GMX_PORT_MODE

This register sets the number of GMX engines and amount of local storage per GMX engine. The system can have a total of 11 engines (GM0 (4) + GM1 (4) + DPI (1) + LOOP (1) + AGL (1)). This register has no effect on the amount of local storage for the AGL, DPI, and loopback engines (always 2.5 KB).

CN71XX has 27.5 KB of local storage, with 2.5 KB available for the AGL engine, 2.5 KB available for the DPI engine, 2.5 KB for the loopback engine, 10 KB for the GMX0 engines, and 10 KB for the GMX1 engines.

When all the GMX engines are used (i.e. MODE0 = 0x2), each engine has 2.5 KB of local storage. Increasing the value of MODE0 by 1 decreases the number of GMX engines by a power of 2 and increases the local storage per GMX engine by a power of 2. MODE0 values of 0x0 and 0x1 should not be used, and if selected, are treated as 0x2. Table 10–6 shows the storage for the GMX interface.

Table 10–6 GMX Per-Engine Local Storage

MODE0/1 Value	Number of GMX0/1 Engines	GMX0/1 Storage/ Engine	GMX0 Storage MODE1=5	GMX1 Storage MODE0=5	DPI Storage/ Engine	LOOP Storage/ Engine	AGL Storage/ Engine
0x0	4	2.5 KB	5.0 KB	5.0 KB	2.5 KB	2.5 KB	2.5 KB
0x1	4	2.5 KB	5.0 KB	5.0 KB	2.5 KB	2.5 KB	2.5 KB
0x2	4	2.5 KB	5.0 KB	5.0 KB	2.5 KB	2.5 KB	2.5 KB
0x3	2	5.0 KB	10.0 KB	10.0 KB	2.5 KB	2.5 KB	2.5 KB
0x4	1	10.0 KB	20.0 KB	20.0 KB	2.5 KB	2.5 KB	2.5 KB

See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:3>	MODE1	R/W	0x0	0x0	Number of GM1 engines. 0x0 = 4 engines 0x1 = 4 engines 0x2 = 4 engines 0x3 = 2 engines 0x4 = 1 engine 0x5 = GMX1 is disabled 0x6 = reserved 0x7 = reserved When MODE1 is 0x5, GMX0's storage is doubled (20.0 KB).
<2:0>	MODE0	R/W	0x0	0x0	Number of GM0 engines. 0x0 = 4 engines 0x1 = 4 engines 0x2 = 4 engines 0x3 = 2 engines 0x4 = 1 engine 0x5 = GMX0 is disabled 0x6 = reserved 0x7 = reserved When MODE0 is 0x5, GMX1's storage is doubled (20.0 KB).

Cavium Confidential For Bowen Yang ALU 02/27/2014

PKO Queue Mode Register
PKO_REG_QUEUE_MODE

This register sets the number of queues and amount of local storage per queue. The system has a total of 256 queues and (256 × 8) words of local command storage. This register sets the number of queues that are used. Increasing the value of MODE by 1 decreases the number of queues by a power of 2 and increases the local storage per queue by a power of 2. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	X	0x0	Reserved
<1:0>	MODE	R/W	0x0	0x0	Specifies the PKO queue mode. **Value** **Queues** **Storage/queue** 0x0 256 64 bytes (8 words) 0x1 128 128 bytes (16 words) 0x2 64 256 bytes (32 words) 0x3 Reserved

PKO In-flight Packets Register
PKO_REG_ENGINE_INFLIGHT

This register sets the maximum number of in-flight packets, per engine. Values greater than four are illegal. Setting an engine's value to zero effectively stops the engine. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:44>	—	RAZ	—	—	Reserved
<43:40>	ENGINE10	R/W	0x4	0x4	Maximum number of in-flight packets for engine10
<39:36>	ENGINE9	R/W	0x4	0x4	Maximum number of in-flight packets for engine9
<35:32>	ENGINE8	R/W	0x4	0x4	Maximum number of in-flight packets for engine8
<31:28>	ENGINE7	R/W	0x4	0x4	Maximum number of in-flight packets for engine7
<27:24>	ENGINE6	R/W	0x4	0x4	Maximum number of in-flight packets for engine6
<23:20>	ENGINE5	R/W	0x4	0x4	Maximum number of in-flight packets for engine5
<19:16>	ENGINE4	R/W	0x4	0x4	Maximum number of in-flight packets for engine4
<15:12>	ENGINE3	R/W	0x4	0x4	Maximum number of in-flight packets for engine3
<11:8>	ENGINE2	R/W	0x4	0x4	Maximum number of in-flight packets for engine2
<7:4>	ENGINE1	R/W	0x4	0x4	Maximum number of in-flight packets for engine1
<3:0>	ENGINE0	R/W	0x4	0x4	Maximum number of in-flight packets for engine0

Cavium Confidential For Bowen Yang AEO 04/27/2014

PKO Engine Packet-Send Threshold Register
PKO_REG_ENGINE_THRESH

This register enables/disables the packet-send threshold for the PKO engines. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved
<13:0>	MASK	R/W	0x0	0x0	MASK<*n*> = 0 disables packet-send threshold for engine *n*. MASK<*n*> = 1 enables packet-send threshold for engine *n*. Bits <13:11> are unused. When disabled, packet data may be sent as soon as it is written into PKO's internal buffers. When enabled and the packet fits entirely in the PKO's internal buffer, none of the packet data is sent until all of it has been written into the PKO's internal buffer. Note that a packet is considered to fit entirely only if the packet's size is ≤ BUFFER_SIZE – 8. When enabled and the packet does not fit entirely in the PKO's internal buffer, none of the packet data is sent until at least BUFFER_SIZE – 256 bytes of the packet have been written into the PKO's internal buffer. **NOTE:** BUFFER_SIZE is a function of PKO_REG_GMX_PORT_MODE).

PKO Timestamp Register
PKO_REG_TIMESTAMP

See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	WQE_WORD	R/W	0x2	0x2	Specifies the eight-byte word in the work-queue entry to which a PTP timestamp is written. Values 0x0 and 0x1 are illegal.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PKO BIST Result Register
PKO_REG_BIST_RESULT

This register provides access to the internal BIST results. Each bit is the BIST result of an individual memory (per bit, 0 = pass and 1 = fail). See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:30>	—	RAZ	—	—	Reserved
<29>	CSR	RO	—	0	BIST result of the CSR memories
<28>	IOB	RO	—	0	BIST result of the IOB memories
<27>	OUT_DAT	RO	—	0	BIST result of OUT_DAT memories
<26>	OUT_CTL	RO	—	0x0	BIST result of the OUT_CTL memories
<25>	OUT_STA	RO	—	0	BIST result of the OUT_STA memories
<24>	OUT_WIF	RO	—	0	BIST result of the OUT_WIF memories
<23:21>	PRT_CHK	RO	—	0x0	BIST result of the PRT_CHK memories
<20>	PRT_NXT	RO	—	0	BIST result of the PRT_NXT memories
<19:12>	PRT_PSB	RO	—	0x0	BIST result of the PRT_PSB memories
<11>	NCB_INB	RO	—	0x0	BIST result of the NCB_INB memories
<10>	PRT_QCB	RO	—	0x0	BIST result of the PRT_QCB memories
<9:8>	PRT_QSB	RO	—	0x0	BIST result of the PRT_QSB memories
<7:6>	PRT_CTL	RO	—	0x0	BIST result of PRT_CTL memories
<5:4>	DAT_DAT	RO	—	0x0	BIST result of the DAT_DAT memories
<3:0>	DAT_PTR	RO	—	0x0	BIST result of the DAT_PTR memories

PKO Error Register
PKO_REG_ERROR

This register contains error flags for PKO. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved
<2>	CURRZERO	R/W1C/H	0	0	A packet-data pointer has size = 0.
<1>	DOORBELL	R/W1C/H	0	0	A doorbell count has overflowed
<0>	PARITY	R/W1C/H	0	0	A read-parity error has occurred in the port data buffer.

PKO Interrupt Mask Register
PKO_REG_INT_MASK

This register contains interrupt-enable mask bits for the PKO errors listed in PKO_REG_ERROR. When a mask bit is set, the corresponding interrupt is enabled. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved
<2>	CURRZERO	R/W/H	0	0	Interrupt-enable bit for packet-data pointer = 0.
<1>	DOORBELL	R/W/H	0	0	Interrupt-enable bit for doorbell overflow errors.
<0>	PARITY	R/W/H	0	0	Interrupt-enable bit for read-parity errors.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PKO Debug Registers
PKO_REG_DEBUG(0..3)

Internal debug registers. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	ASSERTS	RO	0x0	0x0	Various assertion checks.

PKO Interrupt Mask Register
PKO_REG_QUEUE_PTRS1

This register is used with PKO_MEM_QUEUE_PTRS and PKO_MEM_QUEUE_QOS to allow access to queues 128–255, and to allow mapping of up to 16 queues per port.

NOTE: When programming queues 128–255, the programming sequence must first write PKO_REG_QUEUE_PTRS1 and then write PKO_MEM_QUEUE_PTRS or PKO_MEM_QUEUE_QOS for each queue. See the descriptions of PKO_MEM_QUEUE_PTRS and PKO_MEM_QUEUE_QOS for further explanation of queue programming.

See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	IDX3	R/W/H	0	0	Bit[3] of Index (distance from head) in the queue array. See PKO_MEM_QUEUE_PTRS[IDX].
<0>	QID7	R/W/H	0	0	Bit[7] of the Queue ID. See PKO_MEM_QUEUE_PTRS[QID]

Cavium Confidential
Bowen Yang
ALU
02/27/2014

PKO Queue Preempt Register
PKO_REG_QUEUE_PREEMPT

This register is used with PKO_MEM_QUEUE_PTRS and PKO_REG_QUEUE_PTRS1. When programming queues, the programming sequence must first write PKO_REG_QUEUE_PREEMPT, then PKO_REG_QUEUE_PTRS1, and then PKO_MEM_QUEUE_PTRS for each queue. Preemption is supported only on queues that are ultimately mapped to engines 0–7. It is illegal to set PREEMPTEE or PREEMPTER for a queue that is ultimately mapped to engines 8–9. See the descriptions of PKO_MEM_QUEUE_PTRS for further explanation of queue programming. Also, PKO_REG_ENGINE_INFLIGHT must be at least 0x2 for any engine on which preemption is enabled.

Per QID, setting both PREEMPTER = 1 and PREEMPTEE = 1 is illegal and sets only PREEMPTER = 1.

See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	PREEMPTEE	R/W/H	0	0	Allow this QID to be preempted. 0 = Cannot be preempted. 1 = Can be preempted.
<0>	PREEMPTER	R/W/H	0	0	Preempts the servicing of packet on PID to allow this QID immediate servicing. 0 = Do not cause preemption 1 = Cause preemption. Per PID, at most one QID can have this bit set.

PKO Preempt Register
PKO_REG_PREEMPT

See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	MIN_SIZE	R/W	0x0	0x0	Threshold for packet preemption, measured in bytes. Only packets which have at least MIN_SIZE bytes remaining to be read can be preempted.

Cavium Confidential For Bowen Yang ALU 07/7/2014

PKO Memory-Queue Pointers Register
PKO_MEM_QUEUE_PTRS

NOTE: This register is used with PKO_REG_QUEUE_PTRS1 to allow access to queues 128–255 and to allow mapping of up to 16 queues per port.

This register sets the queue-to-port mapping and the initial command-buffer pointer per queue. Each queue can map to at most one port, and no more than 16 queues can map to a port. The set of queues that is mapped to a port must be a contiguous array of queues.

- The queue designated in QID is mapped to the port designated in PID.
- The index of queue QID in port PID's queue list is IDX.
- The last queue in port PID's queue array must have its TAIL bit set.
- Unused queues must be mapped to port 63.

STATIC_Q marks queue QID as having static priority. STATIC_P marks the port PID to which QID is mapped as having at least one queue with static priority.

If any QID that maps to PID has static priority, then all QID that map to PID must have STATIC_P set. Queues marked as static priority must be contiguous and begin at IDX 0. The last queue that is marked as having static priority must have its S_TAIL bit set.

This CSR is a memory of 256 entries and consequently the PKO_REG_READ_IDX register must be written before any CSR read operations to this address can be performed. A read of any entry that has not been previously written is illegal and results in unpredictable CSR read data. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	S_TAIL	R/W/H	—	0	Set if this QID is the tail of the static queues
<62>	STATIC_P	R/W/H	—	0	Set if any QID in this PID has static priority
<61>	STATIC_Q	R/W/H	—	0	Set if this QID has static priority
<60:53>	QOS_MASK	R/W/H	—	0x0	Mask to control priority across eight QOS rounds.
<52:17>	BUF_PTR	R/W/H	—	0x0	Command buffer pointer, bits <23:17> must be 0x0.
<16>	TAIL	R/W/H	—	0	Set if this QID is the tail of the queue array.
<15:13>	IDX	WO	—	0x0	Index[2:0] (distance from head) in the queue array.
<12:7>	PID	WO	—	0x0	Port ID to which this queue is mapped.
<6:0>	QID	R/W/H	—	0x0	Queue ID[6:0].

Cavium Confidential For Bowen Yang ALU 02/12/2014

PKO Memory Queue QoS Bits Register
PKO_MEM_QUEUE_QOS

NOTE: This register is used with PKO_REG_QUEUE_PTRS1 to allow access to queues 128–255 and to allow mapping of up to 16 queues per port.

This register sets the QOS mask per queue. The QOS_MASK field is logically and physically the same QOS_MASK field in PKO_MEM_QUEUE_PTRS. This CSR allows the QOS_MASK bits to be written during PKO operation without affecting any other queue state. The port to which queue QID is mapped is port PID. Note that the queue-to-port mapping must be the same as was programmed via PKO_MEM_QUEUE_PTRS.

This CSR is a memory of 256 entries, and thus, the PKO_REG_READ_IDX CSR must be written before any CSR read operations to this address can be performed. A read of any entry that has not been previously written is illegal and results in unpredictable CSR read data. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:61>	—	RAZ	—	—	Reserved
<60:53>	QOS_MASK	R/W/H	—	0x0	Mask to control priority across eight QOS rounds
<52:13>	—	RAZ	—	—	Reserved
<12:7>	PID	WO	—	0x0	Port ID to which this queue is mapped
<6:0>	QID	R/W/H	—	0x0	Queue ID

PKO Memory Port Pointers Register
PKO_MEM_PORT_PTRS

This register sets the port-to-engine mapping, per port. Ports marked as static priority need not be contiguous, but they must be the lowest numbered PIDs mapped to this EID and must have QOS_MASK = 0xFF.

BP_PORTs are assumed to belong to an interface as follows:

BP_Port Range	Interface
36 ≤ BP_PORT ≤ 39	loopback interface
32 ≤ BP_PORT ≤ 35	DPI
0 ≤ BP_PORT ≤ 15	GMX0
0 ≤ BP_PORT ≤ 15	RXAUI interface 0

Cavium Confidential For Bowen Yang ALU 02/27/2014

The reset configurations for the fields are the following:

PID	EID (ext. port)	BP_PORT	QOS_MASK	STATIC_P
0	0 (0)	0	0xFF	0
1	1 (1)	1	0xFF	0
2	2 (2)	2	0xFF	0
3	3 (3)	3	0xFF	0
4	0 (0)	4	0xFF	0
5	1 (1)	5	0xFF	0
6	2 (2)	6	0xFF	0
7	3 (3)	7	0xFF	0
8	0 (0)	8	0xFF	0
9	1 (1)	8	0xFF	0
10	2 (2)	10	0xFF	0
11	3 (3)	11	0xFF	0
12	0 (0)	12	0xFF	0
13	1 (1)	13	0xFF	0
14	2 (2)	14	0xFF	0
15	3 (3)	15	0xFF	0
16	4 (16)	16	0xFF	0
17	5 (17)	17	0xFF	0
18	6 (18)	18	0xFF	0
19	7 (19)	19	0xFF	0
20	4 (16)	—	0xFF	0
21	5 (17)	—	0xFF	0
22	6 (18)	—	0xFF	0
23	7 (19)	—	0xFF	0
24	4 (16)	—	0xFF	0
25	5 (17)	—	0xFF	0
26	6 (18)	—	0xFF	0
27	7 (19)	—	0xFF	0
28	4 (16)	—	0xFF	0
29	5 (17)	—	0xFF	0
30	6 (18)	—	0xFF	0
31	7 (19)	—	0xFF	0
32	8 (32)	32	0xFF	0
33	8 (33)	33	0xFF	0
34	8 (34)	34	0xFF	0
35	8 (35)	35	0xFF	0
36	9 (36)	36	0xFF	0
37	9 (37)	37	0xFF	0
38	9 (38)	38	0xFF	0
39	9 (39)	39	0xFF	0
40	10 (24)	—	0xFF	0

Cavium Confidential For Bowen Yang ALU 02/27/2014

This register is a memory of 41 entries, and thus, PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. Read operations to any entries that have not been previously written are illegal and result in unpredictable CSR read data. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:62>	—	RAZ	—	—	Reserved
<61>	STATIC_P	R/W/H	—	0	Set if this PID has static priority.
<60:53>	QOS_MASK	R/W/H	—	0x0	Mask to control priority across eight QOS rounds.
<52:16>	—	RAZ	—	—	Reserved
<15:10>	BP_PORT	R/W/H	—	0x0	PID listens to BP_PORT for per-packet backpressure. Legal BP_PORT values are 0–15 (0xF), 32–39 (0x20–0x27), 63 (0x3F). **NOTE:** BP_PORT = 63 means that the PKO-internal port is not backpressured.
<9:6>	EID	R/W/H	—	0x0	Engine ID to which this port is mapped. Legal EID values are 0x0–0x9, 0xF. **NOTE:** If EID = 0x8 or 0x9, then PID[1:0] is used to direct the packet to the correct port on that interface. **NOTE:** EID = 0xF can be used for unused PKO-internal ports.
<5:0>	PID	R/W/H	—	0x0	Port ID[5:0]

PKO Memory Port QOS Register
PKO_MEM_PORT_QOS

This register sets the QOS mask, per port. These QOS_MASK bits are logically and physically the same as QOS_MASK bits in PKO_MEM_PORT_PTRS. This address allows the QOS_MASK bits to be written during PKO operation without affecting any other port state. The engine to which port PID is mapped is engine EID. Note that the port-to-engine mapping must be the same as was previously programmed via PKO_MEM_PORT_PTRS.

This register is a memory of 41 entries, and thus, PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. A read of any entry that has not been previously written is illegal and will result in unpredictable CSR read data. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:61>	—	RAZ	—	—	Reserved
<60:53>	QOS_MASK	R/W/H	—	0x0	Mask to control priority across eight QOS rounds.
<52:10>	—	RAZ	—	0x0	Reserved
<9:6>	EID	R/W/H	—	0x0	Engine ID to which this port is mapped. Legal EID values are 0–10.
<5:0>	PID	R/W/H	—	0x0	Port ID[5:0]

Cavium Confidential For Bowen Yang 02/27/2014

PKO Memory Port Rate0 Register
PKO_MEM_PORT_RATE0

This register is a memory of 41 entries, and thus, PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. A read of any entry that has not been previously written is illegal and will result in unpredictable CSR read data. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:51>	—	RAZ	—	—	Reserved
<50:32>	RATE_WORD	R/W/H	—	0x0	Rate limiting adder per eight bytes
<31:8>	RATE_PKT	R/W/H	—	0x0	Rate limiting adder per packet
<7:6>	—	RAZ	—	—	Reserved
<5:0>	PID	R/W/H	—	0x0	Port ID[5:0]

PKO Memory Port Rate1 Register
PKO_MEM_PORT_RATE1

This register is a memory of 41 entries, and thus, PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. A read of any entry that has not been previously written is illegal and will result in unpredictable CSR read data.

Writing PKO_MEM_PORT_RATE1[PID,RATE_LIM] has the side effect of setting the corresponding accumulator to zero. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:8>	RATE_LIM	R/W/H	—	0x0	Rate limiting accumulator limit
<7:6>	—	RAZ	—	—	Reserved
<5:0>	PID	R/W/H	—	0x0	Port ID[5:0]

PKO Memory Count0 Register
PKO_MEM_COUNT0

This register provides the total number of packets seen by PKO per port. Writing to this address clears the entry whose index is specified as COUNT[5:0].

This register is a memory of 41 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. A read of any entry that has not been previously cleared is illegal and results in unpredictable CSR read data. If the counter reaches the maximum value, it wraps to 0x0 and starts over. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	COUNT	R/W1C/H	—	0x0	Total number of packets seen by PKO

Cavium Confidential For Bowen Yang ALU 04/27/2014

PKO Memory Count1 Register
PKO_MEM_COUNT1

This register provides the total number of bytes seen by PKO per port. Writing to this address clears the entry whose index is specified as COUNT[5:0].

This register is a memory of 41 entries and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. A read of any entry that has not been previously cleared is illegal and results in unpredictable CSR read data. If the counter reaches the maximum value, it wraps to 0x0 and starts over. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	COUNT	R/W1C/H	—	0x0	Total number of bytes seen by PKO

PKO Memory Debug0 Register
PKO_MEM_DEBUG0

This register provides internal per-port state intended for debug use only. It is a memory of 11 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:36>	FAU	RO/H	—	0x0	Internal state.
<35:22>	CMD	RO/H	—	0x0	Internal state.
<21:16>	SEGS	RO/H	—	0x0	Internal state.
<15:0>	SIZE	RO/H	—	0x0	Internal state.

PKO Memory Debug1 Register
PKO_MEM_DEBUG1

This register provides internal per-port state intended for debug use only. It is a memory of 11 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	I	RO/H	—	0	Internal state.
<62:59>	BACK	RO/H	—	0x0	Internal state.
<58:56>	POOL	RO/H	—	0x0	Internal state.
<55:40>	SIZE	RO/H	—	0x0	Internal state.
<39:0>	PTR	RO/H	—	0x0	Internal state.

Cavium Confidential For Bowen Yang ALU 02/26/2014

PKO Memory Debug2 Register
PKO_MEM_DEBUG2

This register provides internal per-port state intended for debug use only. It is a memory of 11 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	I	RO/H	—	0	Internal state.
<62:59>	BACK	RO/H	—	0x0	Internal state.
<58:56>	POOL	RO/H	—	0x0	Internal state.
<55:40>	SIZE	RO/H	—	0x0	Internal state.
<39:0>	PTR	RO/H	—	0x0	Internal state.

PKO Memory Debug3 Register
PKO_MEM_DEBUG3

This register provides internal per-port state intended for debug use only. It is a memory of 11 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	DATA	RO/H	—	0x0	Internal state.

PKO Memory Debug4 Register
PKO_MEM_DEBUG4

This register provides internal per-port state intended for debug use only. It is a memory of 11 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:56>	CURR_SIZ	RO/H	—	0x0	Internal state.
<55:40>	CURR_OFF	RO/H	—	0x0	Internal state.
<39:34>	CMND_SEGS	RO/H	—	0x0	Internal state.
<33:18>	CMND_SIZ	RO/H	—	0x0	Internal state.
<17:12>	CMND_OFF	RO/H	—	0x0	Internal state.
<11:10>	UID	RO/H	—	0x0	Internal state.
<9>	DREAD_SOP	RO/H	—	0	Internal state.
<8>	INIT_DWRITE	RO/H	—	0	Internal state.
<7>	CHK_ONCE	RO/H	—	0	Internal state.
<6>	CHK_MODE	RO/H	—	0	Internal state.
<5>	WAIT	RO/H	—	0	Internal state.
<4:3>	MINOR	RO/H	—	0x0	Internal state.
<2:0>	MAJOR	RO/H	—	0x0	Internal state.

Cavium Confidential For Bowen Yang ADI 02/27/2014

PKO Memory Debug5 Register
PKO_MEM_DEBUG5

This register provides internal per-port state intended for debug use only. It is a memory of 11 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:56>	—	RAZ	—	—	Reserved
<55>	PTP	RO/H	—	0	Internal state.
<54>	MAJOR_3	RO/H	—	0	Internal state.
<53:48>	NXT_INFLT	RO/H	—	0x0	Internal state.
<47:8>	CURR_PTR	RO/H	—	0x0	Internal state.
<7:0>	CURR_SIZ	RO/H	—	0x0	Internal state.

PKO Memory Debug6 Register
PKO_MEM_DEBUG6

This register provides internal per-port state intended for debug use only. It is a memory of 40 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:37>	—	RO/H	—	0x0	Internal state.
<36:33>	QID_OFFRES	RO/H	—	0x0	Internal state.
<32:29>	QID_OFFTHS	RO/H	—	0x0	Internal state.
<28>	PREEMPTER	RO/H	—	0	Internal state.
<27>	PREEMPTEE	RO/H	—	0	Internal state.
<26>	PREEMPTED	RO/H	—	0	Internal state.
<25>	ACTIVE	RO/H	—	0	Internal state.
<24>	STATIC	RO/H	—	0	Internal state.
<23:21>	QOS	RO/H	—	0x0	Internal state.
<20:16>	QCB_RIDX	RO/H	—	0x0	Internal state.
<15:12>	QID_OFFMAX	RO/H	—	0x0	Internal state.
<11:8>	QID_OFF	RO/H	—	0x0	Internal state.
<7:0>	QID_BASE	RO/H	—	0x0	Internal state.

Cavium Confidential For Boyen Yang ALU 02/27/2014

PKO Memory Debug7 Register
PKO_MEM_DEBUG7

This register provides internal per-queue state intended for debug use only. It is a memory of 256 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:59>	QOS	RO/H	—	0x0	Internal state.
<58>	TAIL	RO/H	—	0	Internal state.
<57:45>	BUF_SIZ	RO/H	—	0x0	Internal state.
<44:12>	BUF_PTR	RO/H	—	0x0	Internal state.
<11:6>	QCB_WIDX	RO/H	—	0x0	Internal state.
<5:0>	QCB_RIDX	RO/H	—	0x0	Internal state.

PKO Memory Debug8 Register
PKO_MEM_DEBUG8

This register provides internal per-port state intended for debug use only. It is a memory of 256 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:42>	—	RAZ	—	—	Reserved
<41:34>	QID_QQOS	RO/H	—	—	Internal state.
<33>	—	RO/H	—	—	Reserved
<32:29>	QID_IDX	RO/H	—	—	Internal state.
<28>	PREEMPTER	RO/H	—	0	Internal state.
<27:8>	DOORBELL	RO/H	—	0x0	Internal state.
<7>	—	RAZ	—	—	Reserved
<6>	PREEMPTEE	RO/H	—	0	Internal state.
<5>	STATIC_P	RO/H	—	0	Internal state.
<4>	S_TAIL	RO/H	—	0	Internal state.
<3>	STATIC_Q	RO/H	—	0	Internal state.
<2:0>	QOS	RO/H	—	0x0	Internal state.

Cavium Confidential For Boyen Yang ALU 02/27/2014

PKO Memory Debug9 Register
PKO_MEM_DEBUG9

This register provides internal per-port state intended for debug use only. It is a memory of 11 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:49>	—	RAZ	—	—	Reserved
<48:32>	PTRS0	RO/H	—	0x0	Internal state.
<31:17>	—	RAZ	—	—	Reserved
<16:0>	PTRS3	RO/H	—	0x0	Internal state.

PKO Memory Debug10 Register
PKO_MEM_DEBUG10

This register provides internal per-port state intended for debug use only. It is a memory of 11 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:49>	—	RAZ	—	—	Reserved
<48:32>	PTRS1	RO/H	—	0x0	Internal state.
<31:17>	—	RAZ	—	—	Reserved
<16:0>	PTRS2	RO/H	—	0x0	Internal state.

PKO Memory Debug11 Register
PKO_MEM_DEBUG11

This register provides internal per-port state intended for debug use only. It is a memory of 11 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:23>	—	RAZ	—	—	Reserved
<22>	MAJ	RO/H	—	0	Internal state.
<21:19>	UID	RO/H	—	0x0	Internal state.
<18>	SOP	RO/H	—	0	Internal state.
<17>	LEN	RO/H	—	0	Internal state.
<16>	CHK	RO/H	—	0	Internal state.
<15:3>	CNT	RO/H	—	0x0	Internal state.
<2:0>	MOD	RO/H	—	0x0	Internal state.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PKO Memory Debug12 Register
PKO_MEM_DEBUG12

This register provides internal per-port state intended for debug use only. It is a memory of 44 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:36>	FAU	RO/H	—	0x0	Internal state.
<35:22>	CMD	RO/H	—	0x0	Internal state.
<21:16>	SEGS	RO/H	—	0x0	Internal state.
<15:0>	SIZE	RO/H	—	0x0	Internal state.

PKO Memory Debug13 Register
PKO_MEM_DEBUG13

This register provides internal per-port state intended for debug use only. It is a memory of 44 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	I	RO/H	—	0	Internal state.
<62:59>	BACK	RO/H	—	0x0	Internal state.
<58:56>	POOL	RO/H	—	0x0	Internal state.
<55:40>	SIZE	RO/H	—	0x0	Internal state.
<39:0>	PTR	RO/H	—	0x0	Internal state.

PKO Memory Debug14 Register
PKO_MEM_DEBUG14

This register provides internal per-port state intended for debug use only. It is a memory of 132 entries, and consequently PKO_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 10–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	DATA	RO/H	—	0	Internal state.

Cavium Confidential For Bowen Yang ALU 02-12-2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 11

Hyper Finite Automata (HFA)

This chapter contains the following subjects:

- Overview
- Introduction
- Hardware Features
- Theory of Operation
- External Data Structures
- HFA Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

This chapter describes the architecture and design implementation of the regular expression coprocessor (hyper finite automata, HFA). The HFA offers hardware-acceleration for deep-packet-inspection (DPI) applications such as:

- intrusion detection/prevention (IDP)
- packet classification
- server load balancing
- web-switching
- storage array networks (SANs)
- firewall load balancing
- virus scanning

This hardware-acceleration unit offloads the heavy burden of performing memory-intensive pattern-matching from the cnMIPS III cores.

11.1 Introduction

HFA is Cavium's third-generation pattern-search technology. It offers pattern-database size, larger graph storage space, standard DDR3 memory for patterns and faster per-thread processing than its DFA predecessor, allowing it to reach speeds up to 500Mb/s line rate character-search performance.

HFA incorporates advantages of both deterministic finite automata (DFA) and nondeterministic finite automata (NFA).

- It uses DFA concepts to make search deterministic and maintain smaller context state (for match across packets).
- It uses NFA concepts to avoid state explosion due to wild characters, ranges, etc.

As a result, HFA can support the Perl Compatible Regular Expression rule set with a very small memory footprint and without state explosion.

Any cnMIPS III core can issue an HFA instruction to the HFA instruction queue in system memory. This instruction is then assigned to one of the 16 HFA thread engines (HTEs) whose function is to walk the graph to completion and write the search results to system memory. The initiating cnMIPS III core can be alerted to the instruction completion either by polling or by a work-queue interrupt.

Cavium Confidential For Bowen Yang 02/10/2014

The software compiler takes input patterns and generates a complete graph (stored in external system memory) and optionally a graph image gist (GIG), which is a condensed image of the complete graph that has a complete view of all patterns.

- The GIG is stored in 40+ KB of internal graph cache (a static cache that is loaded with each HFA graph's gist for faster processing).
- The complete graph is stored in up to 4GB of system memory. Multiple graphs (up to 4GB per graph) can be stored in system memory, up to the architectural limit of 32GB over one port of DDR3 memory.

NOTE: System memory is shared by the HFA, which is noncoherent, and the level-two cache (L2C), which is coherent. Both HFA and L2C use the local memory controller (LMC) to access system memory. HFA access to memory are not coherent.

11.2 Hardware Features

- Sixteen concurrent HTEs
- On-chip HFA graph cache = 40+ KB (RAM1, RAM2, RAM3)

NOTE: The on-chip HFA cache is available only to HFA and is separate from the L2 cache.

- HFA memory support:
  - shared in system memory with L2C coherent memory
  - 400–533 MHz clock (800–1066 MHz dual data rate) operation
  - a 36-bit (32 bits of data and 4 bits of ECC) LMC memory port
  - architectural support for up to 128GB of graph space
  - ECC protection (single-error correct/double-error detect)

NOTE: The HFA memory is in system DDR3 memory. Refer to Chapter 4.

- Aggregate performance up to 500Mb/s. Hardware supports up to 64 cacheable graphs and an infinite number of memory-only graphs, bound only by attached DDR3 memory size.

  With the help of a software linker, it is possible to support up to 512 cacheable graphs (see the SDK for more details).
- The number of strings/regex patterns supported is limited only by the size of pattern memory space
- Packet data and graph gather mode support. Packet data and graph data need not be stored contiguously in system memory.

Cavium Confidential For Bowen Yang ALU 01/27/2014

11.3 Theory of Operation

Any cnMIPS III core can write an HFA instruction to the HFA instruction queue in system memory. This instruction is then assigned to one of the 16 HFA thread engines (HTEs), which walks the graph to completion and writes the search results to system memory. The initiating core can be alerted to the instruction completion either by polling or by a work-queue entry.

The cnMIPS III cores have access to the 16 HTEs, each capable of walking graphs that are stored across the 40+ KB of internal graph cache and the HFA- memory (in system memory). The HFA memory and L2C memory both share up to 32GB of attached DDR3 system memory.

The internal graph cache is a static cache, loaded with each HFA graph's GIG for faster processing. Large graphs can then extend into the external HFA memory.

There are two types of HFA graphs:

- Cacheable graphs, which can reside in graph cache and can spill out to HFA memory. HFA hardware supports up to 64 cacheable graphs limited by on-chip graph cache.

NOTE: Using the software linker provided with the SDK, HFA can load 512 cacheable graphs.

- Memory-only graphs, which reside strictly in HFA memory. HFA hardware can support an unlimited number of memory-only graphs. The number, however, is limited by the size of HFA memory. Some minimum amount of L2C memory is required for proper operation, and the remainder can be partitioned for HFA memory.

The HTEs are treated as a pool of coprocessors available to all of the cores. To initiate an HFA walk, a core writes into system memory a fixed-size/format HFA macroinstruction that contains information for the hardware engine to perform the HFA graph walk.

A doorbell register is then written, which triggers the HFA hardware engine to start its graph walk from the node specified in the HFA macroinstruction.

This HFA graph walk continues until one of the following occurs:

- all the packet-input data is exhausted
- a terminal node is reached
- the specified match result buffer becomes full
- an error is detected (e.g. a parity error).

A *match* is noted whenever a marked node (i.e. a node marked as a match) is visited. During the HFA graph walk, for each match detected, intermediate results are written into a specified result buffer. The content of the intermediate results contains the character offset, and handle to the marked state of the HFA graph. A marked transition is a transition to a marked node.

If the input data is exhausted before a terminal node is reached, the HFA context is written to the result buffer and control is returned to the requesting packet processor through either an interrupt (work-queue entry) or via software polling. The last entry in the result buffer contains the HFA context, which contains the input data length and also the current state of the HFA graph. This facilitates packets that span multiple input-data segments, so that subsequent search requests may pick up where

Cavium Confidential For Bowen Yang ALU 02/27/2014

the previous search left off. The HTE also supports a packet gather mode, which allows a user to specify a gather-pointer list, which allows multisegmented packets to be autotraversed by the HFA hardware itself.

11.3.1 HFA Memory Debug Access

HFA supports IOI-direct read-only CSR accesses to HFA memory space for debug purposes. All IOI-direct read accesses are limited to 64 bits of data. To read the HFA memory, a device must issue an IOI request from HFA memory space, where addr[34:32] = 0x5, and addr[31:3] is the aligned address to read. See also the DFA_DIFCTL[MSEGBASE] field which provides the additional upper-address bits to access the entire 32GB of system memory. This fields acts as a segmentation base address.

11.3.2 HFA Graph Types

Software is responsible for building the HFA graphs that are written into HFA graph cache and external HFA memory. The HTEs efficiently walk the graph to completion.

NOTE: The HFA hardware can directly access system memory through the LMC without having to probe the L2 cache. Therefore, most applications will partition system memory into L2C coherent space and HFA graph space, and the two should not intersect. However, to load the HFA graphs into HFA memory, the cores will use the SW_MLOAD SDK routine to ensure the graph data is resident in system memory for access by the HFA.

Cavium supplies an exhaustive SDK that includes a regex compiler, a linker, and a functional HTE software emulation. Refer to SDK documentation for more details.

11.4 External Data Structures

11.4.1 HFA Instruction Queue

A HFA instruction queue (IQ) must be setup in system memory prior to initiating a hardware-assist pattern search. The IQ is maintained by both hardware and software.

The HFA hardware maintains the head pointer (read) for the queue (current instruction), and the core software maintains the tail pointer (write) where new HFA instructions are inserted.

At boot time, software must initialize the IQ head pointer maintained by hardware (DFA_DIFRDPTR[RDPTR]) once before any HFA instructions are issued. This seeds the initial head pointer so that HFA hardware knows where to fetch the next available instruction to be processed in the queue.

The IQ memory is maintained as a linked list of HFA instruction chunks. Each instruction chunk contains a programmable number of fixed length HFA instructions. Software allocates these instruction chunks and the HFA hardware frees them. The instruction-chunk size is user defined by DFA_DIFCTL[SIZE].

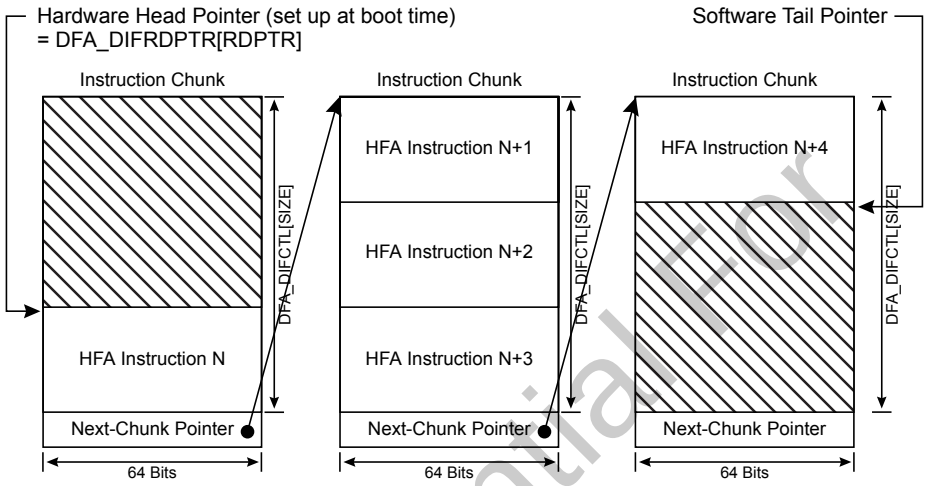
Immediately following the instruction chunk is a 64-bit word that contains the next-chunk pointer link. The hardware may read the next-chunk pointer as soon as the doorbell count indicates that the last HFA instruction in an instruction chunk

Cavium Confidential For Bowen Yang ALU 02/7/2014

contains a valid HFA instruction. This requires that software allocate the next chunk, and then write the next-chunk pointer into the prior chunk to point to it as soon as it writes the last HFA instruction into the chunk.

The HFA hardware reads words from the IQ (starting with the head pointer), and when the last instruction of an instruction chunk is used, the HFA hardware traverses to the next instruction chunk using the next chunk pointer link. In Figure 11–1, the instruction-chunk size is three, which is the minimum legal value.

Figure 11–1 HFA IQ Format



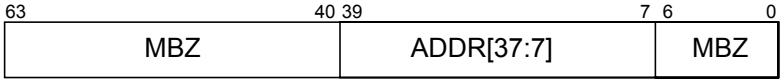
In this way, as the last instruction of a chunk is retired, hardware can automatically free the retired memory chunk back to the FPA hardware-managed pool selected by DFA_DIFCTL[POOL]. For L2 cache performance, the discarded memory chunk can also be tagged with a don't-write-back (DWB) count (DFA_DIFCTL[DWBCNT]), which prevents the cacheline(s) from being written back to system memory, to avoid memory bus congestion (i.e: cachelines are tagged with DIRTY = 0 to prevent writeback).

NOTE: Software must ensure that a HFA instruction is ALWAYs contained within a defined instruction-chunk boundary (i.e. never allowed to straddle into the next chunk).

11.4.1.1 Next-Chunk Pointer

The next 64-bit word after the last instruction within an instruction chunk is the next-chunk pointer (shown in Figure 11–2), which provides the forward link to access the next instruction chunk in system memory. The address is a byte address, and must be naturally aligned to a cache-line boundary (i.e. 128 bytes: the software must ensure that the least-significant seven bits of the address are always 0s).

Figure 11–2 Next-Chunk Pointer Format



Cavium Confidential For Bowen Yang 02/27/2014

11.4.1.2 HFA Instruction Issue

To insert a HFA instruction into the queue, software first writes the instruction to the tail pointer maintained by software, followed by a write operation to the HFA doorbell-count register (DFA_DBELL[DBELL]), with the number of instructions to be added to the IQ. Write operations to the doorbell register are cumulative, and reflect the total number of pending instructions. As the HFA hardware retires instructions, the doorbell register is autodecremented. The doorbell register can accumulate up to one million pended instruction requests (overflowing the count is catastrophic and causes a DBLOVF interrupt). The hardware continues to execute instructions until all pending requests have been serviced (i.e. the accumulated doorbell register = 0).

Software can issue a doorbell write operation for each individual instruction, or it can accumulate multiple instructions into a single doorbell write operation. The only requirements are the following

- the number of valid instructions in the queue must be at least as large as the doorbell count
- the next-chunk pointers that are interspersed among the instructions must also be set up properly.

Software must guarantee that L2C/DRAM contains the number of instructions indicated in the doorbell write operation before the doorbell write operation reaches the HFA. The distance between the instruction head and tail pointers is both the size of the IQ and the outstanding doorbell count. The size of the IQ is limited only by the available memory and the 20-bit outstanding-doorbell count. Software must guarantee that the maximum IQ never exceeds more than 2^{20} instructions.

Hardware makes no attempt to ensure order of execution of HFA instructions. It is up to software to ensure the proper ordering using the following rules:

- Before issuing a Graph Free (GFREE) instruction, software must do the following checks:
  - ensure that there are no outstanding HFA instructions to the same graph (IWORD2[VGID]/IWORD0[PGID]).
  - wait for the completion status (RWORD0[REA]) for all outstanding HFA instructions to the same graph.
  - wait for GFREE RWORD0 completion status before it issues newer Graph CacheLoad (CLOAD) instructions.
- Before issuing a CLOAD instruction, software must do the following checks:
  - ensure that there are no outstanding HFA instructions to the same graph (IWORD2[VGID]/IWORD0[PGID]).
  - wait for the completion status (RWORD0[REA]) for all outstanding HFA instructions to the same graph.
  - wait for CLOAD RWORD0 completion status before it issues Graph Walk (GWALK) instructions to use the new graphs loaded.
- Before issuing a GWALK instruction, software must do the following checks:
  - ensure that there are no outstanding Graph MemLoad (SW_MLOAD), CLOAD, or GFREE instructions to the SAME graph (IWORD2[VGID]/ IWORD0[PGID]). However, other GWALK instructions are allowed to be issued.

Cavium Confidential For Bowen Yang ALU 09/27/2014

NOTE: The loading of graphs into HFA memory is performed by the SW_MLOAD routine that is provided in the Cavium SDK kit. The SW_MLOAD routine ensures that graph data written by the core to L2C coherent memory is properly flushed out of the L2 cache and LMC write buffers into system memory, such that HFA memory accesses source data directly from system memory.

11.4.2 HFA Instruction Formats

There are three 32-byte fixed-size HFA instructions that contain four 64-bit instruction words.

- Graph CacheLoad (CLOAD)
- Graph Walk (GWALK)
- Graph Free (GFREE)

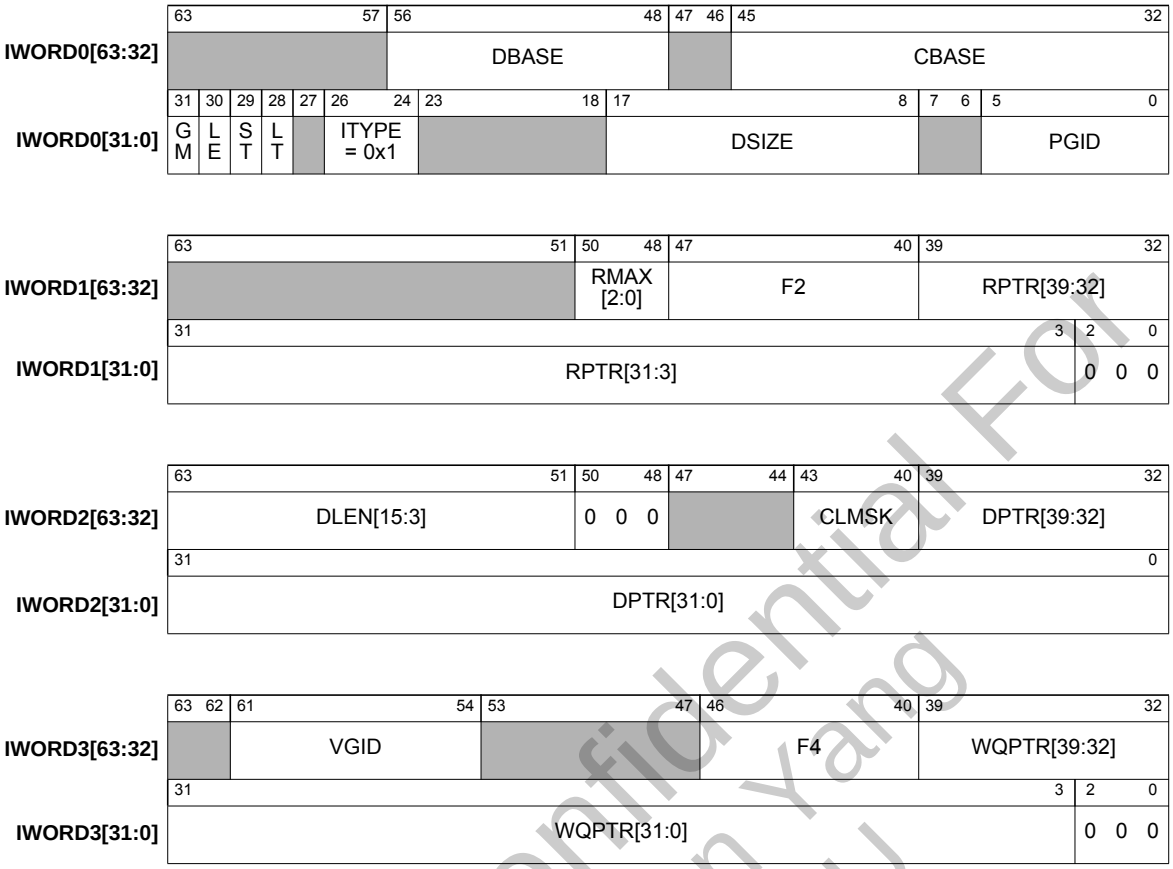
Based on the HFA instruction type (IWORD0[ITYPE]), the 32-byte instruction takes on different formats (as shown in Figures 11–3 through 11–5).

Cavium Confidential For Bowen Yang ALU 02/27/2014

11.4.2.1 Graph CacheLoad Instruction

The CLOAD instruction is used to transfer the GIG information from system memory into the HFA graph cache.

Figure 11–3 HFA Instruction Format: Graph CacheLoad (ITYPE = 0x1)



NOTE: Shaded bits are reserved and must be 0.

Software Restrictions:

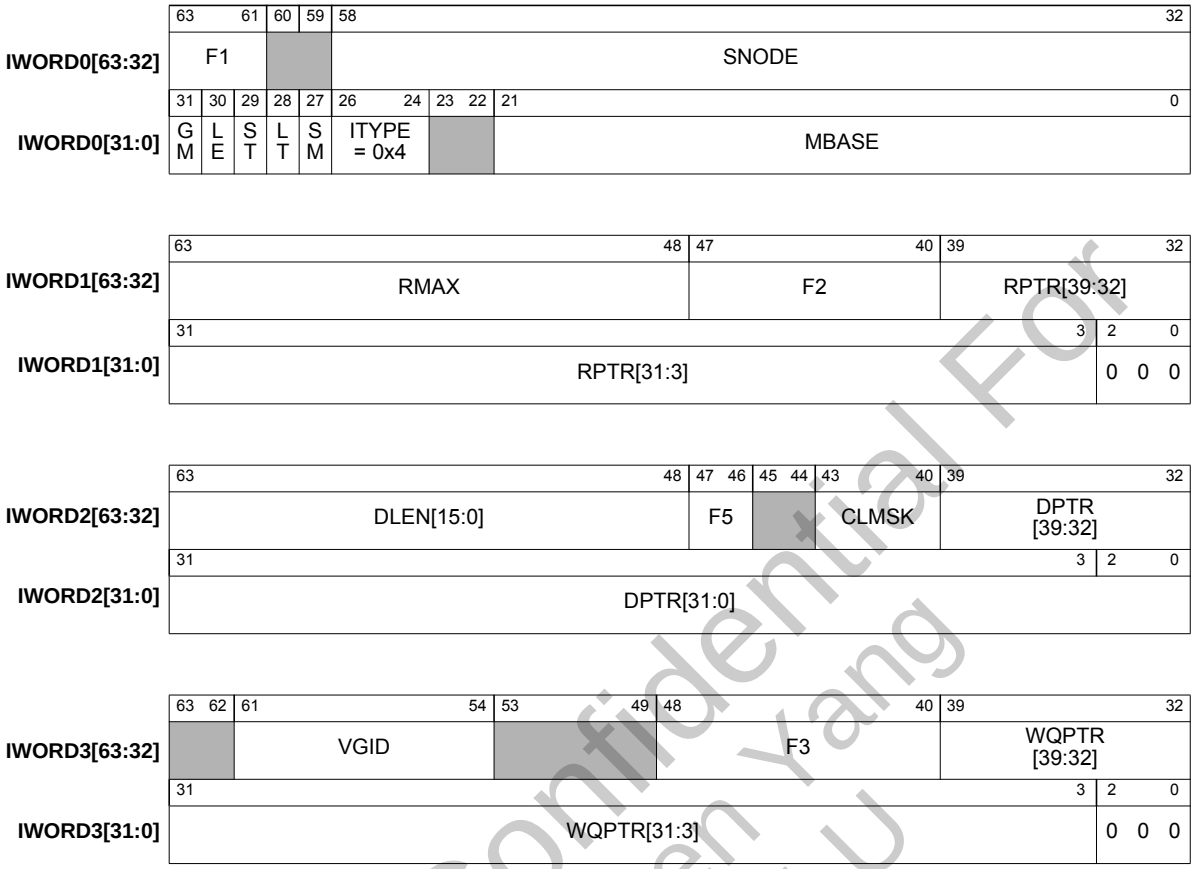
1. IWORD2[CLMSK] must be 0x1.
2. IWORD2[DLEN<2:0>] must be 0x0 (must be a doubleword multiple)
3. IWORD2[DPTR<2:0>] must be 0x0 (must be doubleword-aligned in system memory)

Cavium Confidential For Bowen Yang ALU 6/27/2014

11.4.2.2 Graph Walk Instruction

The GWALK instruction is used for walking the graph. It contains information about where packet-data is read from system memory, and where result buffer information is written to system memory along with graph ID and the rest of the information.

Figure 11–4 HFA Instruction Format: Graph Walk (ITYPE = 0x4)



NOTE: Shaded bits are reserved and must be 0.

Software Restrictions:

1. IWORD2[CLMSK] must be 0x1.

Cavium Confidential For Bowen Yang ALU 02/27/2014

11.4.2.3 Graph Free Instruction

The GFREE instruction is used to free a virtual graph from HFA hardware.

Figure 11–5 HFA Instruction Format: Graph Free (ITYPE = 0x3)

NOTE: Shaded bits are reserved and must be 0.

Software Restrictions:

1. IWORD2[CLMSK] must be 0x1.

11.4.3 Detailed IWORD Field Definitions

The following IWORD definitions are made based on the GWALK instruction format, where each subfield for other instructions is superimposed and described in more detail in the text that follows.

Each graph has unique characteristics that are encoded into five opaque fields, specified as F1, F2, F3, F4, and F5. Of the five fields, F1, F2, F3, and F5 are context specific.

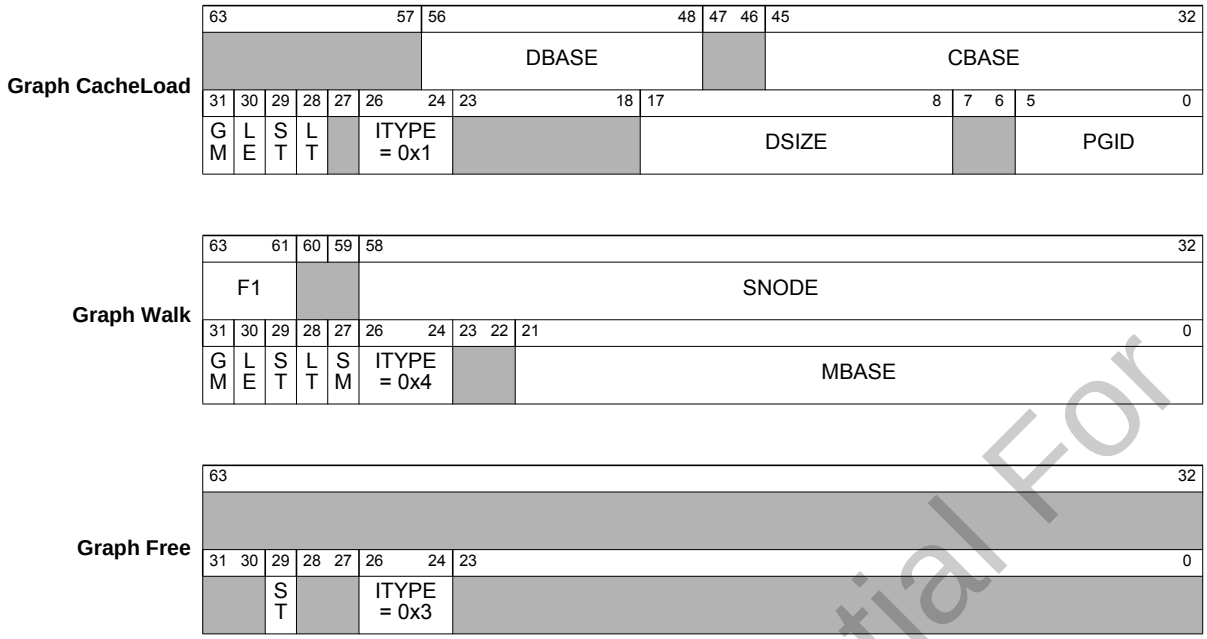
Initially, all five fields are extracted from the compiled graph information file (refer to the SDK documentation for details). For subsequent requests to same graph and for packets belonging to same flow (flow definition is application-dependent and in this context it means the virtual entity within which the application cares about matches across the packet boundary), fields F1, F2, and F3 are extracted from RWORD1+ [47:28] and field F5 is extracted from RWORD0+ [19:18] (see Section 11.4.4) of the previous request and inserted at respective locations in the IWORD.

Cavium Confidential For Bowen Yang ALU 02/27/2014

11.4.3.1 IWORD0

IWORD0 for all the graph instructions is shown in Figure 11–6.

Figure 11–6 IWORD0



IWORD0[63:61] [F1] Present in GWALK (ITYPE = 0x4) instruction.
Refer to description in Section 11.4.3 and RWORD1[30:28] [F1] definition for details.

IWORD0[58:32] [SNODE] Present in GWALK (ITYPE = 0x4) instruction.
Starting-node pointer (relative to graph).

IWORD0[56:48] [DBASE] Present in CLOAD (ITYPE = 0x1) instruction.
Represents the starting RAM2 address (single-entry aligned) for this graph. During the CLOAD instruction, this specifies where the RAM2 contents are loaded for a given cluster specified in the CLMSK[2:0] field. Also, during the CLOAD instruction, this field is stored in a hardware graph-usage table (GUT) for future GWALK instructions.

HFA hardware uses WORD2[DLEN] to determine the total number of 64-bit doublewords to read from system memory for the cache-load operation. The first 32 system-memory doublewords read are written into 64 RAM3 entries using IWORD0[PGID] as a RAM3 base address. The next [DSIZE/2] system-memory doublewords read are written into [DSIZE] RAM2 entries using IWORD0[DBASE] as a RAM2 base address. And the remaining [DLEN – (32 + DSIZE/2)] system-memory doublewords read are written into RAM1 using IWORD0[CBASE] as a RAM1 base address. For more details, refer to Section 11.4.6.1].

IWORD0[45:32] [CBASE] Present in CLOAD (ITYPE = 0x1) instruction.
Represents the starting RAM1 address (single-entry aligned) for this graph. During the CLOAD instruction, this specifies where the RAM1 contents are loaded for a given cluster specified in the CLMSK[2:0] field. Also, during the CLOAD instruction, this field is stored in a hardware graph-usage table (GUT), for future GWALK instructions.

Each HTE has access to a 4K × 81-bit graph cache which is shared among 16 HTEs and may contain multiple graphs.

GraphStartCAddr = (IWORD0[CBASE]);

Cavium Confidential For Bowen Yang ALU 12/27/2013

IWORD0[31] [GM] Present in all instructions except GFREE (ITYPE = 0x3).
Gather mode.

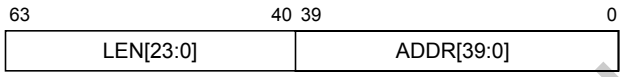
0 = the instructions do not use gather mode; they use direct mode to obtain the input packet-data bytes for the HFA graph traversal or graph bytes for CacheLoad.

1 = the instructions use gather mode to obtain their information indirectly:

- the GWALK instruction obtains input packet-data bytes for the graph walk operation
- the CLOAD instruction obtains the cache graph image to load from system memory into the specified cache RAMs

Each 64-bit HTE gather-list entry (shown in Figure 11–7) contains a **length** (in bytes) and an **address field** (an L2C/DRAM byte address). The HTE gather-list entries must always be naturally aligned on a 64-bit boundary (i.e. IWORD2[DPTR<2:0>] = 0x0).

Figure 11–7 HTE Gather-List-Entry Format



NOTE: **Length Software Restrictions**:

For **CLOAD instructions**, LEN[2:0] must be 0x0. Software must ensure that all gather-list LENs for CLOAD instructions are sized as quadword-multiples for proper operation. (DFA Cache granularity is quadword (64 bits)). Also, software must ensure that the maximum CLOAD size does not exceed the DFA cache size.

For **GWALK instructions**, LEN[23:16] must be 0x0. Software must ensure that gather-list LENs for GWALK instructions are limited to a maximum size of 64KB – 1.

IWORD0[30] [LE] Present in all instructions except GFREE (ITYPE = 0x3).
Little-endian mode.

0 = graph or packet data in system memory is in big-endian mode
1 = graph or packet data in system memory is in little-endian mode

NOTE: The graph data read by the HTEs from the graph cache or HFA DDR3 memory is always read/interpreted in little-endian format, as shown in Section 11.4.5, "HFA Graph Formats".

IWORD0[29] [ST] Present in all instructions that require system memory write operations.
Store-Full Hint (zero fill the unused bytes of the cacheline). This mode bit is used to force STF (store full hint) commands to L2C/DRAM, to indicate that unused bytes of the write operation are zero filled to L2C/DRAM memory for the cacheline. This is a performance enhancement feature, as the STF is used by the L2C to zero fill unused data in the cacheline (rather than to fetch DRAM memory).

If ST = 0, the HTE must write only individual words when it places result data into L2C/DRAM.

If ST = 1, the HTE can write entire cache blocks surrounding the words of result data, and it can write result words using the STF hint to L2C/DRAM indicating its intent to write the entire cacheline.

Core software must never set ST = 1 when any of the possible result words may reside in the same 128-byte cache block as other useful data structures.

Cavium Confidential For Bowen Yang ALI 02/21/2014

IWORD0[28] [LT] Present in all instructions except GFREE (ITYPE = 0x3) (i.e. all instructions that require system memory read operations).
Load-through mode. This mode bit is used to force LDT commands onto the I/O Bus, so that L2C does not allocate L2 data during the read operation.

IWORD0[27] [SM] Present in GWALK (ITYPE = 0x4) instruction.
Global small memory flag. HFA supports two types of memory nodes, small and large, and this bit determines the size of the memory nodes: 0 = large, 1 = small. Refer HFA SDK documentation for details.

IWORD0[26:24] [ITYPE] Present in all instructions.
Specifies the type of instruction:

0x0 = reserved
0x1 = CLOAD. Used to load internal graph caches with graph data (refer to Section 11.4.6.1, "Cacheable Graph Loading").
0x2 = reserved
0x3 = GFREE. Used to free a cacheable HFA graph from the graph cache.
0x4 = GWALK. Used to walk a HFA graph.
0x5–0x7 = reserved

NOTE: An HFA instruction with an ITYPE of *reserved* is terminated with an RWORD0[REAS] = GERR code (refer to "RWORD0[63:61] [REAS]" on page 609).

IWORD0[21:0] [MBASE] Present in GWALK (ITYPE = 0x4) instruction.
Specifies where in external HFA memory the extended graph exists. Multiple graphs are resident in external memory at any given time, and MBASE establishes the graph's base address in HFA memory.

NOTE: When software loads a cacheable graph, it is assumed that the SW_MLOAD SDK routine is completed prior to a CLOAD instruction. This assumption is made because, during the HFA instruction scheduling, the graph usage table (GUT) lookup determines if a cache graph is valid. If a GUT[VGID] (virtual graph ID) lookup cannot find a valid node cluster, then the result word reported indicates "NOGRAPH". Since the CLOAD completion validates the GUT node cluster, then all later HFA instructions targeting that VGID will source the proper graph data.

HFA Graph Memory Base = mem_addr[37:16] (64KB-aligned)
GraphStartMAddr = IWORD0[MBASE] << 16;

IWORD0[21:13] [DSIZE] Present in CLOAD (ITYPE = 0x1) instruction.
Represents the number of RAM2 entries to be loaded for this graph, starting at IWORD0[DBASE].

Cavium Confidential For Bowen Yang 07/25/14

IWORD0[5:0] [PGID] Present in CLOAD (ITYPE = 0x1) instruction.

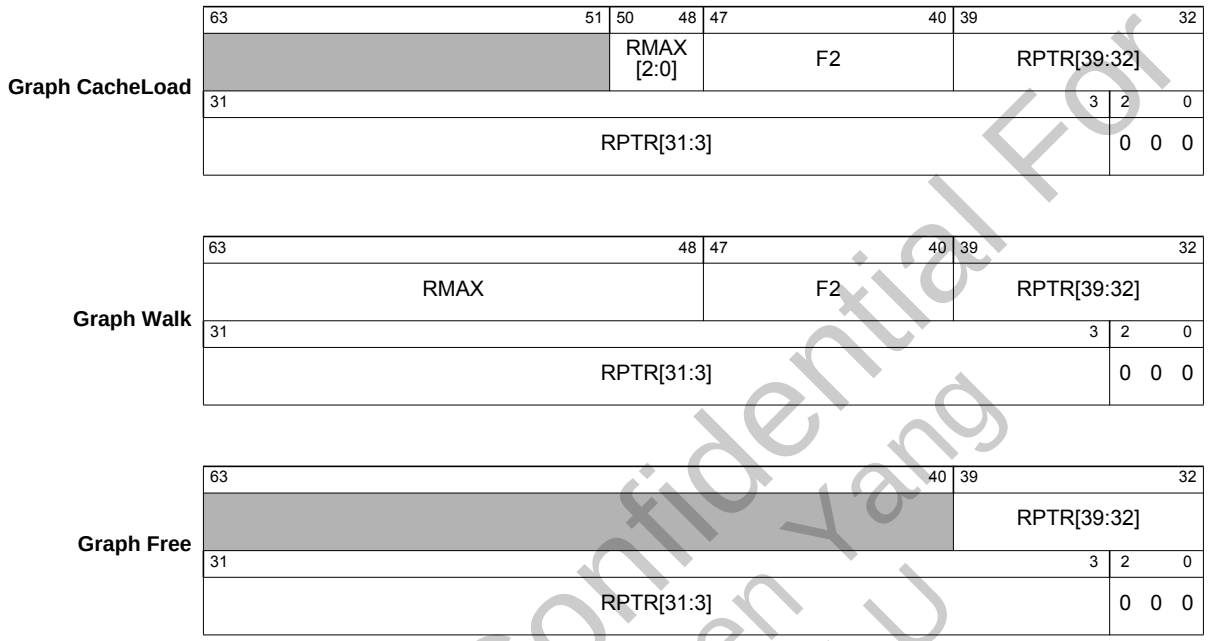
Physical graph ID number (1 of 64) that is to be loaded. The HFA hardware captures the PGID[5:0] into an internal GUT at an index specified by IWORD3[VGID]. During subsequent GWALK instructions, the new instruction's VGID is used to lookup in the GUT, to determine if the cacheable graph exists and where it exists within the graph cache.

The PGID is used by the HTEs to determine the RAM3 PBASE address. PBASE = PGID × 64.

11.4.3.2 IWORD1

IWORD1 for all the graph instructions is shown in Figure 11–8.

Figure 11–8 IWORD1



IWORD1[63:48] [RMAX] Present in all instructions.

Maximum result buffer words (8 bytes).

For GWALK instructions: the maximum number of 64-bit result words beyond RWORD[0] of the result. The maximum total number of result words is (RMAX + 1).

NOTE: **Software Note:** RMAX must be at least 0x1.

Cavium Confidential For Bowen Yang ALU 02/27/2014

For CLOAD instructions with IWORD0[GM] = 0: RMAX[2:0] is used in determining the number of system memory bytes to read that will be written to HFA memory space. When IWORD0[GM] = 1, RMAX is not used to determine the graph length.

GraphCLen(B) = {RMAX[2:0],DLEN[15:0]}.

NOTE: CLOAD instructions do not write RWORD1+ entries to the result buffer (i.e. RMAX = 0 is implied by the hardware), thus the RMAX field is reused to specify the size of system memory to load for the graph.

IWORD1[47:40] [F2] Present in GWALK (ITYPE = 0x4) and CLOAD (ITYPE = 0x1) instructions.

Refer to description in Section 11.4.3 and RWORD1[38:31] [F2] definition for details.

IWORD1[39:3] [RPTR<39:3>] Present in all instructions.

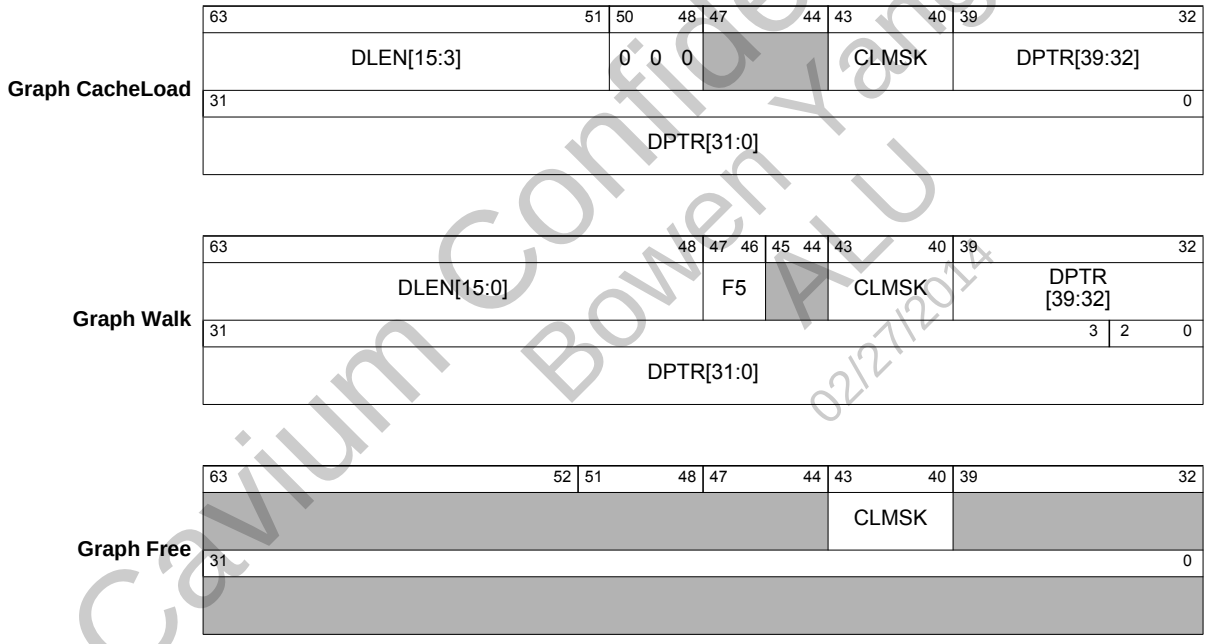
Result pointer. An L2C/DRAM pointer that indicates where the hardware puts the results. Since the HFA result data structure must be 64-bit aligned, RPTR[2:0] (i.e. IWORD1[2:0]) must be 0x0 (see Section 11.4.4).

An RPTR must be assigned for all HFA instruction types so that at least RWORD0 can be written.

11.4.3.3 IWORD2

IWORD2 for all the graph instructions is shown in Figure 11–9.

Figure 11–9 IWORD2



Cavium Confidential For Bowen Yang ALU 02/27/2014

IWORD2[63:48] [DLEN<15:0>] Present in all instructions except GFREE (ITYPE = 0x3).

Input data length.

For GWALK instructions: DLEN must be in the range 1 – (64K–1).

- If IWORD0[GM] = 0, DLEN is the number of input-packet data bytes at IWORD1[DPTR].
- If IWORD0[GM] = 1, DLEN is the number of HFA gather-list entries at IWORD1[DPTR].

For CLOAD instructions: DLEN<15:3> represents the total number of doublewords from system memory to read for the Cache load operation.

NOTE: **Software Restriction**: For CLOAD instructions, DLEN must be doubleword-sized (i.e. 64 bits, DLEN[2:0] = 0x0).

IWORD2[47:46] [F5] Present in MLOAD (ITYPE = 0x0) and GWALK (ITYPE = 0x4) instructions.

Refer to description in Section 11.4.3 and RWORD0[19:18] [F5] definition for details.

IWORD2[43:40] [CLMSK] Present in all instructions.

Cluster mask. Must be 0x1.

IWORD2[39:0] [DPTR<39:0>] Present in all instructions except GFREE (ITYPE = 0x3).

Data packet pointer.

For GWALK instructions:

- If IWORD0[GM] = 0, direct mode is used to fetch packet-data bytes, and DPTR is the L2C/DRAM location of the first packet data byte.
  - if IWORD0[LE] = 0, DPTR is a big-endian pointer.
  - if IWORD0[LE] = 1, DPTR is a little-endian pointer.
- If IWORD0[GM] = 1, gather mode is used to fetch packet-data bytes, and DPTR is the L2C/DRAM location of the HFA gather list. DPTR<2:0> must be 0x0 in this case, since the gather-list start must be aligned on a 64-bit boundary.

For CLOAD instructions: DPTR points to the graph source address in host memory. The definition of IWORD0[GM] remains the same and graph data be retrieved in either direct mode or gather mode.

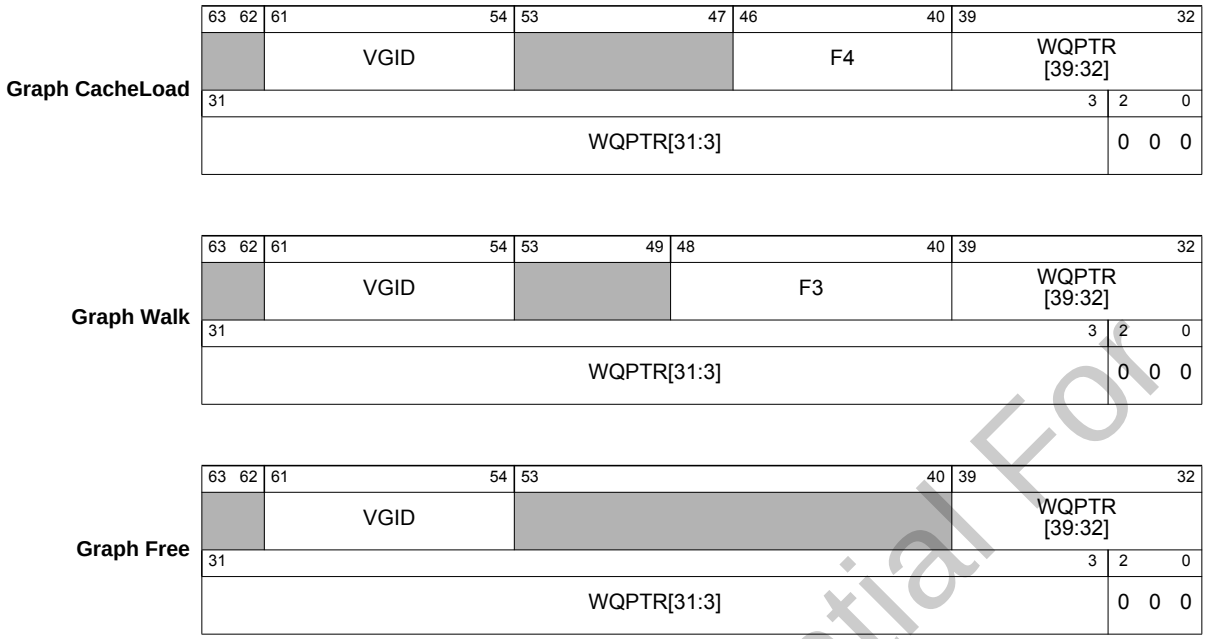
NOTE: The DPTR must be doubleword-bit-aligned (i.e. DPTR<2:0> = 0x0) (hardware forces these bits zero).

Cavium Confidential For Edward Yang ALU 02/27/2014

11.4.3.4 IWORD3

IWORD3 for all the graph instructions is shown in Figure 11–10.

Figure 11–10 IWORD3



IWORD3[61:54] [VGID] Present in all instructions.

Virtual graph ID number. Specifies the virtual graph ID number (of 256), which is uniquely assigned by software for each cacheable graph.

IWORD3[48:40] [F3] Present in the GWALK (ITYPE = 0x4) instruction.

Refer to description in Section 11.4.3 and RWORD1[47:39] [F3] definition for details.

IWORD3[46:40] [F4] Present in the CLOAD (ITYPE = 0x1) instruction.

Refer to description in Section 11.4.3.

IWORD3[39:0] [WQPTR] Present in all instructions.

Work-queue-entry pointer.

If WQPTR ≠ 0x0, the hardware submits the work after it completes the instruction and writes the result.

If WQPTR = 0x0, the hardware does not submit any work after it completes the instruction.

NOTE: IWORD3[WQPTR<2:0>] must be 0x0, since all work-queue pointers must be 64-bit-aligned.

Cavium Confidential For Bowen Yang ALU 02/27/2014

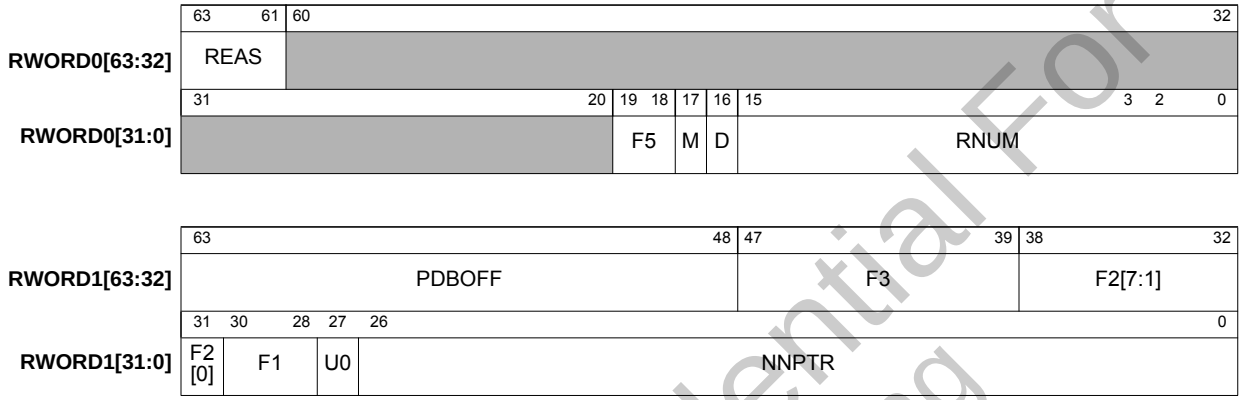
11.4.4 HFA Result Format

The HFA result has one or more 64-bit naturally aligned doublewords in L2C/DRAM. The hardware writes these doublewords to L2C/DRAM during and after it processes the HFA instruction. The structure is variable length to accommodate instructions that hit a variable number of marked nodes, but the result length can be limited by the IWORD1[RMAX] field of the instruction. Once an instruction has completed, software is notified of the completion using either a software polling scheme or by a work-queue interrupt. The signaled completion ensures that software can read/process the result buffer information in system memory.

All GWALK instructions return RWORD0 and at least one RWORD1 result. All CLOAD/GFREE instructions return only a single RWORD0.

Figure 11–11 shows the result format.

Figure 11–11 HFA Result Format



NOTE: Shaded bits are reserved and must be 0.

RWORD0 Definition:

RWORD0[63:61] [REAS] Completion reason code.

0x0 = DDONE. The HTE finished after processing all input-packet data bytes. DDONE is lower priority than either ERR or TERM codes.

0x1 = ERR. During a GWALK instruction, the HTE encountered an error (ECC or parity) while processing arc fills from memory or graph cache (RAM1,2,3).

→ RAM parity errors are also locked down in DFA_ERROR[DC0PERR] along with the corresponding failing RAM address (see DFA_DTCFADR[RAM*n*FADR]).

→ ECC errors (in DDR3 memory) are locked down in LMC memory (see LMC*n*_INT, LMC*n*_FADR, and LMC*n*_ECC_SYND).

ERR has highest priority.

NOTE:

Software Note:

In most cases (except a CNDRD parity error), a parity error detected terminates the GWALK instruction with RWORD0[REAS] = ERR. Also an RWORD1+[PDBOFF, NNPTR, F1, F2, F3] is written that reflects Graph-Walk-instruction information for the previous node that launched the request, to help in isolation of the error.

The parity-error information is also capture in DFA_ERROR[DC0PERR] along with the failing address in DFA_DTCFADR[RAM*n*FADR] field. A maskable interrupt is generated if the corresponding RAM's DFA_INTMSK[DC0PENA] bit is set.

Cavium Confidential For Bowen Yang ALU 3/2/2014

0x2 = RFULL. The HTE consumed the last result word when it processed a packet data byte. RFULL has lowest priority.

0x3 = TERM. The HTE found a terminal node ID when it processed a HTE byte. TERM is lower priority than ERR, but higher than DDONE or RFULL.

0x4 = GDONE. The HTE finished loading (CLOAD instruction) or freeing (GFREE instruction) a graph successfully.

0x5 = NOGRAPH. The HTE graph walk for a cacheable graph (IWORD0[F1] ≠ MONODE) was nonexistent (i.e. no MBASE comparison)

NOTE: **Implementation Note:** HFA hardware keeps track of CLOAD instructions and which VGID was captured. For cacheable graphs, GWALK instructions always check to ensure their VGID exists in GUT before use. If the VGID does not exist, then REAS = NOGRAPH is returned.

0x6: = GERR. Either a CLOAD instruction (IWORD0[ITYPE] = 0x1) was issued, and the HTE encountered errors due to software misprogramming; or a reserved instruction ([ITYPE] = 0x0, 0x2, 0x5–0x7) was issued.

NOTE: **Software Restrictions:**
IWORD2[DLEN<15:3>] ≥ (32 + IWORD2[DSIZE]/2).

- +32 is required to load entire RAM3 graph table.
- +DSIZE/2 specifies how much RAM2 to load (starting at DBASE).
- the remaining DLEN doublewords are loaded to RAM1 (starting at CBASE).

NOTE: **Software Note:**
In the case a CLOAD operation returns RWORD0[REAS] = GERR, (i.e. misprogrammed CLOAD), software must issue a GFREE instruction with the same VGID, to clear internal VGID valid state inside the HFA hardware. Only an error-free CLOAD can be used by software to gate the issuance of GWALK instructions to the same VGID.

0x7 = Reserved

NOTE: **Software Note:**
During a GWALK instruction, it is possible to encounter any/all of the ERR, TERM, DDONE and RFULL conditions all within the same cycle. In these cases, the highest-to-lowest reporting priority are {ERR, TERM, DDONE, RFULL}.

RWORD0[19:18] [F5] Partial parse status. If application software requires to continue/resume inspection of subsequent packet of same flow from where current packet stopped, it copies RWORD0[19:18] into IWORD2[47:46] while submitting next GWALK instruction.

RWORD0[17] [M] Last marked node.
Set to 1 when the last result word was created from a marked transition.
Clear to 0 when the last result word was not created from a marked transition.

NOTE:
M is unpredictable when REAS = 0x1 (ERR).
M is always 0 when REAS = 0x3 (TERM).
M is always 1 when REAS = 0x2 (RFULL).

RWORD0[16] [D] Done processing.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Set to 1 by the hardware after it completes the instruction and all results have been written. Used by polling software to recognize when the instruction has completed.

NOTE: Hardware ensures that all of the results (RWORD1+) have properly been written/committed to L2C/DRAM memory, before the DONE bit is set.

RWORD0[15:0] [RNUM] Number of result words.
Indicates the number of 64-bit result words that follow RWORD0. RNUM + 1 is the total number of 64-bit words in the result. RNUM never exceeds IWORD1[RMAX].

RWORD1 Definition:

RWORD1[63:48] [PDBOFF] Packet-data byte offset. Indicates the byte offset at which the last input HTE byte was processed.
If the HTE created this word during its processing of the first HTE byte, PDBOFF = 0.

If the HTE created this word during its processing of the nth HTE byte, PDBOFF = (n–1). PDBOFF can never exceed one less than the number of HTE bytes IWORD2[DLEN] or the sum of the gather-list lengths, depending on IWORD0[GM].

RWORD1[47:39] [F3] Partial parse status. If application software requires to continue/resume inspection of subsequent packet of same flow from where current packet stopped, it copies RWORD1[47:39] into IWORD3[48:40] while submitting next GWALK instruction.

RWORD1[38:31] [F2] Partial parse status. If application software requires to continue/resume inspection of subsequent packet of same flow from where current packet stopped, it copies RWORD1[38:31] into IWORD1[47:40] while submitting next GWALK instruction.

RWORD1[30:28] [F1] Partial parse status. If application software requires to continue/resume inspection of subsequent packet of same flow from where current packet stopped, it copies RWORD1[30:28] into IWORD0[63:61] while submitting next GWALK instruction.

RWORD1[27] [UDATA<0>] The HFA compiler embeds some information in the HFA graph nodes that are used for post-processing.

RWORD1[26:0] [NNPTR] Next-node pointer (relative to the graph).
When RWORD0[REAS] = {DDONE,RFULL,TERM}, NNPTR references the next-node pointer that should be used to process any subsequent HTE bytes.
When RWORD0[REAS] = {ERR,NOGRAPH}, NNPTR references the current-node pointer that encountered the error.

NOTE: NNPTR is a graph-relative address (i.e. does not include the +GUT.*BASE offset(s) to ensure a continuation assigned to a different cluster will use the proper +GUT.*BASE).

11.4.5 HFA Graph Formats

The HTEs support two graph formats: cacheable graphs and noncacheable (or memory-only) graphs.

Cavium Confidential For Bowen Yang AL U 02/27/2014

11.4.5.1 Cacheable Graphs

For higher performance, HTEs support caching of the GIG of the whole graph into on-chip graph cache. The HFA compiler has the option while compiling to chose between cacheable or memory-only graphs.

For details on cacheable graphs, refer to Cavium HFA SDK documentation.

11.4.5.2 Memory-Only Graph Format

In addition to the 64 cacheable graphs partially housed in the internal graph cache/ HFA memory, there can also be an unlimited number of memory-only HFA graphs (MONODE) that reside solely within HFA memory. The number of memory-only graphs is only restricted by the physical size of HFA memory.

11.4.6 Loading Graphs

At any given time, there can be 64 cacheable HFA graphs[1] housed within the internal graph cache+memory data structures, as well as a number of memory-only graphs (limited only by available external memory). Software must properly load these graphs before issuing GWALK instructions that use them.

Software is responsible for managing graphs housed in the internal graph cache and in HFA memory. There are two types of graphs (cacheable and memory-only), and a request is differentiated by IWORD0[F1].

- If IWORD0[F1] = 0x6, it is a memory-only graph request.
- If IWORD0[F1] ≠ 0x6, it is a cacheable graph request.

11.4.6.1 Cacheable Graph Loading

Cacheable graphs are stored in both the internal graph cache and to external HFA memory and require a SW_MLOAD SDK routine call followed by a CLOAD instruction (with proper VGID/PGID/CBASE/DBASE). It is important for software to perform the graph-load instructions in this order for internal hardware to properly capture hardware state to validate the graphs for future use.

- The initial SW_MLOAD SDK routine call loads the external HFA memory in DDR3 at the MBASE starting address.
- The CLOAD instruction loads the targeted physical graph cache using the PGID/ CBASE/DBASE fields.
  - The PGID specifies the physical graph within the node cluster.
  - the CBASE/DBASE specifies the RAM1/2 base addresses to properly load the HFA graph cache data.

At this time, the graph-use table (GUT) is also initialized. The GUT entry specified by the VGID also loads the CBASE, DBASE, and PGID fields. Any GWALK instructions issued prior to the completion of the cache load will always terminate with a no-graph status in the result word. Likewise, any GWALK instructions issued after the completion of the cache load will source newly loaded HFA memory/cache data for proper operation.

A CLOAD instruction can never be issued with IWORD0[F1] = 0x6.

Cavium Confidential For Peiwen Yang 10/2/2014

1. There are 64 graphs visible to hardware. With the help of the software linker, application software can load up to 512 cacheable graphs into graph cache.

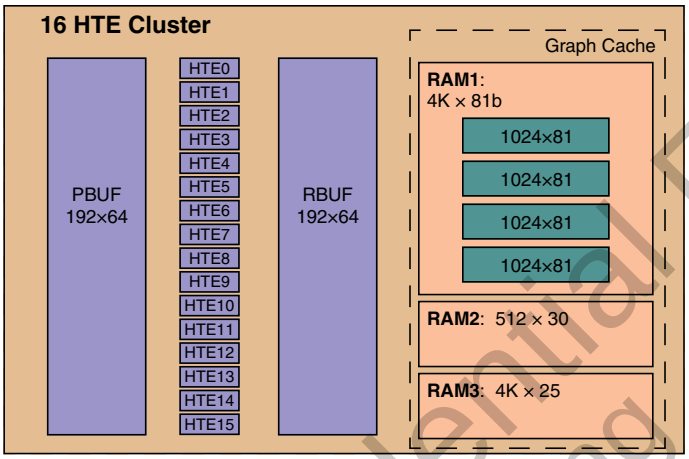
11.4.6.2 Graph CacheLoad Data Format

The CLOAD data format setup by software in system memory to be loaded into the internal graph cache is implementation specific. Physically, the graph cache is realized as three underlying RAM structures:

1. RAM1: 4K × 81 bits
2. RAM2: 512 × 28 bits
3. RAM3: 4K × 25 bits

Each of these three RAM structures is paired with 16 unique HTEs, to constitute a node cluster. This is shown in Figure 11–12.

Figure 11–12 Node-Cache Structure



RAM1 is a 4K × 81 data structure. Each row (of 81 bits) is called a graph cacheline. The number of graph cachelines allocated for a graph is configurable during the pattern-compilation stage. Refer to the Cavium Regex SDK notes for more details.

The RAM1 physical partitioning allows each graph's variable-sized data structures to be efficiently packed to leave no entries to waste in the graph cache. Software knows the base address of each graph in RAM1, which is reflected in the HFA instruction IWORD0[CBASE].

RAM2 is a 512 × 30 data structure, with each row being 30 bits wide; the number of rows per graph is variable in size. The RAM2 physical partitioning allows each graph's variable-sized RAM2 data structures to be efficiently packed to leave no entries to waste in the graph cache. Software knows the base address of each graph in RAM2, which is reflected in IWORD0[DBASE].

RAM3 is a 4096 × 25 data structure, with each row being 25 bits wide; the number of rows is fixed per graph. Each graph contains 64 entries that are stored in RAM3. The RAM3 address is reflected in IWORD0[PGID].

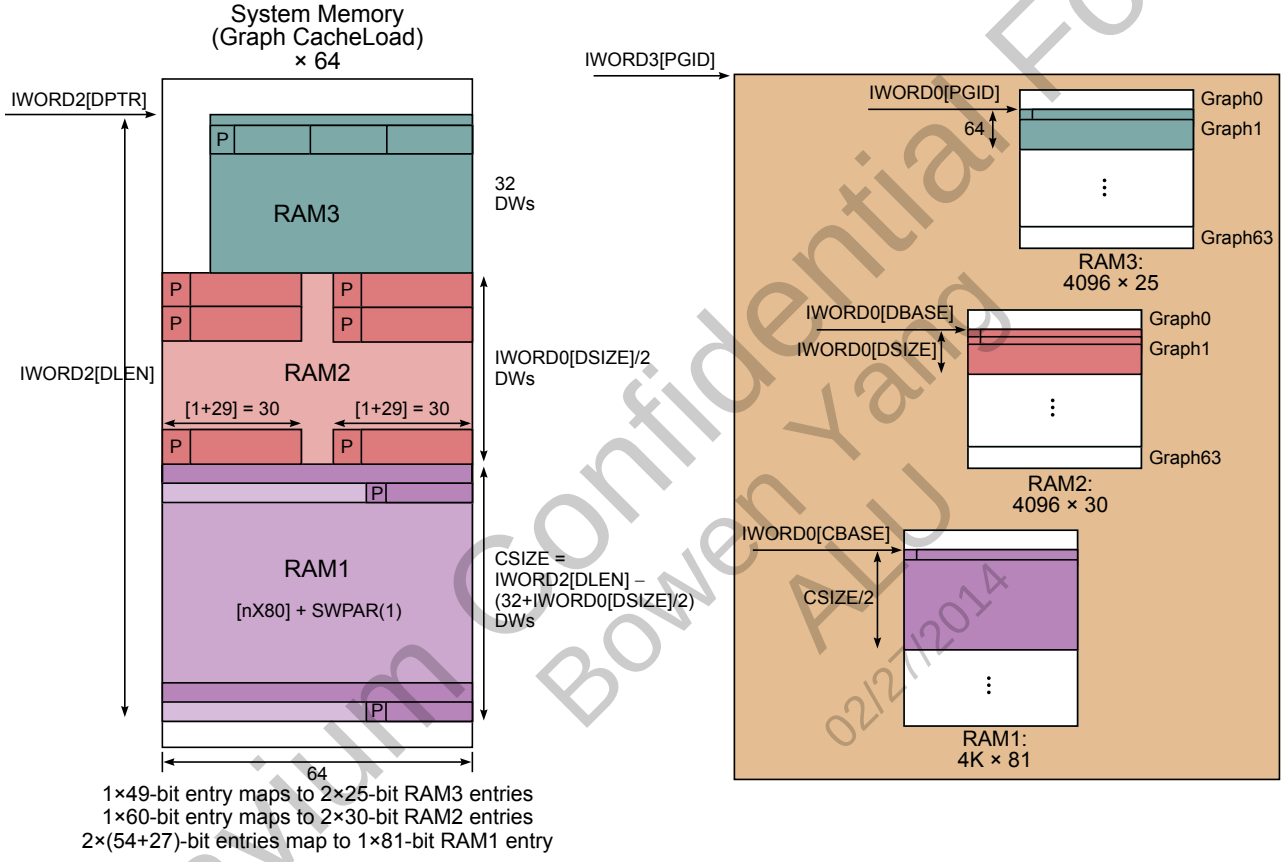
NOTE: **Software Note**: It is up to software to manage the physical allocation of the graph cache across physical node clusters and cluster graphs. The CN71XX implementation-specific details of the graph cache require the CLOAD instructions to be aware of the 64 cluster graphs (PGID) and the cluster node base addresses (CBASE/DBASE).

To load a cacheable HFA graph into the internal cache/external HFA memory, software must perform the following two steps in order:

Cavium Confidential For Bowen Fang ABU 6/27/2014

1. Load HFA memory using an SW_MLOAD SDK routine call.
2. Load the graph-cache data structures using a CLOAD instruction with the following parameters (refer to Figure 11–13):
   - IWORD0[PGID<5:0>]: determines which physical graph is to be loaded.
   - IWORD0[VGID]: determines which virtual graph to load.
   - IWORD0[CBASE]: determines the one-entry-aligned starting address to load RAM1 data into the cluster graph cache.
   - IWORD0[DBASE]: determines the one-entry-aligned starting address to load RAM2 data into the cluster graph cache.
   - IWORD0[DSIZE]: determines the number of entries to load into RAM2.
   - IWORD2[DLEN]: specifies the total number of doublewords to read from system memory required to perform the cacheload operation.

Figure 11–13 Graph-Cache Data Structures



Cavium Confidential For Bowen Yang ALU 02/27/2014

NOTE:

Software Notes:

1. The first 32 doublewords of system memory (at IWORD2[DPTR]) are mapped to the 64 entries in the RAM3 starting at offset IWORD0[PGID]<<6. Note that eight entry/sysMem doublewords are mapped to two four-entry/RAM3 entries.
2. The next RNDUP(IWORD0[DSIZE]/2) doublewords of system memory are mapped into IWORD0[DSIZE] entries in RAM2 starting at offset IWORD0[DBASE].

   Note that one system memory doublewords contains two entries that are mapped to two RAM2 entries.
3. The remaining (IWORD2[DLEN] – (32+RNDUP(IWORD0[DSIZE]/2))) doublewords of system memory are mapped into RAM1 entries starting at offset IWORD0[CBASE]. Note that, since the RAM1 is 81 bits wide, the first doubleword (64 bits) is merged with the second doublewords (remainder) to generate a single 81-bit RAM1 entry.

Software is responsible for pregenerating the proper software parity bits that are loaded directly into the graph cache. Hardware checks parity of the RAMs only during read operations required during a GWALK instruction.

NOTE:

Software Restrictions: Please note that DLEN/DSIZE parameters must be properly programmed during a CLOAD.

1. DLEN[2:0] = 0. DLEN must be at minimum a doubleword multiple.
2. DLEN[15:3] ≥ (32 + RNDUP(DSIZE/2)). Otherwise CLOAD will report RWORD0[REA] = GERR.
3. DLEN[15:3] – (32 + RNDUP(DSIZE/2) must be an quadword multiple (since each RAM1 entry requires two system memory doublewords).
4. DSIZE[8:0] ≤ 256 (DNODEs/Graph = 256).

Graph Allocation

The GUT[VGID] entry is validated and the PGID/CBASE/DBASE values from the instruction are written when the CLOAD instruction is first started. It is up to software to manage the coherency of valid graphs (refer to the ordering rules in IWORD0[26:24] [ITYPE] on page 604). It remains allocated for use by future HFA instructions until a GFREE instruction is issued, which deallocates a graph from cache-node space.

Note that the order of the steps in the above note is very important, because the graph-cache data structures must be loaded before the graph ID is validated for use. The hardware maintains a GUT that is validated when the CLOAD instruction is started.

NOTE:

Software Note: Only one bit may be set in CLMSK during a CLOAD instruction.

11.4.6.3 Memory-Only Graph Loading

Memory-only graphs are stored only to HFA memory, and therefore only require an SW_MLOAD SDK routine call with an MBASE value. There can be an unlimited number of MONODE graphs in external memory, only bound by the overall size of attached DDR3 memory. The HFA memory is loaded starting at 16-byte offset (i.e. IWORD0[MBASE] << 6) with total graph size(16-byte) = {IWORD1[RMAX<11:0>], IWORD2[DLEN<15:0>]}.

Cavium Confidential For Bowen Yang AU 02/27/2014

11.4.7 Freeing Cacheable HFA Graphs

To free a cacheable HFA graph from the internal graph cache, software must issue a GFREE instruction, where:

- IWORD3[VGID] provides the virtual graph ID (1 of 256)
- IWORD2[CLMSK] specifies the node clusters to free.

NOTE: **Software Note:** Only one bit must be set in CLMSK during a GFREE instruction

Upon successful completion, HFA returns RWORD0[REAS] = 0x4 (GDONE) in response to the instruction.

11.5 HFA Registers

The HFA registers are listed in Table 11–1.

Table 11–1 HFA Registers

Register	Address	CSR Type[1]	Detailed Description
DFA_CONFIG	0x0001180037000000	RSL	See page 617
DFA_CONTROL	0x0001180037000020	RSL	See page 618
DFA_ERROR	0x0001180037000028	RSL	See page 619
DFA_INTMSK	0x0001180037000030	RSL	See page 620
DFA_DEBUG2	0x0001180037000050	RSL	See page 620
DFA_DEBUG3	0x0001180037000058	RSL	See page 621
DFA_DTCFADR	0x0001180037000060	RSL	See page 621
DFA_BIST0	0x00011800370007F0	RSL	See page 622
DFA_BIST1	0x00011800370007F8	RSL	See page 622
DFA_DBELL	0x0001370000000000	NCB	See page 623
DFA_DIFRDPTR	0x0001370200000000	NCB	See page 623
DFA_DIFCTL	0x0001370600000000	NCB	See page 624

1. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

HFA Configuration Register
DFA_CONFIG

See Table 11–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:11>	—	RAZ	—	—	Reserved
<10>	DLCCLEAR_BIST	R/W	0	0	When DLCSTART_BIST is written 0→1, if DLCCLEAR_BIST = 1, all previous DLC BIST state is cleared. Note the following: **1.** DLCCLEAR_BIST must be written to 1 before DLCSTART_BIST is written to 1 using a separate CSR write. **2.** DLCCLEAR_BIST must not be changed after writing DLCSTART_BIST 0→1 until the BIST operation completes.
<9>	DLCSTART_BIST	R/W	0	0	When software writes DLCSTART_BIST = 0→1, a BIST is executed for the DLC sub-block RAMs which contains DCLK domain asynchronous RAMs. Note the following: **NOTE:** This bit should only be written after DCLK has been enabled by software and is stable. (See LMC initialization routine for details on how to enable the DDR3 memory (DCLK)—which requires LMC PLL init, clock divider, and proper DLL initialization sequence.)
<8>	REPL_ENA	R/W	0	0	Replication mode enable. This bit must be zero for proper operation, since there is only a single LMC port, port replication is not supported by hardware.
<7:4>	CLMSKCRIP	R/W	0x0	0x0	Cluster cripple mask. A 1 in each bit of the mask represents which HTE cluster to cripple. **NOTE:** CN71XX has only a single cluster (therefore CLMSKCRIP<0> is the only bit used.) **NOTE:** The MIO_FUS_DFA_CLMASK_CRIPPLE[2:0] fuse bits are forced into this register at reset. Any fuse bits that contain 1 are disallowed during a write and are always read as 1.

Cavium Confidential
Bowen Yang
ALU
02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<3:1>	CLDTECRIP	R/W	0x0	0x0	Encoding that represents the number of HTEs to cripple. Typically DTE_CLCRIP = 0, which enables all HTEs. However, when the HFA performance counters are used, software may want to limit the number of HTEs available, as there are only four parallel performance counters. DTE_CLCRIP — No. of HTEs crippled 0 — 0 — HTE[15:0]:ON 1 — 1/2 — HTE[15:8]:OFF /HTE[7:0]:ON 2 — 1/4 — HTE[15:12]:OFF /HTE[11:0]:ON 3 — 3/4 — HTE[15:4]:OFF /HTE[3:0]:ON 4 — 1/8 — HTE[15:14]:OFF /HTE[13:0]:ON 5 — 5/8 — HTE[15:6]:OFF /HTE[5:0]:ON 6 — 3/8 — HTE[15:10]:OFF /HTE[9:0]:ON 7 — 7/8 — HTE[15:2]:OFF /HTE[1:0]:ON **NOTE:** Higher numbered HTEs are crippled first. For instance, on CN71XX (with 16 HTEs/cluster), if DTE_CLCRIP = 1 (½), then [15:8] within the cluster are crippled and only HTE numbers [7:0] are available. **NOTE:** The MIO_FUS_DFA_NUMDTE_CRIPPLE[2:0] fuse bits are forced into this register at reset. Any fuse bits that contain 1 are disallowed during a write and are always read as 1.
<0>	DTECLKDIS	R/W	1	0	HFA clock disable source. When set, the HFA clocks for HTE (thread engine) operations are disabled to conserve overall chip clocking power when the HFA function is not used. **NOTE:** When set, software must never issue IOI-direct CSR operations to the HFA (will result in IOI timeout errors). **NOTE:** This should only be written to a different value during power-on software initialization. **NOTE:** The MIO_FUS_DFA_DTE_DISABLE fuse bit is forced into this register at reset. If the fuse bit contains 1, write operations to DTECLKDIS are disallowed and are always read as 1.

HFA Control Register
DFA_CONTROL

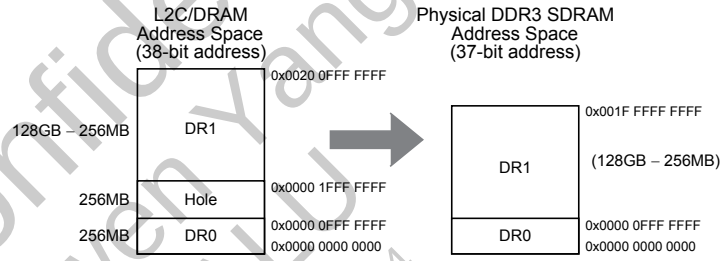
See Table 11–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved
<11:6>	SBDNUM	R/W	0x0	0x0	Reserved
<5>	SBDLCK	R/W	0	0	Reserved
<4:3>	—	RAZ	—	—	Reserved
<2>	PMODE	R/W	1	1	Reserved
<1>	QMODE	R/W	1	1	Reserved
<0>	IMODE	R/W	1	1	Reserved

Cavium Confidential
Bowen Yang
ALU
02/27/2014

HFA Error Register
DFA_ERROR

See Table 11–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:19>	—	RAZ	—	—	Reserved
<18>	REPLERR	R/W1C	0	0	HFA illegal replication factor error. HFA only supports 1×, 2×, and 4× port replication. Legal configurations for memory are to support 2 port or 4 port configurations. The REPLERR interrupt is set in the following illegal configuration cases: **1.** An 8x replication factor is detected for any memory reference. **2.** A 4x replication factor is detected for any memory reference when only 2 memory ports are enabled. **NOTE:** If REPLERR is set during a HFA graph walk operation, the walk prematurely terminates with RWORD0[REA]=ERR. If REPLERR is set during a IOI-direct CSR read access to HFA memory region, the CSR read response data is unpredictable.
<17>	DFANXM	R/W1C	0	0	HFA nonexistent memory access. DTEs (and backdoor CSR HFA memory region reads) have access to the following 38bit L2/DRAM address space, which maps to a 37-bit physical DDR3 SDRAM address space. **DR0:** 0x0 0000 0000 0000 to 0x0 0000 0FFF FFFF maps to lower 256MB of physical DDR3 SDRAM **DR1:** 0x0 0000 2000 0000 to 0x0 0020 0FFF FFFF maps to upper 127.75GB of DDR3 SDRAM  In the event the HFA generates a reference to the L2/DRAM address hole (0x0000.0FFF.FFFF – 0x0000.1FFF.FFFF) or to an address above 0x0020.0FFF.FFFF, the DFANXM programmable interrupt bit will be set. **Software Note:** Both the 1) software HFA graph compiler and the 2) software IOI-direct CSR accesses to HFA memory region must avoid making references to these nonexistent memory regions. **NOTE:** If DFANXM is set during a HFA graph walk operation, the walk is prematurely terminated with RWORD0[REA]=ERR. If DFANXM is set during a IOI-direct CSR read access to HFA memory region, the CSR read response data is forced to 128'hBADE_FEED_DEAD_BEEF_FACE_CAFE_BEAD_C0DE. (The 64-bit doubleword being accessed, either the upper or lower doubleword will be returned).
<16>	CNDRD	RO	0	0	If DC0PERR<0> = 1 indicating a RAM1 parity error, this additional bit further specifies that the RAM1 parity error was detected during a CND-RD (cache node metadata read).
<15:14>	—	RAZ	—	—	Reserved
<13>	DLC0_ OVFERR	R/W1C	0	0	DLC0 FIFO overflow error detected. This condition should never architecturally occur, and is here in case hardware credit/debit scheme is not working.

Cavium Confidential Ben Yang 02/2/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<12:4>	—	RAZ	—	—	Reserved
<3:1>	DC0PERR	R/W1C	0x0	0x0	Cluster 0 RAM[3:1] parity error detected. See also DFA_DTCFADR register, which contains the failing addresses for the graph-cache RAMs.
<0>	DBLOVF	R/W1C	0	0	Doorbell overflow detected – status bit. When set, the 20b accumulated doorbell register had overflowed (software wrote too many doorbell requests). If the DBLINA had previously been enabled (set), an interrupt will be posted. Software can clear the interrupt by writing a 1 to this register bit. **NOTE:** Detection of a doorbell register overflow is a catastrophic error that may leave the HFA hardware in an unrecoverable state.

HFA Error Interrupt Mask Register
DFA_INTMSK

See Table 11–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:19>	—	RAZ	—	—	Reserved
<18>	REPLERRENA	R/W	0	0	HFA illegal replication factor interrupt enable
<17>	DFANXMENA	R/W	0	0	HFA nonexistent memory access interrupt enable
<16:4>	—	RAZ	—	—	Reserved
<3:1>	DC0PENA	R/W	0x0	0x0	RAM<3:1> parity error enabled node cluster 0.
<0>	DBLINA	R/W	0	0	Doorbell overflow interrupt enable bit. When set, doorbell overflow conditions are reported.

HFA Scoreboard Debug 2 Register
DFA_DEBUG2

When the DFA_CONTROL[SBDLCK] bit is written to 1, the contents of this register are locked down. Otherwise, the contents of this register are the “active” contents of the HFA scoreboard at the time of the CSR read. See Table 11–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	SBD2	RO	—	—	HFA scoreboard 2 data. <63:45> (19) UNUSED <44:42> (3) Instruction type <41:5> (37) rwptr<39:3>: Result write pointer <4:0> (5) prwcnt<4:0>: Pending result write counter

Cavium Confidential For Baiwen Yang 02/27/2014

HFA Scoreboard Debug 3 Register
DFA_DEBUG3

When the DFA_CONTROL[SBDLCK] bit is written to 1, the contents of this register are locked down. Otherwise, the contents of this register are the “active” contents of the HFA scoreboard at the time of the CSR read. See Table 11–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	SBD3	RO	—	—	HFA scoreboard 3 data. <63:52> (11) rptr<39:29>: Result base pointer (QW-aligned) <52:16> (37) glptr<39:3>: Gather list pointer <15:0> (16) glcnt Gather list counter

HFA DTC Failing Address Register
DFA_DTCFADR

This register contains HFA graph-cache failing address/control error capture information. It contains useful information to help in isolating a graph-cache RAM failure.

NOTE: The first detected PERR failure is captured in DFA_DTCFADR (locked down), until the corresponding PERR interrupt is cleared by writing 1 (W1C). (See DFA_ERROR[DC0PERR].)

See Table 11–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:44>	—	RAZ	—	—	Reserved
<43:32>	RAM3FADR	RO	—	—	HFA RAM3 failing address. If DFA_ERR[DC0PERR<2>] = 1, this field indicates the failing RAM3 address. The failing address is locked down until the DC0PERR<2> W1C occurs.
<31:25>	—	RAZ	—	—	Reserved
<24:16>	RAM2FADR	RO	—	—	HFA RAM2 failing address. If DFA_ERR[DC0PERR<1>] = 1, this field indicates the failing RAM2 address. The failing address is locked down until the DC0PERR<1> W1C occurs.
<15:14>	—	RAZ	—	—	Reserved
<13:0>	RAM1FADR	RO	—	—	HFA RAM1 failing address. If DFA_ERR[DC0PERR<0>] = 1, this field indicates the failing RAM1 address. The failing address is locked down until the DC0PERR<0> W1C occurs.

Cavium Confidential For Bowen Yang 9/27/2016

HFA BIST Status (per-DTC) Register
DFA_BIST0

This register shows the result of the BIST run on the HFA (per-DTC).

1 = BIST error, 0 = BIST passed, is in progress, or never ran.

See Table 11–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	0x0	0x0	Reserved
<24>	GFB	RO	0	0	BIST results for GFB RAMs.
<23:18>	—	RAZ	0x0	0x0	Reserved
<17:16>	STX	RO	0x0	0x0	BIST results for STX RAMs
<15:10>	—	RAZ	0x0	0x0	Reserved
<9:8>	DTX	RO	0x0	0x0	BIST results for DTX RAMs
<7:5>	—	RAZ	0x0	0x0	Reserved
<4>	RDF	RO	0	0	BIST results for RWB[3:0] RAMs
<3:1>	—	RAZ	0x0	0x0	Reserved
<0>	PDB	RO	0	0	BIST results for PDB RAMs

HFA Bist Status (Globals) Register
DFA_BIST1

This register shows the result of the BIST run on the HFA (globals).

1 = BIST error, 0 = BIST passed, is in progress, or never ran.

See Table 11–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	0x0	0x0	Reserved
<12>	RAM3	RO	0	0	BIST results for RAM3 RAM
<11>	RAM2	RO	0	0	BIST results for RAM2 RAM
<10>	RAM1	RO	0	0	BIST results for RAM1 RAM
<9>	CRQ	RO	0	0	BIST results for CRQ RAM
<8>	GUTV	RO	0	0	BIST results for GUTV RAM
<7:5>	—	RAZ	0x0	0x0	Reserved
<4>	GUTP	RO	0	0	BIST results for GUTP RAM
<3>	NCD	RO	0	0	BIST results for NCD RAM
<2>	GIF	RO	0	0	BIST results for GIF RAM
<1>	GIB	RO	0	0	BIST results for GIB RAM
<0>	GFU	RO	0	0	BIST results for GFU RAM

Cavium Confidential For Bowen Yang 9/27/2014

HFA Doorbell Register
DFA_DBELL

To write to DFA_DBELL, a device issues an IOBST request directed at the HFA with addr[34:33] = 0x0. To read the DFA_DBELL register, a device issues an IOBLD64 request directed at the HFA with addr[34:33] = 0x0.

If DFA_CONFIG[DTECLKDIS] = 1 (i.e. HFA-HTE clocks are disabled) or if FUSE[90] = "HFA HTE disable" is blown, read/write operations to DFA_DBELL do not take effect.

See Table 11–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:20>	—	RAZ	—	—	Reserved
<19:0>	DBELL	R/W	0x0	—	Represents the cumulative total of pending HFA instructions that software has previously written into the HFA instruction FIFO (DIF) in main memory. Each HFA instruction contains a fixed size 32B instruction word which is executed by the HFA hardware. The DBL register can hold up to 1M–1 (2^{20-1}) pending HFA instruction requests. During a read (by software), the "most recent" contents of the DFA_DBELL register are returned at the time the IOI-INB bus is driven. **NOTE:** Since HFA hardware updates this register, its contents are unpredictable in software.

HFA Instruction FIFO (DIF) RDPTR Register
DFA_DIFRDPTR

To write to the DFA_DIFRDPTR register, a device issues an IOBST request directed at the HFA with addr[34:33] = 0x1. To read the DFA_DIFRDPTR register, a device issues an IOBLD64 request directed at the HFA with addr[34:33] = 0x1.

If DFA_CONFIG[DTECLKDIS] = 1 (i.e. HFA-HTE clocks are disabled) or if FUSE[90] = "HFA HTE disable" is blown, read/write operations to the DFA_DIFRDPTR register do not take effect.

See Table 11–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:40>	—	RAZ	—	—	Reserved
<39:5>	RDPTR	R/W	0x0	—	Represents the 32-byte-aligned address of the current instruction in the HFA Instruction FIFO in main memory. The RDPTR must be seeded by software at boot time, and is then maintained thereafter by HFA hardware. During the seed write operation (by software), RDPTR<6:5> = 0, since HFA instruction chunks must be 128-byte-aligned. During a read operation (by software), the most-recent contents of the RDPTR register are returned at the time the IOI-INB bus is driven. **NOTE:** Since HFA hardware updates this register, its contents are unpredictable in software (unless it is guaranteed that no new doorbell register write operations have occurred and the doorbell register is read as zero).
<4:0>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang ALU 02/27/2014

HFA Instruction FIFO (DIF) Control Register DFA_DIFCTL

To write to the DFA_DIFCTL register, a device issues an IOBST request directed at the HFA with addr[34:32] = 0x6. To read the DFA_DIFCTL register, a device issues an IOBLD64 request directed at the HFA with addr[34:32] = 0x6. This register is intended to only be written once (at power-up). Any future write operations could cause the HFA and FPA hardware to become unpredictable.

If DFA_CONFIG[DTECLKDIS] = 1 (i.e. HFA-HTE clocks are disabled) or if FUSE[90] = "HFA HTE disable" is blown, read/write operations to the DFA_DIFCTL register do not take effect.

See Table 11–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:26>	—	RAZ	—	—	Reserved
<25:20>	MSEGBASE	R/W	0x0	—	Memory segmentation base address. For debug purposes, back door accesses to the HFA memory are supported via IOI-direct CSR accesses to the HFA memory region (if addr[34:32]=5). However due to the existing IOI address decoding scheme, the address only offers a 4GB extent into the HFA memory region. Therefore, the MSEGBASE CSR field provides the additional upper memory address bits to allow access to the full extent of memory (128GB MAX). For HFA memory region read IOI-direct CSR accesses, the 38-bit L2/DRAM memory byte address is generated as follows: memaddr[37:0] = {DFA_DIFCTL[MSEGBASE],ncb_addr[31:3],3'b0} **NOTE:** See the upper 6 bits of the memory address are sourced from DFA_DIFCTL[MSEGBASE] CSR field. The lower 4GB address offset is directly referenced using the IOI address bits during the reference itself. **NOTE:** The DFA_DIFCTL[MSEGBASE] is shared amongst all references. As such, if multiple cores are accessing different segments in memory, there must be a software mutual exclusive lock during each HFA memory region access to avoid collisions between cores using the same MSEGBASE CSR field. **NOTE:** See also DFA_ERROR[DFANXM] programmable interrupt which is flagged if software tries to access non-existent memory space (address hole or upper unused region of 38bit address space).
<19:12>	DWBCNT	R/W	0x1	—	Represents the number of cache lines in the instruction buffer that may be dirty and should not be written back to memory when the instruction chunk is returned to the free page list. **NOTE:** Typically software will want to mark all HFA instruction memory returned to the free page list as DWB (Don't-Write-Back), therefore software should seed this register as: DFA_DIFCTL[DWBCNT] = (DFA_DIFCTL[SIZE] + 4)/4.

Cavium Confidential For Review Only 10/27/2011

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<11:9>	POOL	R/W	0x0	—	Represents the 3-bit buffer pool ID used by HFA hardware when the HFA instruction chunk is recycled back to the free page list maintained by the HPA hardware (once the HFA instruction has been issued).
<8:0>	SIZE	R/W	0x3	—	Represents the number of 32-byte instructions contained within each HFA instruction chunk. At power-on, software seeds the SIZE register with a fixed chunk-size (must be at least 0x3). HFA hardware uses this field to determine the size of each HFA instruction chunk, in order to: a) determine when to read the next HFA instruction chunk pointer which is written by software at the end of the current HFA instruction chunk (see HFA description of next chunk buffer Ptr for format). b) determine when a HFA instruction chunk can be returned to the free page list maintained by the FPA hardware.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 12

DMA Packet Interface Unit (DPI) and Switch Logic Interface Unit (SLI)

This chapter describes the following topics:

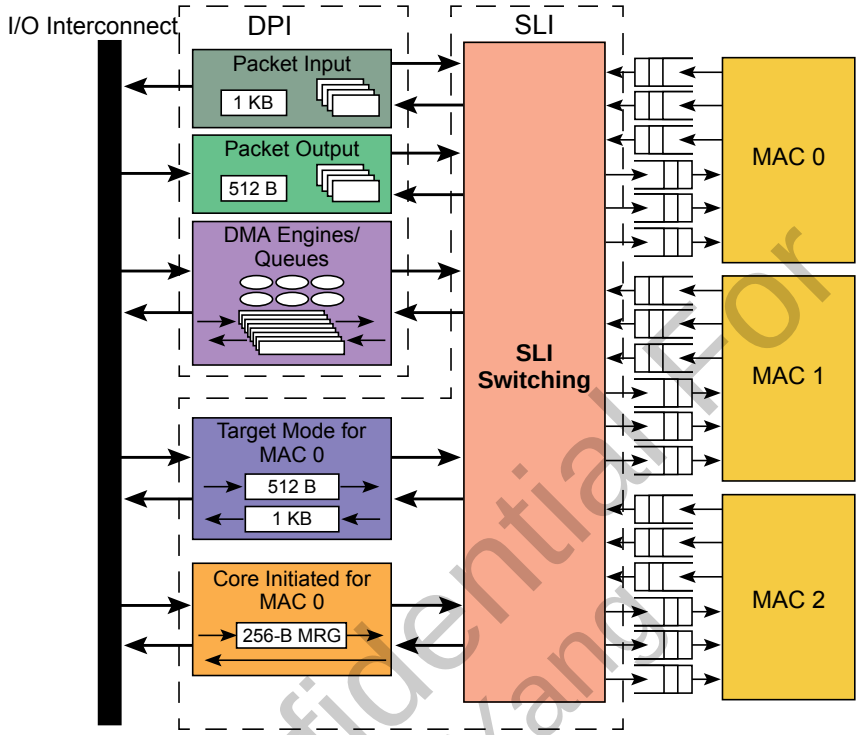
- Overview
- DPI Instruction Input From an External Host
- DPI Packet Output From CN71XX
- DPI DMA Queues and Engines
- SLI MAC BAR1/BAR2 Target Mode L2C/DRAM Load/Store Operations
- SLI Core-initiated Request Logic
- CN71XX as a MAC Interrupt Destination
- Endian Swapping
- SLI Bus Operations
- Configuration Summary
- DPI Registers
- SLI Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

CN71XX supports three PCIe MACs. The DMA packet interface (DPI) provides DMA support for the MACs, and the switch-logic interface (SLI) provides the interface with the cores, memory, and packet I/O. Figure 12–1 shows the DPI/SLI block diagram.

Figure 12–1 DPI/SLI Block Diagram



Cavium Confidential For
Bowen Yang
ALU
02/27/2014

12.1 DPI Instruction Input From an External Host

A PCIe host or other external device may submit instructions to DPI. This is a way to feed packets and/or commands into CN71XX, and to create CN71XX work-queue entries. Instructions are often submitted to DPI when CN71XX is a device on the PCIe bus, and may be less used in other circumstances.

The remaining description in this section assumes that CN71XX is a PCIe device and an external host inputs the instructions.

There are four DPI input ports in PIP/IPD (refer to Section 9.1) that are filled by a total of 32 DPI instruction input rings. DPI fills each port with packets that arrive via the eight DPI instruction input rings that are attached to it. The 32 rings are numbered 0..31 and the four DPI ports are numbered 32..35. (Refer to Section 9.1.)

The eight rings attached to a port are:

ring# = 4 × *m* + (port# – 32)

where *m* is the relative position of the ring in the port, ranging 0..7. For example, rings 0, 4, ... 24, 28 attach to port 32, and rings 3, 7, ... 27, 31 attach to port 35.

Each DPI instruction input ring has a separate enable (SLI_PKT_INSTR_ENB[ENB<*n*>] enables ring *n*). The DPI input ring can be disabled and re-enabled. Refer to Section 12.1.5.

DPI implements the following priority arbitration schemes:

- When SLI_PKT_INPUT_CONTROL[PKT_RR] = 0, the priority of the rings is fixed (highest to lowest) in this order: ring 0>1>2> ... 29>30>31.
- When SLI_PKT_INPUT_CONTROL[PKT_RR] = 1, the priority of each ring is equal and round-robin arbitration is used between the rings.

DPI fetches a maximum number of instructions from each ring before progressing to the next ring. This maximum is programmable on a per-ring basis (SLI_PKT_INSTR_RD_SIZE).

DPI maintains the following state for an instruction input ring:

- a configured base address (SLI_PKT*n*_INSTR_BADDR[ADDR])
- a physical DPI MAC number that the DPI input ring is attached to (SLI_PKT_IN_PCIE_PORT)
- a configured ring size (SLI_PKT*n*_INSTR_FIFO_RSIZE[RSIZE])
- a tail pointer (SLI_PKT*n*_INSTR_BAOFF_DBELL[AOFF] gives an approximate view of the tail, and SLI_PKT_IN_DONE*n*_CNTS[CNT] can also be used to determine where the tail is).

Note that each DPI input ring is attached to one of the two physical PCIe (i.e. MAC) ports that are attached to CN71XX. DPI directs all MAC read operations that implement each DPI input ring to the physical MAC port it is attached to. This includes the read operations for the DPI instructions and the read operations for the input packet data. Each DPI input ring may attach to a different physical MAC port, but a single DPI input ring only uses one physical MAC port.

Host software maintains the head of the ring and uses doorbell write operations (to SLI_PKT*n*_INSTR_BAOFF_DBELL[DBELL]) to notify DPI that the head advanced. DPI maintains the doorbell count (SLI_PKT*n*_INSTR_BAOFF_DBELL[DBELL]), which also indicates the current number of DPI instructions in the input queue, as well as the distance from head to tail.

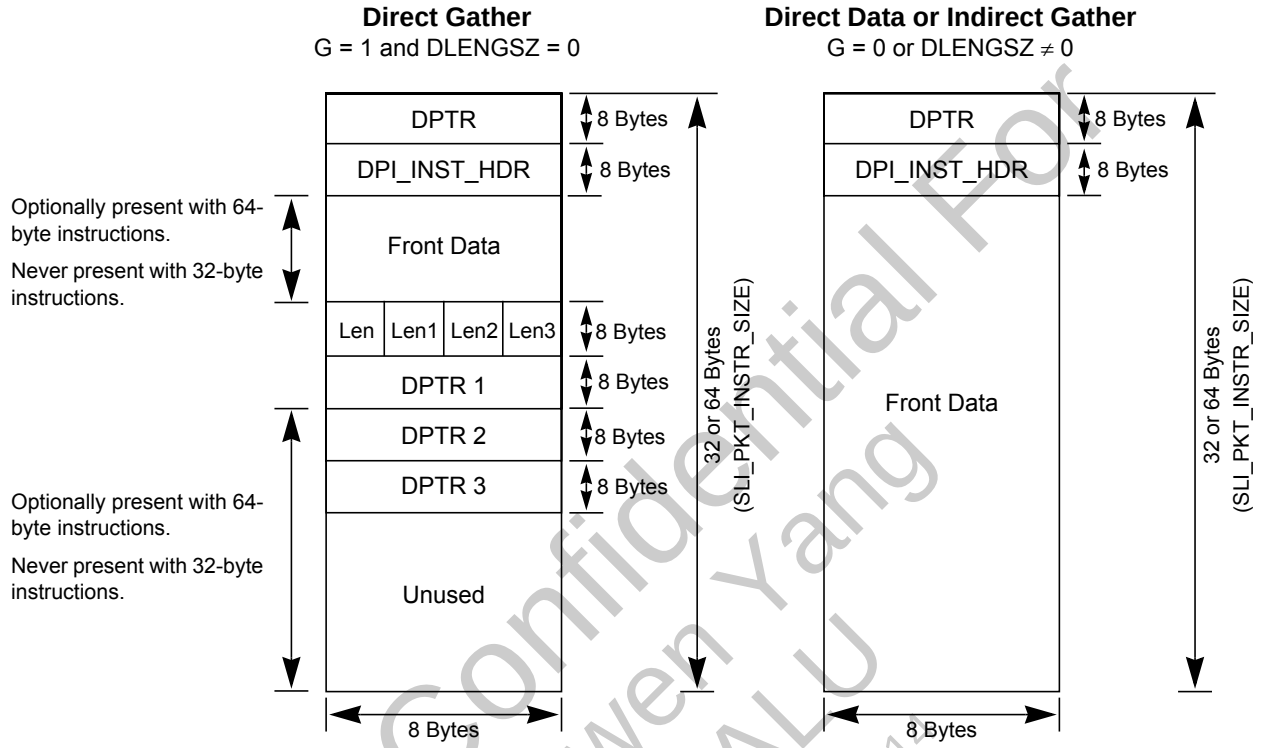
Cavium Confidential For Bowen Yang ALU 06/27/2014

Host software can poll the done count (SLI_PKT_IN_DONE*n*_CNTS[CNT]) associated with each ring to determine how far the hardware has advanced (and prevent ring overflow). DPI increments SLI_PKT_IN_DONE*n*_CNTS[CNT] by one for each instruction after it has read the instruction and all its associated data and gather lists. Thus, when SLI_PKT_IN_DONE*n*_CNTS[CNT] increments, host software can reuse the associated host memory locations of the tail DPI instruction.

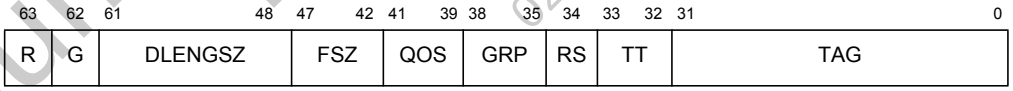
12.1.1 DPI Instruction Formats

Each DPI instruction is either 32 or 64 bytes, controlled by SLI_PKT_INSTR_SIZE[IS_64B<*n*>]. Figure 12–2 shows the instruction formats:

Figure 12–2 Instruction Formats



The DPI_INST_HDR format is as follows:



- **R** - RAW* packet.

  When R is set, the packet is either RAWFULL or RAWSCH, depending on the parse mode selected for the packet. RAWFULL and RAWSCH packets, as defined in Section 9.2.1, use the QOS, GRP, TT, and TAG fields in the DPI_INST_HDR for their scheduling information (refer to Chapter 9 for more details).

  When R is clear, the packet is neither RAWFULL nor RAWSCH, and the QOS, GRP, TT, and TAG fields are ignored by the hardware.

- **G** - gather is used.

  When G is set and DLENGSZ ≠ 0, the instruction is an indirect gather instruction (see Figure 12–2). In this case:

Cavium Confidential For Bowen Yang ALU 02/27/2015

- DPTR points at (indirect) gather-list components (see Figure 12–4) in the remote host's memory, and the pointers in that (indirect) gather list point at the packet data in the host's memory.
- DLENGSZ is the number of entries in the indirect gather list.
- The length of the packet is:

  [sum of the lengths of the DLENGSZ indirect gather-list entries] + FSZ + IHI × 8

  where IHI = 1 when the DPI instruction header is included at the beginning of the constructed packet.

When G is set and DLENGSZ = 0, the instruction is a direct gather instruction (see Figure 12–2). In this case:

- The instruction contains a number of DPTRs (DPTR, DPTR 1, ...) and associated lengths (LEN, LEN 1, ..., respectively) that point at the packet data in the host's memory.
- The length word (containing LEN, LEN 1, LEN 2, and LEN 3) is present immediately following the (eight-byte-aligned) Front Data, if front data is present. If the instruction is 32 bytes or if FSZ = 0, then front data is not present, and the length word immediately follows the DPI_INST_HDR in the direct gather instruction.
- LEN and LEN 1 must be nonzero and must indicate the number of input packet bytes pointed at by DPTR and DPTR 1, respectively.
- If LEN 2 is nonzero, then DPTR 2 is present and LEN 2 must indicate the number of input packet bytes pointed at by DPTR 2. LEN 2 must be zero for 32-byte direct gather instructions.
- If LEN 3 is nonzero, then DPTR 3 is present, and LEN 3 must indicate the number of input packet bytes pointed at by DPTR 3, and LEN 2 must also be nonzero. LEN 3 must be zero for 32-byte direct gather instructions.
- The length of the packet is:

  LEN + (LEN 1) + (LEN 2) + (LEN 3) + FSZ + IHI × 8

When G is clear, the instruction is a direct data instruction (see Figure 12–2). In this case:

- The instruction contains exactly one DPTR, which points directly at the packet data in the host's memory.
- The length of the packet is:

  DLENGSZ + FSZ + IHI × 8

- **DLENGSZ** - data length/gather-list size.

  When gather is not used (the G bit is clear), DLENGSZ, which must be nonzero, indicates the length of the packet data (length in bytes) directly pointed at by DPTR.

  When gather is used (the G bit is set), DLENGSZ selects between direct and indirect gather and indicates the number of entries in the indirect gather list. If DLENGSZ is zero, direct gather is selected. If DLENGSZ is not zero, indirect gather is selected. Note that with indirect gather, the number of indirect gather components used by the instruction is (DLENGSZ + 3)/4.

- **FSZ** - front-data size.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Indicates the number of bytes of packet data before the DPTR data (and after the PKT_INST_HDR when it is included in the constructed packet).

For the direct data and indirect gather cases (i.e. DPI_INST_HDR[G] = 0 or DPI_INST_HDR[DLENGSZ] ≠ 0, refer to Figure 12–2):

- DPI_INST_HDR[FSZ] must be ≤ 23 with a 32-byte instruction, and it must be ≤ 55 with a 64-byte instruction. An FSZ value between 17 and 23 with a 32-byte instruction can be used to insert a pad between the front data and the DPTR data. The hardware inserts unpredictable bytes for the extra bytes that are not actually contained in the DPI instruction in this case. The same is true for FSZ values between 49 and 55 with a 64-byte instruction.

For the direct gather case (i.e. DPI_INST_HDR[G] ≠ 0 and DPI_INST_HDR[DLENGSZ] = 0, refer to Figure 12–2):

- With 32-byte direct gather instructions:
  - FSZ must be zero (and the Front Data field is not present).
- With 64-byte direct gather instructions:
  - When DPI_INST_HDR[FSZ] is zero, the Front Data field is not present.
  - When FSZ is nonzero, the size of the Front Data field (refer to Figure 12–2) is FSZ rounded up to the nearest multiple of 8 bytes.
  - FSZ can legally range between 0 and 32 bytes (inclusive) when LEN 2 and LEN 3 are both zero.
  - FSZ can legally range between 0 and 24 bytes (inclusive) when LEN 3 is zero.
  - FSZ can legally range between 0 and 16 bytes (inclusive) in all circumstances.

- **QOS** - quality of service. Selects the one of the eight POW queues that holds the work-queue entry when the R bit is set (i.e. when the packet is RAWFULL or RAWSCH). QOS is ignored by the hardware when the R bit is clear. Refer to Chapters 7 and 9 for more details.
- **GRP** - group. Selects the core group that the work-queue entry is scheduled to when the R bit is set (i.e. when the packet is RAWFULL or RAWSCH). GRP is ignored by the hardware when the R bit is clear. Refer to Chapters 7 and 9 for more details.
- **RS** - real short. When RS is set, the ordinary L2/DRAM copy of the packet is avoided if the packet is dynamically considered short. Section 9.4 describes dynamic-short packets in more detail, but they are largely packets that can fit entirely in the work-queue entry created for the packet.

  The hardware ignores the RS bit if the corresponding PIP_PRT_CFG(32..35)[DYN_RS] bit is set, which indicates that the port should always avoid buffering dynamic-short packets that arrive via any ring attached to the port. The hardware also ignores the RS bit if a DPI_INST_HDR is not included with the constructed packet (i.e. if both the R bit and corresponding SLI_PKT*n*_INSTR_HEADER[USE_IHDR] are clear).

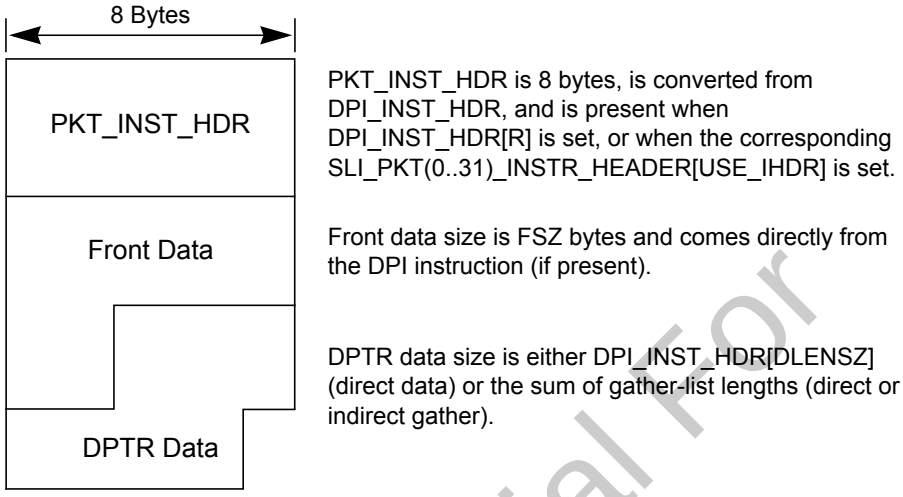
- **TT** - tag type. Selects the tag type that the work-queue entry has when the R bit is set (i.e. when the packet is RAWFULL or RAWSCH). TT is ignored by the hardware when the R bit is clear. Refer to Chapters 7 and 9 for more details.
- **TAG**. Selects the tag that the work-queue entry has when the R bit is set (i.e. when the packet is RAWFULL or RAWSCH). TAG is ignored by the hardware when the R bit is clear. Refer to Chapters 7 and 9 for more details.

Cavium Confidential For Bowen Yang ALU 10/27/2015

12.1.2 DPI Input Packets

Figure 12–3 shows the packet that DPI creates from a DPI instruction. The complete packet is the PKT_INST_HDR (optionally included, converted from the DPI_INST_HDR), front data (if any), plus the DPTR data.

Figure 12–3 Input Packet Format



The PKT_INST_HDR is included with the packet constructed for a DPI_INST instruction when one or both of the two following conditions is met:

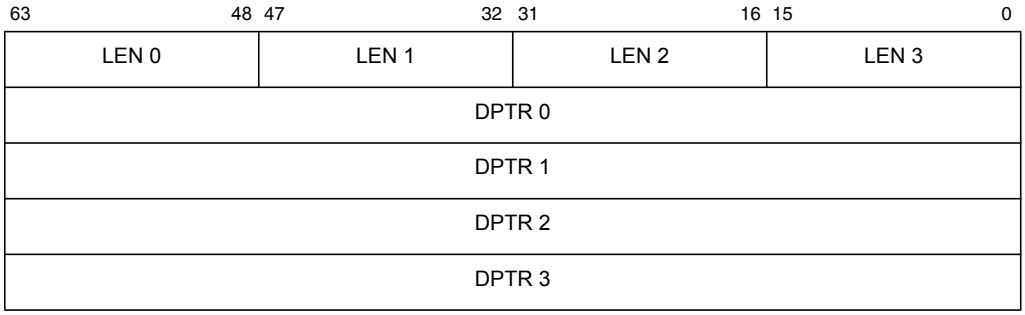
- DPI_INST_HDR[R] is set
- the corresponding SLI_PKT*n*_INSTR_HEADER[USE_IHDR] bit is set, indicating that the DPI instruction header should be included with all packets.

Whenever the DPI instruction header is included in the constructed packet, it is first converted to a PKT_INST_HDR. Sections 9.2.2 and 9.2.5 and Figure 9–3 describe the transformation from DPI_INST_HDR to PKT_INST_HDR in that case. SLI_PKT*n*_INSTR_HEADER provides the default values for <62:42> of the constructed header for each corresponding ring. Additionally, when the corresponding SLI_PKT*n*_INSTR_HEADER[PBP] is set, the parse mode and skip values come from the DPTR field directly in the DPI instruction, as described below. Refer to Section 9.2 for more details on parse mode and skip.

When a DPI instruction uses indirect gather mode (i.e. when DPI_INST_HDR[G] is set and DPI_INST_HDR[DLENGSZ] is nonzero), the DPTR in the DPI instruction points to the indirect gather list in the memory of the remote host. A indirect gather list is as many gather components as necessary to contain the entire indirect gather list. Figure 12–4 shows the structure of a DPI gather component. Each LEN, DPTR pair specifies a fragment of the DPTR data in the memory of the remote host. Each DPI gather component must be aligned on a 64-bit boundary (i.e. the DPTR<2:0> must be 0x0 in the DPI instruction when in gather mode), but the DPTRs in the component can have any byte alignment, and the LENs can be any (nonzero) byte length.

Cavium Confidential For Bowen Yang 02/12/2014

Figure 12–4 Structure of a DPI Gather Component Used in Indirect Gather Mode



DPI constructs DPI input packets by MAC read operations for the necessary DPI instructions, gather components, and DPTR data from the remote hosts memory. These MAC read requests require the following elements:

- an address
- an endian-swap mode (see Section 12.7)
- a two-bit ADDRTYPE
  - ADDRTYPE<0> is the relaxed-order attribute.
  - ADDRTYPE<1> is the no-snoop attribute.

There are four possible formats for the DPTRs in the DPI instruction and in the DPI gather components: format 0, format 1, format 2, and format 3.

These four formats produce the following five elements via various means:

1. MAC addresses
2. endian-swap modes
3. ADDRTYPE<1:0>
4. parse mode
5. skip lengths

The parse mode and skip lengths are only present in the DPTR in the DPI instruction when selecting the parse mode and skip length on a packet-by-packet basis (i.e. when SLI_PKT*n*_INSTR_HEADER[PBP] is set for the ring the packet arrived on).

Cavium Confidential For Bowen Yang ALU 02/27/2014

12.1.3 DPTR Formats

There are four DPTR formats that may be used depending on SLI_PKT_INPUT_CONTROL[USE_CSR] and various SLI_PKT*n*_INSTR_HEADER[PBP] values, as well as the type of instruction.

We define the various types of DPTRs (Figure 12–2 shows the different instruction types):

First Direct = First DPTR field in a direct data or direct gather instruction. These DPTRs always point to the first piece of data from the packet on the remote host (excluding PKT_INST_HDR and Front data, if they are present).

First Indirect = First DPTR field in a indirect gather instruction. These DPTRs always point to the indirect gather list on the remote host.

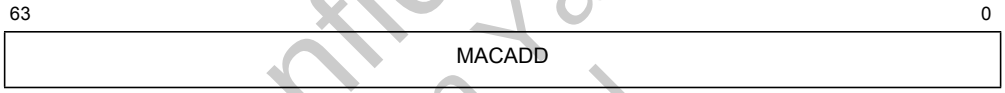
Gather = DPTRs from the indirect gather list fetched to process an indirect gather instruction. Also, DPTR 1, DPTR 2, and DPTR 3 from a direct gather instruction.

Table 12–1 describes the DPTR formats used in all cases.

Table 12–1 DPTR Formats

SLI_PKT_INPUT_CONTROL [USE_CSR]	SLI_PKT*n*_ INSTR_HEADER [PBP]	First Direct	First Indirect	Gather
0	0	Format 1	Format 0	Format 1
0	1	Format 3	Format 2	Format 1
1	0	Format 0	Format 0	Format 0
1	1	Format 2	Format 2	Format 0

DPTR Format 0



For format 0, the required elements are derived in the following manner:

1. MACADD<63:0> = MACADD field shown above
2. endian swap = Use SLI_PKT_INPUT_CONTROL[ESR] with First Indirect DPTRs; use SLI_PKT_INPUT_CONTROL[D_ESR] with First Direct and Gather DPTRs in Table 12–1.
3. ADDRTYPE<1> = Use SLI_PKT_INPUT_CONTROL[NSR] with First Indirect DPTRs; use SLI_PKT_INPUT_CONTROL[D_NSR] with First Direct and Gather DPTRs in Table 12–1.
4. ADDRTYPE<0> = Use SLI_PKT_INPUT_CONTROL[ROR] with First Indirect DPTRs; use SLI_PKT_INPUT_CONTROL[D_ROR] with First Direct and Gather DPTRs in Table 12–1.
5. parse mode = N/A
6. skip length = N/A

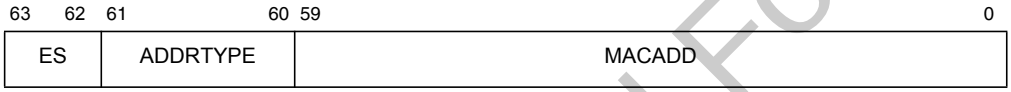
- Direct data mode: the DPTR in the DPI instruction is in DPTR format 0 when both of the following conditions are met:
  - SLI_PKT_INPUT_CONTROL[USE_CSR] = 1.
  - SLI_PKT*n*_INSTR_HEADER[PBP] = 0 for the ring that the packet arrived on.
- Indirect gather mode:

Cavium Confidential For Boren Yang 08/07/14

- the DPTR in the DPI instruction is in DPTR format 0 when SLI_PKT*n*_INSTR_HEADER[PBP] = 0 for the ring that the packet arrived on.
- the DPTRs in the DPI components that represent the gather list for the instruction are in DPTR format 0 when SLI_PKT_INPUT_CONTROL[USE_CSR] = 1.

- Direct gather mode:
  - the DPTR in the DPI instruction is in DPTR format 0 when both of the following conditions are met:
    - SLI_PKT_INPUT_CONTROL[USE_CSR] = 1.
    - SLI_PKT*n*_INSTR_HEADER[PBP] = 0 for the ring that the packet arrived on.
  - DPTR 1, DPTR 2, and DPTR 3 are in DPTR format 0 when SLI_PKT_INPUT_CONTROL[USE_CSR] = 1.

DPTR Format 1



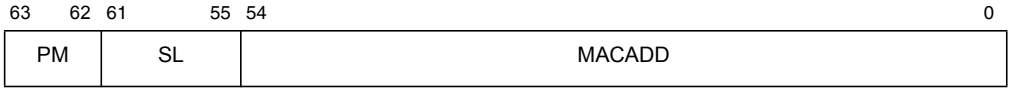
For format 1, the required elements are derived in the following manner:

1. MACADD<59:0> = MACADD field shown above
   MACADD<60> = SLI_PKT_INPUT_CONTROL[D_ROR]
   MACADD<61> = SLI_PKT_INPUT_CONTROL[D_NSR]
   MACADD<63:62> = SLI_PKT_INPUT_CONTROL[D_ESR]
2. endian swap = ES field shown above
3. ADDRTYPE<1:0> = ADDRTYPE field shown above
4. parse mode = N/A
5. skip length = N/A

- Direct data mode: the DPTR in the DPI instruction is in DPTR format 1 when both of the following conditions are met:
  - SLI_PKT_INPUT_CONTROL[USE_CSR] = 0.
  - SLI_PKT*n*_INSTR_HEADER[PBP] = 0 for the ring that the packet arrived on.
- Indirect gather mode:
  - the DPTR in the DPI instruction is never DPTR format 1.
  - the DPTRs in the DPI components that represent the gather list for the instruction are DPTR format 1 when SLI_PKT_INPUT_CONTROL[USE_CSR] = 0.
- Direct gather mode:
  - the DPTR in the DPI instruction is in DPTR format 1 when both of the following conditions are met:
    - SLI_PKT_INPUT_CONTROL[USE_CSR] = 0.
    - SLI_PKT*n*_INSTR_HEADER[PBP] = 0 for the ring that the packet arrived on.
  - DPTR 1, DPTR 2, and DPTR 3 are in DPTR format 1 when SLI_PKT_INPUT_CONTROL[USE_CSR] = 0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

DPTR Format 2

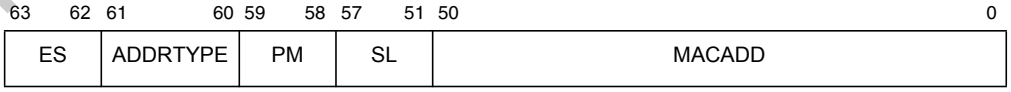


For format 2, the required elements are derived in the following manner:

1. MACADD<54:0> = MACADD field shown above
   MACADD<63:55> = SLI_PKT_INPUT_CONTROL[PBP_DHI<12:4>]
2. endian swap = Use SLI_PKT_INPUT_CONTROL[ESR] with First Indirect DPTRs; use SLI_PKT_INPUT_CONTROL[D_ESR] with First Direct DPTRs in Table 12–1.
3. ADDRTYPE<1> = Use SLI_PKT_INPUT_CONTROL[NSR] with First Indirect DPTRs; use SLI_PKT_INPUT_CONTROL[D_NSR] with First Direct DPTRs in Table 12–1.
4. ADDRTYPE<0> = Use SLI_PKT_INPUT_CONTROL[ROR] with First Indirect DPTRs; use SLI_PKT_INPUT_CONTROL[D_ROR] with First Direct DPTRs in Table 12–1.
5. parse mode = PM field shown above
6. skip length = SL field shown above

- Direct data mode: the DPTR in the DPI instruction is in DPTR format 2 when both of the following conditions are met:
  - SLI_PKT_INPUT_CONTROL[USE_CSR] = 1.
  - SLI_PKT*n*_INSTR_HEADER[PBP] = 1 for the ring that the packet arrived on.
- Indirect gather mode:
  - the DPTR in the DPI instruction is in DPTR format 2 when SLI_PKT*n*_INSTR_HEADER[PBP] = 1 for the ring that the packet arrived on.
  - the DPTRs in the DPI components that represent the gather list for the instruction are never in DPTR format 2.
- Direct gather mode:
  - the DPTR in the DPI instruction is in DPTR format 2 when both of the following conditions are met:
    - SLI_PKT_INPUT_CONTROL[USE_CSR] = 1.
    - SLI_PKT*n*_INSTR_HEADER[PBP] = 1 for the ring that the packet arrived on.
  - DPTR 1, DPTR 2, and DPTR 3 are never DPTR format 2.

DPTR Format 3



For format 3, the required elements are derived in the following manner:

1. MACADD<50:0> = MACADD field shown above
   MACADD<59:51) = SLI_PKT_INPUT_CONTROL[PBP_DHI<8:0>]
   MACADD<60> = SLI_PKT_INPUT_CONTROL[D_ROR]
   MACADD<61> = SLI_PKT_INPUT_CONTROL[D_NSR]
   MACADD<63:62> = SLI_PKT_INPUT_CONTROL[D_ESR]
2. endian swap = ES field shown above
3. ADDRTYPE<1:0> = ADDRTYPE field shown above

Cavium Confidential For Bowen Yang ALU 02/17/2014

4. relaxed order = RO field shown above
5. parse mode = PM field shown above
6. skip length = SL field shown above

- Direct data mode: the DPTR in the DPI instruction is in DPTR format 3 when both of the following conditions are met:
  - SLI_PKT_INPUT_CONTROL[USE_CSR] = 0.
  - SLI_PKT*n*_INSTR_HEADER[PBP] = 1 for the ring that the packet arrived on.
- Indirect gather mode:
  - the DPTR in the DPI instruction is never DPTR format 3.
  - the DPTRs in the DPI components that represent the gather list for the instruction are never in DPTR format 3.
- Direct gather mode:
  - the DPTR in the DPI instruction is in DPTR format 3 when both of the following conditions are met:
    - SLI_PKT_INPUT_CONTROL[USE_CSR] = 0.
    - SLI_PKT*n*_INSTR_HEADER[PBP] = 1 for the ring that the packet arrived on.
  - DPTR 1, DPTR 2, and DPTR 3 are never DPTR format 3.

12.1.4 DPI Instruction Input Ring Backpressure

DPI has two separate hardware mechanisms that stop instruction fetch for a ring: per-port backpressure, and per-ring backpressure. DPI stops fetching instructions on a ring whenever either mechanism is enabled and active for the ring.

Section 9.6 describes CN71XX's per-port backpressure and describes how to enable it. Note that SLI_PKT_CTL[PKT_BP] must be set to enable DPI per-port packet input backpressure. Per-port backpressure is the more coarse backpressure mechanism, as it applies equally to all the 8 rings attached to the PIP/IPD input port. Basically, CN71XX hardware increments a counter corresponding to a port and compares it to a threshold for the port. When the counter is too large, PIP/IPD can backpressure the corresponding port to limit the number of input packets. When per-port backpressure is used, software must decrement the counter later, typically after it processes the input packet, to indicate that the hardware can receive more input packets. When per-port backpressure is enabled and active for a DPI input port, DPI stops fetching instructions from all DPI instruction input rings attached to the port.

DPI also implements per-ring backpressure for the DPI rings. Per-ring backpressure is similar to CN71XX's per-port backpressure, except that the hardware backpressures individual DPI instruction input rings rather than entire PIP/IPD ports.

DPI per-ring backpressure hardware increments SLI_PKT*n*_IN_BP[CNT] (by 1 always) whenever PIP/IPD creates a work-queue entry for an instruction that entered via DPI input ring *n*. Software write operations to SLI_PKT*n*_IN_BP[CNT] subtract from the per-ring backpressure counters. (Note that the SLI_PKT*n*_IN_BP[CNT] write operations should all be 32-bit write operations, not 64-bit, so that SLI_PKT*n*_IN_BP[WMARK] need not be written.) When SLI_PKT*n*_IN_BP[CNT] > SLI_PKT*n*_IN_BP[WMARK], DPI stops fetching from input ring *n*.

Cavium Confidential For Bowen Yang ALU

12.1.5 DPI Instruction Input Initialization

NOTE: Prior to any DPI instruction input initialization, the DPI DMA global initialization (refer to Section 12.3.10.1) should be complete.

There are some configuration parameters in SLI_PKT_INPUT_CONTROL that must be initialized prior to enabling any input rings. These configuration parameters are shared by all rings.

The remaining configuration parameters are all different per-ring. Each ring can be initialized/disabled/re-initialized independently.

12.1.5.1 DPI Instruction Input Ring Initialization

DPI instruction input ring *n* should be configured in the following way:

- SLI_PKT_INSTR_SIZE[IS_64b<*n*>]
- SLI_PKT_IN_PCIE_PORT[PP<2*n*+1:2*n*>]
- SLI_PKT_INSTR_RD_SIZE[RDSIZE<2*n*+1:2*n*>]
- SLI_PKT*n*_INSTR_BADDR[ADDR]
- SLI_PKT*n*_INSTR_FIFO_RSIZE[RSIZE]
- SLI_PKT*n*_INSTR_HEADER[*]
- SLI_PKT*n*_IN_BP[WMARK] (use 4-byte write here)

Before software issues any doorbell write operations, it should sample the current SLI_PKT_IN_DONE*n*_CNTS[CNT] for the ring once. This sampled value should be subtracted (32-bit unsigned) from the value any time that SLI_PKT_IN_DONE*n*_CNTS[CNT] is subsequently read. Then software can correctly determine the number of instructions that hardware has read in for the ring.

Finally, the ring should be enabled via setting SLI_PKT_INSTR_ENB[ENB<*n*>]=1.

12.1.5.2 DPI Instruction Input Ring Disable

A DPI instruction input ring should be disabled before any of the configuration parameters of the ring can be changed.

The procedure to disable ring *n* is as follows:

1. Clear SLI_PKT_INSTR_ENB[ENB<*n*>]=0.
2. Wait until SLI_PKT_PORT_IN_RST[IN_RST<*n*>]=1.
3. Clear SLI_PKTn_INSTR_BAOFF_DBELL[DBELL,AOFF] by writing SLI_PKT-*n*_INSTR_BAOFF_DBELL=0xFFFFFFFF. Read SLI_PKT*n*_INSTR_BAOFF_D-BELL back and wait for the result to return before continuing.
4. Read SLI_PKT_IN_INSTR_COUNTS[RD_CNT,WR_CNT].
   Set low = SLI_PKT_IN_INSTR_COUNTS[WR_CNT].
   Set high = SLI_PKT_IN_INSTR_COUNTS[RD_CNT].
5. Read *x* = SLI_PKT_IN_INSTR_COUNTS[WR_CNT]. Continue reading as long as low ≤ *x* < high (taking into account the 32-bit wrap-around).

It is possible to remain stuck in step 5 when PIP/IPD cannot accept packets due to lack of any packet and/or work queue entry buffers.

After this sequence, the ring can be freely re-initialized.

Cavium Confidential For Bowen Yang ALU 12/7/2014

12.2 DPI Packet Output From CN71XX

DPI has 32 PCIe packet output rings that are often used when CN71XX is a PCIe endpoint, when there is a remote host. DPI packet output rings can also be used when CN71XX is a PCIe host, but the remainder of this section assumes that there is a remote host.

There are four PKO-external PCIe output ports in PKO (refer to the Chapter 10 overview and Section 10.1), numbered 32–35, that supply packets to the DPI packet output logic. Internally, these ports appear much like other ports that exit packet interfaces, until the packets reach DPI. DPI packet output logic places packets from PKO-external DPI ports into one of the DPI packet output rings attached to each PKO-external DPI port. There are eight rings attached to each PKO-external DPI port—a total of 32 DPI output rings.

CN71XX DPI packet output logic maps packets arriving on PKO-external ports to DPI packet output rings differently depending on the mode.

When SLI_PKT_CTL[RING_EN]=0, one ring is attached to each PKO-external DPI port. Only four of the rings (0–3) may be used and there is a one-to-one mapping between PKO-external ports and DPI packet output rings:

ring# = (port# – 32)

When SLI_PKT_CTL[RING_EN]=1, 8 rings are attached to each PKO-external DPI port. All 32 rings may be used, with different rings selected by the relative position of the PKO output queue in the PKO-internal port:

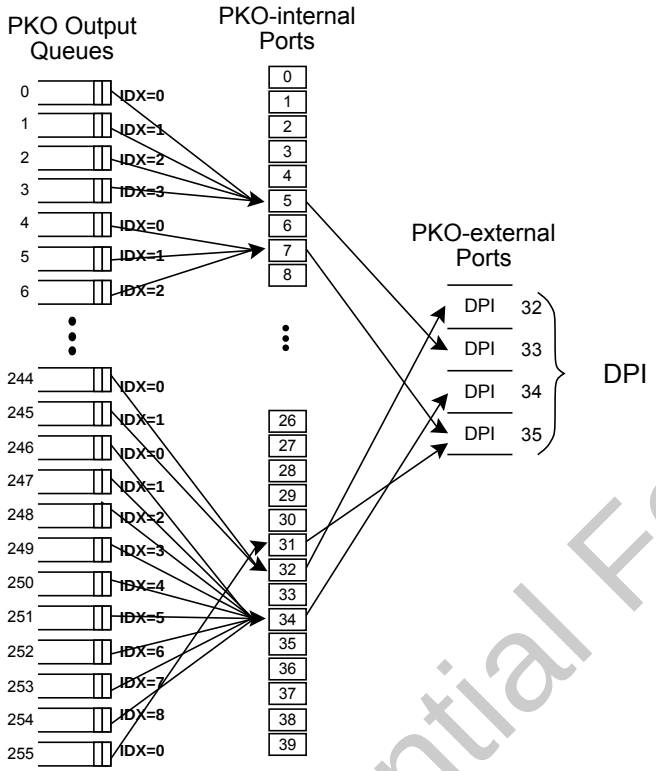
ring# = (IDX modulo 8) × 4 + (port# – 32)

where IDX = the programmed PKO_MEM_QUEUE_PTRS/ PKO_REG_QUEUE_PTRS1[IDX*] value (a 4-bit value as a whole) for the PKO output queue, which indicates the relative position for the PKO output queue in the range of PKO output queues that are attached to the PKO-internal port through which the packet arrived at a PKO-external DPI port.

Many mappings of PKO output queues to DPI rings are possible. Refer to Chapter 10 for details. Figure 12–5 shows one example mapping.

Cavium Confidential For BoWen Yang ALD 02/27/2014

Figure 12–5 PKO Output Queues to DPI Rings Example Mapping



The PKO output queues would map to DPI output rings as shown in Table 12–2 with this example mapping:

Table 12–2 PKO Output Queues to DPI Rings Example Mapping

PKO Output Queue	Corresponding DPI Packet Output Ring	
	SLI_PKT_CTL[RING_EN]=0	SLI_PKT_CTL[RING_EN]=1
0	1	1
1	1	5
2	1	9
3	1	13
4	3	3
5	3	7
6	3	11
244	0	0
245	0	4
246	2	2
247	2	6
248	2	10
249	2	14
250	2	18
251	2	22
252	2	26
253	2	30

Cavium Confidential For Bowen Yang AL D 02/27/2014

Table 12–2 PKO Output Queues to DPI Rings Example Mapping (Continued)

PKO Output Queue	Corresponding DPI Packet Output Ring	
	SLI_PKT_CTL[RING_EN]=0	SLI_PKT_CTL[RING_EN]=1
254	2	2
255	3	3

DPI packet output ring hardware implements the mechanisms to transfer the packets sent to the ring to the remote host for further processing. The (remote) host software receives packets by supplying buffer and info pointers for the ring. The host software supplies buffer and info pointers to the packet output hardware via an input ring. The ring head is managed by host software: host software rings the doorbell to notify DPI of an advance.

DPI maintains the following information for each of the 32 output rings:

- INPUT RING BASE ADDRESS, INPUT RING SIZE (SLI_PKT*n*_SLIST_BADDR[ADDR], SLI_PKT*n*_SLIST_FIFO_RSIZE[RSIZE]). Describes the input ring that supplies buffer/information pairs to DPI. The input ring must be naturally aligned to 16 bytes in MAC memory space.
- PHYSICAL MAC PORT. A physical MAC port number that the DPI output ring is attached to (SLI_PKT_PCIE_PORT). DPI directs both buffer/info pointer pair fetches and packet data write operations for a ring to the selected physical MAC port. Different rings can reside on different physical MAC ports, but all accesses by a single ring use the same one.
- TAIL POINTER. The hardware uses this to fetch buffer/info pairs from the input ring. SLI_PKT*n*_SLIST_BAOFF_DBELL[AOFF] approximates the tail pointer for the ring.
- DOORBELL (SLI_PKT*n*_SLIST_BAOFF_DBELL[DBELL]). Write operations to the doorbell add to the doorbell count. The doorbell count is the number of buffer/info pairs that is available for the hardware to fetch from the input ring.
- PACKET/BYTE COUNT (SLI_PKT*n*_CNTS[CNT]). Indicates either the number of unacknowledged packets or the number of unacknowledged bytes. DPI increments the unacknowledged count once all buffer pointer and info pointer write operations corresponding to a packet are complete. SLI_PKT_OUT_BMODE determines whether the increment is a one or the number of bytes in the packet. The remote MAC host processor decrements this count (by writing it) to indicate receipt of packets/bytes.
- BUFFER SIZE (SLI_PKT*n*_OUT_SIZE[BSIZE]). Indicates the number of bytes available at each buffer pointer.
- INFO BYTES (SLI_PKT*n*_OUT_SIZE[ISIZE]). Indicates the number of bytes from the front of the packet that should appear at the info pointer. The remaining bytes in the packet appear at the data pointer. (In buffer-pointer-only mode (see Section 12.2.2), the INFO BYTES information is not needed nor used.)

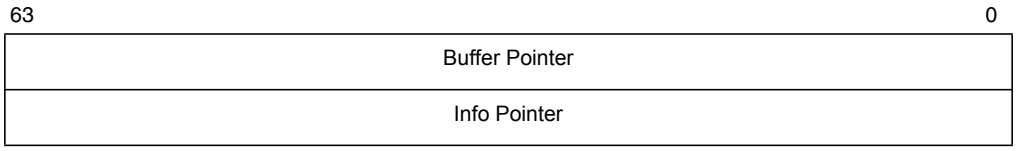
When a packet arrives at one of the 32 DPI output rings, DPI must first have been given a buffer/info pointer pair to write the packet into. The remote host supplies buffer/info pointer pairs via write operations to the doorbell for a ring. Once given a buffer/info pair, DPI writes the packet into the supplied buffer pointer.

Individual DPI packet output rings are enabled by SLI_PKT_OUT_ENB. The DPI output rings can be disabled and reenabled. Section 12.2.3 describes how to initialize and enable/disable the rings.

Cavium Confidential For Bowen Yang 02/12/2014

Figure 12–6 shows the format of a buffer/info pointer pair (in the host processors memory):

Figure 12–6 Buffer/Info Pointer Pair



Buffer pointers may point to any byte, but all info pointers must point to an aligned, 8-byte quantity (i.e. the lower 3-bits must be zero). The format of the 64-bit buffer and info pointers depends on configuration.

- When SLI_PKT_DPADDR[DPTR<*n*>]=1 for ring *n*, the formats of the buffer and info pointers are identical to DPTR format 0, as shown in Section 12.1, except that SLI_PKT_DATA_OUT_ES[ES<2*n*+1:2*n*>], SLI_PKT_DATA_OUT_NS[NSR<*n*>], and SLI_PKT_DATA_OUT_ROR[ROR<*n*>] are used as the effective endian-swap, no-snoop, and relaxed ordering modes rather than SLI_PKT_INPUT_CONTROL[D_ESR,D_NSR,D_ROR], respectively.
- When SLI_PKT_DPADDR[DPTR<*n*>]=0 for ring *n*, the formats of the buffer and info pointers is identical to DPTR format 1, as shown in Section 12.1, except that SLI_PKT_DATA_OUT_ES[ES<2*n*+1:2*n*>], SLI_PKT_DATA_OUT_NS[NSR<*n*>], and SLI_PKT_DATA_OUT_ROR[ROR<*n*>] are the upper bits <63:60> of the effective PCIe address rather than SLI_PKT_INPUT_CONTROL[D_ESR,D_NSR,D_ROR], respectively.

PKO handles all the arbitration amongst the different packets/queues/ports that wish to send out DPI packet output using its standard techniques that are described in Sections 10.5, 10.6, 10.7 and 10.9. PKO includes mixtures of fixed-priority and weighted round-robin at multiple levels with rate-limiting. The DPI packet output processing logic writes the packets into the output rings in the order selected by PKO.

When a ring is disabled or when the doorbell count for a ring falls below SLI_PKT_OUTPUT_WMARK[WMARK], DPI packet output ring logic backpressures for the ring. This means that PKO will stop sending packets from PKO output queues whose packets will arrive at the ring. Sections 10.5 and 10.7 describe PKO's usage for the individual ring backpressures. After PKO chooses to send a packet to any DPI output ring, there is effectively a FIFO of packets to be sent out the MAC interfaces. PKO inserts packets at the head of this effective FIFO. The DPI packet output ring logic extracts packets from the tail of this effective FIFO and sends them out. All DPI packet output rings share one single effective FIFO. With the smallest possible packets (8 bytes or smaller), this FIFO of packets might consume up to 38 buffers. With the more common smallest packet of 64 bytes, the FIFO should consume at most 14 buffers. If there are not enough buffer/info pointer pairs available in the ring of the packet at the tail of the FIFO, that packet and all subsequent packets in the FIFO will stall. If this happens, packet flow to all DPI output rings will stop until buffer/info pointers become available in the one ring. As this is likely to be undesirable for most applications, we recommend that SLI_PKT_OUTPUT_WMARK [WMARK] be set to eliminate this possibility, so that a packet can always be sent to any ring that has sufficient buffer/info pointer pairs. A SLI_PKT_OUTPUT_WMARK [WMARK] value of 14 or less should work well for most applications.

Cavium Confidential For Bowen Yang 06/2/2014

DPI can generate interrupts for a remote host in two ways, both based on the PACKET/BYTE COUNT.

1. THRESHOLD COMPARE: whenever the packet/byte count for a ring (SLI_PKT*n*_CNTS[CNT]) exceeds the programmed threshold for the ring (SLI_PKT_INT_LEVELS[CNT]), SLI_PKT_CNT_INT[PORT<*n*>] is set. If SLI_PKT_CNT_INT_ENB[PORT<*n*>] is set, SLI_INT_SUM[PCNT] will be set when SLI_PKT_CNT_INT[PORT<*n*>] is.

2. TIME THRESHOLD INTERRUPT: whenever the packet/byte count for a ring (SLI_PKT*n*_CNTS[CNT]) is non-zero, DPI increments the time counter for a ring (SLI_PKT*n*_CNTS[TIMER]) once every 1024 coprocessor clock cycles. When this time counter exceeds its threshold (SLI_PKT_INT_LEVELS[TIME]), DPI sets SLI_PKT_TIME_INT[PORT<*n*>]. If SLI_PKT_TIME_INT_ENB[PORT<*n*>] is set, SLI_INT_SUM[PTIME] will be set when SLI_PKT_TIME_INT[PORT<*n*>] is. DPI clears the time counter for a ring when SLI_PKT*n*_CNTS[CNT]=0 or when software clears SLI_PKT_TIME_INT[PORT<*n*>].

There are two packet-output modes (info-pointer mode and buffer-pointer-only mode), selected by SLI_PKT_IPTR[IPTR], and described in Sections 12.2.1 and 12.2.2. Both modes use the identical buffer/info pair format. However, when SLI_PKT_IPTR[IPTR<*n*>]=0 for a ring (Section 12.2.2 describes this case), the Info Pointer field is not used, though it is still read into CN71XX (via the MAC), as it is when SLI_PKT_IPTR[IPTR<*n*>]=1 for a ring (Section 12.2.1 describes this case).

12.2.1 DPI Packet-Output Info-Pointer Mode

Info pointer mode is enabled for ring *n* when SLI_PKT_IPTR[IPTR<*n*>] is set. DPI writes the information to the info pointer after writing the remaining packet data to the buffer pointer in info-pointer mode, so the MAC host can poll in its memory for packet arrival. (The packet is guaranteed to be available after DPI writes the Packet Length field, so the host can look for changes in the memory at the next Packet Length location.) The MAC host can also determine packet arrivals in info-pointer mode by polling the packet/byte count information held in DPI, or by receiving an interrupt that is the result of packet/byte-count changes.

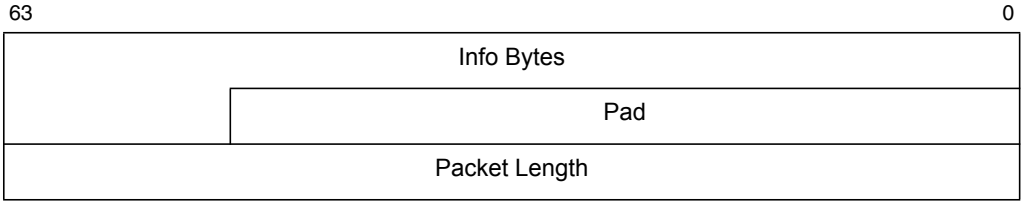
DPI writes the packet into the host's memory as follows in info-pointer mode. In the typical case when the packet (minus Info Bytes) is less than the buffer size for the ring, DPI writes the packet, excluding the first Info Bytes bytes of the packet, to the Buffer Pointer location. After this, it writes the first Info Bytes bytes from the packet together with the Packet Length field to the associated info pointer.

If the packet (minus Info Bytes) is larger than the buffer size for the ring, DPI writes the first buffer-size bytes of the packet, excluding the first Info Bytes bytes of the packet, at the first buffer pointer. It continues allocating and writing further buffer-size byte chunks from the packet into subsequent buffer pointers. After DPI writes the entire packet (excluding the first Info Bytes bytes) to the buffer pointers, it writes the first Info Bytes bytes from the packet together with the Packet Length field to the info pointer associated with the first buffer pointer used to store the packet. Note that when the packet spans across multiple Buffer Pointers in this way, the info pointers other than the first are not used by DPI.

Figure 12–7 shows the format written to the Info Pointer:

Cavium Confidential For Edward Yang AU 02/21/2014

Figure 12–7 Format Written to the Info Pointer



As the info pointer must be 64-bit aligned, the Info Bytes are first written aligned at the info pointer. DPI adds Pad so that the Packet Length field, like the Info Bytes bytes from the start of the packet, is also aligned on an 8-byte boundary. The value of the Pad bytes is not defined.

The (64-bit) Packet Length field is the total number of bytes in the packet.

MAC host software can calculate the actual number of buffers DPI used for a packet in info-pointer only mode:

```
# bufs = ((Packet Length
            - SLI_PKT_SLIST_ID_SIZE[ISIZE]     // subtract info bytes
            + SLI_PKT_SLIST_ID_SIZE[BSIZE] - 1) // add BUFFER SIZE - 1
            / SLI_PKT_SLIST_ID_SIZE[BSIZE])     // divide by BUFFER SIZE
```

In info-pointer mode, the remote PCI host can learn that a packet arrived in any one of the following ways:

- Receipt of an interrupt (SLI_PKT_CNT_INT[PORT<*n*>] or SLI_PKT_TIME_INT[PORT<*n*>] set) due to a packet/byte count (SLI_PKT*n*_CNTS[CNT]) change
- Polling packet/byte count (SLI_PKT*n*_CNTS[CNT]) directly
- Polling for changes in the Packet Length field pointed at by an info pointer.

It is illegal to send packets with fewer than Info Bytes in info-pointer mode.

12.2.2 DPI Packet-Output Buffer-Pointer-Only Mode

Buffer-pointer-only mode is enabled for ring *n* when SLI_PKT_IPTR[IPTR<*n*>]=0. In this mode, DPI ignores the value of the info-pointer field in the buffer/info pair format. Only the buffer pointer is used.

In buffer-pointer-only mode, DPI first writes the eight-byte Packet-Length field (the same format as shown at the bottom of Figure 12–7) at the Buffer Pointer start location, then follows this with the remaining packet data. The Packet-Length field acts exactly as eight bytes of additional packet data preceding the real packet data.

As with info-pointer mode, the PCIe hardware allocates and writes multiple buffer pointers when necessary in buffer-pointer-only mode. The hardware accounts for the Packet-Length field, and buffer size always limits the number of bytes written in each supplied buffer pointer.

Cavium Confidential For Bowen Yang ALU 02/27/2014

The (64-bit) Packet Length field is the total number of bytes in the packet. MAC host software can calculate the actual number of buffers DPI used for a packet in buffer-pointer mode:

```
# bufs = ((Packet Length
        + 8                                  // for Packet Length field
        + SLI_PKT_SLIST_ID_SIZE[BSIZE] - 1)  // add BUFFER SIZE - 1
        / SLI_PKT_SLIST_ID_SIZE[BSIZE])      // divide by BUFFER SIZE
```

In buffer-pointer-only mode, the remote MAC host can learn that a packet arrived in either of the following ways:

- Receipt of an interrupt (SLI_PKT_CNT_INT[PORT<*n*>] or SLI_PKT_TIME_INT[PORT<*n*>] set) due to a packet/byte count (SLI_PKT*n*_CNTS[CNT]) change
- Polling packet/byte count (SLI_PKT*n*_CNTS[CNT]) directly

NOTE: A remote MAC host cannot learn of packet arrival by polling the Packet Length field in buffer-pointer-only mode, because the hardware writes the Packet Length field before writing the remainder of the packet.

12.2.3 DPI Packet Output Initialization

NOTE: Prior to any DPI packet-output initialization, the DPI DMA global initialization (refer to Section 12.3.10.1) should be complete.

The SLI_PKT_OUTPUT_WMARK[WMARK] threshold (refer to Section 12.2) is shared by all DPI packet output rings. They should be initialized before any output rings are initialized.

SLI_PKT_CTL[RING_EN] must be set to enable use of any of packet output rings 4–31. If none of output rings 4–31 will be used, SLI_PKT_CTL[RING_EN] should be cleared to zero.

There are also the SLI_PKT_INT_LEVELS[CNT,TIME] interrupt thresholds that are shared amongst all rings, and will typically be initialized before any output rings are initialized.

The remaining configuration parameters are all different per-ring. Each ring can be initialized/disabled/re-initialized independently.

12.2.3.1 DPI Packet Output Ring Initialization

DPI Packet Output Ring *n* should be configured:

- SLI_PKT_PCIE_PORT[PP<2*n*+1:2*n*>]
- SLI_PKT_IPTR[IPTR<*n*>]
- SLI_PKT_OUT_BMODE[BMODE<*n*>]
- SLI_PKT*n*_SLIST_BADDR[ADDR]
- SLI_PKT*n*_OUT_SIZE[BSIZE,ISIZE]
- SLI_PKT*n*_SLIST_FIFO_RSIZE[RSIZE]
- SLI_PKT_DPADDR[DPTR<*n*>]
- SLI_PKT_SLIST_ROR[ROR<*n*>]

Cavium Confidential For Bowen Yang ALU 02/27/2014

- SLI_PKT_SLIST_NS[NSR<*n*>]
- SLI_PKT_SLIST_ES[ES<2*n*+1:2*n*>]
- SLI_PKT_DATA_OUT_ROR[ROR<*n*>]
- SLI_PKT_DATA_OUT_NS[NSR<*n*>]
- SLI_PKT_DATA_OUT_ES[ES<2*n*+1:2*n*>]
- SLI_PKT_CNT_INT_ENB[PORT<*n*>]
- SLI_PKT_TIME_INT_ENB[PORT<*n*>]

Finally, the ring should be enabled via setting SLI_PKT_OUT_ENB[ENB<*n*>]=1.

At this point, software can provide data/info pairs to the ring.

12.2.3.2 DPI Packet Output Ring Disable

A DPI Packet Output Ring should be disabled before any of the configuration parameters of the ring can be changed.

The procedure to disable ring *n* is as follows:

1. Clear SLI_PKT_OUT_ENB[ENB<*n*>]=0.
2. Wait until SLI_PKT_PORT_IN_RST[OUT_RST<*n*>] = 1.
3. Clear SLI_PKT*n*_SLIST_BAOFF_DBELL[DBELL,AOFF] by writing SLI_PKT-*n*_SLIST_BAOFF_DBELL = 0xFFFFFFFF. Read SLI_PKT*n*_SLIST_BAOFF_D-BELL back and wait for the result to return before continuing.
4. Read SLI_PKT*n*_CNTS[CNT] and write back the same value to zero it out.
5. If necessary, clear out the SLI_PKT_CNT_INT[PORT<*n*>] and SLI_PKT_-TIME_INT[PORT<*n*>] interrupt bits associated with the ring.

After this sequence, the ring can be freely re-initialized.

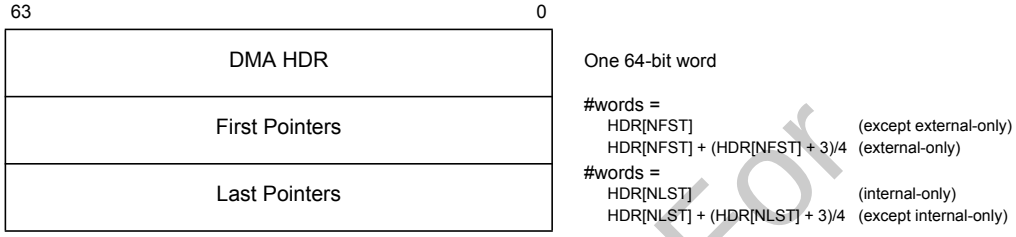
Cavium Confidential For Bowen Yang ALU 02/27/2014

12.3 DPI DMA Queues and Engines

The cnMIPS II cores can move data by submitting DPI DMA instructions to the DPI DMA queues. There are a total of eight instruction queues backed up by six DMA engines that perform the necessary MAC memory-space and L2C/DRAM operations to move data asynchronously to core execution. The engines can notify the cores of completion via either L2C/DRAM polling or work-queue-entry submittal.

A DPI DMA instruction can be from a minimum of three to a maximum of 32 64-bit words. Figure 12–8 shows the format of a DMA instruction.

Figure 12–8 DPI DMA Instruction Format



The DMA instruction has three major components: DMA HDR, first pointers, and last pointers.

- The DMA instruction header (DMA HDR) defines the type of the transfer and contains control information for the transfer. Section 12.3.1 describes DMA HDR in detail.
- The first pointers are the first block of pointers; the last pointers are the second block of pointers. Both have different meanings depending on which of four transfer types is specified in the DMA HDR.

Table 12–3 defines the four transfer types: outbound, inbound, internal-only, and external-only, and illustrates how the first pointers and last pointers are used for the different transfer types.

Table 12–3 DPI DMA Transfer Types

Transfer Type	Definition	First Pointers Use	Last Pointers Use
outbound	Data moves from L2/DRAM to MAC memory space	Local pointers (L2/DRAM) to read from	DPI components with pointers to write to
inbound	Data moves from MAC memory space to L2/DRAM	Local pointers (L2/DRAM) to write to	DPI components with pointers to read from
internal-only	Data moves from L2/DRAM to L2/DRAM	Local pointers (L2/DRAM) to read from	Local pointers (L2/DRAM) to write to
external-only	Data moves from MAC memory space to MAC memory space	DPI components with pointers to read from	DPI components with pointers to write to

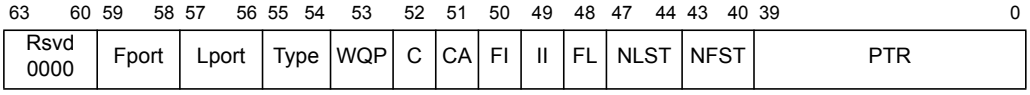
Note that a single DMA instruction may be either contiguous in memory or split into two pieces, depending on how it resides within the DMA instruction chunk for the DMA engine, which is defined in the following subsections.

12.3.1 DPI DMA Instruction-Header Format

The DMA instruction header (DMA HDR) describes the DPI DMA instruction. Figure 12–9 shows the header format.

Cavium Confidential For Bowen Yang ALU 05/7/2014

Figure 12–9 DMA Instruction-Header Format



- **Fport** – First port. FPort indicates the physical MAC port used for the MAC memory space pointers in the first-pointers block in the external-only case. Must be zero in the outbound, inbound, and internal-only cases.
- **Lport** – Last port. LPort indicates the physical MAC port used for the MAC memory space pointers in the last-pointers block in the outbound, inbound, and external-only cases. Must be zero in the internal-only case.
- **Type** – A given DMA transfer is either outbound (read operation from L2/DRAM, write operation into MAC memory space), inbound (read operation from MAC memory space, write operation into L2/DRAM), internal-only (read operation from L2/DRAM, write operation into L2/DRAM), or external-only (read operation from MAC memory space, write operation into MAC memory space). Refer to Table 12–3.
  - outbound = 0x0
  - inbound = 0x1
  - internal-only = 0x2
  - external-only = 0x3
- **WQP** – Work-queue pointer.
  - when WQP = 1, PTR (if non-zero) is a pointer to a work-queue entry that DPI submits after completing the DMA operation.
  - when WQP = 0, PTR (if non-zero) is a pointer to a byte in local memory that DPI writes to 0 after completing the DMA operation.
- **C** – Counter. 1 = use counter 1, 0 = use counter 0.

  The C bit selects between the two counters (SLI_DMA0/1_CNT[CNT]) that DPI can optionally update after an outbound or external-only transfer, and also selects between the two forced-interrupt bits (SLI_INT_SUM[DMAFI]) that DPI can optionally set after an outbound or external-only transfer. C must be zero for inbound or internal-only transfers.
- **CA** – Counter add.

  When CA = 1, DPI updates the selected counter after it completes the DMA outbound or external-only instruction.
  - If C = 0, DPI updates SLI_DMA0_CNT[CNT]
  - If C = 1, DPI updates SLI_DMA1_CNT[CNT]

  Note that this update may indirectly cause SLI_INT_SUM[DCNT,DTIME] to become set (depending on the SLI_DMA0/1_INT_LEVEL settings), so may cause interrupts to occur on a remote MAC host.
  - If DPI_DMA_CONTROL[O_ADD1] = 1, DPI updates the counter by 1.
  - If DPI_DMA_CONTROL[O_ADD1] = 0, DPI updates the counter by the total bytes in the transfer.

  When CA = 0, DPI does not update any counters.

  For an inbound or internal-only DMA transfer, CA must never be set, and DPI never adds to the counters.

Cavium Confidential For Bowen Yang ALI 02/27/2014

- **FI** – Force interrupt.

  When FI is set for an outbound or external-only transfer, DPI sets a forced interrupt bit after it completes the DMA instruction. If C = 0, DPI sets SLI_INT_SUM[DMAFI<0>], else sets SLI_INT_SUM[DMAFI<1>]. For an inbound or internal-only DMA operation, FI must never be set, and DPI never generates interrupts.

- **II**– Ignore the I bit (i.e. the I bit of the DPI DMA instruction local pointer. See Section 12.3.2).

  For outbound transfers when II = 1, ignore the I bit and the FL bit in the DMA HDR alone determines whether the hardware frees any/all of the local buffers in the first-pointers area:

  - when FL = 1, DPI frees the local buffer when II = 1.
  - when FL = 0, DPI does not free the local buffer when II = 1.

  For outbound transfers when II = 0, the I bit in the local pointer selects whether local buffers are freed on a pointer-by-pointer basis:

  - when (FL ⊕ I) is true, DPI frees the local buffer when II = 0.

  For inbound, internal-only, and external-only DMA transfers, II must never be set, and DPI never frees local buffers.

- **FL** – Free local buffer.

  When FL = 1, for an outbound operation, it indicates that the local buffers in the FIRST BUFFERS area should be freed.

  If II = 1, the FL bit alone indicates whether the local buffer should be freed:

  - when FL = 1, DPI frees the local buffer when II = 1.
  - when FL = 0, DPI does not free the local buffer when II = 1.

  If II = 0, the I bit in the local pointer (refer to Figure 12–14) determines whether the local buffer is freed:

  - when (FL ⊕ I) is true, DPI frees the local buffer when II = 0.

  For an inbound, internal-only, or external-only DMA transfer, FL must never be set, and DPI never frees local buffers.

- **NLST** – Number last pointers.

  The number of pointers in the last-pointers area.

  In the inbound, outbound, and external-only cases, the last-pointers area contains DPI components, and the number of 64-bit words required in the last-pointers area is:

  HDR[NLST] + ((HDR[NLST] + 3)/4)

  where the division removes the fraction.

  In the internal-only case, the last-pointers area contains local pointers, and the number of 64-bit words required in the last-pointers area is:

  HDR[NLST]

  Note that the sum of the number of 64-bit words in the last-pointers and first-pointers area must never exceed 31.

Cavium Confidential For Bowen Yang ALU 03/27/2014

- **NFST** – Number of first pointers.

  The number of pointers in the first-pointers area.

  In the inbound, outbound, and internal-only cases, the first-pointers area contains local pointers, and the number of 64-bit words required in the first-pointers area is:

  HDR[NFST]

  In the external-only case, the first-pointers area contains DPI components, and the number of 64-bit words required in the first-pointers area is:

  HDR[NFST] + ((HDR[NFST] + 3)/4)

  where the division removes the fraction.

  Note that the sum of the number of 64-bit words in the last-pointers and first-pointers area must never exceed 31.

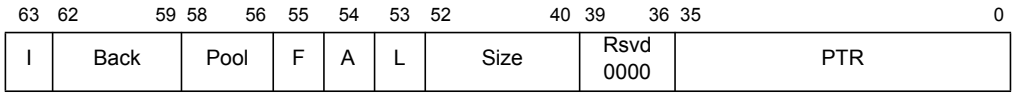
- **PTR** – Pointer, either a work-queue-entry pointer (when WQP = 1) or a local memory pointer (WQP = 0).
  - When WQP = 1 and PTR ≠ 0x0, DPI inserts the work-queue entry indicated by PTR into a POW input queue after the DMA operation is complete. (Section 7.4 describes the work-queue-entry requirements in this case.) When WQP = 1, PTR<2:0> must be 0x0.
  - When WQP = 0, PTR ≠ 0x0, and either PTR ≥ 64 or DPI_DMA_CONTROL[DICI_MODE]=0, DPI writes the single byte in local memory indicated by PTR to 0x0 after the DMA operation is complete. DPI_DMA_CONTROL[B0_LEND] selects the endian-ness of PTR in this case.
  - When WQP = 0, PTR ≠ 0x0, PTR < 64, and DPI_DMA_CONTROL[DICI_MODE]=1, DPI increments the DPI_DMA_PP*n*_CNT[CNT], where the value for core *n* is PTR<5:0>–1.
  - When PTR = 0x0, DPI performs no operation after the DMA operation is complete.

  Section 12.3.6 describes when a DPI DMA instruction completes.

12.3.2 DPI DMA Local-Pointer Format

The DPI DMA local-pointer format, shown in Figure 12–10, describes a memory fragment in the L2/DRAM attached to CN71XX.

Figure 12–10 DPI DMA Local-Pointer Format



Local pointers exist in the first-pointers area for outbound, inbound, and internal-only transfers. HDR[NFST] indicates the number of local pointers in the first-pointers area in these transfers.
Local pointers exist in the last-pointers area for internal-only transfers. HDR[NLST] indicates the number of local pointers in the last-pointers area in these transfers.

Cavium Confidential For Bowen Yang AU 02/27/2014

- **I** – Invert free.

  This bit gives the software the ability to free buffers independently for an outbound DMA transfer. See the II and FL bit descriptions in Section 12.3.1.

  I is not used by DPI when II is set. I must not be set, and buffers are never freed, for inbound, internal-only, and external-only DMA transfers.

- **Back** – Backup amount.

  Allows the start of a buffer that is to be freed during an outbound transfer to be different from the PTR value. Back specifies the amount to subtract from the pointer to reach the start when freeing a buffer.

  The address that is the start of the buffer being freed is:

  $$\text{Buffer start address} = ((\text{PTR} >> 7) - \text{Back}) << 7.$$

  Back is only used by the hardware when the buffer corresponding to PTR is freed. Back must be 0x0, and buffers are never freed, for inbound, internal-only, and external-only DMA transfers.

- **Pool** – Free pool.

  Specifies which pool (of the eight hardware-managed FPA free pools) receives the buffer associated with PTR when freed during an outbound transfer (refer to Chapter 8).

  Pool is only used when the buffer corresponding to PTR is freed. Pool must be 0x0, and buffers are never freed, for inbound, internal-only, and external-only DMA transfers.

- **F** – Full-block write operations are allowed.

  When set, the hardware is permitted to write all the bytes in the cache blocks covered by PTR, PTR + Size – 1. This can improve memory system performance when the write misses in the L2 cache.

  F can only be set for local pointers that can be written to:

  - The local pointers in the first-pointers area that are write pointers for inbound transfers
  - The local pointers in the last-pointers area that are always write pointers (when present for internal-only transfers).

  F must not be set for local pointers that are not written to:

  - The local pointers in the first-pointers area for outbound and internal-only transfers

- **A** – Allocate L2.

  This is a hint to DPI that the cache blocks should be allocated in the L2 cache (if they were not already).

  Should typically be set to allocate the referenced cache blocks into the L2 cache.

  When the local pointer is a source of data (e.g. an outbound transfer), the referenced cache blocks are not allocated into the L2 cache as part of completing the DMA (when not already present in the L2) if the A bit is clear.

Cavium Confidential For Rowen Yang ALU 02/27/2014

When the local pointer is a destination for data (e.g. an inbound transfer), the referenced cache blocks are not allocated into the into the cache as part of completing the DMA (when not already present in the L2) if the A bit is clear, and either:

- the entire cache block is written by this local pointer, or
- the F bit is set so that the entire cache block can be written

- **L** – Little-endian.

  When L is set, the data at PTR is in little-endian format rather than big-endian.

- **Size** – Size in bytes of the contiguous space specified by PTR. A Size value of 0 is illegal.

  Note that the sum of the sizes in the first-pointers area must always exactly equal the sum of the sizes/lengths in the last-pointers area:

  - In the outbound and inbound cases, the HDR[NFST] size fields in the local pointers in the first-pointers area must exactly equal the lengths of the HDR[NLST] fragments in the DPI components in the last-pointers area.
  - In the internal-only case, the HDR[NFST] size fields in the local pointers in the first-pointers area must equal the HDR[NLST] size fields in the local pointers in the last-pointers area.

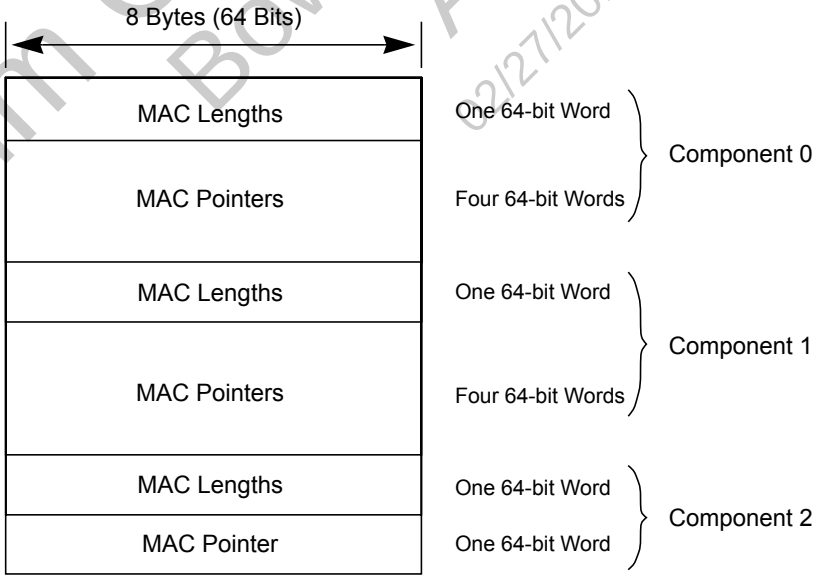
- **PTR** – L2/DRAM byte pointer. Points to where the packet data starts.

  PTR can be any byte alignment. Note that PTR is interpreted as a big-endian byte pointer when L is clear, a little-endian byte pointer when L is set.

12.3.3 DPI Components and Processing

The DPI-components format is a compressed list of 64-bit MAC memory space pointers and their associated lengths. Up to four lengths are encoded in a 64-bit length word. A component is a length word and its associated 64-bit pointers. For example, if HDR[NLST] = 9 for an outbound transfer, the DPI components in the last pointers are of the DMA instruction would consist of three components totaling 12 64-bit words as shown in Figure 12–11.

Figure 12–11 DPI Component Format Example When HDR[NLST] = 9

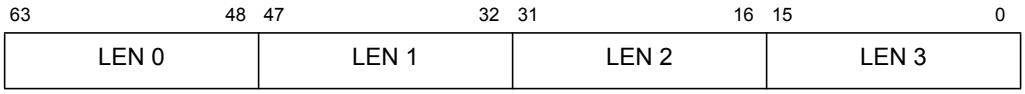


Cavium Confidential For Bowen Yang ALU 02/27/2014

Most components are five words: one for the length and four for the pointers. If HDR[NLST] or HDR[NFST] is not a multiple of 4, the last component is always truncated. The Figure 12–11 example shows a case where three DPI components are necessary to hold the nine MAC memory space pointers. The last component is partial and is only two 64-bit words: one to hold the necessary MAC length field, and one to hold the necessary ninth remote pointer.

The format of the MAC length field is shown in Figure 12–12.

Figure 12–12 MAC Length-Field Format



Together with a corresponding length, the DPI pointer describes a fragment of up to 64KB in MAC memory space.

Note that the sum of the first HDR[NLST] or HDR[NFST] length fields in the DPI components (i.e. the total number of MAC memory space pointer bytes) in the area must exactly equal the total of the sizes/lengths in the other area:

- In the outbound and inbound cases, the sum of the first HDR[NLST] length fields in the DPI components in the last-pointers area must exactly equal the sum of the HDR[NFST] size fields in the local pointers in the first-pointers area.
- In the external-only case, the sum of the first HDR[NLST] length fields in the DPI components in last-pointers area must exactly equal the sum of the HDR[NFST] length fields in the DPI components in first-pointers area.

The DPI DMA engines send the MAC memory space read and write operations out the port selected by HDR.LPort or HDR.FPort to complete the DMA transaction:

- The hardware directs the read/write operations for the MAC memory space pointers in the last-pointers area to the port selected by HDR.LPort for outbound, inbound, and external-only transfers.
- The hardware directs the read operations for the MAC memory space pointers in the first-pointers area to the port selected by HDR.FPort for external-only transfers.

Each MAC pointer is a 64-bit MAC memory-space pointer that takes either DPTR format 0 or DPTR format 1 (as described in Section 12.1.3), except that DPI_DMA_CONTROL[O_ES,O_NS,O_RO] is used instead of SLI_PKT_INPUT_CONTROL[D_ESR,D_NSR,D_ROR] in all places. (SLI_PKT_INPUT_CONTROL[D_ESR,D_NSR,D_ROR] is never used by the DMA engines.)

When DPI_DMA_CONTROL[O_MODE] is set, DPTR format 0 is used with DPI_DMA_CONTROL[O_ES,O_NS,O_RO]. When DPI_DMA_CONTROL[O_MODE] is clear, DPTR format 1 is used with DPI_DMA_CONTROL[O_ES,O_NS,O_RO].

DPI_SLI_PRT(0..2)_CFG[RD_MODE,MRRS,MRRS_LIM] per-MAC port configuration controls how DPI generates the reads for the read MAC pointers in inbound or external-only DMA instructions. DPI_SLI_PRT(0..2)_CFG[RD_MODE] controls what surrounding bytes DPI is allowed to read.

DPI_SLI_PRT(0..2)_CFG[MRRS,MRRS_LIM] controls how DPI breaks large read operations up into smaller ones. DPI_SLI_PRT(0..2)_CFG[MPS,MPS_LIM] per-port configuration controls how DPI breaks up large MAC write pointers from outbound or external-only DMA instructions into smaller write operations.

Cavium Confidential For Bowen Yang ALU 12/27/2014

12.3.4 DPI DMA Instruction Queues

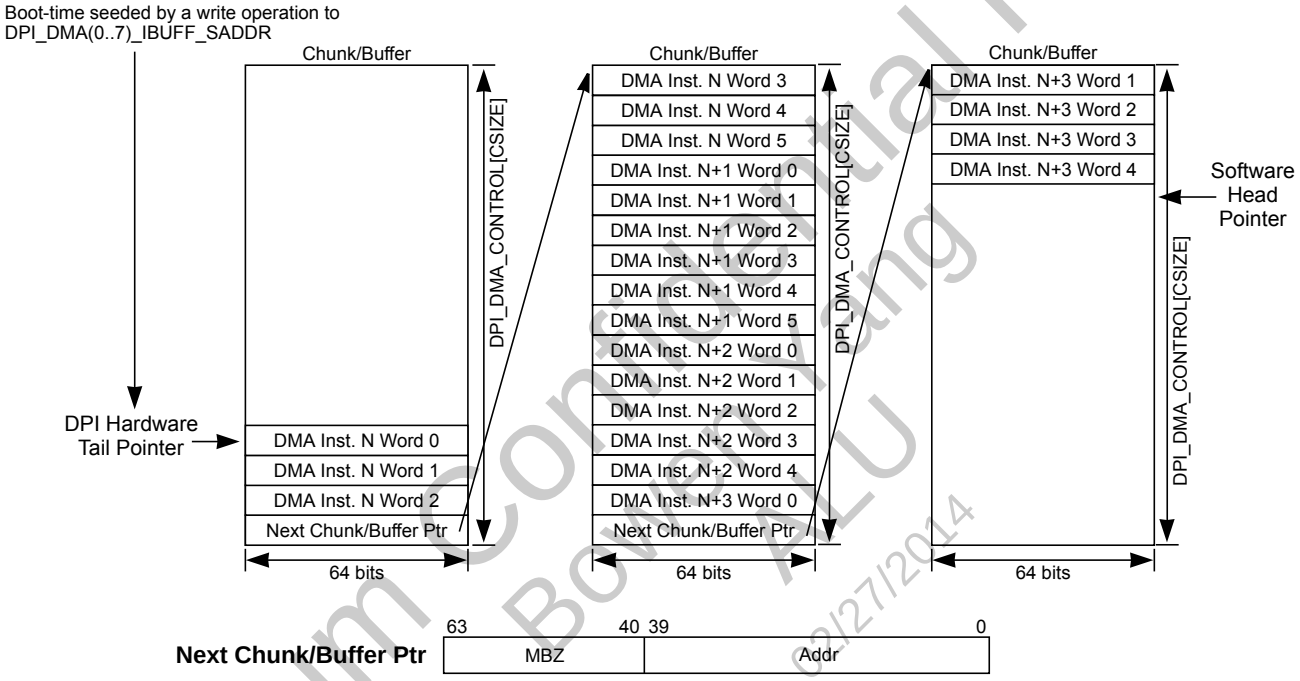
DPI has a total of eight DMA queues, each of which can process any of the four possible transfer types. Software loads the queues with DPI DMA instructions, and DPI can schedule these instructions to one of the six available DMA engines that actually execute the instructions.

Each queue is a linked-list of memory chunks in L2/DRAM memory containing DMA instruction information. Core software adds DMA instructions at the head of a queue and notifies the hardware of the arrival by writing the hardware doorbell register for the corresponding queue (DPI_DMA(0..7)_DBELL[DBELL]), typically with the number of words in the instructions.

DPI accumulates the doorbell write operations. DPI reads an instruction from the tail of a queue and performs the DMA operation. When DPI exhausts the chunk at the tail of the queue (i.e. reads all of the instructions that filled the chunk), it frees the tail chunk and starts reading instructions from the next chunk.

Figure 12–13 is an example of the list.

Figure 12–13 Example: Queue Linked-List Memory Chunks



DPI knows the chunk size through the configuration variable DPI_DMA_CONTROL[CSIZE]. The DPI_DMA_CONTROL[FPA_QUE] field selects the free pool that receives a chunk once the tail advances through the next_ptr of the chunk (refer to Chapter 7).

The software writes, and DPI reads, instructions contiguously in a chunk, starting at the first word. A given DMA instruction is normally contiguous in local DRAM memory, but a single instruction must span two chunks if the contiguous words in the instruction would otherwise over-run the next_ptr of the chunk. In this case, the remainder of the instruction resides at the beginning of the next chunk. The software must allocate the next chunk (and update the next_ptr in the previous chunk) whenever it inserts an instruction that either abuts or over-runs the next chunk buffer pointer. This allows DPI to read the next_ptr, and traverse to the next chunk, with the same fetch that it uses to read the last instruction word of the chunk.

Cavium Confidential For Bowen Zhang ALU 02/27/2014

DPI knows there are DMA instructions available to be fetched from the queue after the software updates the doorbell count (via write operations to DPI_DMA(0..7)_DBELL[DBELL]). The software may launch multiple instructions with a single doorbell write operation. Software can also use multiple doorbell write operations to announce the availability of a single instruction, but this provides poorer performance than a single write operation.

For this and other reasons, DPI does not assume that individual doorbell write operations correlate to entire DMA instructions. DPI only uses the doorbell write operations to indicate valid instruction words in the queue.

The distance between the head pointer and the tail pointer is both the size of the DMA instruction queue and the outstanding doorbell count. The size of the DMA instruction queue is limited only by the available memory.

At boot time, software must configure the DMA instruction queues with the original next chunk buffer pointer (i.e. the starting tail pointer) with a write to DPI_DMA(0..7)_IBUFF_SADDR[SADDR].

Figure 12–13 also shows the next chunk buffer pointer format. The primary component is the Addr field that selects a legal L2/DRAM byte location. Though Addr is a byte address, it must be naturally aligned on a 128-byte cache-block boundary, so its least-significant 7 bits must be zero. The same is true for the original next chunk buffer pointer.

Since the hardware maintains the doorbell counts and tail pointers for the queues, the software normally does not need to know them. Approximations of them are available for debug purposes, however.

- DPI_DMA(0..7)_COUNTS[DBELL] + DPI_DMA(0..7)_COUNTS[FCNT] approximates the doorbell count for the queue.
- DPI_DMA(0..7)_NADDR[ADDR] approximates the tail pointer for a queue.

12.3.5 DMA Queue-to-Engine Mapping

DPI contains eight DMA queues (0–7) for DMA instructions and six DMA engines (0–5) to execute the instructions.

When DPI instruction input (see Section 12.1) or DPI packet output are used (see Section 12.2), DPI_DMA_CONTROL[PKT_EN] is set, and DPI DMA engine 5 is not available for assignment to DMA queues.

Otherwise, software can assign any combination of DMA engines to any combination of DPI queues. DPI_DMA_ENG(0..5)_EN[QEN] specifies the assignment. DPI arbitrarily selects a DMA engine to execute an instruction in a queue when more than one DMA engine is assigned to the DMA queue.

DPI guarantees forward progress of DMA queues. For example, when one DMA engine is the only engine assigned to multiple instruction-ready DMA queues, DPI round-robins the DMA engine across the queues.

12.3.6 DPI DMA Instruction Ordering and Completion

There are up to six DPI DMA engines that run in parallel. If any ordering is required between different simultaneously in-flight DMA instructions, the instructions must be issued to the same DPI instruction queue. It is possible for DPI to schedule two

Cavium Confidential For Bowen Yang ALU 02/21/2014

instructions from the same DPI instruction queue simultaneously to different DPI DMA engines. DPI follows these rules:

- The DMA hardware will not perform any of the following functions until the instruction is complete:
  - add a work-queue entry for an instruction
  - write an L2/DRAM byte to zero
  - increment the counter selected by the C bit
  - set any interrupt bit.
- An inbound or internal-only DMA instruction is complete once all required L2/DRAM stores (excluding any necessary completion zero byte or work-queue-entry write operations) have completed and are visible to all devices. This completion implies that all read operations necessary to execute the instruction are also complete (except for any read operations necessary to submit a work-queue entry).
- An outbound or external-only DMA instruction is complete once all required DPI MAC memory-space write operations are ordered before any subsequently generated read or write operations to the physical DPI MAC port. This completion implies that all read operations necessary to execute the instruction are also complete (excluding any read operations necessary to submit a work-queue entry).
- DPI submits work-queue entries for instructions from the same DPI instruction queue in order.
- DPI increments counters selected by the C bit in instructions in the same DPI instruction queue in order.
- DPI may execute the L2/DRAM store operations needed to complete a single inbound or internal-only DPI DMA instruction (excluding any completion-zero byte or work-queue entry write operation) in any order.
- The L2/DRAM read operations required to complete an outbound or internal-only DMA instruction (excluding any read operations necessary to submit a work-queue entry) will complete after all L2/DRAM write operations to the same address generated from any earlier inbound or internal-only DMA instructions in the same DPI instruction queue (excluding any necessary completion zero-byte or work-queue-entry write operations).
- The L2/DRAM read operations required to complete an outbound or internal-only DMA instruction (excluding any read operations necessary to submit a work-queue entry) will complete before all L2/DRAM write operations to the same address generated from any later inbound or internal-only DPI DMA instructions in the same DPI instruction queue (excluding any necessary completion zero-byte or work-queue-entry write operations).
- All DPI MAC memory space read/write operations initiated for different inbound, outbound, or external-only instructions in the same DPI instruction queue and destined for the same physical DPI MAC port appear on the DPI MAC port in order.
- Within a single inbound, outbound, or external-only DMA instruction, any DPI MAC memory space read operations are ordered as in the instruction, and any DPI MAC memory space write operations are ordered as in the instruction. In an external-only DMA instruction with read and write operations destined to the same physical DPI MAC port, the read and write operations are unordered with respect to each other, except that a write operation cannot occur until DPI reads the data that will be written.

Cavium Confidential For Power Yang ALU 8/7/2014

12.3.7 DPI DMA Engine Don't-Write-Back Calculation

An FPA hardware pool free command includes a don't-write-back (DWB) argument that specifies the maximum number of DWB commands to execute on the coherent memory bus. The DMA engine generates FPA free commands in two different cases:

- To free an instruction chunk
- To free a buffer when directed for an outbound transfer

When freeing an instruction chunk, DPI_DMA_CONTROL[DWB_ICHK] is the DWB count used.

When freeing a buffer as part of an outbound transfer, a DWB count of zero is used when DPI_DMA_CONTROL[DWB_DENB] is clear. If DPI_DMA_CONTROL[DWB_DENB] is set, the following equation specifies the DWB count that is used:

```
(((PTR & 127) + Size + 127) >> 7) + Back
```

which may result in a CMI DWB command for all cache blocks from the start of the buffer up to the block that includes the last byte of the segment. Back, ptr, and Size here come directly from the local pointer that caused the FPA free to be generated.

12.3.8 DPI Host Output Queuing Via the DMA Engine

The DPI DMA engines provide flexible primitives which, when combined with core and host software protocols, can efficiently implement a variety of host output queuing protocols and handle many simultaneous host queues.

We will illustrate with an example. Suppose one million host output queues are required, and each host queue is a simple ring similar to the rings that are managed by the hardware for the four MAC-instruction input ports and four MAC packet-output ports. (This is just one simple example. Many other queue/buffer structures are also possible.) Each ring has a head, tail, and outstanding doorbell count.

Here is how to implement the example one million queues using one of the DPI DMA engines:

- The core software maintains base and head/tail pointers for each of the one million queues. This information resides in memory locations present in either the L2 cache or the DRAM attached to CN71XX. The core software selects the detailed format of the information.
- The host software must issue "doorbell rings" to indicate space availability. For the 32 hardware-managed packet-output rings, a doorbell ring is a direct MAC CSR write operation (to SLI_PKT(0..31)_SLIST_BAOFF_DBELL[DBELL]) by the host. But since the doorbells/heads/tails for the one million queues are maintained by core software in this case, their doorbell ring is instead a special work-queue entry or packet received by the core software from the host. The host software can create/submit the special work-queue entry or packet accomplishing the doorbell ring via MAC instruction input as described in Section 12.1. Core and host software selects the detailed format of the doorbell rings.
- In addition to the one million rings, the core and host software also maintain two additional rings for control. These control rings are tied to SLI_DMA0/1_CNT[CNT] (and indirectly to SLI_INT_SUM[DCNT, DTIME]), as selected by the C and CA bits in the DMA instruction header.

Cavium Confidential For Bowen Jiang 2/2/2014

As with the one million rings, the base and head/tail pointers for each of the two control queues is maintained in CN71XX memory, and “doorbell rings” for them would arrive via MAC instruction input.

- When the core software wishes to send a packet to one of the one million rings, it writes the packet data into the appropriate host memory ring and also writes a control packet to one of the two available control rings. This control packet includes ring-identifier and packet-size fields and is the indication to host software that the packet data arrived at the identified ring.

  Both the packet and the control packet can be transferred via a single outbound DPI DMA instruction.

To construct the DPI DMA instruction, the core software first creates DMA local pointers and DMA host pointers (DPI components) to transfer the packet data from L2/DRAM into the host memory. The core software then appends additional DMA local pointers and DMA host pointers (DPI components) to the lists to transfer the control packet into the host memory after the packet data.

The core software consults the base and head/tail pointers that it maintains in L2/DRAM to determine the locations where both the packet data and the control-packet data should be placed in the host memory. These addresses are supplied in the DMA instruction DPI components of the outbound transfer.

The core software can set the CA bit in the DMA instruction HDR to indicate the arrival of the control packet. (Of course, it must also set the C bit correctly to select which of the two control queues was used.)

12.3.9 DPI DMA Engine Performance

DPI has six DMA engines, and each engine has an associated FIFO, allowing it to read ahead up to the size of its FIFO. There is a total of 8KB of storage for all the engines. Software specifies the size of each engine’s FIFO through DPI_ENG(0..5)_BUF[BLKS]. The FIFOs can range from 512 bytes up to 8KB in 512-byte increments, and the engine-read performance may be higher with a larger FIFO.

When DPI_DMA_CONTROL[PKT_EN]=1 and DPI_DMA_CONTROL[PKT_EN1]=0, DPI reserves DMA engine 5 for all DPI instruction-input (see Section 12.1) and packet-output (see Section 12.2) read operations.

When DPI_DMA_CONTROL[PKT_EN]=1 and DPI_DMA_CONTROL[PKT_EN1]=1, DPI reserves DMA engine 5 for DPI input gather list and packet data reads, and reserves DMA engine 4 for the input instruction reads and all read operations required for packet output. When both packet input and packet output are used, when the reset (zero) value for FPA_PACKET_THRESHOLD is used, and when FPA may run out of pool 0 buffers for PIP/IPD, this mode (with both PKT_EN and PKT_EN1 set) is required to guarantee DPI packet output forward progress.

- DPI instruction input gather list and packet data reads generally requires more larger read operations, so may require a large DMA engine 5 for maximum performance.
- DPI instruction input instruction and packet output generally requires fewer smaller read operations, so a smaller DMA engine size may be adequate for engine 4, or a very small for engine 5 when only packet output is used.

Section 12.3.5 describes engine assignment to execute DMA queue instructions.

Cavium Confidential For Bowen Yang 9/8/2014

The DMA engine MAC memory space read performance is a function of the engine FIFO size, the maximum read size (DPI_SLI_PRT(0..2)_CFG[MRRS]), and the MAC performance. Other performance factors include:

- the number of outstanding read operations for the MAC (DPI_SLI_PRT(0..2)_CFG[MOLR])
- the naturally aligned MRRS termination requirement (controlled by DPI_SLI_PRT(0..2)_CFG[MRRS_LIM])
- the read precision specified for the MAC (DPI_SLI_PRT(0..2)_CFG[RD_MODE]).

Clearing DPI_SLI_PRT(0..2)_CFG[MRRS_LIM] allows one read operation (that crosses the natural block boundary) to replace two read operations when possible, which can improve performance. Setting DPI_SLI_PRT(0..2)_CFG[RD_MODE] can also allow DPI to complete the instruction with fewer read operations, which can also improve performance.

DPI DMA engines generally terminate MAC memory space read operations at naturally aligned block boundaries, where the size of the naturally aligned block is the maximum read size (MRRS) for the MAC. Within these and other hard constraints (FIFO size, MRRS, 4KB boundary, etc), DMA engines maximize the size of the MAC-memory-space read operations, so generally minimize the number of MAC memory space read operations. DPI never generates a MAC memory space read or write operation that crosses a 4KB boundary.

DPI DMA engines split the MAC memory space write operations needed for large destination pointers in outbound or external-only instructions at naturally aligned block sizes, where the block size is the programmed maximum payload size (DPI_SLI_PRT(0..2)_CFG[MPS]). DPI_SLI_PRT(0..2)_CFG[MPS_LIM] = 0 eliminates this alignment restriction for destination pointers that can be completed with one write operation rather than two.

Internal-only DMA operations can dominate the IOBO bandwidth, as CN71XX gives IOBO DMA read responses statically higher priority than IOBO core-initiated requests (REQ), store commit responses (COM), and IOB DWB free completions (FPA) when IOB_OUTB_REQ_PRI_CNT[CNT_ENB] = 0, IOB_OUTB_COM_PRI_CNT[CNT_ENB] = 0, and IOB_OUTB_FPA_PRI_CNT[CNT_ENB] = 0, respectively.

To avoid IOBO starvation when internal-only DMA operations are used, all of IOB_OUTB_*_PRI_CNT[CNT_ENB] should be set to 1, with IOB_OUTB_*_PRI_CNT[CNT_VAL] also set to reasonable values. The values 10, 50, and 100 may be reasonable CNT_VAL values for REQ, COM, and FPA, respectively. These settings may also be useful in the outbound case.

DMA engines statically prioritize read operations over write operations when necessary. When there is contention among the read/write operations required by multiple engines for the same IOBO or MAC resources at a given time, DPI arbitrates round-robin among the engines for the resource.

Cavium Confidential For Bowen Yang ALU 12/7/2014

12.3.10 DMA Configuration and Initialization

12.3.10.1 DMA Global Initialization

There are some global initializations that should only occur under limited circumstances and generally first in an initialization sequence.

The DPI_ENG(0..5)_BUF[BLKS] DMA engine FIFO size selection and DPI_DMA_CONTROL[PKT_EN,PKT_EN1] changes must only occur when all DMA engines are idle. Note that DPI instruction input and DPI packet output must be idle to satisfy this condition when DPI_DMA_CONTROL[PKT_EN,PKT_EN1] is set.

Also, DPI_DMA_CONTROL contains a number of global configuration fields (O_MODE, O_ES, O_NS, O_RO, O_ADD1, FPA_QUE, DWB_ICHK, DWB_DENB, BO_LEND, COMMIT_MODE) that are shared by DMA queues/engines. These configuration fields should change value only when all DMA queues and all DMA engines are idle, except that engine 5 need not be idle when DPI_DMA_CONTROL[PKT_EN]=1, and engine 4 need not be idle when DPI_DMA_CONTROL[PKT_EN1]=1.

DPI_CTL[EN] should be set before any subsequent DPI initialization.

12.3.10.2 DPI DMA Interrupts

There are two DMA shared counters, SLI_DMA0/1_CNT[CNT], and their corresponding thresholds, SLI_DMA0/1_INT_LEVEL[CNT,TIME], that cause interrupts in SLI_INT_SUM[DCNT,DTIME]. These interrupts have corresponding enable signals SLI_INT_ENB_PORT0/1[DCNT,DTIME].

These interrupts are most useful for host output queueing (refer to Section 12.3.8). SLI_DMA0/1_INT_LEVEL[CNT,TIME] values should not change when SLI_DMA0/1_CNT[CNT] are non-zero.

There are DPI DMA instruction-completion interrupts (one per core). If DPI_DMA_CONTROL[DICI_MODE] is set, DPI increments DPI_DMA_PP(0..3)_CNT[CNT] upon a DPI DMA instruction completion, based on the DMA instruction header. See Section 12.3.1. CIU_SUM1_*xx_yy*[DPI_DMA] asserts whenever DPI_DMA_PP(0..3)_CNT[CNT] is non-zero.

There are also DPI DMA error interrupts in DPI_INT_REG and corresponding enable signals in DPI_INT_EN.

Chapter 13 describes how the CIU delivers these interrupts either locally or to a remote MAC host.

12.3.10.3 DPI DMA Engine Initialization

Software must allocate space for the DMA engines and select whether engines 4 and 5 are used by DPI instruction input/packet output or generally available. Section 12.3.10.1 above covers this as part of the global DPI initialization that should occur before other DPI DMA initialization.

Finally, the engines should be assigned to the appropriate queues by setting bits in DPI_DMA_ENG(0..5)_EN[QEN]. Refer to Section 12.3.6.

Cavium Confidential For Bowen Yang 02/27/2014

12.3.10.4 DMA Instruction Queue Initialization

DMA queue initialization starts by writing DPI_DMA*n*_IBUFF_SADDR[SADDR] with a pointer to the first instruction chunk that is used for the queue, and DPI_DMA*n*_IBUFF_SADDR[CSIZE] with the chunk size that is used for the queue. (Section 12.3.4 describes the DMA instruction chunks). This write operation resets both the DMA engine's doorbell (DPI_DMA*n*_COUNTS[DBELL,FCNT]) and its tail pointer (DPI_DMA*n*_NADDR[ADDR]).

Then the enable for the DPI DMA queue (the appropriate bit of DPI_REQ_GBL_EN[QEN]) should be set to 1, which allows the DPI DMA queue to respond to subsequent doorbell rings.

12.3.10.5 DMA Instruction Queue Disable

First, disable the DPI DMA queue by clearing the appropriate bit of DPI_REQ_GBL_EN[QEN] to 0. This stops the DMA queue from starting new DMA instructions on engines.

Next, wait for DPI_DMA*n*_IBUFF_SADDR[IDLE] to be set, indicating that the DMA queue is idle. The DMA queue is now disabled. All data will have been moved (and SLI_DMA0/1_CNT[CNT] adjusted, if necessary) for all processed instructions at this point, but the corresponding zero-byte write operation or work-queue entry add might not be done.

At this point, it may be appropriate for software to free the instruction chunks that were used for the DMA engine instruction queue (refer to Section 12.3.4). DPI_DMA*n*_IBUFF_SADDR[SADDR] indicates the first (i.e. tail) instruction chunk. Software can free up this and any other instruction chunks between this tail and the software head pointer.

If the corresponding DPI_REQ_ERR_RSP_EN[EN] bit is set (as it normally should be), then the corresponding DPI_REQ_ERR_RSP[QERR] may need to be cleared out before the queue can be re-enabled. This is similarly true for DPI_REQ_ERR_RST_EN[EN]/DPI_REQ_ERR_RST[QERR].

NOTE: Be aware that disabling a DMA queue does not modify any of SLI_DMA0/1_CNT[CNT] bits. It may also be appropriate for software to zero SLI_DMA0/1_CNT[CNT] after disabling a DPI DMA queue.

12.3.10.6 DMA Engine Unmapping

A DPI DMA engine can be unmapped from instruction queues by clearing corresponding DPI_DMA_ENG(0..5)_EN[QEN] bits. After that, DPI does not schedule instructions from the DPI instruction queue to the unmapped engine.

Cavium Confidential For Bowen Yang 02/27/2014

12.4 SLI MAC BAR1/BAR2 Target Mode L2C/DRAM Load/Store Operations

This section describes the response of SLI to arriving MAC read/write operations to L2C/DRAM. SLI can service BAR1/BAR2 MAC read operations. Figure 12–1 shows this target-mode logic within the entire SLI logic whole. There are two “target mode” blocks within the SLI logic – one per MAC/ physical port. References from the different cores flow independently.

Each received read/write operation has the following attributes:

- Endian-swap
- Cache-allocate
- No-snoop
- Relaxed order

The endian-swap and cache-allocate attributes are CN71XX-specific, not part of any MAC specification.

Section 12.7 describes the various **endian-swap** modes that CN71XX supports. SLI applies the relevant swap to all read/write operations.

- For BAR1, the endian-swap mode comes from PEM0/1_BAR1_INDEX*n*[END_SWP].
- For BAR2, the endian-swap mode comes from PEM0/1_BAR_CTL[BAR2_ESX] ⊕ PCI_Address<39:38>

The **cache-allocate** attribute determines L2C behavior on a cache miss. If set, L2C does not leave the block in the L2 cache after the reference if it was not already present. Note that the cache-allocate attribute has no effect when the reference hits in the L2 cache. Note also that the attribute only affects behavior for store operations for a full cache-block write operation that misses in the L2 cache. In all other cases, L2C must allocate the block in the L2 cache before completing the store operation.

- For BAR1, the cache-allocate attribute comes from PEM0/1_BAR1_INDEX*n*[CA].
- For BAR2, the cache-allocate attribute comes from PEM0/1_BAR_CTL[BAR2_CAX] ⊕ PCI_Address<40>.

SLI ignores the **no-snoop** attribute of any received PCIe read/write operation.

SLI uses the **relaxed-order** attribute to alter ordering as described in the following paragraphs. The relaxed-order attribute is an attribute of PCIe memory space read/write operation.

The read and write operations that arrive on the two physical MAC ports attached to SLI and are destined for L2/DRAM eventually arrive at the target-mode block for the physical MAC port (refer to Figure 12–1). The read/write operations often arrive at the target-mode logic in the order that they arrived at the physical MAC port, but may arrive in a different order that is consistent with the MAC ordering rules due to the internal switch.

Read operations are kept strictly in order, write operations are kept strictly in order, but write operations may pass read operations that arrived on the same physical PCIe port earlier.

The SLI target-mode logic forwards all read and write operations onto the IOI in an order that is consistent with the PCI ordering rules. With read operations, the target-mode logic allocates internal FIFO space for the returning read data before it forwards the read onto the IOI. After the read and write operations reach the IOI, CN71XX hardware forwards the read and write operations through IOB and onto the

Cavium Confidential For Bowei Yang ALI 02/27/2014

coherent bus in order, as it does with other CN71XX-internal DMA devices.

For strict store ordering, SLI target-mode logic can delay a subsequent L2/DRAM store operation until all prior L2/DRAM store operations are visible throughout the CN71XX. Strictly speaking, these waits are necessary for the arriving L2/DRAM stores to be totally ordered under all possible circumstances. But the waits can be detrimental to performance and may be overly restrictive in some circumstances. Consequently, SLI provides higher-performance options. SLI target-mode logic does not cause an L2/DRAM store to wait for prior stores to commit when either one of the following is true:

- The relaxed order attribute is set for the store and SLI_CTL_PORT(0..2)[PTLP_RO] = 1 to enable relaxed-ordering for the store, or
- SLI_CTL_PORT(0..2)[WAIT_COM] = 0.

NOTE: It may be unlikely that any system/software will observe a store ordering difference with or without the waits. Therefore, we recommend that SLI_CTL_PORT(0..2)[WAIT_COM] normally be clear.

Unlike store operations, the SLI target-mode logic never delays L2/DRAM read operations because prior target mode L2/DRAM stores are uncommitted. SLI follows the normal PCIe ordering rules: the read may appear on the IOI immediately following prior target mode L2/DRAM stores on the IOI. SLI always sends loads on to the IOI in the order that they arrived at the SLI target-mode logic.

IOB read-response data always returns to SLI in the order that SLI sent the read requests to the IOI, and SLI forwards the data requested by the MAC in the same order.

SLI also forwards new MAC memory-space requests to the MAC via its internal switch. These new requests could be either read or write requests generated by either the cnMIPS cores or the DPI packet-input, DPI packet-output, or DPI DMA engines. The MAC further orders the read responses and the new requests. The specific information for the MACs are as follows:

- SLI/PEM always split PCIe read completions on 128-byte naturally aligned boundaries, regardless of the maximum payload size. PEM has posted, nonposted, and completion buffers to receive data from the SLI switch. PEM allows completions with the relaxed-ordering attribute set to pass prior posted (new) store operations when PEM0/1_CTL_STATUS[RO_CTLP] = 1.

Cavium Confidential For Bowen Zheng ARU 02/2/2011

12.5 SLI Core-initiated Request Logic

This section details the SLI-internal processing of MAC memory-space requests that are initiated by the cores. Figure 12–1 shows this core-initiated logic within the entire SLI logic whole. Note that this "core-initiated" logic is separate for each MAC core/physical MAC port. Section 12.8 describes how the cores initiate PCIe configuration space, PCIe I/O space, and PCIe (i.e. MAC) memory-space operations via core load, store, and IOBDMA operations.

- Core load operations create naturally-aligned MAC memory space read operations of 1-byte, 2-byte, 4-byte, and 8-byte.
- Core IOBDMA operations can read up to 255 naturally-aligned 64-bit words in a single operation. (IOBDMA operations must not exceed the maximum read-request size and must not cross 4KB boundaries—refer to Section 12.8.2.)
- Core store operations create naturally-aligned MAC memory space write operations of 1B, 2B, 4B, and 8B, and the CN71XX DPI logic can merge the 8B stores as described below.

The DPI core-initiated logic tags these requests with the following attributes based on address:

- endian-swap
- a two-bit ADDRTYPE
  - ADDRTYPE<0> is the relaxed-order attribute.
  - ADDRTYPE<1> is the no-snoop attribute.

Section 12.8 details the addressing/window and configuration combinations that tag a read/write with these attributes.

The SLI core-initiated logic modifies request and response address and data according to the selected endian-swap mode. Section 12.7 details the various endian swap modes that DPI/SLI supports.

The SLI core-initiated logic forwards all read and write operations into the internal SLI switching logic in the same order as the core loads and stores arrive via the IOI for the physical MAC port. When presented with a stream of store operations, the SLI core-initiated logic attempts to merge sequential MAC memory space store operations before it forwards the combined write onto the internal SLI switch.

The read and write operations that exit the SLI core-initiated logic block eventually appear on the corresponding physical MAC port attached to CN71XX (refer to Figure 12–1). The read/write operations destined for a single physical MAC port often appear in the order that they arrived at the internal IOI, but may arrive in a different order that is consistent with the DPI ordering rules due to the internal switching. (Read operations are kept strictly in order, write operations are kept strictly in order, but write operations may pass read operations that arrived on the IOI earlier.)

CN71XX processor cores themselves never merge any stores that are destined for the SLI core-initiated logic. The processor cores can only pass individual stores on to SLI. All MAC memory space store merging is performed by the SLI core-initiated logic.

The SLI core-initiated hardware can combine sequential MAC memory space stores from the same window. (Section 12.8.6 defines a window.) It can then send the stores out as a single (burst) MAC transaction of up to 128 bytes, with the maximum size selected by SLI_MEM_ACCESS_CTL[MAX_WORD].

Cavium Confidential For Bowen Yang 2012/12/21

This SLI core-initiated combining hardware is very simple and limited by these constraints:

- SLI only merges 64-bit store operations; it never merges 32-bit, 16-bit nor 8-bit core stores.
- All cores share a single active combining buffer for the MAC, and SLI combines store operations originating from any core. A single combined store operation may contain store data originating from different cores.
- SLI permits combining only when the new MAC memory space store operation sequentially follows the previous store operation.
- SLI disables combining and activates the combined store operation if another store operation does not arrive within a programmable time period (SLI_MEM_ACCESS_CTL[TIMER]).
- SLI disables combining and activates the previous combined store operation immediately if a load, IOBDMA, or store operation that will not combine arrives. SLI does the same for load/store operations from the core in the I/O or CONFIG spaces and does the same when any of the window registers are written. Non-contiguous store operations do not combine. Store operations in different MAC memory space windows do not combine. Only store operations to MAC memory space combine.
- SLI does not combine store operations and sends out all store operations immediately when combining is disabled. Combining is disabled when SLI_MEM_ACCESS_CTL[MAX_WORD] = 1 or when SLI_MEM_ACCESS_SUBID*[NMERGE] = 1 for the window.
- SLI disables combining and sends out the store operation when the size of the merged store operation equals the maximum indicated by SLI_MEM_ACCESS_CTL[MAX_WORD].

The SLI core-initiated logic eventually receives a completion for each pending MAC read operation. It may return the responses to the cores out of order relative to when the corresponding requests appeared on the IOI. The SLI core-initiated logic can delay load/IOBDMA responses to the cores until prior target mode L2C/DRAM store operations from the same physical MAC interface have been written to L2/CDRAM and are committed. This is necessary to guarantee that earlier MAC (posted) L2C/DRAM store operations from the same physical MAC port are visible to the cores before the core receives the read completion. Store ordering has two phases: the store arrives on the IOI, and the store commits. The commit is the absolute indication of order, as the order of the store operations on the IOI may not be maintained by L2C. These ordering delays can have negative performance effects and it may not be required in many cases. Consequently, CN71XX provides higher-performance options. Note that completions never stall for target mode L2C/DRAM stores from the other physical MAC port.

If SLI_CTL_PORT*n*[WAITL_COM] = 0, a read completion never stalls for commits of prior target mode L2C/DRAM stores from the same physical MAC port. When a completion stalls due to a prior store, it can only ever stall for the store to be sent on the IOI, and not for the commit of the store.

If SLI_CTL_PORT*n*[WAITL_COM] = 1, when a read completion stalls, it stalls for the commit of prior target mode L2C/DRAM stores from the same physical MAC port (which also implies that it is stalling for the prior stores to be sent on the IO interconnect).

Cavium Confidential For Bowen Yang 09/20/24

NOTE: It may be difficult for any system/software to observe any L2/DRAM ordering difference with or without the waits. If that is the case, it is appropriate for SLI_CTL_PORT*n*[WAITL_COM] to be clear.

When the completion has its relaxed ordering bit set and SLI_CTL_PORT*n*[CTLP_RO]=1, the completion does not wait for any prior stores—core-initiated completions can pass any stores in this case. Otherwise, the completion waits for prior target mode stores that arrived from the same physical MAC port according to the selected SLI_CTL_PORT*n*[WAITL_COM] mode.

Core-initiated read/write operations are buffered in various points inside CN71XX—the CPU cores, IOB, and the core-initiated SLI block. The CPU cores have mechanisms to wait for the outstanding load/IOBDMA operations that cause read operations, but there is no corresponding mechanism to wait for posted store operations, so it is difficult for the cores to determine when the posted store operations complete. One approach: software can guarantee that the posted store operation reached the physical MAC port by following the store operation with a load operation and waiting for the load operation to complete. This can be expensive, so SLI provides a lower-overhead alternative.

Software can guarantee that prior MAC store operations to physical port n reach the other side of SLI's internal switch, which is sufficient to consider the store operation completed in many circumstances, with the following sequence:

- Read SLI_DATA_OUT_CNT[P*n*_FCNT,P*n*_UCNT]. If SLI_DATA_OUT_CNT[P*n*_FCNT] = 0, no further action is necessary. Let A = the original SLI_DATA_OUT_CNT[P*n*_UCNT] value, N = the original SLI_DATA_OUT_CNT[P*n*_FCNT] value, and B = (A + (N – 1)) modulo 2^{16}.
- Read SLI_DATA_OUT_CNT[P*n*_UCNT] until it does not equal a value in the range A .. B. (As the arithmetic is modulo 2^{16}, note that B may be smaller than A, and that the range A .. B always contains exactly *N* elements.)

12.6 CN71XX as a MAC Interrupt Destination

This section describes CN71XX behavior in response to receiving PCIe interrupts. External devices can send interrupts to CN71XX via one of two in-band mechanisms:

- the PCI Express-defined legacy INTA/INTB/INTC/INTD scheme, or
- Message-Signalled Interrupts (MSI).

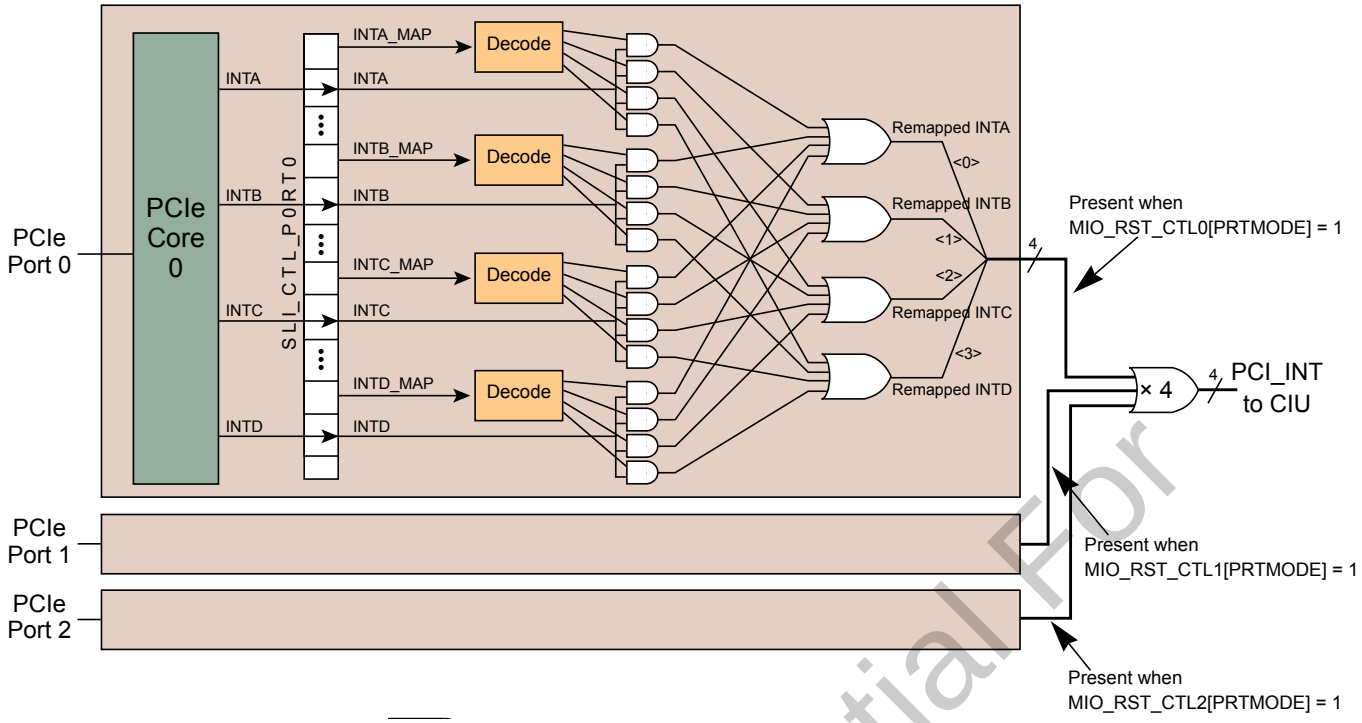
12.6.1 CN71XX Receiving Emulated INTA/INTB/INTC/INTD

The PCI Express specifications describe how the legacy PCI INTA/INTB/INTC/INTD are implemented in-band on the DPI interface. The CN71XX DPI core translates these messages into real INTA/INTB/INTC/INTD interrupt wires using the techniques described in the PCIe specifications.

Figure 12–14 shows how SLI further handles the resultant interrupt inputs. SLI_CTL_PORT*n*[INTA,INTB,INTC,INTD] indicate the current value of these wires for DPI MAC port *n*. SLI remaps these wires as directed by SLI_CTL_PORT*n*[INTA_MAP,INTB_MAP,INTC_MAP,INTD_MAP]. Each INT wire contributes to one of the remapped INTs. Any combination of INT wires can contribute to a given remapped INT.

Cavium Confidential For Bowen Yang 02/7/2014

Figure 12–14 CN71XX Handling of INTA/INTB/INTC/INTD Inputs



SLI sends the four summed remapped INTs to CIU for further distribution. Figure 13–3 describes CN71XX's further processing of the interrupts.

Note that only CN71XX PCIe MACs that are in RC mode can receive INTA/INTB/INTC/INTD interrupts. PCIe MACs ports 0 and 1 are in RC mode when corresponding MIO_RST_CTL0/1[PRTMODE] = 1. When MIO_RST_CTL*n*[PRTMODE] = 0, PCIe MAC port *n* is in EP mode and cannot receive SLI INTA/INTB/INTC/INTD interrupts (but can generate outbound INTA interrupts as shown in Chapter 13 and Figure 13–3).

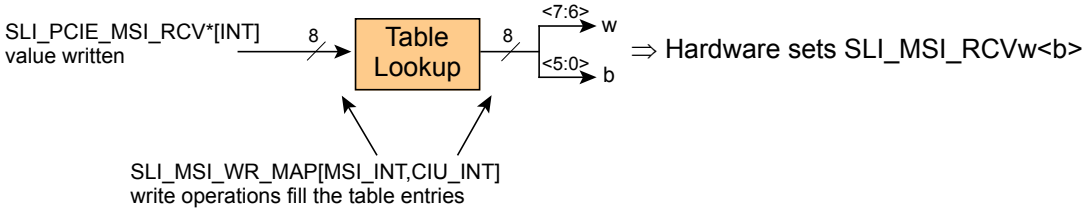
12.6.2 CN71XX Receiving Message-Signaled Interrupts

A message-signaled interrupt (MSI) arrives at CN71XX as a MAC memory space write operation to one of the four SLI BAR0 CSRs called SLI_PCIE_MSI_RCV*. SLI sets one of the 256 bits in the SLI_MSI_RCV* CSRs on receipt of these write operations, which can cause an interrupt. Software can clear SLI_MSI_RCV* bits—typically as part of the interrupt handler to prepare for future interrupts.

Figure 12–15 depicts how SLI selects the particular bit that gets set for each SLI_PCIE_MSI_RCV* write. SLI internally contains a remap table that is filled by software via write operations to SLI_MSI_WR_MAP[MSI_INT,CIU_INT]. Each SLI_PCIE_MSI_RCV*[INT] value maps to exactly one of the 256 bits in SLI_MSI_RCV*. Each SLI_MSI_RCV* bit can be set by any combination of the SLI_PCIE_MSI_RCV*[INT] values with proper configuration of the table.

Cavium Confidential For Power Ping ALC 8/4/2014

Figure 12–15 SLI Bit Selection for Each SLI_PCIE_MSI_RCV* Write



The SLI table remap shown in Figure 12–15 can be advantageous, for example, with external devices that can only send contiguous SLI_DPI_MSI_RCV*[INT] values. The remap can convert any SLI_DPI_MSI_RCV*[INT] value into any desired CIU "priority."

Chapter 13 and Figure 13–1 depict how the SLI_MSI_RCV* interrupt bits are used in CN71XX. CN71XX reduces the 256 interrupts into four separate "priorities" (i.e. the CIU_INT*n*_SUM0/4[PCI_MSI] field is 4 bits) that can be distributed independently amongst the cores at different priorities. SLI_MSI_ENB* contain 256 enables, one for each SLI_MSI_RCV* bit.

- CIU_INT*n*_SUM0/4[PCI_MSI<0>] is set whenever both SLI_MSI_RCV0<*n*> and SLI_MSI_ENB0<*n*> is set for any one of the possible 64 values of *n*.
- CIU_INT*n*_SUM0/4[PCI_MSI<1>] is set whenever both SLI_MSI_RCV1<*n*> and SLI_MSI_ENB1<*n*> is set for any one of the possible 64 values of *n*. ...

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

12.7 Endian Swapping

For all cases, assume the 64-bit data on the MAC bus is represented as:

[A-B-C-D-E-F-G-H]

where each letter represents a byte. Eight bytes represent 64-bits. A is the most-significant byte, H is the least-significant byte.

In the case of 32-bit PCIe bus transfers, [E-F-G-H] is the first 32-bits, corresponding to the case when <2> of the PCIe byte address is clear, and [A-B-C-D] is the second 32-bits, corresponding to the case when <2> of the PCIe byte address is set.

12.7.1 Pass-Through Mode

No swap by SLI: [A-B-C-D-E-F-G-H]. CN71XX internal byte order is A-B-C-D-E-F-G-H, from first to last.

An aligned 64-bit load/store operation references [A-B-C-D-E-F-G-H].

An aligned 32-bit load/store operation:

	Core in LE mode	Core in BE mode
OFFSET<2:0> == 0 references	[E-F-G-H]	[A-B-C-D]
OFFSET<2:0> == 4 references	[A-B-C-D]	[E-F-G-H]

An aligned 16-bit load/store operation:

	Core in LE mode	Core in BE mode
OFFSET<2:0> == 0 references	[G-H]	[A-B]
OFFSET<2:0> == 2 references	[E-F]	[C-D]
OFFSET<2:0> == 4 references	[C-D]	[E-F]
OFFSET<2:0> == 6 references	[A-B]	[G-H]

An 8-bit load/store operation:

	Core in LE mode	Core in BE mode
OFFSET<2:0> == 0 references	[H]	[A]
OFFSET<2:0> == 1 references	[G]	[B]
OFFSET<2:0> == 2 references	[F]	[C]
OFFSET<2:0> == 3 references	[E]	[D]
OFFSET<2:0> == 4 references	[D]	[E]
OFFSET<2:0> == 5 references	[C]	[F]
OFFSET<2:0> == 6 references	[B]	[G]
OFFSET<2:0> == 7 references	[A]	[H]

Cavium Confidential For Bowen Yang ALU 02/27/2014

12.7.2 64-bit Byte-Swap Mode

Full swap by SLI, [H-G-F-E-D-C-B-A], CN71XX internal byte order is H-G-F-E-D-C-B-A, from first to last.

An aligned 64-bit load/store operation references [H-G-F-E-D-C-B-A].

An aligned 32-bit load/store operation:

	Core in LE mode	Core in BE mode
OFFSET<2:0> = 0 references	[D-C-B-A]	[H-G-F-E]
OFFSET<2:0> = 4 references	[H-G-F-E]	[D-C-B-A]

An aligned 16-bit load/store operation:

	Core in LE mode	Core in BE mode
OFFSET<2:0> == 0 references	[B-A]	[H-G]
OFFSET<2:0> == 2 references	[D-C]	[F-E]
OFFSET<2:0> == 4 references	[F-E]	[D-C]
OFFSET<2:0> == 6 references	[H-G]	[B-A]

An 8-bit load/store operation:

	Core in LE mode	Core in BE mode
OFFSET<2:0> == 0 references	[A]	[H]
OFFSET<2:0> == 1 references	[B]	[G]
OFFSET<2:0> == 2 references	[C]	[F]
OFFSET<2:0> == 3 references	[D]	[E]
OFFSET<2:0> == 4 references	[E]	[D]
OFFSET<2:0> == 5 references	[F]	[C]
OFFSET<2:0> == 6 references	[G]	[B]
OFFSET<2:0> == 7 references	[H]	[A]

12.7.3 32-bit Byte-Swap Mode

Swap of the bytes within a 32-bit PCI Dword by SLI: [D-C-B-A-H-G-F-E]. CN71XX internal byte order is D-C-B-A-H-G-F-E, from first to last.

An aligned 64-bit load/store references [D-C-B-A-H-G-F-E].

An aligned 32-bit load/store operation:

	Core in LE mode	Core in BE mode
OFFSET<2:0> == 0 references	[H-G-F-E]	[D-C-B-A]
OFFSET<2:0> == 4 references	[D-C-B-A]	[H-G-F-E]

An aligned 16-bit load/store operation:

	Core in LE mode	Core in BE mode
OFFSET<2:0> == 0 references	[F-E]	[D-C]
OFFSET<2:0> == 2 references	[H-G]	[B-A]
OFFSET<2:0> == 4 references	[B-A]	[H-G]
OFFSET<2:0> == 6 references	[D-C]	[F-E]

Cavium Confidential For Bowen Yang ALU 02/27/2014

An 8-bit load/store operation:

	Core in LE mode	Core in BE mode
OFFSET<2:0> == 0 references	[E]	[D]
OFFSET<2:0> == 1 references	[F]	[C]
OFFSET<2:0> == 2 references	[G]	[B]
OFFSET<2:0> == 3 references	[H]	[A]
OFFSET<2:0> == 4 references	[A]	[H]
OFFSET<2:0> == 5 references	[B]	[G]
OFFSET<2:0> == 6 references	[C]	[F]
OFFSET<2:0> == 7 references	[D]	[E]

12.7.4 32-bit Exchange Mode

PCI Dword (32-bit) exchange: [E-F-G-H-A-B-C-D]. CN71XX internal byte order is E-F-G-H-A-B-C-D, from first to last.

An aligned 64-bit load/store references [E-F-G-H-A-B-C-D].

An aligned 32-bit load/store operation:

	Core in LE mode	Core in BE mode
OFFSET<2:0> == 0 references	[A-B-C-D]	[E-F-G-H]
OFFSET<2:0> == 4 references	[E-F-G-H]	[A-B-C-D]

An aligned 16-bit load/store operation:

	Core in LE mode	Core in BE mode
OFFSET<2:0> == 0 references	[C-D]	[E-F]
OFFSET<2:0> == 2 references	[A-B]	[G-H]
OFFSET<2:0> == 4 references	[G-H]	[A-B]
OFFSET<2:0> == 6 references	[E-F]	[C-D]

An 8-bit load/store operation:

	Core in LE mode	Core in BE mode
OFFSET<2:0> == 0 references	[D]	[E]
OFFSET<2:0> == 1 references	[C]	[F]
OFFSET<2:0> == 2 references	[B]	[G]
OFFSET<2:0> == 3 references	[A]	[H]
OFFSET<2:0> == 4 references	[H]	[A]
OFFSET<2:0> == 5 references	[G]	[B]
OFFSET<2:0> == 6 references	[F]	[C]
OFFSET<2:0> == 7 references	[E]	[D]

Cavium Confidential For Bowen Yang ALU 02/27/2014

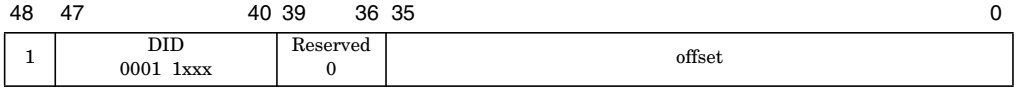
12.8 SLI Bus Operations

Operations by the cores to SLI generate MAC requests.

Cores access MAC addresses with physical addresses of the format described in the following subsections.

12.8.1 Load/Store Operations

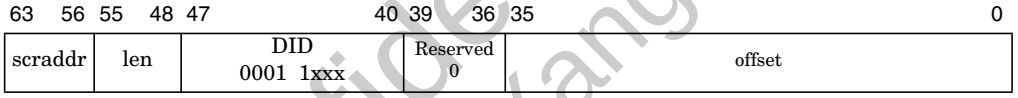
Physical Address Format for Load/Store Operations



- **DID** – Device IDs of the SLI block. The address space for DID values 0x18–0x1E to the SLI block are defined in Sections 12.8.3 – 12.8.7.
  - 0x18 = RSL access space
  - 0x19 = PCIe configuration space
  - 0x1A = PCIe I/O space
  - 0x1B, 0x1C, 0x1D, 0x1E = MAC memory space
  - 0x1F = SLI CSRs
- **Offset** – Used differently depending on DID value.

12.8.2 IOBDMA Operations

IOBDMA Addressing



- **scraddr** – Defined in Section 2.7.
- **len** – Indicates the length of the IOBDMA operation in 64-bit doublewords.
  - any len value is legal as long as it is within the range allowed by SLI_S2M_PORT0/1_CTL[MRRS].
  - must be 1 for DIDs 0x18 and 0x1F.
  - len must not be so large such that the contiguous address range covered by the IOBDMA operation crosses a naturally aligned 4KB boundary.
- **DID** – Selects the space from among those described in Sections 12.8.3, 12.8.6, and 12.8.7.
  - 0x18 = RSL access space
  - 0x1B, 0x1C, 0x1D, 0x1E = MAC memory space
  - 0x1F = SLI CSR address space
- **offset** – Used differently depending on the DID value. Must be naturally aligned on a 64-bit boundary (i.e. <2:0> must be zero). The address range

  offset .. offset + (len × 8) – 1

  must not cross a naturally aligned 4KB boundary.

NOTE: IOBDMA operations are not supported to either PCIe configuration space (DID = 0x19) nor PCIe I/O space (DID = 0x1A).

Cavium Confidential For Boyan Yang ALU 02/27/2014

12.8.3 RSL Access Space (DID = 0x18)

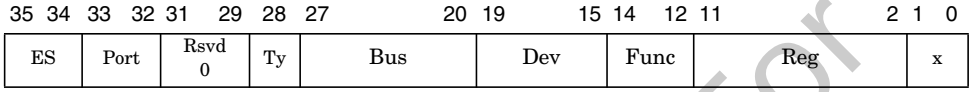
Cores can access CN71XX's various RSL-type CSRs via load/store/IOBDMA operations to this DID = 0x18 space. Any size load is allowed, but only 64-bit store operations are allowed to this space. IOBDMA operations to this space must be of length 1 (8 bytes).

12.8.4 PCIe Configuration Space (DID = 0x19)

Cores can issue PCIe CONFIG requests via load/store operations to this DID = 0x19 space. IOBDMA and 64-bit load/store operations are not allowed to this space.

The following is the format of the OFFSET field.

OFFSET Field Format



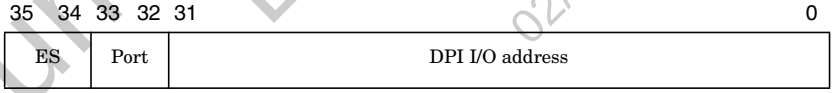
- **ES** – Indicates the endian-swap mode.
- **Port** – Selects which CN71XX PCIe MAC port the request is sent out. Legal values are 0 and 1.
- **Ty** – Selects the type of the configuration request (0 = type 0, 1 = type 1).
- **Bus** – Target bus number sent in the ID in the request.
- **Dev** – Target device number sent in the ID in the request. Note that Dev must be zero for type 0 configuration requests.
- **Func** – Target function number sent in the ID in the request.
- **Reg** – Selects a register in the configuration space of the target.

Note that 32-bit, 16-bit, and 8-bit aligned loads/stores are allowed; 64-bit accesses and IOBDMA operations are illegal.

12.8.5 PCIe I/O Space (DID = 0x1A)

Cores can issue PCIe I/O load/store operations in this space. OFFSET<31:0> is the I/O address sent in the request. IOBDMA and 64-bit load/store operations are not allowed to this space.

OFFSET Field Format in I/O space



- **ES** – Indicates the endian-swap mode.
- **Port** – Selects which CN71XX PCIe MAC port the request is sent out. Legal values are 0 and 1.

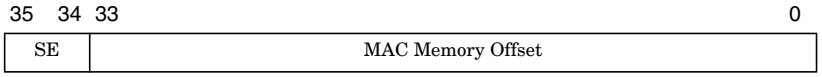
Cavium Confidential For Bowen Yang AU 02/27/2014

12.8.6 MAC Memory Space (DID = 0x1B, 0x1C, 0x1D, 0x1E)

Software must not issue an IOBDMA operation that is larger than the maximum read request size for the MAC port (SLI_S2M_PORT0/1_CTL[MRRS]).

Cores can issue Load/Store/IOBDMA operations in this space to generate MAC memory space read/write requests. (Note that CN71XX cannot generate PCIe memory read-lock requests.)

OFFSET DECODE in this Memory space



- **SE** – DID extender. With DID, selects window.
- **MAC Memory Offset** – Low 34 bits of the address in the generated MAC memory space request.

The combination of the DID and SE in the address selects one of the 16 windows available to translate the address onto a MAC port. Each window is defined by its corresponding SLI_MEM_ACCESS_SUBID(12..27) register, which defines the following fields for the window:

- **[ZERO]** Set to generate zero-length read operations. When set, the hardware generates a zero-length read and returns a result of zero for all core load-byte instructions.
- **[PORT]** MAC port the request is sent out.
- **[NMERGE]** No merging is allowed in this window.
- **[ESR]** Endian-swap mode for read operations.
- **[ESW]** Endian-swap mode for write operations.
- **[WTYPE]** ADDRTYPE<1:0> for write operations.
- **[RTYPE]** ADDRTYPE<1:0> for read operations.
- **[BA]** The MAC Address bits <63:34>.

The 16 windows are apportioned to the CSRs in the following way:

SLI_MEM_ACCESS_SUBID12 corresponds to DID = 0x1B, SE = 0x0.
SLI_MEM_ACCESS_SUBID13 corresponds to DID = 0x1B, SE = 0x1.
SLI_MEM_ACCESS_SUBID14 corresponds to DID = 0x1B, SE = 0x2.
SLI_MEM_ACCESS_SUBID15 corresponds to DID = 0x1B, SE = 0x3.

SLI_MEM_ACCESS_SUBID16 corresponds to DID = 0x1C, SE = 0x0.
SLI_MEM_ACCESS_SUBID17 corresponds to DID = 0x1C, SE = 0x1.
SLI_MEM_ACCESS_SUBID18 corresponds to DID = 0x1C, SE = 0x2.
SLI_MEM_ACCESS_SUBID19 corresponds to DID = 0x1C, SE = 0x3.

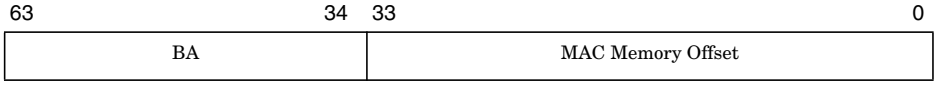
SLI_MEM_ACCESS_SUBID20 corresponds to DID = 0x1D, SE = 0x0.
SLI_MEM_ACCESS_SUBID21 corresponds to DID = 0x1D, SE = 0x1.
SLI_MEM_ACCESS_SUBID22 corresponds to DID = 0x1D, SE = 0x2.
SLI_MEM_ACCESS_SUBID23 corresponds to DID = 0x1D, SE = 0x3.

SLI_MEM_ACCESS_SUBID24 corresponds to DID = 0x1E, SE = 0x0.
SLI_MEM_ACCESS_SUBID25 corresponds to DID = 0x1E, SE = 0x1.
SLI_MEM_ACCESS_SUBID26 corresponds to DID = 0x1E, SE = 0x2.
SLI_MEM_ACCESS_SUBID27 corresponds to DID = 0x1E, SE = 0x3.

Cavium Confidential For Binbin Yang ALU 02/27/2014

The 64-bit address presented on the MAC port for these references is:

64-bit Form MAC Memory Offset (i.e. MACADD<63:0>)



Section 12.5 details the CN71XX SLI core-initiated logic processing for MAC memory space load/store operations.

12.8.7 SLI CSRs (DID = 0x1F)

Cores can access CN71XX's SLI CSRs via addresses in this DID. IOBDMA operations are allowed in this DID, but must be of length 1 (8 bytes).

Note that the SLI CSRs are also BAR0 CSRs, and can also be accessed from an external device via EP or RC Internal BAR0. Refer to Section 22.2.2.1 and Section 22.2.2.2.

Section 12.11 describes the SLI internal CSRs in more detail.

12.9 Configuration Summary

This section contains lists of important state that should be configured, and a few hints on things that can go wrong.

DPI/SLI CSRs (RC only)

- INTA/B/C/D remap (SLI_CTL_PORT(0..2)[INTA_MAP, INTB_MAP, INTC_MAP, INTD_MAP]).
  - Refer to Section 12.6.1.
- (External) MSI (SLI_MSI_RCV*, SLI_MSI_ENB*, SLI_MSI_*_MAP)

DPI/SLI CSRs (EP or RC)

- Maximum payload size (DPI_SLI_PRT(0..2)_CFG[MPS])
  - must not exceed PCIE*_CFG030[MPS]
- Maximum read request size (DPI_SLI_PRT(0..2)_CFG[MRRS], SLI_S2M_PORT(0..2)_CTL[MRRS])
  - must not exceed PCIE*_CFG030[MRRS]
- BAR0 disable (DPI_SLI_PRT(0..2)_CFG[BAR0_D])
- Window disable (DPI_SLI_PRT(0..2)_CFG[WIND_D])
- For target memory-space accesses (refer to Section 12.4)
  - Relaxed ordering (SLI_CTL_PORT(0..2)[PTLP_RO,CTLP_RO, WAIT_COM])
    - PTLP_RO and CTLP_RO should normally be set (except for debug)
    - WAIT_COM = 0 will likely work for all applications
- For core-initiated PCIe memory-space accesses
  - Memory Access subIDs (SLI_MEM_ACCESS_SUBID*)
    - MAC port selection, base address, ADDRTYPE, endian-swap
  - Store merge control (SLI_MEM_ACCESS_CTL[TIMER,MAX_WORD])
  - Load completion relaxed ordering (SLI_CTL_PORT(0..2)[WAITL_COM])
    - Refer to Section 12.5.
- DPI DMA queues and engines (refer to Section 12.3)

Cavium Confidential For Bowen Yang ALU 8/27/2014

- Enable engines (DPI_CTL[EN])
- Engine control (DPI_DMA_CONTROL)
- Enable queues (DPI_REQ_GBL_EN)
- Allow instruction queues (DPI_DMA_ENG*_EN)
- Instruction buffer starting address (DPI_DMA*_IBUFF_SADDR)
- Doorbells (DPI_DMA*_DBELL)
- Engine FIFO size (DPI_ENG*_BUF)
- Interrupt enables (SLI_DMA0/1_INT_LEVEL, SLI_INT_ENB_*[DCNT, DTIME,DMAFI])

- DPI Packet/Instruction input rings (refer to Section 12.1)
  - Ring enables (SLI_PKT_INSTR_ENB)
  - Ring selected MAC port (SLI_PKT_IN_PCIE_PORT)
  - Ring size (SLI_PKT*_INSTR_FIFO_RSIZE[RSIZE])
  - Ring base address (SLI_PKT*_INSTR_BADDR)
  - Ring instruction sizes (SLI_PKT_INSTR_SIZE)
  - Ring instruction headers (SLI_PKT*_INSTR_HEADER)
  - Ring backpressure threshold (SLI_PKT*_IN_BP[WMARK])
  - Endian-swap, ADDRTYPE (SLI_PKT_INPUT_CONTROL[ROR, ESR,NSR,USE_CSR,D_ROR,D_ESR,D_NSR,PBP_DHI])
  - Per-port backpressure enable (SLI_PKT_CTL[PKT_BP] – refer also to Section 9.6)
- DPI Packet output rings (refer to Section 12.2)
  - Ring enables (SLI_PKT_OUT_ENB)
  - Ring selected MAC port (SLI_PKT_PCIE_PORT)
  - Ring size (SLI_PKT*n*_SLIST_FIFO_RSIZE[RSIZE])
  - Ring base address (SLI_PKT*n*_SLIST_BADDR)
  - Ring buffer size (SLI_PKT*n*_OUT_SIZE[BSIZE])
  - Ring info-pointer or buffer-pointer-only selection (SLI_PKT_IPTR)
  - Ring info-pointer bytes (SLI_PKT*n*_OUT_SIZE[ISIZE])
  - Ring endian-swap, relaxed-order, no-snoop for buffer/info pair fetches (SLI_PKT_SLIST_ROR,SLI_PKT_SLIST_NS)
  - Ring endian-swap, ADDRTYPE for buffer write operations (SLI_PKT_DPADDR, SLI_PKT_DATA_OUT_ROR, SLI_PKT_DATA_OUT_NS, SLI_PKT_DATA_OUT_ES)
  - Ring interrupt control (SLI_PKT_OUT_BMODE, SLI_PKT_INT_LEVELS, SLI_PKT_CNT_INT_ENB, SLI_PKT_TIME_INT_ENB)
  - Buffer/info pair backpressure threshold (SLI_PKT_OUTPUT_WMARK[WMARK])
  - Ring Arbitration (SLI_PKT_INPUT_CONTROL[PKT_RR])

- CSR access control (SLI_S2M_PORT(0..2)_CTL[BAR0_D,WIND_D])

Cavium Confidential For Bowen Yang ALU 02/23/2014

12.10 DPI Registers

Table 12–4 lists the DPI registers.

Table 12–4 DPI Registers

Register	Address	CSR Type	Detailed Description
DPI_BIST_STATUS	0x0001DF0000000000	NCB	See page 680
DPI_INT_REG	0x0001DF0000000008	NCB	See page 680
DPI_INT_EN	0x0001DF0000000010	NCB	See page 681
DPI_CTL	0x0001DF0000000040	NCB	See page 681
DPI_DMA_CONTROL	0x0001DF0000000048	NCB	See page 682
DPI_REQ_GBL_EN	0x0001DF0000000050	NCB	See page 683
DPI_REQ_ERR_RSP	0x0001DF0000000058	NCB	See page 683
DPI_REQ_ERR_RST	0x0001DF0000000060	NCB	See page 684
DPI_REQ_ERR_RSP_EN	0x0001DF0000000068	NCB	See page 684
DPI_REQ_ERR_RST_EN	0x0001DF0000000070	NCB	See page 684
DPI_PKT_ERR_RSP	0x0001DF0000000078	NCB	See page 684
DPI_DMA_ENG0_EN DPI_DMA_ENG1_EN ... DPI_DMA_ENG5_EN	0x0001DF0000000080 0x0001DF0000000088 ... 0x0001DF00000000A8	NCB	See page 685
DPI_DMA0_DBELL ... DPI_DMA7_DBELL	0x0001DF0000000200 ... 0x0001DF0000000238	NCB	See page 685
DPI_DMA0_IBUFF_SADDR ... DPI_DMA7_IBUFF_SADDR	0x0001DF0000000280 ... 0x0001DF00000002B8	NCB	See page 685
DPI_DMA0_COUNTS ... DPI_DMA7_COUNTS	0x0001DF0000000300 ... 0x0001DF0000000338	NCB	See page 686
DPI_DMA0_NADDR ... DPI_DMA7_NADDR	0x0001DF0000000380 ... 0x0001DF00000003B8	NCB	See page 686
DPI_DMA0_REQBNK0 ... DPI_DMA7_REQBNK0	0x0001DF0000000400 ... 0x0001DF0000000438	NCB	See page 686
DPI_DMA0_REQBNK1 ... DPI_DMA7_REQBNK1	0x0001DF0000000480 ... 0x0001DF00000004B8	NCB	See page 686
DPI_NCB0_CFG	0x0001DF0000000800	NCB	See page 687
DPI_PINT_INFO	0x0001DF0000000830	NCB	See page 687
DPI_REQ_ERR_SKIP_COMP	0x0001DF0000000838	NCB	See page 688
DPI_ENG0_BUF ... DPI_ENG5_BUF	0x0001DF0000000880 ... 0x0001DF00000008A8	NCB	See page 688
DPI_SLI_PRT0_CFG DPI_SLI_PRT1_CFG	0x0001DF0000000900 0x0001DF0000000908	NCB	See page 689
DPI_SLI_PRT0_ERR DPI_SLI_PRT1_ERR	0x0001DF0000000920 0x0001DF0000000928	NCB	See page 690
DPI_SLI_PRT0_ERR_INFO DPI_SLI_PRT.1_ERR_INFO	0x0001DF0000000940 0x0001DF0000000948	NCB	See page 691
DPI_INFO_REG	0x0001DF0000000980	NCB	See page 691
DPI_DMA0_IFLIGHT DPI_DMA1_IFLIGHT ... DPI_DMA7_IFLIGHT	0x0001DF0000000A00 0x0001DF0000000A08 ... 0x0001DF0000000A38	NCB	See page 692

Cavium Confidential For Bowen Yang At Dell 12/7/2015

Table 12–4 DPI Registers

Register	Address	CSR Type	Detailed Description
DPI_DMA0_ERR_RSP_STATUS DPI_DMA1_ERR_RSP_STATUS ... DPI_DMA7_ERR_RSP_STATUS	0x0001DF0000000A80 0x0001DF0000000A88 ... 0x0001DF0000000AB8	NCB	See page 692
DPI_DMA_PP0_CNT DPI_DMA_PP1_CNT ... DPI_DMA_PP3_CNT	0x0001DF0000000B00 0x0001DF0000000B08 ... 0x0001DF0000000B18	NCB	See page 692

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

DPI BIST Status Register
DPI_BIST_STATUS

This is the built-in self-test (BIST) status register. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:45>	—	RAZ	—	—	Reserved
<44:0>	BIST	R/W	0x0	0x0	BIST results. Hardware sets a bit in BIST for memory that fails.

DPI Interrupt Register
DPI_INT_REG

This register contains error flags for DPI. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:27>	—	RAZ	—	—	Reserved
<26>	SPRT2_RST	R/W1C/H			SLI port 2 in-reset error. DMA instruction was dropped because the source or destination port was in reset.
<25>	SPRT1_RST	R/W1C/H	0	0	SLI port 1 in-reset error. DMA instruction was dropped because the source or destination port was in reset.
<24>	SPRT0_RST	R/W1C/H	0	0	SLI port 0 in-reset error 0. DMA instruction was dropped because the source or destination port was in reset.
<23>	—	RAZ	—	—	Reserved.
<22>	REQ_BADFIL	R/W1C/H	0	0	DMA instruction unexpected fill error. This bit is set when an instruction fill is received when there is no outstanding request.
<21>	REQ_INULL	R/W1C/H	0	0	DMA instruction null pointer error. This bit is set when an instruction's next pointer is NULL.
<20>	REQ_ANULL	R/W1C/H	0	0	DMA instruction filled with bad instruction. This bit is set when a fetched instruction word was 0x0.
<19>	REQ_UNDFLW	R/W1C/H	0	0	Instruction FIFO underflow error. DPI tracks outstanding instructions fetches. This bit is set when the instruction FIFO underflows.
<18>	REQ_OVRFLW	R/W1C/H	0	0	Instruction FIFO overflow error. DPI tracks outstanding instructions fetches. This bit is set when the instruction FIFO overflows.
<17>	REQ_BADLEN	R/W1C/H	0	0	Fetch with length zero error. This bit is set when DPI forms an instruction fetch with length of zero.
<16>	REQ_BADADR	R/W1C/H	0	0	Fetch with bad pointer error. This bit is set when DPI forms an instruction fetch to the NULL pointer.
<15:8>	DMADBO	R/W1C/H	0x0	0x0	DMA*x* doorbell overflow error. DPI has a 32-bit counter for each request queue's outstanding doorbell counts. A bit is set when the corresponding doorbell count overflows.
<7:2>	—	RAZ	—	—	Reserved
<1>	NFOVR	R/W1C/H	0	0	IOI CSR FIFO overflow error. The DPI can store up to 16 CSR requests, and the FIFO overflows if that number is exceeded. This bit is set when the FIFO overflows.
<0>	NDERR	R/W1C/H	0	0	IOI decode error. This bit is set when the DPI received an IOI transaction on the outbound bus to the DPI device ID, but the command was not recognized.

Cavium Confidential For Bowen Yang ALU 5/12/2014

DPI Interrupt Enable Register
DPI_INT_EN

This register contains enable bits for the error flags in DPI_INT_REG. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:27>	—	RAZ	—	—	Reserved
<26>	SPRT2_RST	R/W/H	0	0	SLI port 2 in-reset error enable.
<25>	SPRT1_RST	R/W/H	0	0	SLI port 1 in-reset error enable.
<24>	SPRT0_RST	R/W/H	0	0	SLI port 0 in-reset error enable.
<23>	—	RAZ	—	—	Reserved
<22>	REQ_BADFIL	R/W/H	0	0	Unexpected fill error enable
<21>	REQ_INULL	R/W/H	0	0	Null pointer error enable
<20>	REQ_ANULL	R/W/H	0	0	Instruction bad error enable
<19>	REQ_UNDFLW	R/W/H	0	0	Instruction FIFO underflow error enable
<18>	REQ_OVRFLW	R/W/H	0	0	Instruction FIFO overflow error enable
<17>	REQ_BADLEN	R/W/H	0	0	Fetch with length zero error enable
<16>	REQ_BADADR	R/W/H	0	0	Fetch with bad pointer error enable
<15:8>	DMADBO	R/W/H	0x0	0x0	Doorbell overflow error enable
<7:2>	—	RAZ	—	—	Reserved
<1>	NFOVR	R/W	0	0	CSR FIFO overflow error enable
<0>	NDERR	R/W	0	0	IOI decode error enable

DPI Control Register
DPI_CTL

This register provides the enable bit for the DMA and packet state machines. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	EN	R/W	0	1	Turns on the DMA and packet state machines.

Cavium Confidential For Bowen Yang ALU 02/27/2014

DPI DMA Control Register
DPI_DMA_CONTROL

This register controls the operation of DMA input and output. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:62>	—	RAZ	—	—	Reserved
<61>	DICI_MODE	R/W	0	—	DMA instruction completion interrupt mode. When set, this bit turns on the mode to increment DPI_DMA_PP*x*_CNT counters.
<60>	PKT_EN1	R/W	0	—	Packet interface 1 enable. When the packet interface is enabled, engine 4 is used for packets and is not available for DMA. The packet interfaces must be enabled in order. When PKT_EN1=1, PKT_EN=1. When PKT_EN1=1, DMA_ENB<4>=0.
<59>	FFP_DIS	R/W	0	0	Force forward progress disable. The DMA engines will compete for shared resources. If the hardware detects that particular engines are not able to make requests to an interface, the hardware will periodically trade-off throughput for fairness.
<58>	COMMIT_MODE	R/W	0	0	DMA engine commit mode. When COMMIT_MODE=1, DPI considers an instruction complete when the hardware internally generates the final write for the current instruction. When COMMIT_MODE=0, DPI additionally waits for the final write to reach the interface coherency point to declare the instructions complete. Please note: when COMMIT_MODE == 1, DPI may not follow the HRM ordering rules. DPI hardware performance may be better with COMMIT_MODE == 1 than with COMMIT_MODE == 0 due to the relaxed ordering rules. If the HRM ordering rules are required, set COMMIT_MODE == 0.
<57>	—	RAZ	—	—	Reserved
<56>	PKT_EN	R/W	0	—	Packet interface 0 enable. When the packet interface is enabled, engine 5 is used for packets and is not available for DMA. When PKT_EN = 1, DMA_ENB<5> = 0. When PKT_EN1 = 1, PKT_EN = 1.
<55:54>	—	RAZ	—	—	Reserved
<53:48>	DMA_ENB	R/W	0x0	0x3F	DMA engine enable. One bit for each engine. After being enabled, an engine should not be disabled while processing instructions. When PKT_EN = 1, DMA_ENB<5> = 0. When PKT_EN1 = 1, DMA_ENB<4> = 0.
<47:34>	—	RAZ	—	—	Reserved
<33>	B0_LEND	R/W	0	0	Little-endian. When set to 1 and the DPI is in the mode to write 0 to L2C when a DMA transaction is done, the address to be written is treated as a little-endian address.
<32>	DWB_DENB	R/W	0	1	Don't-write-back data enable. When set to 1, DPI sends a value in the DWB field for a free-page operation for the memory that contained the data.
<31:23>	DWB_ICHK	R/W	0x0	—	Don't-write-back instruction chunk. When instruction chunks for DMA operations are freed, this value is used for the DWB field of the operation.
<22:20>	FPA_QUE	R/W	0x0	—	FPA queue. Specifies the FPA queue that the instruction-chunk page is returned to when used.

Cavium Confidential For Bowen Yang 01/27/2015

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<19>	O_ADD1	R/W	0	1	Add one. 1 = add 1 to the SLI DMA counters 0 = the number of bytes in the DMA transfer is added to SLI_DMA*_CNT.
<18>	O_RO	R/W	0	—	Relaxed ordering mode for DMA transactions
<17>	O_NS	R/W	0	—	No swoop
<16:15>	O_ES	R/W	0x0	—0	Endian swap mode for DMA. 0 = pass-through mode (no swap) 1 = 64-bit byte-swap mode [ABCD_EFGH] → [HGFE_DCBA] 2 = 32-bit byte-swap mode [ABCD_EFGH] → [DCBA_HGFE] 3 = 32-bit exchange mode [ABCD_EFGH] → [EFGH_ABCD]
<14>	O_MODE	R/W	0	1	Select PCI_POINTER mode. 0 = DPTR format 1 is used. Use register values for address; use pointer values for ES, NS, RO. 1 = DPTR format 0 is used. Use pointer values for address; use register values for ES, NS, RO.
<13:0>	—	RAZ	—	—	Reserved

DPI Request Global Enable Register
DPI_REQ_GBL_EN

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	QEN	R/W	0xFF	0xFF	Indicates which instruction queues are enabled, and can dispatch instructions to a requesting engine.

DPI Instruction Queue ErrorResponse Register
DPI_REQ_ERR_RSP

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	QERR	R/W1C/H	0x0	0x0	Indicates which instruction queue received an ErrorResponse from the I/O subsystem. Software must clear the bit before the corresponding instruction queue will continue processing instructions if DPI_REQ_ERR_RSP_EN[EN] is set.

Cavium Confidential For Bowen Yang ALU 02/27/2014

DPI Instruction Queue Reset Error Register
DPI_REQ_ERR_RST

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	QERR	R/W1C/H	0x0	0x0	Indicates which instruction queue dropped an instruction because the source or destination was in reset. Software must clear the bit before the corresponding instruction queue will continue processing instructions if DPI_REQ_ERR_RST_EN[EN] is set.

DPI Instruction Queue ErrorResponse Enable Register
DPI_REQ_ERR_RSP_EN

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	EN	R/W	0xFF	0xFF	Indicates which instruction queues should stop dispatching instructions when an ErrorResponse is received from the I/O subsystem.

DPI Instruction Queue Reset Error Enable Register
DPI_REQ_ERR_RST_EN

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	EN	R/W	0xFF	0xFF	Indicates which instruction queues should stop dispatching instructions when an instruction is dropped because the source or destination port is in reset.

DPI Packet Error Response Register
DPI_PKT_ERR_RSP

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	PKTERR	R/W1C/H	0	0	Indicates that an ErrorResponse was received from the I/O subsystem.

Cavium Confidential For Bowen Yang ALU 02/27/2014

DPI DMA-Engine Queue Enable Registers
DPI_DMA_ENG(0..5)_EN

These registers provide control for the DMA engines. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	QEN	R/W	0xFF	0xFF	Instruction queue enable. This field specifies which of eight logical instruction queues can be serviced by the DMA engine. 1 = allow service, 0 = prevent service Setting QEN = 0x0 effectively disables the engine. When DPI_DMA_CONTROL[PKT_EN] = 1, then DPI_DMA_ENG5_EN [QEN] must be zero. When DPI_DMA_CONTROL[PKT_EN1] = 1, DPI_DMA_ENG4_EN[QEN] must be zero.

DPI DMA Doorbell Registers
DPI_DMA(0..7)_DBELL

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	DBELL	WO/H	0x0	—	Doorbell value. The value written to this register is added to the number of eight-byte words to be read and processed for the low-priority DMA queue.

DPI DMA Instruction-Buffer Starting-Address Registers
DPI_DMA(0..7)_IBUFF_SADDR

These registers provide the address to start reading instructions for the eight DMA instruction queues. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:62>	—	RAZ	—	—	Reserved
<61:48>	CSIZE	R/W	0x40	—	The size in 8-byte words of the DMA instruction chunk. This value should only be written at known times in order to prevent corruption of the instruction queue. The minimum CSIZE is 16 (one cache block).
<47:41>	—	RAZ	—	—	Reserved
<40>	IDLE	RO/H	1	—	Request queue is idle. When asserted, the associated request queue is idle.
<39:36>	—	RAZ	—	—	Reserved
<35:7>	SADDR	R/W/H	0x0	—	Starting address. The 128-byte aligned starting or chunk address. SADDR is address bit <35:7> of the starting instructions address. When new chunks are fetched by the hardware, SADDR is updated to reflect the address of the current chunk. A write to SADDR resets both the queue's doorbell (DPI_DMA(0..7)_COUNTS[DBELL) and its tail pointer (DPI_DMA(0..7)_NADDR[ADDR]).
<6:0>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang ALU 6/6/2014

DPI DMA Instruction Counts Registers
DPI_DMA(0..7)_COUNTS

These registers provide values for determining the number of instructions in the local instruction FIFO. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:39>	—	RAZ	—	—	Reserved
<38:32>	FCNT	RO/H	0x0	0x0	FIFO count. Specifies the number of words in the instruction FIFO locally cached within DPI.
<31:0>	DBELL	RO/H	0x0	0x0	Doorbell. Specifies the number of available words of instructions to read.

DPI DMA Next Ichunk Address Registers
DPI_DMA(0..7)_NADDR

These registers provide the L2C addresses to read the next Ichunk data. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:36>	—	RAZ	—	—	Reserved
<35:0>	ADDR	RO/H	0x0	—	Address. Provides the next L2C address to read instructions.

DPI DMA Request State Bank 0 Registers
DPI_DMA(0..7)_REQBNK0

These registers provide the current contents of the request state machine, bank 0. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	STATE	RO/H	0x0	—	State. Provides the current contents of the request state machine.

DPI DMA Request State Bank 1 Registers
DPI_DMA(0..7)_REQBNK1

These registers provide the current contents of the request state machine, bank 1. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	STATE	RO/H	0x0	—	State. Provides the current contents of the request state machine.

Cavium Confidential For Bowen Yang ALU 02/27/2014

DPI IOI Configuration Register
DPI_NCB0_CFG

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	MOLR	R/W	0x20	0x20	Max outstanding load requests. Limits the number of outstanding load requests on the I/O Interconnect. This value can range from 1 to 32. Setting a value of 0 halts all read traffic to the I/O Interconnect. There are no restrictions on when this value can be changed.

DPI Packet Interrupt Information Register
DPI_PINT_INFO

This register provides DPI packet interrupt information. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved
<13:8>	IINFO	RO/H	0x0	—	Packet instruction doorbell count overflow information.
<7:6>	—	RAZ	—	—	Reserved
<5:0>	SINFO	RO/H	0x0	—	Packet scatter list doorbell count overflow information.

Cavium Confidential For Bowen Yang ALU 02/27/2014

DPI Instruction Queue Skip Completion Register DPI_REQ_ERR_SKIP_COMP

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:24>	—	RAZ	—	—	Reserved
<23:16>	EN_RST	R/W/H	0x0	0x0	Indicates which instruction queue should skip the completion phase once a port reset is detected as indicated by DPI_REQ_ERR_RST. All completions to the effected instruction queue are skipped as long as DPI_REQ_ERR_RSP[QERR<*ique*>] & EN_RSP<*ique*> or DPI_REQ_ERR_RST[QERR<*ique*>] & EN_RST<*ique*> are set.
<15:8>	—	RAZ	—	—	Reserved
<7:0>	EN_RSP	R/W/H	0x0	0x0	Indicates which instruction queue should skip the completion phase once an ErrorResponse is detected as indicated by DPI_REQ_ERR_RSP. All completions to the effected instruction queue are skipped as long as DPI_REQ_ERR_RSP[QERR<*ique*>] & EN_RSP<*ique*> or DPI_REQ_ERR_RST[QERR<*ique*>] & EN_RST<*ique*> are set.

DPI DMA Engine Buffer Registers DPI_ENG(0..5)_BUF

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:37>	—	RAZ	—	—	Reserved
<36:32>	COMPBLKS	RO	—	—	Computed engine block size
<31:9>	—	RAZ	—	—	Reserved
<8:4>	BASE	RO	—	—	The base address in 512-byte blocks of the DMA engine FIFO.
<3:0>	BLKS	R/W	0x2	—	The size of the DMA engine FIFO. Legal values are 0–10. 0x0 = disabled 0x1 = 0.5KB FIFO 0x2 = 1KB FIFO 0x3 = 1.5KB FIFO 0x4 = 2KB FIFO 0x5 = 2.5KB FIFO 0x6 = 3KB FIFO 0x7 = 3.5KB FIFO 0x8 = 4KB FIFO 0x9 = 6.0KB FIFO 0xA = 8.0KB FIFO **NOTE:** The total amount of storage allocated to the six DPI DMA engines (via DPI_ENG(0..5)_BUF[BLKS]) must not exceed 8 KB.

Cavium Confidential Peer
Bowen Yang
AL U
2/27/2014

DPI SLI Port Configuration Register
DPI_SLI_PRT(0..2)_CFG

This register configures the max read request size, max payload size, and max number of SLI tags in use. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved
<24>	HALT	RO/H	0	0	When set, HALT indicates that the MAC has detected a reset condition. No further instructions that reference the MAC from any instruction queue will be issued until the MAC comes out of reset and HALT is cleared in SLI_CTL_PORT(0..2)[DIS_PORT].
<23:17>	—	RAZ	—	—	Reserved
<16>	RD_MODE	R/W	0	0	Read mode. 0 = Exact read mode. The hardware reads on a four-byte granularity. In this mode, the hardware may break a given read into 3 operations to satisfy PCIe rules. 1 = Block mode. The hardware reads more data than requested in order to minimize the number of operations necessary to complete the operation. The memory region must be memory like.
<15:14>	—	RAZ	—	—	Reserved
<13:8>	MOLR	R/W/H	—	—	Maximum outstanding load requests. Limits the number of outstanding load requests on the port by restricting the number of tags used by the SLI to track load responses. This value can range from 1 to 32 depending on the MAC type and number of lanes. MAC = PCIe / 4 lanes: max is 32 MAC = PCIe / 2 lanes: max is 16 MAC = PCIe / 1 lane: max is 8 Reset value is computed based on the MAC config. Setting MOLR to a value of 0 halts all read traffic to the port. There are no restrictions on when this value can be changed.
<7>	MPS_LIM	R/W	0	0	MAC memory space write requests cannot cross the (naturally-aligned) MPS boundary. When clear, DPI is allowed to issue a MAC memory- space read that crosses the naturally-aligned boundary of size defined by MPS. (DPI will still only cross the boundary when it would eliminate a write by doing so.) When set, DPI will never issue a MAC memory space write that crosses the naturally-aligned boundary of size defined by MPS.
<6:5>	—	RAZ	—	—	Reserved
<4>	MPS	R/W	0x0	0x0	Maximum payload size. 0 = 128B 1 = 256B The MPS size must not exceed the size selected by PCIE*_CFG030[MPS].
<3>	MRRS_LIM	R/W	0	0	MAC memory space read requests cannot cross the (naturally-aligned) MRRS boundary. When clear, DPI is allowed to issue a MAC memory-space read that crosses the naturally-aligned boundary of size defined by MRRS. (DPI will still only cross the boundary when it would eliminate a read by doing so.) When set, DPI will never issue a MAC memory space read that crosses the naturally-aligned boundary of size defined by MRRS.

Cavium Confidential For Beverly Yang ALD 02/27/2014

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<2>	—	RAZ	—	—	Reserved
<1:0>	MRRS	R/W	0x0	0x0	Maximum read-request size. 0x0 = 128B 0x1 = 256B 0x2 = 512B 0x3 = 1024B The MRRS size must not exceed the size selected by PCIE*_CFG030[MRRS].

DPI SLI Port Error Address Registers
DPI_SLI_PRT(0..2)_ERR

This register logs the address associated with the reported SLI error response. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:3>	ADDR	RO	0x0	—	Address of the failed load request. Address is locked along with the DPI_SLI_PRT(0..2)_ERR_INFO register. See the DPI_SLI_PRT(0..2)_ERR_INFO[LOCK] description for further information.
<2:0>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang ALU 02/27/2014

DPI SLI MAC Port Error Info Registers
DPI_SLI_PRT(0..2)_ERR_INFO

This register logs information associated with the reported SLI error response. See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8>	LOCK	RO/H	0x0	—	DPI_SLI_PRT(0..2)_ERR and DPI_SLI_PRT(0..2)_ERR_INFO have captured and locked contents. When CN71XX first detects an ErrorResponse, the TYPE, REQQ, and ADDR of the error is saved and an internal lock state is set so the data associated with the initial error is preserved. Subsequent ErrorResponses will optionally raise an interrupt, but will not modify the TYPE, REQQ, or ADDR fields until the internal lock state is cleared. Software can clear the internal lock state by writing a 1 to the appropriate bit in either DPI_REQ_ERR_RSP or DPI_PKT_ERR_RSP depending on the TYPE field. Once the internal lock state is cleared, the next ErrorResponse will set the TYPE, REQQ, and ADDR for the new transaction.
<7:5>	—	RAZ	—	—	Reserved
<4>	TYPE	RO/H	0x0	—	Type of transaction that caused the ErrorResponse. 0 = DMA Instruction 1 = PKT Instruction
<3>	—	RAZ	—	—	Reserved
<2:0>	REQQ	RO/H	0x0	—	Request queue that made the failed load request.

DPI Information Register
DPI_INFO_REG

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:4>	FFP	R/W1C/H	0x0	0x0	Force forward progress indicator.
<3:2>	—	RAZ	—	—	Reserved
<1>	NCB	R/W1C/H	0	0	NCB register access. This interrupt fires in normal operation when software reads a DPI register through the IOI interface.
<0>	RSL	R/W1C/H	0	0	RSL register access. This interrupt fires in normal operation when software reads a DPI register through the RSL interface.

Cavium Confidential For Bowen Yang ALU 02/27/2014

DPI DMA In-Flight Register
DPI_DMA(0..7)_IFLIGHT

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved
<2:0>	CNT	R/W/H	0x6	0x6	The number of instructions from a given queue that can be in flight to the DMA engines at a time. Reset value matches the number of DMA engines.

DPI DMA Error Response Status Register
DPI_DMA(0..7)_ERR_RSP_STATUS

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	STATUS	R/W1C/H	0x0	0x0	QUE captures the ErrorResponse status of the last 6 instructions for each instruction queue. STATUS<5> represents the status for first instruction in instruction order while STATUS<0> represents the last or most recent instruction. If STATUS<*n*> is set, then the nth instruction in the given queue experienced an ErrorResponse. Otherwise, it completed normally.

DPI DMA Per-Core Instruction Completion Counter Register
DPI_DMA_PP(0..1)_CNT

See Table 12–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	CNT	R/W/H	0x0	—	DPI DMA per-core instruction completion counter. The counter is incremented according to conditions described below and is decremented by values written to this field. A CNT of nonzero causes an interrupt in CIU_SUM1_PP*_IPX. When DPI_DMA_CONTROL[DICI_MODE] is enabled, every DMA instruction that has the WQP=0 and a PTR value of 1 to 0x4 will increment DPI_DMA_PP*_CNT value–1 counter. Instructions with WQP=0 and PTR values higher than 0x3F will still send a zero byte write. Hardware reserves the values 0x5 to 0x3F for future use and will treat them as a PTR of 0 and do nothing.

Cavium Confidential For Boyen Yang 6/12/2014

12.11 SLI Registers

Most SLI registers can be accessed either from a CN71XX-internal (i.e. NCB) address or from an external device via CN71XX's BAR0 (refer to Section 22.2.2). Table 12–5 lists the 64-bit SLI registers.

Table 12–5 SLI Registers

Register	Address (CN71XX internal)	Address (External)	CSR Type	Detailed Description
SLI_WIN_WR_ADDR	—	0x0000000000000000	PEXP	See page 696
SLI_WIN_RD_ADDR	—	0x0000000000000010	PEXP	See page 696
SLI_WIN_WR_DATA	—	0x0000000000000020	PEXP	See page 697
SLI_WIN_WR_MASK	—	0x0000000000000030	PEXP	See page 697
SLI_WIN_RD_DATA	—	0x0000000000000040	PEXP	See page 697
SLI_CTL_PORT0 SLI_CTL_PORT1 SLI_CTL_PORT2	0x00011F0000010050 0x00011F0000010060 0x00011F0000010070	0x0000000000000050 0x0000000000000060 0x0000000000000070	PEXP_NCB	See page 698
SLI_MEM_ACCESS_SUBID12 SLI_MEM_ACCESS_SUBID13 . . . SLI_MEM_ACCESS_SUBID27	0x00011F00000100E0 0x00011F00000100F0 . . . 0x00011F00000101D0	0x00000000000000E0 0x00000000000000F0 . . . 0x00000000000001D0	PEXP_NCB	See page 699
SLI_WINDOW_CTL	0x00011F00000102E0	0x00000000000002E0	PEXP_NCB	See page 699
SLI_MEM_ACCESS_CTL	0x00011F00000102F0	0x00000000000002F0	PEXP_NCB	See page 700
SLI_INT_SUM	0x00011F0000010330	0x0000000000000330	PEXP_NCB	See page 701
SLI_INT_ENB_PORT0 SLI_INT_ENB_PORT1	0x00011F0000010340 0x00011F0000010350	0x0000000000000340 0x0000000000000350	PEXP_NCB	See page 703
SLI_SCRATCH_1 SLI_SCRATCH_2	0x00011F00000103C0 0x00011F00000103D0	0x00000000000003C0 0x00000000000003D0	PEXP_NCB	See page 705
SLI_DMA0_INT_LEVEL SLI_DMA1_INT_LEVEL	0x00011F00000103E0 0x00011F00000103F0	0x00000000000003E0 0x00000000000003F0	PEXP_NCB	See page 705
SLI_DMA0_CNT SLI_DMA1_CNT	0x00011F0000010400 0x00011F0000010410	0x0000000000000400 0x0000000000000410	PEXP_NCB	See page 705
SLI_DMA0_TIM SLI_DMA1_TIM	0x00011F0000010420 0x00011F0000010430	0x0000000000000420 0x0000000000000430	PEXP_NCB	See page 705
SLI_CTL_STATUS	0x00011F0000010570	0x0000000000000570	PEXP_NCB	See page 706
SLI_BIST_STATUS	0x00011F0000010580	0x0000000000000580	PEXP_NCB	See page 706
SLI_DATA_OUT_CNT	0x00011F00000105F0	0x00000000000005F0	PEXP_NCB	See page 707
SLI_LAST_WIN_RDATA0 SLI_LAST_WIN_RDATA1	0x00011F0000010600 0x00011F0000010610	0x0000000000000600 0x0000000000000610	PEXP_NCB	See page 707
SLI_STATE1	0x00011F0000010620	0x0000000000000620	PEXP_NCB	See page 707
SLI_STATE2	0x00011F0000010630	0x0000000000000630	PEXP_NCB	See page 708
SLI_STATE3	0x00011F0000010640	0x0000000000000640	PEXP_NCB	See page 708
SLI_PCIE_MSI_RCV_B1	0x00011F0000010650	0x0000000000000650	PEXP_NCB	See page 708
SLI_PCIE_MSI_RCV_B2	0x00011F0000010660	0x0000000000000660	PEXP_NCB	See page 709
SLI_PCIE_MSI_RCV_B3	0x00011F0000010670	0x0000000000000670	PEXP_NCB	See page 709
SLI_LAST_WIN_RDATA2	0x00011F00000106C0	0x00000000000006C0	PEXP_NCB	See page 707
SLI_LAST_WIN_RDATA3	0x00011F00000106D0	0x00000000000006D0	PEXP_NCB	See page 707
SLI_PKT0_OUT_SIZE SLI_PKT1_OUT_SIZE . . . SLI_PKT31_OUT_SIZE	0x00011F0000010C00 0x00011F0000010C10 . . . 0x00011F0000010DF0	0x0000000000000C00 0x0000000000000C10 . . . 0x0000000000000DF0	PEXP_NCB	See page 709
SLI_PKT_INSTR_ENB	0x00011F0000011000	0x0000000000001000	PEXP_NCB	See page 709
SLI_PKT_OUT_ENB	0x00011F0000011010	0x0000000000001010	PEXP_NCB	See page 710
SLI_PKT_INSTR_SIZE	0x00011F0000011020	0x0000000000001020	PEXP_NCB	See page 710
SLI_PKT_SLIST_ROR	0x00011F0000011030	0x0000000000001030	PEXP_NCB	See page 710
SLI_PKT_SLIST_NS	0x00011F0000011040	0x0000000000001040	PEXP_NCB	See page 710
SLI_PKT_SLIST_ES	0x00011F0000011050	0x0000000000001050	PEXP_NCB	See page 711
SLI_PKT_IPTR	0x00011F0000011070	0x0000000000001070	PEXP_NCB	See page 711

Cavium Confidential For 2/17/2014

Table 12–5 SLI Registers (Continued)

Register	Address (CN71XX internal)	Address (External)	CSR Type	Detailed Description
SLI_PKT_DPADDR	0x00011F0000011080	0x0000000000001080	PEXP_NCB	See page 711
SLI_PKT_DATA_OUT_ROR	0x00011F0000011090	0x0000000000001090	PEXP_NCB	See page 712
SLI_PKT_DATA_OUT_NS	0x00011F00000110A0	0x00000000000010A0	PEXP_NCB	See page 712
SLI_PKT_DATA_OUT_ES	0x00011F00000110B0	0x00000000000010B0	PEXP_NCB	See page 712
SLI_PKT_OUT_BMODE	0x00011F00000110D0	0x00000000000010D0	PEXP_NCB	See page 713
SLI_PKT_PCIE_PORT	0x00011F00000110E0	0x00000000000010E0	PEXP_NCB	See page 713
SLI_PKT_INT_LEVELS	0x00011F0000011120	0x0000000000001120	PEXP_NCB	See page 713
SLI_PKT_CNT_INT	0x00011F0000011130	0x0000000000001130	PEXP_NCB	See page 713
SLI_PKT_TIME_INT	0x00011F0000011140	0x0000000000001140	PEXP_NCB	See page 714
SLI_PKT_CNT_INT_ENB	0x00011F0000011150	0x0000000000001150	PEXP_NCB	See page 714
SLI_PKT_TIME_INT_ENB	0x00011F0000011160	0x0000000000001160	PEXP_NCB	See page 714
SLI_PKT_INPUT_CONTROL	0x00011F0000011170	0x0000000000001170	PEXP_NCB	See page 714
SLI_PKT_OUTPUT_WMARK	0x00011F0000011180	0x0000000000001180	PEXP_NCB	See page 716
SLI_PKT_INSTR_RD_SIZE	0x00011F00000111A0	0x00000000000011A0	PEXP_NCB	See page 716
SLI_PKT_IN_PCIE_PORT	0x00011F00000111B0	0x00000000000011B0	PEXP_NCB	See page 716
SLI_PKT_PORT_IN_RST	0x00011F00000111F0	0x00000000000011F0	PEXP_NCB	See page 717
SLI_PKT_IN_INSTR_COUNTS	0x00011F0000011200	0x0000000000001200	PEXP_NCB	See page 717
SLI_PKT_IN_BP	0x00011F0000011210	0x0000000000001210	PEXP_NCB	See page 717
SLI_PKT_CTL	0x00011F0000011220	0x0000000000001220	PEXP_NCB	See page 717
SLI_PKT0_SLIST_BADDR SLI_PKT1_SLIST_BADDR . . . SLI_PKT31_SLIST_BADDR	0x00011F0000011400 0x00011F0000011410 . . . 0x00011F00000115F0	0x0000000000001400 0x0000000000001410 . . . 0x00000000000015F0	PEXP_NCB	See page 718
SLI_PKT0_SLIST_BAOFF_DBELL SLI_PKT1_SLIST_BAOFF_DBELL . . . SLI_PKT31_SLIST_BAOFF_DBELL	0x00011F0000011800 0x00011F0000011810 . . . 0x00011F00000119F0	0x0000000000001800 0x0000000000001810 . . . 0x00000000000019F0	PEXP_NCB	See page 718
SLI_PKT0_SLIST_FIFO_RSIZE SLI_PKT1_SLIST_FIFO_RSIZE . . . SLI_PKT31_SLIST_FIFO_RSIZE	0x00011F0000011C00 0x00011F0000011C10 . . . 0x00011F0000011DF0	0x0000000000001C00 0x0000000000001C10 . . . 0x0000000000001DF0	PEXP_NCB	See page 718
SLI_PKT_IN_DONE0_CNTS SLI_PKT_IN_DONE1_CNTS . . . SLI_PKT_IN_DONE31_CNTS	0x00011F0000012000 0x00011F0000012010 . . . 0x00011F00000121F0	0x0000000000002000 0x0000000000002010 . . . 0x00000000000021F0	PEXP_NCB	See page 718
SLI_PKT0_CNTS SLI_PKT1_CNTS . . . SLI_PKT31_CNTS	0x00011F0000012400 0x00011F0000012410 . . . 0x00011F00000125F0	0x0000000000002400 0x0000000000002410 . . . 0x00000000000025F0	PEXP_NCB	See page 719
SLI_PKT0_INSTR_BADDR SLI_PKT1_INSTR_BADDR . . . SLI_PKT31_INSTR_BADDR	0x00011F0000012800 0x00011F0000012810 . . . 0x00011F00000129F0	0x0000000000002800 0x0000000000002810 . . . 0x00000000000029F0	PEXP_NCB	See page 719
SLI_PKT0_INSTR_BAOFF_DBELL SLI_PKT1_INSTR_BAOFF_DBELL . . . SLI_PKT31_INSTR_BAOFF_DBELL	0x00011F0000012C00 0x00011F0000012C10 . . . 0x00011F0000012DF0	0x0000000000002C00 0x0000000000002C10 . . . 0x0000000000002DF0	PEXP_NCB	See page 719
SLI_PKT0_INSTR_FIFO_RSIZE SLI_PKT1_INSTR_FIFO_RSIZE . . . SLI_PKT31_INSTR_FIFO_RSIZE	0x00011F0000013000 0x00011F0000013010 . . . 0x00011F00000131F0	0x0000000000003000 0x0000000000003010 . . . 0x00000000000031F0	PEXP_NCB	See page 720
SLI_PKT0_INSTR_HEADER SLI_PKT1_INSTR_HEADER . . . SLI_PKT31_INSTR_HEADER	0x00011F0000013400 0x00011F0000013410 . . . 0x00011F00000135F0	0x0000000000003400 0x0000000000003410 . . . 0x00000000000035F0	PEXP_NCB	See page 721

Cavium Confidential PCN
Bowei Yang
10/21/2014

Table 12–5 SLI Registers (Continued)

Register	Address (CN71XX internal)	Address (External)	CSR Type	Detailed Description
SLI_PKT0_IN_BP SLI_PKT1_IN_BP . . . SLI_PKT31_IN_BP	0x00011F0000013800 0x00011F0000013810 . . . 0x00011F00000139F0	0x0000000000003800 0x0000000000003810 . . . 0x00000000000039F0	PEXP_NCB	See page 722
SLI_MSI_RCV0	0x00011F0000013C10	0x0000000000003C10	PEXP_NCB	See page 722
SLI_MSI_RCV1	0x00011F0000013C20	0x0000000000003C20	PEXP_NCB	See page 722
SLI_MSI_RCV2	0x00011F0000013C30	0x0000000000003C30	PEXP_NCB	See page 722
SLI_MSI_RCV3	0x00011F0000013C40	0x0000000000003C40	PEXP_NCB	See page 723
SLI_MSI_ENB0	0x00011F0000013C50	0x0000000000003C50	PEXP_NCB	See page 723
SLI_MSI_ENB1	0x00011F0000013C60	0x0000000000003C60	PEXP_NCB	See page 723
SLI_MSI_ENB2	0x00011F0000013C70	0x0000000000003C70	PEXP_NCB	See page 723
SLI_MSI_ENB3	0x00011F0000013C80	0x0000000000003C80	PEXP_NCB	See page 723
SLI_MSI_WR_MAP	0x00011F0000013C90	0x0000000000003C90	PEXP_NCB	See page 724
SLI_MSI_RD_MAP	0x00011F0000013CA0	0x0000000000003CA0	PEXP_NCB	See page 724
SLI_PCIE_MSI_RCV	0x00011F0000013CB0	0x0000000000003CB0	PEXP_NCB	See page 724
SLI_INT_ENB_CIU	0x00011F0000013CD0	0x0000000000003CD0	PEXP_NCB	See page 725
SLI_MSI_W1C_ENB0	0x00011F0000013CF0	0x0000000000003CF0	PEXP_NCB	See page 726
SLI_MSI_W1C_ENB1	0x00011F0000013D00	0x0000000000003D00	PEXP_NCB	See page 726
SLI_MSI_W1C_ENB2	0x00011F0000013D10	0x0000000000003D10	PEXP_NCB	See page 726
SLI_MSI_W1C_ENB3	0x00011F0000013D20	0x0000000000003D20	PEXP_NCB	See page 726
SLI_MSI_W1S_ENB0	0x00011F0000013D30	0x0000000000003D30	PEXP_NCB	See page 727
SLI_MSI_W1S_ENB1	0x00011F0000013D40	0x0000000000003D40	PEXP_NCB	See page 727
SLI_MSI_W1S_ENB2	0x00011F0000013D50	0x0000000000003D50	PEXP_NCB	See page 727
SLI_MSI_W1S_ENB3	0x00011F0000013D60	0x0000000000003D60	PEXP_NCB	See page 727
SLI_MAC_CREDIT_CNT	0x00011F0000013D70	0x0000000000003D70	PEXP_NCB	See page 728
SLI_S2M_PORT0_CTL SLI_S2M_PORT1_CTL SLI_S2M_PORT2_CTL	0x00011F0000013D80 0x00011F0000013D90 0x00011F0000013DA0	0x0000000000003D80 0x0000000000003D90 0x0000000000003DA0	PEXP_NCB	See page 729
SLI_MAC_NUMBER	—	0x0000000000003E00	PEXP	See page 730
SLI_MAC_CREDIT_CNT2	0x00011F0000013E10	0x0000000000003E10	PEXP	See page 730

Cavium Confidential
Bowen Yang
ALU
02/27/2014

SLI Window Write Address Register
SLI_WIN_WR_ADDR

This register contains the address to be written to when a write operation is started by writing the SLI_WIN_WR_DATA register. This register should NOT be used to write SLI_* registers. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:49>	—	RAZ	—	—	Reserved
<48>	IOBIT	RAZ	0	0	A 1 or 0 can be written here, but this will always read as 0.
<47:3>	WR_ADDR	R/W	0x0	—	The address that is written to when the SLI_WIN_WR_DATA register is written. [47:40] = NCB_ID [39:3] = Address When [47:43] specifies SLI and [42:40] = 0x0, bits [39:0] have the following meanings: [39:32] = not used [31:24] = RSL_ID [23:3] = RSL register offset [2:0] = not used
<2:0>	—	RAZ	—	—	Reserved

SLI Window Read Address Register
SLI_WIN_RD_ADDR

This register contains the address to be read when the SLI_WIN_RD_DATA register is read. This register should NOT be used to read SLI_* registers. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:51>	—	RAZ	—	—	Reserved
<50:49>	LD_CMD	R/W	0x3	—	The load command sent with the read. 0x3 = Load 8 bytes 0x1 = Load 2 bytes 0x2 = Load 4 bytes 0x0 = Load 1 bytes
<48>	IOBIT	R/W	0	0	A 1 or 0 can be written here, but will not be used in address generation.
<47:0>	RD_ADDR	R/W	0x0	—	The address to be read. [47:40] = NCB_ID [39:3] = Address When [47:43] specifies SLI and [42:40] = 0x0, bits [39:0] have the following meanings: [39:32] = not used [31:24] = RSL_ID [23:0] = RSL register offset

Cavium Confidential For Bowen Yang ALI 02/27/2014

SLI Window Write Data Register
SLI_WIN_WR_DATA

This register contains the data to write to the address located in the SLI_WIN_WR_ADDR register. Writing the least-significant byte of this register causes a write operation to take place. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	WR_DATA	R/W	0x0	—	The data to be written. Whenever the LSB of this register is written, the window write operation takes place.

SLI Window Write Mask Register
SLI_WIN_WR_MASK

This register contains the mask for the data in the SLI_WIN_WR_DATA Register. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved.
<7:0>	WR_MASK	R/W	0xFF	0x0	The byte mask for the data to be written. When a bit is 1, the corresponding byte will be written. The values of this field must be contiguous and for one-, two-, four-, or eight-byte operations and aligned to operation size. A value of 0x0 produces unpredictable results.

SLI Window Read Data Register
SLI_WIN_RD_DATA

This register contains the address to be read when the SLI_WIN_RD_DATA register is read. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	RD_ADDR	RO/H	0x0	—	The read data

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI Control Port Registers
SLI_CTL_PORT(0..2)

These registers contains control information for access for Port0 or Port1. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:22>	—	RAZ	—	—	Reserved
<21>	INTD	RO/H	1	1	When 0, the INTD wire is asserted before mapping.
<20>	INTC	RO/H	1	1	When 0, the INTC wire is asserted before mapping.
<19>	INTB	RO/H	1	1	When 0, the INTB wire is asserted before mapping.
<18>	INTA	RO/H	1	1	When 0, the INTA wire is asserted before mapping.
<17>	DIS_PORT	R/W1C/H	0	0	When set, the output to the MAC is disabled. This occurs when the MAC reset line transitions from de-asserted to asserted. Writing a 1 to this location clears this condition when the MAC is no longer in reset and the output to the MAC is at the beginning of a transfer.
<16>	WAITL_COM	R/W	0	—	When set to 1, causes the SLI to wait for a commit from the L2C before sending additional completions to the L2C from the MAC. Set this for more conservative behavior. Clear this for more aggressive, higher-performance behavior.
<15:14>	INTD_MAP	R/W	0x3	0x3	Maps INTD: 00 = map to INTA 10 = map to INTC 01 = map to INTB 11 = map to INTD
<13:12>	INTC_MAP	R/W	0x2	0x2	Maps INTC: 00 = map to INTA 10 = map to INTC 01 = map to INTB 11 = map to INTD
<11:10>	INTB_MAP	R/W	0x1	0x1	Maps INTB: 00 = map to INTA 10 = map to INTC 01 = map to INTB 11 = map to INTD
<9:8>	INTA_MAP	R/W	0x0	0x0	Maps INTA: 00 = map to INTA 10 = map to INTC 01 = map to INTB 11 = map to INTD
<7>	CTLP_RO	R/W	0	1	Relaxed ordering enable for completion TLPS
<6>	—	R/W	0	0	Reserved
<5>	PTLP_RO	R/W	0	1	Relaxed ordering enable for posted TLPS
<4:1>	—	RAZ	—	—	Reserved
<0>	WAIT_COM	R/W	0	0	Wait for commit. When set to 1, causes the SLI to wait for a commit from the L2C before sending additional stores to the L2C from the MAC. The SLI requests a commit on the last store if more than one STORE operation is required on the IOI. Most applications will not notice a difference, so this bit should not be set. Setting the bit is more conservative on ordering, lower performance.

Cavium Confidential For Bowen Yang ALU 12/27/2014

SLI Memory Access SubID Registers
SLI_MEM_ACCESS_SUBID(12..27)

These registers contains address index and control bits for access to memory from cores. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:43>	—	RAZ	—	—	Reserved
<42>	ZERO	R/W	0	0	Causes all byte read operations to be zero-length read operations. Returns 0s to the EXEC for all read data.
<41:39>	PORT	R/W	0x0	—	Physical MAC Port that read/write operations to this subID are sent to. Must be ≤ 1, as there are only two ports present.
<38>	NMERGE	R/W	0	0	When set, no merging is allowed in this window.
<37:36>	ESR	R/W	0x0	—	Endian swap for read operations. ES<1:0> for read operations to this subID. ES<1:0> is the endian-swap attribute for these MAC memory space read operations.
<35:34>	ESW	R/W	0x0	—	Endian swap for write operations. ES<1:0> for write operations to this subID. ES<1:0> is the endian-swap attribute for these MAC memory space write operations.
<33:32>	WTYPE	R/W	0x0	—	Write type. ADDRTYPE<1:0> for write operations to this subID. • ADDRTYPE<0> is the relaxed-order attribute • ADDRTYPE<1> is the no-snoop attribute
<31:30>	RTYPE	R/W	0x0	—	Read type. ADDRTYPE<1:0> for read operations to this subID. • ADDRTYPE<0> is the relaxed-order attribute • ADDRTYPE<1> is the no-snoop attribute
<29:0>	BA	R/W	0x0	—	Bus address. Address bits<63:34> for read/write operations that use this subID.

SLI Window Control Register
SLI_WINDOW_CTL

Access to register space on the IOI (caused by window read/write operations) waits for a period of time specified by this register before timing out. Because a window access can access the RML, which has a fixed timeout of 0xFFFF coprocessor-clock cycles, the value of this register should be set to a minimum of 0x200000 to ensure that a timeout to an RML register occurs on the RML 0xFFFF timer before the timeout for a BAR0 access from the MAC. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	TIME	R/W	0x0	0x200000	Time to wait. The number of coprocessor-clock cycles to wait for a BAR 0 access to complete on the IOI before timing out. A value of 0x0 causes no timeouts. A minimum value of 0x200000 should be used when this register is not set to 0x0.

Cavium Confidential For Bowen Yang ALU 02/27/14

SLI Memory Access Control Register
SLI_MEM_ACCESS_CTL

This register contains control signals for access to the MAC address space. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved
<13:10>	MAX_WORD	R/W	0x0	0x0	Maximum number of words. Specifies the maximum number of words to merge into a single write operation from the cores to the MAC. Legal values are 1 to 16, with 0 treated as 16.
<9:0>	TIMER	R/W	0x0	0x32	Merge timer. When the SLI starts a core-to-MAC write, TIMER specifies the maximum wait, in coprocessor-clock cycles, to merge additional write operations from the cores into one large write. The values for this field range from 1 to 1024, with 0 treated as 1024.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

SLI Interrupt Summary Register
SLI_INT_SUM

The fields in this register are set when a corresponding interrupt condition occurs. Interrupts are generated when the corresponding enable bit is set in SLI_INT_ENB_PORT0/1 OR SLI_INT_ENB_CIU. Write 1 to clear. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:61>	—	RAZ	—	—	Reserved.
<60>	ILL_PAD	R/W1C/H	0	0	Set when a BAR0 address R/W falls into the address range of the packet CSR, but for an unused address.
<59>	—	RAZ	—	—	Reserved.
<58>	SPRT2_ERR	R/W1C/H	0	0	SLI port 2 error. When an error response is received on SLI port 2, this bit is set.
<57>	SPRT1_ERR	R/W1C/H	0	0	SLI port 1 error. When an error response is received on SLI port 1, this bit is set.
<56>	SPRT0_ERR	R/W1C/H	0	0	SLI port 0 error. When an error response is received on SLI port 0, this bit is set.
<55>	PINS_ERR	R/W1C/H	0	0	Packet instruction read error. When a read error occurs on a packet instruction, this bit is set.
<54>	POP_ERR	R/W1C/H	0	0	Packet scatter pointer pair error. When a read error occurs on a packet scatter pointer pair, this bit is set.
<53>	PDI_ERR	R/W1C/H	0	0	Packet data read error. When a read error occurs on a packet data read, this bit is set.
<52>	PGL_ERR	R/W1C/H	0	0	Packet gather list read error. When a read error occurs on a packet gather list read, this bit is set.
<51>	PIN_BP	R/W1C/H	0	0	Watermark exceeded. Packet input count has exceeded the WMARK. See SLI_PKT_IN_BP.
<50>	POUT_ERR	R/W1C/H	0	0	PKO packet data error. Set when PKO sends packet data with the error bit set.
<49>	PSLDBOF	R/W1C/H	0	0	Packet scatter list doorbell count overflowed. Which doorbell can be found in DPI_PINT_INFO[PSLDBOF].
<48>	PIDBOF	R/W1C/H	0	0	Packet instruction doorbell count overflowed. Which doorbell can be found in DPI_PINT_INFO[PIDBOF].
<47:38>	—	RAZ	—	—	Reserved.
<37:36>	DTIME	RO/H	0x0	0x0	Whenever SLI_DMA*x*_CNT[CNT] is not 0, the SLI_DMA*x*_TIM[TIM] timer increments every SLI clock. DTIME<*x*> is set whenever SLI_DMA*x*_TIM[TIM] > SLI_DMA*x*_INT_LEVEL[TIME]. DTIME<*x*> is normally cleared by clearing SLI_DMA*x*_CNT[CNT] (which also clears SLI_DMA*x*_TIM[TIM]).
<35:34>	DCNT	RO/H	0x0	0x0	DCNT<*x*> is set whenever SLI_DMA*x*_CNT[CNT] > SLI_DMA*x*_INT_LEVEL[CNT]. DCNT<*x*> is normally cleared by decreasing SLI_DMA*x*_CNT[CNT].
<33:32>	DMAFI	R/W1C/H	0x0	0x0	DMA set forced interrupts.
<31:30>	—	RAZ	—	—	Reserved.
<29>	MAC2_INT	RO/H	0	0	Interrupt from MAC2. See PEM2_INT_SUM (enabled by PEM2_INT_ENB_INT).
<28>	MIO_INT2	RO/H	0	0	Interrupt from MIO for PORT 2. See CIU_INT32_SUM0, CIU_INT_SUM1 (enabled by CIU_INT32_EN0, CIU_INT32_EN1).
<27:24>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang ANU 2017-2-14

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<23>	M2_UN_WI	R/W1C/H	0	0	Received unsupported N-TLP for window register from MAC 2. This occurs when the window registers are disabled and a window register access occurs.
<22>	M2_UN_B0	R/W1C/H	0	0	Received unsupported N-TLP for Bar 0 from MAC 2. This occurs when the BAR 0 address space is disabled.
<21>	M2_UP_WI	R/W1C/H	0	0	Received unsupported P-TLP for window register from MAC 2. This occurs when the window registers are disabled and a window register access occurs.
<20>	M2_UP_B0	R/W1C/H	0	0	Received unsupported P-TLP for Bar 0 from MAC 2. This occurs when the BAR 0 address space is disabled.
<19>	MAC1_INT	RO/H	0	0	Interrupt from MAC1. See PEM1_INT_SUM (enabled by PEM1_INT_ENB_INT).
<18>	MAC0_INT	RO/H	0	0	Interrupt from MAC0. See PEM0_INT_SUM (enabled by PEM0_INT_ENB_INT).
<17>	MIO_INT1	RO/H	0	0	Interrupt from MIO for PORT 1. See CIU_INT33_SUM0, CIU_INT_SUM1 (enabled by CIU_INT33_EN0, CIU_INT33_EN1).
<16>	MIO_INT0	RO/H	0	0	Interrupt from MIO for PORT 0. See CIU_INT32_SUM0, CIU_INT_SUM1 (enabled by CIU_INT32_EN0, CIU_INT32_EN1).
<15>	M1_UN_WI	R/W1C/H	0	0	Received unsupported N-TLP for window register from MAC 1. This occurs when the window registers are disabled and a window register access occurs.
<14>	M1_UN_B0	R/W1C/H	0	0	Received unsupported N-TLP for Bar 0 from MAC 1. This occurs when the BAR 0 address space is disabled.
<13>	M1_UP_WI	R/W1C/H	0	0	Received unsupported P-TLP for window register from MAC 1. This occurs when the window registers are disabled and a window register access occurs.
<12>	M1_UP_B0	R/W1C/H	0	0	Received unsupported P-TLP for Bar 0 from MAC 1. This occurs when the BAR 0 address space is disabled.
<11>	M0_UN_WI	R/W1C/H	0	0	Received unsupported N-TLP for window register from MAC 0. This occurs when the window registers are disabled and a window register access occurs.
<10>	M0_UN_B0	R/W1C/H	0	0	Received unsupported N-TLP for Bar 0 from MAC 0. This occurs when the BAR 0 address space is disabled.
<9>	M0_UP_WI	R/W1C/H	0	0	Received unsupported P-TLP for window register from MAC 0. This occurs when the window registers are disabled and a window register access occurs.
<8>	M0_UP_B0	R/W1C/H	0	0	Received unsupported P-TLP for Bar 0 from MAC 0. This occurs when the BAR 0 address space is disabled.
<7:6>	—	RAZ	0x0	0x0	Reserved.
<5>	PTIME	RO/H	0	0	Packet timer has an interrupt. The specific rings can be found in SLI_PKT_TIME_INT.
<4>	PCNT	RO/H	0	0	Packet counter has an interrupt. The specific rings can be found in SLI_PKT_CNT_INT.
<3>	IOB2BIG	R/W1C/H	0	0	A requested IOBDMA transfer is too large.
<2>	BAR0_TO	R/W1C/H	0	0	BAR0 R/W to an NCB device did not receive read-data/commit in 0xFFFF coprocessor-clock cycles.
<1>	—	RAZ	0	0	Reserved.
<0>	RML_TO	R/W1C/H	0	0	A read or write transfer did not complete within 0xFFFF coprocessor-clock cycles.

Cavium Confidential for Bowen Yang 02/12/2014

SLI Interrupt Enable Register
SLI_INT_ENB_PORT(0..2)

This register is used to allow the generation of interrupts (MSI/INTA) to the PORT X. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:61>	—	R/W	0x0	0x0	Reserved.
<60>	ILL_PAD	R/W	0	0	Illegal packet csr address.
<59>	SPRT3_ERR	R/W	0	0	Error response received on SLI port 3.
<58>	SPRT2_ERR	R/W	0	0	Error response received on SLI port 2.
<57>	SPRT1_ERR	R/W	0	0	Error response received on SLI port 1.
<56>	SPRT0_ERR	R/W	0	0	Error response received on SLI port 0.
<55>	PINS_ERR	R/W	0	0	Read error during packet instruction fetch.
<54>	POP_ERR	R/W	0	0	Read error during packet scatter pointer fetch.
<53>	PDI_ERR	R/W	0	0	Read error during packet data fetch.
<52>	PGL_ERR	R/W	0	0	Read error during gather list fetch.
<51>	PIN_BP	R/W	0	0	Packet input count exceeded WMARK.
<50>	POUT_ERR	R/W	0	0	Packet out interrupt error from PKO.
<49>	PSLDBOF	R/W	0	0	Packet scatterlist doorbell count overflow.
<48>	PIDBOF	R/W	0	0	Packet instruction doorbell count overflow.
<47:38>	—	R/W	0x0	0x0	Reserved.
<37:36>	DTIME	R/W	0x0	0x0	DMA timer interrupts.
<35:34>	DCNT	R/W	0x0	0x0	DMA count interrupts.
<33:32>	DMAFI	R/W	0x0	0x0	DMA set forced interrupts.
<31:30>	—	R/W	0x0	0x0	Reserved.
<29>	MAC2_INT	R/W	0	0	Enables SLI_INT_SUM[MAC2_INT] to generate an interrupt to the PCIE-Port2 for MSI/INTA. SLI_INT_ENB_PORT2[MAC0_INT] and SLI_INT_ENB_PORT2[MAC1_INT] should NEVER be set.
<28>	MIO_INT2	R/W	0	1	Enables SLI_INT_SUM[MIO_INT2] to generate an interrupt to the PCIE core for MSI/INTA. SLI_INT_ENB_PORT2[MIO_INT2] should NEVER be set.
<27>	M3_UN_WI	R/W	0	0	Reserved.
<26>	M3_UN_B0	R/W	0	0	Reserved.
<25>	M3_UP_WI	R/W	0	0	Reserved.
<24>	M3_UP_B0	R/W	0	0	Reserved.
<23>	M2_UN_WI	R/W	0	1	Enables SLI_INT_SUM[M2_UN_WI] to generate an interrupt to the PCIE core for MSI/INTA.
<22>	M2_UN_B0	R/W	0	1	Enables SLI_INT_SUM[M2_UN_B0] to generate an interrupt to the PCIE core for MSI/INTA.
<21>	M2_UP_WI	R/W	0	1	Enables SLI_INT_SUM[M2_UP_WI] to generate an interrupt to the PCIE core for MSI/INTA.
<20>	M2_UP_B0	R/W	0	1	Enables SLI_INT_SUM[M2_UP_B0] to generate an interrupt to the PCIE core for MSI/INTA.
<19>	MAC1_INT	R/W	0	0	Enables SLI_INT_SUM[MAC1_INT] to generate an interrupt to the PCIE core for MSI/INTA. The value of this bit has NO effect on PCIE Port0. SLI_INT_ENB_PORT0[MAC1_INT] should NEVER be set.

Cavium Confidential For Bowen Yang 02/27/2014

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<18>	MAC0_INT	R/W	0	0	Enables SLI_INT_SUM[MAC0_INT] to generate an interrupt to the PCIE-Port0 for MSI/INTA. The value of this bit has no effect on PCIE Port1. SLI_INT_ENB_PORT0[MAC0_INT] should NEVER be set.
<17>	MIO_INT1	R/W	0	1	Enables SLI_INT_SUM[MIO_INT1] to generate an interrupt to the PCIE core for MSI/INTA. SLI_INT_ENB_PORT0[MIO_INT1] should NEVER be set.
<16>	MIO_INT0	R/W	0	1	Enables SLI_INT_SUM[MIO_INT0] to generate an interrupt to the PCIE core for MSI/INTA. SLI_INT_ENB_PORT1[MIO_INT0] should NEVER be set.
<15>	M1_UN_WI	R/W	0	1	Enables SLI_INT_SUM[M1_UN_WI] to generate an interrupt to the PCIE core for MSI/INTA.
<14>	M1_UN_B0	R/W	0	1	Enables SLI_INT_SUM[M1_UN_B0] to generate an interrupt to the PCIE core for MSI/INTA.
<13>	M1_UP_WI	R/W	0	1	Enables SLI_INT_SUM[M1_UP_WI] to generate an interrupt to the PCIE core for MSI/INTA.
<12>	M1_UP_B0	R/W	0	1	Enables SLI_INT_SUM[M1_UP_B0] to generate an interrupt to the PCIE core for MSI/INTA.
<11>	M0_UN_WI	R/W	0	1	Enables SLI_INT_SUM[M0_UN_WI] to generate an interrupt to the PCIE core for MSI/INTA.
<10>	M0_UN_B0	R/W	0	1	Enables SLI_INT_SUM[M0_UN_B0] to generate an interrupt to the PCIE core for MSI/INTA.
<9>	M0_UP_WI	R/W	0	1	Enables SLI_INT_SUM[M0_UP_WI] to generate an interrupt to the PCIE core for MSI/INTA.
<8>	M0_UP_B0	R/W	0	1	Enables SLI_INT_SUM[M0_UP_B0] to generate an interrupt to the PCIE core for MSI/INTA.
<7:6>	—	R/W	0x0	0x0	Reserved.
<5>	PTIME	R/W	0	1	Enables SLI_INT_SUM[PTIME] to generate an interrupt to the PCIE core for MSI/INTA.
<4>	PCNT	R/W	0	1	Enables SLI_INT_SUM[PCNT] to generate an interrupt to the PCIE core for MSI/INTA.
<3>	IOB2BIG	R/W	0	1	Enables SLI_INT_SUM[IOB2BIG] to generate an interrupt to the PCIE core for MSI/INTA.
<2>	BAR0_TO	R/W	0	1	Enables SLI_INT_SUM[BAR0_TO] to generate an interrupt to the PCIE core for MSI/INTA.
<1>	—	R/W	0	1	Reserved.
<0>	RML_TO	R/W	0	1	Enables SLI_INT_SUM[RML_TO] to generate an interrupt to the PCIE core for MSI/INTA.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI Scratch Register
SLI_SCRATCH_1/2

These registers are general purpose 64-bit scratch registers for software use. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	DATA	R/W	0x0	—	The value in this register is totally software dependent.

SLI DMA0/1 Interrupt Level Register
SLI_DMA0/1_INT_LEVEL

These registers contain the thresholds for DMA count and timer interrupts. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	TIME	R/W	0x0	—	Whenever the SLI_DMA0/1_TIM[TIM] timer exceeds this value, SLI_INT_SUM[DTIME<*x*>] is set. The SLI_DMA0/1_TIM[TIM] timer increments every SLI clock whenever SLI_DMAx_CNT[CNT] ≠ 0, and is cleared when SLI_INT_SUM[DTIME<x>] is written with one.
<31:0>	CNT	R/W	0x0	—	Whenever SLI_DMA0/1_CNT[CNT] exceeds this value, SLI_INT_SUM[DCNT<*x*>] is set.

SLI DMA0/1 Count Register
SLI_DMA0/1_CNT

These registers contain the DMA count values. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	R/W/H	0x0	—	The DMA counter. Writing this field will cause the written value to be subtracted from DMA. Hardware optionally increments this field after it completes an OUTBOUND or EXTERNAL-ONLY DMA instruction. These increments may cause interrupts. Refer to SLI_DMA0/1_INT_LEVEL and SLI_INT_SUM[DCNT,DTIME].

SLI DMA0/1 Timer Register
SLI_DMA0/1_TIM

These registers contain the DMA timer values. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	TIM	RO/H	0x0	—	The DMA timer value. The timer increments when SLI_DMA0/1_CNT[CNT]≠0 and clears when SLI_DMA0/1_CNT[CNT]=0.

Cavium Confidential For Bowen Yang ALU 02/2/2014

SLI Control and Status Register
SLI_CTL_STATUS

This register contains control and status for MAC. Write operations to this register are not ordered with write/read operations to the MAC memory space. To ensure that a write has completed, you must read the register before making an access (i.e. MAC memory space) that requires the value of this register to be updated. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:20>	—	RAZ	—	—	Reserved.
<19:14>	—	R/W	0x20	0x20	Reserved.
<13:8>	P0_NTAGS	R/W/H	0x20	0x20	Number of tags available for outbound TLPs to the MACs. One tag is needed for each outbound TLP that requires a CPL TLP. This field should only be written as part of a reset sequence and before issuing any read operations, CFGs, or I/O transactions from the core(s).
<7:0>	CHIP_REV	RO/H	—	—	CN71XX chip revision level

SLI BIST Status Register
SLI_BIST_STATUS

This register contains results from BIST runs of MAC's memories: 0 = pass (or BIST in progress/never run), 1 = fail. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:31>	—	RAZ	—	—	Reserved.
<30>	N2P0_C	RO	0	0	BIST Status for N2P Port0 cmd.
<29>	N2P0_O	RO	0	0	BIST Status for N2P Port0 data.
<28:27>	—	RAZ	—	—	Reserved.
<26>	CPL_P0	RO	0	0	BIST Status for CPL Port 0.
<25>	CPL_P1	RO	0	0	BIST Status for CPL Port 1.
<24:19>	—	RAZ	—	—	Reserved.
<18>	P2N0_C0	RO	0	0	BIST Status for P2N Port0 C0.
<17>	—	RAZ	—	—	Reserved.
<16>	P2N0_N	RO	0	0	BIST Status for P2N Port0 N.
<15>	P2N0_P0	RO	0	0	BIST Status for P2N Port0 P0.
<14>	—	RAZ	—	—	Reserved.
<13>	P2N1_C0	RO	0	0	BIST Status for P2N Port1 C0.
<12>	—	RAZ	—	—	Reserved.
<11>	P2N1_N	RO	0	0	BIST Status for P2N Port1 N.
<10>	P2N1_P0	RO	0	0	BIST Status for P2N Port1 P0.
<9:6>	—	RAZ	—	—	Reserved.
<5>	DSI1_1	RO	0	0	BIST Status for DSI1 Memory 1.
<4>	DSI1_0	RO	0	0	BIST Status for DSI1 Memory 0.
<3>	DSI0_1	RO	0	0	BIST Status for DSI0 Memory 1.
<2>	DSI0_0	RO	0	0	BIST Status for DSI0 Memory 0.
<1>	MSI	RO	0	0	BIST Status for MSI memory map.
<0>	NCB_CMD	RO	0	0	BIST Status for IOI outbound commands.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI Data Out Count Register
SLI_DATA_OUT_CNT

This register contains the EXEC data out FIFO count and the data unload counter. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:44>	—	RAZ	—	—	Reserved
<43:28>	P1_UCNT	RAZ	0x0	—	FIFO1 unload count. This counter is incremented by 1 every time a word is removed from data out FIFO1, whose count is shown in P1_FCNT.
<27:22>	P1_FCNT	RAZ	0x0	—	FIFO1 data out count. Number of address data words presently buffered in the FIFO1. MACs associated with FIFO1: NONE
<21:6>	P0_UCNT	RO/H	0x0	—	FIFO0 unload count. This counter is incremented by 1 every time a word is removed from data out FIFO0. Whose count is shown in P0_FCNT.
<5:0>	P0_FCNT	RO/H	0x0	—	FIFO0 data out count. Number of address data words presently buffered in the FIFO0. MACs associated with FIFO0: PCIe0, PCIe1, PCIe2

SLI Last Window Read Data Registers
SLI_LAST_WIN_RDATA(0..2)

These registers contain the data from the last initiated window read by MACs 0–2. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	DATA	RO/H	0x0	—	Last window read data.

SLI State 1 Register
SLI_STATE1

This register contains state machines in SLI and is for debug. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:52>	CPL1	RO/H	0x1	—	CPL1 state.
<51:40>	CPL0	RO/H	0x1	—	CPL0 state.
<39>	ARB	RO/H	0	—	ARB state.
<38:0>	CSR	RO/H	0x1	—	CSR state.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI State 2 Register
SLI_STATE2

This register contains state machines in SLI and is for debug. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:56>	—	RAZ	—	—	Reserved.
<55:48>	NNP1	RAZ	0x0	—	NNP1 state.
<47>	—	RAZ	—	—	Reserved.
<46>	RAC	RO/H	1	—	RAC state.
<45:31>	CSM1	RO/H	0x1	—	CSM1 state.
<30:16>	CSM0	RO/H	0x1	—	CSM0 state.
<15:8>	NNP0	RO/H	0x1	—	NNP0 state.
<7:0>	NND	RO/H	0x1	—	NND state.

SLI State 3 Register
SLI_STATE3

This register contains state machines in SLI and is for debug. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:56>	—	RAZ	—	—	Reserved.
<55:41>	PSM1	RO/H	0x1	—	PSM1 state.
<40:26>	PSM0	RO/H	0x1	—	PSM0 state.
<25:13>	NSM1	RO/H	0x1	—	NSM1 state.
<12:0>	NSM0	RO/H	0x1	—	NSM0 state.

SLI MAC MSI Receive Byte 1 Register
SLI_PCIE_MSI_RCV_B1

This register is where MSI write operations are directed from the MAC. This CSR can be used by the PCIe MACs. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved.
<15:8>	INT	R/W	0x0	—	A write to this register results in a bit in one of the SLI_MSI_RCV*n* registers being set. Which bit is set depends on what is previously written using the SLI_MSI_WR_MAP register. If nothing is previously written, the bit set is the reset value of the MAP.
<7:0>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang ALU 02/26/2014

SLI MAC MSI Receive Byte 2 Register
SLI_PCIE_MSI_RCV_B2

This register is where MSI write operations are directed from the MAC. See Table 12–5 for address. This CSR can be used by PCIe MACs.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:24>	—	RAZ	—	—	Reserved.
<23:16>	INT	R/W	0x0	—	A write to this register results in a bit in one of the SLI_MSI_RCV*n* registers being set. Which bit is set depends on what is previously written using the SLI_MSI_WR_MAP register. If nothing is previously written, the bit set is the reset value of the MAP.
<15:0>	—	RAZ	—	—	Reserved.

SLI MAC MSI Receive Byte 3 Register
SLI_PCIE_MSI_RCV_B3

This register is where MSI write operations are directed from the MAC. This CSR can be used by PCIe MACs. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved.
<31:24>	INT	R/W	0x0	—	A write to this register results in a bit in one of the SLI_MSI_RCV*n* registers being set. Which bit is set depends on what is previously written using the SLI_MSI_WR_MAP register. If nothing is previously written, the bit set is the reset value of the MAP.
<23:0>	—	RAZ	—	—	Reserved.

SLI Packet Out Size Register
SLI_PKT(0..31)_OUT_SIZE

This register contains the BSIZE and ISIZE for output packet ports. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:23>	—	RAZ	—	—	Reserved.
<22:16>	ISIZE	R/W	0x0	—	Info bytes size (bytes) for rings 0–31. Legal sizes are 0 to 120. Not used in buffer-pointer-only mode.
<15:0>	BSIZE	R/W	0x0	—	Buffer size (bytes) for rings 0–31.

SLI Packet Instruction Enable Register
SLI_PKT_INSTR_ENB

This register enables the instruction fetch for a packet ring. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved.
<31:0>	ENB	R/W	0x0	—	Packet ring enable. When ENB<*i*>=1, instruction input ring *i* is enabled.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI Packet Output Enable Register
SLI_PKT_OUT_ENB

This register enables the output packet engines. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	ENB	R/W	0x0	—	Packet output enable. When ENB<*i*>=1, packet output ring *i* is enabled. If an error occurs on reading pointers for an output ring, the ring is disabled by clearing to 0 the bit associated with the ring.

SLI Packet Instruction Size Register
SLI_PKT_INSTR_SIZE

This register determines if packet rings use 32- or 64-byte instructions. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	IS_64B	R/W	0x0	—	64-byte instruction size. When IS_64B<*i*>=1, instruction input ring *i* uses 64-byte instructions, otherwise it uses 32-byte instructions.

SLI Packet Scatter List Relaxed-Ordering Register
SLI_PKT_SLIST_ROR

This register provides relaxed-ordering information for the TLP when fetching scatter lists. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	ROR	R/W	0x0	—	Relaxed-ordering enable. ADDRTYPE<0> for the packet-output ring read operations that fetch buffer/info pointer pairs. ROR<*i*> becomes ADDRTYPE<0> in DPI/SLI read operations that fetch buffer/info pairs from packet output ring *i* (from address SLI_PKT*i*_SLIST_BADDR+ in MAC memory space.) ADDRTYPE<0> is the relaxed-order attribute for PCIe.

SLI Packet Scatter List No-Snoop Register
SLI_PKT_SLIST_NS

This register provides no-snoop information for the TLP when fetching scatter lists. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	NSR	R/W	0x0	—	No-snoop enable. ADDRTYPE<1> for the packet-output ring read operations that fetch buffer/info pointer pairs. NSR<*i*> becomes ADDRTYPE<1> in DPI/SLI read operations that fetch buffer/info pairs from packet output ring *i* (from address SLI_PKT*i*_SLIST_BADDR+ in MAC memory space.) ADDRTYPE<1> is the no-snoop attribute for PCIe.

Cavium Confidential For Bowen Yang 02/2/2018

SLI Packet Scatter List Endian-Swap Register
SLI_PKT_SLIST_ES

This register provides endian-swap information for the TLP when fetching scatter lists. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	ES	R/W	0x0	—	Endian-swap mode. 0 = pass-through mode (no swap) 1 = 64-bit byte-swap mode [ABCD_EFGH] → [HGFE_DCBA] 2 = 32-bit byte-swap mode [ABCD_EFGH] → [DCBA_HGFE] 3 = 32-bit exchange mode [ABCD_EFGH] → [EFGH_ABCD] ES<1:0> for the packet-output ring read operations that fetch buffer/info pointer pairs. ES<2*i*+1:2*i*> becomes ES<1:0> in DPI/SLI read operations that fetch buffer/info pairs from packet output ring *i* (from address SLI_PKT*i*_SLIST_BADDR+ in MAC memory space.) ES<1:0> is the endian-swap attribute for these MAC memory space read operations.

SLI Packet Information Pointer Register
SLI_PKT_IPTR

This register specifies the use of the information pointer to store length and data. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved.
<31:0>	IPTR	R/W	0x0	—	Information-pointer enable. When IPTR<*i*>=1, packet output ring *i* is in info-pointer mode, else buffer-pointer-only mode.

SLI Packet Data-Pointer Address Register
SLI_PKT_DPADDR

This register determines address and attributes for packet-data write operations. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved.
<31:0>	DPTR	R/W	0x0	0xFFFFFFFF	Determines whether buffer/info pointers are DPTR format 0 or DPTR format 1. When DPTR<*i*>=1, the buffer/info pointers fetched from packet output ring *i* are DPTR format 0. When DPTR<*i*>=0, the buffer/info pointers fetched from packet output ring *i* are DPTR format 1. (Replace SLI_PKT_INPUT_CONTROL[D_ESR,D_NSR, D_ROR] in the descriptions of DPTR format 0/1 in Section 12.1.3 with SLI_PKT_DATA_OUT_ES[ES<2*i*+1:2*i*>], SLI_PKT_DATA_OUT_NS [NSR<*i*>], and SLI_PKT_DATA_OUT_ROR[ROR<*i*>], respectively for DPTR0/1 of type First Direct.)

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI Packet Data Out Relaxed-Ordering Register SLI_PKT_DATA_OUT_ROR

This register provides relaxed-ordering information for the TLP when writing packet data. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	ROR	R/W	0x0	—	Relaxed-ordering enable. ADDRTYPE<0> or MACADD<60> for buffer/info write operations. ROR<*i*> becomes either ADDRTYPE<0> or MACADD<60> for write operations to buffer/info pair MAC memory space addresses fetched from packet output ring *i*. ADDRTYPE<0> if SLI_PKT_DPADDR[DPTR<*i*>]=1, else MACADD<60>. In the latter case, ADDRTYPE<0> comes from DPTR<60>. ADDRTYPE<0> is the relaxed-order attribute for PCIe.

SLI Packet Scatter List No-Snoop Register SLI_PKT_DATA_OUT_NS

This register provides no-snoop information for the TLP when writing packet data. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	NSR	R/W	0x0	—	No-snoop enable. ADDRTYPE<1> or MACADD<61> for buffer/info write operations. NSR<*i*> becomes either ADDRTYPE<1> or MACADD<61> for write operations to buffer/info pair MAC memory space addresses fetched from packet output ring *i*. ADDRTYPE<1> if SLI_PKT_DPADDR[DPTR<*i*>]=1, else MACADD<61>. In the latter case, ADDRTYPE<1> comes from DPTR<61>. ADDRTYPE<1> is the no-snoop attribute for PCIe.

SLI Packet Data Out Endian-Swap Register SLI_PKT_DATA_OUT_ES

This register provides the endian swap for writing data out. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	ES	R/W	0x0	—	Endian swap mode. 0 = pass-through mode (no swap) 1 = 64-bit byte-swap mode [ABCD_EFGH] → [HGFE_DCBA] 2 = 32-bit byte-swap mode [ABCD_EFGH] → [DCBA_HGFE] 3 = 32-bit exchange mode [ABCD_EFGH] → [EFGH_ABCD] ES<1:0> or MACADD<63:62> for buffer/info write operations. ES<2*i*+1:2*i*> becomes either ES<1:0> or MACADD<63:62> for write operations to buffer/info pair MAC memory space addresses fetched from packet output ring *i*. ES<1:0> if SLI_PKT_DPADDR[DPTR<*i*>]=1, else MACADD<63:62>. In the latter case, ES<1:0> comes from DPTR<63:62>. ES<1:0> is the endian-swap attribute for these MAC memory space write operations.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI Packet Out Byte-Mode Register
SLI_PKT_OUT_BMODE

This register controls the updating of the SLI_PKT*n*_CNTS registers. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	BMODE	R/W	0x0	—	Byte mode. Determines whether SLI_PKT*i*_CNTS[CNT] is a byte or packet counter. When BMODE<*i*>=1, SLI_PKT*i*_CNTS[CNT] is a byte counter, else SLI_PKT*i*_CNTS[CNT] is a packet counter.

SLI Packet-To-MAC Port Assignment Register
SLI_PKT_PCIE_PORT

This register assigns packet ports to MAC ports. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	PP	R/W	0x0	—	The physical MAC port that the output ring uses. Two bits are used per ring (i.e. ring 0 <1:0>, ring 1 <3:2>, etc.). 0x0 Packet ring is assigned to MAC Port 0 0x1 Packet ring is assigned to MAC Port 1 0x2, 0x3 Reserved

SLI Packet-Interrupt Levels Register
SLI_PKT_INT_LEVELS

This register contains output-packet interrupt levels. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:54>	—	RAZ	—	—	Reserved
<53:32>	TIME	R/W	0x0	—	Output ring counter time interrupt threshold. SLI sets SLI_PKT_TIME_INT[PORT<*i*>] whenever SLI_PKT*i*_CNTS[TIMER] > TIME.
<31:0>	CNT	R/W	0x0	—	Output ring counter interrupt threshold. SLI sets SLI_PKT_CNT_INT[PORT<*i*>] whenever SLI_PKT*i*_CNTS[CNT] > CNT.

SLI Packet Counter Interrupt Register
SLI_PKT_CNT_INT

This register specifies which packet rings are interrupting because of packet counters. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	PORT	R/W1C/H	0x0	—	Output ring packet counter interrupt bits. SLI sets PORT<*i*> whenever SLI_PKT*i*_CNTS[CNT] > SLI_PKT_INT_LEVELS[CNT]. SLI_PKT_CNT_INT_ENB[PORT<*i*>] is the corresponding enable.

Cavium Confidential For Bowen Yang AEU 02/27/2014

SLI Packet Timer Interrupt Register
SLI_PKT_TIME_INT

This register specifies which packet rings are interrupting because of packet timers. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	PORT	R/W1C/H	0x0	—	Output ring packet timer interrupt bits. SLI sets PORT<*i*> whenever SLI_PKT*i*_CNTS[TIMER] > SLI_PKT_INT_LEVELS[TIME]. SLI_PKT_TIME_INT_ENB[PORT<*i*>] is the corresponding enable.

SLI Packet Counter Interrupt Enable Register
SLI_PKT_CNT_INT_ENB

This register provides the enable for the packet rings that are interrupting because of packet counters. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	PORT	R/W	0x0	—	Output ring packet counter interrupt enables. When both PORT<*i*> and corresponding SLI_PKT_CNT_INT[PORT<*i*>] are set, for any *i*, then SLI_INT_SUM[PCNT] is set, which can cause an interrupt.

SLI Packet Timer Interrupt Enable Register
SLI_PKT_TIME_INT_ENB

This register provides the enable for the packet rings that are interrupting because of packet timers. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	PORT	R/W	0x0	—	Output ring packet timer interrupt enables. When both PORT<*i*> and corresponding SLI_PKT_TIME_INT[PORT<*i*>] are set, for any *i*, then SLI_INT_SUM[PTIME] is set, which can cause an interrupt.

SLI Packet Input Control Register
SLI_PKT_INPUT_CONTROL

This register is the control for read operations for gather list and instructions. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63>	PRD_ERST	RO/H	0	—	PRD error reset.
<62:56>	PRD_RDS	RO/H	0x0	—	PRD reads out.
<55>	GII_ERST	RO/H	0	—	GII error reset.
<54:48>	GII_RDS	RO/H	0x0	—	GII reads out.
<47:41>	—	RAZ	—	—	Reserved.
<40>	PRC_IDLE	RO/H	0	—	PRC in idle.

Cavium Confidential For Bowen Yang ALU 02/2/2014

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<39:24>	—	RAZ	—	—	Reserved.
<23>	PIN_RST	R/W/H	0	0	Packet in reset. When a gather-list read receives an error, this bit (along with SLI_INT_SUM[PGL_ERR]) is set. When receiving a PGL_ERR interrupt the software should: 1. Wait 2ms to allow any outstanding reads to return or be timed out. 2. Write a 0 to this bit. 3. Startup the packet input again (all previous CSR setting of the packet-input will be lost). When this bit is set, the CSRs associated with packet out will return undefined results.
<22>	PKT_RR	R/W	0	1	When set to 1, the input packet selection is made with a round robin arbitration. When 0, the input packet ring is fixed in priority, where the lower ring number has higher priority.
<21:9>	PBP_DHI	R/W	0x0	0x0	PBP_DHI replaces address bits that are used for parse mode and skip-length when SLI_PKT*i*_INSTR_HEADER[PBP] = 1. PBP_DHI becomes either MACADD<63:55> or MACADD<59:51> for the instruction DPTR read operations in this case. The instruction DPTR read operations are called *first direct* or f*irst indirect*. When PBP = 1, if first direct and USE_CSR = 0, PBP_DHI becomes MACADD<59:51>, else MACADD<63:55>.
<8>	D_NSR	R/W	0	—	ADDRTYPE<1> or MACADD<61> for packet input data read operations. D_NSR becomes either ADDRTYPE<1> or MACADD<61> for MAC memory space read operations of packet input data fetched for any packet input ring. ADDRTYPE<1> if USE_CSR = 1, else MACADD<61>. In the latter case, ADDRTYPE<1> comes from DPTR<61>. ADDRTYPE<1> is the no-snoop attribute for PCIe.
<7:6>	D_ESR	R/W	0x0	—	ES<1:0> or MACADD<63:62> for packet input data read operations. D_ESR becomes either ES<1:0> or MACADD<63:62> for MAC memory space read operations of packet input data fetched for any packet input ring. ES<1:0> if USE_CSR = 1, else MACADD<63:62>. In the latter case, ES<1:0> comes from DPTR<63:62>. ES<1:0> is the endian-swap attribute for these MAC memory space read operations.
<5>	D_ROR	R/W	0	—	ADDRTYPE<0> or MACADD<60> for packet input data read operations. D_ROR becomes either ADDRTYPE<0> or MACADD<60> for MAC memory space read operations of packet input data fetched for any packet input ring. ADDRTYPE<0> if USE_CSR = 1, else MACADD<60>. In the latter case, ADDRTYPE<0> comes from DPTR<60>. ADDRTYPE<0> is the relaxed-order attribute for PCIe.
<4>	USE_CSR	R/W	0	1	When set to 1, the CSR value is used for ROR, ESR, and NSR. When clear to 0, the value in DPTR is used. In turn, the bits not used for ROR, ESR, and NSR are used for bits [63:60] of the address used to fetch packet data.
<3>	NSR	R/W	0	—	ADDRTYPE<1> for packet input instruction read operations and gather list (i.e. DPI component) read operations from MAC memory space. ADDRTYPE<1> is the no-snoop attribute for PCIe.
<2:1>	ESR	R/W	0x0	—	ES<1:0> for packet input instruction read operations and gather list (i.e. DPI component) read operations from MAC memory space. ES<1:0> is the endian-swap attribute for these MAC memory space read operations.
<0>	ROR	R/W	0	—	ADDRTYPE<0> for packet input instruction read operations and gather list (i.e. DPI component) read operations from MAC memory space. ADDRTYPE<0> is the relaxed-order attribute for PCIe.

Cavium Confidential For Review Only

SLI Packet Output Watermark Register
SLI_PKT_OUTPUT_WMARK

This register sets the value that determines when backpressure is applied to the PKO. When SLI_PKT(0..31)_SLIST_BAOFF_DBELL[DBELL] is less than [WMARK], backpressure for the rings is applied. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	WMARK	R/W	0x0	0xE	Packet output backpressure watermark. When the value of SLI_PKT(0..31)_SLIST_BAOFF_DBELL[DBELL] drops below this value, backpressure for the ring is applied to the PKO.

SLI Packet Instruction Read Size Register
SLI_PKT_INSTR_RD_SIZE

This register contains the number of instructions allowed to be read at one time. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	RDSIZE	R/W	0x0	—	Read size. Number of instructions to be read in one MAC read request for the four ports, eight rings. Every two bits (i.e., 1:0, 3:2, 5:4, etc.) are assigned to the following port/ring combinations. 15:0 PIP/IPD Port0, Ring 7..0 31:16 PIP/IPD Port1, Ring 7..0 47:32 PIP/IPD Port2, Ring 7..0 63:48 PIP/IPD Port3, Ring 7..0 Two bit values are: 0x0 = one instruction, 0x1 = two instructions, 0x2 = three instructions, 0x3 = four instructions.

SLI Packet-Input-to-MAC Port Assignment Register
SLI_PKT_IN_PCIE_PORT

This register assigns packet input rings to MAC ports. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	PP	R/W	0x0	—	The physical MAC port to which the packet ring number is assigned. Two bits are used per ring (i.e. ring 0 <1:0>, ring 1 <3:2>, etc.). 0x0 Packet ring is assigned to MAC Port 0 0x1 Packet ring is assigned to MAC Port 1 0x2 Packet ring is assigned to MAC Port 2 0x3 Packet ring is assigned to MAC Port 3

Cavium Confidential For Bowen Yang ALU 07/2014

SLI Packet Port In Reset Register
SLI_PKT_PORT_IN_RST

This register contains vector bits related to packet rings that are reset. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	IN_RST	RO/H	0x0	—	Inbound packet ring reset. When set to 1, the vector bit corresponding to the inbound packet ring is in reset.
<31:0>	OUT_RST	RO/H	0x0	—	Outbound packet ring reset. When set to 1, the vector bit corresponding to the outbound packet ring is in reset.

SLI Packet Input Instruction Register
SLI_PKT_IN_INSTR_COUNTS

This register contains keeps track of the number of instructions read into the FIFO and packets sent to IPD. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	WR_CNT	RO/H	0x0	—	Write count. This field shows the number of packets sent to the IPD.
<31:0>	RD_CNT	RO/H	0x0	—	Read count. This field shows the value of instructions that have had read operations issued for them.

SLI Packet Input Backpressure Register
SLI_PKT_IN_BP

This register indicates which input rings have backpressure applied. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	BP	RO/H	0x0	—	Backpressure is applied when a packet input ring has its count greater than the value in SLI_PKT(0..31)_IN_BP[WMARK]. Each of the 32 bits corresponds to an input ring. When the bit associated with a ring is 1, it has backpressure applied and will not fetch more instructions, but will process any previously fetched instructions.

SLI Packet Control Register
SLI_PKT_CTL

This register provides control signals for packets. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved.
<4>	RING_EN	R/W	0	0	When '0' forces "relative Q position" received from PKO to be zero, and replicates the backpressure indication for the first ring attached to a PKO port across all the rings attached to a PKO port. When '1' backpressure is on a per port/ring.
<3:0>	PKT_BP	R/W	0xF	0xF	Packet backpressure. When set to 1, the port-level backpressure for PKO ports associated with the bit are enabled.

Cavium Confidential For Bowen Yang 02/21/2014

SLI Packet Ring (0..31) Scatter List Base-Address Registers
SLI_PKT(0..31)_SLIST_BADDR

This register contains the start of scatter list for output-packet pointers. This address must be 16-byte aligned. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:4>	ADDR	R/W/H	0x0	—	Base address for scatter list pointers.
<3:0>	—	RAZ	—	—	Reserved.

SLI Packet Ring (0..31) Scatter List Base-Address Offset and Doorbell Registers
SLI_PKT(0..31)_SLIST_BAOFF_DBELL

This register contains the doorbell and base-address offset for next read operation. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	AOFF	RO/H	0x0	—	Address offset. The offset from the SLI_PKT*n*_SLIST_BADDR where the next SLIST pointer is read. A write of 0xFFFFFFFF to [DBELL] clears [DBELL] and [AOFF].
<31:0>	DBELL	R/W/H	0x0	0x0	Scatter-list doorbell count. Write operations to this field increment the present value here. Read operations return the present value. The value of this field is decremented as read operations are issued for scatter pointers. A write of 0xFFFFFFFF to this field clears [DBELL] and [AOFF].

SLI Packet Ring (0..31) Scatter List FIFO and Ring Size Registers
SLI_PKT(0..31)_SLIST_FIFO_RSIZE

This register contains the number of scatter pointer pairs in the scatter list. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RO	0x0	—	Reserved.
<31:0>	RSIZE	R/W	0x0	—	The number of scatter pointer pairs contained in the scatter list ring.

SLI Instruction Done Ring Number Counts Registers
SLI_PKT_IN_DONE(0..31)_CNTS

This register contains counters for instructions completed on input rings. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	—	RO	0x0	—	Reserved.
<31:0>	CNT	RO/H	0x0	—	This field is incremented by 1 when an instruction is completed. It is incremented as the last of the data is read from the MAC.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI Packet Ring (0..31) Counts Registers
SLI_PKT(0..31)_CNTS

This register contains the counters for output rings. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:54>	—	RO	—	—	Reserved.
<53:32>	TIMER	RO/H	0x0	0x0	Timer, incremented every 1024 coprocessor-clock cycles when SLI_PKT(0..31)_CNTS[CNT] is not zero. This field is cleared when SLI_PKT(0..31)_CNTS[CNT] goes to 0. This field is also cleared when SLI_PKT_TIME_INT is cleared. The first increment of this count can occur between 0 to 1023 coprocessor-clock cycles.
<31:0>	CNT	R/W/H	0x0	0x0	Ring counter. This field is incremented as packets are sent out and decremented in response to write operations to this field. When SLI_PKT_OUT_BMODE is set to 0, a value of 1 is added to this field for each packet; when SLI_PKT_OUT_BMODE is set to 1 and the info-pointer is not used, the length of the packet plus 8 is added; when SLI_PKT_OUT_BMODE is set to 1 and info-pointer mode is used, the packet length is added to this field.

SLI Packet Ring (0..31) Instruction Base Address Registers
SLI_PKT(0..31)_INSTR_BADDR

This register contains the start-of-instruction for input packets. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:3>	ADDR	R/W/H	0x0	—	Base address for instructions.
<2:0>	—	RAZ	—	—	Reserved.

SLI Packet Ring (0..31) Instruction Base Address Offset and Doorbell Registers
SLI_PKT(0..31)_INSTR_BAOFF_DBELL

This register contains the doorbell and base address offset for the next read. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	AOFF	RO/H	0x0	—	The offset from the SLI_PKT(0..31)_INSTR_BADDR where the next instruction is read.
<31:0>	DBELL	R/W/H	0x0	0x0	Instruction doorbell count. Write operations to this field increment the value here. Read operations return present value. A write of 0xFFFFFFFF to this field sets [DBELL] and [AOFF] to 0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI Packet Ring (0..31) Instruction FIFO and Ring Size Registers
SLI_PKT(0..31)_INSTR_FIFO_RSIZE

This register contains the FIFO field and ring size for instructions. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:55>	MAX	RO/H	0x10	—	Max FIFO size.
<54:46>	RRP	RO/H	0x0	—	FIFO read pointer.
<45:37>	WRP	RO/H	0x0	—	FIFO write pointer.
<36:32>	FCNT	RO/H	0x0	—	FIFO count.
<31:0>	RSIZE	R/W	0x0	—	Instruction ring size.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI Packet Ring (0..31) Instruction Header Registers
SLI_PKT(0..31)_INSTR_HEADER

This register contains values used to build input packet header. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:44>	—	RAZ	—	—	Reserved. Must be zero.
<43>	PBP	R/W	0	—	Enable packet-by-packet mode. Allows DPI to generate PKT_INST_HDR[PM,SL] differently per DPI instruction. USE_IHDR must be set whenever PBP is set.
<42:38>	—	RAZ	0x0	—	Reserved. Must be zero.
<37:36>	RPARMODE	R/W	0x0	—	Parse mode. Becomes PKT_INST_HDR[PM] when DPI_INST_HDR[R]==1 and PBP==0.
<35>	—	RAZ	0	—	Reserved. Must be zero.
<34:28>	RSKP_LEN	R/W	0x0	—	Skip length. Becomes PKT_INST_HDR[SL] when DPI_INST_HDR[R]==1 and PBP==0.
<27:26>	—	R/W	0x0	—	Reserved. Must be zero.
<25>	RNQOS	R/W	0	—	RNQOS. Becomes PKT_INST_HDR[NQOS] when DPI_INST_HDR[R]==1.
<24>	RNGRP	R/W	0	—	RNGRP. Becomes PKT_INST_HDR[NGRP] when DPI_INST_HDR[R]==1.
<23>	RNTT	R/W	0	—	RNTT. Becomes PKT_INST_HDR[NTT] when DPI_INST_HDR[R]==1.
<22>	RNTAG	R/W	0	—	RNTAG. Becomes PKT_INST_HDR[NTAG] when DPI_INST_HDR[R]==1.
<21>	USE_IHDR	R/W	0	—	When set to 1, DPI always prepends a PKT_INST_HDR as part of the packet data sent to PIP/IPD, regardless of DPI_INST_HDR[R]. (DPI also always prepends a PKT_INST_HDR when DPI_INST_HDR[R]=1.) USE_IHDR must be set whenever PBP is set.
<20:16>	—	R/W	0x0	—	Reserved. Must be zero.
<15:14>	PAR_MODE	R/W	0x0	—	Parse mode. Becomes PKT_INST_HDR[PM] when DPI_INST_HDR[R]==0 and USE_IHDR==1 and PBP==0.
<13>	—	R/W	0	—	Reserved. Must be zero.
<12:6>	SKP_LEN	R/W	0x0	—	Skip length. Becomes PKT_INST_HDR[SL] when DPI_INST_HDR[R]==0 and USE_IHDR==1 and PBP==0.
<5:4>	—	R/W	0x0	—	Reserved. Must be zero.
<3>	NQOS	R/W	0	—	NQOS. Becomes PKT_INST_HDR[NQOS] when DPI_INST_HDR[R]==0 (and USE_IHDR==1).
<2>	NGRP	R/W	0	—	NGRP. Becomes PKT_INST_HDR[NGRP] when DPI_INST_HDR[R]==0 (and USE_IHDR==1).
<1>	NTT	R/W	0	—	NTT. Becomes PKT_INST_HDR[NTT] when DPI_INST_HDR[R]==0 (and USE_IHDR==1).
<0>	NTAG	R/W	0	—	NTAG. Becomes PKT_INST_HDR[NTAG] when DPI_INST_HDR[R]==0 (and USE_IHDR==1).

Cavium Confidential For Beverly Yang ALU 6/2/2014

SLI Packet Ring (0..31) Input Backpressure Registers
SLI_PKT(0..31)_IN_BP

This register contains the counters and thresholds for input packets to apply backpressure to processing of the packets. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:32>	WMARK	R/W	0xFFFFFFFF	—	Watermark. When the CNT field is greater than this threshold, no additional packets are processed for this ring. When writing to this field, use a 4-byte write so as to not write any other field of this register.
<31:0>	CNT	R/W/H	0x0	0x0	Ring counter. This field is incremented by one whenever CN71XX receives, buffers, and creates a work queue entry for a packet that arrives by the corresponding input ring. A write to this field is subtracted from the field value. When writing to this field, use a 4-byte write so as to not write any other field of this register.

SLI MSI Receive0 Register
SLI_MSI_RCV0

This register contains bits <63:0> of the 256 bits of MSI interrupts. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	INTR	R/W1C/H	0x0	0x0	Bits 63–0 of the 256 bits of MSI interrupt.

SLI MSI Receive1 Register
SLI_MSI_RCV1

This register contains bits <127:64> of the 256 bits of MSI interrupts. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	INTR	R/W1C/H	0x0	0x0	Bits 127–64 of the 256 bits of MSI interrupt.

SLI MSI Receive2 Register
SLI_MSI_RCV2

This register contains bits <191:128> of the 256 bits of MSI interrupts. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	INTR	R/W1C/H	0x0	0x0	Bits 191–128 of the 256 bits of MSI interrupt.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI MSI Receive3 Register
SLI_MSI_RCV3

This register contains bits <255:192> of the 256 bits of MSI interrupts. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	INTR	R/W1C/H	0x0	0x0	Bits 255–192 of the 256 bits of MSI interrupt.

SLI MSI Enable0 Register
SLI_MSI_ENB0

This register is used to enable interrupt generation for the bits in SLI_MSI_RCV0. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	ENB	R/W/H	0x0	all-ones	Enables bits <63:0> of SLI_MSI_RCV0.

SLI MSI Enable1 Register
SLI_MSI_ENB1

This register is used to enable interrupt generation for the bits in SLI_MSI_RCV0. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	ENB	R/W/H	0x0	all-ones	Enables bits <63:0> of SLI_MSI_RCV1.

SLI MSI Enable2 Register
SLI_MSI_ENB2

This register is used to enable interrupt generation for the bits in SLI_MSI_RCV0. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	ENB	R/W/H	0x0	all-ones	Enables bits <63:0> of SLI_MSI_RCV2.

SLI MSI Enable3 Register
SLI_MSI_ENB3

This register is used to enable interrupt generation for the bits in SLI_MSI_RCV0. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	ENB	R/W/H	0x0	all-ones	Enables bits <63:0> of SLI_MSI_RCV3.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI MSI Write Map Register
SLI_MSI_WR_MAP

This register is used to write the mapping function of the SLI_PCIE_MSI_RCV to SLI_MSI_RCV registers.

At reset, the mapping function is one-to-one, that is MSI_INT 1 maps to CIU_INT 1, 2 to 2, 3 to 3, etc. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved.
<15:8>	CIU_INT	R/W	0x0	—	Selects which bit in the SLI_MSI_RCV(0..3) (0–255) will be set when the value specified in the MSI_INT of this register is received during a write operation to the SLI_PCIE_MSI_RCV register.
<7:0>	MSI_INT	R/W	0x0	—	Selects the value that is received when the SLI_PCIE_MSI_RCV register is written.

SLI MSI Read Map Register
SLI_MSI_RD_MAP

This register is used to read the mapping function of the SLI_PCIE_MSI_RCV to SLI_MSI_RCV registers. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:8>	RD_INT	RO/H	0x0	—	The value of th.e map at the location previously written to the MSI_INT field of this register.
<7:0>	MSI_INT	R/W	0x0	—	Selects the value that is received when the SLI_PCIE_MSI_RCV register is written.

SLI MAC MSI Receive Register
SLI_PCIE_MSI_RCV

This is the register where MSI write operations are directed from the MAC. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	INT	R/W	0x0	—	A write operation to this register results in a bit in one of the SLI_MSI_RCV(0..3) registers being set. Which bit is set depends on the value previously written using the SLI_MSI_WR_MAP register, or if not previously written, the reset value of the MAP.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI Interrupt Enable CIU Register
SLI_INT_ENB_CIU

This register is used to enable for the CIU the various interrupting conditions of SLI. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:61>	—	R/W	0x0	0x0	Reserved
<60>	ILL_PAD	R/W	0	0	Illegal packet csr address.
<59>	SPRT3_ERR	R/W	0	0	Error response received on SLI port 3.
<58>	SPRT2_ERR	R/W	0	0	Error response received on SLI port 2.
<57>	SPRT1_ERR	R/W	0	0	Error response received on SLI port 1.
<56>	SPRT0_ERR	R/W	0	0	Error response received on SLI port 0.
<55>	PINS_ERR	R/W	0	0	Read error during packet instruction fetch.
<54>	POP_ERR	R/W	0	0	Read error during packet scatter pointer fetch.
<53>	PDI_ERR	R/W	0	0	Read error during packet data fetch.
<52>	PGL_ERR	R/W	0	0	Read error during gather list fetch.
<51>	PIN_BP	R/W	0	0	Packet input count exceeded WMARK.
<50>	POUT_ERR	R/W	0	0	Packet out interrupt error from PKO.
<49>	PSLDBOF	R/W	0	0	Packet scatterlist doorbell count overflow.
<48>	PIDBOF	R/W	0	0	Packet instruction doorbell count overflow.
<47:38>	—	R/W	0x0	0x0	Reserved.
<37:36>	DTIME	R/W	0x0	0x0	DMA timer interrupts.
<35:34>	DCNT	R/W	0x0	0x0	DMA count interrupts.
<33:32>	DMAFI	R/W	0x0	0x0	DMA set forced interrupts.
<31:29>	—	R/W	0x0	0x0	Reserved.
<28>	MIO_INT2	R/W	0	1	Enables SLI_INT_SUM[MIO_INT2] to generate an interrupt on the RSL.
<27>	M3_UN_WI	R/W	0	0	Reserved.
<26>	M3_UN_B0	R/W	0	0	Reserved.
<25>	M3_UP_WI	R/W	0	0	Reserved.
<24>	M3_UP_B0	R/W	0	0	Reserved.
<23>	M2_UN_WI	R/W	0	1	Enables SLI_INT_SUM[M2_UN_WI] to generate an interrupt on the RSL.
<22>	M2_UN_B0	R/W	0	1	Enables SLI_INT_SUM[M2_UN_B0] to generate an interrupt on the RSL.
<21>	M2_UP_WI	R/W	0	1	Enables SLI_INT_SUM[M2_UP_WI] to generate an interrupt on the RSL.
<20>	M2_UP_B0	R/W	0	1	Enables SLI_INT_SUM[M2_UP_B0] to generate an interrupt on the RSL.
<19:18>	—	R/W	0x0	0x0	Reserved.
<17>	MIO_INT1	R/W	0	0	Enables SLI_INT_SUM[MIO_INT1] to generate an interrupt on the RSL.
<16>	MIO_INT0	R/W	0	0	Enables SLI_INT_SUM[MIO_INT0] to generate an interrupt on the RSL.
<15>	M1_UN_WI	R/W	0	1	Enables SLI_INT_SUM[M1_UN_WI] to generate an interrupt on the RSL.
<14>	M1_UN_B0	R/W	0	1	Enables SLI_INT_SUM[M1_UN_B0] to generate an interrupt on the RSL.
<13>	M1_UP_WI	R/W	0	1	Enables SLI_INT_SUM[M1_UP_WI] to generate an interrupt on the RSL.
<12>	M1_UP_B0	R/W	0	1	Enables SLI_INT_SUM[M1_UP_B0] to generate an interrupt on the RSL.
<11>	M0_UN_WI	R/W	0	1	Enables SLI_INT_SUM[M0_UN_WI] to generate an interrupt on the RSL.
<10>	M0_UN_B0	R/W	0	1	Enables SLI_INT_SUM[M0_UN_B0] to generate an interrupt on the RSL.
<9>	M0_UP_WI	R/W	0	1	Enables SLI_INT_SUM[M0_UP_WI] to generate an interrupt on the RSL.
<8>	M0_UP_B0	R/W	0	1	Enables SLI_INT_SUM[M0_UP_B0] to generate an interrupt on the RSL.

Cavium Confidential For Bowen Yang ALU 12/2/2014

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<7:6>	—	R/W	0x0	0x0	Reserved
<5>	PTIME	R/W	0	1	Enables SLI_INT_SUM[PTIME] to generate an interrupt on the RSL.
<4>	PCNT	R/W	0	1	Enables SLI_INT_SUM[PCNT] to generate an interrupt on the RSL.
<3>	IOB2BIG	R/W	0	1	Enables SLI_INT_SUM[IOB2BIG] to generate an interrupt on the RSL.
<2>	BAR0_TO	R/W	0	1	Enables SLI_INT_SUM[BAR0_TO] to generate an interrupt on the RSL.
<1>	—	R/W	0	1	Enables SLI_INT_SUM<1> to generate an interrupt on the RSL.
<0>	RML_RTO	R/W	0	1	Enables SLI_INT_SUM[RML_RTO] to generate an interrupt on the RSL.

SLI MSI Write-1-to-Clear Enable0 Register
SLI_MSI_W1C_ENB0

This register is used to clear bits in SLI_MSI_ENB0. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	CLR	R/W/H	0x0	0x0	A write of 1 to a vector clears the corresponding bit in SLI_MSI_ENB0. A read to this address returns 0x0.

SLI MSI Write-1-to-Clear Enable1 Register
SLI_MSI_W1C_ENB1

This register is used to clear bits in SLI_MSI_ENB1. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	CLR	R/W/H	0x0	0x0	A write of 1 to a vector clears the corresponding bit in SLI_MSI_ENB1. A read to this address returns 0x0.

SLI MSI Write-1-to-Clear Enable2 Register
SLI_MSI_W1C_ENB2

This register is used to clear bits in SLI_MSI_ENB2. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	CLR	R/W/H	0x0	0x0	A write of 1 to a vector clears the corresponding bit in SLI_MSI_ENB2. A read to this address returns 0x0.

SLI MSI Write-1-to-Clear Enable3 Register
SLI_MSI_W1C_ENB3

This register is used to clear bits in SLI_MSI_ENB3. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	CLR	R/W/H	0x0	0x0	A write of 1 to a vector clears the corresponding bit in SLI_MSI_ENB3. A read to this address returns 0x0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI MSI Write-1-to-Set Enable0 Register
SLI_MSI_W1S_ENB0

This register is used to set bits in SLI_MSI_ENB0. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	SET	R/W/H	0x0	0x0	A write of 1 to a vector sets the corresponding bit in SLI_MSI_ENB0. A read to this address returns 0.

SLI MSI Write-1-to-Set Enable1 Register
SLI_MSI_W1S_ENB1

This register is used to set bits in SLI_MSI_ENB1. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	SET	R/W/H	0x0	0x0	A write of 1 to a vector sets the corresponding bit in SLI_MSI_ENB1. A read to this address returns 0.

SLI MSI Write-1-to-Set Enable2 Register
SLI_MSI_W1S_ENB2

This register is used to set bits in SLI_MSI_ENB2. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	SET	R/W/H	0x0	0x0	A write of 1 to a vector sets the corresponding bit in SLI_MSI_ENB2. A read to this address returns 0.

SLI MSI Write-1-to-Set Enable3 Register
SLI_MSI_W1S_ENB3

This register is used to set bits in SLI_MSI_ENB3. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	SET	R/W/H	0x0	0x0	A write of 1 to a vector sets the corresponding bit in SLI_MSI_ENB3. A read to this address returns 0.

Cavium Confidential For Bowen Yang ALU 9/27/2014

SLI MAC Credit Count Register
SLI_MAC_CREDIT_CNT

This register contains the number of credits for the MAC port FIFOs used by the SLI. This value needs to be set before S2M traffic flow starts. A write operation to this register causes the credit counts in the SLI for the MAC ports to be reset to the value in this register. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:54>	—	RAZ	—	—	Reserved
<53>	P1_C_D	R/W	1	1	When set, does not allow writing of P1_CCNT.
<52>	P1_N_D	R/W	1	1	When set, does not allow writing of P1_NCNT.
<51>	P1_P_D	R/W	1	1	When set, does not allow writing of P1_PCNT.
<50>	P0_C_D	R/W	1	1	When set, does not allow writing of P0_CCNT.
<49>	P0_N_D	R/W	1	1	When set, does not allow writing of P0_NCNT.
<48>	P0_P_D	R/W	1	1	When set, does not allow writing of P0_PCNT.
<47:40>	P1_CCNT	R/W/H	0x80	0x80	Port 1 C-TLP FIFO credits. Legal values are 0x25 to 0x80.
<39:32>	P1_NCNT	R/W/H	0x10	0x10	Port 1 N-TLP FIFO credits. Legal values are 0x5 to 0x10.
<31:24>	P1_PCNT	R/W/H	0x80	0x80	Port 1 P-TLP FIFO credits. Legal values are 0x25 to 0x80.
<23:16>	P0_CCNT	R/W/H	0x80	0x80	Port 0 C-TLP FIFO credits. Legal values are 0x25 to 0x80.
<15:8>	P0_NCNT	R/W/H	0x10	0x10	Port 0 N-TLP FIFO credits. Legal values are 0x5 to 0x10.
<7:0>	P0_PCNT	R/W/H	0x80	0x80	Port 0 P-TLP FIFO credits. Legal values are 0x25 to 0x80.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI SLI-to-MAC Port0–2 Control Registers
SLI_S2M_PORT(0..2)_CTL

These registers contain control for access from SLI to a MAC port. Write operations to these register are not ordered with write/read operations to the MAC Memory space. To ensure that a write operation has completed, read the register before making an access (i.e. MAC memory space) that requires the value of this register to be updated. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved
<4>	WIND_D	R/W	—	—	Window disable. When set to 1, disables access to the window registers from the MAC port. When authenticate mode is set the reset value of this field is 1 else 0.
<3>	BAR0_D	R/W	—	—	BAR0 disable. When set to 1, disables access from the MAC to BAR0 for the following address offsets: • 0x0–0x32F • 0x3CD0 • greater than 0x3D70, excluding 0x3E00. When authenticate mode is set the reset value of this field is 1 else 0.
<2:0>	MRRS	R/W	0x2	0x2	Maximum read-request size. 0x0 = 128 bytes 0x1 = 256 bytes 0x2 = 512 bytes 0x3 = 1024 bytes 0x4 = 2048 bytes 0x5–0x7= reserved This field should not exceed the desired max read request size. This field is used to determine if an IOBDMA is too large. For a PCIe MAC, this field should not exceed PCIE*_CFG030[MRRS].

Cavium Confidential For Bowen Yang ALU 02/27/2014

SLI MAC Number Register
SLI_MAC_NUMBER

When read from a MAC port, this register returns the MAC's port number. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8>	A_MODE	RO/H	—	—	SLI in authenticate mode.
<7:0>	NUM	RO/H	—	—	MAC number.

SLI MAC Number Register
SLI_MAC_CREDIT_CNT2

Contains the number of credits for the MAC port FIFOs (for MAC 2) used by the SLI. This value needs to be set before S2M traffic flow starts. A write to this register causes the credit counts in the SLI for the MAC ports to be reset to the value in this register. See Table 12–5 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:54>	—	RAZ	—	—	Reserved.
<53>	P3_C_D	R/W	1	1	When set, does not allow writing of P3_CCNT.
<52>	P3_N_D	R/W	1	1	When set, does not allow writing of P3_NCNT.
<51>	P3_P_D	R/W	1	1	When set, does not allow writing of P3_PCNT.
<50>	P2_C_D	R/W	1	1	When set, does not allow writing of P2_CCNT.
<49>	P2_N_D	R/W	1	1	When set, does not allow writing of P2_NCNT.
<48>	P2_P_D	R/W	1	1	When set, does not allow writing of P2_PCNT.
<47:40>	P3_CCNT	R/W/H	0x80	0x80	Port 3 C-TLP FIFO credits. Legal values are 0x25 to 0x80.
<39:32>	P3_NCNT	R/W/H	0x10	0x10	Port 3 N-TLP FIFO credits. Legal values are 0x5 to 0x10.
<31:24>	P3_PCNT	R/W/H	0x80	0x80	Port 3 P-TLP FIFO credits. Legal values are 0x25 to 0x80.
<23:16>	P2_CCNT	R/W/H	0x80	0x80	Port 2 C-TLP FIFO credits. Legal values are 0x25 to 0x80.
<15:8>	P2_NCNT	R/W/H	0x10	0x10	Port 2 N-TLP FIFO credits. Legal values are 0x5 to 0x10.
<7:0>	P2_PCNT	R/W/H	0x80	0x80	Port 2 P-TLP FIFO credits. Legal values are 0x25 to 0x80.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Chapter 13

Central Interrupt Unit (CIU)

This chapter contains the following subjects:

- Overview
- Central Interrupt Collection and Distribution
- Per-Core Mailbox Registers
- Per-Core Watchdog Timers
- Ten General Timers
- Core Availability and Reset
- Core Debug-Mode Observability
- Core Debug-Interrupt Generation
- Core Non-Maskable Interrupt Generation
- Chip Soft-Reset Initiation
- CIU Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The CN71XX Central Interrupt unit (CIU) implements the following functions:

- Central interrupt collection, both internal and external (CIU_INT*z*_SUM0/1, CIU_INT*z*_SUM4, CIU_SUM1_*xx*_*yy*).

NOTE: For the summary register designations, the following variables are used:

z represents the selected interrupt. Its value is between 0–7 or 32.

xx represents the destination of the interrupt:
PP0–PP3 indicate core 0 through core 3
IO0 indicates I/O.

yy represents the interrupt level into the destination:
IP2, IP3, IP4 for the cores
INT for I/O.

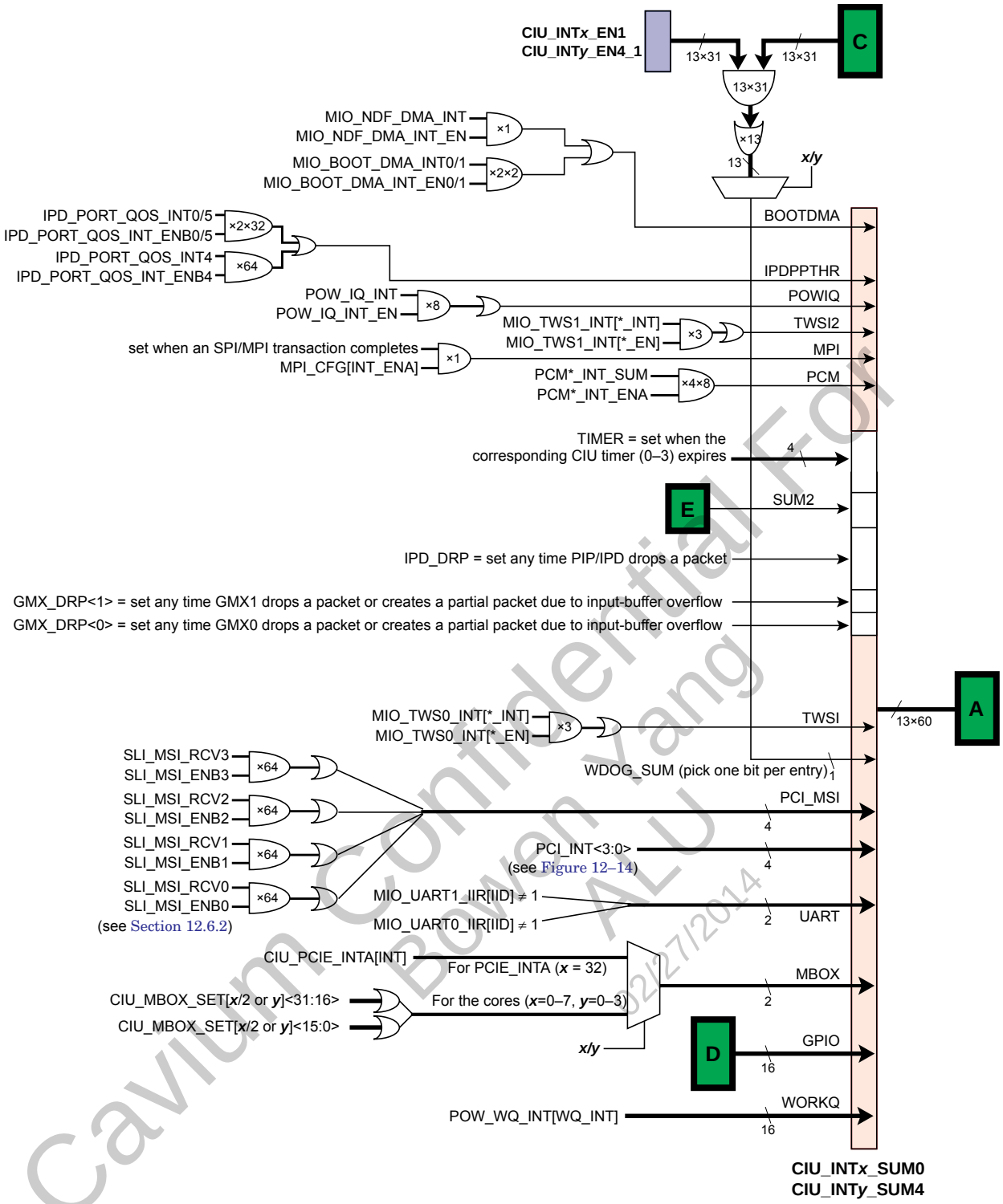
- Central interrupt selection and distribution to cores, and to one PCIe INTA/MSI when CN71XX is an endpoint, not a PCIe host (CIU_INT*z*_EN0/1, CIU_INT*z*_EN4_0/1).
- Per-core mailbox registers (CIU_MBOX_SET*n*, CIU_MBOX_CLR*n*).
- Per-core watchdog timers (CIU_WDOG*n*, CIU_PP_POKE*n*)
- Four general timers (CIU_TIM*n*)
- Core availability and reset (CIU_FUSE and CIU_PP_RST)
- Core debug-mode observability (CIU_PP_DBG)
- Core debug-interrupt generation (CIU_DINT)
- Core nonmaskable interrupt generation (CIU_NMI)
- Chip soft-reset initiation (CIU_SOFT_RST, CIU_SOFT_BIST)
- Global-stop (GSTOP) bit (CIU_GSTOP). Used for multicore debug. Refer to Section 2.22 for more details on GSTOP usage.

13.1 Central Interrupt Collection and Distribution

Figures 13–1, 13–2, 13–3, 13–4, and 13–5 show CN71XX's entire interrupt-distribution network, some of which resides in the CIU. Figure 13–1 shows that the CIU distributes a total of 13 interrupts—three per core plus one more (one per PCIe core) for INTA/MSI generation. The three interrupts for each core become Cause[IP4,IP3,IP2] local to the core. With three separate interrupt destinations per core, different interrupts destined for the core can be prioritized differently by the core-interrupt handlers.

Cavium Confidential For Bowen Yang ALU 5/27/2014

Figure 13–1 CN71XX Interrupt Distribution (CIU_INT*n*_SUM0/4)

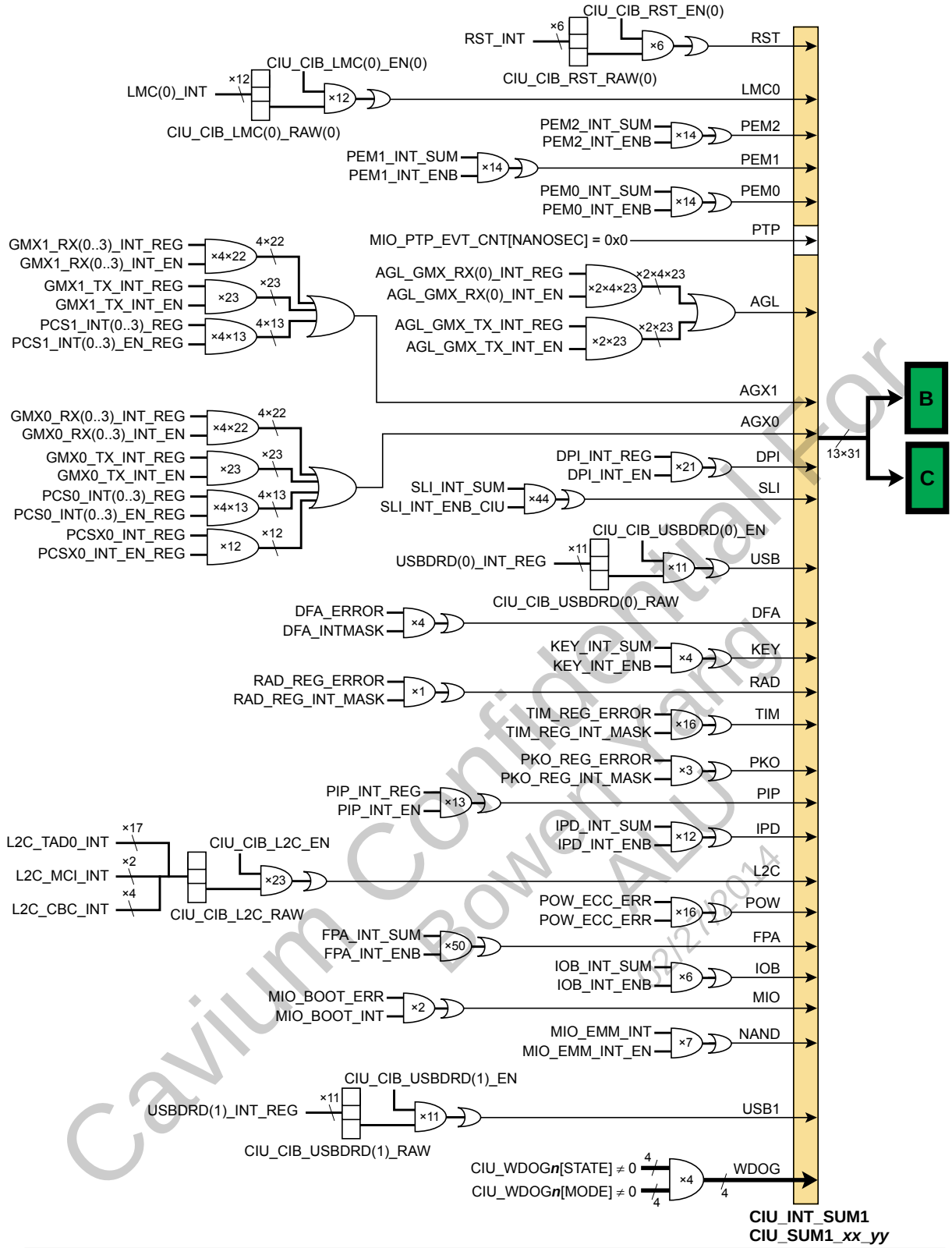


Cavium Confidential For Bowen Yang ALU 02/27/2014

Notes:

1. The variables ***x*** and ***y*** represent the selected interrupts. ***x*** is a value between 0 and 7 or 32 (used with *_SUM0 registers) and ***y*** is a value between 0 and 3 (used with *_SUM4 registers).
2. The sections of CIU_INT(***x/y***)_SUM0/4 that are shown in white are latched. All other bits pass right through.

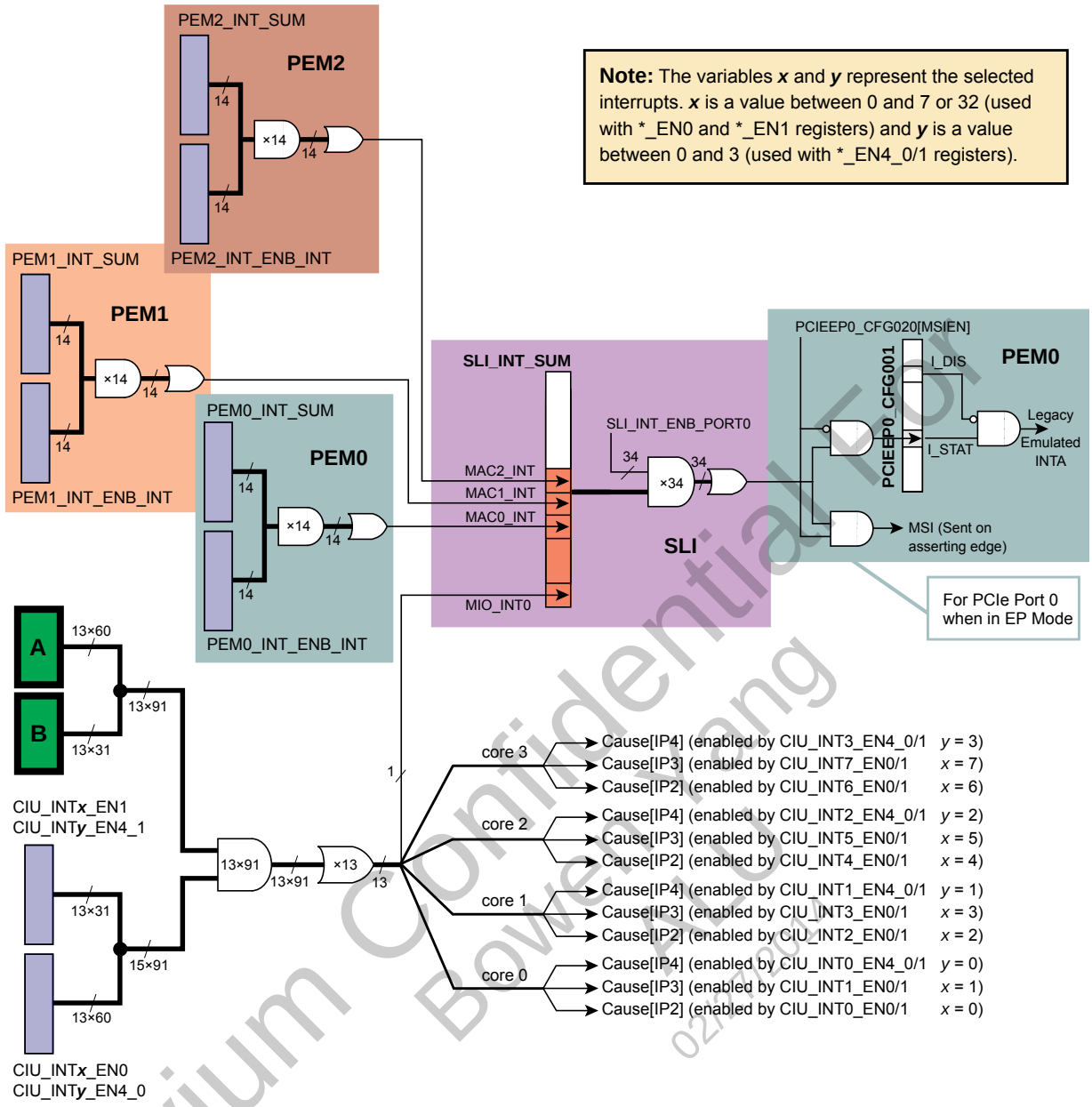
Figure 13–2 CN71XX Interrupt Distribution (CIU_INT*n*_SUM1)



Cavium Confidential For Bowen Yang ADI 10/27/2014

Note: The variable ***n*** represents the number of cores. It is a value between 0 and 3.
The variable ***xx*** represents the destination of the interrupts: core 0 ... core 3, IO0, IO1.
The variable ***yy*** represents the interrupt level into the destination: IP2 ..IP4 for the cores, INT for I/O.

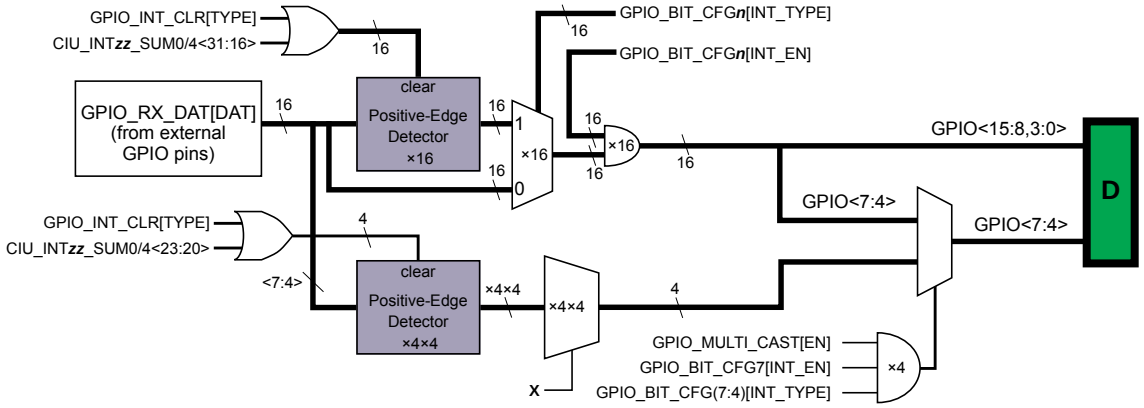
Figure 13–3 CN71XX PCIe Interrupt Distribution



Note: The variables ***x*** and ***y*** represent the selected interrupts. ***x*** is a value between 0 and 7 or 32 (used with *_EN0 and *_EN1 registers) and ***y*** is a value between 0 and 3 (used with *_EN4_0/1 registers).

Cavium Confidential For Bowen Peng 02/[illegible]/2013

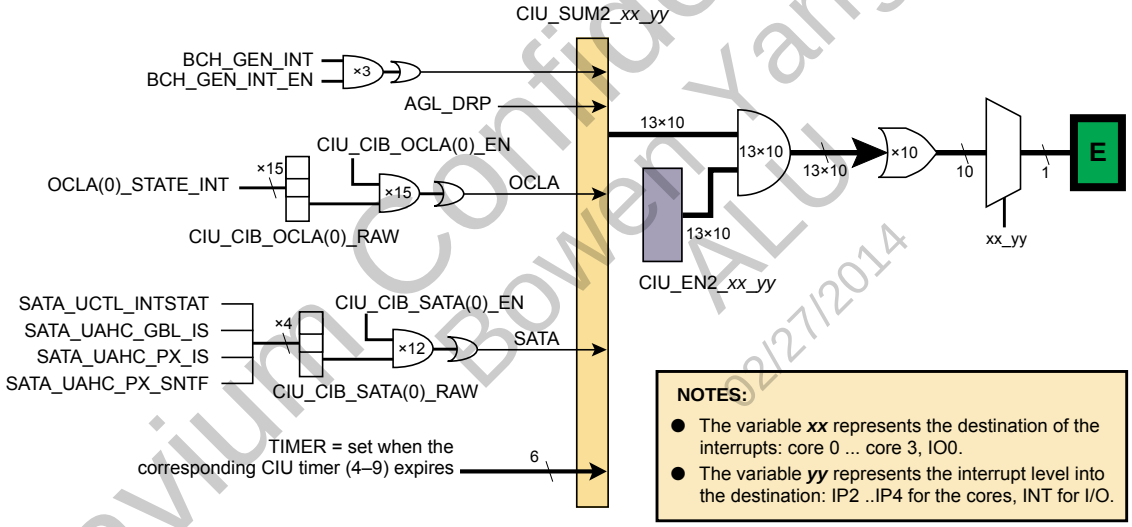
Figure 13–4 CN71XX GPIO Interrupt Distribution



NOTES:

- the positive-edge detector is ×4×4 because there is a flop for each of the four cores
- the variable ***zz*** represents a number between 0–7 or 32 for the *SUM0 registers or between 0–3 for the *SUM4 registers
- the variable **X** represents the selected core (0–3)
- **n** represents the specific GPIO signal (0–15)

Figure 13–5 CN71XX SUM2 Interrupt Distribution



NOTES:

- The variable ***xx*** represents the destination of the interrupts: core 0 ... core 3, IO0.
- The variable ***yy*** represents the interrupt level into the destination: IP2 ..IP4 for the cores, INT for I/O.

Cavium Confidential For Bowen Yang ALU 02/27/2014

As Figures 13–1, 13–2, and 13–3 show, there are more than 500 possible interrupt sources on CN71XX. Most of these sources are accumulated and flow into the 154-bit summary vector CIU_INT*z*_SUM0/4 and CIU_SUM1/2_*xx*_*yy*. The 154-bit summary directly contains the remainder of the interrupt sources; those direct interrupts can be more quickly discerned and processed. The corresponding 154 bits of interrupt-enable vector in {CIU_INT*z*_EN0/1, CIU_INT*z*_EN4_0/1} allow the summarized interrupts to reach their ultimate destination.

CN71XX configuration registers contain a different 154-bit enable (CIU_INTz_EN0/1, CIU_INT*z*_EN4_0/1) for each of the 13 interrupt destinations, so each destination can include arbitrary and independent combinations of the 154 sources. Any interrupt in the 154-bit summary can be directed to any combination of the interrupt destinations.

There is a different summary vector for each interrupt destination. These summary vectors differ only in two fields:

- CIU_INT*z*_SUM0[MBOX].
- CIU_INT*z*_SUM0[WDOG_SUM].

All other fields in CIU_INT*z*_SUM*n* are identical for the different destinations. CIU_INT*z*_SUM0[MBOX] is a summary of the mailbox that is specific to individual cores, as described below. (CIU_INT*z*_SUM0/4[MBOX] is CIU_PCI_INTA[INT] for the PCIe destination.) CIU_INT*z*_SUM0[WDOG_SUM] is a convenient summary of the CIU_INT*z*_SUM1/CIU_INT*z*_EN4_1 result, so a single 64-bit access to CIU_INT*z*_SUM0/4 summarizes the available interrupts for the destination.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Note that Figure 13–1 shows a simple depiction of the interrupt-distribution logic. In the real implementation, there are delays to update interrupt destinations. The delay from when an interrupt source is updated to when the interrupt destination is updated may be as large as 60 cycles.

Except for CIU_INT*z*_SUM0/4[TIMER,IPD_DRP,GMX_DRP] and CIU_INT_SUM1[PTP], CIU_INT*z*_SUM0/1/4 does not actually store the status bits for the interrupt, only summarizes them. Note that there is only one copy of the storage for CIU_INT*z*_SUM0/4[TIMER,IPD_DRP,GMX_DRP] and CIU_INT_SUM1[PTP]. A write of 1 to any bit in these fields in any CIU_INT*z*_SUM0/4 clears the bit in all summaries.

CIU_INT*z*_SUM0/1/4[WDOG,TIMER,MBOX] are discussed in more detail in the following sections of this chapter. Some of the remainder of the interrupts are discussed briefly here:

- CIU_INT*z*_SUM0/4[IPD_DRP,GMX_DRP<2:0>] are set whenever PIP/IPD or GMX0/1 drop a packet. As these are all the possible sources of packet drops (except for the RGMII interface), software can always be notified quickly of any packet drop.
- CIU_INT*z*_SUM0/4[TWSI, TWSI2] can assert on TWSI transmit arrival or completion. Refer to Chapter 31.
- CIU_INT*z*_SUM0/4[PCI_MSI] provides quick access to SLI_MSI_RCV*, which can be used for PCIe message-signaled interrupts. Refer to Section 12.6.
- CIU_INT*z*_SUM0/4[PCI_INT] provides legacy PCI INTA/INTB/INTC/INTD interrupts as described in Section 12.6.1.
- CIU_SUM1_*xx_yy*[DPI_DMA] provides DPI DMA instruction completion interrupts. See Section 12.3.10.2.
- CIU_INT*z*_SUM0/4[UART] are set when the corresponding UART indicates an interrupt. Refer to Chapter 30.
- CIU_INT*z*_SUM0/4[GPIO] provides either level-sensitive or edge-triggered interrupts for GPIO_RX_DAT[DAT]. GPIO_RX_DAT[DAT] is a 16-bit vector. CIU_INT*z*_SUM0/4[GPIO] also provides multicast GPIO interrupts. Refer to Chapter 29.
- CIU_INT*z*_SUM0/4[WORKQ] provides per-group interrupts for POW. Refer to Section 7.6.1.
- CIU_PCI_INTA provides a direct mechanism for cores to send interrupts to the PCIe interfaces.
- CIU_INT*z*_SUM0/4[POWIQ] provides POW input-queue-threshold interrupts. Refer to Section 7.6.2.
- CIU_INT*z*_SUM0/4[IPDPPTHR] provide input packet per-port and QOS threshold interrupts. Refer to Section 9.8.
- CIU_INT*z*_SUM0/4[BOOTDMA] provide interrupts for the boot-bus DMA engines. Refer to Section 25.5.

CIU_INT*z*_SUM1/2[LMC0] is set whenever a bit in CIU_CIB_LMC0_RAW ANDed with CIU_CIB_LMC0_EN is set. Each CIU_CIB_LMC0_RAW bit is set when the corresponding LMC0_INT bit is set. To clear an interrupt, CIU_CIB_LMC0_RAW then LMC0_INT register bits must be written to 1 to clear.

CIU_SUM1_PP(0..3)_IP2[USB, USB1, L2C, OCLA, SATA] likewise have CIU_CIB_* registers.

Cavium Confidential For Bowen Yang ALU 2/27/2014

13.2 Per-Core Mailbox Registers

Each core has a corresponding 32-bit mailbox register whose bits can be set/cleared. This CIU_MBOX_SET*n*/CIU_MBOX_CLR*n* register is useful for inter-core direct interrupts, and a remote host can also direct an interrupt to a specific core with it.

Both CIU_MBOX_SET*n*[SET] and CIU_MBOX_CLR*n*[CLR] return the same copy of the mailbox bits for a core when they are read. Writing 1s to CIU_MBOX_SET*n*[SET] cause corresponding mailbox bits to be set. Writing 1s to CIU_MBOX_CLR*n*[CLR] cause corresponding mailbox bits to be cleared. With this, individual bits can be set and cleared simultaneously by multiple sources.

CIU_INTz_SUM0/4[MBOX] for a core summarizes the 32-bit mailbox register for the core. The core corresponding to summary *n* is n/2. CIU_INTz_SUM0/4[MBOX<1>] is set when any of <31:16> are set in the mailbox for the core.
CIU_INTz_SUM0/4[MBOX<0>] is set when any of <15:0> are set in the mailbox for the core.

13.3 Per-Core Watchdog Timers

There are four watchdog timers in CN71XX, one for each of the four cores. The watchdogs can be accessed by any core (or external PCIe device), though a watchdog may generally only be used by its associated core. The CIU_WDOG(0..3) registers configures each watchdog.

When not OFF, the watchdogs count the time from the last “core poke” for the watchdog. A core poke indicates that the core is alive, and the watchdog should be reset. If a core poke does not arrive soon enough, a timeout occurs and the watchdog checks for another timeout. CIU_WDOG*n*[STATE] indicates the number of timeouts since the last core poke, and CIU_WDOG*n*[CNT]×256 is the cycles left before the next timeout. CIU_WDOG*n*[CNT] normally decrements once every 256 cycles when the watchdog is enabled. CIU_WDOG*n*[LEN]×65536 is the timeout in cycles. A core poke for a watchdog is a write of any value to the corresponding CIU_PP_POKE*n* CSR.

Each of the four watchdogs is in one of four modes, selected by CIU_WDOG*n*[MODE]:

- OFF (MODE = 0x0) – No interrupt/non-maskable interrupt/soft-reset is ever sourced by the watchdog.
- INTERRUPT ONLY (MODE = 0x1) – An interrupt asserts whenever there has been a timeout (i.e. whenever CIU_WDOG*n*[STATE] ≠ 0).
- INTERRUPT + NMI MODE (MODE = 0x2) – An interrupt asserts whenever there has been a timeout (i.e. whenever CIU_WDOG*n*[STATE] ≠ 0). An NMI pulse is generated for the associated core whenever a second timeout occurs (i.e. whenever CIU_WDOG*n*[STATE] is set to 2).
- INTERRUPT + NMI + SOFT-RESET (MODE = 0x3) – An interrupt asserts whenever there has been a timeout (i.e. whenever CIU_WDOG*n*[STATE] ≠ 0). An NMI pulse is generated for the associated core whenever a second timeout occurs (i.e. whenever CIU_WDOG*n*[STATE] is set to 2). A chip-wide soft-reset pulse is generated whenever a third timeout occurs (i.e. whenever CIU_WDOG*n*[STATE] would otherwise be set to 3).

Cavium Confidential For PowerVang ALU 02/27/2014

The watchdogs are integrated with CN71XX's debug features in the following way:

- When CIU_WDOG*n*[DSTOP] is set, the watchdog stops counting (i.e. CIU_WDOG*n*[CNT] does not decrement) whenever the corresponding core is in debug mode (i.e. whenever CIU_PP_DBG[PPDBG*n*] is set).
- When CIU_WDOG*n*[GSTOPEN] is set, the watchdog stops counting (i.e. CIU_WDOG*n*[CNT] does not decrement) whenever CN71XX is in global-stop mode (i.e. whenever CIU_GSTOP[GSTOP] is set).

Refer to Section 2.22 for more information regarding CN71XX debug.

There are three topics that deserve special mention with respect to the use of the CN71XX watchdog timers:

- delayed requests for new work
- long tag switch waits
- delayed Fetch&Add requests

These topics are discussed in the following sections.

Delayed Requests for New Work

The POW hardware can optionally delay a request for new work. When software selects this option, if work is not immediately available in one of the groups that the core is in, the hardware will hold onto the request until work comes available or the request times out. Without timeouts, a core could hang forever waiting for work, causing a watchdog expiration. With delayed work timeouts that are small relative to the watchdog expirations, as would be desired for this and other reasons, the software delayed work timeout handler can issue core-poke requests. The software may need to occasionally issue core-poke requests on timeouts when continuously waiting for work.

Refer to Chapter 7 and the POW_NW_TIM for more details.

Long Tag Switch Waits

Depending on software configuration, it may be possible for the delay of a tag switch wait to cause a watchdog expiration. In these situations, it may be necessary for the software handler that is polling for the tag-switch completion to occasionally issue core pokes. (In some cases, the preferred behavior may be to time out in this case, though.)

Refer to Chapter 7 for more information regarding tag switches.

Delayed Fetch&Add Requests

The CN71XX hardware can optionally delay a Fetch&Add request. When software selects this option, the hardware either delays the operation until the prior tag switch completes, or times out the request. This is very similar to the delayed new work requests mentioned above. The solution could be the same also. But the solution may also be different since the software may not continuously resubmit the requests; it may instead fall into a tag switch wait.

Refer to Section 5.4 and the IOB_FAU_TIMEOUT for more details regarding FAU requests.

Cavium Confidential For Bowen Yang 02/2/2014

13.4 Ten General Timers

CIU contains ten general timers configured via CIU_TIM(0..9). Each timer can either be a one-shot timer or a periodic timer.

The three uses of the timers are:

- **Disabled**
  - Set CIU_TIM*n*[LEN] to zero.
- **One-shot**
  - Set CIU_TIM*n*[ONE_SHOT] to one.
  - Set CIU_TIM*n*[LEN] to the number of cycles.

  CIU_INT(0..11,32)_SUM0[TIMER*n*] is set when the timer expires, after which the timer becomes idle.
- **Periodic**
  - Set CIU_TIM*n*[ONE_SHOT] to zero.
  - Set CIU_TIM*n*[LEN] to the number of cycles–1 between periodic timeouts.

  CIU_INT(0..11,32)_SUM0[TIMER*n*] is set every time the timer expires (i.e. after exactly CIU_TIMn[LEN]+1 cycles each time).

After timer *n* generates a timer expiration (periodic or one-shot), CIU_INT(0..11,32)_SUM0[TIMER*n*] should be cleared by software so that the next expiration from timer *n* can cause a new interrupt.

13.5 Core Availability and Reset

The number of cores available on CN71XX is the number of bits set in CIU_FUSE. Note that the set bits may not be contiguous.

CIU_PP_RST indicates the cores currently in reset on CN71XX. (When a bit is set, the corresponding core in CN71XX is in reset.) Except for the EJTAG TAP control register processor reset bit (ECR[PrRst]) local to each core, CIU_PP_RST is the sole reset source for the cores. Refer to the MIPS specifications and Chapter 2.

Note that the reset value of CIU_PP_RST depends on the external pin REMOTE_BOOT.

- When the external pin REMOTE_BOOT is asserted, all cores are held in reset after any chip hard or soft reset (i.e. CIU_PP_RST resets to 0x3F).
- When the external pin REMOTE_BOOT is deasserted, core 0 is not held in reset after any chip hard or soft reset, but other cores are (i.e. CIU_PP_RST resets to 0x3E).

Cores are taken out of reset by writing a 0 to the corresponding bit in CIU_PP_RST. This can be done either by a core or by a remote PCIe host.

Cores can be put into reset, but CN71XX's stability cannot be guaranteed if the core is in the middle of any CMI, I/O, or low-latency transaction at the time of the reset assertion for the core.

13.6 Core Debug-Mode Observability

CIU_PP_DBG returns the current debug mode state (i.e. Debug[DM]) of the cores. Refer to the MIPS specifications, Chapter 2, and Section 2.14.

Cavium Confidential For Bowen Yang ALU 07/27/2014

13.7 Core Debug-Interrupt Generation

Writing 1s to CIU_DINT cause the corresponding cores to enter debug mode, if they are not already in debug mode. CIU_DINT is one of the sources for core debug interrupts. Also included are the EJTAG TAP control register break bit (ECR[EjtagBrk]) local to each core, and MCD interrupts from the cores and OCLA. Refer to the MIPS specifications, Chapter 2, Section 2.13, and Chapter 6 (for OCLA) for further documentation.

13.8 Core Non-Maskable Interrupt Generation

Writing 1s to CIU_NMI cause NMI pulses to be sent to the corresponding cores. Except for CIU watchdog timers, described in Section 13.3, CIU_NMI is the sole source of core nonmaskable interrupts. Refer to the MIPS specifications and Chapter 2 for more details regarding non-maskable interrupts to the CN71XX cores.

13.9 Chip Soft-Reset Initiation

To initiate a chip-wide soft reset, core or remote host software should read CIU_SOFT_RST, then write a 1 to CIU_SOFT_RST[SOFT_RST] to initiate the soft reset.

When all RST_CTL0/1/2[HOST_MODE] = 1, the effects of chip soft reset are similar to chip cold and warm resets. (A chip cold reset occurs whenever PLL_DC_OK signal is deasserted. A chip warm reset occurs whenever CHIP_RESET_L signal asserts and deasserts while PLL_DC_OK remains asserted.) One difference is that RST_BOOT[LBOOT<8>] is set to indicate the chip soft reset. Software can consult RST_BOOT[LBOOT] values to determine the cause of any chip reset.

When RST_CTL0[HOST_MODE] = 0, MAC port 0 is not reset during a chip soft reset. This means that, in this case:

- when the port is PCIe, PCICONFIGEP0-type and PEM0* RSL-type registers are not reset during a soft reset

When RST_CTL0[HOST_MODE] = 0 and a remote host initiates the CN71XX chip soft reset (via a store operation to CIU_SOFT_RST), CN71XX can only service PCICONFIGEP0 read/write operations from a remote host until it fully completes its internal reset sequence. During this time, the remote host should not access any other CN71XX resources. CN71XX can take as long as 1 ms to complete the reset sequence after it receives the CIU_SOFT_RST store. Refer to Section 20.5.

Cavium Confidential For Bowen Yang NLU 5/27/2011

13.10 CIU Registers

The CIU registers are listed in Table 13–1.

Table 13–1 CIU Registers

Registers	Address[1]	CSR Type[2]	Detailed Description
CIU_INT(0..7, 32)_SUM0	0x1070000000000 + α * 0x8	NCB	See page 745
CIU_INT_SUM1	0x1070000000108	NCB	See page 747
CIU_INT(0..7,32)_EN0	0x1070000000200 + α * 0x10	NCB	See page 748
CIU_INT(0..7,32)_EN1	0x1070000000208 + α * 0x10	NCB	See page 749
CIU_TIM(0..9)	0x1070000000480 + α * 0x8	NCB	See page 750
CIU_TIM_MULTI_CAST	0x10700000004F0	NCB	See page 750
CIU_WDOG(0..3)	0x1070000000500 + α * 0x8	NCB	See page 750
CIU_PP_POKE(0..3)	0x1070000000580 + α * 0x8	NCB	See page 751
CIU_MBOX_SET(0..3)	0x1070000000600 + α * 0x8	NCB	See page 751
CIU_MBOX_CLR(0..3)	0x1070000000680 + α * 0x8	NCB	See page 751
CIU_PP_RST	0x1070000000700	NCB	See page 751
CIU_PP_DBG	0x1070000000708	NCB	See page 752
CIU_GSTOP	0x1070000000710	NCB	See page 752
CIU_NMI	0x1070000000718	NCB	See page 752
CIU_DINT	0x1070000000720	NCB	See page 752
CIU_FUSE	0x1070000000728	NCB	See page 752
CIU_BIST	0x1070000000730	NCB	See page 753
CIU_SOFT_BIST	0x1070000000738	NCB	See page 753
CIU_PP_RST_PENDING	0x1070000000740	NCB	See page 753
CIU_PCI_INTA	0x1070000000750	NCB	See page 753
CIU_INTR_SLOWDOWN	0x10700000007D0	NCB	See page 754
CIU_INT(0..3)_SUM4	0x1070000000C00 + α * 0x8	NCB	See page 755
CIU_INT(0..3)_EN4_0	0x1070000000C80 + α * 0x10	NCB	See page 757
CIU_INT(0..3)_EN4_1	0x1070000000C88 + α * 0x10	NCB	See page 758
CIU_INT(0..7,32)_EN0_W1C	0x1070000002200 + α * 0x10	NCB	See page 759
CIU_INT(0..7,32)_EN1_W1C	0x1070000002208 + α * 0x10	NCB	See page 760
CIU_INT(0..3)_EN4_0_W1C	0x1070000002C80 + α * 0x10	NCB	See page 761
CIU_INT(0..3)_EN4_1_W1C	0x1070000002C88 + α * 0x10	NCB	See page 762
CIU_INT(0..7,32)_EN0_W1S	0x1070000006200 + α * 0x10	NCB	See page 763
CIU_INT(0..7,32)_EN1_W1S	0x1070000006208 + α * 0x10	NCB	See page 764
CIU_INT(0..3)_EN4_0_W1S	0x1070000006C80 + α * 0x10	NCB	See page 765
CIU_INT(0..3)_EN4_1_W1S	0x1070000006C88 + α * 0x10	NCB	See page 766
CIU_SUM1_PP(0..3)_IP2	0x1070000008000 + α * 0x8	NCB	See page 768
CIU_SUM1_PP(0..3)_IP3	0x1070000008200 + α * 0x8	NCB	See page 768
CIU_SUM1_PP(0..3)_IP4	0x1070000008400 + α * 0x8	NCB	See page 768
CIU_SUM1_IO0_INT	0x1070000008600	NCB	See page 768
CIU_SUM2_PP(0..3)_IP2	0x1070000008800 + α * 0x8	NCB	See page 770
CIU_SUM2_PP(0..3)_IP3	0x1070000008A00 + α * 0x8	NCB	See page 770
CIU_SUM2_PP(0..3)_IP4	0x1070000008C00 + α * 0x8	NCB	See page 770
CIU_SUM2_IO0_INT	0x1070000008E00	NCB	See page 770

Cavium Confidential For ALU Bowen Yang 02/27/2014

Table 13–1 CIU Registers (Continued)

Registers	Address[1]	CSR Type[2]	Detailed Description
CIU_EN2_PP(0..3)_IP2	0x107000000A000 + α * 0x8	NCB	See page 771
CIU_EN2_PP(0..3)_IP3	0x107000000A200 + α * 0x8	NCB	See page 771
CIU_EN2_PP(0..3)_IP4	0x107000000A400 + α * 0x8	NCB	See page 771
CIU_EN2_IO0_INT	0x107000000A600	NCB	See page 771
CIU_EN2_PP(0..3)_IP2_W1S	0x107000000A800 + α * 0x8	NCB	See page 771
CIU_EN2_PP(0..3)_IP3_W1S	0x107000000AA00 + α * 0x8	NCB	See page 771
CIU_EN2_PP(0..3)_IP4_W1S	0x107000000AC00 + α * 0x8	NCB	See page 771
CIU_EN2_IO0_INT_W1S	0x107000000AE00	NCB	See page 771
CIU_EN2_PP(0..3)_IP2_W1C	0x107000000C800 + α * 0x8	NCB	See page 772
CIU_EN2_PP(0..3)_IP3_W1C	0x107000000CA00 + α * 0x8	NCB	See page 772
CIU_EN2_PP(0..3)_IP4_W1C	0x107000000CC00 + α * 0x8	NCB	See page 772
CIU_EN2_IO0_INT_W1C	0x107000000CE00	NCB	See page 772
CIU_CIB_L2C_RAW(0)	0x107000000E000 + α * 0x10	NCB	See page 773
CIU_CIB_L2C_EN(0)	0x107000000E100 + α * 0x10	NCB	See page 775
CIU_CIB_LMC(0)_RAW(0)	0x107000000E200 + α * 0x100 + β * 0x10	NCB	See page 776
CIU_CIB_LMC(0)_EN(0)	0x107000000E300 + α * 0x100 + β * 0x10	NCB	See page 776
CIU_CIB_RST_RAW(0)	0x107000000E400 + α * 0x10	NCB	See page 777
CIU_CIB_RST_EN(0)	0x107000000E500 + α * 0x10	NCB	See page 777
CIU_CIB_SATA_RAW(0)	0x107000000E600 + α * 0x10	NCB	See page 778
CIU_CIB_SATA_EN(0)	0x107000000E700 + α * 0x10	NCB	See page 778
CIU_CIB_USBDRD(0..1)_RAW(0)	0x107000000E800 + α * 0x100 + β * 0x10	NCB	See page 779
CIU_CIB_USBDRD(0..1)_EN(0)	0x107000000EA00 + α * 0x100 + β * 0x10	NCB	See page 781
CIU_CIB_OCLA(0)_RAW(0)	0x107000000EC00 + α * 0x100 + β * 0x10	NCB	See page 782
CIU_CIB_OCLA(0)_EN(0)	0x107000000EE00 + α * 0x100 + β * 0x10	NCB	See page 783

1. For registers with multiple versions, α represents the values within the specified range and β the second range
2. NCB-type registers are accessed directly across the IOI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

CIU Interrupt Summary 0 Registers
CIU_INT(0..7, 32)_SUM0

Indicates where an interrupt has occurred. The nine registers are apportioned in the following manner:

CIU_INT0_SUM0 is used for core0/IP2
CIU_INT1_SUM0 is used for core0/IP3
CIU_INT2_SUM0 is used for core1/IP2
CIU_INT3_SUM0 is used for core1/IP3
...
CIU_INT6_SUM0 is used for core3/IP2
CIU_INT7_SUM0 is used for core3/IP3
CIU_INT32_SUM0 are used for PCI/INTA

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	BOOTDMA	RO/H	0	0	Boot bus DMA engines interrupt. See MIO_NDF_DMA_INT
<62>	—	RO/H	—	—	Reserved.
<61>	IPDPPTHR	RO/H	0	0	IPD per-port counter threshold interrupt. See IPD_PORT_QOS_INT*.
<60>	POWIQ	RO/H	0	0	POW IQ interrupt. See POW_IQ_INT.
<59>	TWSI2	RO/H	0	0	Second TWSI interrupt. See MIO_TWS1_INT.
<58>	MPI	R/W1C/H	0	0	MPI/SPI interrupt. Set when MPI transaction is finished; see MPI_CFG[INT_ENA] and MPI_STS[BUSY].
<57>	PCM	RO	0	0	PCM/TDM interrupt
<56>	—	RAZ	—	—	Reserved.
<55:52>	TIMER	R/W1C/H	0x0	0x0	General timer 0–3 interrupts. When CIU_TIM_MULTI_CAST[EN] == 0, this interrupt is common for all cores/IRQs; writing 1 to any core/IRQ clears all TIMER*x* (*x*=0..9) interrupts. When CIU_TIM_MULTI_CAST[EN] = 1, TIMER*x* (*x*=0..9) are set at the same time, but clearing is per cnMIPS core. See CIU_TIM_MULTI_CAST for details. The combination of this field and the CIU_SUM2_*[TIMER] field implement all 10 CIU_TIM* interrupts.
<51>	SUM2	RO/H	0	0	SUM2 & EN2 summary bit. This read-only bit reads as a 1 whenever any CIU_SUM2_PP(0..3)_IP*n* or CIU_SUM2_IO0_INT bit is set and corresponding enable bit in CIU_EN2_PP(0..3)_IP*x* or CIU_EN2_IO0_INT is set. Note that SUM2 only summarizes the SUM2/EN2 result and does not have a corresponding enable bit, so does not directly contribute to interrupts.
<50>	IPD_DRP	R/W1C/H	0	0	IPD QOS packet drop interrupt. Set any time PIP/IPD drops a packet.
<49:48>	GMX_DRP	R/W1C/H	0x0	0x0	GMX0/1 packet drop interrupt. Set any time corresponding GMX0/1 drops a packet.
<47:46>	—	RO	—	—	Reserved.
<45>	TWSI	RO/H	0	0	TWSI Interrupt. See MIO_TWS0_INT.
<44>	WDOG_SUM	RO/H	0	0	SUM1 & EN1 summary bit. This read-only bit reads as a one whenever any CIU_SUM1_PP(0..3)_IP*x* or CIU_SUM2_IO0_INT bit is set and corresponding enable bit in CIU_EN2_PP(0..3)_IP*x* or CIU_EN2_IO0_INT is set. • Cores use CIU_INT(0..7)_SUM0 where *x* = 0–7. • PCIe uses the CIU_INT32_SUM0 where *x* = 32. Note that WDOG_SUM only summarizes the SUM1/EN1 result and does not have a corresponding enable bit, so does not directly contribute to interrupts.
<43:40>	PCI_MSI	RO/H	0x0	0x0	PCIe MSI. See SLI_MSI_RCV*n* for bit <40+*n*>.

Cavium Confidential For Power A10 02/16/16

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<39:36>	PCI_INT	RO/H	0x0	0x0	PCIe INTA/B/C/D. Refer to Section 12.6.1, CN61XX Receiving Emulated INTA/INTB/INTC/INTD. PCI_INT<3> = INTD PCI_INT<2> = INTC PCI_INT<1> = INTB PCI_INT<0> = INTA
<35:34>	UART	RO/H	0x0	0x0	Two UART interrupts. See MIO_UARTn_IIR[IID] for bit <34+*n*>.
<33:32>	MBOX	RO/H	0x0	0x0	Two mailbox interrupts for entries 0–7. <33> is the OR of <31:16> <32> is the OR of <15:0> Two PCIe internal interrupts for entries 32–33 which equal CIU_PCI_INTA[INT]. CIU_INT33_SUM0[MBOX] is a read-only copy of CIU_PCI_INTA[INT].
<31:16>	GPIO	R/W1C/H	0x0	0x0	Sixteen GPIO interrupts. When GPIO_MULTI_CAST[EN] = 1, write 1 to clear either the per core or common GPIO edge-triggered interrupts, depending on the mode. See GPIO_MULTI_CAST for details. When GPIO_MULTI_CAST[EN] = 0, this field is read-only.
<15:0>	WORKQ	RO/H	0x0	0x0	Sixteen work-queue interrupts. See POW_WQ_INT[WQ_INT]. One bit/group. A copy of the R/W1C bit in the POW.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

CIU Interrupt Summary 1 Register
CIU_INT_SUM1

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	RST	RO/H	0	0	MIO RST interrupt. See RST_INT.
<62:53>	—	RAZ	—	—	Reserved
<52>	LMC0	RO/H	0	0	LMC0 interrupt. (CIU_CIB_LMC(0)_RAW[0] AND CIU_CIB_LMC(0)_EN[0]). See CIU_CIB_LMC(0)_RAW[0] and CIU_CIB_LMC(0)_EN[0].
<51>	—	RAZ	—	—	Reserved
<50>	PEM2	RO/H	0	0	PEM2 interrupt. See PEM2_INT_SUM (enabled by PEM2_INT_ENB).
<49>	PEM1	RO/H	0	0	PEM1 interrupt. See PEM1_INT_SUM (enabled by PEM1_INT_ENB).
<48>	PEM0	RO/H	0	0	PEM0 interrupt. See PEM0_INT_SUM (enabled by PEM0_INT_ENB).
<47>	PTP	R/W1C/H	0	0	PTP interrupt. Set when hardware decrements MIO_PTP_EVT_CNT to 0.
<46>	AGL	RO/H	0	0	AGL interrupt. See AGL_GMX_RX0/1_INT_REG, AGL_GMX_TX_INT_REG.
<45:38>	—	RAZ	—	—	Reserved
<37>	AGX1	RO/H	0	0	GMX1 interrupt. See GMX1_RX0/1_INT_REG, GMX1_TX_INT_REG PCS1_INT(0..3)_REG, PCSX1_INT_REG.
<36>	AGX0	RO/H	0	0	GMX0 interrupt. See GMX0_RX0/1_INT_REG, GMX0_TX_INT_REG PCS0_INT(0..3)_REG, PCSX0_INT_REG.
<35>	DPI	RO/H	0	0	DPI interrupt. See DPI_INT_REG.
<34>	SLI	RO/H	0	0	SLI interrupt. See SLI_INT_SUM (enabled by SLI_INT_ENB_CIU).
<33>	USB	RO/H	0	0	USBDRD0 interrupt. (CIU_CIB_USBDRD(0)_RAW[0] AND CIU_CIB_USBDRD(0)_EN[0]). See CIU_CIB_USBDRD(0)_RAW[0] and CIU_CIB_USBDRD(0)_EN[0].
<32>	DFA	RO/H	0	0	HFA interrupt. See DFA_ERROR.
<31>	KEY	RO/H	0	0	KEY interrupt. See KEY_INT_SUM.
<30>	RAD	RO/H	0	0	RAD interrupt. See RAD_REG_ERROR.
<29>	TIM	RO/H	0	0	TIM interrupt. See TIM_REG_ERROR.
<28>	—	RAZ	—	—	Reserved
<27>	PKO	RO/H	0	0	PKO interrupt. See PKO_REG_ERROR.
<26>	PIP	RO/H	0	0	PIP interrupt. See PIP_INT_REG.
<25>	IPD	RO/H	0	0	IPD interrupt. See IPD_INT_SUM.
<24>	L2C	RO/H	0	0	L2C interrupt. See L2C_INT_REG.
<23>	POW	RO/H	0	0	POW err interrupt. See POW_ECC_ERR.
<22>	FPA	RO/H	0	0	FPA interrupt. See FPA_INT_SUM.
<21>	IOB	RO/H	0	0	IOB interrupt. See IOB_INT_SUM.
<20>	MIO	RO/H	0	0	MIO boot interrupt. See MIO_BOOT_ERR.
<19>	NAND	RO/H	0	0	eMMC flash controller interrupt. See MIO_EMM_INT.
<18>	—	RAZ	—	—	Reserved

Cavium Confidential For Review Only 07/7/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<17>	USB1	RO/H	0	0	USBDRD1 interrupt. (CIU_CIB_USBDRD(1)_RAW[0] AND CIU_CIB_USBDRD(1)_EN[0]). See CIU_CIB_USBDRD(1)_RAW[0] and CIU_CIB_USBDRD(1)_EN[0].
<16:4>	—	RO/H	—	—	Reserved
<3:0>	WDOG	RO/H	0x0	0x0	Watchdog interrupts. Bit<0> for core0 watchdog, and bit<n> for core*n*

CIU Interrupt Enable 0 Registers
CIU_INT(0..7,32)_EN0

Provide enable bits for interrupts. The nine registers are apportioned in the following manner:

CIU_INT0_EN0 is used for core0/IP2
CIU_INT1_EN0 is used for core0/IP3
CIU_INT2_EN0 is used for core1/IP2
CIU_INT3_EN0 is used for core1/IP3
...
CIU_INT6_EN0 is used for core3/IP2
CIU_INT7_EN0 is used for core3/IP3
CIU_INT32_EN0 is used for PCI/INTA

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	BOOTDMA	R/W	0	0	Boot bus DMA engines interrupt enable
<62>	—	RAZ	—	—	Reserved.
<61>	IPDPPTHR	R/W	0	0	IPD per-port counter threshold interrupt enable
<60>	POWIQ	R/W	0	0	POW IQ interrupt enable
<59>	TWSI2	R/W	0	0	Second TWSI interrupt enable
<58>	MPI	R/W	0	0	MPI/SPI interrupt
<57>	PCM	R/W	0	0	PCM/TDM interrupt
<56>	—	RAZ	—	—	Reserved.
<55:52>	TIMER	R/W	0x0	0x0	General timer interrupt enables
<51>	—	RAZ	—	—	Reserved.
<50>	IPD_DRP	R/W	0	0	IPD QOS packet drop interrupt enable
<49:48>	GMX_DRP	R/W	0x0	0x0	GMX packet drop interrupt enable
<47:46>	—	RAZ	—	—	Reserved.
<45>	TWSI	R/W	0	0	TWSI interrupt enable
<44>	—	RAZ	—	—	Reserved
<43:40>	PCI_MSI	R/W	0x0	0x0	PCIe MSI enables
<39:36>	PCI_INT	R/W	0x0	0x0	PCIe INTA/B/C/D enables
<35:34>	UART	R/W	0x0	0x0	Two UART interrupt enables
<33:32>	MBOX	R/W	0x0	0x0	Two mailbox/PCIe interrupt enables
<31:16>	GPIO	R/W	0x0	0x0	16 GPIO interrupt enables
<15:0>	WORKQ	R/W	0x0	0x0	16 work queue interrupt enables

Cavium Confidential For Boren Yang ALU 10/27/2014

CIU Interrupt Enable 1 Registers
CIU_INT(0..7,32)_EN1

Provide enable bits for watchdog vectors. The 10 registers are apportioned in the following manner:

CIU_INT0_EN1 is used for core0/IP2
CIU_INT1_EN1 is used for core0/IP3
CIU_INT2_EN1 is used for core1/IP2
CIU_INT3_EN1 is used for core1/IP3
...
CIU_INT6_EN1 is used for core3/IP2
CIU_INT7_EN1 is used for core3/IP3
CIU_INT32_EN1 is used for PCI/INTA

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	RST	R/W	0x0	0x0	MIO RST interrupt enable
<62:53>	—	RAZ	—	—	Reserved
<52>	LMC0	R/W	0	0	LMC0 interrupt enable
<51>	—	RAZ	—	—	Reserved
<50>	PEM2	R/W	0	0	PEM2 interrupt enable
<49>	PEM1	R/W	0	0	PEM1 interrupt enable
<48>	PEM0	R/W	0	0	PEM0 interrupt enable
<47>	PTP	R/W	0	0	PTP interrupt enable
<46>	AGL	R/W	0	0	AGL interrupt enable
<45:41>	—	R/W	0x0	0x0	Reserved
<40>	DPI_DMA	R/W	0	0	DPI DMA interrupt enable
<39:38>	—	RAZ	—	—	Reserved
<37>	AGX1	R/W	0	0	GMX1 interrupt enable
<36>	AGX0	R/W	0	0	GMX0 interrupt enable
<35>	DPI	R/W	0	0	DPI interrupt enable
<34>	SLI	R/W	0	0	SLI interrupt enable
<33>	USB	R/W	0	0	USBDRD0 interrupt enable
<32>	DFA	R/W	0	0	DFA interrupt enable
<31>	KEY	R/W	0	0	KEY interrupt enable
<30>	RAD	R/W	0	0	RAD interrupt enable
<29>	TIM	R/W	0	0	TIM interrupt enable
<28>	—	RAZ	—	—	Reserved
<27>	PKO	R/W	0	0	PKO interrupt enable
<26>	PIP	R/W	0	0	PIP interrupt enable
<25>	IPD	R/W	0	0	IPD interrupt enable
<24>	L2C	R/W	0	0	L2C interrupt enable
<23>	POW	R/W	0	0	POW err interrupt enable
<22>	FPA	R/W	0	0	FPA interrupt enable
<21>	IOB	R/W	0	0	IOB interrupt enable
<20>	MIO	R/W	0	0	MIO boot interrupt enable
<19>	NAND	R/W	0	0	eMMC flash controller interrupt enable
<18>	—	RAZ	—	—	Reserved
<17>	USB1	R/W	0	0	USBDRD1 interrupt enable
<16:4>	—	RAZ	—	—	Reserved
<3:0>	WDOG	R/W	0	0	Watchdog summary interrupt-enable vector

Cavium Confidential For Robert Yang ALU 02/27/2014

CIU General Timer Registers
CIU_TIM(0..9)

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:37>	—	RAZ	—	—	Reserved
<36>	ONE_SHOT	R/W	0	0	One-shot mode when LEN ≠ 0x0: 1 = timer is in **one-shot** mode, 0 = timer is in **periodic** mode.
<35:0>	LEN	R/W	0x0	0x0	Time-out length in coprocessor-clock cycles. The timer disabled when LEN = 0x0.

CIU General Timer Interrupt Multicast Enable Registers
CIU_TIM_MULTI_CAST

See Table 13–1 for the addresses.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	EN	R/W	0	0	General timer interrupt multicast mode: 0 = timer interrupt is common for all PP/IRQs 1 = timer interrupts are set at the same time for all PP/IRQs, but interrupt clearings must be done individually per cnMIPS core. Timer interrupts for I/Os (32) always use common interrupts. Clearing any of the I/O interrupts also clears the common interrupt.

CIU Watchdog Registers
CIU_WDOG(0..3)

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:46>	—	RAZ	—	—	Reserved
<45>	GSTOPEN	R/W	0	0	Global-stop enable.
<44>	DSTOP	R/W	0	0	Debug-stop enable.
<43:20>	CNT	RO/H	0x0	0x0	Number of 256-cycle intervals until next watchdog expiration. Cleared on write to associated CIU_PP_POKE*n* register.
<19:4>	LEN	R/W	0x0	0x0	Watchdog time-expiration length. The most-significant 16 bits of a 24-bit decrementer that decrements every 256 cycles. Must be set > 0.
<3:2>	STATE	RO/H	0x0	0x0	Watchdog state. The number of watchdog time expirations since last core poke. Cleared on write to associated CIU_PP_POKE*n* register.
<1:0>	MODE	R/W	0x0	0x0	Watchdog mode: 0x0 = Off 0x1 = Interrupt only 0x2 = Interrupt + NMI 0x3 = Interrupt + NMI + soft reset

Cavium Confidential For Bowen Yang ALU 9/27/2014

CIU Cores Poke Registers
CIU_PP_POKE(0..3)

Write operations to these registers cause the following:

- clears any pending interrupts generated by the associated watchdog
- resets the CIU_WDOG*n*[STATE] field
- sets CIU_WDOG[CNT] to be (CIU_WDOG[LEN] << 8)

Read operations to these registers return the associated CIU_WDOG register. See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	POKE	RAZ	—	—	Reserved

CIU Mailbox Set Registers
CIU_MBOX_SET(0..3)

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	SET	R/W1S/H	0x0	0x0	Writing a 1 to a bit sets the corresponding bit in mailbox storage. Reading this register returns the contents of mailbox storage.

CIU Mailbox Clear Registers
CIU_MBOX_CLR(0..3)

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CLR	R/W1C/H	0x0	0x0	Writing a 1 to a bit clears the corresponding bit in mailbox storage. Reading this register returns the contents of mailbox storage.

CIU Cores Reset Register
CIU_PP_RST

This register contains the reset control for each core. A value of 1 holds a core in reset; 0 releases it. Resets to 0xF when PCI boot is enabled, 0xE otherwise. See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:1>	RST	R/W	0x7	0x0	Core reset for core 1. Writing a 1 holds the corresponding core in reset, writing a 0 releases from reset.
<0>	RST0	R/W	—	—	Core reset for core 0, depends on standalone mode

Cavium Confidential For Bowen Yang ALU 02/27/2014

CIU Cores Debug Register
CIU_PP_DBG

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	PPDBG	RO	0x0	0x0	Debug[DM] value for each core, whether the cores are in debug mode or not.

CIU Global-Stop Register
CIU_GSTOP

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	GSTOP	R/W	0	0	Set global-stop mode.

CIU NonMaskable Interrupt Register
CIU_NMI

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	NMI	WO	0x0	0x0	Writing a 1 to a bit sends an NMI pulse to the corresponding core vector.

CIU Debug Interrupt Register
CIU_DINT

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	DINT	WO	0x0	0x0	Each bit set sends a DINT pulse to corresponding core.

CIU Fuse Register
CIU_FUSE

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	FUSE	RO	—	—	Each bit set indicates a physical core is present.

Cavium Confidential For Bowen Yang ALU 02/27/2014

CIU BIST Register
CIU_BIST

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<2:0>	BIST	RO	0x0	0x0	BIST results. Hardware sets a bit for each memory that fails BIST.

CIU Soft BIST Register
CIU_SOFT_BIST

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	SOFT_BIST	RAZ	—	—	Reserved

CIU Cores Reset Pending Register
CIU_PP_RST_PENDING

This register contains the reset status for each core. A 1 indicated the core is waiting to change its reset state. See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	PEND	RO	0x0	0x0	Core waiting on reset to deassert complete. This register always returns zero.

CIU PCIe Interrupt Register
CIU_PCI_INTA

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	INTR	R/W	0	0	PCIe interrupt. These bits are observed in CIU_INT32_SUM0<32>.

Cavium Confidential For Bowen Yang ALU 02/27/2014

CIU Interrupt Slow Down Register
CIU_INTR_SLOWDOWN

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved.
<2:0>	CTL	R/W	0x0	0x0	Slow down CIU interrupt walker processing time. IRQ2/3/4 for all four cores are sent to the core (MRC) in a serial bus to reduce global routing. There is no backpressure mechanism designed for this scheme. It will only be a problem when the coprocessor clock is faster; this Control slows down the data send-out rate in INTR interface to the core. With different settings, clock-rate ratio can handle various SLOWDOWN coprocessor-clock frequency/core-clock frequency ratios: 0x0 = 3 0x1 = 6 0xn = 3 × 2^n

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

CIU Interrupt Summary IP4 Registers
CIU_INT(0..3)_SUM4

Indicates where an interrupt has occurred. The four registers are apportioned in the following manner:

CIU_INT0_SUM4 is used for core0/IP4
CIU_INT1_SUM4 is used for core1/IP4
...
CIU_INT3_SUM4 is used for core3/IP4

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	BOOTDMA	RO/H	0	0	Boot bus DMA engines interrupt. See MIO_BOOT_DMA_INT*, MIO_NDF_DMA_INT.
<62>	—	RAZ	—	—	Reserved.
<61>	IPDPPTHR	RO/H	0	0	IPD per-port counter threshold interrupt. See IPD_PORT_QOS_INT*.
<60>	POWIQ	RO/H	0	0	POW IQ interrupt. See POW_IQ_INT.
<59>	TWSI2	RO/H	0	0	2nd TWSI interrupt. See MIO_TWS1_INT.
<58>	MPI	R/W1C/H	0	0	MPI/SPI interrupt
<57>	PCM	RO/H	0	0	PCM/TDM interrupt
<56>	—	RAZ	—	—	Reserved.
<55:52>	TIMER	R/W1C/H	0x0	0x0	General timer 0–3 interrupts. When CIU_TIM_MULTI_CAST[EN] = 0, this interrupt is common for all cores/IRQs; writing 1 to any core/IRQ clears all TIMER*x* (*x*=0..9) interrupts. When CIU_TIM_MULTI_CAST[EN] = 1, TIMER*x* (*x*=0..9) are set at the same time, but clearing is per cnMIPS core. See CIU_TIM_MULTI_CAST for details. The combination of this field and the CIU_SUM2_*[TIMER] field implement all 10 CIU_TIM* interrupts.
<51>	SUM2	RO/H	0	0	SUM2 & EN2 summary bit. This read-only bit reads as a one whenever any CIU_SUM2_PP(0..3)_IP*x* (CIU_SUM2_IO0_INT) bit is set and corresponding enable bit in CIU_EN2_PP(0..3)_IP*x* (CIU_EN2_IO0_INT) is set. Note that SUM2 only summarizes the SUM2/EN2 result and does not have a corresponding enable bit, so does not directly contribute to interrupts.
<50>	IPD_DRP	R/W1C/H	0	0	IPD QOS packet drop interrupt. Set any time PIP/IPD drops a packet.
<49>	—	RAZ	—	—	Reserved
<49:48>	GMX_DRP	R/W1C/H	0	0	GMX packet drop interrupt. Set any time corresponding GMX drops a packet.
<47:46>	—	RAZ	—	—	Reserved
<45>	TWSI	RO/H	0	0	TWSI interrupt. See MIO_TWS0_INT.
<44>	WDOG_SUM	RO/H	0	0	SUM1 & EN1 summary bit. This read-only bit reads as a one whenever any CIU_SUM1_PP(0..3)_IP*x* (CIU_SUM1_IO0_INT) bit is set and corresponding enable bit in CIU_INT0_EN is set. • Cores use CIU_INT*x*_SUM0 where *x* = 0–7. • PCIe uses the CIU_INT0_SUM0 where *x* = 32. Note that WDOG_SUM only summarizes the SUM1/EN1 result and does not have a corresponding enable bit, so does not directly contribute to interrupts.
<43:40>	PCI_MSI	RO/H	0x0	0x0	PCIe MSI. See SLI_MSI_RCV*n* for bit <40+*n*>.

Cavium Confidential For Beijen Ren 02/17/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<39:36>	PCI_INT	RO/H	0x0	0x0	PCIe INTA/B/C/D. Refer to Section 12.6.1, CN61XX Receiving Emulated INTA/INTB/INTC/INTD. PCI_INT<3> = INTD PCI_INT<2> = INTC PCI_INT<1> = INTB PCI_INT<0> = INTA
<35:34>	UART	RO/H	0x0	0x0	Two UART interrupts. See MIO_UART*n*_IIR[IID] for bit <34+*n*>.
<33:32>	MBOX	RO/H	0x0	0x0	Two mailbox interrupts for entries 0–5. <33> is the OR of <31:16> <32> is the OR of <15:0>
<31:16>	GPIO	R/W1C/H	0x0	0x0	Sixteen GPIO interrupts. When GPIO_MULTI_CAST[EN] == 1, write 1 to clear either the per core or common GPIO edge-triggered interrupts, depending on the mode. See GPIO_MULTI_CAST for details. When GPIO_MULTI_CAST[EN] == 0, this field is read-only.
<15:0>	WORKQ	RO/H	0x0	0x0	Sixteen work-queue interrupts. See POW_WQ_INT[WQ_INT]. One bit/group. A copy of the R/W1C bit in the SSO.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

CIU Interrupt Enable IP4 Registers 0
CIU_INT(0..3)_EN4_0

Provide enable bits for interrupts. The four registers are apportioned in the following manner:

CIU_INT0_EN4_0 is used for core0/IP4
CIU_INT1_EN4_0 is used for core1/IP4
...
CIU_INT3_EN4_0 is used for core3/IP4

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	BOOTDMA	R/W	0	0	Boot bus DMA engines interrupt enable.
<62>	—	RAZ	—	—	Reserved
<61>	IPDPPTHR	R/W	0	0	IPD per-port counter threshold interrupt enable.
<60>	POWIQ	R/W	0	0	POW IQ interrupt enable.
<59>	TWSI2	R/W	0	0	Second TWSI interrupt enable.
<58>	MPI	R/W	0	0	MPI/SPI interrupt enable.
<57>	PCM	R/W	0	0	PCM/TDM interrupt enable.
<56>	—	RAZ	—	—	Reserved
<55:52>	TIMER	R/W	0x0	0x0	General timer interrupt enables.
<51>	—	RAZ	—	—	Reserved
<50>	IPD_DRP	R/W	0	0	IPD QOS packet drop interrupt enable.
<49:48>	GMX_DRP	R/W	0	0	GMX packet-drop interrupt enable.
<47:46>	—	RAZ	—	—	Reserved
<45>	TWSI	R/W	0	0	TWSI interrupt enable.
<44>	—	RAZ	—	—	Reserved
<43:40>	PCI_MSI	R/W	0x0	0x0	PCIe MSI enables.
<39:36>	PCI_INT	R/W	0x0	0x0	PCIe INTA/B/C/D enables.
<35:34>	UART	R/W	0x0	0x0	Two UART interrupt enables.
<33:32>	MBOX	R/W	0x0	0x0	Two mailbox interrupt enables.
<31:16>	GPIO	R/W	0x0	0x0	16 GPIO interrupt enables.
<15:0>	WORKQ	R/W	0x0	0x0	16 work-queue interrupt enables.

Cavium Confidential For Bowen Yang ALU 02/27/2014

CIU Interrupt Enable IP4 Registers 1
CIU_INT(0..3)_EN4_1

Provide enable bits for watchdog vectors. The four registers are apportioned in the following manner:

CIU_INT0_EN4_1 is used for core0/IP4
CIU_INT1_EN4_1 is used for core1/IP4
...
CIU_INT3_EN4_1 is used for core3/IP4

See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	RST	R/W	0	0	MIO RST interrupt enable
<62:53>	—	RAZ	—	—	Reserved
<52>	LMC0	R/W	0	0	LMC0 interrupt enable
<51>	—	RAZ	—	—	Reserved
<50>	PEM2	R/W	0	0	PEM2 interrupt enable
<49>	PEM1	R/W	0	0	PEM1 interrupt enable
<48>	PEM0	R/W	0	0	PEM0 interrupt enable
<47>	PTP	R/W	0	0	PTP interrupt enable
<46>	AGL	R/W	0	0	AGL interrupt enable
<45:41>	—	RAZ	—	—	Reserved
<40>	DPI_DMA	R/W	0	0	DPI DMA interrupt enable. See DPI DMA instruction completion.
<39:38>	—	RAZ	—	—	Reserved
<37>	AGX1	R/W	0	0	GMX1 interrupt enable
<36>	AGX0	R/W	0	0	GMX0 interrupt enable
<35>	DPI	R/W	0	0	DPI interrupt enable
<34>	SLI	R/W	0	0	SLI interrupt enable
<33>	USB	R/W	0	0	USBDRD0 interrupt enable
<32>	DFA	R/W	0	0	DFA interrupt enable
<31>	KEY	R/W	0	0	KEY interrupt enable
<30>	RAD	R/W	0	0	RAD interrupt enable
<29>	TIM	R/W	0	0	TIM interrupt enable
<28>	—	RAZ	—	—	Reserved
<27>	PKO	R/W	0	0	PKO interrupt enable
<26>	PIP	R/W	0	0	PIP interrupt enable
<25>	IPD	R/W	0	0	IPD interrupt enable
<24>	L2C	R/W	0	0	L2C interrupt enable
<23>	POW	R/W	0	0	POW err interrupt enable
<22>	FPA	R/W	0	0	FPA interrupt enable
<21>	IOB	R/W	0	0	IOB interrupt enable
<20>	MIO	R/W	0	0	MIO boot interrupt enable
<19>	NAND	R/W	0	0	eMMC flash controller interrupt enable
<18>	—	RAZ	—	—	Reserved
<17>	USB1	R/W	0	0	USBDRD1 interrupt enable
<16:4>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<3:0>	WDOG	R/W	0x0	0x0	Watchdog summary interrupt-enable vector

CIU Write-1-to-Clear Interrupt Enable 0 Registers
CIU_INT(0..7,32)_EN0_W1C

This is the write-1-to-clear version of the CIU_INT(0..7,32)_EN0 registers. See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	BOOTDMA	R/W1C/H	0	0	Write 1 to clear boot bus DMA engines interrupt enable.
<62>	—	RAZ	—	—	Reserved
<61>	IPDPPTHR	R/W1C/H	0	0	Write 1 to clear IPD per-port counter threshold interrupt enable.
<60>	POWIQ	R/W1C/H	0	0	Write 1 to clear POW IQ interrupt enable.
<59>	TWSI2	R/W1C/H	0	0	Write 1 to clear second TWSI interrupt enable.
<58>	MPI	R/W1C/H	0	0	Write 1 to clear MPI/SPI interrupt enable.
<57>	PCM	R/W1C/H	0	0	Write 1 to clear PCM/TDM interrupt enable.
<56>	—	RAZ	—	—	Reserved
<55:52>	TIMER	R/W1C/H	0x0	0x0	Write 1 to clear general timer interrupt enables.
<51>	—	RAZ	—	—	Reserved
<50>	IPD_DRP	R/W1C/H	0	0	Write 1 to clear IPD QOS packet drop interrupt enable.
<49>	—	RAZ	—	—	Reserved
<48>	GMX_DRP	R/W1C/H	0	0	Write 1 to clear GMX packet drop interrupt enable
<47:46>	—	RAZ	—	—	Reserved
<45>	TWSI	R/W1C/H	0	0	Write 1 to clear TWSI interrupt enable.
<44>	—	RAZ	—	—	Reserved
<43:40>	PCI_MSI	R/W1C/H	0x0	0x0	Write 1s to clear PCIe MSI enables.
<39:36>	PCI_INT	R/W1C/H	0x0	0x0	Write 1s to clear PCIe INTA/B/C/D enables.
<35:34>	UART	R/W1C/H	0x0	0x0	Write 1s to clear UART interrupt enables.
<33:32>	MBOX	R/W1C/H	0x0	0x0	Write 1s to clear mailbox/PCIe interrupt enables.
<31:16>	GPIO	R/W1C/H	0x0	0x0	Write 1s to clear GPIO interrupt enables.
<15:0>	WORKQ	R/W1C/H	0x0	0x0	Write 1s to clear work queue interrupt enables.

Cavium Confidential For Bowen Yang ALU 07/2014

CIU Write-1-to-Clear Interrupt Enable 1 Registers
CIU_INT(0..7,32)_EN1_W1C

This is the write-1-to-clear version of the CIU_INT(0..3,32)_EN1 registers, read back corresponding CIU_INT*x*_EN1 value. See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	RST	R/W1C/H	0	0	Write 1 to clear MIO RST interrupt enable.
<62:60>	—	RAZ	0x0	0x0	Reserved.
<59:56>	—	RAZ	0x0	0x0	Reserved for DFM expansion.
<55:53>	—	RAZ	0x0	0x0	Reserved for LMC expansion.
<52>	LMC0	R/W1C/H	0	0	Write 1 to clear LMC0 interrupt enable.
<51>	—	RAZ	—	—	Reserved
<50>	PEM2	R/W1C/H	0	0	Write 1 to clear PEM2 interrupt enable.
<49>	PEM1	R/W1C/H	0	0	Write 1 to clear PEM1 interrupt enable.
<48>	PEM0	R/W1C/H	0	0	Write 1 to clear PEM0 interrupt enable.
<47>	PTP	R/W1C/H	0	0	Write 1 to clear PTP interrupt enable.
<46>	AGL	R/W1C/H	0	0	Write 1 to clear AGL interrupt enable.
<45:44>	—	RAZ	0x0	0x0	Reserved for future Interlaken.
<43:41>	—	RAZ	0x0	0x0	Reserved.
<40>	DPI_DMA	R/W1C/H	0	0	DPI DMA interrupt enable. See DPI DMA instruction completion.
<39:38>	—	RAZ	0x0	0x0	Reserved for GMX expansion.
<37>	AGX1	R/W1C/H	0	0	Write 1 to clear GMX1 interrupt enable.
<36>	AGX0	R/W1C/H	0	0	Write 1 to clear GMX0 interrupt enable.
<35>	DPI	R/W1C/H	0	0	Write 1 to clear DPI interrupt enable.
<34>	SLI	R/W1C/H	0	0	Write 1 to clear SLI interrupt enable.
<33>	USB	R/W1C/H	0	0	Write 1 to clear USBDRD0 interrupt enable.
<32>	DFA	R/W1C/H	0	0	Write 1 to clear HFA interrupt enable.
<31>	KEY	R/W1C/H	0	0	Write 1 to clear KEY interrupt enable.
<30>	RAD	R/W1C/H	0	0	Write 1 to clear RAD interrupt enable.
<29>	TIM	R/W1C/H	0	0	Write 1 to clear TIM interrupt enable.
<28>	—	RAZ	—	—	Reserved
<27>	PKO	R/W1C/H	0	0	Write 1 to clear PKO interrupt enable.
<26>	PIP	R/W1C/H	0	0	Write 1 to clear PIP interrupt enable.
<25>	IPD	R/W1C/H	0	0	Write 1 to clear IPD interrupt enable.
<24>	L2C	R/W1C/H	0	0	Write 1 to clear L2C interrupt enable.
<23>	POW	R/W1C/H	0	0	Write 1 to clear POW err interrupt enable.
<22>	FPA	R/W1C/H	0	0	Write 1 to clear FPA interrupt enable.
<21>	IOB	R/W1C/H	0	0	Write 1 to clear IOB interrupt enable.
<20>	MIO	R/W1C/H	0	0	Write 1 to clear MIO boot interrupt enable.
<19>	NAND	R/W1C/H	0	0	Write 1 to clear eMMC flash controller interrupt enable.
<18>	—	RAZ	—	—	Reserved for additional Interrupts.
<17>	USB1	R/W1C/H	0	0	Write 1 to clear USBDRD1 interrupt enable.
<16:4>	—	RAZ	—	—	Reserved
<3:0>	WDOG	R/W1C/H	0x0	0x0	Write 1s to clear Watchdog summary interrupt enable.

Cavium Confidential For Bowen Yang ALU 02/27/2014

CIU Write-1-to-Clear Interrupt Enable IP4 Registers 0
CIU_INT(0..3)_EN4_0_W1C

This is the write-1-to-clear version of the CIU_INT(0..1)_EN4_0 registers, read back corresponding CIU_INT32_EN4_0 value. See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	BOOTDMA	R/W1C/H	0	0	Write 1 to clear boot bus DMA engines interrupt enable.
<62>	—	RAZ	0	0	Reserved
<61>	IPDPPTHR	R/W1C/H	0	0	Write 1 to clear IPD per-port counter threshold interrupt enable.
<60>	POWIQ	R/W1C/H	0	0	Write 1 to clear POW IQ interrupt enable.
<59>	TWSI2	R/W1C/H	0	0	Write 1 to clear second TWSI interrupt enable.
<58>	MPI	R/W1C/H	0	0	Write 1 to clear MPI/SPI interrupt enable.
<57>	PCM	R/W1C/H	0	0	Write 1 to clear PCM/TDM interrupt enable.
<56>	—	RAZ	—	—	Reserved
<55:52>	TIMER	R/W1C/H	0x0	0x0	Write 1 to clear general timer interrupt enables.
<51>	—	RAZ	—	—	Reserved
<50>	IPD_DRP	R/W1C/H	0	0	Write 1 to clear IPD QOS packet drop interrupt enable.
<49>	—	RAZ	—	—	Reserved
<48>	GMX_DRP	R/W1C/H	0	0	Write 1 to clear GMX packet drop interrupt enable.
<47:46>	—	RAZ	—	—	Reserved
<45>	TWSI	R/W1C/H	0	0	Write 1 to clear TWSI interrupt enable.
<44>	—	RAZ	—	—	Reserved
<43:40>	PCI_MSI	R/W1C/H	0x0	0x0	Write 1s to clear PCIe MSI enables.
<39:36>	PCI_INT	R/W1C/H	0x0	0x0	Write 1s to clear PCIe INTA/B/C/D enables.
<35:34>	UART	R/W1C/H	0x0	0x0	Write 1s to clear UART interrupt enables.
<33:32>	MBOX	R/W1C/H	0x0	0x0	Write 1s to clear mailbox interrupt enables.
<31:16>	GPIO	R/W1C/H	0x0	0x0	Write 1s to clear GPIO interrupt enables.
<15:0>	WORKQ	R/W1C/H	0x0	0x0	Write 1s to clear work queue interrupt enables.

Cavium Confidential For Bowen Yang ALU 02/27/2014

CIU Write-1-to-Clear Interrupt Enable IP4 Registers 1
CIU_INT(0..3)_EN4_1_W1C

This is the write-1-to-clear version of the CIU_INT(0..1)_EN4_1 register, read back corresponding CIU_INT*x*_EN4_1 value. See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	RST	R/W1C/H	0	0	Write 1 to clear MIO RST interrupt enable.
<62:60>	—	RAZ	0x0	0x0	Reserved.
<59:56>	—	RAZ	0x0	0x0	Reserved for DFM expansion.
<55:53>	—	RAZ	0x0	0x0	Reserved for LMC expansion.
<52>	LMC0	R/W1C/H	0	0	Write 1 to clear LMC0 interrupt enable.
<51>	—	R/W1C/H	—	—	Reserved.
<50>	PEM2	R/W1C/H	0	0	Write 1 to clear PEM2 interrupt enable.
<49>	PEM1	R/W1C/H	0	0	Write 1 to clear PEM1 interrupt enable.
<48>	PEM0	R/W1C/H	0	0	Write 1 to clear PEM0 interrupt enable.
<47>	PTP	R/W1C/H	0	0	Write 1 to clear PTP interrupt enable.
<46>	AGL	R/W1C/H	0	0	Write 1 to clear AGL interrupt enable.
<45:44>	—	RAZ	0x0	0x0	Reserved for future Interlaken.
<43:41>	—	RAZ	0x0	0x0	Reserved.
<40>	DPI_DMA	R/W1C/H	0	0	DPI DMA interrupt enable. See DPI DMA instruction completion.
<39:38>	—	RAZ	0x0	0x0	Reserved.
<37>	AGX1	R/W1C/H	0	0	Write 1 to clear GMX1 interrupt enable.
<36>	AGX0	R/W1C/H	0	0	Write 1 to clear GMX0 interrupt enable.
<35>	DPI	R/W1C/H	0	0	Write 1 to clear DPI interrupt enable.
<34>	SLI	R/W1C/H	0	0	Write 1 to clear SLI interrupt enable.
<33>	USB	R/W1C/H	0	0	Write 1 to clear USBDRD0 interrupt enable.
<32>	DFA	R/W1C/H	0	0	Write 1 to clear HFA interrupt enable.
<31>	KEY	R/W1C/H	0	0	Write 1 to clear KEY interrupt enable.
<30>	RAD	R/W1C/H	0	0	Write 1 to clear RAD interrupt enable.
<29>	TIM	R/W1C/H	0	0	Write 1 to clear TIM interrupt enable.
<28>	—	RAZ	—	—	Reserved.
<27>	PKO	R/W1C/H	0	0	Write 1 to clear PKO interrupt enable.
<26>	PIP	R/W1C/H	0	0	Write 1 to clear PIP interrupt enable.
<25>	IPD	R/W1C/H	0	0	Write 1 to clear IPD interrupt enable.
<24>	L2C	R/W1C/H	0	0	Write 1 to clear L2C interrupt enable.
<23>	POW	R/W1C/H	0	0	Write 1 to clear POW err interrupt enable.
<22>	FPA	R/W1C/H	0	0	Write 1 to clear FPA interrupt enable.
<21>	IOB	R/W1C/H	0	0	Write 1 to clear IOB interrupt enable.
<20>	MIO	R/W1C/H	0	0	Write 1 to clear MIO boot interrupt enable.
<19>	NAND	R/W1C/H	0	0	Write 1 to clear eMMC flash controller interrupt enable.
<18>	—	RAZ	—	—	Reserved for additional interrupts.
<17>	USB1	R/W1C/H	0	0	Write 1 to clear USBDRD1 interrupt enable.
<16>	—	RAZ	—	—	Reserved for Third UART interrupt.
<15:4>	—	RAZ	—	—	Reserved.
<3:0>	WDOG	R/W1C/H	0x0	0x0	Write 1s to clear Watchdog summary interrupt enable.

Cavium Confidential For Bowen Yang ALU 02/27/2014

CIU Write-1-to-Set Interrupt Enable 0 Registers
CIU_INT(0..7,32)_EN0_W1S

This is the write-1-to-set version of the CIU_INT(0..3,32)_EN0 registers, read back corresponding CIU_INT*x*_EN0 value. CIU_INT33_EN0_W1S is reserved. See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	BOOTDMA	R/W1S/H	0	0	Write 1 to set boot bus DMA engines interrupt enable.
<62>	—	RAZ	0	0	Reserved.
<61>	IPDPPTHR	R/W1S/H	0	0	Write 1 to set IPD per-port counter threshold interrupt enable.
<60>	POWIQ	R/W1S/H	0	0	Write 1 to set POW IQ interrupt enable.
<59>	TWSI2	R/W1S/H	0	0	Write 1 to set second TWSI interrupt enable.
<58>	MPI	R/W1S/H	0	0	Write 1 to set MPI/SPI interrupt enable.
<57>	PCM	R/W1S/H	0	0	Write 1 to set PCM/TDM interrupt enable.
<56>	—	RAZ	0	0	Reserved for USB interrupt enable.
<55:52>	TIMER	R/W1S/H	0x0	0x0	Write 1 to set general timer interrupt enables.
<51>	—	RAZ	—	—	Reserved
<50>	IPD_DRP	R/W1S/H	0	0	Write 1 to set IPD QOS packet drop interrupt enable.
<49:48>	GMX_DRP	R/W1S/H	0x0	0x0	Write 1 to set GMX packet drop interrupt enable.
<47:46>	RESERVED_0	R/W1S/H	0x0	0x0	Reserved for Key Zeroization interrupt enable.
<45>	TWSI	R/W1S/H	0	0	Write 1 to set TWSI interrupt enable.
<44>	—	RAZ	—	—	Reserved.
<43:40>	PCI_MSI	R/W1S/H	0x0	0x0	Write 1s to set PCIe MSI enables.
<39:36>	PCI_INT	R/W1S/H	0x0	0x0	Write 1s to set PCIe INTA/B/C/D enables.
<35:34>	UART	R/W1S/H	0x0	0x0	Write 1s to set UART interrupt enables.
<33:32>	MBOX	R/W1S/H	0x0	0x0	Write 1s to set mailbox/PCIe interrupt enables.
<31:16>	GPIO	R/W1S/H	0x0	0x0	Write 1s to set GPIO interrupt enables.
<15:0>	WORKQ	R/W1S/H	0x0	0x0	Write 1s to set work-queue interrupt enables.

Cavium Confidential For Bowen Yang ALD 02/27/2014

CIU Write-1-to-Set Interrupt Enable 1 Registers
CIU_INT(0..7,32)_EN1_W1S

This is the write-1-to-set version of the CIU_INT(0..3,32/33)_EN1 registers. See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	RST	R/W1S/H	0	0	Write 1 to set MIO RST interrupt enable.
<62:60>	—	RAZ	0x0	0x0	Reserved.
<59:56>	—	RAZ	0x0	0x0	Reserved for DFM expansion.
<55:53>	—	RAZ	0x0	0x0	Reserved for LMC expansion.
<52>	LMC0	R/W1S/H	0	0	Write 1 to set LMC0 interrupt enable.
<51>	—	RAZ	0	0	Reserved
<50>	PEM2	R/W1S/H	0	0	Write 1 to set PEM2 interrupt enable.
<49>	PEM1	R/W1S/H	0	0	Write 1 to set PEM1 interrupt enable.
<48>	PEM0	R/W1S/H	0	0	Write 1 to set PEM0 interrupt enable.
<47>	PTP	R/W1S/H	0	0	Write 1 to set PTP interrupt enable.
<46>	AGL	R/W1S/H	0	0	Write 1 to set AGL interrupt enable.
<45:44>	—	RAZ	0x0	0x0	Reserved for future Interlaken.
<43:41>	—	RAZ	0x0	0x0	Reserved.
<40>	DPI_DMA	R/W1S/H	0	0	Write 1 to set DPI_DMA interrupt enable.
<39:38>	—	RAZ	0x0	0x0	Reserved for GMX expansion.
<37>	AGX1	R/W1S/H	0	0	Write 1 to set GMX1 interrupt enable.
<36>	AGX0	R/W1S/H	0	0	Write 1 to set GMX0 interrupt enable.
<35>	DPI	R/W1S/H	0	0	Write 1 to set DPI interrupt enable.
<34>	SLI	R/W1S/H	0	0	Write 1 to set SLI interrupt enable.
<33>	USB	R/W1S/H	0	0	Write 1 to set USBDRD0 interrupt enable.
<32>	DFA	R/W1S/H	0	0	Write 1 to set HFA interrupt enable.
<31>	KEY	R/W1S/H	0	0	Write 1 to set KEY interrupt enable.
<30>	RAD	R/W1S/H	0	0	Write 1 to set RAD interrupt enable.
<29>	TIM	R/W1S/H	0	0	Write 1 to set TIM interrupt enable.
<28>	RESERVED_1	R/W1S	0	0	Reserved.
<27>	PKO	R/W1S/H	0	0	Write 1 to set PKO interrupt enable.
<26>	PIP	R/W1S/H	0	0	Write 1 to set PIP interrupt enable.
<25>	IPD	R/W1S/H	0	0	Write 1 to set IPD interrupt enable.
<24>	L2C	R/W1S/H	0	0	Write 1 to set L2C interrupt enable.
<23>	POW	R/W1S/H	0	0	Write 1 to set POW err interrupt enable.
<22>	FPA	R/W1S/H	0	0	Write 1 to set FPA interrupt enable.
<21>	IOB	R/W1S/H	0	0	Write 1 to set IOB interrupt enable.
<20>	MIO	R/W1S/H	0	0	Write 1 to set MIO boot interrupt enable.
<19>	NAND	R/W1S/H	0	0	Write 1 to set eMMC flash controller interrupt enable.
<18>	—	RAZ	0	0	Reserved for additional interrupts.
<17>	USB1	R/W1S/H	0	0	Write 1 to set USBDRD1 interrupt enable.
<16:4>	—	RAZ	—	—	Reserved.
<3:0>	WDOG	R/W1S/H	0x0	0x0	Write 1s to set Watchdog summary interrupt enable.

Cavium Confidential For Bowei Yang ARO 02/27/2014

CIU Write-1-to-Set Interrupt Enable IP4 Registers 0
CIU_INT(0..3)_EN4_0_W1S

This is the write-1-to-set version of the CIU_INT(0..1)_EN4_0 registers, read back corresponding CIU_INT*x*_EN4_0 value. See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	BOOTDMA	R/W1S/H	0	0	Write 1 to set boot bus DMA engines interrupt enable.
<62>	—	RAZ	0	0	Reserved.
<61>	IPDPPTHR	R/W1S/H	0	0	Write 1 to set IPD per-port counter threshold interrupt enable.
<60>	POWIQ	R/W1S/H	0	0	Write 1 to set POW IQ interrupt enable.
<59>	TWSI2	R/W1S/H	0	0	Write 1 to set second TWSI interrupt enable.
<58>	MPI	R/W1S/H	0	0	Write 1 to set MPI/SPI interrupt enable.
<57>	PCM	R/W1S/H	0	0	Write 1 to set PCM/TDM interrupt enable.
<56>	—	RAZ	0	0	Reserved for USB interrupt enable.
<55:52>	TIMER	R/W1S/H	0x0	0x0	Write 1 to set general timer interrupt enables.
<51>	—	RAZ	0	0	Reserved for SUM2 interrupt enable.
<50>	IPD_DRP	R/W1S/H	0	0	Write 1 to set IPD QOS packet drop interrupt enable.
<49:48>	GMX_DRP	R/W1S/H	0x0	0x0	Write 1 to set GMX packet drop interrupt enable.
<47:46>	RESERVED_0	R/W1S/H	0x0	0x0	Reserved.
<45>	TWSI	R/W1S/H	0	0	Write 1 to set TWSI interrupt enable.
<44>	—	RAZ	—	—	Reserved.
<43:40>	PCI_MSI	R/W1S/H	0x0	0x0	Write 1s to set PCIe MSI enables.
<39:36>	PCI_INT	R/W1S/H	0x0	0x0	Write 1s to set PCIe INTA/B/C/D enables.
<35:34>	UART	R/W1S/H	0x0	0x0	Write 1s to set UART interrupt enables.
<33:32>	MBOX	R/W1S/H	0x0	0x0	Write 1s to set mailbox interrupt enables.
<31:16>	GPIO	R/W1S/H	0x0	0x0	Write 1s to set GPIO interrupt enables.
<15:0>	WORKQ	R/W1S/H	0x0	0x0	Write 1s to set work queue interrupt enables.

Cavium Confidential For Bowen Yang ATD 02/27/2014

CIU Write-1-to-Set Interrupt Enable IP4 Registers 1
CIU_INT(0..3)_EN4_1_W1S

This is the write-1-to-set version of the CIU_INT(0..1)_EN4_1 register. See Table 13–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	RST	R/W1S/H	0	0	Write 1 to set MIO RST interrupt enable.
<62:60>	—	RAZ	0x0	0x0	Reserved.
<59:56>	—	RAZ	0x0	0x0	Reserved for DFM expansion.
<55:53>	—	RAZ	0x0	0x0	Reserved for LMC expansion.
<52>	LMC0	R/W1S/H	0	0	Write 1 to set LMC0 interrupt enable.
<51>	—	RAZ	0	0	Reserved.
<50>	PEM2	R/W1S/H	0	0	Write 1 to set PEM2 interrupt enable.
<49>	PEM1	R/W1S/H	0	0	Write 1 to set PEM1 interrupt enable.
<48>	PEM0	R/W1S/H	0	0	Write 1 to set PEM0 interrupt enable.
<47>	PTP	R/W1S/H	0	0	Write 1 to set PTP interrupt enable.
<46>	AGL	R/W1S/H	0	0	Write 1 to set AGL interrupt enable.
<45:44>	—	RAZ	0x0	0x0	Reserved for future Interlaken.
<43:41>	—	RAZ	0x0	0x0	Reserved.
<40>	DPI_DMA	R/W1S/H	0	0	DPI DMA interrupt enable. See DPI DMA instruction completion.
<39:38>	—	RAZ	0x0	0x0	Reserved for GMX expansion.
<37>	AGX1	R/W1S/H	0	0	Write 1 to set GMX1 interrupt enable.
<36>	AGX0	R/W1S/H	0	0	Write 1 to set GMX0 interrupt enable.
<35>	DPI	R/W1S/H	0	0	Write 1 to set DPI interrupt enable.
<34>	SLI	R/W1S/H	0	0	Write 1 to set SLI interrupt enable.
<33>	USB	R/W1S/H	0	0	Write 1 to set USBDRD0 interrupt enable.
<32>	DFA	R/W1S/H	0	0	Write 1 to set HFA interrupt enable.
<31>	KEY	R/W1S/H	0	0	Write 1 to set KEY interrupt enable.
<30>	RAD	R/W1S/H	0	0	Write 1 to set RAD interrupt enable.
<29>	TIM	R/W1S/H	0	0	Write 1 to set TIM interrupt enable.
<28>	RESERVED_1	R/W1S	0	0	Reserved for LMC expansion.
<27>	PKO	R/W1S/H	0	0	Write 1 to set PKO interrupt enable.
<26>	PIP	R/W1S/H	0	0	Write 1 to set PIP interrupt enable.
<25>	IPD	R/W1S/H	0	0	Write 1 to set IPD interrupt enable.
<24>	L2C	R/W1S/H	0	0	Write 1 to set L2C interrupt enable.
<23>	POW	R/W1S/H	0	0	Write 1 to set POW err interrupt enable.
<22>	FPA	R/W1S/H	0	0	Write 1 to set FPA interrupt enable.
<21>	IOB	R/W1S/H	0	0	Write 1 to set IOB interrupt enable.
<20>	MIO	R/W1S/H	0	0	Write 1 to set MIO boot interrupt enable.
<19>	NAND	R/W1S/H	0	0	Write 1 to set eMMC Flash Controller interrupt enable.
<18>	—	RAZ	0	0	Reserved for additional interrupts.
<17>	USB1	R/W1S/H	0	0	Write 1 to set USBDRD1 interrupt enable.
<16>	—	RAZ	—	—	Reserved for Third UART interrupt.
<15:4>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang ARO 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<3:0>	WDOG	R/W1S/H	0x0	0x0	Write 1s to set Watchdog summary interrupt enable.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

CIU Interrupt Summary 1 Registers

CIU_SUM1_PP(0..3)_IP2
CIU_SUM1_PP(0..3)_IP3
CIU_SUM1_PP(0..3)_IP4
CIU_SUM1_IO0_INT

Only field <40> DPI_DMA will have a different value per core (IP) for CIU_SUM1_PP(0..1)_IP*y*, and <40> DPI_DMA is always zero for CIU_SUM1_IO0_INT. All other fields ([63:41] and [39:0]) values are identical for different cores (same value as CIU_INT_SUM1).

Write operations to any IRQs PTP field clears PTP for all IRQs' PTP fields. See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	RST	RO/H	0	0	MIO RST interrupt. See RST_INT.
<62:60>	—	RAZ	—	—	Reserved.
<59:56>	—	RAZ	—	—	Reserved for DFM expansion.
<55:53>	—	RAZ	—	—	Reserved for LMC expansion.
<52>	LMC0	RO/H	0	0	LMC0 interrupt. (CIU_CIB_LMC(0)_RAW[0] AND CIU_CIB_LMC(0)_EN[0]). See CIU_CIB_LMC(0)_RAW[0] and CIU_CIB_LMC(0)_EN[0].
<51>	—	RAZ	—	—	Reserved.
<50>	PEM2	RO/H	0	0	PEM2 interrupt. See PEM2_INT_SUM (enabled by PEM2_INT_ENB).
<49>	PEM1	RO/H	0	0	PEM1 interrupt. See PEM1_INT_SUM (enabled by PEM1_INT_ENB).
<48>	PEM0	RO/H	0	0	PEM0 interrupt. See PEM0_INT_SUM (enabled by PEM0_INT_ENB).
<47>	PTP	R/W1C/H	0	0	PTP interrupt. Set when hardware decrements MIO_PTP_EVT_CNT to zero.583
<46>	AGL	RO/H	0	0	AGL interrupt. See AGL_GMX_RX0/1_INT_REG, AGL_GMX_TX_INT_REG.
<45:44>	—	RAZ	—	—	Reserved for future Interlaken.
<43:41>	—	RAZ	—	—	Reserved.
<40>	DPI_DMA	RO/H	0	0	DPI DMA interrupt enable. See DPI DMA instruction completion.
<39:38>	—	RAZ	—	—	Reserved for GMX expansion.
<37>	AGX1	RO/H	0	0	GMX1 interrupt. See GMX1_RX0/1_INT_REG, GMX1_TX_INT_REG PCS1_INT(0..3)_REG, PCSX1_INT_REG.
<36>	AGX0	RO/H	0	0	GMX0 interrupt. See GMX0_RX0/1_INT_REG, GMX0_TX_INT_REG PCS0_INT(0..3)_REG, PCSX0_INT_REG.
<35>	DPI	RO/H	0	0	DPI interrupt. See DPI_INT_REG.
<34>	SLI	RO/H	0	0	SLI interrupt. See SLI_INT_SUM (enabled by SLI_INT_ENB_CIU).
<33>	USB	RO/H	0	0	USBDRD0 interrupt. (CIU_CIB_USBDRD(0)_RAW[0] AND CIU_CIB_USBDRD(0)_EN[0]). See CIU_CIB_USBDRD(0)_RAW[0] and CIU_CIB_USBDRD(0)_EN[0].
<32>	DFA	RO/H	0	0	DFA interrupt. See HFA_ERROR.
<31>	KEY	RO/H	0	0	KEY interrupt. See KEY_INT_SUM.
<30>	RAD	RO/H	0	0	RAD interrupt. See RAD_REG_ERROR.
<29>	TIM	RO/H	0	0	TIM interrupt. See TIM_REG_ERROR.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<28>	—	RAZ	—	—	Reserved.
<27>	PKO	RO/H	0	0	PKO interrupt. See PKO_REG_ERROR.
<26>	PIP	RO/H	0	0	PIP interrupt. See PIP_INT_REG.
<25>	IPD	RO/H	0	0	IPD interrupt. See IPD_INT_SUM.
<24>	L2C	RO/H	0	0	L2C interrupt. See L2C_INT_REG.
<23>	POW	RO/H	0	0	POW err interrupt. See POW_ECC_ERR.
<22>	FPA	RO/H	0	0	FPA interrupt. See FPA_INT_SUM.
<21>	IOB	RO/H	0	0	IOB interrupt. See IOB_INT_SUM.
<20>	MIO	RO/H	0	0	MIO boot interrupt. See MIO_BOOT_ERR.
<19>	NAND	RO/H	0	0	eMMC flash controller interrupt. See MIO_EMM_INT.
<18>	—	RAZ	—	—	Reserved for additional interrupts.
<17>	USB1	RO/H	0	0	USBDRD1 interrupt. (CIU_CIB_USBDRD(1)_RAW[0] AND CIU_CIB_USBDRD(1)_EN[0]). See CIU_CIB_USBDRD(1)_RAW[0] and CIU_CIB_USBDRD(1)_EN[0].
<16>	—	RAZ	—	—	Reserved for Third UART interrupt.
<15:4>	—	RAZ	—	—	Reserved.
<3:0>	WDOG	RO/H	0x0	0x0	Per-core watchdog interrupts.

Cavium Confidential For Bowen Yang ALU 02/27/2014

CIU Interrupt Summary 2 Registers

CIU_SUM2_PP(0..3)_IP2
CIU_SUM2_PP(0..3)_IP3
CIU_SUM2_PP(0..3)_IP4
CIU_SUM2_IO0_INT

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:20>	—	RAZ	—	—	Reserved
<19>	BCH	RO/H	0	0	BCH interrupt. See BCH_GEN_INT.
<18>	AGL_DRP	R/W1C	0	0	AGL packet-drop interrupt. TBD.
<17>	OCLA	RO/H	0	0	OCLA interrupt summary. (CIU_CIB_OCLA(0)_RAW[0]ANDCIU_CIB_OCLA(0)_EN[0]). See CIU_CIB_OCLA(0)_RAW[0] and CIU_CIB_OCLA(0)_EN[0].
<16>	SATA	RO/H	0	0	SATA interrupt summary. (CIU_CIB_SATA(0)_RAW[0]ANDCIU_CIB_SATA(0)_EN(0)). See CIU_CIB_SATA(0)_RAW[0] and CIU_CIB_SATA(0)_EN(0).
<15:10>	—	RAZ	—	—	Reserved
<9:4>	TIMER	R/W1C/H	0x0	0x0	General timer 4–9 interrupts. When CIU_TIM_MULTI_CAST[EN] = 0, this interrupt is common for all cores/IRQs; writing 1 to any core/IRQ clears all TIMER*x* (*x* = 0..9) interrupts. When CIU_TIM_MULTI_CAST[EN] = 1, TIMER*x* (*x* = 0..9) are set at the same time, but clearing is per-core. See CIU_TIM_MULTI_CAST for details. The combination of this field and the CIU_INT*_SUM0/4[TIMER] field implement all ten CIU_TIM* interrupts.
<3:0>	—	RAZ	—	—	Reserved for TIMER interrupts.

Cavium Confidential For Bowen Tang ALU 02/27/2014

CIU Interrupt Enable Registers

CIU_EN2_PP(0..3)_IP2
CIU_EN2_PP(0..3)_IP3
CIU_EN2_PP(0..3)_IP4
CIU_EN2_IO0_INT

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:20>	—	RAZ	—	—	Reserved.
<19>	BCH	R/W	0	0	BCH interrupt enable.
<18>	AGL_DRP	R/W	0	0	AGL packet drop interrupt.
<17>	OCLA	R/W	0	0	OCLA interrupt enable.
<16>	SATA	R/W	0	0	SATA interrupt enable.
<15:10>	—	RAZ	—	—	Reserved.
<9:4>	TIMER	R/W	0x0	0x0	General timer 4-9 interrupt enable.
<3:0>	—	RAZ	—	—	Reserved.

CIU Interrupt Enable Write-1-to-Set Registers

CIU_EN2_PP(0..3)_IP2_W1S
CIU_EN2_PP(0..3)_IP3_W1S
CIU_EN2_PP(0..3)_IP4_W1S
CIU_EN2_IO0_INT_W1S

Write-1-to-set version of the CIU_EN2_*xx_yy* register, read back corresponding CIU_EN2_*xx_yy* value. See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:20>	—	RAZ	—	—	Reserved.
<19>	BCH	R/W1S/H	0	0	Write 1 to set BCH interrupt enable.
<18>	AGL_DRP	R/W1S/H	0	0	Write 1 to set AGL_DRP interrupt enable.
<17>	OCLA	R/W1S/H	0	0	Write 1 to set OCLA interrupt enable.
<16>	SATA	R/W1S/H	0	0	Write 1 to set SATA interrupt enable.
<15:10>	—	RAZ	—	—	Reserved.
<9:4>	TIMER	R/W1S/H	0x0	0x0	Write 1 to set general timer 4–9 interrupt enable.
<3:0>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang ARU 02/27/2014

CIU Interrupt Enable Write-1-to-Clear Registers

CIU_EN2_PP(0..3)_IP2_W1C
CIU_EN2_PP(0..3)_IP3_W1C
CIU_EN2_PP(0..3)_IP4_W1C
CIU_EN2_IO0_INT_W1C

Write-1-to-clear version of the CIU_EN2_*xx_yy* register, read back corresponding CIU_EN2_*xx_yy* value. See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved
<19>	BCH	R/W1C/H	0	0	Write 1 to clear BCH interrupt enable.
<18>	AGL_DRP	R/W1C/H	0	0	Write 1 to clear AGL_DRP interrupt enable.
<17>	OCLA	R/W1C/H	0	0	Write 1 to clear OCLA interrupt enable.
<16>	SATA	R/W1C/H	0	0	Write 1 to clear SATA interrupt enable.
<15:10>	—	RAZ	—	—	Reserved
<9:4>	TIMER	R/W1C/H	0x0	0x0	Write 1 to clear general timer 4–9 interrupt enable.
<3:0>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang ALU 02/27/2014

CIU Delivery L2C RAW Register

CIU_CIB_L2C_RAW(0)

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:23>	—	RAZ	—	—	Reserved
<22>	CBCX_INT_IOCCMDDBE	R/W1C/H	0	0	Set when L2C_CBC(0)_INT[IOCCMDDBE] is set. Edge-sensitive interrupt, so software should clear [CBCX_INT_IOCCMDDBE] before clearing L2C_CBC(0)_INT[IOCCMDDBE].
<21>	CBCX_INT_IOCCMDSBE	R/W1C/H	0	0	Set when L2C_CBC(0)_INT[IOCCMDSBE] is set. Edge-sensitive interrupt, so software should clear [CBCX_INT_IOCCMDSBE] before clearing L2C_CBC(0)_INT[IOCCMDSBE].
<20>	CBCX_INT_RSDDBE	R/W1C/H	0	0	Set when L2C_CBC(0)_INT[RSDDBE] is set. Edge-sensitive interrupt, so software should clear [CBCX_INT_RSDDBE] before clearing L2C_CBC(0)_INT[RSDDBE].
<19>	CBCX_INT_RSDSBE	R/W1C/H	0	0	Set when L2C_CBC(0)_INT[RSDSBE] is set. Edge-sensitive interrupt, so software should clear [CBCX_INT_RSDSBE] before clearing L2C_CBC(0)_INT[RSDSBE].
<18>	MCIX_INT_VBFDBE	R/W1C/H	0	0	Set when L2C_MCI(0)_INT[VBFDBE] set. Edge-sensitive interrupt, so software should clear [MCIX_INT_VBFDBE] before clearing L2C_MCI(0)_INT[VBFDBE].
<17>	MCIX_INT_VBFSBE	R/W1C/H	0	0	Set when LL2C_MCI(0)_INT[VBFSBE] is set. Edge-sensitive interrupt, so software should clear [MCIX_INT_VBFSBE] before clearing L2C_MCI(0)_INT[VBFSBE].
<16>	TADX_INT_RTGDBE	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[RTGDBE] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_RTGDBE] before clearing L2C_TAD(0)_INT[RTGDBE].
<15>	TADX_INT_RTGSBE	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[RTGSBE] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_RTGSBE] before clearing L2C_TAD(0)_INT[RTGSBE].
<14>	TADX_INT_RDDISLMC	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[RDDISLMC] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_RDDISLMC] before clearing L2C_TAD(0)_INT[RDDISLMC].
<13>	TADX_INT_WRDISLMC	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[WRDISLMC] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_WRDISLMC] before clearing L2C_TAD(0)_INT[WRDISLMC].
<12>	TADX_INT_BIGRD	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[BIGRD] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_BIGRD] before clearing L2C_TAD(0)_INT[BIGRD].
<11>	TADX_INT_BIGWR	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[BIGWR] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_BIGWR] before clearing L2C_TAD(0)_INT[BIGWR].

Cavium Confidential For Bowen Ang 02 2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<10>	TADX_INT_HOLERD	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[HOLERD] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_HOLERD] before clearing L2C_TAD(0)_INT[HOLERD].
<9>	TADX_INT_HOLEWR	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[HOLEWR] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_HOLEWR] before clearing L2C_TAD(0)_INT[HOLEWR].
<8>	TADX_INT_NOWAY	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[NOWAY] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_NOWAY] before clearing L2C_TAD(0)_INT[NOWAY].
<7>	TADX_INT_TAGDBE	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[TAGDBE] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_TAGDBE] before clearing L2C_TAD(0)_INT[TAGDBE].
<6>	TADX_INT_TAGSBE	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[TAGSBE] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_TAGSBE] before clearing L2C_TAD(0)_INT[TAGSBE].
<5>	TADX_INT_FBFDBE	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[FBFDBE] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_FBFDBE] before clearing L2C_TAD(0)_INT[FBFDBE].
<4>	TADX_INT_FBFSBE	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[FBFSBE] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_FBFSBE] before clearing L2C_TAD(0)_INT[FBFSBE].
<3>	TADX_INT_SBFDBE	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[SBFDBE] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_SBFDBE] before clearing L2C_TAD(0)_INT[SBFDBE].
<2>	TADX_INT_SBFSBE	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[SBFSBE] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_SBFSBE] before clearing L2C_TAD(0)_INT[SBFSBE].
<1>	TADX_INT_L2DDBE	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[L2DDBE] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_L2DDBE] before clearing L2C_TAD(0)_INT[L2DDBE].
<0>	TADX_INT_L2DSBE	R/W1C/H	0	0	Set when L2C_TAD(0)_INT[L2DSBE] is set. Edge-sensitive interrupt, so software should clear [TADX_INT_L2DSBE] before clearing L2C_TAD(0)_INT[L2DSBE].

Cavium Confidential For Bowei Yang 12/7/2015

CIU Delivery L2C Enable Register

CIU_CIB_L2C_EN(0)

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:23>	—	RAZ	—	—	Reserved.
<22>	CBCX_INT_IOCCMDDBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<21>	CBC_INT_IOCCMDSBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<20>	CBCX_INT_RSDDBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<19>	CBC_INT_RSDSBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<18>	MCIX_INT_VBFDBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<17>	MCIX_INT_VBFSBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<16>	TADX_INT_RTGDBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<15>	TADX_INT_RTGSBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<14>	TADX_INT_RDDISLMC	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<13>	TADX_INT_WRDISLMC	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<12>	TADX_INT_BIGRD	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<11>	TADX_INT_BIGWR	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<10>	TADX_INT_HOLERD	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<9>	TADX_INT_HOLEWR	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<8>	TADX_INT_NOWAY	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<7>	TADX_INT_TAGDBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<6>	TADX_INT_TAGSBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<5>	TADX_INT_FBFDBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<4>	TADX_INT_FBFSBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<3>	TADX_INT_SBFDBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<2>	TADX_INT_SBFSBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<1>	TADX_INT_L2DDBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.
<0>	TADX_INT_L2DSBE	R/W	0	0	Enable corresponding CIU_CIB_L2C_RAW bit.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

CIU Delivery LMC RAW Register

CIU_CIB_LMC(0)_RAW(0)

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved.
<11>	INT_DDR_ERR	R/W1C/H	0	0	Set when LMC(0)_INT[DDR_ERR] is set. Edge-sensitive interrupt, so software should clear [INT_DDR_ERR] before clearing LMC(0)_INT[DDR_ERR].
<10>	INT_DLC_DED	R/W1C/H	0	0	Set when LMC(0)_INT[DLCRAM_DED_ERR] is set. Edge-sensitive interrupt, so software should clear [INT_DLC_DED] before clearing LMC(0)_INT[DLCRAM_DED_ERR].
<9>	INT_DLC_SEC	R/W1C/H	0	0	Set when LMC(0)_INT[DLCRAM_SEC_ERR] is set. Edge-sensitive interrupt, so software should clear [INT_DLC_SEC] before clearing LMC(0)_INT[DLCRAM_SEC_ERR].
<8:5>	INT_DED_ERR	R/W1C/H	0x0	0x0	Set when LMC(0)_INT[DED_ERR<b>] set. Edge-sensitive interrupts, so software should clear [INT_DED_ERRX<b>] before clearing LMC(0)_INT[DED_ERR<b>].
<4:1>	INT_SEC_ERR	R/W1C/H	0x0	0x0	Set when LMC(0)_INT[SEC_ERR<b>] set. Edge-sensitive interrupts, so software should clear [INT_SEC_ERRX<b>] before clearing LMC(0)_INT[SEC_ERR<b>].
<0>	INT_NXM_WR_ERR	R/W1C/H	0	0	Set when LMC(0)_INT[NXM_WR_ERR] set. Edge-sensitive interrupt, so software should clear [INT_NXM_WR_ERR] before clearing LMC(0)_INT[NXM_WR_ERR].

CIU Delivery LMC Enable Register

CIU_CIB_LMC(0)_EN(0)

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved.
<11>	INT_DDR_ERR	R/W	0	0	Enable corresponding CIU_CIB_LMC_RAW bit.
<10>	INT_DLC_DED	R/W	0	0	Enable corresponding CIU_CIB_LMC_RAW bit.
<9>	INT_DLC_SEC	R/W	0	0	Enable corresponding CIU_CIB_LMC_RAW bit.
<8:5>	INT_DED_ERR	R/W	0x0	0x0	Enable corresponding CIU_CIB_LMC_RAW bit.
<4:1>	INT_SEC_ERR	R/W	0x0	0x0	Enable corresponding CIU_CIB_LMC_RAW bit.
<0>	INT_NXM_WR_ERR	R/W	0	0	Enable corresponding CIU_CIB_LMC_RAW bit.

Cavium Confidential
Bowen Yang
ALU
09/27/2024

CIU Delivery Reset RAW Register

CIU_CIB_RST_RAW(0)

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved.
<5:3>	INT_PERSTX	R/W1C/H	0x0	0x0	Set when RST_INT[PERST<a>] set. Edge-sensitive interrupts, so software should clear [INT_PERSTX<a>] before clearing RST_INT[PERST<a>].
<2:0>	INT_LINKX	R/W1C/H	0x0	0x0	Set when RST_INT[RST_LINK<a>] set. Edge-sensitive interrupts, so software should clear [INT_LINKX<a>] before clearing RST_INT[RST_LINK<a>].

CIU Delivery Reset Enable Register

CIU_CIB_RST_EN(0)

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved.
<5:3>	INT_PERSTX	R/W	0x0	0x0	Enable corresponding CIU_CIB_RST_RAW bit.
<2:0>	INT_LINKX	R/W	0x0	0x0	Enable corresponding CIU_CIB_RST_RAW bit.

Cavium Confidential For Bowen Yang ALU 02/27/2014

CIU Delivery SATA RAW Register

CIU_CIB_SATA_RAW(0)

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved.
<3>	UAHC_PME_REQ_IP	R/W1C/H	0	0	Summary for chip-internal level-sensitive interrupt that is asserted any of SATA_UAHC_P(0..1)_IS[CPDS,DMPS,PCS] are set, and also asserts when both SATA_UAHC_P(0..1)_IS[SDBS] and SATA_UAHC_P(0..1)_SNTF[PMN] are set. Hardware sets [UAHC_PME_REQ_IP] sometime after the underlying interrupt condition changes from deasserting to asserting, and clears [UAHC_PME_REQ_IP] sometime after the underlying interrupt condition changes from asserting to deasserting. R/W1C, but software need not clear [UAHC_PME_REQ_IP], and perhaps should only ever clear [UAHC_PME_REQ_IP] when the underlying interrupt condition is known to be asserted.
<2>	UAHC_INTRQ_IP	R/W1C/H	0	0	Summary for chip-internal level-sensitive interrupt that is asserted when any bit in SATA_UAHC_GBL_IS[IPS] is set. Hardware sets [UAHC_INTRQ_IP] sometime after the underlying interrupt condition changes from de-asserting to asserting, and clears [UAHC_INTRQ_IP] sometime after the underlying interrupt condition changes from asserting to deasserting. R/W1C, but software need not clear [UAHC_INTRQ_IP], and perhaps should only ever clear [UAHC_INTRQ_IP] when the underlying interrupt condition is known to be asserted.
<1>	INTSTAT_XM_BAD_DMA	R/W1C/H	0	0	Set when SATA_UCTL_INTSTAT[XM_BAD_DMA] = 1. Edge-sensitive interrupt, so software should clear [INTSTAT_XM_BAD_DMA] before clearing SATA_UCTL_INTSTAT[XM_BAD_DMA].
<0>	INTSTAT_XS_NCB_OOB	R/W1C/H	0	0	Set when SATA_UCTL_INTSTAT[XS_NCB_OOB] = 1. Edge-sensitive interrupt, so software should clear [INTSTAT_XS_NCB_OOB] before clearing SATA_UCTL_INTSTAT[XS_NCB_OOB].

CIU Delivery SATA Enable Register

CIU_CIB_SATA_EN(0)

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved.
<3>	UAHC_PME_REQ_IP	R/W/H	0	0	Enable corresponding CIU_CIB_SATA_RAW bit.
<2>	UAHC_INTRQ_IP	R/W/H	0	0	Enable corresponding CIU_CIB_SATA_RAW bit.
<1>	INTSTAT_XM_BAD_DMA	R/W/H	0	0	Enable corresponding CIU_CIB_SATA_RAW bit.
<0>	INTSTAT_XS_NCB_OOB	R/W/H	0	0	Enable corresponding CIU_CIB_SATA_RAW bit.

Cavium Confidential For Bowen Yang N.D.A. 2017

CIU Delivery USB RAW Registers
CIU_CIB_USBDRD(0..1)_RAW(0)

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:11>	—	RAZ	—	—	Reserved.
<10>	UAHC_DEV_INT	R/W1C/H	0	0	Summary for chip-internal level-sensitive interrupt that is asserted when corresponding (USBDRD(0..1)_UAHC_GEVNTCOUNT(0)[EVNTCOUNT]≠0). Hardware sets [UAHC_DEV_INT] sometime after corresponding USBDRD(0..1)_UAHC_GEVNTCOUNT(0)[EVNTCOUNT] changes from zero, and clears [UAHC_DEV_INT] sometime after corresponding USBDRD(0..1)_UAHC_GEVNTCOUNT(0)[EVNTCOUNT] changes to zero. R/W1C, but software need not clear [UAHC_DEV_INT], and perhaps should only ever clear [UAHC_DEV_INT] when corresponding USBDRD(0..1)_UAHC_GEVNTCOUNT(0)[EVNTCOUNT] is known to be non-zero.
<9>	UAHC_IMAN0_IP	R/W1C/H	0	0	Summary for chip-internal level-sensitive interrupt that is asserted when corresponding USBDRD(0..1)_UAHC_IMAN(0)[IP] is set. Hardware sets [UAHC_IMANX_IP] sometime after corresponding USBDRD(0..1)_UAHC_IMAN(0)[IP] changes 0->1, and clears [UAHC_IMANX_IP] sometime after corresponding USBDRD(0..1)_UAHC_IMAN(0)[IP] changes 1->0. R/W1C, but software need not clear [UAHC_IMANX_IP], and perhaps should only ever clear [UAHC_IMANX_IP] when corresponding USBDRD(0..1)_UAHC_IMAN(0)[IP] is known to be set.
<8>	UAHC_USBSTS_HSE	R/W1C/H	0	0	Summary for chip-internal level-sensitive interrupt that is asserted when corresponding USBDRD(0..1)_UAHC_USBSTS[HSE] is set. Hardware sets [UAHC_USBSTS_HSE] sometime after corresponding USBDRD(0..1)_UAHC_USBSTS[HSE] changes 0->1, and clears [UAHC_USBSTS_HSE] sometime after corresponding USBDRD(0..1)_UAHC_USBSTS[HSE] changes 1->0. R/W1C, but software need not clear [UAHC_USBSTS_HSE], and perhaps should only ever clear [UAHC_USBSTS_HSE] when corresponding USBDRD(0..1)_UAHC_USBSTS[HSE] is known to be set.
<7>	INTSTAT_RAM2_DBE	R/W1C/H	0	0	Set when corresponding USBDRD(0..1)_UCTL_INTSTAT[RAM2_DBE] set. Edge-sensitive interrupt, so software should clear [INTSTAT_RAM2_DBE] before clearing corresponding USBDRD(0..1)_UCTL_INTSTAT[RAM2_DBE].

Cavium Confidential For Bowei Only 04/24/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<6>	INTSTAT_RAM2_SBE	R/W1C/H	0	0	Set when corresponding USBDRD(0..1)_UCTL_INTSTAT[RAM2_SBE] set. Edge-sensitive interrupt, so software should clear [INTSTAT_RAM2_SBE] before clearing corresponding USBDRD(0..1)_UCTL_INTSTAT[RAM2_SBE].
<5>	INTSTAT_RAM1_DBE	R/W1C/H	0	0	Set when corresponding USBDRD(0..1)_UCTL_INTSTAT[RAM1_DBE] set. Edge-sensitive interrupt, so software should clear [INTSTAT_RAM1_DBE] before clearing corresponding USBDRD(0..1)_UCTL_INTSTAT[RAM1_DBE].
<4>	INTSTAT_RAM1_SBE	R/W1C/H	0	0	Set when corresponding USBDRD(0..1)_UCTL_INTSTAT[RAM1_SBE] set. Edge-sensitive interrupt, so software should clear [INTSTAT_RAM1_SBE] before clearing corresponding USBDRD(0..1)_UCTL_INTSTAT[RAM1_SBE].
<3>	INTSTAT_RAM0_DBE	R/W1C/H	0	0	Set when corresponding USBDRD(0..1)_UCTL_INTSTAT[RAM0_DBE] set. Edge-sensitive interrupt, so software should clear [INTSTAT_RAM0_DBE] before clearing corresponding USBDRD(0..1)_UCTL_INTSTAT[RAM0_DBE].
<2>	INTSTAT_RAM0_SBE	R/W1C/H	0	0	Set when corresponding USBDRD(0..1)_UCTL_INTSTAT[RAM0_SBE] set. Edge-sensitive interrupt, so software should clear [INTSTAT_RAM0_SBE] before clearing corresponding USBDRD(0..1)_UCTL_INTSTAT[RAM0_SBE].
<1>	INTSTAT_XM_BAD_DMA	R/W1C/H	0	0	Set when corresponding USBDRD(0..1)_UCTL_INTSTAT[XM_BAD_DMA] set. Edge-sensitive interrupt, so software should clear [INTSTAT_XM_BAD_DMA] before clearing corresponding USBDRD(0..1)_UCTL_INTSTAT[XM_BAD_DMA].
<0>	INTSTAT_XS_NCB_OOB	R/W1C/H	0	0	Set when corresponding USBDRD(0..1)_UCTL_INTSTAT[XS_NCB_OOB] set. Edge-sensitive interrupt, so software should clear [INTSTAT_XS_NCB_OOB] before clearing corresponding USBDRD(0..1)_UCTL_INTSTAT[XS_NCB_OOB].

Cavium Confidential For Bowen Yang 02/27/2015

CIU Delivery USB Enable Registers

CIU_CIB_USBDRD(0..1)_EN(0)

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:11>	—	RAZ	—	—	Reserved.
<10>	UAHC_DEV_INT	R/W	0	0	Enable corresponding CIU_CIB_USBDRD(0..1)_RAW bit.
<9>	UAHC_IMAN0_IP	R/W	0	0	Enable corresponding CIU_CIB_USBDRD(0..1)_RAW bit.
<8>	UAHC_USBSTS_HSE	R/W	0	0	Enable corresponding CIU_CIB_USBDRD(0..1)_RAW bit.
<7>	INTSTAT_RAM2_DBE	R/W	0	0	Enable corresponding CIU_CIB_USBDRD(0..1)_RAW bit.
<6>	INTSTAT_RAM2_SBE	R/W	0	0	Enable corresponding CIU_CIB_USBDRD(0..1)_RAW bit.
<5>	INTSTAT_RAM1_DBE	R/W	0	0	Enable corresponding CIU_CIB_USBDRD(0..1)_RAW bit.
<4>	INTSTAT_RAM1_SBE	R/W	0	0	Enable corresponding CIU_CIB_USBDRD(0..1)_RAW bit.
<3>	INTSTAT_RAM0_DBE	R/W	0	0	Enable corresponding CIU_CIB_USBDRD(0..1)_RAW bit.
<2>	INTSTAT_RAM0_SBE	R/W	0	0	Enable corresponding CIU_CIB_USBDRD(0..1)_RAW bit.
<1>	INTSTAT_XM_BAD_DMA	R/W	0	0	Enable corresponding CIU_CIB_USBDRD(0..1)_RAW bit.
<0>	INTSTAT_XS_NCB_OOB	R/W	0	0	Enable corresponding CIU_CIB_USBDRD(0..1)_RAW bit.

Cavium Confidential For Bowen Yang ALU 02/27/2014

CIU Delivery OCLA RAW Register

CIU_CIB_OCLA(0)_RAW(0)

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:15>	—	RAZ	—	—	Reserved.
<14>	STATE_DDRFULL	R/W1C/H	0	0	Set when OCLA(0)_STATE_INT[DDRFULL] set. Edge-sensitive interrupt, so software should clear [STATE_DDRFULL] before clearing OCLA(0)_STATE_INT[DDRFULL].
<13>	STATE_WMARK	R/W1C/H	0	0	Set when OCLA(0)_STATE_INT[WMARK] set. Edge-sensitive interrupt, so software should clear [STATE_WMARK] before clearing OCLA(0)_STATE_INT[WMARK].
<12>	STATE_OVERFULL	R/W1C/H	0	0	Set when OCLA(0)_STATE_INT[OVERFULL] set. Edge-sensitive interrupt, so software should clear [STATE_OVERFULL] before clearing OCLA(0)_STATE_INT[OVERFULL].
<11>	STATE_TRIGFULL	R/W1C/H	0	0	Set when OCLA(0)_STATE_INT[TRIGFULL] set. Edge-sensitive interrupt, so software should clear [STATE_TRIGFULL] before clearing OCLA(0)_STATE_INT[TRIGFULL].
<10>	STATE_CAPTURED	R/W1C/H	0	0	Set when OCLA(0)_STATE_INT[CAPTURED] set. Edge-sensitive interrupt, so software should clear [STATE_CAPTURED] before clearing OCLA(0)_STATE_INT[CAPTURED].
<9>	STATE_FSM1_INT	R/W1C/H	0	0	Set when OCLA(0)_STATE_INT[FSM1_INT] set. Edge-sensitive interrupt, so software should clear [STATE_FSM1_INT] before clearing OCLA(0)_STATE_INT[FSM1_INT].
<8>	STATE_FSM0_INT	R/W1C/H	0	0	Set when OCLA(0)_STATE_INT[FSM0_INT] set. Edge-sensitive interrupt, so software should clear [STATE_FSM0_INT] before clearing OCLA(0)_STATE_INT[FSM0_INT].
<7:5>	STATE_MCDX	R/W1C/H	0x0	0x0	Set when OCLA(0)_STATE_INT[MCD<b>] set. Edge-sensitive interrupts, so software should clear [STATE_MCDX<b>] before clearing OCLA(0)_STATE_INT[MCD<b>].
<4>	STATE_TRIG	R/W1C/H	0	0	Set when OCLA(0)_STATE_INT[TRIG] set. Edge-sensitive interrupt, so software should clear [STATE_TRIG] before clearing OCLA(0)_STATE_INT[TRIG].
<3:0>	STATE_OVFLX	R/W1C/H	0x0	0x0	Set when OCLA(0)_STATE_INT[OVFL<b>] set. Edge-sensitive interrupts, so software should clear [STATE_OVFLX<b>] before clearing OCLA(0)_STATE_INT[OVFL<b>].

Cavium Confidential For Bowen Yang 2017/3/1

CIU Delivery OCLA Enable Register

CIU_CIB_OCLA(0)_EN(0)

See Table 13–1 for the addresses.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:15>	—	RAZ	—	—	Reserved.
<14>	STATE_DDRFULL	R/W	0	0	Enable corresponding CIU_CIB_OCLA_RAW bit.
<13>	STATE_WMARK	R/W	0	0	Enable corresponding CIU_CIB_OCLA_RAW bit.
<12>	STATE_OVERFULL	R/W	0	0	Enable corresponding CIU_CIB_OCLA_RAW bit.
<11>	STATE_TRIGFULL	R/W	0	0	Enable corresponding CIU_CIB_OCLA_RAW bit.
<10>	STATE_CAPTURED	R/W	0	0	Enable corresponding CIU_CIB_OCLA_RAW bit.
<9>	STATE_FSM1_INT	R/W	0	0	Enable corresponding CIU_CIB_OCLA_RAW bit.
<8>	STATE_FSM0_INT	R/W	0	0	Enable corresponding CIU_CIB_OCLA_RAW bit.
<7:5>	STATE_MCDX	R/W	0x0	0x0	Enable corresponding CIU_CIB_OCLA_RAW bit.
<4>	STATE_TRIG	R/W	0	0	Enable corresponding CIU_CIB_OCLA_RAW bit.
<3:0>	STATE_OVFLX	R/W	0x0	0x0	Enable corresponding CIU_CIB_OCLA_RAW bit.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 14

Reset Unit (RST)

This chapter contains the following subjects:

- Core Availability and Reset
- Reset/Fuse Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The Reset (RST) unit controls the CN71XX chip reset. For information on resetting the individual units within the CN71XX, refer to the appropriate chapter.

The registers for the Fusing unit (MIO_FUS) are also covered in this chapter.

14.1 Core Availability and Reset

The number of cores available on CN71XX is the number of bits set in CIU_FUSE. Note that the set bits may not be contiguous.

CIU_PP_RST indicates the cores currently in reset on CN71XX. (When a bit is set, the corresponding core in CN71XX is in reset.) Except for the EJTAG TAP control register processor reset bit (ECR[PrRst]) local to each core, CIU_PP_RST is the sole reset source for the cores. Refer to the MIPS specifications and Chapter 2. CIU_PP_RST should not be changed while CIU_PP_RST_PENDING is not zero. This register indicates that power management logic is currently changing for a given processor.

Note that the reset value of CIU_PP_RST depends on the external pin REMOTE_BOOT.

- When the external pin REMOTE_BOOT is asserted, all cores are held in reset after any chip hard or soft reset (i.e. CIU_PP_RST resets to 0xF).
- When the external pin REMOTE_BOOT is deasserted, core 0 is not held in reset after any chip hard or soft reset, but other cores are (i.e. CIU_PP_RST resets to 0xE).

Cores are taken out of reset by writing a 0 to the corresponding bit in CIU_PP_RST. This can be done either by a core or by a remote PCIe host.

Cores can be put into reset, but CN71XX's stability cannot be guaranteed if the core is in the middle of any CMI, I/O, or low-latency transaction at the time of the reset assertion for the core.

14.1.1 Chip Soft-Reset Initiation

To initiate a chip-wide soft reset, core or remote host software should read RST_SOFT_RST, then write a 1 to RST_SOFT_RST[SOFT_RST] to initiate the soft reset.

When all RST_CTL(0..2)[HOST_MODE] = 0x1, the effects of chip soft reset are similar to both cold and warm reset. (A cold reset occurs whenever the PLL_DC_OK signal is deasserted. A chip warm reset occurs whenever CHIP_RESET_L signal asserts and deasserts while PLL_DC_OK remains asserted.) One difference is that RST_BOOT[LBOOT<8>] is set to indicate the chip soft reset. Software can consult RST_BOOT[LBOOT] values to determine the cause of any chip reset.

When RST_CTL(0..2)[HOST_MODE] = 0x0, PCIe port *n* is not reset during a chip soft reset. This means that PCICONFIGEP*n*-type and PEM*n** RSL-type registers are not reset during a soft reset.

When RST_CTL(0..2)[HOST_MODE] = 0x0 and a remote host initiates the CN71XX chip soft reset (via a store operation to RST_SOFT_RST), CN71XX can only service PCICONFIGEP*n* read/write operations from remote host *n* until it fully completes its internal reset sequence. During this time, the remote host should not access any other CN71XX resources. CN71XX can take as long as 1 ms to complete the reset sequence after it receives the RST_SOFT_RST store. Refer to Section 22.5.2.

Cavium Confidential FOR Bowen Yang ALU 2/2/2014

14.2 Reset/Fuse Registers

The reset and fuse registers are listed in Table 14–1.

Table 14–1 MIO_FUS, RST Registers

Registers	Address[1]	CSR Type[2]	Detailed Description
MIO_FUS_DAT0	0x0001180000001400	RSL	See page 787
MIO_FUS_DAT1	0x0001180000001408	RSL	See page 787
MIO_FUS_DAT2	0x0001180000001410	RSL	See page 788
MIO_FUS_DAT3	0x0001180000001418	RSL	See page 789
MIO_FUS_DAT4	0x0001180000001420	RSL	See page 789
MIO_FUS_PDF	0x0001180000001428	RSL	See page 790
MIO_FUS_TGG	0x0001180000001430	RSL	See page 790
MIO_FUS_RCMD	0x0001180000001500	RSL	See page 790
MIO_FUS_WADR	0x0001180000001508	RSL	See page 791
MIO_FUS_BNK_DAT(0..1)	0x0001180000001520 + α*0x8	RSL	See page 791
MIO_FUS_READ_TIMES	0x0001180000001570	RSL	See page 792
MIO_FUS_PLL	0x0001180000001580	RSL	See page 793
RST_BOOT	0x0001180006001600	RSL	See page 794
RST_DELAY	0x0001180006001608	RSL	See page 795
RST_CFG	0x0001180006001610	RSL	See page 795
RST_INT	0x0001180006001628	RSL	See page 795
RST_CKILL	0x0001180006001638	RSL	See page 796
RST_CTL(0..2)	0x0001180006001640 + α*0x8	RSL	See page 796
RST_SOFT_RST	0x0001180006001680	RSL	See page 797
RST_SOFT_PRST(0..2)	0x00011800060016C0 + α*0x8	RSL	See page 797
RST_PP_POWER	0x0001180006001700	RSL	See page 798

1. For registers with multiple versions, α represents the values within the specified range.
2. NCB-type registers are accessed directly across the IOI, and RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO Fuse Data0 Register
MIO_FUS_DAT0

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved.
<31:0>	MAN_INFO	RO	—	—	Fuse information - manufacturing info [31:0]

MIO Fuse Data1 Register
MIO_FUS_DAT1

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved.
<31:0>	MAN_INFO	RO	—	—	Fuse information - manufacturing info [63:32]

MIO Fuse Data2 Register
MIO_FUS_DAT2

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved.
<47>	FUS118	RO	—	—	Fuse information - Ignore Authentik disable.
<46:37>	ROM_INFO	RO	—	—	Fuse information - ROM info.
<36:35>	POWER_LIMIT	RO	—	—	Fuse information - Power limit.
<34>	DORM_CRYPTO	RO	—	—	Fuse information - Dormant encryption enable.
<33>	FUS318	RO	—	—	Reserved.
<32>	RAID_EN	RO	—	—	Fuse information - RAID enabled.
<31:29>	—	RO	—	—	Spare fuses.
<28>	NODFA_CP2	RO	—	—	Fuse information - HFA disable (CP2).
<27>	NOMUL	RO	—	—	Fuse information - VMUL disable.
<26>	NOCRYPTO	RO	—	—	Fuse information - DORM_CRYPTO and NOCRYPTO together select one of four mutually-exclusive modes: DORM_CRYPTO = 0, NOCRYPTO = 0 AES/DES/HASH enabled. DORM_CRYPTO = 0, NOCRYPTO = 1 AES/DES/HASH disabled. DORM_CRYPTO = 1, NOCRYPTO = 0 Dormant encryption enable. DORM_CRYPTO = 1, NOCRYPTO = 1 Authentik mode.
<25:24>	—	RO	—	—	Spare fuses.
<23:16>	CHIP_ID	RO	—	—	Fuse information - chip ID.
<15:0>	—	RO	—	—	Spare fuses.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO Fuse Data3 Register
MIO_FUS_DAT3

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:58>	EMA0	RO	—	—	Fuse information - EMA0.
<57:48>	PLL_CTL	RO	—	—	Fuse information - PLL control.
<47:45>	DFA_INFO_DTE	RO	—	—	Fuse information - HFA information (HTE).
<44:41>	DFA_INFO_CLM	RO	—	—	Fuse information - HFA information (cluster mask).
<40:38>	—	RAZ	—	—	Reserved.
<37>	EFUS_LCK_RSV	RO	—	—	Fuse information - efuse lockdown.
<36>	EFUS_LCK_MAN	RO	—	—	Fuse information - efuse lockdown.
<35>	PLL_HALF_DIS	RO/H	—	—	Fuse information - Coprocessor-clock PLL control.
<34:32>	L2C_CRIP	RO	—	—	Fuse information - L2C cripple: 0x0 = Full cache (4-way 512KB) 0x1 = 3/4 ways (3-way 384KB) 0x2 = 1/2 ways (2-way 256KB) 0x3 = 1/4 ways (1-way 128KB) 0x4–0x7 = Reserved
<31>	SPARE	RO	—	—	Spare fuse.
<30:29>	ZIP_INFO	RO	—	—	Fuse information - Zip information.
<28>	BAR2_EN	RO	—	—	Fuse information - BAR2 present (when blown ‘1’).
<27>	EFUS_LCK	RO	—	—	Fuse information - efuse lockdown.
<26>	EFUS_IGN	RO	—	—	Fuse information - efuse ignore.
<25>	NOZIP	RO	—	—	Fuse information - ZIP disable.
<24>	NODFA_DTE	RO	—	—	Fuse information - HFA disable (HTE).
<23:18>	EMA1	RO	—	—	Fuse information - EMA1.
<17:0>	—	RAZ	—	—	Reserved.

MIO Fuse Data4 Register
MIO_FUS_DAT4

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	GLOBAL_RCLK_BYP_SELECT	RO	0	—	Reserved.
<62:52>	GLOBAL_RCLK_BYP_SETTING	RO	0x0	—	Bits 11:1. Reserved.
<51>	EAST_RCLK_BYP_SELECT	RO	0	—	Reserved.
<50:39>	EAST_RCLK_BYP_SETTING	RO	0x0	—	Reserved.
<38>	CMB_RCLK_BYP_SELECT	RO	0	—	Reserved.
<37:26>	CMB_RCLK_BYP_SETTING	RO	0x0	—	Reserved.
<25>	PP_RCLK_BYP_SELECT	RO	0	—	Reserved.
<24:13>	PP_RCLK_BYP_SETTING	RO	0x0	—	Reserved.
<12>	TAD_RCLK_BYP_SELECT	RO	0	—	Reserved.
<11:0>	TAD_RCLK_BYP_SETTING	RO	0x0	—	Reserved.

Cavium Confidential For Bowen Yang ALU 02/7/2014

MIO Fuse Product Definition Field Register
MIO_FUS_PDF

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	PDF	RO	—	—	Fuse information—product definition field.

MIO Fuse TGG Register
MIO_FUS_TGG

This register exists to support Authentik. Authentik code should read this register, then clear VAL to prevent other software from observing the value of the TGG fuses.See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	VAL	R/W/H	—	—	Software may write VAL to zero, but cannot write VAL to a one. When VAL is a 1, DAT reads the corresponding TGG fuses. When VAL is a zero, DAT reads as zeros. The reset value of this VAL bit is normally its fuse setting (i.e. TGG<63>).
<62:0>	DAT	RO/H	—	—	When VAL is clear, DAT will always read as zero, regardless of the value of the TGG<62:0> fuses. When VAL is set, DAT will return the value of the TGG<62:0> fuses.

MIO Fuse Read Command Register
MIO_FUS_RCMD

To read an efuse, software writes MIO_FUS_RCMD[ADDR,PEND] with the byte address of the fuse in question, then software can poll MIO_FUS_RCMD[PEND]. When PEND is clear, then MIO_FUS_RCMD[DAT] is valid. In addition, if the efuse read went to the efuse banks (e.g. (ADDR/16) not {0,1,7}) || EFUSE), software can read MIO_FUS_BNK_DAT*x* which contains all 128 fuses in the bank associated in ADDR. See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:24>	—	RAZ	—	—	Reserved.
<23:16>	DAT	RO/H	—	—	8 bits of fuse data.
<15:13>	—	RAZ	—	—	Reserved.
<12>	PEND	R/W/H	0	0	Software sets this bit on a write to start FUSE read operation. Hardware clears when read is complete and the DAT is valid.
<11>	—	RAZ	—	—	Reserved.
<10:9>	ADDR_HI	R/W	0x0	0x0	Upper fuse address bits to extend space beyond 2k fuses. Valid range is 0x0-0x2.
<8>	EFUSE	R/W	0	0	When set, return data from the efuse storage rather than the local storage.
<7:0>	ADDR	R/W	0x0	0x0	The byte address of the fuse to read.

Cavium Confidential For Powerkang ALU 02/27/2014

MIO Fuse Read Command Register
MIO_FUS_WADR

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved.
<5:0>	ADDR	R/W	—	—	Indicates which of the banks of 128 fuses to blow.

MIO Fuse Bank Store Register
MIO_FUS_BNK_DAT(0..1)

The initial state of MIO_FUS_BNK_DAT0/1 is as if bank6 were just read, i.e. DAT* = fus[895:768]. See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	DAT	R/W/H	—	—	Efuse bank store. For read operations, the DAT gets the fus bank last read. For write operations, the DAT determines which fuses to blow.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

MIO Fuse Read Times Register
MIO_FUS_READ_TIMES

IFB fuses are 0 to 1791. The reset values are for IFB fuses for PLL_REF_CLK up to 100MHz when core PLL is engaged. If any of the formulas below result in a value less than zero, the corresponding timing parameter should be set to zero.

Prior to issuing a read to the fuse banks (via MIO_FUS_RCMD), this register should be written with the timing parameters that will be read.

This register should not be written while MIO_FUS_RCMD[PEND] = 1. See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved.
<31:28>	DONE	R/W	0x0	—	Hold time of CSB, PGENB, and LOAD with respect to falling edge of STROBE for read and write mode in PLL_REF_CLK + 1 cycles. Timing specs are th_CS = 6ns, th_PG = 10ns, th_LD_p = 7ns. Default of 0x0 yields 20ns for a PLL_REF_CLK of 50 MHz, 10ns at 100MHz.
<27:24>	AHD	R/W	0x0	—	Hold time of A with respect to falling edge of STROBE for read and write modes in PLL_REF_CLK + 2 cycles. Timing spec of tsu_A_r and tsu_A_p is 3ns min. Default of 0x0 yields 40ns for a PLL_REF_CLK of 50 MHz, 20ns at 100MHz.
<23:12>	WRSTB_WH	R/W	0x1F3	—	Pulse width high of STROBE in write mode in PLL_REF_CLK + 1 cycles. Timing spec of twh_SB_p is 9.8us max. Default of 0x1F3 yields 10 us at PLL_REF_CLK of 50 MHz.
<11:8>	RDSTB_WH	R/W	0x1	—	Pulse width high of STROBE in read mode in PLL_REF_CLK + 1 cycles. Timing spec of twh_SB_p is 20ns min. Default of 0x1 yields 40 ns at PLL_REF_CLK of 50 MHz, 20ns at 100MHz.
<7:4>	ASU	R/W	0x1	—	Setup time of A to rising edge of STROBE for read and write modes in PLL_REF_CLK cycles. Timing spec of tsu_A_r and tsu_A_p is 12 ns min. Default of 0x1 yields 40 ns at PLL_REF_CLK of 50 MHz, 20ns at 100MHz.
<3:0>	SETUP	R/W	0x0	—	Setup of CSB, PGENB, LOAD to rising edge of STROBE in read and write modes in PLL_REF_CLK + 1 cycles. tsu_CS = 16ns, tsu_PG = 14ns, tsu_LD_r = 10ns. Default of 0x0 yields 20 ns plus ASU cycles at PLL_REF_CLK of 50 MHz, 10ns + ASU at 100MHz.

Cavium Confidential For Bowen Yang ALD 02/27/2014

MIO Fuse PLL Register
MIO_FUS_PLL

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:15>	—	RAZ	—	—	Reserved.
<14:12>	CORE_STATUS	RO	—	—	Core clock PLL status information.
<11>	—	RAZ	—	—	Reserved.
<10:8>	PNR_STATUS	RO	—	—	PNR clock PLL status information.
<7>	C_COUT_RST	R/W	0	—	Core clockout postscaler reset. The core clockout postscaler should be placed in reset at least 10 ref-clock cycles prior to changing C_COUT_SEL. The core clockout postscaler should remain under reset for at least 10 ref-clock cycles after C_COUT_SEL changes.
<6:5>	C_COUT_SEL	R/W	0x0	—	Core clockout select: 0x0 = core clock 0x1 = PS output 0x2 = PLL output 0x3 = undivided core clock
<4>	PNR_COUT_RST	R/W	0	—	PNR clockout postscaler reset. The PNR clockout postscaler should be placed in reset at least 10 ref-clock cycles prior to changing PNR_COUT_SEL. The PNR clockout postscaler should remain under reset for at least 10 ref-clock cycles after PNR_COUT_SEL changes.
<3:2>	PNR_COUT_SEL	R/W	0x0	—	PNR clockout select: 0x0 = coprocessor clock 0x1 = PS output 0x2 = PLL output 0x3 = undivided core clock
<1:0>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang ALU 02/27/2014

RST Boot Register
RST_BOOT

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	CHIPKILL	R/W1S	0	0	A 0->1 transition of CHIPKILL starts the CHIPKILL timer. When CHIPKILL=1 and the timer expires, internal chip reset is asserted forever until the next chip reset. The CHIPKILL timer can be stopped only by a chip (cold, warm, soft) reset. The length of the CHIPKILL timer is specified by RST_CKILL[TIMER].
<62>	JTCSRDIS	R/W	—	0	When set, internal CSR access via JTAG TAP controller is disabled This field resets to 1 in Authentik mode, else 0.
<61>	EJTAGDIS	R/W	—	0	When set, external EJTAG access is disabled. This field resets to 1 in Authentik mode, else 0.
<60>	ROMEN	R/W	—	0	When set, Authentik/eMMC boot ROM is visible in the boot bus address space. This field resets to 1 in an Authentik part or when booting from eMMC. Else, resets to 0.
<59>	CKILL_PPDIS	R/W	1	—	When set, cores other than 0 are disabled during a CHIPKILL. Writes have no effect when RST_BOOT[CHIPKILL]=1.
<58>	JT_TSTMODE	RO	—	—	JTAG test mode.
<57>	VRM_ERR	RO	—	—	VRM did not complete operations within 5.25mS of DCOK being asserted. PLLs were released automatically.
<56:37>	—	RAZ	—	—	Reserved.
<36:30>	C_MUL	RO/H	—	—	Core-clock multiplier. C_MUL = (core-clock speed) / (ref-clock speed). “ref-clock speed” should always be 50MHz.
<29:24>	PNR_MUL	RO/H	—	—	Coprocessor-clock multiplier. PNR_MUL = (coprocessor-clock speed) /(ref-clock speed). “ref-clock speed” should always be 50MHz. For PCIe Gen1, the coprocessor-clock speed must be greater than 250MHz; for PCIe Gen2, the coprocessor-clock speed must be greater than 500MHz; for PCIe Gen3, the coprocessor-clock speed must be greater than 800MHz.
<23:18>	—	RAZ	—	—	Reserved.
<17:12>	LBOOT_EXT	R/W1C	—	—	Last boot cause mask; resets only with DCOK. <17> Warm reset due to Cntl3 link-down or hot-reset. <16> Warm reset due to Cntl2 link-down or hot-reset. <15> Cntl3 reset due to PERST3_L pin. <14> Cntl2 reset due to PERST2_L pin. <13> Warm reset due to PERST3_L pin. <12> Warm reset due to PERST2_L pin.
<11:2>	LBOOT	R/W1C/H	—	—	Last boot cause mask; resets only with DCOK. <11> Soft reset due to watchdog. <10> Soft reset due to RST_SOFT_RST write. <9> Warm reset due to Cntl0 link-down or hot-reset. <8> Warm reset due to Cntl1 link-down or hot-reset. <7> Cntl1 reset due to PERST1_L pin. <6> Cntl0 reset due to PERST0_L pin. <5> Warm reset due to PERST1_L pin. <4> Warm reset due to PERST0_L pin. <3> Warm reset due to CHIP_RESET_L pin. <2> Cold reset due to PLL_DC_OK pin.
<1>	RBOOT	R/W	—	—	Determines whether core 0 remains in reset after chip cold/warm/soft reset.
<0>	RBOOT_PIN	RO/H	—	—	Read-only access to REMOTE_BOOT pin.

Cavium Confidential For Bowen Wang 04/12/2016 At Dongguan

RST Delay Register
RST_DELAY

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved.
<31:16>	WARM_RST_DLY	R/W	0x7FF	—	Warm reset delay. A warm reset immediately causes an early warm-reset notification, but the assertion of warm reset is delayed this many coprocessor-clock cycles. A warm/soft reset will not change this field. **NOTE:** This must be at least 500 DCLK cycles.
<15:0>	SOFT_RST_DLY	R/W	0x7FF	—	Soft reset delay. A soft reset immediately causes an early soft-reset notification, but the assertion of soft reset is delayed this many coprocessor-clock cycles. A warm/soft reset will not change this field. **NOTE:** This must be at least 500 DCLK cycles.

RST Configuration Register
RST_CFG

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	BIST_DELAY	RO/H	—	—	Reserved.
<5:3>	—	RAZ	—	—	Reserved.
<2>	CNTL_CLR_BIST	R/W	0	0	Perform clear BIST during control-only reset, instead of a full BIST. A warm/soft reset will not change this field.
<1>	WARM_CLR_BIST	R/W	0	0	Perform clear BIST during warm reset, instead of a full BIST. A warm/soft reset does not change this field. Note that a cold reset always performs a full BIST.
<0>	SOFT_CLR_BIST	R/W	0	0	Perform clear BIST during soft reset, instead of a full BIST. A warm/soft reset does not change this field. Note that a cold reset always performs a full BIST.

RST Interrupt Register
RST_INT

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved.
<11:8>	PERST	R/W1C/H	0x0	—	PERST*_L asserted while RST_CTL*[RST_RCV] = 1 and RST_CTL*[RST_CHIP] = 0. One bit corresponds to each controller.
<7:4>	—	RAZ	—	—	Reserved.
<3:0>	RST_LINK	R/W1C/H	0x0	—	A controller link-down/hot-reset occurred while RST_CTL*[RST_LINK] = 0. Software must assert then deassert RST_SOFT_PRST*[SOFT_PRST]. One bit corresponds to each controller.

Cavium Confidential For Bowen Yang ALi 02/27/2014

RST Chipkill Timer Register
RST_CKILL

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:47>	—	RAZ	—	—	Reserved.
<46:0>	TIMER	R/W/H	0x3FFFFFFFF	—	Chipkill timer measured in SCLKs. Reads return current chipkill timer. Writes have no effect when RST_BOOT[CHIPKILL] = 1.

RST Controllers Registers
RST_CTL(0..2)

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved.
<9>	PRST_LINK	R/W	0	—	Controls whether corresponding controller link-down or hot-reset causes the assertion of RST_SOFT_PRST*[SOFT_PRST]. A warm/soft reset does not change this field. On cold reset, this field is initialized to 0.
<8>	RST_DONE	RO/H	—	—	Read-only access to controller reset status. RST_DONE is always zero (i.e. the controller is held in reset) when: • RST_SOFT_PRST*[SOFT_PRST] = 1, or • RST_RCV = 1 and PERST*_L pin is asserted.
<7>	RST_LINK	R/W	—	—	Reset link. Controls whether corresponding controller link-down reset or hot reset causes a warm chip reset. On cold reset, this field is initialized as follows: 0 when RST_CTL*[HOST_MODE] = 1 1 when RST_CTL*[HOST_MODE] = 0 Note that a link-down or hot-reset event can never cause a warm chip reset when the controller is in reset (i.e. can never cause a warm reset when RST_DONE = 0).
<6>	HOST_MODE	RO	—	—	Read-only access to the corresponding straps field indicating PCIE*_MODEPEM*n* is host. For controllers 1 and 2 this field is always set.
<5:4>	—	RAZ	—	—	Reserved.
<3>	RST_DRV	R/W	—	—	Controls whether PERST*_L is driven. A warm/soft reset does not change this field. On cold reset, this field is initialized as follows: 0 when RST_CTL*[HOST_MODE] = 0 1 when RST_CTL*[HOST_MODE] = 1 When set, CN71XX drives the corresponding PERST*_L pin. Otherwise, CN71XX does not drive the corresponding PERST*_L pin.

Cavium Confidential For Bowlyang 02/27/2024

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<2>	RST_RCV	R/W	—	—	Controls whether PERST*_L is received. A warm/soft reset will not change this field. On cold reset, this field is initialized as follows: 0 when RST_CTL*[HOST_MODE] = 1 1 when RST_CTL*[HOST_MODE] = 0 When RST_RCV is equal to 1, the PERST*_L value is received and may be used to reset the controller and (optionally, based on RST_CHIP) warm reset the chip. When RST_RCV is equal to 1 (and RST_CHIP = 0), RST_INT[PERST*] gets set when the PERST*_L pin asserts. (This interrupt can alert software whenever the external reset pin initiates a controller reset sequence.) RST_VAL gives the PERST*_L pin value when RST_RCV = 1. When RST_RCV = 0, the PERST*_L pin value is ignored.
<1>	RST_CHIP	R/W	0	—	Controls whether PERST*_L causes a chip warm reset like CHIP_RESET_L. A warm/soft reset will not change this field. On cold reset, this field is initialized to 0. When RST_RCV = 0, RST_CHIP is ignored. When RST_RCV = 1, RST_CHIP = 1, and PERST*_L asserts, a chip warm reset is generated.
<0>	RST_VAL	RO/H	—	—	Read-only access to PERST*_L. Unpredictable when RST_RCV = 0. Reads as 1 when RST_RCV is equal to 1 and the PERST*_L pin is asserted. Reads as 0 when RST_RCV = 1 and the PERST*_L pin is not asserted.

RST Soft Reset Register
RST_SOFT_RST

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	SOFT_RST	WO	0	—	When set, resets the CN71XX core. When performing a soft reset from a remote PCIe host, always read this register and wait for the results before setting SOFT_RST to 1.

RST PCIe Soft Reset Registers
RST_SOFT_PRST(0..2)

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	SOFT_PRST	R/W	—	—	Resets the PCIe logic and corresponding common logic associated with the SLI controller in all modes, not just RC mode. • If the RST_CTL*[HOST_MODE] = 0, SOFT_PRST resets to 0. • If the RST_CTL*[HOST_MODE] = 1, SOFT_PRST resets to 1. When CN71XX is configured to drive PERST*_L (i.e. RST_CTL(0..2)[RST_DRV] = 1), this controls the output value on PERST*_L.

Cavium Confidential For Bowen Yang ALU 02/7/2014

RST Core Power Gating Control Register
RST_PP_POWER

See Table 14–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved.
<3:0>	GATE	R/W	0x0	0x0	Powerdown enable. When both a bit and the corresponding CIU_PP_RST bit are set, the core has voltage removed to save power. In typical operation these bits are setup during initialization and PP resets are controlled through CIU_PP_RST. These bits may only be changed when the corresponding core is in reset using CIU_PP_RST.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Chapter 15

Precision Time Protocol (PTP) Clocks and Timestamping

This chapter contains the following subjects:

- PTP Clock
- PTP Clock Outputs
- CN71XX Outbound PTP Timestamp Architecture
- GMX Inbound PTP Timestamp Support
- PIP/IPD Inbound PTP Timestamp Support
- PKO and GMX Outbound Timestamp Support
- PTP Timestamp
- PTP Event Counter
- Controlling External Clocks
- Timestamp Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The CN71XX hardware supports very accurate timestamping in the PTP, GMX, PIP/IPD, and PKO blocks. This timestamping is suitable for use in IEEE 1588 Precision Time Protocol (PTP) or other purposes.

This chapter refers to CN71XX GPIO inputs and outputs. Refer to Chapter 33. The GPIO inputs available to the PTP logic are GPIO_RX_DAT, so are after programmable filtering and XORing — see Figure 33–1.

15.1 PTP Clock

CN71XX includes a digitally-tunable PTP clock that can be used for precision timestamping. This clock is used for timestamping at the physical packet-transfer interfaces.

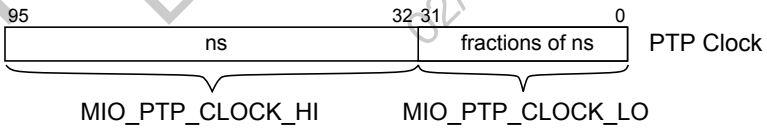
The base oscillator for the PTP clock can be one of these options:

1. The coprocessor clock. MIO_PTP_CLOCK_CFG[EXT_CLK_EN] = 0 selects this option.
2. A reference clock attached to a GPIO pin. MIO_PTP_CLOCK_CFG[EXT_-CLK_EN] = 1 selects this option when the proper MIO_PTP_-CLOCK_CFG[EXT_CLK_IN] value is used. The frequency of this reference clock may be somewhere in the 10-MHz to 200-MHz range. Any GPIO pin and DLM*_REF_CLK can be used for this function.
3. A reference clock attached to chip input pin DLM*_REF_CLK*. MIO_PTP_-CLOCK_CFG[EXT_CLK_EN] = 1 selects this option when the proper MIO_PT-P_CLOCK_CFG[EXT_CLK_IN] value is used.

The full precision of the PTP clock is 96 bits. A timestamp is a sample of the upper 64-bits of the full PTP clock. By convention, the timestamp represents a count of nanoseconds, and the lower (hidden) 32-bits of the PTP clock represents fractions of nanoseconds. This is only one of many possible conventions.

Regardless of the convention used, the GMX hardware uses the upper 64-bits of the full 96-bit PTP clock for the timestamp. Figure 15–1 shows the PTP clock.

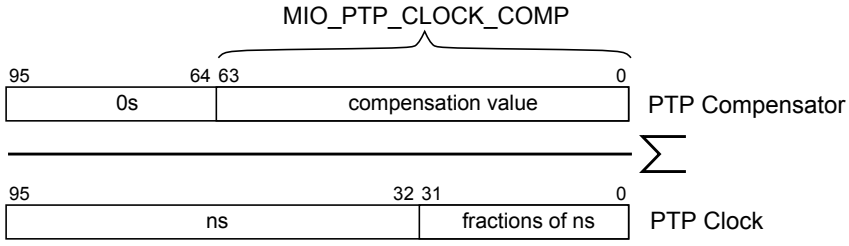
Figure 15–1 PTP Clock



CN71XX hardware implements digital tuning around the base oscillator for the timestamp clock. It does this by periodically adding a programmable 64-bit compensation value to a 96-bit PTP clock. Figure 15–2 shows the process, assuming the convention that the timestamp represents nanoseconds.

Cavium Confidential For Bowen Yang 02/27/2014

Figure 15–2 Digital Tuning to the PTP Clock



When MIO_PTP_CLOCK_CFG[EXT_CLK_EN] = 0 (choice 1 using the coprocessor clock), the CN71XX hardware adds the compensator to the PTP clock add every coprocessor clock cycle.

When MIO_PTP_CLOCK_CFG[EXT_CLK_EN] = 1 (not using the coprocessor clock), the CN71XX hardware adds the compensator on one or both of the rising and falling base oscillator edges. MIO_PTP_CLOCK_CFG[EXT_CLK_EDGE] selects which edge(s) cause compensator adds.

Larger compensation values speed up the PTP clock relative to the base oscillator, while smaller compensation values slow down the PTP clock relative to the base oscillator. Thus, by changing the compensation value, software can tune the PTP clock speed to match an external source, such as an IEEE 1588 grand-master clock.

The 64-bit compensation value can be updated at any point via a CSR write. To implement the nanosecond convention shown in Figure 15–1, the compensation value will generally be $2^{32} \times T$, where T is the time between compensation adds (in nanoseconds, on average). T is one internal coprocessor clock cycle when MIO_PTP_CLOCK_CFG[EXT_CLK_EN]=0. Otherwise, T is generally one or one half of the supplied base oscillator clock cycles, depending on MIO_PTP_CLOCK_CFG[EXT_CLK_EDGE].

Using the 64-bit compensation value, software can tune the long-term PTP clock rate very precisely (within 2^{-32} ns per T when using the nanosecond convention). Long-term PTP clock precision can be largely a function of the stability of the base oscillator and the frequency that software returns to match a grand-master clock.

Short-term PTP clock precision is better with a smaller T. The smallest possible T occurs when MIO_PTP_CLOCK_CFG[EXT_CLK_EN]=0. When MIO_PTP_CLOCK_CFG[EXT_CLK_EN]=1, a faster base oscillator and/or using more edges gives a smaller T and better short-term precision.

Software can tune the PTP clock by changing the compensation value in MIO_PTP_CLOCK_COMP. Software can directly observe the PTP clock (nanoseconds portion) via reading MIO_PTP_CLOCK_HI, but may typically only observe it indirectly via timestamp reception. Software can also directly update the PTP clock via MIO_PTP_CLOCK_HI writes at any point.

Cavium Confidential For Power Long ALU 12/17/2014

15.2 PTP Clock Outputs

CN71XX can source/output clocks derived from its PTP clock in these ways:

- The PTP PPS logic can output a clock sourced from the internal PTP clock to a GPIO output.
- The PTP CKOUT logic can output a clock based on the internal PTP clock to a GPIO output.

The PTP PPS and CKOUT clocks are both digitally-produced clocks that work as follows:

- For each, there is a 96-bit threshold to be compared to the 96-bit PTP clock (MIO_PTP_PPS/CKOUT_THRESH_HI,MIO_PTP_PPS/CKOUT_THRESH_LO), and two corresponding 64-bit incrementors somewhat akin to the compensation value (MIO_PTP_PPS/CKOUT_HI_INCR and MIO_PTP_PPS/CKOUT_LO_INCR).
- The corresponding MIO_PTP_CLOCK_CFG[PPS_EN/CKOUT_EN]=1 enables the PPS/CKOUT clock.
- When the clock is enabled and the 96-bit threshold is smaller than the 96-bit PTP clock, the PTP PPS/CKOUT logic creates a rising or falling edge on the clock and increments the 96-bit threshold. If a rising edge, MIO_PTP_PPS/CKOUT_HI_INCR is the increment. If a falling edge, MIO_PTP_PPS/CKOUT_LO_INCR is the increment. So MIO_PTP_PPS/CKOUT_HI_INCR is the width of the high phase of the clock and MIO_PTP_PPS/CKOUT_LO_INCR is the width of the low phase of the clock.
- Outside of all the above, MIO_PTP_CLOCK_CFG[PPS_INV/CKOUT_INV] can subsequently invert the corresponding produced clock.
- GPIO_BIT_CFG*[OUTPUT_SEL] specifies which GPIO PPS/CKOUT will be sent out.
- Software can read the current clock value (subsequent to the conditional MIO_PTP_CLOCK_CFG[PPS_INV/CKOUT_INV] inversion) via reading MIO_PTP_CLOCK_CFG[PPS/CKOUT].

Following is an example, with 1PPS (pulse per second) signal, using the coprocessor clock (600Mhz), generate 1s 50% duty cycle signal, reports to GPIO0.

1. MIO_PTP_CLOCK_CFG:

   [PTP_EN] = 1

   [EXT_CLK_EN] = 0

   [PPS_EN] = 1

2. PTP_CLOCK_HI, PTP_CLOCK_LO: do not modify
3. MIO_PTP_PPS_THRESH_LO: 0x0

   MIO_PTP_PPS_THRESH_HI: 0x0

4. MIO_PTP_PPS_HI_INCR: 0x1DCD650000000000 (match .5s)

   MIO_PTP_PPS_LO_INCR: 0x1DCD650000000000 (match .5s)

5. MIO_PTP_CLOCK_COMP: 0x1AAAAAAAA (match 1.67 ns, which is 600MHz)
6. GPIO_BIT_CFG0[OUTPUT_SEL] = 0x2

Cavium Confidential For Al Li 02/27/2014 Bowen Yang

15.3 CN71XX Outbound PTP Timestamp Architecture

The CN71XX outbound timestamping hardware captures timestamps, but cannot insert the timestamp into any outgoing packets. Using IEEE 1588 vernacular, this means that the CN71XX outbound timestamping hardware is most useful when implementing a two-step clock, not a one-step clock, for PTP event messages sent from CN71XX. After the hardware captures the timestamp for the outgoing 1588 event message and delivers it to software, the 1588 software running on the CN71XX may need to send the timestamp to required recipients in a subsequent 1588 general message as a second step.

15.4 GMX Inbound PTP Timestamp Support

GMX supports a mode where it timestamps every received packet. In this mode (GMX*_RX*_FRM_CTL[PTP_MODE]=1), GMX samples the PTP clock as it starts receiving each packet, then locally prepends this 64-bit timestamp as an additional eight-byte header at the beginning of the packet. Finally, GMX forwards the timestamp plus packet to PIP/IPD as a single packet that is eight bytes longer than the actual packet that was received.

Chapters 21 and 22 describes GMX's processing in detail.

15.5 PIP/IPD Inbound PTP Timestamp Support

PIP/IPD has no specific mode supporting PTP timestamping, but contains a number of features to compensate for the PTP timestamp introduced by GMX when it is timestamping.

When GMX is in the receive-timestamp mode, PIP/IPD receive the timestamp plus packet from the GMX. PIP/IPD packet parsing is able to skip over the received timestamps (the Skip I field). PIP/IPD MAXERR, MINERR, and LENERR checks can compensate for the additional header, since the MAXERR and MINERR byte counts are programmable, and the programmed value can easily be increased by 8. The LENERR check already compensates based on the skip value.

PIP/IPD makes the timestamp available for software with every packet. The timestamp can be present in the work-queue entry and/or the packet buffer in L2/DRAM for software, depending on PIP/IPD configuration. The software will likely use the timestamp when IEEE 1588 event messages arrive. For other packets, the software may choose to discard or ignore the timestamp that is present with every packet in this mode.

Chapter 9 describes PIP/IPD in detail.

15.6 PKO and GMX Outbound Timestamp Support

Some CN71XX outbound-packet transmit operations involve the PKO and GMX blocks. PKO outbound timestamping is not supported when outputting packets to the DPI interface. PKO and GMX support timestamps for packets sent out to either SGMII/QSGMII or RXAUI interfaces.

Cavium Confidential For Bowen Yang 02-27-2014

Unlike the inbound timestamping implementation, outbound timestamping is not applied to all packets. Instead, outbound timestamping is selectively applied to only those packets that need to be timestamped. Software selects which packets are timestamped by the hardware.

1. Software inserts PKO commands into queues to send packets. These PKO commands can direct PKO to provide a timestamp to software after sending the packet (see WORD2[PTP] description in the PKO chapter). (Software could request a timestamp for outgoing IEEE 1588 event packets, for example.)
2. After PKO schedules the corresponding packet to be sent, DMAs it from L2/DRAM, and GMX starts sending it out, GMX and PKO capture the timestamp.
3. PKO returns the timestamp as directed by software. PKO can either write the timestamp directly to an arbitrary 64-bit-word-aligned L2/DRAM location, or write the timestamp to a location in a work queue entry supplied by software and submit the work queue entry to POW. (PKO_REG_TIMESTAMP[WQE_WORD] specifies where PKO will put the timestamp in a work-queue entry.)
4. Software retrieves the timestamp from L2/DRAM or the POW work-queue entry. Software can include work-queue-entry data structures to associate the received timestamp with the timestamped packet, or with any desired data structure.

Chapter 10 describes PKO in detail. Chapters 23 and 24 describes GMX in detail. Chapter 7 describes POW in detail.

The CN71XX PKO/GMX outbound timestamping may not be able to send minimum-sized timestamped packets at the maximum possible speed of the packet interfaces. The implementation assumes that outbound packet timestamps are not frequently required. A software application running on CN71XX that requests a timestamp for every outbound packet may find that small packet transfer performance is poor. Software should request a timestamp infrequently, and only when necessary.

15.7 PTP Timestamp

When MIO_PTP_CLOCK_CFG[TSTMP_EN]=1, the PTP logic samples the PTP clock into MIO_PTP_TIMESTAMP on an edge of an external input (GPIO or DLM*_REF_CLK). MIO_PTP_CLOCK_CFG[TSTMP_IN] selects the external input. MIO_PTP_CLOCK_CFG[TSTMP_EDGE] selects the edge.

This feature can be useful to very accurately compare the PTP clock to an external clock received via a GPIO input, for example. GPIOs can also cause interrupts, if that is desired. Refer to Chapter 33.

15.8 PTP Event Counter

When MIO_PTP_CLOCK_CFG[EVCNT_EN]=1, the PTP logic decrements MIO_PTP_EVT_CNT by one on an edge of a GPIO input or DLM*_REF_CLK* input. MIO_PTP_CLOCK_CFG[EVCNT_IN] selects the input. MIO_PTP_CLOCK_CFG[EVCNT_EDGE] selects the edge. When the PTP logic decrements MIO_PTP_EVT_CNT to zero, it sets CIU_INT_SUM1[PTP], which can cause an interrupt.

Software writes to MIO_PTP_EVT_CNT atomically incrementing it by the value written — these writes can be used to set MIO_PTP_EVT_CNT up for an accurate subsequent interrupt.

Cavium Confidential For Beren Years 9/2/2014

15.9 Controlling External Clocks

As Section 15.1 describes, CN71XX includes a digitally-tunable PTP clock. But some systems may instead use an externally-tunable oscillator which drives the CN71XX PTP clock through either GPIO inputs or DLM*_REF_CLK*.

These external clocks can be tuned in many ways with CN71XX. Among the options: TWSI interface(s), SPI/MPI interface(s). In all these cases, software drives the clock tuning.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

15.10 Timestamp Registers

The PTP timestamp registers are listed in Table 15–1.

Table 15–1 PTP Registers

Register	Address	CSR Type[1]	Detailed Description
MIO_PTP_CLOCK_CFG	0x0001070000000F00	NCB	See page 806
MIO_PTP_CLOCK_LO	0x0001070000000F08	NCB	See page 807
MIO_PTP_CLOCK_HI	0x0001070000000F10	NCB	See page 808
MIO_PTP_CLOCK_COMP	0x0001070000000F18	NCB	See page 808
MIO_PTP_TIMESTAMP	0x0001070000000F20	NCB	See page 808
MIO_PTP_EVT_CNT	0x0001070000000F28	NCB	See page 808
MIO_PTP_CKOUT_THRESH_LO	0x0001070000000F30	NCB	See page 808
MIO_PTP_CKOUT_THRESH_HI	0x0001070000000F38	NCB	See page 809
MIO_PTP_CKOUT_HI_INCR	0x0001070000000F40	NCB	See page 809
MIO_PTP_CKOUT_LO_INCR	0x0001070000000F48	NCB	See page 809
MIO_PTP_CKOUT_THRESH_LO	0x0001070000000F50	NCB	See page 809
MIO_PTP_CKOUT_THRESH_HI	0x0001070000000F58	NCB	See page 809
MIO_PTP_CKOUT_HI_INCR	0x0001070000000F60	NCB	See page 810
MIO_PTP_CKOUT_LO_INCR	0x0001070000000F68	NCB	See page 810

1. NCB-type registers are accessed directly across the IOI.

PTP Clock Configuration Register MIO_PTP_CLOCK_CFG

This register configures the timestamp architecture. See Table 15–1 for address

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:42>	—	RAZ	—	—	Reserved
<41>	PPS	RO	—	0	PTP PPS output; reflects ptp__pps after PPS_INV inverter.
<40>	CKOUT	RO	—	0	PTP clock output; reflects ptp__ckout after CKOUT_INV inverter.
<39:38>	EXT_CLK_EDGE	R/W	0x0	0x0	External clock input edge 00 = rising edge 01 = falling edge 10 = both rising and falling edge 11 = reserved
<37:32>	—	RAZ	—	—	Reserved
<31>	PPS_INV	R/W	0	0	Invert PTP PPS. 0 = Don't invert 1 = Invert
<30>	PPS_EN	R/W	0	0	Enable PTP PPS.
<29:26>	—	RAZ	—	—	Reserved
<25>	CKOUT_INV	R/W	0	0	Invert PTP clock out. 0 = Don't invert 1 = Invert
<24>	CKOUT_EN	R/W	0	0	Enable PTP clock out.

Cavium Confidential For Bowen Yang A.U 12/27/2011

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<23:18>	EVCNT_IN	R/W	0x0	0x0	Source for event counter input: 0x00–0x0F = GPIO*n*, where *n* is EVCNT_IN<3:0> 0x10 = DLM0_REF_CLK 0x11 = DLM1_REF_CLK 0x12 = DLM2_REF_CLK 0x13–0x1F = reserved 0x20 =GPIO<16> 0x21 =GPIO<17> 0x22 =GPIO<18> 0x23 =GPIO<19> 0x24–0x3F = reserved
<17>	EVCNT_EDGE	R/W	0	0	Event counter input edge: 0 = falling edge, 1 = rising edge
<16>	EVCNT_EN	R/W	0	0	Enable event counter.
<15:10>	TSTMP_IN	R/W	0x0	0x0	Source for timestamp input: 0x00–0x0F = GPIO*n*, where *n* is TSTMP_IN<3:0> 0x10 = DLM0_REF_CLK 0x11 = DLM1_REF_CLK 0x12 = DLM2_REF_CLK 0x13–0x1F = reserved 0x20 =GPIO<16> 0x21 =GPIO<17> 0x22 =GPIO<18> 0x23 =GPIO<19> 0x24–0x3F = reserved
<9>	TSTMP_EDGE	R/W	0	0	External timestamp input edge: 0 = falling edge, 1 = rising edge
<8>	TSTMP_EN	R/W	0	0	Enable external timestamp.
<7:2>	EXT_CLK_IN	R/W	0x0	0x0	Source for external clock: 0x00–0x0F = GPIO*n*, where *n* is EXT_CLK_IN<3:0> 0x10 = DLM0_REF_CLK 0x11 = DLM1_REF_CLK 0x12 = DLM2_REF_CLK 0x13–0x1F = reserved 0x20 =GPIO<16> 0x21 =GPIO<17> 0x22 =GPIO<18> 0x23 =GPIO<19> 0x24–0x3F = reserved
<1>	EXT_CLK_EN	R/W	0	0	Use external clock.
<0>	PTP_EN	R/W	0	0	Enable PTP module.

Cavium Confidential For Bowen Yang ATT 02/27/2014

PTP Clock Low Register
MIO_PTP_CLOCK_LO

This register provides bits<31:0> of the PTP clock. See Table 15–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	FRNANOSEC	R/W	0x0	0x0	Fractions of nanoseconds. Bits<31:0> of the PTP clock.

PTP Clock High Register
MIO_PTP_CLOCK_HI

This register provides bits<95:32> of the PTP clock. Writes to MIO_PTP_CLOCK_HI also clear MIO_PTP_CLOCK_LO. To update all 96 bits, write MIO_PTP_CLOCK_HI followed by MIO_PTP_CLOCK_LOSee Table 15–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	NANOSEC	R/W	0x0	0x0	Clock value in nanoseconds. Bits<95:32> of the PTP clock.

PTP Clock Compensator Register
MIO_PTP_CLOCK_COMP

This register provides the compensation value the PTP clock. See Table 15–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	NANOSEC	R/W	0x0	0x0	Compensation value in nanoseconds.
<31:0>	FRNANOSEC	R/W	0x0	0x0	Fractions of nanoseconds.

PTP Timestamp Register
MIO_PTP_TIMESTAMP

This register contains the timestamp latched on MIO_PTP_CLOCK_CFG[TSTMP_EDGE] edge of MIO_PTP_CLOCK_CFG[TSTMP_IN]. See Table 15–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	NANOSEC	R/W	0x0	0x0	Timestamp in nanoseconds.

PTP Event Counter Register
MIO_PTP_EVT_CNT

This register contains the PTP event counter. See Table 15–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	CNTR	R/W	0x0	0x0	Count nanoseconds. Writing to this register increments this register by the value of the write data. The register counts down by 1 for every MIO_PTP_CLOCK_CFG[EVCNT_EDGE] edge of MIO_PTP_CLOCK_CFG[EVCNT_IN]. When this register equals 0, an interrupt is generated

PTP Clock Out Low Register
MIO_PTP_CKOUT_THRESH_LO

This register contains the low bytes of the PTP clock out. See Table 15–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	FRNANOSEC	R/W	0xFFFFFFFF	0x0	Fractions of nanoseconds.

Cavium Confidential For
Bowen Yang
ALU
2014

PTP Clock Out High Register
MIO_PTP_CKOUT_THRESH_HI

This register contains the high bytes of the PTP clock out. Writes to MIO_PTP_CKOUT_THRESH_HI also clear MIO_PTP_CKOUT_THRESH_LO. To update all 96 bits, write MIO_PTP_CKOUT_THRESH_HI followed by MIO_PTP_CKOUT_THRESH_LO. See Table 15–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	NANOSEC	R/W	0xFFFFFFFFFFFFFFFF	0x0	Nanoseconds.

PTP Clock Out High Increment Register
MIO_PTP_CKOUT_HI_INCR

This register contains the high bytes of the PTP clock out increment. See Table 15–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	NANOSEC	R/W	0x0	0x0	Nanoseconds
<31:0>	FRNANOSEC	R/W	0x0	0x0	Fractions of nanoseconds.

PTP Clock Out Low Increment Register
MIO_PTP_CKOUT_LO_INCR

This register contains the low bytes of the PTP clock out increment. See Table 15–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	NANOSEC	R/W	0x0	0x0	Nanoseconds
<31:0>	FRNANOSEC	R/W	0x0	0x0	Fractions of nanoseconds.

PTP PPS Low Register
MIO_PTP_PPS_THRESH_LO

This register contains the low bytes of the PTP PPS. See Table 15–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	FRNANOSEC	R/W	0xFFFFFFFF	0x0	Fractions of nanoseconds.

PTP PPS High Register
MIO_PTP_PPS_THRESH_HI

This register contains the high bytes of the PTP PPS. See Table 15–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	NANOSEC	R/W	0xFFFFFFFFFFFFFFFF	0x0	Nanoseconds

Cavium Confidential For Bowen Yang ALU 02/27/2014

PTP PPS High Increment Register
MIO_PTP_PPS_HI_INCR

This register contains the high bytes of the PTP PPS increment. See Table 15–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	NANOSEC	R/W	0x0	0x0	Nanoseconds
<31:0>	FRNANOSEC	R/W	0x0	0x0	Fractions of nanoseconds

PTP PPS Low Increment Register
MIO_PTP_PPS_LO_INCR

This register contains the low bytes of the PTP PPS increment. See Table 15–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	NANOSEC	R/W	0x0	0x0	Nanoseconds
<31:0>	FRNANOSEC	R/W	0x0	0x0	Fractions of nanoseconds

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 16

Timer (TIM)

This chapter contains the following subjects:

- Overview
- Timer Features
- TIM Support
- Software Responsibilities
- Timer Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

This chapter discusses the CN71XX Timer block (TIM). The timer block enables software to schedule POW work-queue entries for a future time.

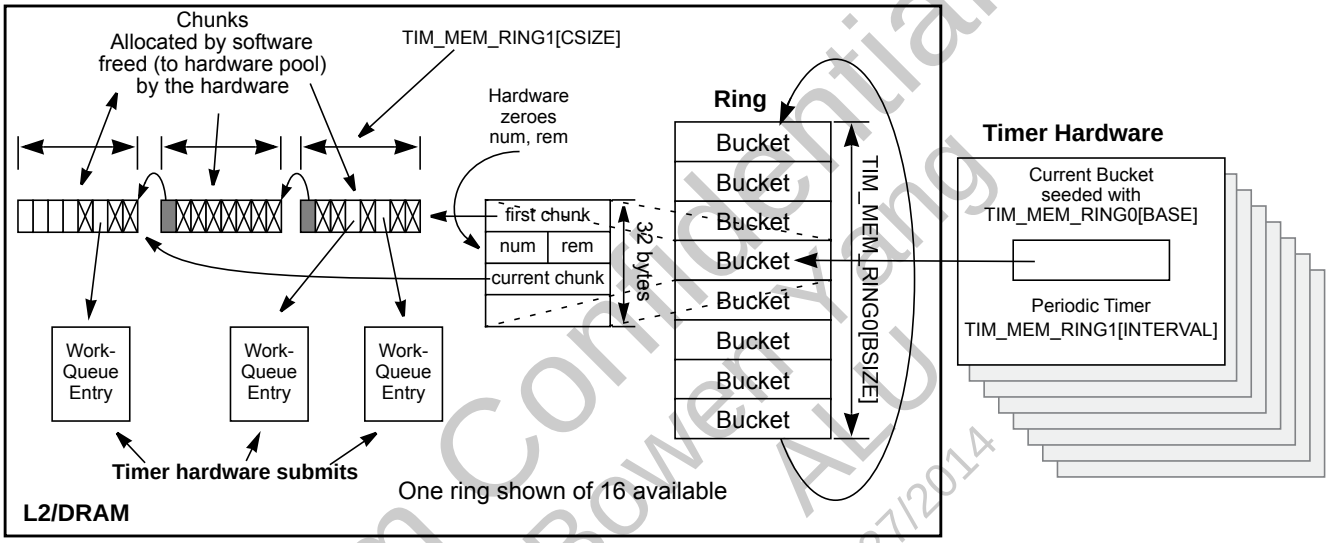
Software creates bucketed timer-list entries containing a pointer to a work-queue entry, and inserts them into a ring. Rings are typically allocated to each core to reduce locking, but this is not required by hardware. Timer-list entries may be invalidated by software before they expire.

Each ring specifies the amount of time that defines an interval. Upon interval expiration, hardware traverses the timer list for the expired bucket, and schedules work to the POW using the pointer that was provided when the timer-list entry was created.

16.1 Timer Features

Figure 16–1 shows a conceptual illustration of the CN71XX Timer Unit (TIM).

Figure 16–1 CN71XX Timer



The CN71XX TIM consists of 16 bucketed rings held in L2/DRAM.

- Software can program a separate ring per core, if desired. There is no fixed relationship between cores and timer rings.
- Each bucket corresponds to a different time slice.
- Each ring has a programmable number of buckets.
- Each ring has a programmable time interval between buckets.

At each periodic bucket time expiration, hardware processes the next bucket.

- Each timer entry within a bucket may be a work queue entry pointer that hardware submits
- A bucket is a chunked list, and hardware frees chunks (to a hardware pool) after using them
- Hardware also resets the bucket data structure

Cavium Confidential For Bowen Yang ALU 02/27/2014

The hardware traverses up to 80 million timer entries per second.

Software inserts work-queue entries into the appropriate bucket.

- Software allocates bucket chunks as needed
- Software can remove added entries from the bucket later by rewriting the entry to NULL (before the hardware processes the bucket)

16.2 TIM Support

The CN71XX support for timers is a mix of software and hardware.

Software creates bucketed timer-list entries and can later invalidate them. Each timer-list entry is a pointer to a work-queue entry that was created by software.

See "Packet Order / Work Unit (POW)" on page 391. The work queue entry can be as small as 16 bytes.

Upon expiration, hardware traverses the timer lists created previously by software. For all entries in the list that are still valid, hardware schedules the work. Hardware also initializes the list for subsequent adds. Because there are 16 timer lists, software can allocate one list for each core and avoid synchronization between cores.

See Figure 16–2. Each bucket corresponds to a timer expiration at a different time. The next time increment is the next sequential entry in the bucket array, wrapping around. The timer interval between bucket traversals is also programmable.

The space that held the timer entries is dynamically allocated by the cores and freed by hardware as it traverses the list. The list is allocated in chunks. Timer entries must be 32-byte aligned; timer chunk buffers must be 128-byte aligned.

The rings are configured via the TIM_MEM_RING0 and TIM_MEM_RING1 registers. Each ring has a separate:

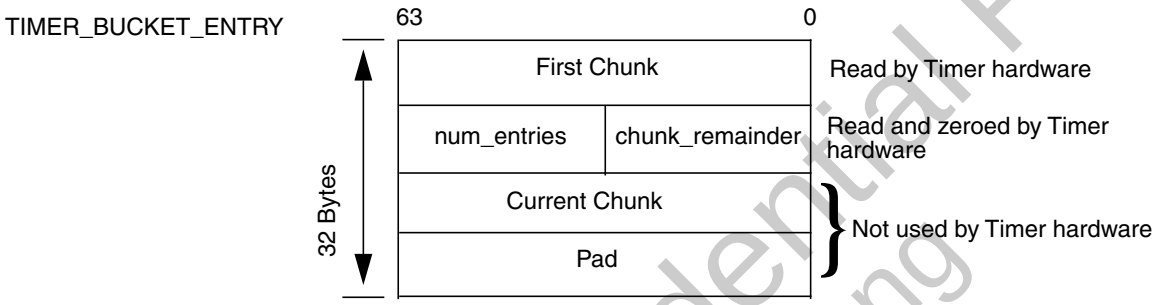
- Enable = TIM_MEM_RING1[ENA] (readable via TIM_MEM_DEBUG0[ENA]).
- Time between bucket traversals = TIM_MEM_RING1[INTERVAL] (readable via TIM_MEM_DEBUG0[INTERVAL]). TIM defines this interval in coprocessor-clock cycles.
- Base pointer (i.e. pointer to the first bucket) = TIM_MEM_RING0[BASE] (readable via TIM_MEM_DEBUG1[BASE]).
- Number of buckets in the ring (minus 1) = TIM_MEM_RING0[BSIZE] (readable via TIM_MEM_DEBUG1[BSIZE]).
- Chunk size (i.e. number of work queue entry pointers in each buffer chunk linked to the base bucket data structure) = TIM_MEM_RING1[CSIZE] (readable via TIM_MEM_DEBUG2[CSIZE]).
- FPA pool to free the chunks to (see "Free Pool Unit (FPA)" on page 439) = TIM_MEM_RING1[CPOOL] (readable via TIM_MEM_DEBUG2[CPOOL]).
- Current bucket (the number of 1024-cycle ticks before the timer expires and this bucket list is traversed) = TIM_MEM_DEBUG2[BUCKET], TIM_MEM_DEBUG1[BUCKET]
- Current time between buckets = TIM_MEM_DEBUG0[COUNT] (set to TIM_MEM_RING1[INTERVAL] and is reset by hardware). TIM measures this in coprocessor-clock cycles.

Cavium Confidential For Review Only ALL 02/27/2014

TIM_MEM_RING0[RID] and TIM_MEM_RING1[RID] select the ring when programming the above values. The following describes the data structures recognized by the CN71XX TIM:

```
//must be 128-byte aligned
struct timer_entry_chunk {
  uint64   entries[TIM_MEM_RING1[CSIZE]-1]; //if all 0s, the entry is invalid
  struct   timer_entry_chunk *next_ptr;
};
//must be 32-byte aligned
//this description assumes big-endian
struct TIMER_BUCKET_ENTRY {
  struct   timer_entry_chunk *first_chunk;
  uint32   num_entries;                 //zeroed by HW after traversing list
  uint32   chunk_remainder;             //zeroed by HW after traversing list

  //the remaining 16 bytes are not touched by hardware
  struct   timer_entry_chunk *current_chunk;
  uint64   pad;
};
```

Figure 16–2 Bucket Data Structure



The following is a sample add routine:

```
// must be 32-byte aligned
TIMER_BUCKET_ENTRY buckets[TIM_MEM_RING0[BSIZE]+1
uint64 *add_bucket(uint64 wqptr, int bucket_id) {
  TIMER_BUCKET_ENTRY *e = &buckets[bucket_id];
  uint64 *res
  uint32 remainder = e -> chunk_remainder;
  e -> num_entries++;
  if(remainder == 0) {
    TIMER_ENTRY_CHUNK *ptr = malloc_chunk();
    if(cur == NULL) e -> first_chunk = ptr;
    else e -> current_chunk -> next_ptr = ptr;
    e -> current_chunk = ptr;
    res = &(ptr -> entries[0]);
    remainder = TIM_MEM_RING1[CSIZE] - 1
  }
  else {
    res = &(e -> current_chunk -> entries[TIM_MEM_RING1[CSIZE] - remainder]);
    remainder --;
  }
  *res = wqptr
  e -> chunk_remainder = remainder
  return(res);
}
```

At some later point, but not while or after the hardware traverses the list, software can invalidate a timer entry by writing it to all 0s. In that case, hardware will not submit the work-queue entry.

Cavium Confidential For Bowen Yang ALU 04/27/2014

Note that the CN71XX TIM zeroes the num_entries and chunk_remainder fields while it traverses the list associated with a bucket. The CN71XX TIM also reads the first_chunk field, but never references the current_chunk field.

Small bucket interval times (i.e. small TIM_MEM_RING1[INTERVAL] values – zero, especially) can be too short for the TIM bucket traversal hardware, causing the hardware to fall behind and fail. When this failure happens, the hardware sets a TIM_REG_ERROR[MASK] interrupt bit. Larger bucket interval times can avoid this problem.

Software can determine the bucket where to position a work-queue entry using cycle counts. These calculations are straightforward when coprocessor-clock cycles are used, because TIM runs on the coprocessor clock. Note that the software running on the cnMIPS III cores may use core-clock-cycle counts, which complicates the calculations. Section 1.1 defines both the core clock and the coprocessor clock that are present in CN71XX.

16.3 Software Responsibilities

There are two specific race conditions that must be managed by software:

1. Software must not add to a bucket/list when the hardware might be processing the list. This is because hardware may miss the addition.

   In practice this means two things:

   - A timer entry that expires too close to the current time cannot be added.
   - A timer entry that expires too far from the current time cannot be added (i.e. you must not wrap around the buckets).
2. Software must not invalidate a timer entry when hardware might be processing the list that contains the timer entry.

This is because hardware may have already freed the memory that contains the timer entry (in which case, hardware will already have scheduled it). It is also possible that hardware has not freed the memory, but has already traversed past the relevant entry.

In practice this means that a timer entry that is going to expire “too soon” cannot be invalidated. Hardware will schedule the work queue entry anyway in this case, and software must have the appropriate data structures available to prevent it from performing any action.

A good way to prevent any action in this case is to store the expiration time of the timer in some data structure that is accessible from the work queue entry. If the stored timer expiration time is later than the current time, when hardware schedules the work-queue entry to a core, no action should be performed.

Note also that in addition to invalidation, another common timer operation is to reschedule a timer entry further into the future. This should be handled by invalidating the prior timer entry and creating a new one. If the expiration time of the current timer is “too close” for a simple invalidation, software may choose to wait and create the new entry after hardware schedules the prior one. In any case, software must not schedule the same work-queue entry simultaneously. Waiting for hardware to schedule the prior entry allows software to use only a single work-queue entry in this case. Without the wait, the new entry will likely need a new work-queue entry.

Cavium Confidential For Bowen Yang ALU 02/27/2014

16.4 Timer Registers

The timer registers are listed in Table 16–1.

Table 16–1 Timer Registers

Register	Address	CSR Type[1]	Detailed Description
TIM_REG_FLAGS	0x0001180058000000	RSL	See page 816
TIM_REG_READ_IDX	0x0001180058000008	RSL	See page 817
TIM_REG_BIST_RESULT	0x0001180058000080	RSL	See page 817
TIM_REG_ERROR	0x0001180058000088	RSL	See page 817
TIM_REG_INT_MASK	0x0001180058000090	RSL	See page 818
TIM_MEM_RING0	0x0001180058001000	RSL	See page 818
TIM_MEM_RING1	0x0001180058001008	RSL	See page 818
TIM_MEM_DEBUG0	0x0001180058001100	RSL	See page 819
TIM_MEM_DEBUG1	0x0001180058001108	RSL	See page 819
TIM_MEM_DEBUG2	0x0001180058001110	RSL	See page 819

1. RSL-type registers are accessed indirectly across the IOI.

Timer Flags Register TIM_REG_FLAGS

This register provides flags for the TIM. See Table 16–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved
<2>	RESET	RAZ	0	0	Reset. One-shot pulse for free-running structures.
<1>	ENA_DWB	R/W	0	0	Enable non-zero don't-write-backs. When set, enables the use of don't-write-backs during the buffer freeing operations.
<0>	ENA_TIM	R/W	0	0	Enable TIM. When set, TIM is in normal operation. When clear, time is effectively stopped for all rings in TIM. TIM has a counter that causes a periodic tick every 1024 cycles. This counter is shared by all rings. (Each tick causes the hardware to decrement the time offset (i.e. COUNT) for all enabled rings.) When ENA_TIM is 0, the hardware stops this shared periodic counter, so there are no more ticks, and there are no more new bucket traversals (for any ring). If ENA_TIM transitions from 1 to 0, TIM no longer creates new bucket traversals, but there may have been previous ones. If there are ring bucket traversals that were already pending but not currently active (i.e. bucket traversals that need to be done by the hardware, but haven't been yet) during this ENA_TIM 1 to 0 transition, then these bucket traversals will remain pending until ENA_TIM is later set to one. Bucket traversals that were already in progress will complete after the 1 to 0 ENA_TIM transition, though.

Cavium Confidential For Boan Yang 2/2/2015

Timer Read Index Register
TIM_REG_READ_IDX

This register provides the read index during a CSR read operation to any of the CSRs that are physically stored as memories. The mnemonics for these CSRs begin with the prefix TIM_MEM_.

INDEX[7:0] is the read index. INC[7:0] is an increment that is added to INDEX[7:0] after any CSR read. The intended use is to initially write this CSR such that INDEX = 0 and INC = 1. Then, the entire contents of a CSR memory can be read with consecutive CSR read commands. See Table 16–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:8>	INC	R/W	0	0	Increment to add to the current index for the next index.
<7:0>	INDEX	R/W	0	0	Index to use for next memory CSR read operation.

Timer BIST Result Register
TIM_REG_BIST_RESULT

This register provides access to the internal timer BIST results. Each bit is the BIST result of an individual memory (per bit, 0 = pass and 1 = fail). See Table 16–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:2>	STA	RO	—	0x0	BIST result of the STA memories.
<1>	NCB	RO	—	0	BIST result of the IOI memories.
<0>	CTL	RO	—	0	BIST result of the CTL memories.

Timer Error Register
TIM_REG_ERROR

A ring is in error if its interval has elapsed more than once without having been serviced. This register indicates whether a ring is in error. During a CSR write to this register, the write data is used as a mask to clear the selected mask bits:

$$\text{new MASK}[15:0] = \text{MASK}[15:0] \ \text{AND} \ \overline{\text{write_data}[15:0]}$$

See Table 16–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	MASK	R/W1C/H	0x0	0x0	Bit mask. A bit set to 1 indicates the corresponding ring is in error

Cavium Confidential For Bowen Yang ALU 8/27/2014

Timer Error Interrupt Mask Register
TIM_REG_INT_MASK

See Table 16–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	MASK	R/W/H	0x0	0x0	Interrupt-enable bit mask. A bit set to 1 enables the corresponding ring to interrupt when an error is detected.

Timer Ring 0 Register
TIM_MEM_RING0

This CSR is a memory of 16 entries. TIM_REG_READ_IDX must be written before any CSR read operations to this address can be performed.

TIM_MEM_RING0 must not be written for a ring when TIM_MEM_RING1[ENA] is set for the ring. Every write to TIM_MEM_RING0 clears the current bucket for the ring. (The current bucket is readable via TIM_MEM_DEBUG2[BUCKET], TIM_MEM_DEBUG1[BUCKET].) See Table 16–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:55>	—	RAZ	—	—	Reserved
<54:24>	BASE	R/W/H	0x0	0x0	Pointer to bucket 0. This field provides bits [35:5] of a 36-bit, 32-byte-aligned pointer (bits [4:0] = 0x0).
<23:4>	BSIZE	R/W/H	0x0	0x0	Bucket size. Species the number of buckets – 1.
<3:0>	RID	R/W/H	0x0	0x0	Ring ID.

Timer Ring 1 Register
TIM_MEM_RING1

This CSR is a memory of 16 entries. TIM_REG_READ_IDX must be written before any CSR read operations to this address can be performed. See Table 16–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:43>	—	RAZ	—	—	Reserved
<42>	ENA	R/W/H	0	0	Ring-timer enable. When clear, the ring is disabled and TIM will not traverse any new buckets for the ring. After a 1 to 0 transition on ENA, the hardware still completes a bucket traversal for the ring if it was pending or active prior to the transition. (Software must delay to ensure the completion of the traversal before reprogramming the ring.)
<41:39>	CPOOL	R/W/H	0x0	0x0	Chunk pool. Free list used to free chunks.
<38:26>	CSIZE	R/W/H	0x0	0x0	Chunk size. Specifies the number of 8-byte words per chunk. CSIZE must be at least 16. It is illegal to program CSIZE to a value that is less than 16.
<25:4>	INTERVAL	R/W/H	0x0	0x0	Timer interval, measured in 1024 cycle ticks. Specifies the interval – 1. Every write to TIM_MEM_RING1 resets the current time offset for the ring to the INTERVAL value. (The current time offset for the ring is readable via TIM_MEM_DEBUG0[COUNT].)
<3:0>	RID	R/W/H	0x0	0x0	Ring ID.

Cavium Confidential For Bowen Yang ALU 6/27/2014

Timer Debug 0 Register
TIM_MEM_DEBUG0

This CSR is a memory of 16 entries. TIM_REG_READ_IDX must be written before any CSR read operations to this address can be performed. Provides internal per-ring state intended for debug use only - tim.ctl[47:0]. See Table 16–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47>	ENA	RO/H	—	0	Ring timer enable.
<46>	—	RAZ	—	—	Reserved
<45:24>	COUNT	RO/H	—	0x0	Time offset for the ring. Set to INTERVAL and counts down by 1 every 1024 cycles when ENA is 1. The hardware forces a bucket traversal (and resets COUNT to INTERVAL) whenever the decrement would cause COUNT to go negative. COUNT is unpredictable whenever ENA is 0. COUNT is reset to INTERVAL whenever TIM_MEM_RING1 is written for the ring.
<23:22>	—	RAZ	—	—	Reserved
<21:0>	INTERVAL	RO/H	—	0x0	Timer interval. Specifies the interval – 1.

Timer Debug 1 Register
TIM_MEM_DEBUG1

This CSR is a memory of 16 entries. TIM_REG_READ_IDX must be written before any CSR read operations to this address can be performed. Provides internal per-ring state intended for debug use only - tim.sta[63:0]. See Table 16–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:51>	BUCKET	RO/H	—	0x0	Current bucket number, bits [12:0]. Reset to 0 whenever TIM_MEM_RING0 is written for the ring. Incremented (modulo BSIZE) once per bucket traversal. See TIM_MEM_DEBUG2[BUCKET].
<50:20>	BASE	RO/H	—	0x0	Pointer[35:5] to bucket [0].
<19:0>	BSIZE	RO/H	—	0x0	Number of buckets – 1.

Timer Debug 2 Register
TIM_MEM_DEBUG2

This CSR is a memory of 16 entries. TIM_REG_READ_IDX must be written before any CSR read operations to this address can be performed. Provides internal per-ring state intended for debug use only - tim.sta[95:64]. See Table 16–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:24>	—	RAZ	—	—	Reserved
<23:21>	CPOOL	RO/H	—	0x0	Free list used to free chunks
<20:8>	CSIZE	RO/H	—	0x0	Number of words per chunk
<7>	—	RO/H	—	0	Reserved
<6:0>	BUCKET	RO/H	—	0x0	Current bucket number, bits [19:13]. See TIM_MEM_DEBUG1[BUCKET].

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 17

Random-Number Generator (RNG), Random-Number Memory (RNM)

This chapter contains the following subjects:

- Overview
- RNG/RNM Operations
- RNM Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The CN71XX random-number generator/random-number memory (RNG/RNM) unit performs the following functions:

- generates random bits
- buffers random bits
- returns random bits to cores.

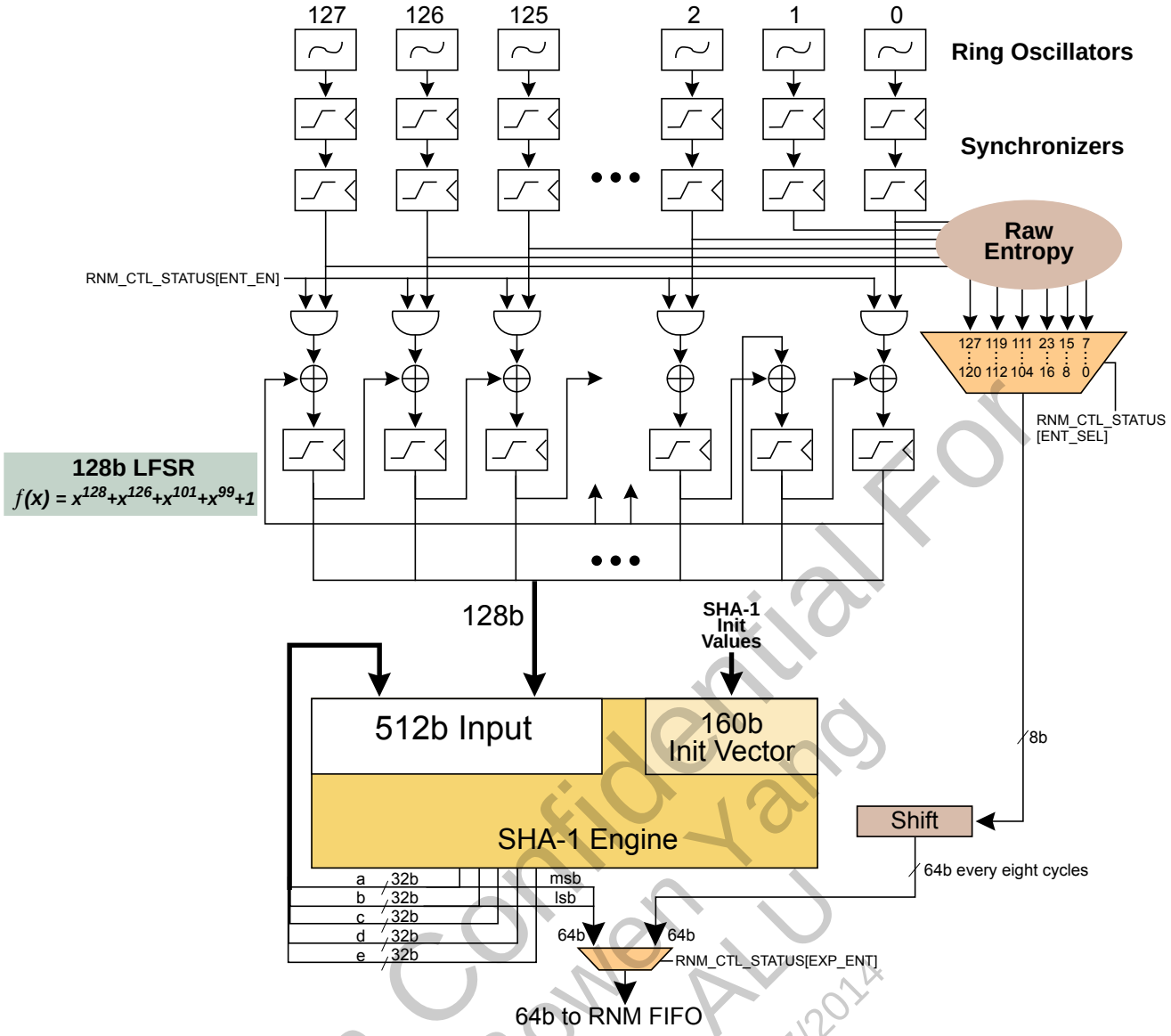
Figure 17–1 shows a block diagram of the random-number generator. The entropy is provided by the jitter of 125 of 128 free-running oscillators XORed into a 128-bit LFSR. The LFSR accumulates the entropy over 81 cycles, after which it is fed into a SHA-1 engine. The SHA-1 engine uses the following 16 32-bit words in order as input:

1. LFSR[31:0]
2. LFSR[63:32]
3. LFSR[95:64]
4. LFSR[127:96]
   (Note that the four LFSR words are all sampled on one cycle.)
5. digest a from the previous SHA-1 iteration
6. digest b from the previous SHA-1 iteration
7. digest c from the previous SHA-1 iteration
8. digest d from the previous SHA-1 iteration
9. digest e from the previous SHA-1 iteration
10. 0x80000000
11. 0x00000000
12. 0x00000000
13. 0x00000000
14. 0x00000000
15. 0x00000000
16. 0x00000120

The SHA-1 engine runs once every 81 cycles.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 17–1 RNG Diagram



The SHA-1 engine uses the 512-bit input with its 160-bit initialization vector to run its algorithm. After completing the SHA-1 operation, the hardware sends the first 64 bits of the digest to the RNM FIFO. The hardware produces new 64-bit random number every 81 cycles.

The RNM FIFO can also be sourced from the raw entropy of the 128 oscillators. The entropy is divided into 16 groups of eight bits (0–7, 8–15, 16–23, ..., 112–119, 120–127) that are fed into a multiplexer, whose output (selected by RNM_CTL_STATUS[ENT_SEL]) is fed into a shift register, which produces the 64-bit input to the RNM FIFO every eight cycles.

The input to the RNM FIFO is determined by RNM_CTL_STATUS[EXP_ENT].

Cavium Confidential For Bowen Yang ALU 10/27/2014

The RNG/RNM unit feeds the random numbers from the unit depicted in Figure 17–1 into a 512-byte FIFO eight bytes at a time. Load operations from the cores extract random bytes from the FIFO to drain it. Whenever the FIFO drains this way, RNG/RNM refills using the random numbers constantly generated by the hardware.

NOTE: Any load operation loads eight bytes of random numbers regardless of the type of load issued.

For chip-test purposes, the entropy source can be disabled from being fed into the LFSR mixing function, to ensure predictability of the random-number generator output. The entropy source is normally enabled by setting the RNM_CTL_STATUS[ENT_EN] bit. On power-up, this bit is clear, which ensures the entropy source is disabled.

When RNM_CTL_STATUS[RNG_EN] = 1, the random-number generator is enabled.

When RNM_CTL_STATUS[RNG_EN] = 0,

- no random numbers can be delivered to any cores
- the FIFO is cleared
- the LFSR mixing function is forced to its reset value (00…1).

If RNM_CTL_STATUS[ENT_EN] = 0, a predictable 512 bytes of numbers fill the FIFO after RNM_CTL_STATUS[RNG_EN] transitions from 0 to 1. Consequently, core operations can receive a predictable sequence of random numbers after enabling RNG/RNM when the entropy source is disabled.

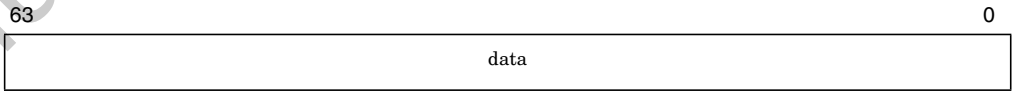
17.1 RNG/RNM Operations

17.1.1 RNG/RNM Load Operation

Load Address Field

Load Result Field

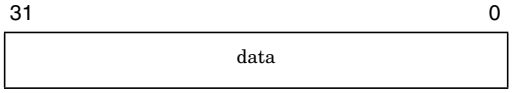
64-bit operation result



- **data** - eight random bytes.

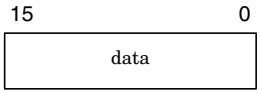
Cavium Confidential For Bowen Yang ALU 02/27/2014

32-bit operation result



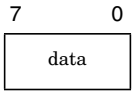
- **data** - four random bytes.

16-bit operation result



- **data** - two random bytes.

8-bit operation result



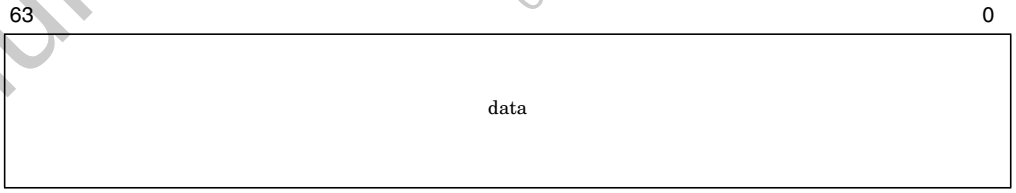
- **data** - random byte.

17.1.2 IOBDMA Operations

IOBDMA Address Field

- **scraddr** - Defined in “cnMIPS III Cores” on page 61.
- **len** - Can be any value (1 to 255 8-byte words). Defined in “cnMIPS III Cores” on page 61.

IOBDMA Result Field



- **data** - [8 × len] random bytes.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

17.2 RNM Registers

The RNM registers are shown in Table 17–1.

Table 17–1 RNM Registers

Register	Address	CSR Type[1]	Detailed Description
RNM_CTL_STATUS	0x0001180040000000	RSL	See page 826
RNM_BIST_STATUS	0x0001180040000008	RSL	See page 826
RNM_EER_KEY	0x0001180040000010	RSL	See page 827
RNM_EER_DBG	0x0001180040000018	RSL	See page 827
RNM_SERIAL_NUM	0x0001180040000020	RSL	See page 827

1. RSL-type registers are accessed indirectly across the IOI.

RNM Control and Status Register
RNM_CTL_STATUS

This register is the RNM interrupt-enable register. See Table 17–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved
<11>	DIS_MAK	R/W1S/H			Disable use of Master AES KEY
<10>	EER_LCK	RO/H	0	0	Encryption enable register locked
<9>	EER_VAL	RO/H	0	0	Dormant encryption key match
<8:5>	ENT_SEL	R/W	0x0	0x0	Select entropy input to RNM FIFO. 0x0 = 0–7 0x1 = 8–15 0x2 = 16–23 0x3 = 24–31 0x4 = 32–39 0x5 = 40–47 0x6 = 48–55 0x7 = 56–63 0x8 = 64–71 0x9 = 72–79 0xA = 80–87 0xB = 88–95 0xC = 96–103 0xD = 104–111 0xE = 112–119 0xF = 120–127
<4>	EXP_ENT	R/W	0	0	Exported entropy enable for random-number generator.
<3>	RNG_RST	R/W	0	0	Reset RNG as core reset.
<2>	RNM_RST	R/W	0	0	Reset the RNM as core reset except for register logic.
<1>	RNG_EN	R/W	0	0	Enables the output of the RNG.
<0>	ENT_EN	R/W	0	0	Entropy enable for random number generator.

RNM BIST Status Register
RNM_BIST_STATUS

This register is the RNM memory BIST status register, indicating status of built-in self-tests. See Table 17–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	RRC	RO	0	0	Status of the RRC memory block BIST: 1 = failed BIST, 0 = passed BIST
<0>	MEM	RO	0	0	Status of MEM block BIST: 1 = failed BIST, 0 = passed BIST

Cavium Confidential For Bowen Yang @2/7/2014

RNM Encryption Enable Register
RNM_EER_KEY

This register is the encryption enable register. See Table 17–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	KEY	WO	0x0	0x0	Dormant encryption key. If dormant crypto is fuse-enabled, crypto can be enabled by writing this register with the correct key.

RNM Encryption Enable Debug Register
RNM_EER_DBG

This register is the encryption enable debug register. See Table 17–1 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	DAT	RO/H	—	—	Dormant encryption debug info.

RNM Fuse Serial Number Register
RNM_SERIAL_NUM

See Table 17–1 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	DAT	RO/H	—	—	Dormant encryption serial number

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 18

RAID Unit (RAD)

This chapter contains the following subjects:

- Overview
- Pipe Calculations
- Instruction Queue
- Instruction Format
- Operations
- Performance
- Ordering
- RAD Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The CN71XX RAID unit (RAD) implements a number of XOR and Galois Field (GF) functions that are used in disk-storage applications. The unit accelerates single-parity RAID 3 / RAID 4 / RAID 5, double or diagonal-parity, and RAID 6 parity+GF.

Instructions direct RAD to create P and Q parity from up to 32 input data blocks. The *n*th byte in the P result is a bit-wise exclusive-OR of the *n*th byte of each input block that participates in the P calculation.

The Q result is similar, except that the hardware multiplies (in GF(2^8)) each participating input byte by a corresponding value before the exclusive-OR. The Q result is called a Reed-Solomon code. If desired, The Q result can be simplified to a simple exclusive-OR for double-parity applications.

The RAD instructions are very flexible in generating the P and Q results. Some example functions that can be performed with a single RAD instruction:

- N-way RAID 5 ($1 \le N \le 32$) parity calculation. Supply N input buffer pointers to the instruction. The P result from the instruction is the required result.
- N-way RAID 6 ($1 \le N \le 32$) parity calculation. Supply N input buffer pointers to the instruction. The P and Q GF results from the instruction are the two required results.
- N-way double parity ($1 \le N \le 16$) calculation. Supply 2×N input buffer pointers to the instruction, N for the P result and N for the simplified Q result. (It is also possible for P and Q to share an input buffer pointer in this case.) The P and Q results from the instruction are two parity results.
- N-way RAID6 ($1 \le N \le 32$) recovery of two (not P nor Q) data disk failures. Supply N input buffer pointers to the instruction, one for the good stored P, one for the good stored Q, and N–2 for the remaining good data blocks, with appropriate Q multipliers for each input buffer. The Q result from the instruction is the first data block that was lost. The P result can be the second data block that was lost. (The P result includes the Q result via CWORD[P_XOR] in this case.)
- N-way RAID6 ($1 \le N \le 30$) validation of RAID6 stored results. (The P and Q compare results can be used in this case via CWORD[P_CMP, Q_CMP].)

This list only covers a few examples of the possible functions that can be implemented with a single RAD instruction. There are many more possibilities.

More complicated or larger functions can be generated via combinations of multiple RAD instructions. When a multi-instruction sequence is required, the entire sequence may be submitted to the RAID unit at once, and the unit will execute them in sequence.

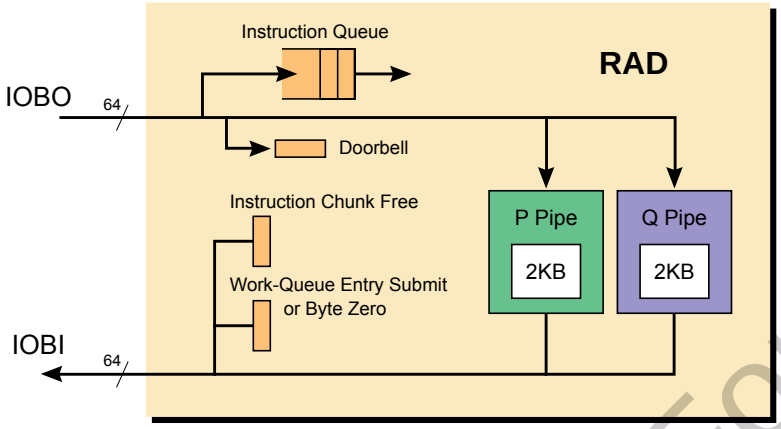
RAD implements a chunked instruction input queue. The core software writes a RAD queue doorbell after it adds instructions to the queue. The RAD hardware reads the instructions from the L2/DRAM chunks and frees the chunks once they are exhausted. After completing an instruction, RAD can either submit a work queue entry or write an L2/DRAM byte to zero.

Figure 18–1 indicates the RAD internal architecture in more detail. Its major components include the P and Q pipe. RAD implements instructions with buffers as large as ~16MB (16MB – 8 bytes), but breaks large instructions into 2 KB chunks as it executes them. It fetches the needed data from the input buffers, which arrives from L2/DRAM via IOBO, and writes the P and Q results, as needed, via IOBI. IOBO

Cavium Confidential For Bowen Yang 02/21/2014

also provides instructions and doorbell write operations. RAD issues instruction chunk frees, submits work queue entries, and performs L2/DRAM zero byte write operations via IOBI.

Figure 18–1 RAD Block Diagram

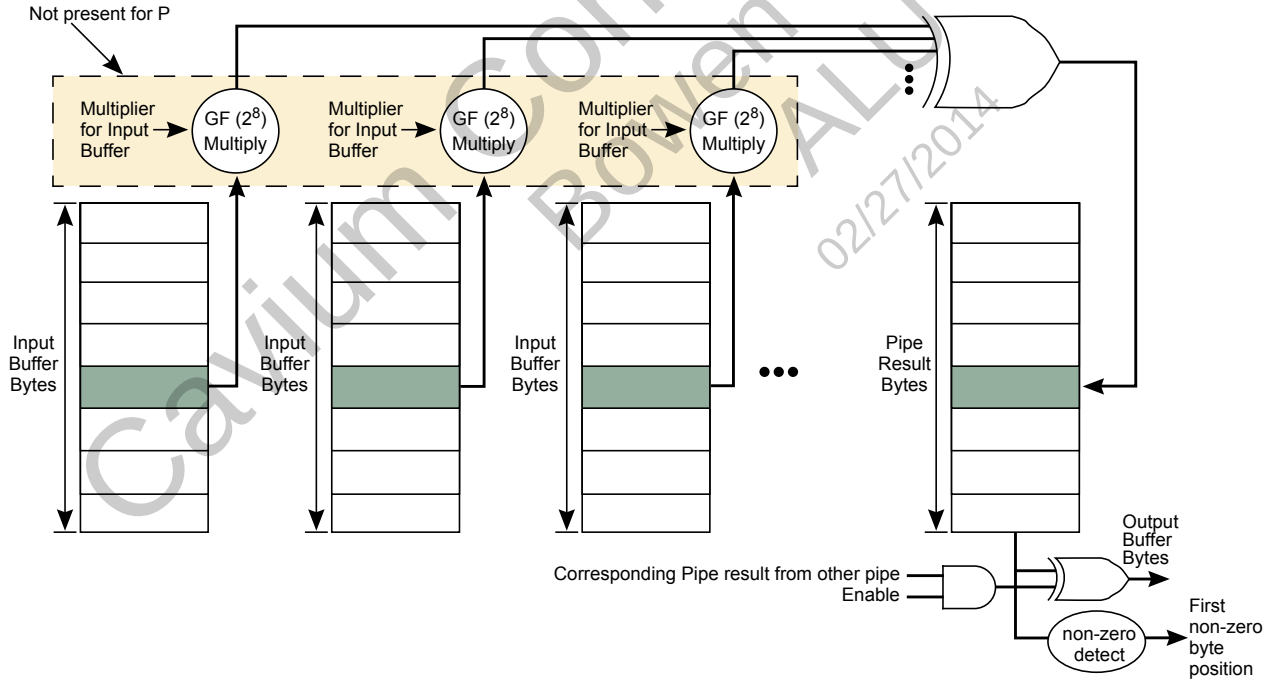


Section 18.1 describes the RAD pipe calculations. Section 18.2 describes the RAD instruction input queue and its software interface. Section 18.3 describes the detailed format of the RAD instructions. Section 18.4 describes the details of the doorbell needed for software to signal that RAD instructions are available. Section 18.5 describes performance considerations. Section 18.6 describes ordering considerations. Section 18.7 describes the RAD registers.

18.1 Pipe Calculations

Figure 18–2 is a general description of the calculations required by one RAD pipe.

Figure 18–2 RAD Calculations For a Pipe



Cavium Confidential For Bowen Yang ALI 02/27/2014

Each RAD instruction indicates streams of input bytes for a number of input buffers. The number of bytes is identical for each input buffer. The RAD calculation produces a result of the same length as each input buffer. Each RAD instruction can have up to 32 input buffers.

RAD calculates the result separately for the corresponding bytes. Result byte i depends only on the input bytes at position i. The RAD Q pipe multiplies (in GF(2^8)) each input byte by the multiplier corresponding to the input buffer. Each input buffer can have a unique multiplier. The multiplication result is an 8-bit value. RAD generates the result byte by bit-wise exclusive-OR'ing the multiply results for each input buffer.

The following pseudo-code indicates the exact GF(2^8) multiplication operation that is performed for each input byte:

```
uint8_t q_pipe_byte_result(uint8_t byte) {
   uint8_t sum = 0;
   uint8_t qmult = IWORD[QMULT];
   uint8_t poly = RAD_REG_POLYNOMIAL[COEFFS];
   while(qmult) {
      if(qmult & 1) sum = sum XOR byte;
      qmult = qmult >> 1;
      byte = ((byte << 1) & 0xFF) XOR ((byte >> 7) ? poly : 0);
   }
   return(sum);
}
```

The multiplier for each input buffer is IWORD[QMULT], which is described below.

RAD_REG_POLYNOMIAL[COEFFS] selects the polynomial used during GF(2^8) multiplication. The RAD_REG_POLYNOMIAL[COEFFS] reset value selects the standard polynomial:

$$x^8 + x^4 + x^3 + x^2 + 1$$

Figure 18–2 is most descriptive for the Q pipe. There are two pipes (P and Q), each of which can be optionally enabled with each RAD instruction. When enabled, both pipes perform their calculations for the same instruction in parallel, each producing a result of the same size. A figure specific to the P pipe would be identical to Figure 18–2, except with the GF(2^8) multiplication removed. If the Q pipe QMULT were 1 for all input buffers, the P and Q pipe result bytes would be identical. Note, however, that the P and Q results from a single RAD instruction can still differ when QMULT=1, as every RAD instruction can provide any combination of its (up to 32) input buffer streams to each pipe.

Note also that, though the P pipe and Q pipe results are generated largely independently, the final P and Q results can be bit-wise exclusive-OR'ed together for each byte, if desired. This is depicted in the lower-right of Figure 18–2. Refer to the CWORD[P_XOR,Q_XOR] descriptions in Section 18.3 below.

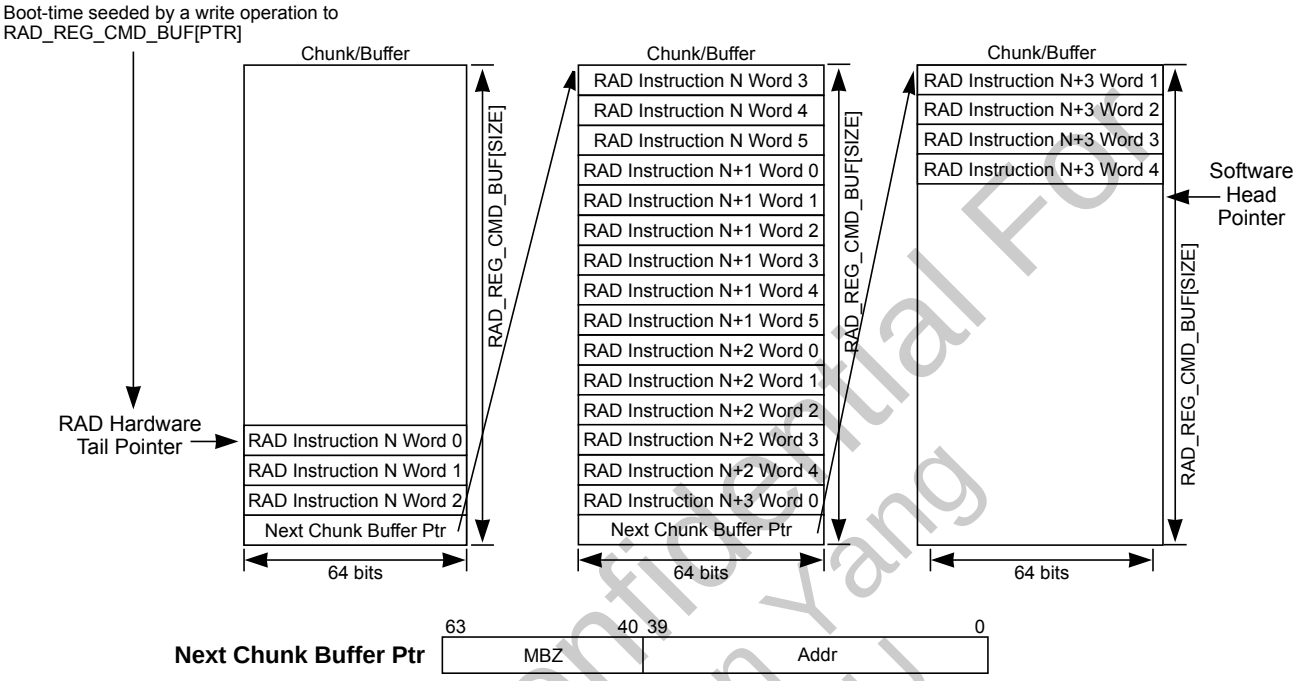
Note also that CN71XX can search for the first non-zero byte in the result byte stream. Refer to CWORD[P_CMP,Q_CMP] descriptions in Section 18.3 below. This is useful for validation of stored data. Previously-generated P/Q results can be included as the input to a later RAD instruction so that the result bytes should normally be zero. When a pipe validates this way, the result is the position of the first non-zero byte, not the actual result bytes.

Cavium Confidential For Bowen Yang 02/21/2017

18.2 Instruction Queue

Figure 18–3 depicts the structure of the RAD instruction queue in L2/DRAM. Each instruction queue is a linked-list of chunks, or buffers. Software allocates these chunks and the RAD hardware frees them. The RAD hardware reads words from the instruction queue, starting at the tail, and traverses the next chunk buffer pointer to the next chunk when it reaches the last instruction of a chunk. When the RAD hardware jumps chunks like this, it frees the earlier chunk/buffer to the FPA hardware-managed pool selected by RAD_REG_CMD_BUF[POOL], and with a don't-write-back (DWB) count of RAD_REG_CMD_BUF[DWB].

Figure 18–3 Structure of RAD Instruction Queue in L2/DRAM



RAD_REG_CMD_BUF[SIZE] indicates the size of a chunk in words. Figure 18–3 shows an example where the chunk size is 16 words. RAD instructions are variable in size—from 32 to 288 bytes. The last 64-bit word in a chunk is a next chunk buffer pointer. Each RAD instruction has the information the RAD hardware needs to start the coprocessor and read/write all the necessary data from main memory. This example shows four instructions in the instruction queue. The instructions are numbered from zero to three, with smaller numbers for the older RAD instructions, which will be started on the RAD coprocessor first.

Note that an individual RAD instruction can straddle a chunk boundary. The minimum RAD_REG_CMD_BUF[SIZE] value is 16 words.

The RAD hardware maintains the tail pointer for the queue and the core software maintains the head pointer. To insert a packet into a queue, software must first write the RAD instruction into the queue, allocating chunks if necessary, and then perform a RAD doorbell operation with the number of instruction (64-bit) words added to the queue. Software must guarantee that L2/DRAM contains the number of RAD instructions indicated in the doorbell write before the doorbell write reaches the RAD coprocessor.

Cavium Confidential For Bowen Yang 8/27/2014

The distance between the head pointer and the tail pointer is both the size of the RAD instruction queue and the outstanding doorbell count. The size of the RAD instruction queue is limited only by the available memory and the 20-bit outstanding doorbell counter for the RAD instruction queue. Core software must guarantee that the queue is smaller than 2^{20} words.

Note that the RAD hardware may read the next chunk buffer pointer as soon as the doorbell count indicates that the second-to-last word in a chunk contains a valid RAD instruction word. This implies that software must allocate the next chunk buffer and set the next chunk buffer pointer in the prior chunk to point to it, as soon as it writes the last RAD instruction into the chunk.

Doorbell write operations are RAD operations (see Section 18.4), have 64-bit word granularity, and software can “ring the doorbell” with any number of RAD instructions.

Software can issue a doorbell write for each individual RAD instruction, or can accumulate multiple RAD instructions into a single doorbell write. The only requirements are that the number of valid RAD instruction words in the queue must be at least as large as the doorbell count, and the next chunk buffer pointers interspersed among the RAD instructions must also be set up properly. Doorbells must not be given for next chunk buffer pointer words.

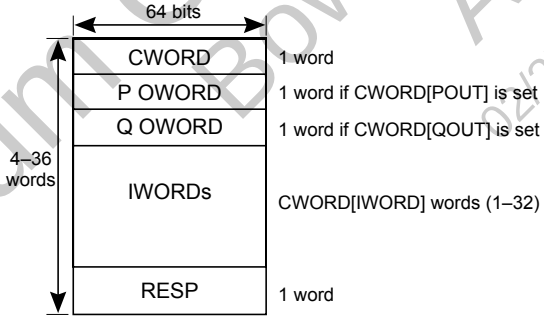
At boot time, software must configure the RAD instruction queue with the original RAD_REG_CMD_BUF[PTR].

Figure 18–3 also shows the next chunk buffer pointer format. The primary component is the Addr field that selects a legal L2/DRAM byte location. Though Addr is a byte address, it must be naturally aligned on a 128 byte cache block boundary, so its least-significant 7 bits must be zero.

18.3 Instruction Format

Figure 18–4 shows the RAD instruction format. Each instruction is variable-length from 4 to 36 64-bit words.

Figure 18–4 RAD Instruction Format



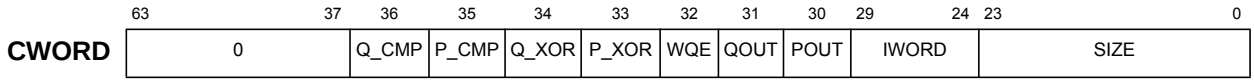
Each RAD instruction starts with a CWORD, is followed by the OWORDs, the IWORDs, and finally the RESP. The following subsections explain each component of the RAD instruction.

Cavium Confidential For Bowen Yang ALU 02/27/2014

18.3.1 CWORD Format

The CWORD contains control information for the entire RAD instruction, as shown in Figure 18–5.

Figure 18–5 RAD CWORD Instruction Format



CWORD[Q_CMP] Indicates whether the Q pipe is in normal mode (CWORD[Q_CMP] = 0) or in non-zero byte detect mode (CWORD[Q_CMP] = 1).

In non-zero byte detect mode, the Q OWORD[PTR] result is the non-zero detect result, which indicates the position of the first non-zero byte in the pipe result bytes. Refer to Section 18.1 and the OWORD[PTR] description below.

CWORD[Q_CMP] must not be set when CWORD[QOUT] = 0, and must not be set when CWORD[Q_XOR] is set.

CWORD[P_CMP] Indicates whether the P pipe is in normal mode (CWORD[P_CMP] = 0) or in non-zero byte detect mode (CWORD[P_CMP] = 1).

In non-zero byte detect mode, the P OWORD[PTR] result is the non-zero detect result, which indicates the position of the first non-zero byte in the pipe result bytes. Refer to Section 18.1 and the OWORD[PTR] description below.

CWORD[P_CMP] must not be set when CWORD[POUT] = 0, and must not be set when CWORD[P_XOR] = 1.

CWORD[Q_XOR] Indicates whether the Q output buffer bytes are the normal Q pipe result or the normal Q pipe result exclusive-ORed with the P pipe result.

When CWORD[Q_XOR] = 0 (and CWORD[Q_CMP]=0), the Q output buffer bytes are the normal Q pipe result, which does not include the P pipe result in any way.

When CWORD[Q_XOR] = 1, the Q output buffer bytes are the normal Q pipe result exclusive-OR'ed with the P pipe result, as if the P pipe result were another Q IWORD for the Q pipe with QMULT = 1.

CWORD[Q_XOR] must not be set unless both CWORD[POUT,QOUT] = 1, and must not be set when CWORD[Q_CMP] = 1.

CWORD[P_XOR] Indicates whether the P output buffer bytes are the normal P pipe result or the normal P pipe result exclusive-OR'ed with the Q pipe result.

When CWORD[P_XOR] = 0 (and CWORD[P_CMP] = 0), the P output buffer bytes are the normal P pipe result, which does not include the Q pipe result in any way.

When CWORD[P_XOR] = 1, the P output buffer bytes are the normal P pipe result exclusive-ORed with the Q pipe result, as if the Q pipe result were another P IWORD for the P pipe.

CWORD[P_XOR] must not be set unless both CWORD[POUT,QOUT] = 1, and must not be set when CWORD[P_CMP] = 1.

CWORD[WQE] Indicates whether RAD submits a work-queue entry or writes an L2/DRAM byte to zero after completing the instruction.

Cavium Confidential For Bowen Yang ALU 02/27/2014

When CWORD[WQE] is set and RESP[PTR] ≠ 0, RAD adds the work queue entry indicated by RESP[PTR] to the selected POW input queue after completing the instruction. Refer to Chapter 7.

When CWORD[WQE] is clear and RESP[PTR] ≠ 0, RAD writes the L2/DRAM byte indicated by RESP[PTR] to zero after completing the instruction.

CWORD[QOUT] Indicates whether the Q pipe is used by this instruction.

Section 18.1 above describes the details of the Q pipe computation.

If CWORD[QOUT] is set, IWORD[QEN] must be set for at least one IWORD.

At least one of CWORD[QOUT,POUT] must be set.

CWORD[POUT] Indicates whether the P pipe is used by this instruction.

Section 18.1 above describes the details of the P pipe computation.

If CWORD[POUT] is set, IWORD[PEN] must be set for at least one IWORD.

At least one of CWORD[QOUT,POUT] must be set.

CWORD[IWORD] Indicates the number of input buffers used.

1 ≤ CWORD[IWORD] ≤ 32.

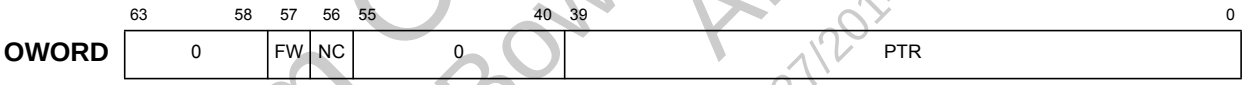
CWORD[SIZE] Indicates the size in bytes of all input buffers. When CWORD[Q_CMP,P_CMP]=0, also indicates the size of the Q/P output buffers.

CWORD[SIZE] must be a multiple of 8 bytes (i.e. <2:0> must be zero).

18.3.2 OWORD Format

The OWORDs point to where to put the P/Q result of the instruction. Section 18.1 above describe the details of the P/Q results, as shown in Figure 18–6.

Figure 18–6 RAD OWORD Instruction Format



OWORD[FW] When set, indicates that RAD can modify any byte in any (128B) cache line touched by L2/DRAM addresses OWORD[PTR] through OWORD[PTR]+CWORD[SIZE]–1.

Setting OWORD[FW] can improve hardware performance, as some DRAM loads can be avoided on L2 cache misses. The Q OWORD[FW] must not be set when CWORD[Q_CMP] is set, and the P OWORD[FW] must not be set when CWORD[P_CMP] is set.

OWORD[NC] When set, indicates that RAD should not allocate L2 cache space for the P/Q data on L2 cache misses.

OWORD[NC] should typically be clear, though setting OWORD[NC] can improve performance in some circumstances, as the L2 cache will not be polluted by P/Q data.

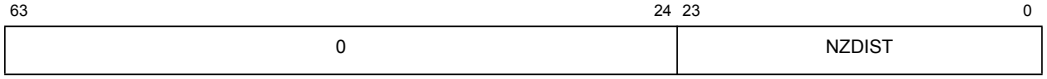
The Q OWORD[NC] must not be set when CWORD[Q_CMP] is set, and the P OWORD[NC] must not be set when CWORD[P_CMP] is set.

Cavium Confidential For Bowen Yang ALU 02/27/2014

OWORD[PTR] When CWORD[P_CMP,Q_CMP]=0, OWORD[PTR] indicates the starting address of the L2/DRAM buffer that will receive the P/Q data. In the non-compare mode, the output buffer receives all of the output buffer bytes.

When CWORD[P_CMP,Q_CMP]=1, the corresponding P/Q pipe is in compare mode, and the only output of the pipe is the non-zero detect result. In this case, OWORD[PTR] indicates the 8-byte location of the non-zero detect result. Figure 18–7 shows the format of the non-zero detect result.

Figure 18–7 Non-Zero Detect Result



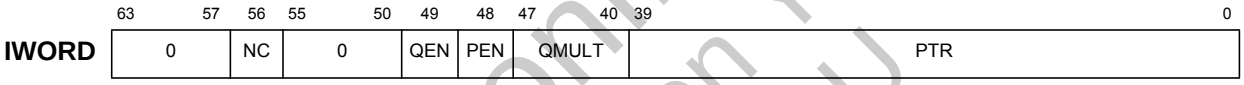
NZDIST= Location of first pipe result byte that is non-zero. If all pipe result bytes are zero for this instruction, NZDIST = CWORD[SIZE]. NZDIST always uses big-endian order. This means that if the most-significant byte in a 64-bit result word is the first non-zero one, NZDIST<2:0>=0, and if the second most-significant byte in a 64-bit result word is the first non-zero one, NZDIST<2:0>=1, etc.

OWORD[PTR] must be naturally-aligned on an 8-byte boundary (i.e. <2:0> must be zero).

18.3.3 IWORD Format

The IWORDs, shown in Figure 18–8, point to where to read the input buffer data from. Each IWORD indicates whether the buffer should be accumulated by the P pipe, the Q pipe, or both. The IWORD also supplies the Q pipe multiplier for the buffer (QMULT).

Figure 18–8 RAD IWORD Instruction Format



IWORD[NC] When set, indicates that RAD should not allocate L2 cache space for this input buffer data on L2 cache misses.

Setting IWORD[NC] may improve performance in some circumstances, as the L2 cache may not be polluted with input buffer data.

IWORD[QEN] Indicates that this input buffer data should participate in the Q pipe result.

The Q pipe hardware multiplies each participating input byte by IWORD[QMULT] before accumulating them by exclusive-OR'ing, as Section 18.1 above indicates.

IWORD[QEN] must not be set when CWORD[QOUT] is not set.

If CWORD[QOUT] is set, IWORD[QEN] must be set for at least one IWORD.

IWORD[PEN] Indicates that this input buffer data should participate in the P pipe result.

The P pipe hardware accumulates each participating input byte by bit-wise exclusive-ORing it, as Section 18.1 indicates.

IWORD[PEN] must not be set when CWORD[POUT] is not set.

If CWORD[POUT] is set, IWORD[PEN] must be set for at least one IWORD.

Cavium Confidential For Bowen Yang ALU 02/27/2014

IWORD[QMULT] The Q pipe multiplier for the input buffer. Section 18.1 above describes the GF(2^8) multiplication algorithm.

IWORD[QMULT] must be zero when IWORD[QEN] is not set.

IWORD[QMULT] must not be zero when IWORD[QEN] is set.

When IWORD[QMULT] is 1, the multiplication simplifies to the identity function, and the Q pipe performs the same XOR function as the P pipe.

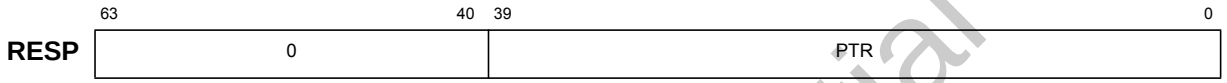
IWORD[PTR] The starting address of the input buffer in L2/DRAM.

IWORD[PTR] must be naturally aligned on an eight-byte boundary (i.e. <2:0> must be zero).

18.3.4 RESP Format

RESP, shown in Figure 18–8, contains a pointer to the work queue entry to submit or L2/DRAM byte to zero after completing the instruction.

Figure 18–9 RAD RESP Instruction Format



RESP[PTR] When CWORD[WQE] is set and RESP[PTR] ≠ 0, RAD adds the work queue entry indicated by RESP[PTR] to the selected POW input queue after completing the instruction. Refer to Chapter 7.

RESP[PTR] must be naturally-aligned on an 8B boundary (i.e. <2:0> must be zero) when CWORD[WQE] is set.

When CWORD[WQE] = 0 and RESP[PTR] ≠ 0, RAD writes the L2/DRAM byte indicated by RESP[PTR] to 0x0 after completing the instruction. RAD_REG_CTL[STORE_LE] indicates the endianness of RESP[PTR].

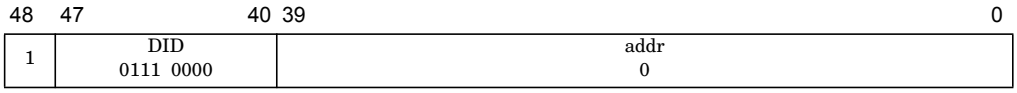
Cavium Confidential For Bowen Yang ALI 02/27/2014

18.4 Operations

RAD operations are used to ring the RAD instruction doorbell only. Only 64-bit store operations to RAD are allowed. RAD Load, IOBDMA, and non 64-bit store operations are not allowed and may result in unpredictable results.

NOTE: The RAD hardware behavior is unpredictable when anything but a 64-bit Store (i.e. anything other than an SD) is used.

Physical Address to Store to RAD



- **DID** = 0x70.
- All address bits must be zero.

Store Data on a Store to RAD

- **<63:20> =** Must be 0x0.
- **dbell cnt =** Number of instruction queue 64-bit words to add to the RAD instruction doorbell count. RAD instructions are from 4 to 36 words.

18.5 Performance

RAD takes a minimum of

(CWORD[POUT] + CWORD[QOUT] + CWORD[IWORD]) × CWORD[SIZE]/8

coprocessor clock cycles to complete a given instruction when not in compare mode (i.e. when CWORD[P_CMP,Q_CMP]=0).

RAD_REG_CTL[MAX_READ] controls the number of outstanding cache line loads that RAD can generate—up to 8. Performance may be higher with larger RAD_REG_CTL[MAX_READ]. However, RAD can consume all of the available IOBO bus bandwidth when running at maximum speed.

It is possible that system performance will improve with smaller RAD_REG_CTL[MAX_READ] values in a given system.

When RAD_REG_CTL[MAX_READ] is set to a large enough value, RAD can dominate the IOBO bandwidth, as its read responses will be given statically higher priority than IOBO core-initiated requests (REQ), store commit responses (COM), and IOB DWB free completions (FPA) when IOB_OUTB_REQ_PRI_CNT[CNT_ENB]=0, IOB_OUTB_COM_PRI_CNT[CNT_ENB]=0, and IOB_OUTB_FPA_PRI_CNT[CNT_ENB]=0, respectively. To avoid IOBO starvation when RAD is used, all of IOB_OUTB_*_PRI_CNT[CNT_ENB] should be set to one, with IOB_OUTB_*_PRI_CNT[CNT_VAL] also set to reasonable values. The values 10, 50, and 100 may be reasonable CNT_VAL values for REQ, COM, and FPA, respectively.

Cavium Confidential For Bowen Yang ALU 03/7/2014

18.6 Ordering

The RAD hardware always completes the OWORD write operations for a single instruction before it does the RESP write or work submit for the same instruction. Otherwise the RAD hardware may execute all operations associated with a single instruction, including any required memory operations, in any order.

The RAD hardware may execute instructions out of order, though it generally processes them in order. For example, it may start fetching the input data for a subsequent instruction before it writes the result out for a prior instruction.

If a RAD instruction earlier in the RAD input queue produces a P/Q result that is input to a later RAD instruction, the RAD hardware guarantees that the two instructions appear to be processed in order.

18.7 RAD Registers

The RAD registers are listed in Table 18–1.

Table 18–1 RAD Registers

Register	Address	CSR Type[1]	Detailed Description
RAD_REG_CTL	0x0001180070000000	RSL	See page 841
RAD_REG_CMD_BUF	0x0001180070000008	RSL	See page 841
RAD_REG_POLYNOMIAL	0x0001180070000010	RSL	See page 841
RAD_REG_READ_IDX	0x0001180070000018	RSL	See page 842
RAD_REG_BIST_RESULT	0x0001180070000080	RSL	See page 842
RAD_REG_ERROR	0x0001180070000088	RSL	See page 842
RAD_REG_INT_MASK	0x0001180070000090	RSL	See page 842
RAD_REG_DEBUG0	0x0001180070000100	RSL	See page 843
RAD_REG_DEBUG1	0x0001180070000108	RSL	See page 843
RAD_REG_DEBUG2	0x0001180070000110	RSL	See page 843
RAD_REG_DEBUG3	0x0001180070000118	RSL	See page 843
RAD_REG_DEBUG4	0x0001180070000120	RSL	See page 843
RAD_REG_DEBUG5	0x0001180070000128	RSL	See page 844
RAD_REG_DEBUG6	0x0001180070000130	RSL	See page 844
RAD_REG_DEBUG7	0x0001180070000138	RSL	See page 844
RAD_REG_DEBUG8	0x0001180070000140	RSL	See page 845
RAD_REG_DEBUG9	0x0001180070000148	RSL	See page 845
RAD_REG_DEBUG10	0x0001180070000150	RSL	See page 845
RAD_REG_DEBUG11	0x0001180070000158	RSL	See page 845
RAD_REG_DEBUG12	0x0001180070000160	RSL	See page 846
RAD_MEM_DEBUG0	0x0001180070001000	RSL	See page 846
RAD_MEM_DEBUG1	0x0001180070001008	RSL	See page 846
RAD_MEM_DEBUG2	0x0001180070001010	RSL	See page 846

1. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang 2/2/2014

RAD Control Register
RAD_REG_CTL

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	0x0	0x0	Reserved
<6>	WC_DIS	R/W/H	0	0	Reserved.
<5:2>	MAX_READ	R/W	0x8	0x8	Maximum number of outstanding data read commands. MAX_READ is a throttle to control IOB usage. Values greater than 0x8 are illegal.
<1>	STORE_LE	R/W	0	0	Force STORE0 byte write address to little endian.
<0>	RESET	WO	0	0	Reset oneshot pulse (lasts for 4 cycles).

Command Buffer Register
RAD_REG_CMD_BUF

This register sets the command-buffer parameters. See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:58>	—	RAZ	0x0	0x0	Reserved
<57:49>	DWB	R/W	0x0	—	Number of don't-write-backs.
<48:46>	POOL	R/W	0x0	—	Specifies 1 of 8 free lists used when freeing command-buffer segments.
<45:33>	SIZE	R/W	0x0	—	Number of uint64 words per command buffer segment.
<32:0>	PTR	R/W	0x0	—	Initial command-buffer pointer bits <39:7> (128-byte aligned). Overwritten each time the command-buffer segment is exhausted.

RAD Polynomial Register
RAD_REG_POLYNOMIAL

The polynomial is $x^8 + C7*x^7 + C6*x^6 + C5*x^5 + C4*x^4 + C3*x^3 + C2*x^2 + C1*x^1 + C0$.
See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	0x0	0x0	Reserved
<7:0>	COEFFS	R/W	0x1D	0x1D	Coefficients of GF(2^8) irreducible polynomial

Cavium Confidential For Bowen Yang ALU 02/21/2014

RAD Read Index Register
RAD_REG_READ_IDX

This register is for debug purposes only. It provides the read index during a CSR read operation to any of the CSRs that are physically stored as memories. The mnemonics for these CSRs begin with the prefix RAD_MEM_.

IDX[15:0] is the read index. INC[15:0] is an increment that is added to IDX[15:0] after any CSR read. The intended use is to initially write this CSR such that IDX = 0 and INC = 1. Then, the entire contents of a CSR memory can be read with consecutive CSR read commands. See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:16>	INC	R/W	0x0	0x0	Increment to add to the current index for the next index.
<15:0>	IDX	R/W	0x0	0x0	Index to use for next memory CSR read operation.

RAD BIST Result Register
RAD_REG_BIST_RESULT

This register provides access to internal BIST results. Each bit is the BIST result of an individual memory (per bit, 0 = pass, 1 = fail). See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5>	STA	RO	—	0	BIST result of the STA memories
<4>	NCB_OUB	RO	—	0	BIST result of the NCB_OUB memories
<3:2>	NCB_INB	RO	—	0x0	BIST result of the NCB_INB memories
<1:0>	DAT	RO	—	0x0	BIST result of the DAT memories

RAD Error Register
RAD_REG_ERROR

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	DOORBELL	R/W1C/H	0	0	Error bit indicating a doorbell count has overflowed.

RAD Interrupt Mask Register
RAD_REG_INT_MASK

When a mask bit is set, the corresponding interrupt is enabled. See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	DOORBELL	R/W/H	0	0	Interrupt-enable bit mask corresponding to the error mask in RAD_REG_ERROR.

Cavium Confidential For Bowen Yang ALU 02/27/2014

RAD Debug0 Register
RAD_REG_DEBUG0

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:57>	—	RO/H	—	—	Reserved
<56:32>	LOOP	RO/H	—	—	Loop offset
<31:22>	—	RO/H	—	—	Reserved
<21:16>	IRIDX	RO/H	—	—	IWords read index
<15:14>	—	RO/H	—	—	Reserved
<13:8>	IWIDX	RO/H	—	—	IWords write index
<7>	OWORDQV	RO/H	—	—	Valid for OWORDQ
<6>	OWORDPV	RO/H	—	—	Valid for OWORDP
<5>	COMMIT	RO/H	—	—	Waiting for write commit
<4:0>	STATE	RO/H	—	—	Main state

RAD Debug1 Register
RAD_REG_DEBUG1

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	CWORD	RO/H	—	—	CWord

RAD Debug2 Register
RAD_REG_DEBUG2

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	OWORDP	RO/H	—	—	OWordP

RAD Debug3 Register
RAD_REG_DEBUG3

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	OWORDQ	RO/H	—	—	OWordQ

RAD Debug4 Register
RAD_REG_DEBUG4

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	RWORD	RO/H	—	—	RWord

Cavium Confidential For Bowen Yang ALU 02/27/2014

RAD Debug5 Register
RAD_REG_DEBUG5

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:53>	—	RO/H	—	—	Reserved
<52:50>	NIROPC7	RO/H	—	—	IOBI ropc (stage7 grant)
<49:48>	NIRQUE7	RO/H	—	—	IOBI rque (stage7 grant)
<47:43>	NIRVAL7	RO/H	—	—	IOBI rval (stage7 grant)
<42:40>	NIROPC6	RO/H	—	—	IOBI ropc (stage6 arb)
<39:38>	NIRQUE6	RO/H	—	—	IOBI rque (stage6 arb)
<37>	NIRARB6	RO/H	—	—	IOBI rarb (stage6 arb)
<36:32>	NIRVAL6	RO/H	—	—	IOBI rval (stage6 arb)
<31:28>	NIRIDX1	RO/H	—	—	IOBI ridx1
<27:24>	NIWIDX1	RO/H	—	—	IOBI widx1
<23:20>	NIRIDX0	RO/H	—	—	IOBI ridx0
<19:16>	NIWIDX0	RO/H	—	—	IOBI widx0
<15:14>	WCCREDS	RO/H	0x0	0x0	WC credits
<13:12>	FPACREDS	RO/H	0x1	0x0	FPA credits
<11:10>	—	RO/H	—	—	Reserved
<9:8>	POWCREDS	RO/H	0x2	0x0	POW credits
<7:4>	N1CREDS	RO/H	0x8	0x0	IOBI1 credits
<3:0>	N0CREDS	RO/H	0x8	0x0	IOBI0 credits

RAD Debug6 Register
RAD_REG_DEBUG6

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:56>	CNT	RO/H	—	—	CCTL count [7:0] in bytes
<55:40>	SIZE	RO/H	—	—	CCTL size in bytes
<39:0>	PTR	RO/H	—	—	CCTL pointer

RAD Debug7 Register
RAD_REG_DEBUG7

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:15>	—	RAZ	—	—	Reserved
<14:0>	CNT	RO/H	—	—	CCTL count [22:8] in bytes

Cavium Confidential For Bowen Yang ALU 02/27/2014

RAD Debug8 Register
RAD_REG_DEBUG8

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:56>	FLAGS	RO/H	—	—	ICTL flags
<55:40>	SIZE	RO/H	—	—	ICTL size in bytes
<39:0>	PTR	RO/H	—	—	ICTL pointer

RAD Debug9 Register
RAD_REG_DEBUG9

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:20>	—	RAZ	—	—	Reserved
<19>	EOD	RO/H	—	—	ICTL eod flag
<18>	INI	RO/H	—	—	ICTL init flag
<17>	Q	RO/H	—	—	ICTL q enable
<16>	P	RO/H	—	—	ICTL p enable
<15:8>	MUL	RO/H	—	—	ICTL multiplier
<7:0>	IDX	RO/H	—	—	ICTL index

RAD Debug10 Register
RAD_REG_DEBUG10

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:56>	FLAGS	RO/H	—	—	OCTL flags
<55:40>	SIZE	RO/H	—	—	OCTL size in bytes
<39:0>	PTR	RO/H	—	—	OCTL pointer

RAD Debug11 Register
RAD_REG_DEBUG11

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	—	—	Reserved
<12>	Q	RO/H	—	—	OCTL Q flag
<11>	P	RO/H	—	—	OCTL P flag
<10>	WC	RO/H	—	—	OCTL write commit flag
<9>	EOD	RO/H	—	—	OCTL EOD flag
<8>	SOD	RO/H	—	—	OCTL SOD flag
<7:0>	IDX	RO/H	—	—	OCTL index

Cavium Confidential For Bowen Yang ALU 02/27/2014

RAD Debug12 Register
RAD_REG_DEBUG12

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:15>	—	RAZ	—	—	Reserved
<14:0>	ASSERTS	RO	—	—	Various assertion checks

RAD Memory Debug0 Register
RAD_MEM_DEBUG0

This CSR is a memory of 32 entries, and thus, the RAD_REG_READ_IDX CSR must be written before any CSR read operations to this address can be performed. A read of any entry that has not been previously written is illegal and will result in unpredictable CSR read data.

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	IWORD	RO/H	—	—	IWord

RAD Memory Debug1 Register
RAD_MEM_DEBUG1

This CSR is a memory of 256 entries, and thus, the RAD_REG_READ_IDX CSR must be written before any CSR read operations to this address can be performed. A read of any entry that has not been previously written is illegal and will result in unpredictable CSR read data.

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	P_DAT	RO/H	—	—	P data

RAD Memory Debug2 Register
RAD_MEM_DEBUG2

This CSR is a memory of 256 entries, and thus, the RAD_REG_READ_IDX CSR must be written before any CSR read operations to this address can be performed. A read of any entry that has not been previously written is illegal and will result in unpredictable CSR read data.

See Table 18–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	Q_DAT	RO/H	—	—	Q data

Cavium Confidential For Bowen Yang ALU 02/27/2014

Chapter 19

BCH Unit

This chapter contains the following subjects:

- Overview
- Instruction Queue
- Instruction Format
- Operations
- Ordering
- BCH Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The CN71XX BCH unit (BCH) implements a BCH (Bose Chaudhuri Hocquenghem) cyclic error-correcting code capable of correcting many errors within a block of data. The hardware unit accelerates both the parity-generation and the error-correction functions. This unit is used mainly by NAND devices for ECC (see Chapter 30).

Instructions direct BCH to perform either a parity generation or a block correction. Instructions also indicate the block size, input L2/DRAM address, and output L2/DRAM address.

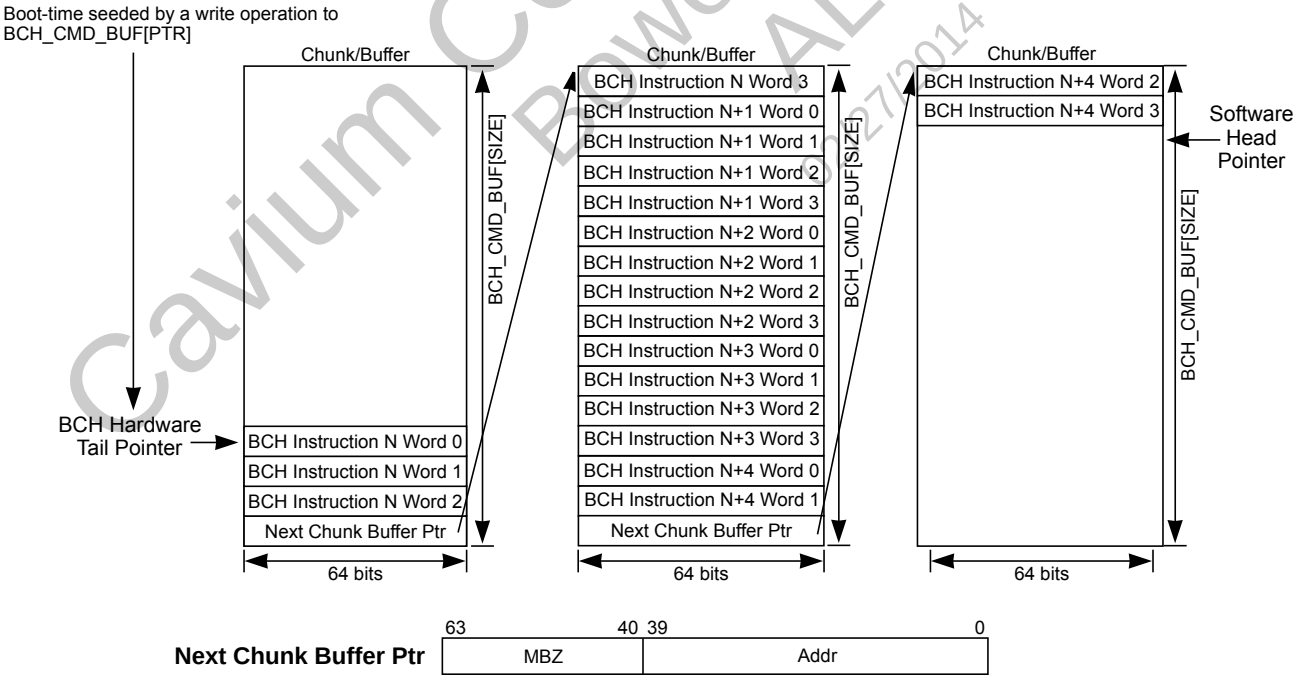
- Parity-generation operations are performed by reading the data block from the input L2/DRAM address, computing the parity bytes, and then writing the parity bytes to output L2/DRAM.
- Block-correction operations are performed by reading the data block and previously generated parity bytes from the input L2/DRAM address. The corrected data block is then written to the output L2/DRAM address.

BCH implements a chunked instruction-input queue. The core software writes a BCH queue doorbell after it adds instructions to the queue. The BCH hardware reads the instructions from the L2/DRAM chunks and frees the chunks once they are exhausted. After completing an instruction, BCH can write a two-byte operation status to L2/DRAM.

19.1 Instruction Queue

Figure 19–1 shows the structure of the BCH instruction queue in L2/DRAM. Each instruction queue is a linked-list of chunks, or buffers. Software allocates these chunks and the BCH hardware frees them. The BCH hardware reads words from the instruction queue, starting at the tail, and traverses the next-chunk buffer pointer to the next chunk when it reaches the last instruction of a chunk. When the BCH hardware jumps chunks like this, it frees the earlier chunk/buffer to the FPA hardware-managed pool selected by BCH_CMD_BUF[POOL], and with a don't-write-back (DWB) count of BCH_CMD_BUF[DWB].

Figure 19–1 Structure of BCH Instruction Queue in L2/DRAM



Cavium Confidential For Power Viewing ALU 10/27/2014

BCH_CMD_BUF[SIZE] indicates the size of a chunk in words. Figure 19–1 shows an example where the chunk size is 16 words. BCH instructions are variable in size—from 32 to 288 bytes. The last 64-bit word in a chunk is a next chunk buffer pointer. Each BCH instruction has the information the BCH hardware needs to start the coprocessor and read/write all the necessary data from main memory. This example shows five instructions in the instruction queue. The instructions are numbered from zero to four, with smaller numbers for the older BCH instructions, which will be started on the BCH coprocessor first.

Note that an individual BCH instruction can straddle a chunk boundary. The minimum BCH_CMD_BUF[SIZE] value is 16 words.

The BCH hardware maintains the tail pointer for the queue and the core software maintains the head pointer. To insert a packet into a queue, software must first write the BCH instruction into the queue, allocating chunks if necessary, and then perform a BCH doorbell operation with the number of instruction (64-bit) words added to the queue. Software must guarantee that L2/DRAM contains the number of BCH instructions indicated in the doorbell write before the doorbell write reaches the BCH coprocessor.

The distance between the head pointer and the tail pointer is both the size of the BCH instruction queue and the outstanding doorbell count. The size of the BCH instruction queue is limited only by the available memory and the 20-bit outstanding doorbell counter for the BCH instruction queue. Core software must guarantee that the queue is smaller than 2^{20} words.

Note that the BCH hardware may read the next chunk buffer pointer as soon as the doorbell count indicates that the second-to-last word in a chunk contains a valid BCH instruction word. This implies that software must allocate the next chunk buffer and set the next chunk buffer pointer in the prior chunk to point to it, as soon as it writes the last BCH instruction into the chunk.

Doorbell write operations are BCH operations (see Section 19.3), have 64-bit word granularity, and software can “ring the doorbell” with any number of BCH instructions.

Software can issue a doorbell write for each individual BCH instruction, or can accumulate multiple BCH instructions into a single doorbell write. The only requirements are that the number of valid BCH instruction words in the queue must be at least as large as the doorbell count, and the next chunk buffer pointers interspersed among the BCH instructions must also be set up properly. Doorbells must not be given for next chunk buffer pointer words.

At boot time, software must configure the BCH instruction queue with the original BCH_CMD_BUF[PTR].

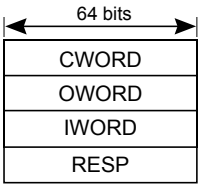
Figure 19–1 also shows the next chunk buffer pointer format. The primary component is the Addr field that selects a legal L2/DRAM byte location. Though Addr is a byte address, it must be naturally aligned on a 128 byte cache block boundary, so its least-significant 7 bits must be zero.

Cavium Confidential For Bowen Yang AU 6/8/2014

19.2 Instruction Format

Figure 19–2 shows the BCH instruction format. Each instruction has four 64-bit words.

Figure 19–2 BCH Instruction Format

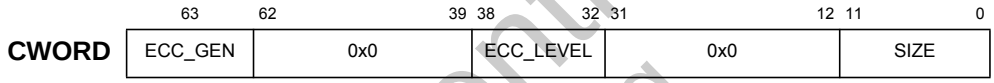


Each BCH instruction starts with a CWORD, is followed by the OWORD, the IWORD, and finally the RESP. The following subsections explain each component of the BCH instruction.

19.2.1 CWORD Format

The CWORD contains control information for the entire BCH instruction, as shown in Figure 19–3.

Figure 19–3 BCH CWORD Instruction Format



CWORD[ECC_GEN] Indicates the BCH function: 1 = perform a parity generation, 0 = perform a block correction.

CWORD[ECC_LEVEL] Indicates the maximum number of errors within a data block that can be corrected. The number of parity bytes is equal to $\lceil(15 \times \text{ECC_LEVEL})/8\rceil$.

ECC_LEVEL must belong to {4, 8, 16, 24, 32, 40, 48, 56, 60, 64}.

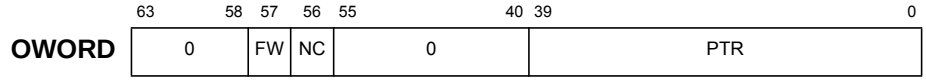
CWORD[SIZE] Indicates the size in bytes of the data block. SIZE must be a multiple of two bytes (i.e. SIZE<0> must be 0).

Cavium Confidential For Bowen Yang ALD 02/27/2014

19.2.2 OWORD Format

The OWORD, which is shown in Figure 19–4, points to where to put the parity/correction result of the instruction. When CWORD[ECC_GEN] = 1, the parity bytes are written to the OWORD. When CWORD[ECC_GEN] = 0, the corrected-data block is written to the OWORD.

Figure 19–4 BCH OWORD Instruction Format



OWORD[FW] When set, indicates that BCH can modify any byte in any (128-byte) cache line touched by L2/DRAM addresses OWORD[PTR] through OWORD[PTR] + SIZE – 1, where SIZE is CWORD[SIZE] when CWORD[ECC_GEN] = 0, else SIZE is the number of parity bytes selected by CWORD[ECC_LEVEL].

Setting FW can improve hardware performance, as some DRAM load operations can be avoided on L2 cache misses.

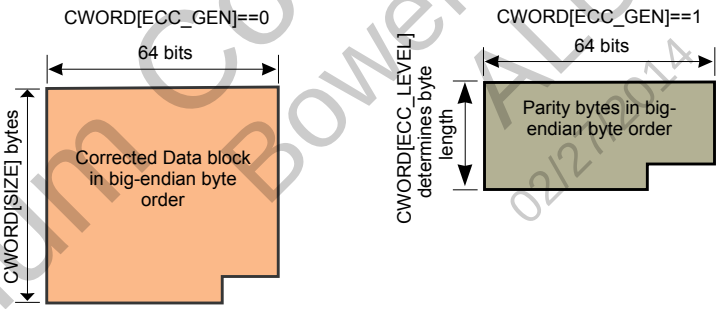
OWORD[NC] When set, indicates that BCH should not allocate L2 cache space for the parity/correction data on L2 cache misses.

OWORD[PTR] Indicates the starting address of the L2/DRAM buffer that will receive the parity/correction data.

PTR must be naturally-aligned on an eight-byte boundary (i.e. <2:0> must be 0s).

Figure 19–5 shows the format of the output parity/correction data addressed by OWORD[PTR].

Figure 19–5 Output Parity/Correction Data Addressed by OWORD[PTR]



All OWORD[PTR] output data from the BCH instruction is big-endian byte-order. (When CWORD[SIZE] is a multiple of 8 bytes, as may be common, the byte order of the corrected data block may not be important.)

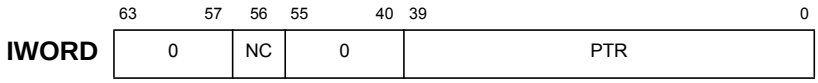
19.2.3 IWORD Format

The IWORD, shown in Figure 19–6, point to where go to read the input data.

- When CWORD[ECC_GEN] = 1, the input data is only the data blocks.
- When CWORD[ECC_GEN] = 0, the input data includes the data block and the parity bytes.

Cavium Confidential For Bowen Yang ALD 02/27/2014

Figure 19–6 BCH IWORD Instruction Format



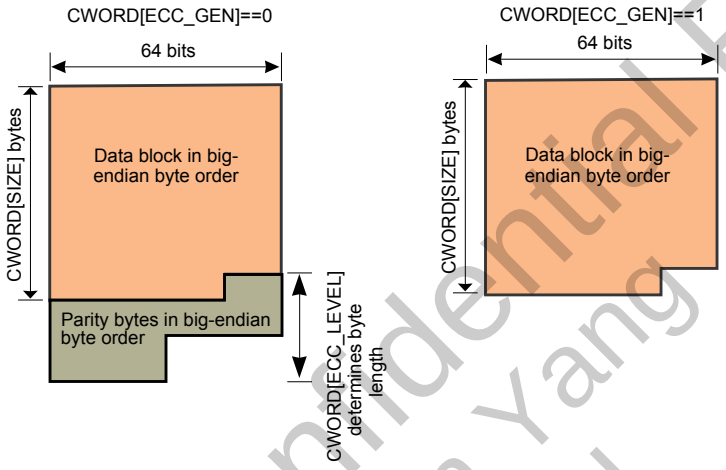
IWORD[NC] When set, indicates that BCH should not allocate L2 cache space for the input data on L2 cache misses.

IWORD[PTR] Indicates the starting address of input data in L2/DRAM.

PTR must be naturally aligned on an eight-byte boundary (i.e. <2:0> must be 0s).

Figure 19-7 shows the format of the input parity/correction data addressed by IWORD[PTR]

Figure 19–7 Input Parity/Correction Data Addressed by IWORD[PTR]



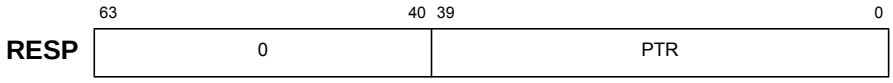
All IWORD[PTR] input data to the BCH instruction is big-endian byte order. (When CWORD[SIZE] is a multiple of 8 bytes, as may be common, the byte order of the data block may not be important.)

Cavium Confidential For Bowen Yang ALU 02/27/2014

19.2.4 RESP Format

RESP, shown in Figure 19–8, contains a pointer to the L2/DRAM location where the two-byte operation status is written after completing the instruction.

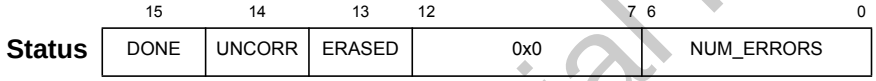
Figure 19–8 BCH RESP Instruction Format



RESP[PTR] When PTR ≠ 0 after completing the instruction, BCH writes the two-byte operation status (shown in Figure 19–8) to the L2/DRAM address specified by PTR.

BCH_CTL[STORE_LE] indicates the endianness of PTR. PTR must be naturally-aligned to a two-byte boundary (i.e. <0> must be 0).

Figure 19–9 Operation-Status Format

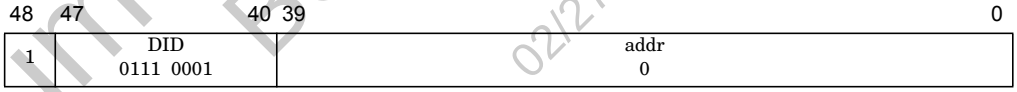


19.3 Operations

BCH operations are used to ring the BCH instruction doorbell only. Only 64-bit store operations to BCH are allowed. BCH Load, IOBDMA, and non-64-bit store operations are not allowed and may result in unpredictable results.

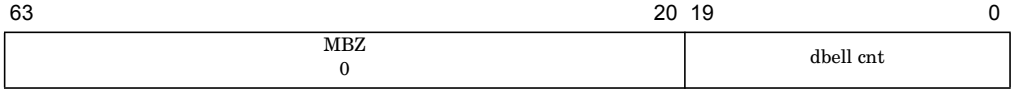
NOTE: The BCH hardware behavior is unpredictable when anything but a 64-bit Store operation (i.e. anything other than an SD) is used.

Physical Address to Store to BCH



- **DID** = 0x71.
- All address bits must be zero.

Store Data on a Store to BCH



- **<63:20> =** Must be 0x0.
- **dbell cnt =** Number of instruction queue 64-bit words to add to the BCH-instruction doorbell count. BCH instructions are always four words.

Cavium Confidential For Bowen Yang ALU 02/27/2014

19.4 Ordering

The BCH hardware always completes the OWORD write operations for a single instruction before it does the RESP write operation for the same instruction. Otherwise the BCH hardware can execute all operations associated with a single instruction, including any required memory operations, in any order.

The BCH hardware may execute instructions out of order, though it generally processes them in order. For example, it can start fetching the input data for a subsequent instruction before it writes the parity/correction data for a prior instruction.

19.5 BCH Registers

The BCH registers are listed in Table 19–1.

Table 19–1 BCH Registers

Register	Address	CSR Type[1]	Detailed Description
BCH_CTL	0x0001180071000000	RSL	See page 855
BCH_CMD_BUF	0x0001180071000008	RSL	See page 855
BCH_ERR_CFG	0x0001180071000010	RSL	See page 855
BCH_BIST_RESULT	0x0001180071000080	RSL	See page 856
BCH_GEN_INT	0x0001180071000088	RSL	See page 856
BCH_GEN_INT_EN	0x0001180071000090	RSL	See page 856

1. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

BCH Control Register
BCH_CTL

See Table 19–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:23>	—	RAZ	—	—	Reserved.
<22>	FREE_ENA	R/W	0	0	Enable freeing of command buffers.
<21:18>	EARLY_TERM	R/W	0x5	0x5	Threshold of zero delta iterations before declaring early termination. 0 forces all iterations to run. Diagnostic only.
<17>	ONE_CMD	RAZ	0	0	Execute a single operation at a time. Diagnostic only.
<16>	ERASE_DISABLE	R/W	0	0	Erase disable. When clear, erased blocks bypass the BCH correction. The 16-byte result word contains an erased block indication. A block is considered erased if the number of 0s found in the block (data + ECC) is less than half the ECC level. For instance, a 2KB block using ECC32 is considered erased if fewer than 16 0s are found in the 2048 + 60 bytes.
<15:6>	—	RAZ	—	—	Reserved.
<5:2>	MAX_READ	R/W	0x8	0x8	Maximum number of outstanding data read commands. MAX_READ is a throttle to control IOB usage. Values greater than 0x8 are illegal.
<1>	STORE_LE	R/W	0	0	Force STORE0 byte write address to little-endian.
<0>	RESET	R/W1	0	0	Reset oneshot pulse (lasts for 4 cycles).

BCH Command Buffer Register
BCH_CMD_BUF

This register sets the command-buffer parameters. See Table 19–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:58>	—	RAZ	0x0	0x0	Reserved
<57:49>	DWB	R/W	0x0	—	Number of don't-write-backs.
<48:46>	POOL	R/W	0x0	—	Free list used to free command buffer segments.
<45:33>	SIZE	R/W	0x0	—	Number of uint64 words per command buffer segment.
<32:0>	PTR	R/W	0x0	—	Initial command-buffer pointer bits <39:7> (128-byte aligned).

BCH Error Configuration Register
BCH_ERR_CFG

See Table 19–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:18>	—	RAZ	—	—	Reserved.
<17:16>	DAT_FLIP	R/W	0x0	0x0	Testing feature. Flip syndrome bits<1:0> on write operations to the DAT RAM to test single-bit or double-bit errors.
<15:1>	—	RAZ	—	—	Reserved.
<0>	DAT_COR_DIS	R/W	0	0	Disable ECC corrector on DAT RAM.

Cavium Confidential For Bowen Yang 02/27/2014

BCH BIST Result Register
BCH_BIST_RESULT

This register provides access to internal BIST results. Each bit is the BIST result of an individual memory (per bit, 0 = pass, 1 = fail). See Table 19–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5>	NCB_OUB	RO	—	0	BIST result of the NCB_OUB memories
<4>	NCB_INB	RO	—	0	BIST result of the NCB_INB memories
<3:0>	DAT	RO	—	0x0	BIST result of the DAT memories

BCH General Interrupt Register
BCH_GEN_INT

See Table 19–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved
<2>	DAT_DBE	R/W1C/H	0	0	An ECC uncorrectable error has occurred in the DAT RAM.
<1>	DAT_SBE	R/W1C/H	0	0	An ECC correctable error has occurred in the DAT RAM.
<0>	DOORBELL	R/W1C/H	0	0	Error bit indicating a doorbell count has overflowed.

BCH Interrupt Mask Register
BCH_GEN_INT_EN

When a mask bit is set, the corresponding interrupt is enabled. See Table 19–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved
<2>	DAT_DBE	R/W	0	0	Interrupt-enable bit mask for BCH_GEN_INT[DAT_DBE].
<1>	DAT_SBE	R/W	0	0	Interrupt-enable bit mask for BCH_GEN_INT[DAT_SBE].
<0>	DOORBELL	R/W	0	0	Interrupt-enable bit mask for BCH_GEN_INT[DOORBELL].

Cavium Confidential For Bowen Yang ALU 02/27/2014

Chapter 20

Key Memory Unit (KEY)

This chapter contains the following subjects:

- Overview
- Key Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The KEY module provides an 8KB common storage area for all cores. This KEY memory can only be accessed by core load/store/IOBDMA operations to physical addresses 0x1 2000 0000 0000 – 0x1 2000 0000 1FFF.

The KEY memory cannot be accessed via CN71XX's PCIe interface, or any other CN71XX interface. The memory is ECC protected, correcting single-bit errors and reporting, via interrupts, both single- and double-bit errors.

20.1 Key Registers

The Key registers are listed in Table 20–1.

Table 20–1 Key Registers

Register	Address	CSR Type[1]	Detailed Description
KEY_INT_SUM	0x0001180020000000	RSL	See page 858
KEY_INT_ENB	0x0001180020000008	RSL	See page 858
KEY_CTL_STATUS	0x0001180020000010	RSL	See page 859
KEY_BIST_REG	0x0001180020000018	RSL	See page 859

1. RSL-type registers are accessed indirectly across the IOI.

KEY Interrupt Summary Register
KEY_INT_SUM

This register contains the different interrupt-summary bits of the KEY. See Table 20–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	KEY_DBE	R/W1C/H	0	0	KEY double-bit error. When set, a KEY double-bit error has occurred.
<0>	KEY_SBE	R/W1C/H	0	0	KEY single-bit error. When set, a KEY single-bit error has occurred.

KEY Interrupt-Enable Register
KEY_INT_ENB

KEY interrupt-enable register. See Table 20–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	KEY_DBE	R/W	0	0	KEY double-bit error interrupt enable. When set to 1 and KEY_INT_SUM[KEY_DBE] = 1, the KEY asserts an interrupt.
<0>	KEY_SBE	R/W	0	0	KEY single-bit error interrupt enable. When set to 1 and KEY_INT_SUM[KEY_SBE] = 1, the KEY asserts an interrupt.

Cavium Confidential For Bowen Yang ALU 02/27/2014

KEY Control and Status Register
KEY_CTL_STATUS

KEY control and status register. See Table 20–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved
<2:1>	KEY_FSYN	R/W	0x0	0x0	Flips the syndrome of the KEY memory.
<0>	KEY_CDIS	R/W	0	0	Turns off the correction disable of the KEY memory.

KEY BIST Status Register
KEY_BIST_REG

The KEY BIST status for memory (0 = pass, 1 = fail). See Table 20–1 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	RRC	RO	0	0	RRC BIST status
<0>	MEM	RO	0	0	MEM BIST status

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

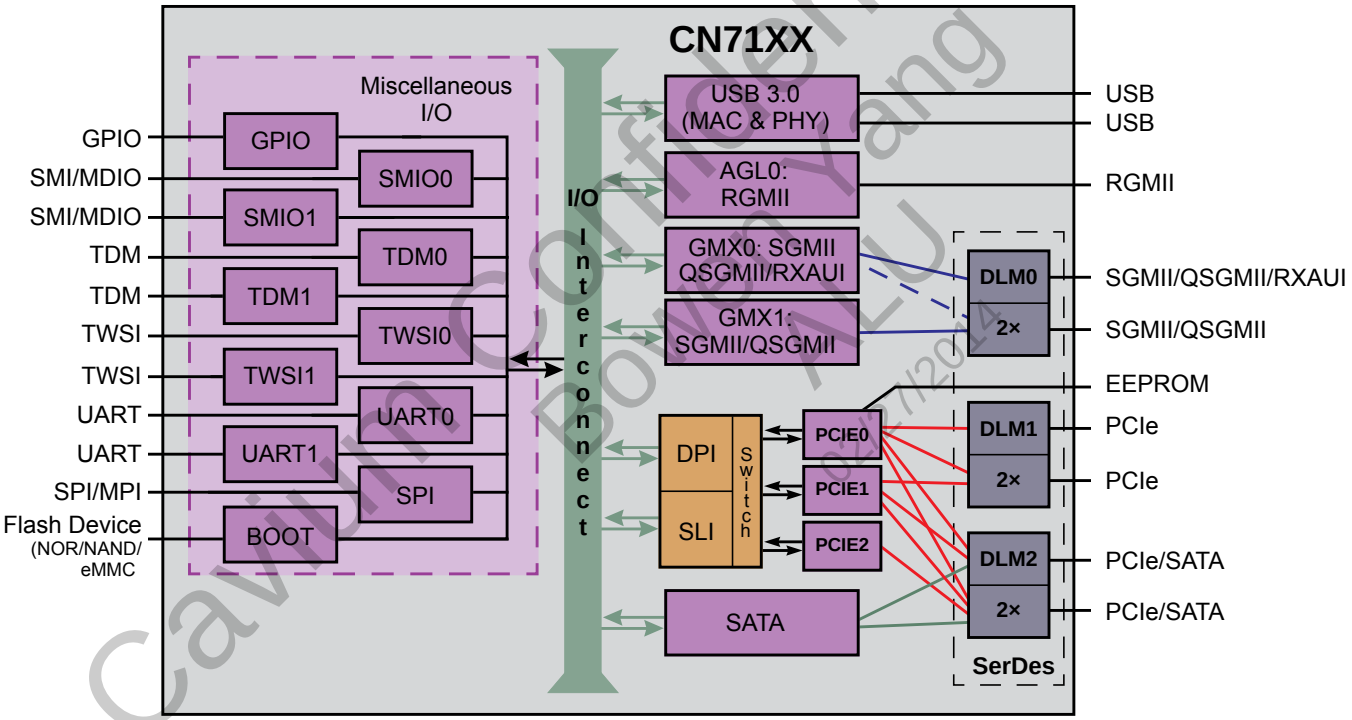
Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Section 3: CN71XX I/O Interfaces

This section contains chapters that describe the various I/O interfaces that are part of the CN71XX (refer to Figure S3–1). They are the following:

- PCI Express interface (PEM)
- Serial Gigabit Media-Independent interface (SGMII)
- Reduced 10-Gigabit Attachment Unit interface (RXAUI)
- Universal Serial Bus (USB) interface
- Serial peripheral interface/multiprocessor peripheral interface (SPI/MPI)
- Reduced Gigabit Media-Independent interface (RGMII)
- Boot-Bus interface (NOR Flash devices)
- Boot-Bus interface (eMMC Flash devices)
- Integrated TDM/PCM interface (TDM)
- Two-Wire Serial interface (TWSI)
- System Management interface (SMI)
- General Purpose I/O bus interface (GPIO)
- Universal Asynchronous Receiver Transmitter interface (UART)

Figure S3–1 CN71XX I/O Interfaces



Cavium Confidential For Bowen Yang ALU 02/7/2014

Serializer-Deserializer (SerDes) Interface

The PCIe, SATA, and SGMII/QSGMII/RXAUI interfaces are in fact shared SerDes interfaces. The SerDes interface is made up of three dual-lane modules (DLM0/1/2).

- DLM0 can be configured for SGMII/QSGMII, or RXAUI.
- DLM1 can be configured for PCIe: ×1, ×2, or ×4 (together with DLM2).
- DLM2 can be configured for SATA or for PCIe: ×1, ×2, or ×4 (together with DLM1).

The possible configurations for each protocol are shown in Table S3–1.

Table S3–1 DLM Configurations

Protocol	Configuration	MAC	DLM
SGMII/QSGMII	× 1	GMX0	DLM0 (lane 0)
		GMX1	DLM0 (lane 1)
RXAUI	× 2	GMX0	DLM0
PCIe	× 4	PEM0	DLM1 & DLM2
	× 2	PEM0	DLM1
		PEM1	DLM2
	× 1	PEM0	DLM1 (lane 2)
		PEM1	DLM1 (lane 3) DLM2 (lane 4)
		PEM2	DLM2 (lane 5)
SATA	× 1	SATA0	DLM2 (lane 4)
		SATA1	DLM2 (lane 5)

Boot Bus Interface

The Boot Bus interface can be used with either NOR, NAND or eMMC devices. This is explained in three chapters: the Boot Bus chapter (explaining the NOR interface, see Chapter 28), the NAND Flash chapter (explaining the NAND interface, see Chapter 30), and the eMMC Flash chapter (explaining the eMMC interface, see Chapter 29).

Cavium Confidential For Bowen Yang ABU 02/27/2017

Chapter 21

General Serializer/Deserializer (GSER) Interface

This chapter contains the following subjects:

- Overview
- Initializing the DLMs
- GSER Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The GSER block controls the SerDes dual-lane modules (DLMs) and manages the data connection to the various MACs in the CN71XX.

21.1 Initializing the DLMs

Typically, a sequence of events is required to initialize the DLMs prior to use with a corresponding MAC.

21.1.1 DLM0 Initialization

DLM0 interfaces only with the two GMX MACs. The main PLL, the TX lanes, and the RX lanes must all be initialized.

Main PLL Initialization

1. Write GSER(0)_DLM(0)_REF_USE_PAD[REF_USE_PAD] = 1 (to select the external reference-clock input, if desired. Otherwise it should be set to 0 to use the internal reference-clock input).
2. Write GSER(0)_DLM(0)_REFCLK_SEL[REFCLK_SEL] if required for the reference-clock selection. This bit is not used if GSER(0)_DLM(0)_REF_USE_PAD[REF_USE_PAD] = 1.
3. If required, write GSER(0)_DLM(0)_REF_CLKDIV2[REF_CLKDIV2] (must be set if reference clock > 100 MHz).
4. Write GSER(0)_DLM(0)_MPLL_MULTIPLIER[MPLL_MULTIPLIER]. See Table 21–1 for programming values.
5. Clear GSER(0)_DLM(0)_TEST_POWERDOWN[TEST_POWERDOWN] = 0.
6. Set GSER(0)_DLM(0)_REF_SSP_EN[REF_SSP_EN] = 1.
7. Set GSER(0)_DLM(0)_MPLL_EN[MPLL_EN] = 1.
8. Clear GSER(0)_DLM(0)_PHY_RESET[PHY_RESET] = 0.
9. Poll until the MPLL locks. Wait for GSER(0)_DLM(0)_MPLL_STATUS[MPLL_STATUS] = 1.

TX Lanes Initialization

1. Write GSER(0)_DLM(0)_TX_RATE[TX*n*_RATE]. Set according to required data rate (see Table 21–1)
2. Set GSER(0)_DLM(0)_TX_EN[TX*n*_EN] = 1.
3. Set GSER(0)_DLM(0)_TX_CM_EN[TXn_CM_EN] = 1.
4. Set GSER(0)_DLM(0)_TX_DATA_EN[TXn_DATA_EN] = 1.
5. Clear GSER(0)_DLM(0)_TX_RESET[TXn_RESET] = 0.
6. Poll GSER(0)_DLM(0)_TX_STATUS[TXn_STATUS, TX*n*_CM_STATUS] until both are set to 1. This prevents GMX from transmitting until the DLM is ready.

Cavium Confidential For Bowen Yang ALU 02/27/2014

RX Lanes Initialization

1. Write GSER(0)_DLM(0)_RX_RATE[RX*n*_RATE] (must match the GSER(0)_DLM(0)_TX_RATE[TX*n*_RATE] setting).
2. Set GSER(0)_DLM(0)_RX_PLL_EN[RX*n*_PLL_EN] = 1.
3. Set GSER(0)_DLM(0)_RX_DATA_EN[RX*n*_DATA_EN] = 1.
4. Clear GSER(0)_DLM(0)_RX_RESET[RX*n*_RESET] = 0.

Now the GMX can be enabled: set GMX(0..1)_INF_MODE[EN] = 1.

The DLM0 data-rate configuration table is shown in Table 21–1.

Table 21–1 DLM0 Data-Rate Configuration Table

Data Rate	Ref. Clock = 100 MHz	Ref. Clock = 125 MHz	Ref. Clock = 156.25 MHz
1.25 Gb/s (SGMII only)	MPLL_MULTIPLIER = 0x19 RX*n*/TX*n*_RATE = 0x2 REF_CLKDIV2 = 0	MPLL_MULTIPLIER = 0x28 RX*n*/TX*n*_RATE = 0x2 REF_CLKDIV2 = 1	MPLL_MULTIPLIER = 0x20 RX*n*/TX*n*_RATE = 0x2 REF_CLKDIV2 = 1
2.5 Gb/s (SGMII only)	MPLL_MULTIPLIER = 0x19 RX*n*/TX*n*_RATE = 0x1 REF_CLKDIV2 = 0	MPLL_MULTIPLIER = 0x28 RX*n*/TX*n*_RATE = 0x1 REF_CLKDIV2 = 1	MPLL_MULTIPLIER = 0x20 RX*n*/TX*n*_RATE = 0x1 REF_CLKDIV2 = 1
3.125 Gb/s (SGMII only)	N/A	MPLL_MULTIPLIER = 0x32 RX*n*/TX*n*_RATE = 0x1 REF_CLKDIV2 = 1	MPLL_MULTIPLIER = 0x28 RX*n*/TX*n*_RATE = 0x1 REF_CLKDIV2 = 1
5.0 Gb/s (QSGMII only)	MPLL_MULTIPLIER = 0x19 RX*n*/TX*n*_RATE = 0x0 REF_CLKDIV2 = 0	MPLL_MULTIPLIER = 0x28 RX*n*/TX*n*_RATE = 0x0 REF_CLKDIV2 = 1	MPLL_MULTIPLIER = 0x20 RX*n*/TX*n*_RATE = 0x0 REF_CLKDIV2 = 1
6.25 Gb/s (RXAUI only)	N/A	MPLL_MULTIPLIER = 0x32 RX*n*/TX*n*_RATE = 0x0 REF_CLKDIV2 = 1	MPLL_MULTIPLIER = 0x28 RX*n*/TX*n*_RATE = 0x0 REF_CLKDIV2 = 1

21.1.2 DLM1/2 Initialization:

DLM1 interfaces only with the PEM0 and PEM1 MACs, while DLM2 can interface with PEMs 0, 1, or 2, or the SATA MAC.

SATA Mode Initialization

1. Write GSER(0)_DLM2_REFCLK_SEL[REFCLK_SEL] if required for reference-clock selection.
2. Write GSER(0)_DLM2_MPLL_MULTIPLIER[MPLL_MULTIPLIER]. For a 100MHz reference clock, set to 0x1E. See Table 21–2 for programming values.
3. Clear GSER(0)_DLM2_TEST_POWERDOWN[TEST_POWERDOWN] = 0.
4. Clear GSER(0)_DLM2_PHY_RESET.
5. Set GSER(0)_SATA_CFG[SATA_EN] = 1 to configure DLM2 multiplexing.
6. Clear either/both lane0 and lane1 resets: GSER(0)_SATA_RX_INVERT[L0_RST, L1_RST] = 0.

Cavium Confidential
Bowen Yang
AU
2/27/2014

The DLM2 data-rate for SATA configuration table is shown in Table 21–2.

Table 21–2 DLM2 SATA Data-Rate Configuration Table

Ref. Clock	Data Rate = 1.5/3.0/6.0 Gb/s
100 MHz	MPLL_MULTIPLIER = 0x1E, REF_CLKDIV2 = 0
125 MHz	MPLL_MULTIPLIER = 0x30, REF_CLKDIV2 = 1

PCIe RC Mode Initialization

1. Write GSER0_DLM(1..2)_REFCLK_SEL[REFCLK_SEL] if required for reference-clock selection
2. If required, write GSER0_DLM(1..2)_REF_CLKDIV2[REF_CLKDIV2] = 1 (must be set if reference clock ≥ 100 MHz)
3. Write GSER0_DLM(1..2)_MPLL_MULTIPLIER[MPLL_MULTIPLIER].
   - for a 100MHz reference clock, set to 0x19.
   - for a 125MHz reference clock, set to 0x28.

   See Table 21–3 for programming values.
4. Configure the PCIE PIPE:
   a. Write GSER0_PCIE_PIPE_PORT_SEL[PIPE_PORT_SEL] to configure the PCIE PIPE.
      0x0 = disables all pipes
      0x1 = enables pipe0 only (PEM0 4-lane)
      0x2 = enables pipes 0 and 1 (PEM0 and PEM1 2-lanes each)
      0x3 = enables pipes 0, 1, 2, and 3 (PEM0, PEM1, and PEM3 are one-lane each)
   b. Configure GSER0_PCIE_PIPE_PORT_SEL[CFG_PEM1_DLM2]. If PEM1 is to be configured, this bit must reflect which DLM it is logically tied to. This bit sets multiplexing logic in GSER, and it is used by the RST logic to determine when the MAC can come out of reset.
      0 = PEM1 is tied to DLM1 (for 3 × 1 PCIe mode).
      1 = PEM1 is tied to DLM2 (for all other PCIe modes).
5. Clear GSER(0)_DLM(1..2)_TEST_POWERDOWN. Configurations that use only DLM1 need not clear GSER(0)_DLM2_TEST_POWERDOWN
6. Clear GSER(0)_DLM(1..2)_PHY_RESET. Configurations that use only DLM1 need not clear GSER(0)_DLM2_PHY_RESET
7. Write the GSER(0)_PCIE_PIPE_RST register to take the appropriate PIPE out of reset. There is a PIPE*n*_RST bit for each PIPE. Clear the appropriate bits based on the configuration (reset is active high).

The DLM1/2 PCIe data-rate configuration table is shown in Table 21–3.

Table 21–3 DLM1/2 PCIe Data-Rate Configuration Table

Data Rate	Ref. Clock = 100 MHz	Ref. Clock = 125 MHz
2.5/5.0 Gb/s	MPLL_MULTIPLIER = 0x19 REF_CLKDIV2 = 0	MPLL_MULTIPLIER = 0x28 REF_CLKDIV2 = 1

Cavium Confidential For Bowen Yang 02/27/2014

21.2 GSER Registers

The GSER registers are listed in Table 21–4.

Table 21–4 GSER Registers

Register	Address	CSR Type[1]	Detailed Description
GSER(0)_DLM(0)_MISC_STATUS	0x1180090000000 + α + β * 0x80000	RSL	See page 869
GSER(0)_DLM(0..2)_REFCLK_SEL	0x1180090000008 + α + β * 0x80000	RSL	See page 869
GSER(0)_DLM(0..2)_MPLL_STATUS	0x1180090001000 + α + β * 0x80000	RSL	See page 869
GSER(0)_DLM(0)_LOOPBK_EN	0x1180090001008 + α + β	RSL	See page 870
GSER(0)_DLM(0..2)_LOS_BIAS	0x1180090001010 + α + β * 0x80000	RSL	See page 870
GSER(0)_DLM(0..2)_LOS_LEVEL	0x1180090001018 + α + β * 0x80000	RSL	See page 870
GSER(0)_DLM(0)_MPLL_EN	0x1180090001020 + α + β	RSL	See page 871
GSER(0)_DLM(0..2)_MPLL_HALF_RATE	0x1180090001028 + α + β * 0x80000	RSL	See page 871
GSER(0)_DLM(0..2)_MPLL_MULTIPLIER	0x1180090001030 + α + β * 0x80000	RSL	See page 871
GSER(0)_DLM(0..2)_PHY_RESET	0x1180090001038 + α + β * 0x80000	RSL	See page 871
GSER(0)_DLM(0..2)_REF_CLKDIV2	0x1180090001040 + α + β * 0x80000	RSL	See page 872
GSER(0)_DLM(0)_REF_SSP_EN	0x1180090001048 + α + β	RSL	See page 872
GSER(0)_DLM(0..2)_REF_USE_PAD	0x1180090001050 + α + β * 0x80000	RSL	See page 872
GSER(0)_DLM(0..2)_TEST_BYPASS	0x1180090001058 + α + β * 0x80000	RSL	See page 872
GSER(0)_DLM(0..2)_TEST_POWERDOWN	0x1180090001060 + α + β * 0x80000	RSL	See page 873
GSER(0)_DLM(0..2)_RX_STATUS	0x1180090002000 + α + β * 0x80000	RSL	See page 873
GSER(0)_DLM(0)_RX_DATA_EN	0x1180090002008 + α + β	RSL	See page 873
GSER(0)_DLM(0..2)_RX_EQ	0x1180090002010 + α + β * 0x80000	RSL	See page 873
GSER(0)_DLM(0)_RX_LOS_EN	0x1180090002018 + α + β	RSL	See page 874
GSER(0)_DLM(0)_RX_PLL_EN	0x1180090002020 + α + β	RSL	See page 874
GSER(0)_DLM(0)_RX_RATE	0x1180090002028 + α + β	RSL	See page 874
GSER(0)_DLM(0)_RX_RESET	0x1180090002030 + α + β	RSL	See page 875
GSER(0)_DLM(0)_RX_TERM_EN	0x1180090002038 + α + β	RSL	See page 875
GSER(0)_DLM(0..2)_TX_STATUS	0x1180090003000 + α + β * 0x80000	RSL	See page 875
GSER(0)_DLM(0)_TX_CM_EN	0x1180090003010 + α + β	RSL	See page 876
GSER(0)_DLM(0)_TX_DATA_EN	0x1180090003018 + α + β	RSL	See page 876
GSER(0)_DLM(0)_TX_EN	0x1180090003020 + α + β	RSL	See page 876
GSER(0)_DLM(0)_TX_PREEMPH	0x1180090003028 + α + β	RSL	See page 877
GSER(0)_DLM(0)_TX_RATE	0x1180090003030 + α + β	RSL	See page 877
GSER(0)_DLM(0)_TX_RESET	0x1180090003038 + α + β	RSL	See page 877
GSER(0)_DLM(0..2)_TX_TERM_OFFSET	0x1180090003040 + α + β * 0x80000	RSL	See page 878
GSER(0)_PCIE_PIPE_STATUS	0x1180090080400 + α	RSL	See page 878
GSER(0)_PCIE_TX_DEEMPH_GEN1	0x1180090080408 + α	RSL	See page 878
GSER(0)_PCIE_TX_DEEMPH_GEN2_3P5DB	0x1180090080410 + α	RSL	See page 879
GSER(0)_PCIE_TX_DEEMPH_GEN2_6DB	0x1180090080418 + α	RSL	See page 879
GSER(0)_PCIE_TX_SWING_FULL	0x1180090080420 + α	RSL	See page 879
GSER(0)_PCIE_TX_SWING_LOW	0x1180090080428 + α	RSL	See page 879
GSER(0)_PCIE_TX_VBOOST_LVL	0x1180090080440 + α	RSL	See page 880

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 21–4 GSER Registers

Register	Address	CSR Type[1]	Detailed Description
GSER(0)_PCIE_PIPE_RST	0x1180090080448 + α	RSL	See page 880
GSER(0)_PCIE_PIPE_RST_STS	0x1180090080450 + α	RSL	See page 881
GSER(0)_PCIE_PIPE_CRST	0x1180090080458 + α	RSL	See page 881
GSER(0)_PCIE_PIPE_PORT_SEL	0x1180090080460 + α	RSL	See page 882
GSER(0)_PCIE_PIPE_PORT_LOOPBK	0x1180090080468 + α	RSL	See page 882
GSER(0)_PCIE_PIPE_COM_CLK	0x1180090080470 + α	RSL	See page 883
GSER(0)_PCIE_PCS_CLK_REQ	0x1180090080478 + α	RSL	See page 883
GSER(0)_PCIE_PIPE(0..3)_TXDEEMPH	0x1180090080480 + α + β * 0x8	RSL	See page 883
GSER(0)_SATA_STATUS	0x1180090100200+ α	RSL	See page 883
GSER(0)_SATA_CFG	0x1180090100208 + α	RSL	See page 884
GSER(0)_SATA_LANE_RST	0x1180090100210 + α	RSL	See page 884
GSER(0)_SATA_RX_INVERT	0x1180090100218 + α	RSL	See page 884
GSER(0)_SATA_TX_INVERT	0x1180090100220 + α	RSL	See page 884
GSER(0)_SATA_SSC_EN	0x1180090100228 + α	RSL	See page 885
GSER(0)_SATA_SSC_RANGE	0x1180090100230 + α	RSL	See page 885
GSER(0)_SATA_SSC_CLK_SEL	0x1180090100238 + α	RSL	See page 885
GSER(0)_SATA_P0_TX_PREEMPH_GEN(1..3)	0x1180090100400 + α + β *0x8	RSL	See page 886
GSER(0)_SATA_P1_TX_PREEMPH_GEN(1..3)	0x1180090100420 + α + β *0x8	RSL	See page 886
GSER(0)_SATA_P0_TX_AMP_GEN(1..3)	0x1180090100480 + α + β *0x8	RSL	See page 886
GSER(0)_SATA_P1_TX_AMP_GEN(1..3)	0x11800901004A0 + α + β *0x8	RSL	See page 886
GSER(0)_SATA_REF_SSP_EN	0x1180090100600 + α	RSL	See page 887
GSER(0)_PHY(0..2)_IDCODE_LO	0x1180090400000 + α + β *0x80000	RSL	See page 887
GSER(0)_PHY(0..2)_IDCODE_HI	0x1180090400008 + α + β *0x80000	RSL	See page 887
GSER(0)_PHY(0..2)_OVRD_IN_LO	0x1180090400088 + α + β *0x80000	RSL	See page 888
GSER(0)_PHY(0..2)_LANE0_TXDEBUG	0x1180090408080 + α + β *0x80000	RSL	See page 888
GSER(0)_PHY(0..2)_LANE0_TX_LBERT_CTL	0x11800904080A8 + α + β *0x80000	RSL	See page 889
GSER(0)_PHY(0..2)_LANE0_RX_LBERT_CTL	0x11800904080B0 + α + β *0x80000	RSL	See page 889
GSER(0)_PHY(0..2)_LANE0_RX_LBERT_ERR	0x11800904080B8 + α + β *0x80000	RSL	See page 890
GSER(0)_PHY(0..2)_LANE1_TXDEBUG	0x1180090408880 + α + β *0x80000	RSL	See page 890
GSER(0)_PHY(0..2)_LANE1_TX_LBERT_CTL	0x11800904088A8 + α + β *0x80000	RSL	See page 890
GSER(0)_PHY(0..2)_LANE1_RX_LBERT_CTL	0x11800904088B0 + α + β *0x80000	RSL	See page 891
GSER(0)_PHY(0..2)_LANE1_RX_LBERT_ERR	0x11800904088B8 + α + β *0x80000	RSL	See page 891

1. RSL-type registers are accessed indirectly across the I/O Interconnect.

Cavium Confidential For Bowen Yang AU 2012014

GSER DLM Miscellaneous Status Register GSER(0)_DLM(0)_MISC_STATUS

This register selects common reference clock 0 or 1. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8>	TX1_UFLOW	R/W1C/H	0	—	When set, indicates transmit FIFO underflow has occurred on lane 1.
<7:1>	—	RAZ	—	—	Reserved.
<0>	TX0_UFLOW	R/W1C/H	0	—	When set, indicates transmit FIFO underflow has occurred on lane 0.

GSER DLM Reference-Clock-Select Registers GSER(0)_DLM(0..2)_REFCLK_SEL

This register selects common reference clock 0 or 1. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	REFCLK_SEL	R/W	0	—	Common reference clock select: 0 = select common reference clock 0, 1 = select common reference clock 1. GSER(0)_DLM(0..2)_REF_USE_PAD[REF_USE_PAD] must be 0 to select either common reference clock.

GSER DLM MPLL Status Registers GSER(0)_DLM(0..2)_MPLL_STATUS

This register is the DLM PLL lock status. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	MPLL_STATUS	RO/H	0	—	Lock status of the PHY PLL. When asserted, it indicates the PHY's MPLL has reached a stable, running state.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GSER DLM Loopback Enable Register GSER(0)_DLM(0)_LOOPBK_EN

This register is the DLM0 transmit-to-receive loopback enable. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8>	LANE1_LOOPBK_EN	R/W	0	—	Lane 1 transmit-to-receive loopback enable. When this signal is asserted, data from the transmit predriver is looped back to the receive slivers. LOS is bypassed and based on the txN_en input so that rxN_los = !txN_data_en.
<7:1>	—	RAZ	—	—	Reserved.
<0>	LANE0_LOOPBK_EN	R/W	0	—	Lane 0 transmit-to-receive loopback enable. When this signal is asserted, data from the transmit predriver is looped back to the receive slivers. LOS is bypassed and based on the txN_en input so that rxN_los = !txN_data_en.

GSER DLM LOS Bias Register GSER(0)_DLM(0..2)_LOS_BIAS

This register is the DLM loss-of-signal detector threshold level control. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved.
<2:0>	LOS_BIAS	R/W/H	0x2	—	A positive, binary bit setting change results in a +15mVp incremental change in the LOS threshold. A negative bit setting change results in a -15-mVp incremental change in the LOS threshold. The 0x0 setting is reserved and must not be used. 0x0 = Reserved 0x1 = 120 mV (default CEI) 0x2 = 135 mV (default PCIe/SATA) 0x3 = 150 mV 0x4 = 45 mV 0x5 = 60 mV 0x6 = 75 mV 0x7 = 90 mV

GSER DLM LOS Level Register GSER(0)_DLM(0..2)_LOS_LEVEL

This register is the DLM loss-of-signal sensitivity level control. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved.
<4:0>	LOS_LEVEL	R/W	0x9	—	Sets the sensitivity level for the loss-of-signal detector. This signal must be set to 0x9.

Cavium Confidential For Ruibin Yang ALU 02/27/2014

GSER DLM MPLL-Enable Register GSER(0)_DLM(0)_MPLL_EN

This register is the DLM0 PHY PLL enable. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	MPLL_EN	R/W	0	—	MPLL enable. When deasserted, the MPLL is off and the PHY is in P2 state.

GSER DLM MPLL Half Rate Register GSER(0)_DLM(0..2)_MPLL_HALF_RATE

This register is the DLM MPLL low-power mode enable. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	MPLL_HALF_RATE	R/W	0	—	Enables a low-power mode feature for the MPLL block. This signal should be asserted only when the MPLL is operating at a clock rate less than or equal to 1.5626 GHz.

GSER DLM MPLL Multiplier Registers GSER(0)_DLM(0..2)_MPLL_MULTIPLIER

This register is the DLM MPLL frequency multiplier control. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved.
<6:0>	MPLL_MULTIPLIER	R/W/H	0x0	—	Multiplies the reference-clock frequency to produce a frequency suitable for the intended operating speed.

GSER DLM Phy Reset Registers GSER(0)_DLM(0..2)_PHY_RESET

This register is the DLM core and state machine reset. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	PHY_RESET	R/W	1	—	Resets the core and all state machines with the exception of the reference-clock buffer and JTAG interface. Asserting PHY_RESET triggers the assertion of the transmit and receive reset signals. Power and clocks are required before deasserting PHY_RESET.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GSER DLM Reference-Clock Divide-by-2 Registers
GSER(0)_DLM(0..2)_REF_CLKDIV2

This register is the DLM input reference clock divider control. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	REF_CLKDIV2	R/W	0	—	If the input reference clock is greater than 100MHz, this signal must be asserted, which causes the reference-clock frequency to be divided by two to keep it in the range required by the MPLL.

GSER DLM REF_SSP-Enable Register
GSER(0)_DLM(0)_REF_SSP_EN

This register is the DLM0 reference clock enable for the PHY. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	REF_SSP_EN	R/W	0	—	Enables the PHY's internal reference clock.

GSER DLM Reference USE PAD Registers
GSER(0)_DLM(0..2)_REF_USE_PAD

This register is the DLM select reference clock. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	REF_USE_PAD	R/W	0	—	When asserted, selects the external ref_pad_clk_{p,m} inputs as the reference-clock source. When deasserted, ref_alt_clk_{p,m} are selected from an on-chip source of the reference clock.

GSER DLM Test Power Down Registers
GSER(0)_DLM(0..2)_TEST_BYPASS

This register is the DLM test bypass. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	TEST_BYPASS	R/W	0	—	When asserted, all circuits power-down but leave reference clock active.

Cavium Confidential For Bowen Yang ALL 02/27/2014

GSER DLM Test Power Down Registers
GSER(0)_DLM(0..2)_TEST_POWERDOWN

This register is the DLM test power down. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	TEST_POWERDOWN	R/W	1	—	When asserted, powers down all circuitry in the PHY for IDDQ testing.

GSER DLM Receive Status Registers
GSER(0)_DLM(0..2)_RX_STATUS

This register is the DLM receive DPLL state indicator. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8>	RX1_STATUS	RO/H	0	—	Indicates the current state of the Lane 1 receiver DPLL and clock. When cleared, rxN_clk can be disabled or not running at its target rate.
<7:1>	—	RAZ	—	—	Reserved.
<0>	RX0_STATUS	RO/H	0	—	Indicates the current state of the Lane 0 receiver DPLL and clock. When cleared, rxN_clk can be disabled or not running at its target rate.

GSER DLM Receive-Data-Enable Register
GSER(0)_DLM(0)_RX_DATA_EN

This register is the DLM receiver enable. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8>	RX1_DATA_EN	R/W	0	—	Enables the clock and data-recovery logic for lane1.
<7:1>	—	RAZ	—	—	Reserved.
<0>	RX0_DATA_EN	R/W	0	—	Enables the clock and data-recovery logic for lane0.

GSER DLM Receive EQ Register
GSER(0)_DLM(0..2)_RX_EQ

This register is the DLM receiver equalization setting. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:11>	—	RAZ	—	—	Reserved.
<10:8>	RX1_EQ	R/W	0	—	Selects the amount of equalization in the lane 1 receiver.
<7:3>	—	RAZ	—	—	Reserved.
<2:0>	RX0_EQ	R/W	0	—	Selects the amount of equalization in the lane 0 receiver.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GSER DLM Receive LOS Enable Register GSER(0)_DLM(0)_RX_LOS_EN

This register is the DLM loss of signal detector enable. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8>	RX1_LOS_EN	R/W	0	—	Lane 1 loss of signal detector enable.
<7:1>	—	RAZ	—	—	Reserved.
<0>	RX0_LOS_EN	R/W	0	—	Lane 0 loss of signal detector enable.

GSER DLM Receive PLL Enable Register GSER(0)_DLM(0)_RX_PLL_EN

This register is the DLM0 DPLL enable. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8>	RX1_PLL_EN	R/W	0	—	Lane1 receiver DPLL enable.
<7:1>	—	RAZ	—	—	Reserved.
<0>	RX0_PLL_EN	R/W	0	—	Lane0 receiver DPLL enable.

GSER DLM Receive Rate Register GSER(0)_DLM(0)_RX_RATE

This register is the DLM0 receive data rate. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved.
<9:8>	RX1_RATE	R/W	0x0	—	Lane 1 receive data rate. 0x0 = mpll_baud_clk 0x1 = mpll_baud_clk/2 0x2 = mpll_baud_clk/4 0x3 = not supported.
<7:2>	—	RAZ	—	—	Reserved.
<1:0>	RX0_RATE	R/W	0x0	—	Lane 0 receive data rate. 0x0 = mpll_baud_clk 0x1 = mpll_baud_clk/2 0x2 = mpll_baud_clk/4 0x3 = not supported.

Cavium Confidential For Bowen Yang BOALU 02/27/2014

GSER DLM Receive Reset Register GSER(0)_DLM(0)_RX_RESET

This register is the DLM0 receiver reset. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8>	RX1_RESET	R/W	1	—	Lane1 receiver reset.
<7:1>	—	RAZ	—	—	Reserved.
<0>	RX0_RESET	R/W	1	—	Lane0 receiver reset.

GSER DLM Receive TERM Enable Register GSER(0)_DLM(0)_RX_TERM_EN

This register is the DLM0 PMA receiver termination. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8>	RX1_TERM_EN	R/W	0	—	Lane 1 PMA receiver termination. 0x0 = Terminations removed 0x1 = Terminations present
<7:1>	—	RAZ	—	—	Reserved.
<0>	RX0_TERM_EN	R/W	0	—	Lane 0 PMA receiver termination. 0x0 = Terminations removed 0x1 = Terminations present

GSER DLM Transmit Status Registers GSER(0)_DLM(0..2)_TX_STATUS

This register is the DLM transmit common mode control status. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved.
<9>	TX1_CM_STATUS	RO/H	0	—	When asserted, the lane1 transmitter differential pair is held to half of vptxN during an electrical IDLE. Otherwise, weakly held to ground through a high-impedance connection.
<8>	TX1_STATUS	RO/H	0	—	Signals when the lane1 transmitter is ready to properly sample the incoming data for transmission.
<7:2>	—	RAZ	—	—	Reserved.
<1>	TX0_CM_STATUS	RO/H	0	—	When asserted, the lane0 transmitter differential pair is held to half of vptxN during an electrical IDLE. Otherwise, weakly held to ground through a high-impedance connection.
<0>	TX0_STATUS	RO/H	0	—	Signals when the lane0 transmitter is ready to properly sample the incoming data for transmission.

Cavium Confidential For Bowen Yang ALU 12/17/2014

GSER DLM Transmit Amplitude Register GSER(0)_DLM(0)_TX_AMPLITUDE

This register is the DLM0 transmit amplitude (full swing mode). See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:15>	—	RAZ	—	—	Reserved.
<14:8>	TX1_AMPLITUDE	R/W	0x0	—	This static value sets the launch amplitude of the lane 1 transmitter when pipeP_tx_swing is set to 0x7F (default state).
<7>	—	RAZ	—	—	Reserved.
<6:0>	TX0_AMPLITUDE	R/W	0x0	—	This static value sets the launch amplitude of the lane 0 transmitter when pipeP_tx_swing is set to 0x7F (default state).

GSER DLM Transmit CM Enable Register GSER(0)_DLM(0)_TX_CM_EN

This register is the DLM0 transmit common-mode control enable. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8>	TX1_CM_EN	R/W	0	—	Enables the lane1 transmitter's common mode-hold circuitry.
<7:1>	—	RAZ	—	—	Reserved.
<0>	TX0_CM_EN	R/W	0	—	Enables the lane0 transmitter's common mode-hold circuitry.

GSER DLM Transmit-Data-Enable Register GSER(0)_DLM(0)_TX_DATA_EN

This register is the DLM0 transmit driver enable. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8>	TX1_DATA_EN	R/W	0	—	Enables the lane1 primary transmitter-driver for serial data.
<7:1>	—	RAZ	—	—	Reserved.
<0>	TX0_DATA_EN	R/W	0	—	Enables the lane0 primary transmitter driver for serial data.

GSER DLM Transmit-Enable Register GSER(0)_DLM(0)_TX_EN

This register is the DLM transmit clocking and data sampling enable. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8>	TX1_EN	R/W	0	—	Enables the lane1 transmit-clock path and transmit word alignment.
<7:1>	—	RAZ	—	—	Reserved.
<0>	TX0_EN	R/W	0	—	Enables the lane0 transmit-clock path and transmit word alignment.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GSER DLM DLM Transmit Preemphasis Register GSER(0)_DLM(0)_TX_PREEMPH

This register is the DLM0 transmit deemphasis. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:15>	—	RAZ	—	—	Reserved.
<14:8>	TX1_PREEMPH	R/W	0x0	—	Sets the lane 1 transmit driver deemphasis value to meet the transmit eye mask.
<7>	—	RAZ	—	—	Reserved.
<6:0>	TX0_PREEMPH	R/W	0x0	—	Sets the Lane 0 transmit driver deemphasis value to meet the transmit eye mask.

GSER DLM Transmit-Rate Register GSER(0)_DLM(0)_TX_RATE

This register is the DLM0 transmit data rate. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved.
<9:8>	TX1_RATE	R/W	0x0	—	Selects the lane1 baud rate for the transmit path. 0x0 = baud 0x1 = baud / 2 0x2 = baud / 4 0x3 = not supported
<7:2>	—	RAZ	—	—	Reserved.
<1:0>	TX0_RATE	R/W	0x0	—	Selects the lane0 baud rate for the transmit path. 0x0 = baud 0x1 = baud / 2 0x2 = baud / 4 0x3 = not supported

GSER DLM Transmit-Reset Register GSER(0)_DLM(0)_TX_RESET

This register is the DLM0 transmit reset. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8>	TX1_RESET	R/W	1	—	Resets all lane1 transmitter settings and state machines.
<7:1>	—	RAZ	—	—	Reserved.
<0>	TX0_RESET	R/W	1	—	Resets all lane0 transmitter settings and state machines.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GSER DLM Transmit Term Offset Register GSER(0)_DLM(0..2)_TX_TERM_OFFSET

This register is the DLM transmitter termination offset. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	—	—	Reserved.
<12:8>	TX1_TERM_OFFSET	R/W	0x0	—	Applies an offset to the lane 1 resistor calibration value. Not to be used during normal operation.
<7:5>	—	RAZ	—	—	Reserved.
<4:0>	TX0_TERM_OFFSET	R/W	0x0	—	Applies an offset to the lane 0 resistor calibration value. Not to be used during normal operation.

GSER PCIe Pipe Status Register GSER(0)_PCIE_PIPE_STATUS

This register is the PCIE PIPE status. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved.
<3>	PIPE3_CLKREQN	RO	1	—	When asserted, indicates that the PCS/PHY layer is in a mode where reference clocks are required. When deasserted, the PIPE PCS is powered down into a state where the external reference clocks can be turned off.
<2>	PIPE2_CLKREQN	RO	1	—	When asserted, indicates that the PCS/PHY layer is in a mode where reference clocks are required. When deasserted, the PIPE PCS is powered down into a state where the external reference clocks can be turned off.
<1>	PIPE1_CLKREQN	RO	1	—	When asserted, indicates that the PCS/PHY layer is in a mode where reference clocks are required. When deasserted, the PIPE PCS is powered down into a state where the external reference clocks can be turned off.
<0>	PIPE0_CLKREQN	RO	1	—	When asserted, indicates that the PCS/PHY layer is in a mode where reference clocks are required. When deasserted, the PIPE PCS is powered down into a state where the external reference clocks can be turned off.

GSER PCIe Transmit GEN1 Deemphasis Register GSER(0)_PCIE_TX_DEEMPH_GEN1

This register is the PCIe transmit deemphasis at 3.5 dB. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved.
<5:0>	TX_DEEMPH_GEN1	R/W	0x18	—	This static value sets the launch amplitude of the transmitter when pipeP_tx_swing is set to 0x0 (default state). Used for tuning at the board level for receive eye compliance.

Cavium Confidential For Bowen Yang 02/12/2014

GSER PCIe Transmit 3P5DB Deemphasis Register GSER(0)_PCIE_TX_DEEMPH_GEN2_3P5DB

This register is the PCIe transmit deemphasis at 3.5 dB. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved.
<5:0>	TX_DEEMPH_GEN2_3P5DB	R/W	0x18	—	This static value sets the transmit driver deemphasis value in the case where pipeP_tx_deemph is set to 1'b1 (default setting) and the PHY is running at the Gen2 rate. Used for tuning at the board level for receive eye compliance.

GSER PCIe Transmit GEN2 6DB Deemphasis Register GSER(0)_PCIE_TX_DEEMPH_GEN2_6DB

This register is the PCIe transmit deemphasis at 6 dB. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved.
<5:0>	TX_DEEMPH_GEN2_6DB	R/W	0x23	—	This static value sets the transmit driver deemphasis value in the case where pipeP_tx_deemph is set to 1'b0 and the PHY is running at the Gen2 rate. Used for tuning at the board level for receive eye compliance.

GSER PCIe Transmit Swing Full Register GSER(0)_PCIE_TX_SWING_FULL

This register is the PCIe transmit amplitude (full swing mode). See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved.
<6:0>	TX_SWING_HI	R/W	0x7F	—	This static value sets the launch amplitude of the transmitter when pipeP_tx_swing is set to 1'b0 (default state). Used for tuning at the board level for receive eye compliance.

GSER PCIe Transmit Swing Low Register GSER(0)_PCIE_TX_SWING_LOW

This register is the PCIe transmit amplitude (low swing mode). See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved.
<6:0>	TX_SWING_LO	R/W	0x79	—	This static value sets the launch amplitude of the transmitter when pipeP_tx_swing is set to 1'b1 (low swing mode). Used for tuning at the board level for receive eye compliance.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GSER PCIe Transmit Voltage Boost Level Register GSER(0)_PCIE_TX_VBOOST_LVL

This register is the PCIe transmit voltage boost level. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved.
<2:0>	TX_VBOOST_LVL	R/W	0x4	—	Controls the launch amplitude only when VPTX is less than the launch amplitude corresponding to tx_vboost_lvl. Valid settings: 0x3 = Corresponds to a launch amplitude of 0.844V 0x4 = Corresponds to a launch amplitude of 1.008V 0x5 = Corresponds to a launch amplitude of 1.156V.

GSER PCIe Pipe-Reset Register GSER(0)_PCIE_PIPE_RST

This register is the PCIE PIPE reset. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved.
<3>	PIPE3_RST	R/W	1	—	Pipe 3 reset. Setting this bit puts pipe 3 into reset. PEM2 is always tied to Pipe 3.
<2>	PIPE2_RST	R/W	1	—	Pipe 2 reset. Setting this bit puts pipe 2 into reset. PEM1 is tied to Pipe 2 in 3x1 PCIe mode (GSER_PCIE_PIPE_PORT_SEL[PIPE_PORT_SEL] is set to 4-pipe mode, and GSER_PCIE_PIPE_PORT_SEL [CFG_PEM1_DLM2] is also set).
<1>	PIPE1_RST	R/W	1	—	Pipe 1 reset. Setting this bit puts pipe 1 into reset. PEM1 is tied to Pipe 1 in 2x2 PCIe or 2x1 PCIe with SATA modes.
<0>	PIPE0_RST	R/W/H	0	—	Pipe 0 reset. Setting this bit puts pipe 0 into reset. PEM0 is always tied to Pipe 0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GSER PCIe Pipe Reset Status Register GSER(0)_PCIE_PIPE_RST_STS

This register is the PCIE PIPE status reset. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved.
<3>	PIPE3_RST	RO	0	—	Reflects the current state of the pipe3_rst_n which includes the rst__pem2_pcs_rst_n term from the reset controller. Note that when PIPE3_RST is asserted (active low), no Pipe clocks are generated to PEM3 and any RSL reads to the application side registers will time out.
<2>	PIPE2_RST	RO	0	—	Reflects the current state of the pipe2_rst_n which includes the rst__pem2_pcs_rst_n term from the reset controller. Note that when PIPE2_RST is asserted (active low) and PEM1 is being used in 3x1 PCIe mode (4-Pipe Mode with CFG_PEM1_DLM2 set), no Pipe clocks are generated to PEM1 and any RSL reads to the application side registers will time out.
<1>	PIPE1_RST	RO	0	—	Reflects the current state of the pipe1_rst_n which includes the rst__pem1_pcs_rst_n term from the reset controller. Note that when PIPE1_RST is asserted (active low) and PEM1 is being used in 2x2 PCIe or 2x1 PCIe with SATA, no Pipe clocks are generated to PEM1and any RSL reads to the application side registers will time out.
<0>	PIPE0_RST	RO	—	—	Reflects the current state of the pipe0_rst_n which includes the rst__pem0_pcs_rst_n term from the reset controller. Note that when PIPE0_RST is asserted (active low), no Pipe clocks are generated to PEM0 and any RSL reads to the application side registers will time out.

GSER PCIe Pipe Reset Register GSER(0)_PCIE_PIPE_CRST

This register is the PCIE PIPE cold reset. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	PIPE_CRST	R/W	0	—	PCIE PIPE async cold reset control.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GSER PCIe Pipe Port-Select Register GSER(0)_PCIE_PIPE_PORT_SEL

This register is the PCIE PIPE enable request. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved.
<2>	CFG_PEM1_DLM2	R/W	0	—	The Pipe/PHY configuration for PEM1 when in four-pipe mode. Used as status for determining which DLM PEM1 is tied to. This bit should not be set in single-pipe or two-pipe mode. 0x0 = PEM1 is tied to DLM1. This is 3 × 1 PCIe mode when all four pipes are enabled. 0x1 = PEM1 is tied to DLM2. This is 2 × 1 PCIe mode with SATA
<1:0>	PIPE_PORT_SEL	R/W	0x0	—	Pipe enable request. Change only when phy_reset is asserted. 0x0 = disables all pipes 0x1 = single-pipe mode. Enables pipe0 only. This is 1 × 4 PCIe mode. 0x2 = two-pipe mode. Enables pipes 0 and 1. This is 2 × 2 PCIe mode or 1 × 2 PCIe mode with SATA. 0x3 = four-pipe mode. Enables pipes 0, 1, 2, and 3. This is 2 × 1 PCIe mode with SATA or 3 × 1 PCIe mode.

GSER PCIe Pipe Port-Loopback Register GSER(0)_PCIE_PIPE_PORT_LOOPBK

This register is the PCIe transmit-to-receive loopback enable. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved.
<3>	PIPE3_LOOPBK	R/W	0	—	When this signal is asserted, data from the PIPE3 transmit predriver is looped back to the receive slicers. LOS is bypassed and based on the TxN_en input so that rxN_los !txN_data_en.
<2>	PIPE2_LOOPBK	R/W	0	—	When this signal is asserted, data from the PIPE2 transmit predriver is looped back to the receive slicers. LOS is bypassed and based on the TxN_en input so that rxN_los !txN_data_en.
<1>	PIPE1_LOOPBK	R/W	0x0	—	When this signal is asserted, data from the PIPE1 transmit predriver is looped back to the receive slicers. LOS is bypassed and based on the TxN_en input so that rxN_los !txN_data_en.
<0>	PIPE0_LOOPBK	R/W	0	—	When this signal is asserted, data from the PIPE0 transmit predriver is looped back to the receive slicers. LOS is bypassed and based on the TxN_en input so that rxN_los !txN_data_en.

Cavium Confidential For Bowen Yang ALU 08/16/14

GSER PCIe Pipe Common Clock Register GSER(0)_PCIE_PIPE_COM_CLK

This register is the PCIe select common clock mode for receive data path. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	COM_CLK	R/W	0	—	When both ends of a PCIe link share a common reference clock, the latency through the receiver's elasticity buffer can be shorter, because no frequency offset exists between the two ends of the link. Assert this control only if all physical lanes of the PHY are guaranteed to be connected to links that share a common reference clock.

GSER PCIe Pipe PCS Clock Request GSER(0)_PCIE_PCS_CLK_REQ

This register is the PCIE PIPE clock required. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	CLK_REQ	R/W	0	—	When asserted, indicates that the external logic requires the PHY's PCLK to remain active, preventing the PIPE PCS from powering off that clock while in the P2 state. Note, the PCS hangs if this bit is asserted when a Fundamental Reset is issued to the PEM.

GSER PCIe Pipe Transmit Deemphasis Register GSER(0)_PCIE_PIPE(0..3)_TXDEEMPH

This register is the PCIE PIPE transmitter deemphasis. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	PIPE_TXDEEMPH	R/W	0	—	Selects transmitter deemphasis. 0x0 = enabled (6 dB / 3.5 dB) 0x1 = No deemphasis

GSER SATA Status Register GSER(0)_SATA_STATUS

This register is the SATA PHY ready status. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved.
<1>	P1_RDY	RO	0	—	PHY Lane 1 is ready to send and receive data.
<0>	P0_RDY	RO	0	—	PHY Lane 0 is ready to send and receive data.

Cavium Confidential For Bowen Yang ALU 04/27/2014

GSER SATA Configuration Register GSER(0)_SATA_CFG

This register is the SATA config enable. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	SATA_EN	R/W	0	—	SATA enable: 1 = configure DLM2 for SATA, 0 = not configured for SATA.

GSER SATA Lane-Reset Register GSER(0)_SATA_LANE_RST

This register is the lane reset control. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved.
<1>	L1_RST	R/W	1	—	Independent reset for lane1.
<0>	L0_RST	R/W	1	—	Independent reset for lane0.

GSER SATA Receive Invert Register GSER(0)_SATA_RX_INVERT

This register is the SATA receive polarity inversion. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved.
<1>	RX1_INVERT	R/W	0	—	Instructs the PHY to perform a polarity inversion on the Lane 1 received data. 0x0 = PHY does not perform polarity inversion 0x1 = PHY performs polarity inversion
<0>	RX0_INVERT	R/W	0	—	0x0 = PHY does not perform polarity inversion 0x1 = PHY performs polarity inversion

GSER SATA Transmit Invert Register GSER(0)_SATA_TX_INVERT

This register is the SATA transmit polarity inversion. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved.
<1>	TX1_INVERT	R/W	0	—	Instructs the PHY to perform a polarity inversion on the Lane 1 transmitted data. 0x0 = PHY does not perform polarity inversion 0x1 = PHY performs polarity inversion
<0>	TX0_INVERT	R/W	0	—	Instructs the PHY to perform a polarity inversion on the Lane 0 transmitted data. 0x0 = PHY does not perform polarity inversion 0x1 = PHY performs polarity inversion

Cavium Confidential For Bowen Yang ADI 02/27/2014

GSER SATA SSC Enable Register GSER(0)_SATA_SSC_EN

This register is the SATA spread spectrum disable. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	SSC_EN	R/W	0	—	Enables spread spectrum clock production (0.5% down-spread at ~31.5 KHz) in the SATA 6G PHY. If the reference clock already has spread spectrum applied, this bit must stay deasserted

GSER SATA SSC Range Register GSER(0)_SATA_SSC_RANGE

This register is the SATA spread spectrum range. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved.
<2:0>	SSC_RANGE	R/W	0x0	—	Selects the range of spread spectrum modulation when SSC_EN is asserted and the PHY is spreading the high-speed transmit clocks. Applies a fixed offset to the accumulator. 0x0 = -4.980 ppm 0x1 = -4.492 ppm 0x2 = -4.003 ppm 0x3 = -2.000 ppm 0x4 = 4.980 ppm 0x5 = 4.492 ppm 0x6 = 4.003 ppm 0x7 = 2.000 ppm

GSER SATA SSC Clock Select Register GSER(0)_SATA_SSC_CLK_SEL

This register is the SATA spread spectrum reference clock shifting. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved.
<8:0>	SSC_CLK_SEL	R/W	0x0	—	Enables non-standard oscillator frequencies to generate targeted MPLL output rates. Input corresponds to frequency-synthesis coefficient. [8:6]: modulous - 1 [5:0] = 2's compliment push amount.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GSER SATA Port Transmit Preemphasis Gen Register GSER(0)_SATA_P0_TX_PREEMPH_GEN(1..3)

This register is the SATA lane 0 transmit preemphasis at Gen 1, 2 and 3 speeds. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved.
<6:0>	TX_PREEMPH	R/W	0x0	—	This static value sets the transmit driver deemphasis value in the case where the PHY is running at the Gen1, Gen2, and Gen3 rates. Used for tuning at the board level for receive eye compliance.

GSER SATA Port Transmit Preemphasis Gen Register GSER(0)_SATA_P1_TX_PREEMPH_GEN(1..3)

This register is the SATA lane 1 transmit preemphasis at Gen 1, 2 and 3 speeds. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved.
<6:0>	TX_PREEMPH	R/W	0x0	—	This static value sets the transmit driver deemphasis value in the case where the PHY is running at the Gen1, Gen2, and Gen3 rates. Used for tuning at the board level for receive eye compliance.

GSER SATA Port Transmit Amplitude Gen Register GSER(0)_SATA_P0_TX_AMP_GEN(1..3)

This register is the SATA lane 0 transmit launch amplitude at Gen 1,2 and 3 speeds. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved.
<6:0>	TX_AMP_GEN	R/W	0x0	—	This status value sets the transmit driver launch amplitude in the case where the PHY is running at the Gen1, Gen2, and Gen3 rates. Used for tuning at the board level for receive eye compliance.

GSER SATA Port Transmit Amplitude Gen Register GSER(0)_SATA_P1_TX_AMP_GEN(1..3)

This register is the SATA lane 1 transmit launch amplitude at Gen 1,2 and 3 speeds. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved.
<6:0>	TX_AMP_GEN	R/W	0x0	—	This status value sets the transmit driver launch amplitude in the case where the PHY is running at the Gen1, Gen2, and Gen3 rates. Used for tuning at the board level for receive eye compliance.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GSER SATA Ref SSP Enable Register GSER(0)_SATA_REF_SSP_EN

This register is the SATA reference clock enable for the PHY. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	REF_SSP_EN	R/W	0	—	Reference clock enable for the PHY.

GSER PHY ID CODE LO Register GSER(0)_PHY(0..2)_IDCODE_LO

This register is the PHY version low. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved.
<0>	IDCODE_LO	RO	0x74CD	—	The PHY version low.

GSER PHY ID CODE HI Register GSER(0)_PHY(0..2)_IDCODE_HI

This register is the PHY version hi. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved.
<0>	IDCODE_HI	RO	0x9223	—	The PHY version high.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GSER PHY Override Input Low Register GSER(0)_PHY(0..2)_OVRD_IN_LO

This register is the PHY override input low register. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved.
<15>	RES_ACK_IN_OVRD	R/W	0	—	Override enable for RES_ACK_IN input. It is not expected Software will need to set this bit.
<14>	RES_ACK_IN	R/W	0	—	Override value for RES_ACK_IN input. It is not expected Software will need to set this bit.
<13>	RES_REQ_IN_OVRD	R/W	0	—	Override enable for RES_REQ_IN input. It is not expected Software will need to set this bit.
<12>	RES_REQ_IN	R/W	0	—	Override value for RES_REQ_IN input. It is not expected Software will need to set this bit.
<11>	RTUNE_REQ_OVRD	R/W	0	—	Override enable for RTUNE_REQ input. It is not expected Software will need to set this bit.
<10>	RTUNE_REQ	R/W	0	—	Override value for RTUNE_REQ input. It is not expected Software will need to set this bit.
<9>	MPLL_MULTIPLIER_OVRD	R/W	0	—	Override enable for MPLL_MULTIPLIER. It is not expected Software will need to set this bit.
<8:2>	MPLL_MULTIPLIER	R/W	0x19	—	Override value for MPLL_MULTIPLIER inputs. It is not expected Software will need to set this bit.
<1>	MPLL_EN_OVRD	R/W	0	—	Override enable for MPLL_EN input. For EP Mode PEMs, Software should set this bit after reset.
<0>	MPLL_EN	R/W	0	—	Override value for MPLL_EN input. For EP Mode PEMs, Software should set this bit after reset.

GSER PHY Lane N TX DEBUG Register GSER(0)_PHY(0..2)_LANE0_TXDEBUG

This register is the PHY LANE0 TX DEBUG. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved.
<11:4>	RXDET_MEAS_TIME	R/W	0x10	—	Time to wait for rxdet measurement.
<3>	DETRX_ALWAYS	R/W	0	—	Always signals 1 for rx_detect ignoring analog.
<2:0>	DTB_SEL	R/W	0x0	—	Selects data to drive on the DTB.

Cavium Confidential
Bowen Yang ALU
02/27/2014

GSER PHY Lane 0 TX BERT Control Register GSER(0)_PHY(0..2)_LANE0_TX_LBERT_CTL

This register is the PHY LANE0 TX LBERT control. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:15>	—	RAZ	—	—	Reserved.
<14:5>	PAT0	R/W	0x0	—	10-bit pattern for modes that use this field. Ignored for other modes.
<4>	TRIG_ERR	R/W	0	—	Single shot inversion of the LSB of the current symbol. Any write of 1 to this bit will insert an error.
<3:0>	MODE	R/W	0x0	—	Pattern to generate. When changing modes, the field must be set to zero first. 0x0 = disabled 0x1 = lfsr31 X^31 + X^28 + 1 0x2 = lfsr23 X^23 + X^18 + 1 0x3 = lfsr15 X^15 + X^14 + 1 0x4 = lfsr7 X^7 + X^6 + 1 0x5 = Fixed word (PAT0) 0x6 = DC-balanced word (PAT0, ~PAT0) 0x7 = Word pattern (20-bit) 0xF-0x8 = Reserved.

GSER PHY Lane 0 RX LBERT Control Register GSER(0)_PHY(0..2)_LANE0_RX_LBERT_CTL

This register is the PHY LANE0 RX LBERT control. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved.
<4>	SYNC	R/W	0	—	Synchronizes pattern matcher with incoming data. A write of a 1 to this bit resets the error counter and starts a synchronization of the PM. Once this bit is set, there is no need to write the field back to a zero.
<3:0>	MODE	R/W	0x0	—	Pattern to match. When changing modes, the field must be set to zero first. This field should match what was configured for the TX LBERT control register. 0x0 = disabled 0x1 = lfsr31 X^31 + X^28 + 1 0x2 = lfsr23 X^23 + X^18 + 1 0x3 = lfsr15 X^15 + X^14 + 1 0x4 = lfsr7 X^7 + X^6 + 1 0x5 = d[n] = d[n-10] 0x6 = d[n] = !d[n-10] 0x7 = d[n] = !d[n-20] 0xF-0x8 = Reserved.

Cavium Confidential For Bowen Yang AE 5/7/2014

GSER PHY Lane 0 RX LBERT Error Register
GSER(0)_PHY(0..2)_LANE0_RX_LBERT_ERR

This register is the PHY LANE0 RX LBERT error. A read of this register, or a SYNC from the RX LBERT control register resets the error count. If all bits in this register are set, the error counter has saturated. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved.
<15>	OV14	R/W	0	—	If this bit is set, and COUNT[15] is also set, signals a overflow of counter.
<14:0>	COUNT	R/W	0x0	—	Current error count if OV14 field is active, then multiply count by 128 to get the actual count.

GSER PHY Lane N TX DEBUG Register
GSER(0)_PHY(0..2)_LANE1_TXDEBUG

This register is the PHY LANE1 TX DEBUG. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved.
<11:4>	RXDET_MEAS_TIME	R/W	0x10	—	Time to wait for rxdet measurement.
<3>	DETRX_ALWAYS	R/W	0	—	Always signals 1 for rx_detect ignoring analog.
<2:0	DTB_SEL	R/W	0x0	—	Selects data to drive on the DTB.

GSER PHY Lane 1 TX BERT Control Register
GSER(0)_PHY(0..2)_LANE1_TX_LBERT_CTL

This register is the PHY LANE1 TX LBERT control. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:15>	—	RAZ	—	—	Reserved.
<14:5>	PAT0	R/W	0x0	—	10-bit pattern for modes that use this field. Ignored for other modes.
<4>	TRIG_ERR	R/W	0	—	Single shot inversion of the LSB of the current symbol. Any write of 1 to this bit will insert an error.
<3:0	MODE	R/W	0x0	—	Pattern to generate. When changing modes, the field must be set to zero first. 0x0 = disabled 0x1 = lfsr31 X^31 + X^28 + 1 0x2 = lfsr23 X^23 + X^18 + 1 0x3 = lfsr15 X^15 + X^14 + 1 0x4 = lfsr7 X^7 + X^6 + 1 0x5 = Fixed word (PAT0) 0x6 = DC-balanced word (PAT0, ~PAT0) 0x7 = Word pattern (20-bit) 0xF-0x8 = Reserved.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GSER PHY Lane 1 RX LBERT Control Register GSER(0)_PHY(0..2)_LANE1_RX_LBERT_CTL

This register is the PHY LANE1 RX LBERT control. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved.
<3>	SYNC	R/W	0	—	Synchronizes pattern matcher with incoming data. A write of a 1 to this bit resets the error counter and starts a synchronization of the PM. Once this bit is set, there is no need to write the field back to a zero.
<2:0>	MODE	R/W	0x0	—	Pattern to match. When changing modes, the field must be set to zero first. This field should match what was configured for the TX LBERT Control register. 0x0 = disabled 0x1 = lfsr31 X^31 + X^28 + 1 0x2 = lfsr23 X^23 + X^18 + 1 0x3 = lfsr15 X^15 + X^14 + 1 0x4 = lfsr7 X^7 + X^6 + 1 0x5 = d[n] = d[n-10] 0x6 = d[n] = !d[n-10] 0x7 = d[n] = !d[n-20]

GSER PHY Lane 0 RX LBERT Error Register GSER(0)_PHY(0..2)_LANE1_RX_LBERT_ERR

This register is the PHY LANE1 RX LBERT error. A read of this register, or a SYNC from the RX LBERT control register resets the error count. If all bits in this register are set, the error counter has saturated. See Table 21–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved.
<15>	OV14	R/W	0	—	If this bit is set, and COUNT[15] is also set, signals an overflow of counter.
<14:0>	COUNT	R/W	0x0	—	Current error count if OV14 field is active, then multiply count by 128 to get the actual count.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 22

PCI Express Interface (PEM)

This chapter contains the following subjects:

- Overview
- Summary of PEM Features
- PEM as a PCIe Receiver
- PEM as a PCIe Transmitter
- PEM0 Configuration-EEPROM Interface
- PCIe Reset Sequences
- PCIe Configuration Summary
- PCIe Configuration Registers
- PEM Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

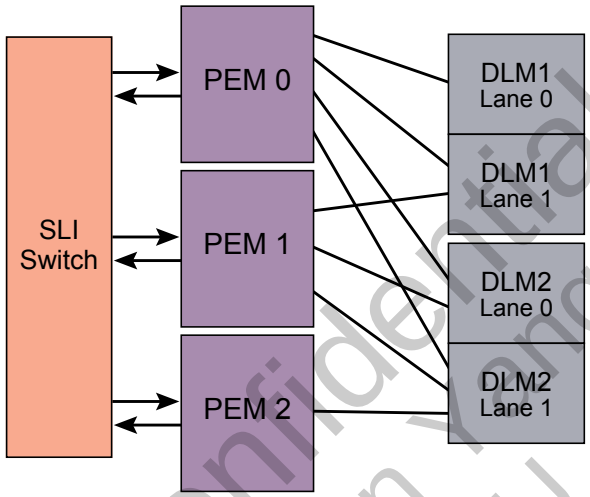
Overview

The PEM is a PCIe MAC that internally interfaces to DPI/SLI and corresponding DLM SerDes external PCIe interface.

- Section 22.1 outlines the PEM features.
- Sections 22.2 and 22.3 describe logical characteristics of PEM as a PCIe receiver and as a PCIe transmitter.
- Section 22.4 describes the optional EEPROM that can store PCIe configuration information.
- Section 22.5 describes the reset sequences of the PEM logic.
- Sections 22.7 and 22.8 define the PEM PCIe and PEM registers.

Figure 22–1 is a block diagram that shows the internal architecture of the PEM block on CN71XX.

Figure 22–1 CN71/70XX PEM Block Diagram



22.1 Summary of PEM Features

- PCIe Root Complex and Endpoint support
- PCIe Specification V2.0 compliant (2.5 and 5.0 Gb/s only)
- One, two, or three PCIe ports
  - Port 0 can be either root complex (RC) or endpoint (EP) mode; ports 1 and 2 support RC mode only.
  - Reset pin per port
    - PERST0_L – MAC reset input when endpoint, else driven
    - PERST1_L – always driven
    - PERST2_L – always driven
- PCIe Memory Space Master Support (with DPI and SLI)
  - 16 separate 38-bit windows (translated to 64-bit) available for direct core access to PCIe memory space in SLI (refer to Section 12.8.6)
    - Programmable store merging

Cavium Confidential For Bowen Yang ALU 8/27/2014

- Direct core PCIe I/O space and configuration space request generation windows in SLI (refer to Section 12.8.4, Section 12.8.5)
- Four DPI packet input ports, 32 packet input rings (refer to Section 12.1)
  - Efficient remote host interface
  - Scatter/gather support on the remote host
- Four DPI packet output ports, 32 packet output rings (refer to Section 12.2)
  - Efficient remote host buffer input
  - Multibuffer support with optional info/buffer split
  - Condensed interrupts
- DPI DMA engines (refer to Section 12.3)

- PCIe Target Support (with SLI)
  - Independent endpoint and root complex BAR0, BAR1, and BAR2 support
  - CN71XX internal CSR access in SLI through PCIe memory space BAR0
  - PCIe Memory Space BAR1 (16-entry, translated and protected) 32-bit compatible access to entire L2/DRAM
    - EP mode: BAR1 size is 64MB
    - RC mode: BAR1 size is 64MB–2GB
  - PCIe memory space BAR2 (protected) for direct access to L2/DRAM
  - PCIe configuration space access in EP mode
  - PCIe expansion ROM BAR (forwarded to the boot bus) on port 0 in EP mode
- Detailed PCIe support summary
  - One virtual channel
  - Max_Payload_Size up to 256 bytes
  - Read Completion Boundary = 128 bytes
  - ECRC implemented
  - Advanced Error Reporting implemented
  - MSI implemented
    - Target support for 64 destinations at each of 4 priorities (256 total)
      - → Flexible priority translation
    - Initiator support for single message, no per-vector masking
  - ASPM support for both L0s and L1
    - CN71XX DLM/PHY power is not reduced in these states, but the power consumed by the link partner may be
  - Extended tag supported, but never generated
  - Phantom function numbers not generated
  - Relaxed ordering / no snoop support
  - Lane reversal supported
    - Lane 0 for a port must always be connected
  - Power management support
    - No support for L2/P2 nor D1 nor D2, nor beacons

Cavium Confidential For Bowen Yang ALU 01/7/2014

22.2 PEM as a PCIe Receiver

This section lists the capabilities of CN71XX as a recipient of PCIe transaction-layer packets (TLPs) and data-link-layer packets (DLLPs).

PEM capabilities as a PCIe receiver depend on whether the PCIe port is in root complex (RC) or endpoint (EP) mode. As per the PCIe specifications, when a PCIe port is in RC mode, PEM presents a type 1 configuration space header. When a PCIe port is in EP mode, PEM presents a type 0 configuration space header.

22.2.1 Receiving PCIe TLP I/O Space Requests

PCIe ports do not support I/O requests as a target in either EP or RC mode.

In EP mode, the type 0 configuration header contains no I/O space BARs.

In RC mode, the I/O base and limit registers at offsets 0x1C–0xD and 0x30–0x33 of the type 1 configuration header should be disabled. (See PCIERC(0..2)_CFG007, PCIERC(0..2)_CFG012.)

22.2.2 Receiving PCIe TLP Memory Space Requests

As per the PCIe specifications, the PEM PCIe ports do not support PCIe memory-space lock-read operations in EP or in RC modes.

The remainder of this section describes PCIe port responses to normal memory space read and write operations in EP and RC modes.

22.2.2.1 Receiving PCIe Memory space requests as an EP

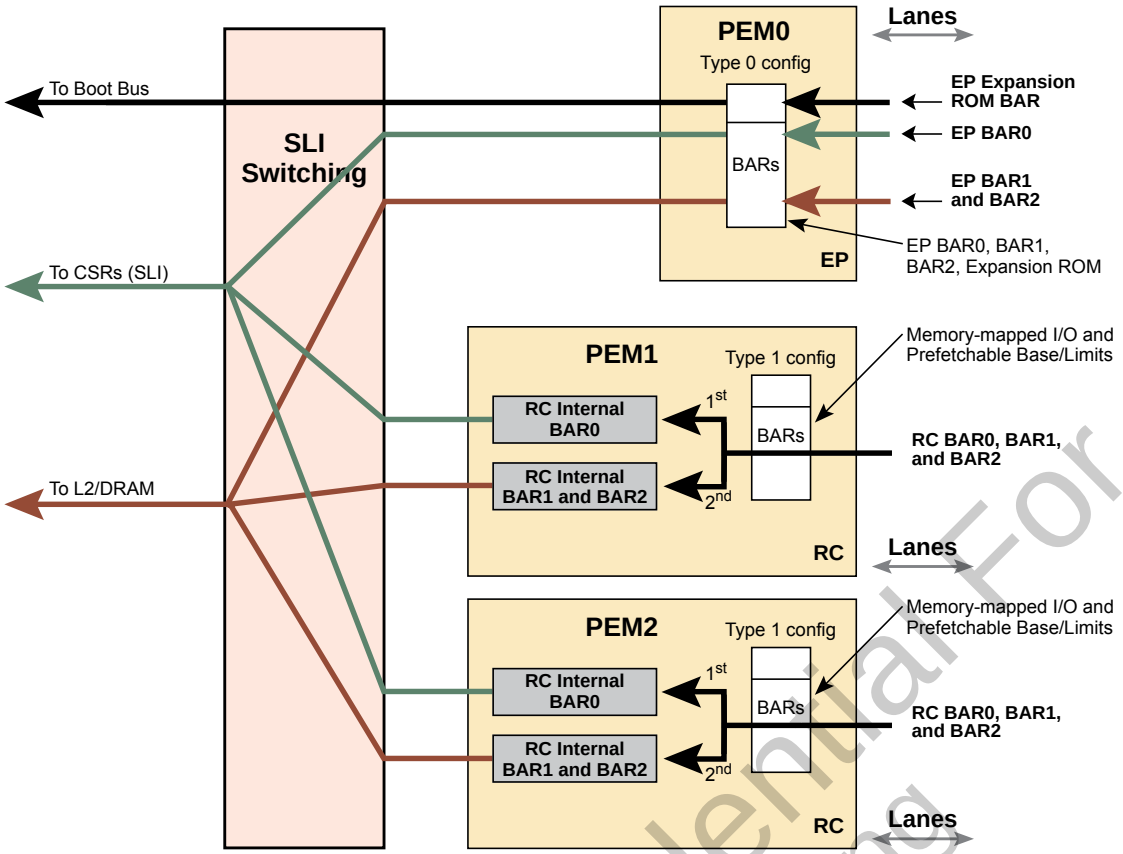
In EP mode, successful PCIe memory space requests either read/write PEM internal CSRs, read/write L2/DRAM, or read the CN71XX boot bus.

As per the PCIe specifications, CN71XX memory space access via the PCIe link is dictated by the base address registers (BARs) in type 0 configuration space when the PCIe port is in endpoint (EP) mode. Refer to Section 22.7.1. PEM implements an EP BAR0, BAR1, BAR2, and expansion ROM BAR.

The top half of Figure 22–2 shows the conceptual path of PCIe memory space requests that successfully enter PEM0 through EP BAR0, EP BAR1, EP BAR2, or the expansion ROM BAR. All requests cross the internal switch and are processed further by the internal logic. The EP BAR0 addresses reach the internal CSRs. The EP BAR1 and EP BAR2 addresses reach the L2/DRAM. The expansion ROM BAR addresses reach the boot bus.

Cavium Confidential For Bowen Yang ALU 02/21/2014

Figure 22–2 Memory Space Requests: PEM0 in EP Mode, PEM1/2 in RC Mode



Note that the EP BAR0, EP BAR1, EP BAR2, and EP Expansion ROM BAR are only present in EP mode and are not present in RC mode. Figure 22–2 shows the case when only PEM0 is in EP mode. It is also possible for all of PEM0, PEM1, and PEM2 to be in RC mode.

EP BAR0 – Memory-Mapped CSR Region

PEM's EP BAR0 PCIe control registers reside at offset 0x10–0x17 in type 0 PCIe configuration space. (See PCIEEP(0..2)_CFG004, PCIEEP(0..2)_CFG005.) EP BAR0 does not exist when the PCIe port is in RC mode.

Prefetchable	1
Type	Locate anywhere within 64-bit address space.
Size	16 KB

EP BAR0 is a 64-bit base-address register of 16 KB that provides access to CN71XX internal CSRs, just like RC internal BAR0 does in RC mode (refer to Section 22.2.2.2). Note that BAR0 CSRs are also SLI CSRs, and can also be addressed via internal addresses. Refer to Section 12.8.7.

EP BAR0 supports a restricted programming model that does not support individual read/write operations that are larger than 8 bytes nor that cross an 8B boundary. (The EP BAR0 CSRs are spaced apart such that this is only possible by accessing nonexistent addresses, so this is generally not a concern.)

Cavium Confidential For Bowen Yang AU 02/27/2014

EP BAR1 – 32-Bit-compatible Memory-Mapped Region

PEM's EP BAR1 PCI control registers reside at offset 0x18–0x1F in type 0 PCIe configuration space (see PCIEEP(0..2)_CFG006, PCIEEP(0..2)_CFG007). EP BAR1 does not exist when the PCIe port is in RC mode.

Prefetchable	1
Type	Locate anywhere within 64-bit address space.
Size	64 MB

EP BAR1 is a 64-bit base-address register of 64MB that provides indirect access to CN71XX L2/DRAM addresses, like the RC internal BAR1 does in RC mode (refer to Section 22.2.2.2). PEM remaps EP BAR1 addresses in 4 MB chunks using 16 PEM_BAR1_INDEX registers, as described below..

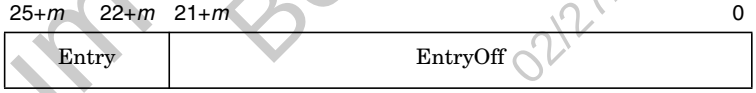
NOTE: Note that EP BAR1 is always present out of reset. After reset, software can change the present/not-present status of EP BAR1 with write operations to PCIEEP(0..2)_CFG006_MASK/PCIEEP(0..2)_CFG007_MASK. (Note that all write operations to the PCIe configuration registers are done indirectly via PEM(0..2)_CFG_WR).

- To make EP BAR1 not present, perform the following steps:
  1. Write PCIEEP(0..2)_CFG007_MASK to 0x0.
  2. Write PCIEEP(0..2)_CFG006_MASK to 0x0.
- To make EP BAR1 present, perform the following steps:
  1. Write PCIEEP(0..2)_CFG006_MASK to 0x00000001.
  2. Write PCIEEP(0..2)_CFG006_MASK to 0x0000000C.
  3. Write PCIEEP(0..2)_CFG006_MASK to 0xFFFFFFFF.
  4. Write PCIEEP(0..2)_CFG007_MASK to 0x0000007F.

The remainder of this BAR1 description applies to both EP BAR1 and RC internal BAR1, and specifies a BAR1 of size 2^{26+m}. EP BAR1 is always 64 MB, so m always equals 0. RC internal BAR1 can be any of 64 MB, 128 MB, 256 MB, 512 MB, 1 GB, or 2 GB, so m can be any of 0..5.

When PEM receives an address in the EP BAR1 range (RC internal BAR1 range in RC mode), it decodes the lower 26+m address bits as follows:

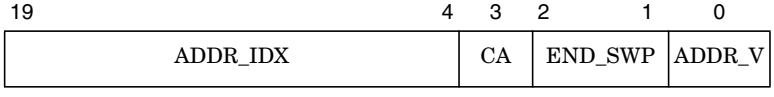
BAR1 26-bit Address Offset



- **Entry** – Selects one of 16 BAR map-table entries.
- **EntryOff** – Offset within the region selected by the BAR1 map table entry.

BAR1 Map-Table Entry Format

There are 16 entries configured via the PEM(0..2)_BAR1_INDEX(0..15) CSRs.

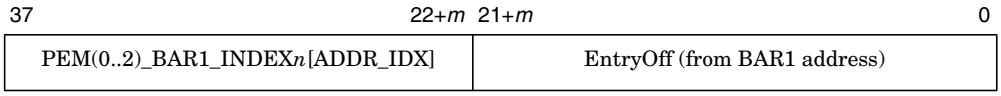


- **CA** – Cache attribute. When set, hardware does not allocate into L2 cache.
- **END_SWP** – Endian-swap mode. Refer to Section 12.7.
- **ADDR_V** – Valid bit. When set, entry is valid. When clear, the entry is invalid.

Cavium Confidential For Bowen Yang ALU 02/27/2014

When the corresponding map table entry is invalid, CN71XX responds “unsupported request” to PCIe read and write operations.

The 38-bit L2/DRAM address referenced by the PCIe read/write operations in BAR1 space is:



Note that the low *m* bits of PEM(0..2)_BAR1_INDEX*n*[ADDR_IDX] are not used by PEM when it creates the resultant L2/DRAM address. Section 12.4 details the response of PEM/SLI to L2/DRAM references as a target.

EP BAR2 – 64-bit Memory-Mapped Region

PEM’s BAR2 PCI control registers reside at offset 0x20–0x27 in type 0 PCIe configuration space (see PCIEEP(0..2)_CFG008, PCIEEP(0..2)_CFG009). EP BAR2 does not exist when the PEM port is in RC mode.

Note that the PEMs’ EP BAR2 is always present and sized to 1MB at reset. But they can advertise a size capability of 512GB (which conforms with the PCIe specification), or a size capability of 2TB (which exceeds the PCIe specification). This is necessary as some PCI discovery sequence firmware versions (running on remote hosts) improperly map BARs when the non-conforming PEM EP BAR2 is advertised.

NOTE: When an EEPROM is not attached, MIO_FUS_DAT3[BAR2_SZ_CONF] determines whether EP BAR2’s advertised size capability conforms to the PCIe specification out of reset.

When an EEPROM is attached in EP mode, the EEPROM PCIEEP(0..2)_CFG008_MASK/ PCIEEP(0..2)_CFG009_MASK values determine whether EP BAR2 is present out of reset.

After reset, software can change the present/not-present status of EP BAR2 with write operations to PCIEEP(0..2)_CFG008_MASK/PCIEEP(0..2)_CFG009_MASK. (Note that all write operations to the PCIe configuration registers are done indirectly via PEM(0..2)_CFG_WR.)

- To make EP BAR2 not present, perform the following steps:
  1. Write PCIEEP(0..2)_CFG009_MASK to 0x0.
  2. Write PCIEEP(0..2)_CFG008_MASK to 0x0.
- To make EP BAR2 present, perform the following steps:
  1. Write PCIEEP(0..2)_CFG008_MASK to 0x00000001.
  2. Write PCIEEP(0..2)_CFG008 to 0x0000000C. (Via PEM*_CFG_WR, NOT via config space writes!)
  3. Write PCIEEP(0..2)_CFG008_MASK to 0xFFFFFFFF.
  4. Write PCIEEP(0..2)_CFG009_MASK to 0x000001FF.

In order for BAR2 to be used, it must be both present and enabled. Refer to the PEM(0..2)_BAR_CTL[BAR2_ENB] description.

The discussion in the remainder of this section assumes that EP BAR2 is present.

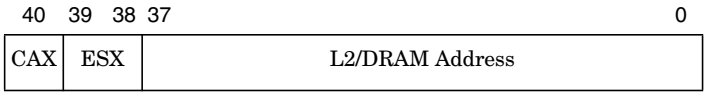
Prefetchable	1
Type	Locate anywhere within 64-bit address space.
Size	2^{39} bytes

EP BAR2 provides direct access to CN71XX L2/DRAM, just like RC internal BAR2 does in RC mode. Refer to Section 22.2.2.2.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PEM interprets the lower 41 bits of the PCIe address as follows in BAR2:

BAR2 41-bit PCIe Addressing



- **CAX** – After XORing with a CSR default for physical PEM port *n* (PEM*n*_BAR_CTL[BAR2_CAX]), determines the L2 cache attribute.

  If CAX ⊕ PEM*n*_BAR_CTL[BAR2_CAX] = 1, load/store operations are not allocated to L2 cache.

- **ESX** – After XORing with a CSR default for physical PCIe port *n* (PEM*n*_BAR_CTL[BAR2_ESX]), determines the endian-swap mode.
- **L2/DRAM Address** – PEM L2/DRAM address.

There is a CSR enable for physical PCIe port *n* (PEM*n*_BAR_CTL[BAR2_ENB]) for BAR2. When BAR2 is disabled, PEM returns "Unsupported Request" in response to BAR2 read and write operations.

Section 12.4 details the response of PEM/SLI to L2/DRAM references as a target.

EP Expansion ROM BAR

PEM supports a PCI expansion ROM of 64 KB controlled at offset 0x30–0x33 in the type 0 configuration space header (see PCIEEP(0..2)_CFG012). EP Expansion ROM BAR does not exist when the PCIe port is in RC mode. The EP Expansion ROM BAR is disabled in PEM1, so can only be used with PEM0 when it is in EP mode.

CN71XX forwards these expansion ROM read operations to its boot bus. See Chapter 25.

22.2.2.2 Receiving PCIe Memory Space Requests as an RC

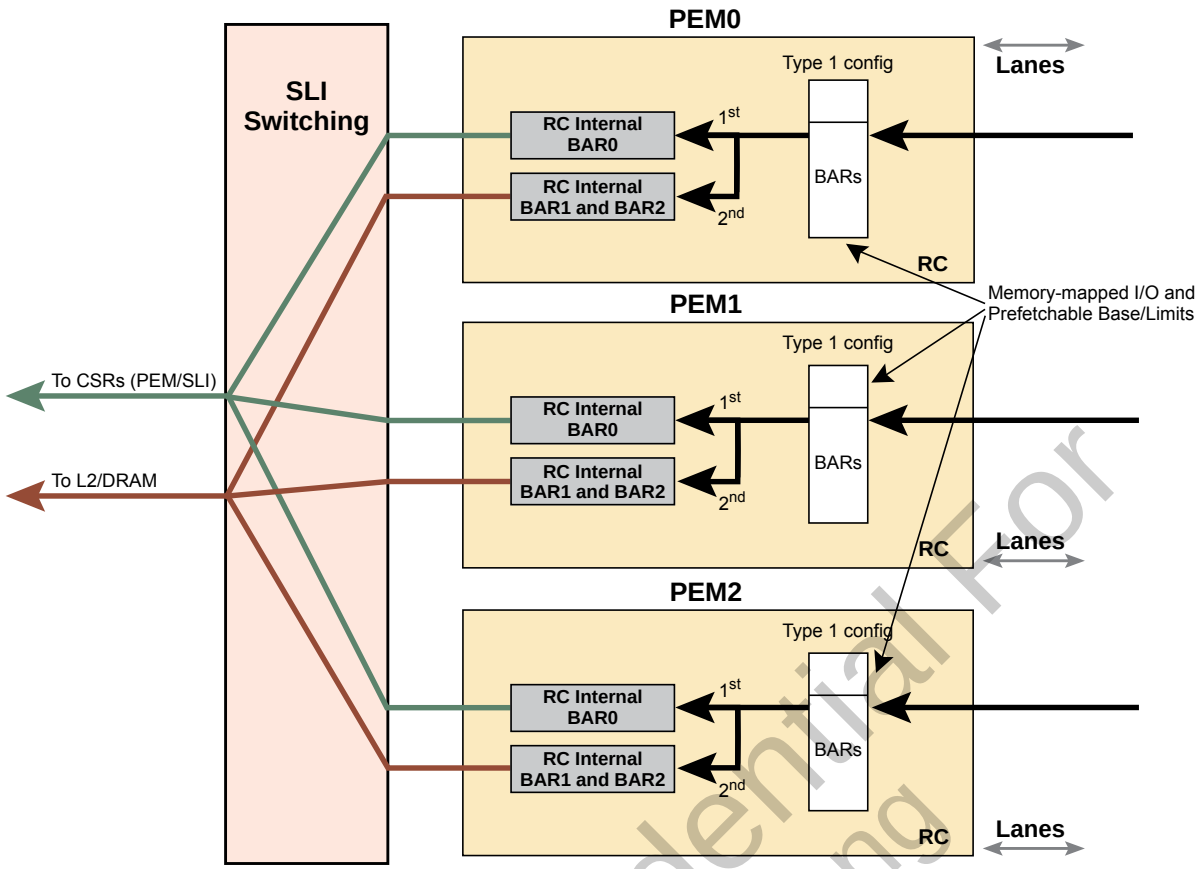
In RC mode, successful PCIe memory space requests either are forwarded to read/write CN71XX internal CSRs or read/write CN71XX L2/DRAM.

As per the PCIe specifications, PEM implements a type 1 PCIe configuration space for each port that is in root complex (RC) mode. The PEM address processing is affected by the internal BARs. PEM implements three internal BARs—BAR0, BAR1, and BAR2—used to access internal CSRs and L2/DRAM.

Figure 22–3 and the bottom half of Figure 22–2 conceptually show the path for PCIe memory space requests that successfully enter PEM.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 22–3 Memory Space Requests When PEM Ports are in RC Mode



Note the conceptual processing order for PCIe memory space read/write requests implied by Figure 22–2 and Figure 22–3:

- The read/write request first passes through the memory-mapped I/O or prefetchable base/limit registers in the type 1 configuration space header. These BARs are normally disabled, so all requests pass through them.
- PEM compares the address to its RC internal BAR0, BAR1, and BAR2 in order. The address ranges covered by these BARs may overlap amongst each other; PEM processes the address according to the first match.

Note that the base/limit registers in the type 1 configuration header are not present in EP mode. None of the RC internal BAR0, RC internal BAR1, and RC internal BAR2 are used by EP mode requests.

RC Type 1 Base Address Registers

PEM does not use the base address registers in the type 1 configuration header, and offsets 0x10–0x17 (see PCIERC(0..2)_CFG004, PCIERC(0..2)_CFG005) in the type 1 configuration header read as zero.

RC Type 1 Memory Mapped I/O Base and Limit Registers

PEM implements these registers at offsets 0x20–0x23 in the type 1 configuration header in RC mode (see PCIERC(0..2)_CFG008). RC type 1 memory mapped I/O base and limit registers do not exist when the PEM is in EP mode.

PEM terminates a read/write that has an address in the memory-mapped I/O range as an unsupported request. A typical application will disable the RC type 1 memory mapped base and limit registers by setting the base above the limit.

Cavium Confidential For Bowen Yang AEU 02-27-2019

RC Type 1 Prefetchable Memory Base and Limit Registers

PEM implements these registers at offsets 0x24–0x2F in the type 1 configuration header in RC mode (see PCIERC(0..2)_CFG009, PCIERC(0..2)_CFG010). RC type 1 prefetchable memory base and limit registers do not exist when the PEM is in EP mode.

PEM terminates a read/write operation that has an address in the prefetchable range as an unsupported request. A typical application will disable the RC type 1 prefetchable memory mapped base and limit registers by setting the base above the limit.

RC Type 1 Expansion ROM

PEM does not implement the expansion ROM BAR in the type 1 configuration header. Offsets 0x38–0x3B in the type 1 configuration header read as zero (see PCIERC(0..2)_CFG014).

RC Internal BAR0

RC internal BAR0 is a 50-bit internal CSR (PEM(0..2)_P2N_BAR0_START) that specifies the location of a naturally-aligned 16 KB region in the 64-bit PCIe memory space. This region provides access to CN71XX internal CSRs, just like EP BAR0 does in EP mode (refer to Section 22.2.2.1). RC internal BAR0 is not used when the PEM port is in EP mode. There is a separate RC internal BAR0 for each physical PEM port.

BAR0 CSRs are also SLI CSRs, and may also be accessed via internal addresses. Refer to Section 12.8.7.

Note that RC internal BAR0 may overlap with the RC internal BAR1 and BAR2. As Figure 22–2 and Figure 22–3 indicate, RC internal BAR0 is ordered before RC internal BAR1 and BAR2, so will take precedence over them.

RC Internal BAR1

RC internal BAR1 is a 38-bit internal CSR (PEM(0..2)_P2N_BAR1_START) that specifies the location of a naturally-aligned 64 MB–2 GB region in the 64-bit PCIe memory space. This region provides indirect access to CN71XX L2/DRAM addresses, like the EP BAR1 does in EP mode (refer to Section 22.2.2.1). RC internal BAR1 is not used when the PEM port is in EP mode. There is a separate RC internal BAR1 for each physical PEM port.

The address translation description for EP BAR1 in Section 22.2.2.1 also applies to RC internal BAR1. PEM(0..2)_BAR_CTL[BAR1_SIZ] selects the size of RC internal BAR1. Referring to the EP BAR1 description in Section 22.2.2.1, m = PEM(0..2)_BAR_CTL[BAR1_SIZ] – 1. PEM breaks the 2^{26+m} byte RC internal BAR1 address range into 16 chunks of 2^{22+m} bytes each. Each chunk ranges from 4 MB (m = 0) up to the largest possible 128 MB (m = 5).

Note that RC internal BAR1 may overlap with RC internal BAR0 and BAR2. RC internal BAR1 is ordered after RC internal BAR0, so BAR0 takes precedence when there is overlap, as Figure 22–2 and Figure 22–3 indicate. RC internal BAR2 is ordered after, so RC internal BAR1 takes precedence when there is overlap.

RC Internal BAR2

RC internal BAR2 is a 23-bit internal CSR (PEM(0..2)_P2N_BAR2_START) that specifies the location of a naturally-aligned 2^{41} byte region in the 64-bit PCIe memory space. This region provides direct access to CN71XX L2/DRAM addresses, like the EP BAR2 does in EP mode (refer to Section 22.2.2.1). RC internal BAR2 is not used when the PEM port is in EP mode. There is a separate RC internal BAR2 for each physical PEM port.

Note that RC internal BAR2 may overlap with RC internal BAR0 and BAR1. As Figure 22–2 and Figure 22–3 indicate, these other internal BARs are ordered first, so they take precedence when overlap occurs.

Cavium Confidential PDF
Boren Yang
02/27/2014

22.2.3 Receiving PCIe TLP Configuration-Space Requests

PEM receives and processes type 0 configuration read/write operations only when its port is in EP mode. PEM always terminates them. PEM neither receives nor processes type 1 configuration read/write requests. PEM never returns a configuration request retry status in response to a configuration read request.

On receipt of a type 0 configuration write in EP mode, PEM updates PEM(0..2)_CTL_STATUS[PBUS, DNUM] with the bus number and device numbers contained in any configuration write address. These become the bus number and device number contained in the Requestor ID and Completer ID for outgoing messages in EP mode, as per the PCIe specifications.

Note that in RC mode, PEM(0..2)_CTL_STATUS[PBUS] is a read-only copy of PCIERC(0..2)_CFG006[PBNUM]. This becomes the outgoing bus number contained in the Requestor ID and Completer ID for outgoing requests in RC mode. The function number for requests initiated in RC mode is always zero in RC mode.

22.2.4 Receiving PCIe TLP Messages

22.2.4.1 Routing of Received TLP Messages

All messages that arrive at PEM are terminated at PEM.

22.2.4.2 Receiving INTX Interrupt Signaling Messages

PEM receives interrupt assertion/deassertion messages as per the PCIe specifications for EP and RC.

In EP mode, PEM does not receive interrupt assertion/deassertion messages.

In RC mode, PEM receives interrupt assertion/deassertion messages.

Section 12.6 describes the PEM response to INTx messages.

22.2.4.3 Receiving Power Management Messages

PEM implements the power management messages as per the PCIe specifications for RC and EP modes.

22.2.4.4 Receiving Error Signaling Messages

PEM implements the error signaling messages as per the PCIe specifications for RC and EP modes.

In EP mode, PEM does not receive error messages.

In RC mode, PEM receives and processes error messages as per the PCIe specifications with advanced error reporting implemented.

22.2.4.5 Receiving Locked Transactions Messages

PEM does not support unlocked messages.

22.2.4.6 Receiving Slot Power Limit Messages

PEM implements the slot power limit messages as per the PCIe specifications for RC and EP modes.

Cavium Confidential For Bowen Yang ALU 02/27/2014

22.2.4.7 Receiving Vendor_Defined Messages

PEM does not support Vendor_Defined messages.

22.2.4.8 Receiving Ignored messages

PEM ignores the 1.0a messages as per the PCIe 3.0 specifications.

22.2.5 Receiving PCIe TLP Completions

22.2.5.1 Completion ID Routing

When any PEM port is in EP mode, and when all three ports are in RC mode, all completions that arrive at PEM are terminated at PEM.

22.2.5.2 Receiving Successful Completion

Successful completions that terminate at PEM are the normal completion of prior I/O read, configuration read, memory read, I/O write, or configuration write requests. PEM handles these as per the PCIe specifications.

22.2.5.3 Receiving Unsupported Request

Unsupported request (UR) completions that terminate at PEM are normal for configuration-read operations during PCIe discovery. In all other conditions, they are typically an error.

Upon receipt of a UR, PEM internally synthesizes all 1s as the response to the core or DMA engine that initiated the read operation. PEM also sets PEM(0..2)_DBG_INFO[RACUR] to 1. (If a DPI DMA engine initiated the read operation, SLI_INT_SUM[SPRT*_ERR] is also set to 1.)

22.2.5.4 Receiving Configuration Request Retry Status

Configuration request retry status (CRS) messages always terminate at PEM, and they may be a normal response for configuration read operations during PCIe discovery.

PEM does not implement CRS software visibility.

If PEM(0..2)_CTL_STATUS[CFG_RTRY] ≠ 0, then PEM retries a CRS response as long as the timeout interval (PEM(0..2)_CTL_STATUS[CFG_RTRY] × 0x10000 coprocessor-clock cycles) has not expired. If the timeout expires without successful completion, CN71XX cores receive all 1s as the result and PEM(0..2)_INT_SUM[CRS_ER] is set to 1. There must be only one outstanding configuration read operation whenever PEM receives a CRS with this mode enabled.

If PEM(0..2)_CTL_STATUS[CFG_RTRY] = 0, then CN71XX cores receive all 1s as the result upon receipt of a CRS response to a configuration read operation and PEM(0..2)_INT_SUM[CRS_DR] is set to 1.

22.2.5.5 Receiving Completer Abort

Completer abort (CA) completions that terminate at PEM are abnormal.

Cavium Confidential For Bowen Yang ALU 2/27/2014

Upon receipt of a CA, PEM internally synthesizes all 1s as the response to the core or DMA engine that initiated the read operation. PEM also sets PEM(0..2)_DBG_INFO[RACCA] to 1. (If a DPI DMA engine initiated the read operation, SLI_INT_SUM[SPRT*_ERR] is also set to 1.)

22.2.6 Receiving PCIe DLLP Packets

PEM implements all DLLP packets as per the PCIe specifications, with the following exceptions:

- PEM does not implement vendor-specific packets.
- PEM only implements one VC.

22.3 PEM as a PCIe Transmitter

This section lists the capabilities of PEM as a transmitter of PCIe transaction-layer packets (TLPs) and data-link-layer packets (DLLPs).

When PEM initiates a TLP, the transaction ID consists of the fields shown in Table 22–1.

Table 22–1 Transaction ID Fields

	Bus Number	Device Number	Function Number	Tag
Initiated via PEM in EP Mode	PEM_CTL_STATUS[PBUS] Section 22.2.3.	PEM_CTL_STATUS[DNUM] Section 22.2.3.	Zero always.[1]	Always in the range 0–31.[2]
Initiated via PEM*n*[3] in RC Mode	PCIERC*n*_CFG006[PBNUM]	Zero.		

1. PEM never uses phantom functions.
2. PEM never uses extended tags.
3. *n* can be either 0, 1 or 2.

22.3.1 Transmitting PCIe TLP I/O Space Requests

PEM can initiate PCIe I/O space read and write operations on core load and store operations. Refer to Section 12.8.5.

22.3.2 Transmitting PCIe TLP Memory Space Requests

PEM can initiate PCIe memory-space read and write operations on core load, IOBDMA, and store operations. Refer to Section 12.8.6.

PEM can also initiate PCIe memory-space read and write operations from DPI packet-input, packet-output, and DMA engines. Refer to Sections 12.1, 12.2, and 12.3.

PEM cannot initiate PCIe memory read-locked requests.

22.3.3 Transmitting PCIe TLP Configuration Space Requests

PEM can initiate PCIe configuration space read and write operations on core load and store operations. Refer to Section 12.8.4.

Cavium Confidential For Bowen Yang ALU 02/27/2014

22.3.4 Transmitting PCIe TLP Messages

- **INTX interrupt signaling messages**. PEM transmits interrupt assertion/deassertion messages as per the PCIe specifications for EP and RC modes.

  In EP mode, PEM transmits interrupt assertion/deassertion messages. In RC mode, PEM does not transmit interrupt assertion/deassertion messages.
- **Power management messages**. PEM implements the power management messages as per the PCIe specifications for RC and EP modes.
- **Error signaling messages**. PEM implements the error signaling messages as per the PCIe specifications for RC and EP modes.

  In EP mode, PEM transmits error messages as per the PCIe specifications with advanced error reporting implemented. In RC mode, PEM does not transmit any error messages.
- **Locked transactions messages**. PEM does not support unlocked messages.
- **Slot power limit messages**. PEM implements the slot power limit messages as per the PCIe specifications for RC and EP modes.
- **Vendor_Defined messages**. PEM does not support Vendor_Defined messages.
- **Ignored messages**. PEM does not transmit the 1.0a messages that should be ignored, as per the PCIe 3.0 specifications.

22.3.5 Transmitting PCIe TLP Completions

- **Successful completion**. PEM transmits successful completions as the normal response to PCIe memory requests that terminate at PEM, and can also generate a successful completion in response to a type 0 configuration request when PEM is in EP mode.

  When PEM initiates memory-space read completions, it always splits the completions on 128-byte naturally aligned boundaries. This is true regardless of the maximum payload size (PCIE*_CFG030[MPS]).
- **Unsupported request**. PEM can generate unsupported request completions when it terminates the following requests:
  - I/O read/write operations.
  - Memory space read/write operations that do not match any of BAR0, BAR1, or BAR2.
  - Type 1 configuration read/write operations
  - Type 0 configuration read/write operations when PEM is in RC mode
- **Configuration request retry status**. PEM does not generate configuration-request retry-status completions.
- **Completer abort**. PEM does not generate completer aborts.

Cavium Confidential For Bowen Yang ALU 02/27/2014

22.3.6 Transmitting PCIe DLLP Packets

PEM implements all DLLP packets as per the PCIe specification, with the following exceptions:

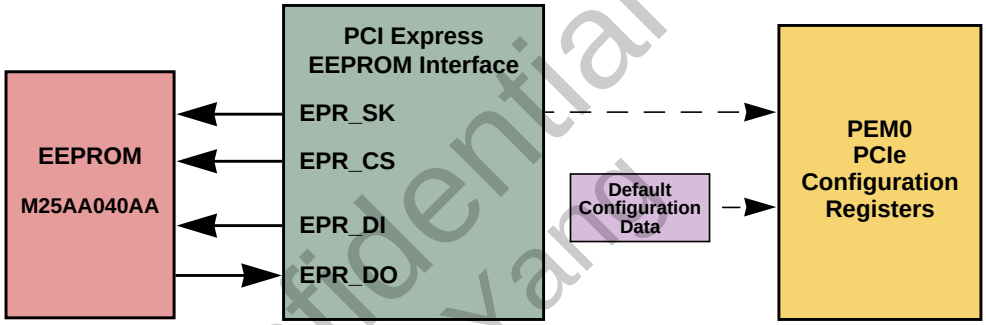
- PEM does not implement vendor-specific packets.
- PEM only implements one VC.

22.4 PEM0 Configuration-EEPROM Interface

The PEM0 block provides an interface to an optional external serial EEPROM that can be used to store PCIe configuration data (shown in Figure 22–4). Note that an EEPROM is never read in host mode (i.e. when PEM(0..2)_CFG[HOSTMD] = 1).

Upon PEM0 power-up in endpoint mode, system configuration data is read from the EEPROM into the specified endpoint-mode configuration registers. The presence of this external EEPROM is detected automatically; if no EEPROM is connected, default values for the CSRs are used (see Section 22.7) and the ID fields are set to their default values. The CSRs are read-only from the perspective of the host CPU.

Figure 22–4 Configuration EEPROM Operation



The EEPROM provides the flexibility to program any number of configuration registers, from one up to the maximum capability of the EEPROM (16, 32 or 64, depending on which device is used). The EEPROM is organized as eight-byte entries, each of which can program one 32-bit configuration register. Each eight-byte entry consists of the elements shown in Figure 22–5.

Figure 22–5 EEPROM Entry

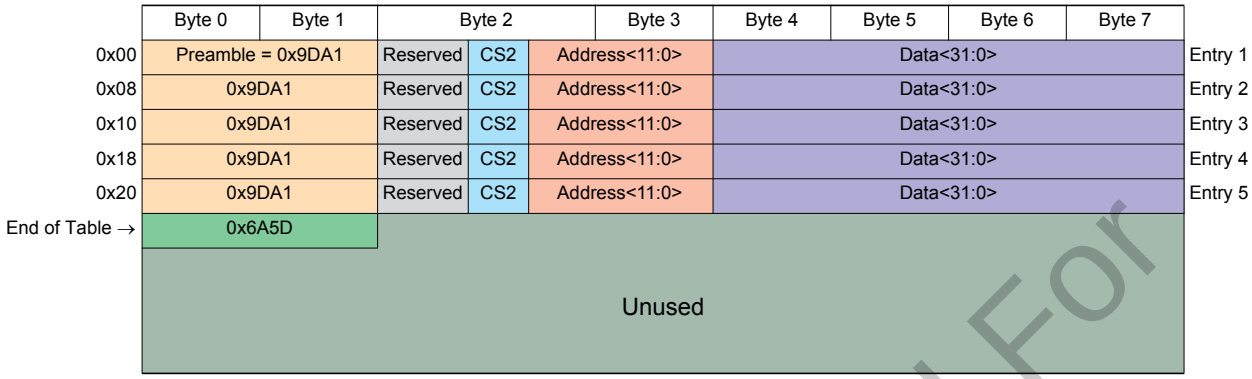
Byte 0		Byte 1		Byte 2			Byte 3		Byte 4		Byte 5		Byte 6		Byte 7	
63	56	55	48	47	45	44	43	40	39	32	31	24	23	16	15	8	7	0
preamble				rsvd		cs2	address				data					

- **preamble** = 0x9DA1
- **cs2** = specifies bit<31> of the address written by hardware (refer to Column 3 of Table 22–3):
  1 = configuration mask register, 0 = configuration register
- **address** = specifies bits<11:0> of the address written by hardware (refer to Column 3 of Table 22–3). Note that bits<30:12> of this address are all 0s.
- **data** = 32 bits

Cavium Confidential PDF
Bowen Yang
ZJU
02/27/2014

The first EEPROM entry is required to start at offset zero. Entries are accessed using the EEPROM page mode, and are read sequentially until an end-of-table value (0x6A5D) is found in place of a preamble. By providing the address of the configuration register in the EEPROM, you specify exactly which configuration registers are to be modified from the default values. Figure 22–6 shows the organization of the EEPROM with an example of a five-entry table.

Figure 22–6 EEPROM Example with Five-Entry Table



If the preamble field of the first entry is all 1s (0xFFFF), either no EEPROM is connected or the EEPROM is programmed to not modify any configuration registers. In both cases, reading from the EEPROM ceases. The EEPROM serial-clock signal (EPR_SK) from the interface to the EEPROM runs until loading of data from the EEPROM is complete. It then automatically shuts down to save power, and does not start again unless CN71XX is reset.

The EEPROM interface supports a standard type (128/256/512 words × 8 bits) serial EEPROM. Devices of this type can be obtained from several semiconductor vendors. Note that the CN71XX runs these devices at 2MHz.

22.4.1 EEPROM Programming

The EEPROM contents can programmed or verified via a pair of registers implemented in PEM register space: PEM(0..2)_SPI_CTL and PEM(0..2)_SPI_DATA.

Cavium Confidential For Bowen Yang ALU 02/27/2014

22.4.1.1 PEM(0..2)_SPI_CTL

PEM(0..2)_SPI_CTL provides a facility to control the reading or writing of the EEPROM.

- EEPROM address[8:0] (PEM(0..2)_SPI_CTL[ADR]): specifies the byte address offset for the serial EEPROM access. Since accesses are eight-byte aligned entries, the three least-significant bits (LSBs) are always treated as 0s.
- Command[2:0] (PEM(0..2)_SPI_CTL[CMD]): the commands are the following:

  0x0 = reserved

  0x1 = WRSR: write status register, a single-byte write access using PEM(0..2)_SPI_DATA[DATA< 7:0>]

  0x2 = WRITE: an eight-byte page-mode write access to the memory array using PEM(0..2)_SPI_DATA

  0x3 = READ: an eight-byte page-mode read access from the memory array using PEM(0..2)_SPI_DATA

  0x4 = WRDI: clear the write-enable latch (i.e. write protect the device)

  0x5 = RDSR: read status register, a single-byte read access using PEM(0..2)_SPI_DATA[DATA< 7:0>]

  0x6 = WREN: set the write-enable latch (i.e. allow writes to occur)

  0x7 = reserved
- Table valid (PEM(0..2)_SPI_CTL[TVALID]): set if at least one valid entry was read (preamble was valid) from the EEPROM, and a configuration register was updated.
- Start/busy status (PEM(0..2)_SPI_CTL[START_BUSY]): when written, indicates to the controller to start a SPI transaction. The bit is cleared by the controller when the write/read operation has completed. If the operation was a read, the contents of PEM(0..2)_SPI_DATA are valid once this bit is cleared.

22.4.1.2 PEM(0..2)_SPI_DATA

PEM(0..2)_SPI_DATA holds the most recent EEPROM read or write data.

- Read data is valid after PEM(0..2)_SPI_CTL[START_BUSY] has been cleared.
- Write data has been written after PEM(0..2)_SPI_CTL[START_BUSY] has been cleared.

The format of PEM(0..2)_SPI_DATA matches the EEPROM entry shown in Figure 22–5.

- bits<63:48> – PREAMBLE
  - valid entry =0x9DA1
  - end-of-table = 0x6A5D
  - null = 0xFFFF
- bits<47:45> reserved
- bit<44> CS2 – to access write protected configuration registers.
- bits<43:32> – address
- bits<31:0> – read or write data

Cavium Confidential For Bowen Yang ALU 02/27/2014

22.4.1.3 Writing the EEPROM via the PEM CSR Interface:

Perform the following steps to write to the EEPROM.

1. Set the write-enable latch prior to programming.
2. The register interface supports eight-byte-aligned entry write operations for EEPROM data write commands. The entire eight-byte entry is written in a single operation. Software first loads PEM(0..2)_SPI_DATA with valid preamble, address, CS2 state and data. A subsequent write operation to PEM(0..2)_SPI_CTL sets the double-word-aligned address, the write command, and the start bit.
3. Write status register operations use bits <7:0> of PEM(0..2)_SPI_DATA for the data. A subsequent write operation to PEM(0..2)_SPI_CTL sets the write command, and the start bit. The address field is not used.
4. When a write or write status operation is complete and the SROM interface is ready for another operation, the Start/Busy bit will be clear.

22.4.1.4 Reading the EEPROM via the PEM CSR Interface:

Perform the following steps to read the EEPROM.

1. The register interface supports eight-byte-aligned entry read operations for EEPROM data-read commands. The entire eight-byte entry is read in a single operation. A write operation to PEM(0..2)_SPI_CTL sets the double-word-aligned address, the read command, and the start bit.
2. Status register read operations are initiated when software writes PEM(0..2)_SPI_CTL with the read-status command and the start bit. The address field is not used.
3. When a read or read-status operation is complete, PEM(0..2)_SPI_CTL[START_BUSY] is clear. The valid data is in PEM(0..2)_SPI_DATA at this time.

See Section 22.7.1 for the EP mode configuration registers that are available to be modified via the PEM0 EEPROM. The corresponding tables for each configuration register identify the fields in the registers that are writable and provide the reset value for those fields. You can modify any writable field as required. Any writable fields that are not to be modified must be loaded with the CN71XX reset value.

Cavium Confidential For Bowen Yang ALU 02/2/12 exp

22.5 PCIe Reset Sequences

Each PCIe port can be reset independently. This section describes the fundamental-reset, soft-reset, and link-down-reset sequences for an individual PCIe port.

The reset sequences vary depending on whether the port is in host mode (RC mode) or non-host mode (EP mode). The mode is specified by RST_CTL(0..2)[HOST_MODE]:

1 = host/RC mode, 0 = non-host/EP mode.

The RST_CTL*n* register is initialized during a cold reset using the external pins PCIE0_MODE<1:0> to the values shown in Table 22–2. (For more information on RST_CTL*n*, see Section 14.2.) The reset values show the configuration flexibility between host and non-host modes.

Table 22–2 MIO_RST_CNTL*n* Reset Settings

	RST_CTL0				
	[PRST_LINK]	**[RST_LINK]**	**[RST_DRV]**	**[RST_RCV]**	**[RST_CHIP]**
PCIE0_MODE<1:0> = 0x3 (RC Mode)	0	0	1 (drive PERST*n*_L)	0	0
PCIE0_MODE<1:0> ≠ 0x3 (EP Mode)	0	1 (reset chip on link down)	0	1 (receive PERST*n*_L)	0

22.5.1 Fundamental PCIe Reset Sequences

22.5.1.1 Fundamental PCIe Reset Sequence in RC Mode

A CN71XX PCIe port acts as a root complex (RC) when RST_CTL*n*[HOST_MODE] = 1.

While RST_SOFT_PRST*n*[SOFT_PRST] = 1 or PEM(0..2)_ON[PEMON] = 0:

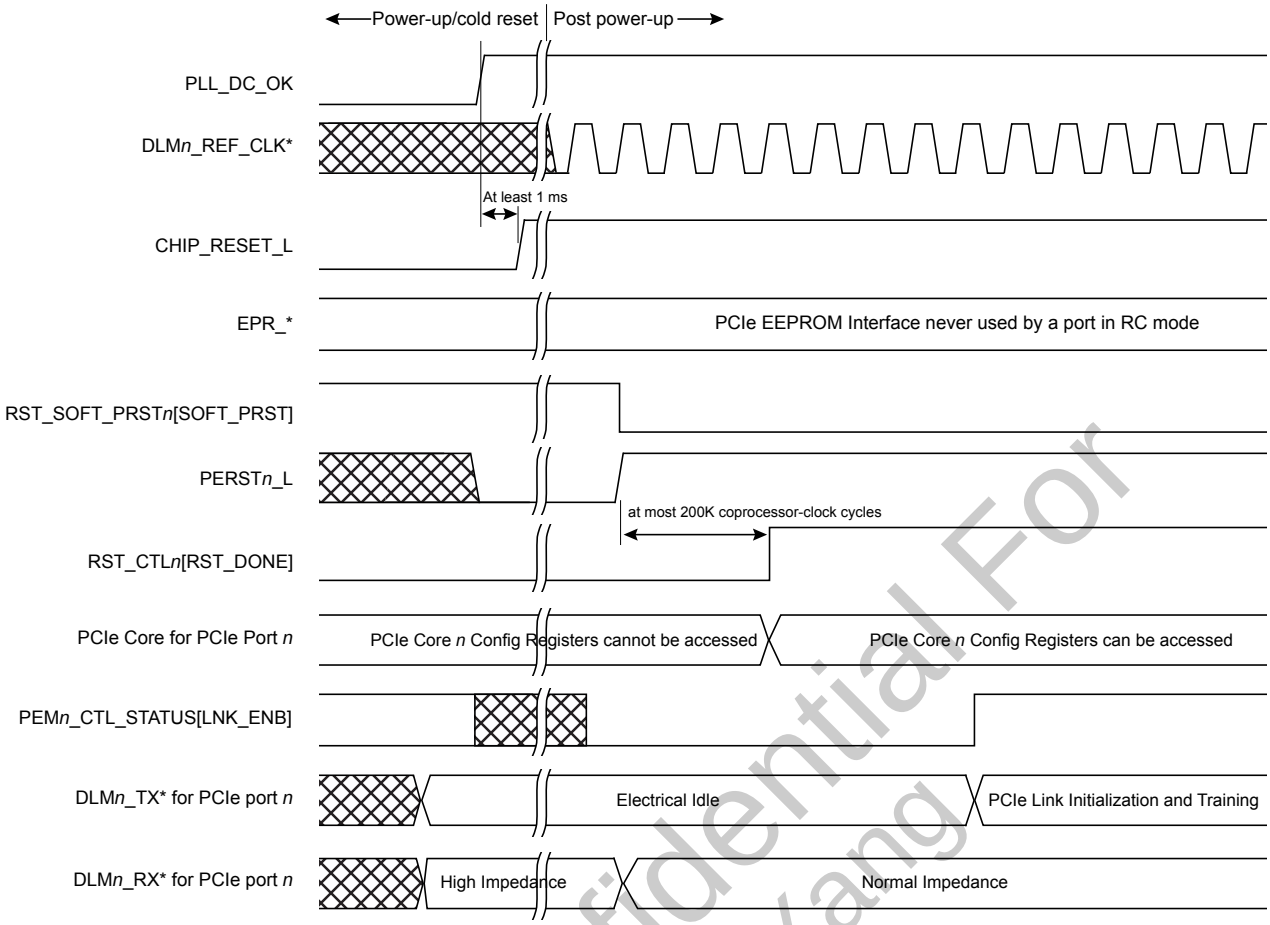
- the PCIe core logic inside CN71XX is in reset
- RST_CTL*n*[RST_DONE] reads as 0 to indicate the reset status
- the SerDes receive lanes attached to port *n* have high impedance (approximately 200 kΩ)
- the SerDes transmit lanes attached to port *n* transmit electrical idle.

Additionally, if RST_CTL*n*[RST_DRV] =1, CN71XX drives PERST1_L low.

Figure 22–7 shows the reset/bringup sequence when a PCIe port is in RC mode. In this case, RST_SOFT_PRST1[SOFT_PRST] resets to a 1, so the PCIe port is in reset while CN71XX boots up. CN71XX software must not access any PCIERC*n*_CFG* registers whenever RST_SOFT_PRST1[SOFT_PRST] = 1. (PCIERC*n*_CFG* registers are accessed via PEM*n*_CFG_RD and PEM*n*_CFG_WR.) Before bringing PCIe port *n* out of reset, software should disable the link by writing PEM*n*_CTL_STATUS[LNK_ENB] = 0. Eventually, CN71XX software transitions RST_SOFT_PRST*n*[SOFT_PRST] from 1 to 0 to take the PCIe port out of reset in RC mode.

Cavium Confidential For Bowen Yang ALU 8/27/2014

Figure 22–7 PCIe Port Reset Sequence in RC Mode



To bring a PCIe port *n* out of reset, CN71XX software should write both PEM(0..2)_ON[PEMON] = 1 and RST_SOFT_PRST*n*[SOFT_PRST] = 0. Once CN71XX software deasserts [SOFT_PRST], CN71XX hardware immediately drives PERST*n*_L high (if RST_CTL*n*[RST_DRV] = 1). However, the CN71XX-internal PCIe core remains in reset while BIST is performed. CN71XX software must wait until RST_CTL*n*[RST_DONE] = 1, indicating BIST has completed and the CN71XX-internal PCIe core is no longer in reset. After RST_CTL*n*[RST_DONE] asserts, software can read PEM*n*_BIST_STATUS* to determine the results of the BIST.

After RST_CTL*n*[RST_DONE] asserts, software should configure the CN71XX-internal PCIe core. If desired, much configuration can be done at this time. But at a minimum, prior to enabling the link, PCIERC*n*_CFG452[LME] must be configured with the correct width (×4, ×2, or ×1). PCIERC*n*_CFG452[LME] can remain at its reset value (0x7 = ×4), or it can be set to a smaller value to limit the number of lanes the PCIe core will use.

Once software completes its configuration of the CN71XX-internal PCIe core in RC mode, it must set PEM*n*_CTL_STATUS[LNK_ENB] = 1. This causes CN71XX to initiate the PCIe link initialization and training sequence on the SerDes. If all goes well, PCIERC*n*_CFG032[DLLA] asserts to indicate that the link is in the DL_Active state, and PCIERC*n*_CFG032[NLW] indicates the negotiated link width. At that point, software should complete any further initialization of the PCIe core and other CN71XX PCIe logic (refer to Section 22.6). Then it can initiate the PCIe discovery sequence (if desired) and start normal operation.

Cavium Confidential For Bowen Yang 02/01/2018

Eventually, PCI-host software may later transition RST_SOFT_PRST*n*[SOFT_PRST] from 0 to 1. This reassertion of [SOFT_PRST] causes CN71XX to drive PERST*n*_L low (if RST_CTL*n*[RST_DRV] = 1). Following the [SOFT_PRST] reassertion, CN71XX PCI-host software should read RST_SOFT_PRST*n*, and wait for the result to return, before performing any other operations. As mentioned above, PCIERC*n*_CFG*n* registers cannot be accessed when RST_SOFT_PRST*n*[SOFT_PRST] is set to 1. Note that DLM*n*_REF_CLK*[1] needs to be stable only while RST_SOFT_PRST*n*[SOFT_PRST] = 1. DLM*n*_REF_CLK* does not need to be stable when RST_SOFT_PRST*n*[SOFT_PRST] = 0.

22.5.1.2 Fundamental PCIe Reset Sequence in EP Mode

The CN71XX PCIe port 0 acts as an endpoint (EP) when RST_CTL0[HOST_MODE] = 0. Only PCIe port 0 uses the PCIe EEPROM port, and only when in EP mode.

While RST_CTL0[RST_RCV] = 1 and PERST0_L is received low:

- the PCIe core logic inside CN71XX is in reset
- RST_CTL0[RST_DONE] reads as 0 to indicate the reset status
- the SerDes receive lanes attached to port *n* have high impedance (approximately 200 kΩ)
- the SerDes transmit lanes attached to port 0 transmit electrical idle.

Additionally, if RST_CTL0[RST_CHIP] = 1, PERST0_L resets all of CN71XX, the same way as CHIP_RESET_L would.

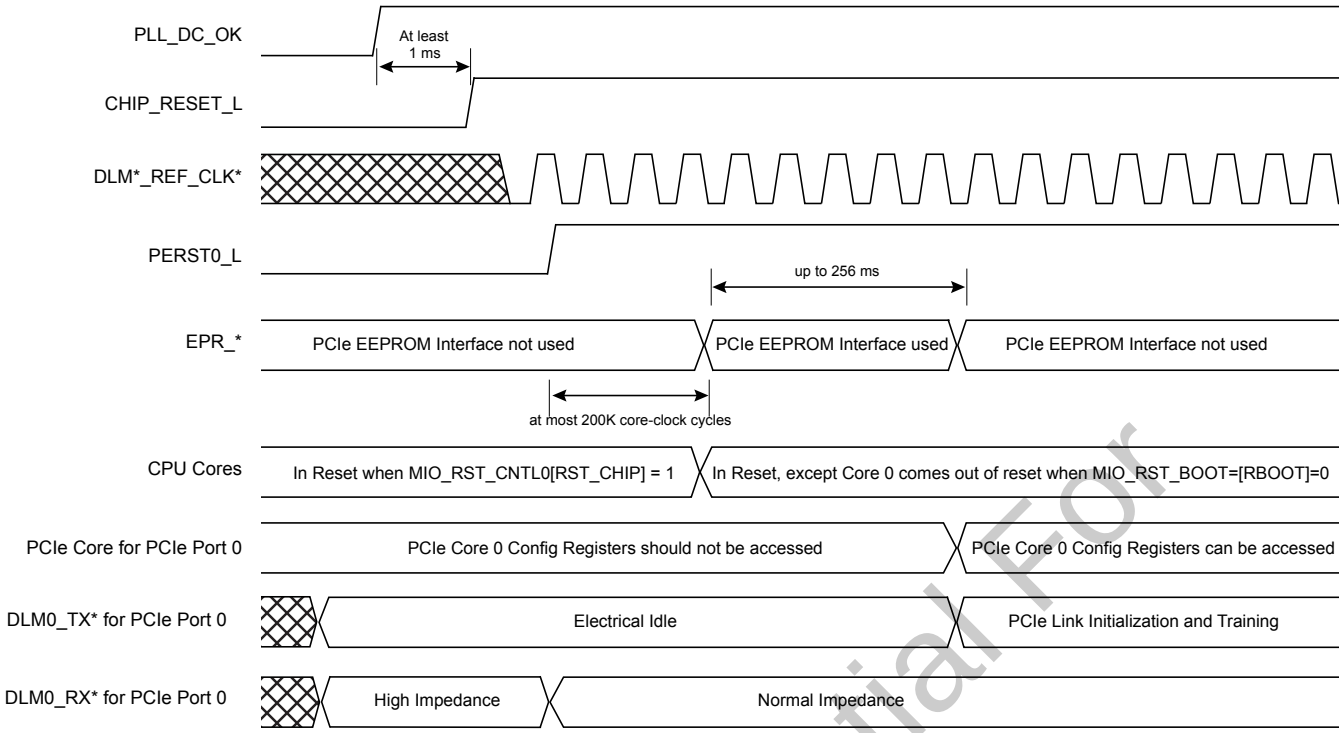
When a PCIe port 0 is in EP mode, RST_SOFT_PRST0[SOFT_PRST] resets to a 0 and PEM(0..2)_ON[PEMON] is 1.

Figure 22–8 shows some of the events of a warm reset sequence when PCIe port 0 is in EP mode, and Figure 22–9 shows some of the events of a cold reset sequence.

Cavium Confidential For Bowen Yang ALU 02/27/2014

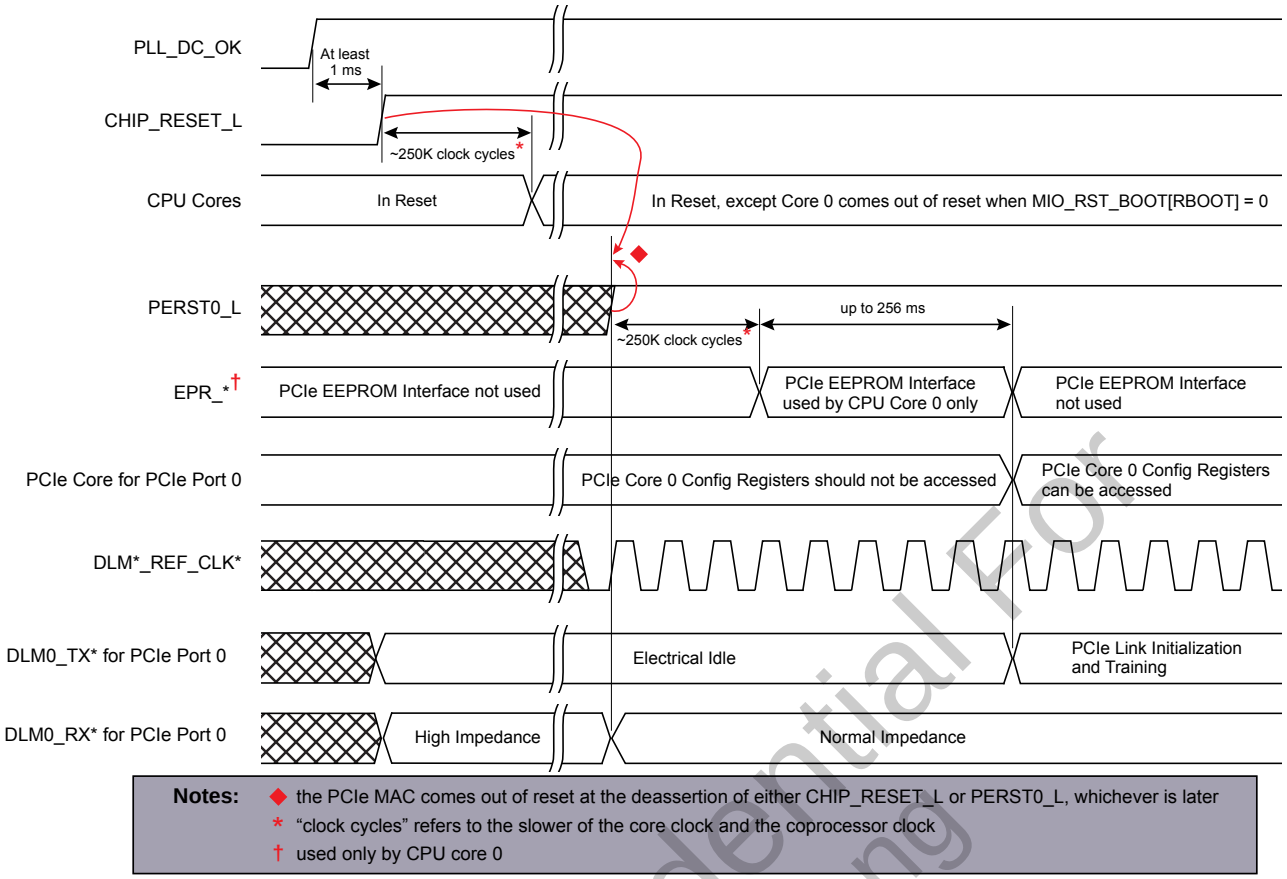
1. DLM*n*_REF_CLK* represents either DLM0_REF_CLK_P/N or DLMC_REF_CLK0/1_P/N.

Figure 22–8 PCIe Port Warm Reset Sequence in EP Mode



Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 22–9 Cold Reset Sequence in EP Mode



In this case, immediately following the deassertion of PERST0_L, CN71XX performs BIST on the PCIe core. When RST_CTL*n*[RST_CHIP] = 1, CN71XX also performs BIST on all internal memories as part of a chip warm-reset sequence that is generated from the assertion of PERST0_L. This takes ~250K core/coprocessor-clock cycles to complete (whichever clock has the lower frequency is used for this calculation, typically the coprocessor clock).

After BIST completes, CPU core 0 attempts to read the PCIe EEPROM via the EPR_* external pins (if PCIe port 0 is in EP mode).

- If a PCIe EEPROM is not present, the read operation stops soon after it starts.
- If a PCIe EEPROM is present, the read operation continues for up to 2.56 ms.

The PCIe core 0 configuration space registers should not be accessed (via PEM0_CFG_RD/PEM0_CFG_WR) until after the PCIe EEPROM read operation is complete.

When the chip is reset as part of the assertion of PERST0_L (i.e. when RST_CTL0[RST_CHIP] = 1), then:

- CPU core 0 comes out of reset immediately after BIST completes if RST_BOOT[RBOOT] = 1. Core 0 must not access the PCIe core 0 configuration space registers nor PCIe port 0 until the PCIe EEPROM load completes.

NOTE: The external pin REMOTE_BOOT is used to initialize RST_BOOT[RBOOT] during cold reset. A warm or soft reset has no effect on [RBOOT].

Cavium Confidential For Bowen Yang 02/27/2014

- all CPU cores remain in reset until released by an external device if RST_BOOT[RBOOT] = 0.

CN71XX starts the PCIe link initialization and training sequence once the PCIe EEPROM load completes. CN71XX normally can be accessed by the remote host 1 ms after PERST0_L deasserts.

When the chip is not reset as part of the assertion of PERST0_L (i.e. when RST_CTL0[RST_CHIP] = 0), only the PCIe core logic is reset and the cnMIPS cores are not directly affected. RST_INT[PERST0] is set and can cause an interrupt in this case.

This section assumes that the reset sequence is initiated by PERST0_L being received low. Note that it is also possible for software to reset the CN71XX PCIe core by asserting RST_SOFT_PRST*[SOFT_PRST] when RST_CTL0[HOST_MODE] = 0. However, RST_SOFT_PRST* is normally used only when RST_CTL0[HOST_MODE] = 1.

22.5.2 Soft Reset for PCIe

Refer to Section 13.9 for more details on soft reset. This section focuses on the affects of soft reset on the CN71XX PCIe ports.

22.5.2.1 Soft Reset for PCIe in RC Mode

When PCIe port *n* is in RC mode (i.e. when RST_CTL*n*[HOST_MODE] = 1), the effects of a chip soft reset on the PCIe logic are identical to a chip reset initiated by the external reset pin CHIP_RESET_L. All chip interfaces and CSR types are reset by a soft reset in RC. The bring-up sequence needed following the soft reset is described in Section 22.5.1.1 and Figure 22–7.

22.5.2.2 Soft Reset for PCIe in EP Mode

When PCIe port 0 is in EP mode (i.e. when RST_CTL0[HOST_MODE] = 0), the PCIe port 0, the PCIEEP0 registers, and PEM registers are not reset by a chip soft reset, but all other interfaces and all other register types are reset.

After a remote host initiates the CN71XX soft reset (via a store operation to RST_SOFT_RST), CN71XX cannot service anything other than PCI configuration space read and write operations until it has fully completed its internal reset sequence, so the remote host should not access CN71XX during this time. CN71XX can take as long as 1 ms to complete the reset sequence after it receives the RST_SOFT_RST store operation.

22.5.3 PCIe Link-Down or Hot Reset

When the PCIe link goes down either because the PHY link goes down or because a hot reset is received from a remote host, a number of things take place or need to take place, including resetting the internal CN71XX PCIe core logic associated with the link.

The PCIe link can go down for a number of reasons related to the physical link, and can also be forced by software setting PCIE*_CFG032[LD] = 1. A link-down event can also be caused by a link hot reset, which can be initiated by PCIERC*n*_CFG015[SBRST].

CN71XX's behavior upon PCIe link down or received link hot-reset command depends on the value of RST_CTL*n*[RST_LINK].

Cavium Confidential For Bowen Yang ALU 02/17/2014

RST_CTL*n*[RST_LINK] = 0:

This is the normal mode when the corresponding PCIE0_MODE<1:0> straps = 0x3 and RST_CTL*n*[HOST_MODE] = 1 (i.e. RC mode), but it could also occur when PCIE0_MODE<1:0> ≠ 0x3 and RST_CTL*n*[HOST_MODE] = 0 (i.e. EP mode).

CN71XX sets RST_INT[RST_LINK*] in this case. The distribution of the interrupt bits RST_INT[RST_LINK*] to various destinations is described in Chapter 13. If RST_CTL*n*[PRST_LINK] = 1, CN71XX also sets RST_SOFT_PRST*[SOFT_PRST], automatically forcing the PCIe controller into immediate reset. If the hardware has not already set RST_SOFT_PRST*[SOFT_PRST], then software should normally set it.

Once software detects that RST_INT[RST_LINK*] was set due to the link going down, it should disable all sources of request generation for the port.

Once that is completed and RST_CTL*n*[HOST_MODE] = 1, the port must be brought up again using the bring-up sequence described in Section 22.5.1.1 and Figure 22–7. Software should clear RST_INT[RST_LINK*] at some point so that future link-down events will be captured.

RST_CTL*n*[RST_LINK] = 1:

This mode normally occurs when the corresponding strap PCIE0_MODE<1:0> ≠ 0x3 and RST_CTL0[HOST_MODE] = 0 (i.e. EP mode). CN71XX initiates a chip-wide warm reset in this case.

The effects on the PCIe logic of a link-down event or received hot reset are identical to a chip reset initiated by the external reset pin CHIP_RESET_L. All chip interfaces and register types are reset. The entire chip is reset, including all register types and all interfaces, and including both PCIe ports and their corresponding CN71XX-internal cores.

CN71XX sets RST_BOOT[LBOOT] to 0x080 (for port 0) or 0x040 (for port 1) when a link-down or hot reset event occurs. For port 2, CN71XX sets RST_BOOT[LBOOT_EXT] to 0x040.

The bring-up sequence is similar to the sequence described in Section 22.5.1.2 and Figure 22–6, as long as RST_CTL0[HOST_MODE] = 0. The only difference is that in this case PERST0_L does not transition, and the DLM*_RX* for the port never enters the high-impedance state.

Note that, as with soft reset, the Status[SR] bit in the processor cores is set following a link-down or hot reset. Software can consult RST_BOOT[LBOOT] to determine whether the reset was due to a soft reset or a link-down/hot reset. Software should clear RST_BOOT[LBOOT] as part of the bring-up sequence so that it will not be set if a subsequent soft reset occurs.

Cavium Confidential For Review ALU 02/27/2014

22.6 PCIe Configuration Summary

This section contains lists of important state that should be configured, and a few hints on things that can go wrong.

PCIe configuration space CSRs (EP or RC):

- Max Payload Size (PCIE*_CFG030[MPS])
- Max Read Request Size (PCIE*_CFG030[MRRS])
- ECRC Generation (PCIE*_CFG070[GE,CE])
- Access Enables (PCIE*_CFG001[MSAE,ME])
  - ME and MSAE should always be set.
- Interrupt Disable (PCIE*_CFG001[I_DIS])
- Relaxed-order, no-snoop enables (PCIE*_CFG030[RO_EN,NS_EN]
- Error Message Enables (PCIE*_CFG030[CE_EN,NFE_EN,FE_EN,UR_EN])
- Advanced Error Recovery Message Enables (PCIE*_CFG066,PCIE*_CFG067,PCIE*_CFG069)
- System Error Message Enable (PCIE*_CFG001[SEE])
- Relaxed Ordering / No Snoop Enable (PCIE*_CFG030[RO_EN,NS_EN])
- Active State Power Management (PCIE*_CFG032[ASLPC])
  - Entrance Latencies (PCIE*_CFG451[L0EL,L1EL])

PCIe configuration space CSRs (RC-only):

- Link Width Mode (PCIERC*n*_CFG452[LME])
  - Must be set prior to enabling link. Refer to Section 22.5.1.1.
- Primary Bus Number (PCIERC*n*_CFG006[PBNUM])
  - Refer to Section 22.2.5.1.
- Memory-mapped I/O BAR (PCIERC*n*_CFG008)
  - Most applications should disable the memory-mapped I/O BAR by setting PCIERC*n*_CFG008[ML_ADDR] < PCIERC*n*_CFG008[MB_ADDR]
- Prefetchable BAR (PCIERC*n*_CFG009, PCIERC*n*_CFG010, PCIERC*n*_CFG011)
  - Most applications should disable the prefetchable BAR by setting PCIERC*n*_CFG011[UMEM_LIMIT],PCIERC*n*_CFG009[LMEM_LIMIT] < PCIERC*n*_CFG010[UMEM_BASE],PCIERC*n*_CFG009[LMEM_BASE]
- System Error Interrupt Enables (PCIERC*n*_CFG035[SECEE,SEFEE,SENFEE])
- PME Interrupt Enables (PCIERC*n*_CFG035[PMEIE])
- Advanced Error Recovery Interrupt Enables (PCIERC*n*_CFG075[CERE,NFERE,FERE])
- Hot Plug Interrupt Enables (PCIERC*n*_CFG034[HPINT_EN], PCIERC*n*_CFG034[DLLS_EN,CCINT_EN])

Cavium Confidential For Brian Kang ALD 6/2/2014

PCIe configuration space CSRs (EP-only):

- EP BAR0 (PCIEEP(0..2)_CFG004,PCIEEP(0..2)_CFG005)
  - Refer to Section 22.2.2.1.
- EP BAR1 (PCIEEP(0..2)_CFG006,PCIEEP(0..2)_CFG007)
  - Refer to Section 22.2.2.1.
- EP BAR2 (PCIEEP(0..2)_CFG008,PCIEEP(0..2)_CFG009,PCIEEP(0..2)_CFG008_MASK, PCIEEP(0..2)_CFG009_MASK)
  - Refer to EP BAR2 – 64-bit Memory-Mapped Region.
- (External) MSI (PCIEEP(0..2)_CFG020,PCIEEP(0..2)_CFG021,PCIEEP(0..2)_CFG022, PCIEEP(0..2)_CFG023)

RST CSRs (EP or RC):

- Port Reset (RST_SOFT_PRST1[SOFT_PRST])
  - This is the reset for the port. Refer to Section 22.5.

PEM CSRs (RC-only):

- Link Enable (PEM*n*_CTL_STATUS[LNK_ENB])
  - Needed to bring up the link in RC mode. Refer to Section 22.5.1.1.
- RC-internal BAR0 (PEM*n*_P2N_BAR0_START)
  - Refer to Section 22.2.2.2.
- RC-internal BAR1 (PEM*n*_P2N_BAR1_START)
  - Refer to Section 22.2.2.2.
- RC-internal BAR2 (PEM*n*_P2N_BAR2_START)
  - Refer to Section 22.2.2.2.
- RC BAR1 Size (PEM*n*_BAR_CTL[BAR1_SIZ])

PEM CSRs (EP or RC):

- For target PCIe memory space accesses
  - BAR1 index registers (PEM*n*_BAR1_INDEX(0..15))
    - If ADDR_V = 0, PEM does not service the portion of its BAR1 serviced by this table entry.
    - Endian-swapping is a typical problem area (END_SWP).
    - ADDR_IDX must also be set up properly.
    - Refer to Section 22.2.2 and Section 12.4.
  - BAR2 Control (PEM*n*_BAR_CTL[BAR2_ENB,BAR2_ESX,BAR2_CAX])
    - enable, default endian-swap, default cache attribute
    - Refer to Section 22.2.2 and Section 12.4.

Cavium Confidential For Bowen Yang ALU 07/7/2014

22.7 PCIe Configuration Registers

The PCIe configuration registers are of type PCICONFIGEP0 (PCIEEP(0..2)_CFG*) and PCICONFIGRC0/1/2 (PCIERC(0..2)_CFG*). All PCIe configuration registers are 32 bits.

Note that the PCIe configuration registers must not be accessed while the PEM is in reset. Refer to Section Section 22.5.1.

PCIe configuration register usage depends on mode configuration (root complex or endpoint) for the PCIe physical ports. In endpoint (EP) mode, PCIEEP(0..2)_CFG* registers are available; in root complex (RC) mode, PCIERC(0..2)_CFG* registers are available.

Section 22.7.1 describes the PCIe configuration registers available in EP mode; Section 22.7.2 describes the PCIe configuration registers available in RC mode.

22.7.1 EP Mode PCIe Configuration Registers

The PCIEEP(0..2)_CFG* registers (PCICONFIGEP) can be accessed from a remote device via PCIe Port 0, or they can be accessed via the PEM0_CFG_WR/PEM0_CFG_RD registers. When accessing the PCIEEP(0..2)_CFG* registers via the PEM0_CFG_WR/PEM0_CFG_RD registers, the ADDR field must match the PCIe configuration space address. Table 22–3 lists the PCICONFIGEP0 registers.

The PCIEEP(0..2)_CFG*_MASK registers are write-only registers (EP mode only) and are only accessible via the PEM0_CFG_WR registers. The ADDR field of the PEM0_CFG_WR registers must match the address listed in Table 22–3.

Note that the RO/WRSL field type applies to PCICONFIGEP0 registers only. The RO/WRSL fields are RO if accessed from the physical PCIe port, and R/W if accessed via PEM0_CFG_WR/PEM0_CFG_RD.

Table 22–3 PCICONFIGEP0 Registers

Register	Address (PCI Configuration Space)	Address (PEM*_CFG_RD/ PEM*_CFG_WR ADDR Field)	CSR Type	Detailed Description
PCIEEP(0..2)_CFG000	0x0000	0x0000	PCICONFIGEP	See page 922
PCIEEP(0..2)_CFG001	0x0004	0x0004	PCICONFIGEP	See page 923
PCIEEP(0..2)_CFG002	0x0008	0x0008	PCICONFIGEP	See page 924
PCIEEP(0..2)_CFG003	0x000C	0x000C	PCICONFIGEP	See page 924
PCIEEP(0..2)_CFG004	0x0010	0x0010	PCICONFIGEP	See page 925
PCIEEP(0..2)_CFG005	0x0014	0x0014	PCICONFIGEP	See page 925
PCIEEP(0..2)_CFG006	0x0018	0x0018	PCICONFIGEP	See page 925
PCIEEP(0..2)_CFG007	0x001C	0x001C	PCICONFIGEP	See page 926
PCIEEP(0..2)_CFG008	0x0020	0x0020	PCICONFIGEP	See page 926
PCIEEP(0..2)_CFG009	0x0024	0x0024	PCICONFIGEP	See page 926
PCIEEP(0..2)_CFG010	0x0028	0x0028	PCICONFIGEP	See page 926
PCIEEP(0..2)_CFG011	0x002C	0x002C	PCICONFIGEP	See page 927
PCIEEP(0..2)_CFG012	0x0030	0x0030	PCICONFIGEP	See page 927
PCIEEP(0..2)_CFG013	0x0034	0x0034	PCICONFIGEP	See page 927
PCIEEP(0..2)_CFG015	0x003C	0x003C	PCICONFIGEP	See page 927
PCIEEP(0..2)_CFG016	0x0040	0x0040	PCICONFIGEP	See page 928

Cavium Confidential For Bowen Yang ALL 02/27/2014

Table 22–3 PCICONFIGEP0 Registers

Register	Address (PCI Configuration Space)	Address (PEM*_CFG_RD/ PEM*_CFG_WR ADDR Field)	CSR Type	Detailed Description
PCIEEP(0..2)_CFG017	0x0044	0x0044	PCICONFIGEP	See page 929
PCIEEP(0..2)_CFG020	0x0050	0x0050	PCICONFIGEP	See page 929
PCIEEP(0..2)_CFG021	0x0054	0x0054	PCICONFIGEP	See page 930
PCIEEP(0..2)_CFG022	0x0058	0x0058	PCICONFIGEP	See page 930
PCIEEP(0..2)_CFG023	0x005C	0x005C	PCICONFIGEP	See page 930
PCIEEP(0..2)_CFG024	0x0060	0x0060	PCICONFIGEP	See page 930
PCIEEP(0..2)_CFG025	0x0064	0x0064	PCICONFIGEP	See page 931
PCIEEP(0..2)_CFG028	0x0070	0x0070	PCICONFIGEP	See page 931
PCIEEP(0..2)_CFG029	0x0074	0x0074	PCICONFIGEP	See page 931
PCIEEP(0..2)_CFG030	0x0078	0x0078	PCICONFIGEP	See page 932
PCIEEP(0..2)_CFG031	0x007C	0x007C	PCICONFIGEP	See page 934
PCIEEP(0..2)_CFG032	0x0080	0x0080	PCICONFIGEP	See page 934
PCIEEP(0..2)_CFG037	0x0094	0x0094	PCICONFIGEP	See page 935
PCIEEP(0..2)_CFG038	0x0098	0x0098	PCICONFIGEP	See page 936
PCIEEP(0..2)_CFG039	0x009C	0x009C	PCICONFIGEP	See page 936
PCIEEP(0..2)_CFG040	0x00A0	0x00A0	PCICONFIGEP	See page 937
PCIEEP(0..2)_CFG064	0x0100	0x0100	PCICONFIGEP	See page 938
PCIEEP(0..2)_CFG065	0x0104	0x0104	PCICONFIGEP	See page 938
PCIEEP(0..2)_CFG066	0x0108	0x0108	PCICONFIGEP	See page 939
PCIEEP(0..2)_CFG067	0x010C	0x010C	PCICONFIGEP	See page 939
PCIEEP(0..2)_CFG068	0x0110	0x0110	PCICONFIGEP	See page 940
PCIEEP(0..2)_CFG069	0x0114	0x0114	PCICONFIGEP	See page 940
PCIEEP(0..2)_CFG070	0x0118	0x0118	PCICONFIGEP	See page 941
PCIEEP(0..2)_CFG071	0x011C	0x011C	PCICONFIGEP	See page 941
PCIEEP(0..2)_CFG072	0x0120	0x0120	PCICONFIGEP	See page 941
PCIEEP(0..2)_CFG073	0x0124	0x0124	PCICONFIGEP	See page 941
PCIEEP(0..2)_CFG074	0x0128	0x0128	PCICONFIGEP	See page 941
PCIEEP(0..2)_CFG082	0x0148	0x0120	PCICONFIGEP	See page 942
PCIEEP(0..2)_CFG083	0x014C	0x0124	PCICONFIGEP	See page 942
PCIEEP(0..2)_CFG084	0x0150	0x0128	PCICONFIGEP	See page 942
PCIEEP(0..2)_CFG448	0x0700	0x0700	PCICONFIGEP	See page 943
PCIEEP(0..2)_CFG449	0x0704	0x0704	PCICONFIGEP	See page 943
PCIEEP(0..2)_CFG450	0x0708	0x0708	PCICONFIGEP	See page 944
PCIEEP(0..2)_CFG451	0x070C	0x070C	PCICONFIGEP	See page 945
PCIEEP(0..2)_CFG452	0x0710	0x0710	PCICONFIGEP	See page 946
PCIEEP(0..2)_CFG453	0x0714	0x0714	PCICONFIGEP	See page 947
PCIEEP(0..2)_CFG454	0x0718	0x0718	PCICONFIGEP	See page 947
PCIEEP(0..2)_CFG455	0x071C	0x071C	PCICONFIGEP	See page 947
PCIEEP(0..2)_CFG456	0x0720	0x0720	PCICONFIGEP	See page 948
PCIEEP(0..2)_CFG458	0x0728	0x0728	PCICONFIGEP	See page 948
PCIEEP(0..2)_CFG459	0x072C	0x072C	PCICONFIGEP	See page 948

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 22–3 PCICONFIGEP0 Registers

Register	Address (PCI Configuration Space)	Address (PEM*_CFG_RD/ PEM*_CFG_WR ADDR Field)	CSR Type	Detailed Description
PCIEEP(0..2)_CFG460	0x0730	0x0730	PCICONFIGEP	See page 949
PCIEEP(0..2)_CFG461	0x0734	0x0734	PCICONFIGEP	See page 949
PCIEEP(0..2)_CFG462	0x0738	0x0738	PCICONFIGEP	See page 949
PCIEEP(0..2)_CFG463	0x073C	0x073C	PCICONFIGEP	See page 950
PCIEEP(0..2)_CFG464	0x0740	0x0740	PCICONFIGEP	See page 950
PCIEEP(0..2)_CFG465	0x0744	0x0744	PCICONFIGEP	See page 950
PCIEEP(0..2)_CFG466	0x0748	0x0748	PCICONFIGEP	See page 951
PCIEEP(0..2)_CFG467	0x074C	0x074C	PCICONFIGEP	See page 952
PCIEEP(0..2)_CFG468	0x0750	0x0750	PCICONFIGEP	See page 952
PCIEEP(0..2)_CFG490	0x07A8	0x07A8	PCICONFIGEP	See page 953
PCIEEP(0..2)_CFG491	0x07AC	0x07AC	PCICONFIGEP	See page 953
PCIEEP(0..2)_CFG492	0x07B0	0x07B0	PCICONFIGEP	See page 954
PCIEEP(0..2)_CFG515	0x080C	0x080C	PCICONFIGEP	See page 954
PCIEEP(0..2)_CFG516	0x0810	0x0810	PCICONFIGEP	See page 955
PCIEEP(0..2)_CFG517	0x0814	0x0814	PCICONFIGEP	See page 955
PCIEEP(0..2)_CFG004_MASK	—	0x80000010	PCICONFIGEP	See page 955
PCIEEP(0..2)_CFG005_MASK	—	0x80000014	PCICONFIGEP	See page 955
PCIEEP(0..2)_CFG006_MASK	—	0x80000018	PCICONFIGEP	See page 956
PCIEEP(0..2)_CFG007_MASK	—	0x8000001C	PCICONFIGEP	See page 956
PCIEEP(0..2)_CFG008_MASK	—	0x80000020	PCICONFIGEP	See page 956
PCIEEP(0..2)_CFG009_MASK	—	0x80000024	PCICONFIGEP	See page 956
PCIEEP(0..2)_CFG012_MASK	—	0x80000030	PCICONFIGEP	See page 957

PCIe Vendor and Device Register PCIEEP(0..2)_CFG000

This register contains the first 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	DEVID	RO/WRSL	0x96	0x96	Device ID for CN71XX, writable through PEM(0..2)_CFG_WR. However, the application must not change this field. For EEPROM loads, also see VENDID of this register. The possible values are: CN63XX = 0x90 CN66XX = 0x92 CN68XX = 0x91 CN61XX = 0x93 CNF71XX = 0x94 CN78XX = 0x95 CN71/70XX = 0x96
<15:0>	VENDID	RO/WRSL	0x177D	0x177D	Cavium's vendor ID, writable through PEM(0..2)_CFG_WR. However, the application must not change this field. During an EPROM Load, if a value of 0xFFFF is loaded to this field and a value of 0xFFFF is loaded to the DEVID field of this register, the value will not be loaded, EEPROM load will stop, and the FastLinkEnable bit will be set in the PCIEEP(0..2)_CFG452 register.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Command/Status Register
PCIEEP(0..2)_CFG001

This register contains the second 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	DPE	R/W1C/H	0	0	Detected parity error
<30>	SSE	R/W1C/H	0	0	Signaled system error
<29>	RMA	R/W1C/H	0	0	Received master abort
<28>	RTA	R/W1C/H	0	0	Received target abort
<27>	STA	R/W1C/H	0	0	Signaled target abort
<26:25>	DEVT	RO	0x0	0x0	DEVSEL timing. Not applicable for PCI Express. Hardwired to 0x0.
<24>	MDPE	R/W1C/H	0	0	Master data parity error
<23>	FBB	RO	0	0	Fast back-to-back capable. Not applicable for PCI Express. Hardwired to 0.
<22>	—	RAZ	—	—	Reserved
<21>	M66	RO	0	0	66 MHz capable. Not applicable for PCI Express. Hardwired to 0.
<20>	CL	RO	1	1	Capabilities list. Indicates presence of an extended capability item. Hardwired to 1.
<19>	I_STAT	RO/H	0	0	When INT*x*# is asserted by CN71XX, this bit is set. When INT*x*# is deasserted by CN71XX, this bit is cleared.
<18:11>	—	RAZ	—	—	Reserved
<10>	I_DIS	R/W	0	0	When set to 1, this bit disables the generation of INT*x*# by CN71XX. When cleared to 0, allows assertion of INT*x*# by CN71XX.
<9>	FBBE	RO	0	0	Fast back-to-back transaction enable. Not applicable for PCI Express. Must be hardwired to 0.
<8>	SEE	R/W	0	0	SERR# enable
<7>	IDS_WCC	RO	0	0	IDSEL stepping/wait cycle control. Not applicable for PCI Express. Must be hardwired to 0.
<6>	PER	R/W	0	0	Parity error response
<5>	VPS	RO	0	0	VGA palette snooping. Not applicable for PCI Express. Must be hardwired to 0.
<4>	MWICE	RO	0	0	Memory write & invalidate command enable. Not applicable for PCI Express. Must be hardwired to 0.
<3>	SCSE	RO	0	0	Special cycle snooping enable. Not applicable for PCI Express. Must be hardwired to 0.
<2>	ME	R/W	0	0	Bus master enable
<1>	MSAE	R/W	0	0	Memory space access enable
<0>	ISAE	R/W	0	0	I/O space access enable

Cavium Confidential For Bowen Tang AE D 9/27/2014

Class Code/Revision ID Register PCIEEP(0..2)_CFG002

This register contains the third 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	BCC	RO/WRSL	0xB	0xB	Base class code, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<23:16>	SC	RO/WRSL	0x30	0x30	Subclass code, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<15:8>	PI	RO/WRSL	0x0	0x0	Programming interface, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<7:0>	RID	RO/WRSL	0x8	0x8	Revision ID, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

BIST, Header Type, Master Latency Timer, Cache Line Size Register PCIEEP(0..2)_CFG003

This register contains the fourth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	BIST	RO	0x0	0x0	The BIST register functions are not supported. All 8 bits of the BIST register are hardwired to 0x0.
<23>	MFD	RO/WRSL	0	0	Multi function device. The multi function device bit is writable through PEM(0..2)_CFG_WR. However, this is a single function device. Therefore, the application must not write a 1 to this bit.
<22:16>	CHF	RO	0x0	0x0	Configuration header format. Hardwired to 0x0 for type 0.
<15:8>	LT	RO	0x0	0x0	Master latency timer. Not applicable for PCI Express, hardwired to 0x0.
<7:0>	CLS	R/W	0x0	0x0	Cache line size. The cache line size register is R/W for legacy compatibility purposes and is not applicable to PCI Express device functionality. Writing to the cache line size register does not impact functionality of the PCI Express bus.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Base Address Register 0 – Low
PCIEEP(0..2)_CFG004

This register contains the fifth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:14>	LBAB	R/W	0x0	0x0	Lower bits of the BAR 0 base address.
<13:4>	—	RAZ	—	—	Reserved. If the Bit Mask Register (PCIE_CFG004_MASK or PCIE_CFG005_MASK) is changed, see LBAB.
<3>	PF	RO/WRSL	1	1	Prefetchable. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<2:1>	TYP	RO/WRSL	0x2	0x2	BAR type. 00: 32-bit BAR 10: 64-bit BAR This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<0>	MSPC	RO/WRSL	0	0	Memory space indicator. 0: BAR 0 is a memory BAR 1: BAR 0 is an I/O BAR This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Base Address Register 0 – High
PCIEEP(0..2)_CFG005

This register contains the sixth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	UBAB	R/W	0x0	0x0	Contains the upper 32 bits of the BAR 0 base address.

Base Address Register 1 – Low
PCIEEP(0..2)_CFG006

This register contains the seventh 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:26>	LBAB	R/W	0x0	0x0	Lower bits of the BAR 1 base address
<25:4>	—	RAZ	—	—	Reserved. If the Bit Mask Register (PCIE_CFG006_MASK or PCIE_CFG007_MASK) is changed, see LBAB.
<3>	PF	RO/WRSL	1	1	Prefetchable. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<2:1>	TYP	RO/WRSL	0x2	0x2	BAR type. 00: 32-bit BAR 10: 64-bit BAR This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<0>	MSPC	RO/WRSL	0	0	Memory space indicator. 0: BAR 1 is a memory BAR 1: BAR 1 is an I/O BAR This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Base Address Register 1 - High
PCIEEP(0..2)_CFG007

This register contains the eighth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	UBAB	R/W	0x0	0x0	Contains the upper 32 bits of the BAR 1 base address.

Base Address Register 2 - Low
PCIEEP(0..2)_CFG008

This register contains the ninth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	LBAB	R/W	0x0	0x0	Lower bits of the BAR 2 base address
<19:4>	—	RAZ	—	—	Reserved. If the Bit Mask Register (PCIE_CFG008_MASK or PCIE_CFG009_MASK) is changed, see LBAB.
<3>	PF	RO/WRSL	1	1	Prefetchable. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<2:1>	TYP	RO/WRSL	0x2	0x2	BAR type. 00: 32-bit BAR 10: 64-bit BAR This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<0>	MSPC	RO/WRSL	0	0	Memory space indicator. 0: BAR 2 is a memory BAR 1: BAR 2 is an I/O BAR This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Base Address Register 2 - High Register
PCIEEP(0..2)_CFG009

This register contains the tenth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	UBAB	R/W	0x0	0x0	Contains the upper 32 bits of the BAR 2 base address.

Card Bus CIS Pointer Register
PCIEEP(0..2)_CFG010

This register contains the eleventh 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	CISP	RO/WRSL	0x0	0x0	CardBus CIS pointer. Optional, writable through PEM(0..2)_CFG_WR.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SubSystem ID/Subsystem Vendor ID Register PCIEEP(0..2)_CFG011

This register contains the twelfth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	SSID	RO/WRSL	0x1	0x1	Subsystem ID. CN71XX = 0x1 Assigned by PCI-SIG, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<15:0>	SSVID	RO/WRSL	0x177D	0x177D	Subsystem vendor ID. CN71XX = 0x177D. Assigned by PCI-SIG, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Expansion ROM Base Address Register PCIEEP(0..2)_CFG012

This register contains the thirteenth 32-bits of type 0 PCIe configuration space. This register is only enabled in PCIE0; PCIE1 and PCIE2 does not have this bar enabled. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	ERADDR	R/W	0x0	0x0	Expansion ROM address
<15:1>	—	RAZ	—	—	Reserved
<0>	ER_EN	R/W	0	0	Expansion ROM enable

Capability Pointer Register PCIEEP(0..2)_CFG013

This register contains the fourteenth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:8>	—	RAZ	—	—	Reserved
<7:0>	CP	RO/WRSL	0x40	0x40	First capability pointer. Points to Power Management Capability structure by default, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Interrupt Line/Interrupt Pin/Bridge Control Register PCIEEP(0..2)_CFG015

This register contains the sixteenth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	ML	RO/H	0x00	0x00	Maximum latency (hardwired to 0x00).
<23:16>	MG	RO/H	0x00	0x00	Minimum grant (hardwired to 0x00).
<15:8>	INTA	RO/WRSL	0x1	0x1	Interrupt pin. Identifies the legacy interrupt message that the device (or device function) uses. The Interrupt Pin register is writable through PEM(0..2)_CFG_WR. In a single-function configuration, only INTA is used. Therefore, the application must not change this field.
<7:0>	IL	R/W	0xFF	0xFF	Interrupt line

Cavium Confidential For Bowen Yang BWALU 02/27/2014

Power Management Capability ID/Power Management Next Item Pointer/Power Management Capabilities Register
PCIEEP(0..2)_CFG016

This register contains the seventeenth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:27>	PMES	RO/WRSL	0x0	0x0	PME_Support. Bit 11: If set, PME Messages can be generated from D0 Bit 12: If set, PME Messages can be generated from D1 Bit 13: If set, PME Messages can be generated from D2 Bit 14: If set, PME Messages can be generated from D3hot Bit 15: If set, PME Messages can be generated from D3cold The PME_Support field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<26>	D2S	RO/WRSL	0	0	D2 support, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<25>	D1S	RO/WRSL	0	0	D1 support, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<24:22>	AUXC	RO/WRSL	0x0	0x0	AUX current, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<21>	DSI	RO/WRSL	0	0	Device specific initialization (DSI), writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<20>	—	RAZ	—	—	Reserved
<19>	PME_CLOCK	RO	0	0	PME clock, hardwired to 0.
<18:16>	PMSV	RO/WRSL	0x3	0x00	Power management specification version, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<15:8>	NCP	RO/WRSL	0x50	0x00	Next capability pointer. Points to the MSI capabilities by default, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<7:0>	PMCID	RO	0x1	0x00	Power management capability ID.

Cavium Confidential
Bowen Kang
ALU
02/27/2014

Power Management Control and Status Register PCIEEP(0..2)_CFG017

This register contains the eighteenth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	PMDIA	RO	0x0	0x0	Data register for additional information (not supported).
<23>	BPCCEE	RO	0	0	Bus power/clock control enable, hardwired to 0.
<22>	BD3H	RO	0	0	B2/B3 support, hardwired to 0.
<21:16>	—	RAZ	—	—	Reserved
<15>	PMESS	R/W1C/H	0	0	PME status. Indicates whether or not a previously enabled PME event occurred.
<14:13>	PMEDSIA	RO	0x0	0x0	Data scale (not supported).
<12:9>	PMDS	RO	0x0	0x0	Data select (not supported).
<8>	PMEENS	R/W	0	0	PME enable. A value of 1 indicates that the device is enabled to generate PME.
<7:4>	—	RAZ	—	—	Reserved
<3>	NSR	RO/WRSL	0	0	No soft reset, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<2>	—	RAZ	—	—	Reserved
<1:0>	PS	R/W	0x0	0x0	Power state. Controls the device power state: 00: D0 01: D1 10: D2 11: D3 The written value is ignored if the specific state is not supported.

MSI Capability ID/MSI Next Item Pointer/MSI Control Register PCIEEP(0..2)_CFG020

This register contains the twenty-first 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:25>	—	RAZ	—	—	Reserved
<24>	PVM	RO	0	0	Per-vector masking capable
<23>	M64	RO/WRSL	1	1	64-bit address capable, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<22:20>	MME	R/W	0x0	0x0	Multiple message enabled. Indicates that multiple message mode is enabled by system software. The number of messages enabled must be less than or equal to the multiple message capable (MMC) value.
<19:17>	MMC	RO/WRSL	0x0	0x0	Multiple message capable, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<16>	MSIEN	R/W	0	0	MSI enabled.When set, INTx must be disabled.
<15:8>	NCP	RO/WRSL	0x70	0x70	Next capability pointer. Points to PCI Express capabilities by default, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<7:0>	MSICID	RO	0x5	0x5	MSI capability ID

Cavium Confidential For Bowen Yang ALU 6/27/2014

MSI Lower 32 Bits Address Register PCIEEP(0..2)_CFG021

This register contains the twenty-second 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:2>	LMSI	R/W	0x0	0x0	Lower 32-bit address
<1:0>	—	RAZ	—	—	Reserved

MSI Upper 32 Bits Address Register PCIEEP(0..2)_CFG022

This register contains the twenty-third 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	UMSI	R/W	0x0	0x0	Upper 32-bit address

MSI Mask Register PCIEEP(0..2)_CFG023

This register contains the twenty-fourth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	—	RAZ	—	—	Reserved
<15:0>	MSIMD	R/W	0x0	0x0	MSI data. Pattern assigned by system software. Bits [4:0] are ORed with MSI_VECTOR to generate 32 MSI messages per function.

MSI Data Register PCIEEP(0..2)_CFG024

This register contains the twenty-fifth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	MSIMM	R/W	0x0	0x0	MSI Mask. For each mask bit that is set, the function is prohibited from sending the associated message.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MSI Pending Register
PCIEEP(0..2)_CFG025

This register contains the twenty-sixth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	MSIMP	RO	0x0	0x0	MSI Pending. For each Pending bit that is set, the function has a pending associated message.

PCIe Capabilities/PCIe Capabilities List Register
PCIEEP(0..2)_CFG028

This register contains the twenty-ninth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:30>	—	RAZ	—	—	Reserved
<29:25>	IMN	RO/WRSL	0x0	0x0	Interrupt message number. Updated by hardware, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<24>	SI	RO/WRSL	0	0	Slot implemented. This bit is writable through PEM(0..2)_CFG_WR. However, it must be 0 for an endpoint device. Therefore, the application must not write a 1 to this bit.
<23:20>	DPT	RO	0x0	0x0	Device port type
<19:16>	PCIECV	RO	0x2	0x2	PCI Express capability version
<15:8>	NCP	RO/WRSL	0x0	0x0	Next capability pointer. Writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<7:0>	PCIEID	RO	0x10	0x10	PCI Express capability ID

Device Capabilities Register
PCIEEP(0..2)_CFG029

This register contains the thirtieth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:29>	—	RAZ	—	—	Reserved
<28>	FLR_CAP	RO	0	0	Function level reset capability. Not supported.
<27:26>	CSPLS	RO	0x0	0x0	Captured slot power limit scale. From message from RC, upstream port only.
<25:18>	CSPLV	RO	0x0	0x0	Captured slot power limit value. From message from RC, upstream port only.
<17:16>	—	RAZ	—	—	Reserved
<15>	RBER	RO/WRSL	1	1	Role-based error reporting, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<14:12>	—	RAZ	—	—	Undefined
<11:9>	EL1AL	RO/WRSL	0x3	0x3	Endpoint L1 acceptable latency, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<8:6	EL0AL	RO/WRSL	0x4	0x4	Endpoint L0s acceptable latency, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<5>	ETFS	RO/WRSL	0	0	Extended tag field supported. This bit is writable through PEM(0..2)_CFG_WR. However, the application must not write a 1 to this bit.
<4:3>	PFS	RO/WRSL	0x0	0x0	Phantom function supported. This field is writable through PEM(0..2)_CFG_WR. However, phantom function is not supported. Therefore, the application must not write any value other than 0x0 to this field.
<2:0>	MPSS	RO/WRSL	0x1	0x1	Max_Payload_Size supported, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Device Control/Device Status Register PCIEEP(0..2)_CFG030

This register contains the thirty-first 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:22>	—	RAZ	—	—	Reserved
<21>	TP	RO/H	0	0	Transaction pending. Set to 1 when non-posted requests are not yet completed and set to 0 when they are completed.
<20>	AP_D	RO	0	0	Aux power detected. Set to 1 if Aux power detected.
<19>	UR_D	R/W1C/H	0	0	Unsupported request detected. Errors are logged in this register regardless of whether or not error reporting is enabled in the device control register. UR_D occurs when we receive something unsupported. Unsupported requests are Nonfatal errors, so UR_D should cause NFE_D. Receiving a vendor-defined message should cause an unsupported request.
<18>	FE_D	R/W1C/H	0	0	Fatal error detected. Errors are logged in this register regardless of whether or not error reporting is enabled in the device control register. This field is set if we receive any of the errors in PCIEEP(0..2)_CFG066 that has a severity set to fatal. Malformed TLPs generally fit into this category.
<17>	NFE_D	R/W1C/H	0	0	Non-fatal error detected. Errors are logged in this register regardless of whether or not error reporting is enabled in the device control register. This field is set if we receive any of the errors in PCIEEP(0..2)_CFG066 that has a severity set to nonfatal and does not meet Advisory nonfatal criteria, which most poisoned TLPs should.
<16>	CE_D	R/W1C/H	0	0	Correctable error detected. Errors are logged in this register regardless of whether or not error reporting is enabled in the device control register. This field is set if we receive any of the errors in PCIEEP(0..2)_CFG068, for example a replay-timer timeout. Also, it can be set if we get any of the errors in PCIEEP(0..2)_CFG066 that has a severity set to Nonfatal and meets the Advisory Nonfatal criteria, which most ECRC errors should.
<15>	I_FLR	R/W	0	0	Initiate function level reset (not supported)

Cavium Confidential For Beyond Yard 09/27/2017

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<14:12>	MRRS	R/W	0x2	0x2	Max read request size. 0x0: 128 B 0x3: 1024 B 0x1: 256 B 0x4: 2048 B 0x2: 512 B 0x5: 4096 B **NOTE:** SLI_S2M_PORT(0..2)_CTL[MRRS] and DPI_SLI_PRT(0..2)_CFG[MRRS] must also be set properly. SLI_S2M_PORT(0..2)_CTL[MRRS] and DPI_SLI_PRT(0..2)_CFG[MRRS] must not exceed the desired max read request size.
<11>	NS_EN	R/W	1	1	Enable no snoop
<10>	AP_EN	R/W	0	0	AUX power PM enable
<9>	PF_EN	R/W	0	0	Phantom function enable. This bit should never be set; PEM requests never use phantom functions.
<8>	ETF_EN	R/W	0	0	Extended tag field enable. This bit should never be set; PEM requests never use extended tags.
<7:5>	MPS	R/W	0x0	0x0	Max payload size. Legal values: 0x0 = 128 B, 0x1 = 256 B. Larger sizes are not supported by PEM. **NOTE:** DPI_SLI_PRT(0..2)_CFG[MPS] must be set to the same value as this field for proper functionality.
<4>	RO_EN	R/W	1	1	Enable relaxed ordering
<3>	UR_EN	R/W	0	0	Unsupported request reporting enable
<2>	FE_EN	R/W	0	0	Fatal error reporting enable
<1>	NFE_EN	R/W	0	0	Non-fatal error reporting enable
<0>	CE_EN	R/W	0	0	Correctable error reporting enable

Cavium Confidential For Bowen Yang ALU 02/27/2014

Link Capabilities Register PCIEEP(0..2)_CFG031

This register contains the thirty-second 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	PNUM	RO/WRSL	0x0	0x0	Port number, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<23>	—	RAZ	—	—	Reserved
<22>	ASPM	RO/WRSL	1	1	ASPM optionality compliance.
<21>	LBNC	RO	0	0	Link bandwidth notification capability. Set to 0 for endpoint devices.
<20>	DLLARC	RO	0	0	Data Link Layer active reporting capable
<19>	SDERC	RO	0	0	Surprise down error reporting capable. Not supported; hardwired to 0.
<18>	CPM	RO/WRSL	0	0	Clock power management. The default value is the value you specify during hardware configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<17:15>	L1EL	RO/WRSL	0x1	0x1	L1 exit latency. The default value is the value you specify during hardware configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<14:12>	L0EL	RO/WRSL	0x0	0x0	L0s exit latency. The default value is the value you specify during hardware configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<11:10>	ASLPMS	RO/WRSL	0x3	0x3	Active state Link PM support. The default value is the value you specify during hardware configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<9:4>	MLW	RO/WRSL	0x4	0x4	Maximum Link width. The default value is the value you specify during hardware configuration (×1, ×2, ×4), writable through PEM(0..2)_CFG_WR.
<3:0>	MLS	RO/WRSL	—	—	Maximum Link speed.The reset value of this field is controlled by a value sent from the MSB of PEM(0..2)_CFG[MD]. **MD[2] / RST_VALUE / NOTE** 1 / 0x1 / 2.5 GHz supported 0 / 0x2 / 5.0 GHz and 2.5 GHz supported This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Link Control/Link Status Register PCIEEP(0..2)_CFG032

This register contains the thirty-third 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	LAB	RO/H	0	0	Link autonomous bandwidth status
<30>	LBM	RO/H	0	0	Link bandwidth management status
<29>	DLLA	RO	0	0	Data-link layer active. Not applicable for an upstream port or endpoint device, hardwired to 0.

Cavium Confidential For Bowen Yang ALU 02/21/2018

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<28>	SCC	RO/WRSL	1	1	Slot clock configuration. Indicates that the component uses the same physical reference clock that the platform provides on the connector. Writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<27>	LT	RO	0	0	Link training. Not applicable for an upstream port or endpoint device, hardwired to 0.
<26>	—	RAZ	—	—	Undefined
<25:20>	NLW	RO/H	0x1	0x4	Negotiated link width. Set automatically by hardware after Link initialization. Value is undefined when link is not up.
<19:16>	LS	RO/H	0x1	0x1	Current link speed. The encoded value specifies a bit location in the supported link speeds vector (in the link capabilities 2 register) that corresponds to the current Link speed. 0x1 = the negotiated link speed = 2.5 Gb/s 0x2 = the negotiated link speed = 5.0 Gb/s 0x4 = the negotiated link speed = 8.0 Gb/s (not supported)
<15:12>	—	RAZ	—	—	Reserved
<11>	LAB_INT_ENB	RO	0	0	Link autonomous bandwidth interrupt enable.This bit is not applicable and is reserved for endpoints.
<10>	LBM_INT_ENB	RO	0	0	Link bandwidth management interrupt enable. This bit is not applicable and is reserved for endpoints.
<9>	HAWD	RO	0	0	Hardware autonomous width disable (not supported)
<8>	ECPM	R/W	0	0	Enable clock power management. Hardwired to 0 if clock power management is disabled in the Link Capabilities register.
<7>	ES	R/W	0	0	Extended synch
<6>	CCC	R/W	0	0	Common clock configuration
<5>	RL	RO	0	0	Retrain link. Not applicable for an upstream port or endpoint device. Hardwired to 0.
<4>	LD	RO	0	0	Link disable. Not applicable for an upstream port or endpoint device. Hardwired to 0.
<3>	RCB	RO	0	0	Read completion boundary (RCB)
<2>	—	RAZ	—	—	Reserved
<1:0>	ASLPC	R/W	0x0	0x0	Active state link PM control

Device Capabilities Register 2 PCIEEP(0..2)_CFG037

This register contains the thirty-eighth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:14>	—	RAZ	—	—	Reserved
<13:12>	TPHS	RO	0x0	0x0	TPH completer supported (not supported)
<11>	—	RAZ	—	—	Reserved
<10>	NOROPRPR	RO/H	0	0	No RO-enabled PR-PR passing. (This bit applies to RCs.)
<9>	ATOM128S	RO	0	0	128-bit AtomicOp supported (not supported)
<8>	ATOM64S	RO	1	1	64-bit AtomicOp supported (not supported)
<7>	ATOM32S	RO	1	1	32-bit AtomicOp supported (not supported)
<6>	ATOM_OPS	RO	0	0	AtomicOp routing supported (not applicable for EP)
<5>	ARI	RO	0	0	Alternate routing ID forwarding supported (not supported)
<4>	CTDS	RO	1	1	Completion timeout disable supported
<3:0>	CTRS	RO/H	0xF	0xF	Completion timeout ranges supported

Cavium Confidential For Bowen Tang ALU 03/27/2014

Device Control Register 2/Device Status Register 2
PCIEEP(0..2)_CFG038

This register contains the thirty-ninth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:10>	—	RAZ	—	—	Reserved
<9>	ID0_CP	RO	0	0	ID based ordering completion enable (not supported)
<8>	ID0_RQ	RO	0	0	ID based ordering request enable (not supported)
<7>	ATOM_OP_EB	RO	0	0	AtomicOp egress blocking (not supported)
<6>	ATOM_OP	RO	0	0	AtomicOp requester enable (not supported)
<5>	ARI	RO	0	0	Alternate routing ID forwarding supported (not supported)
<4>	CTD	R/W	0	0	Completion timeout disable
<3:0>	CTV	RO/H	0x0	0x0	Completion timeout value. Completion timeout programming is not supported. Completion timeout is the range of 16 ms to 55 ms.

Link Capabilities Register 2
PCIEEP(0..2)_CFG039

This register contains the fortieth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:9>	—	RAZ	—	—	Reserved
<8>	CLS	RO	0	0	Crosslink supported.
<7:1>	SLSV	RO/WRSL	—	—	Supported link speeds vector. Indicates the supported link speeds of the associated port. For each bit, a value of 1 indicates that the corresponding link speed is supported; otherwise, the link speed is not supported. Bit definitions are: bit<1>: 2.5 GT/s bit<2>: 5.0 GT/s bit<3>: 8.0 GT/s (not supported) bits<7:4> are reserved The reset value of this field is controlled by a value sent from the TBD of the TBD register. **pi_sel_md[2] RST_VALUE NOTE** 1 0x1 2.5 GHz supported 0 0x3 5.0 GHz and 2.5 GHz supported
<0>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang 02/27/2014

Link Control 2 Register/Link Status 2 Register PCIEEP(0..2)_CFG040

This register contains the forty-first 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:17>	—	RAZ	—	—	Reserved
<16>	CDL	RO/H	0	0	Current de-emphasis level. When the link is operating at 5 GT/s speed, this bit reflects the level of de-emphasis. Encodings: 1 = –3.5 dB 0 = –6 dB **NOTE:** The value in this bit is undefined when the link is operating at 2.5 GT/s speed.
<15:13>	—	RAZ	—	—	Reserved
<12>	CDE	R/W	0	0	Compliance de-emphasis. This bit sets the de-emphasis level in Polling. Compliance state if the entry occurred due to the Tx compliance receive bit being 1. Encodings: 1 = –3.5 dB 0 = –6 dB **NOTE:** When the Link is operating at 2.5 GT/s, the setting of this bit has no effect.
<11>	CSOS	R/W	0	0	Compliance SOS. When set to 1, the LTSSM is required to send SKP ordered sets periodically in between the (modified) compliance patterns. **NOTE:** When the Link is operating at 2.5 GT/s, the setting of this bit has no effect.
<10>	EMC	R/W	0	0	Enter modified compliance. When this bit is set to 1, the device transmits a modified compliance pattern if the LTSSM enters Polling.Compliance state.
<9:7>	TM	R/W/H	0x0	0x0	Transmit margin. This field controls the value of the non-de-emphasized voltage level at the transmitter pins: 0x0 = 800–1200 mV for full swing 400–600 mV for half-swing 0x1–0x2 = Values must be monotonic with a non-zero slope 0x3 = 200–400 mV for full-swing and 100–200 mV for halfswing 0x4–0x7 = Reserved This field is reset to 0x0 on entry to the LTSSM polling compliance substate. When operating in 5.0 GT/s mode with full swing, the de-emphasis ratio must be maintained within ± 1 dB from the specification-defined operational value either –3.5 or –6 dB.
<6>	SDE	RO	0	0	Selectable de-emphasis. Not applicable for an upstream port or endpoint device. Hardwired to 0.
<5>	HASD	RO/H	0	0	Hardware autonomous speed disable. When asserted, the application must disable hardware from changing the link speed for device-specific reasons other than attempting to correct unreliable link operation by reducing link speed. Initial transition to the highest supported common link speed is not blocked by this signal.
<4>	EC	R/W	0	0	Enter compliance. Software is permitted to force a link to enter compliance mode at the speed indicated in the target link speed field by setting this bit to 1 in both components on a link and then initiating a hot reset on the link.

Cavium Confidential For Review Only 02.17.2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<3:0>	TLS	R/W	—	0x2	Target link speed. For downstream ports, this field sets an upper limit on link operational speed by restricting the values advertised by the upstream component in its training sequences: 0x1 = 2.5 Gb/s target link speed 0x2 = 5 Gb/s target link speed 0x4 = 8Gb/s target link speed (not supported) All other encodings are reserved. If a value is written to this field that does not correspond to a speed included in the supported link speeds field, the result is undefined. For both upstream and downstream ports, this field is used to set the target compliance mode speed when software is using the enter compliance bit to force a link into compliance mode. The reset value of this field is controlled by a value sent from the TBD of the TBD register. **pi_sel_md[2] RST_VALUE NOTE** 1 0x1 2.5 GHz supported 0 0x2 5.0 GHz and 2.5 GHz supported

PCI Express Extended Capability Header Register PCIEEP(0..2)_CFG064

This register contains the sixty-fifth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	NCO	RO	0x148	0x148	Next capability offset.
<19:16>	CV	RO	0x2	0x2	Capability version
<15:0>	PCIEEC	RO	0x1	0x1	PCI Express extended capability

Uncorrectable Error Status Register PCIEEP(0..2)_CFG065

This register contains the sixty-sixth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:25>	—	RAZ	—	—	Reserved
<24>	UATOMBS	RO	0	0	Unsupported AtomicOp egress blocked status
<23>	—	RAZ	—	—	Reserved
<22>	UCIES	R/W	1	1	Uncorrectable internal error status
<21>	—	RAZ	—	—	Reserved
<20>	URES	R/W1C/H	0	0	Unsupported request error status
<19>	ECRCES	R/W1C/H	0	0	ECRC error status
<18>	MTLPS	R/W1C/H	0	0	Malformed TLP status
<17>	ROS	R/W1C/H	0	0	Receiver overflow status
<16>	UCS	R/W1C/H	0	0	Unexpected completion status
<15>	CAS	R/W1C/H	0	0	Completer abort status
<14>	CTS	R/W1C/H	0	0	Completion timeout status
<13>	FCPES	R/W1C/H	0	0	Flow control protocol error status
<12>	PTLPS	R/W1C/H	0	0	Poisoned TLP status

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<11:6>	—	RAZ	—	—	Reserved
<5>	SDES	RO	0	0	Surprise down error status (not supported)
<4>	DLPES	R/W1C/H	0	0	Data link protocol error status
<3:0>	—	RAZ	—	—	Reserved

Uncorrectable Error Mask Register PCIEEP(0..2)_CFG066

This register contains the sixty-seventh 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:25>	—	RAZ	—	—	Reserved
<24>	UATOMBM	RO	0	0	Unsupported AtomicOp egress blocked mask
<23>	—	RAZ	—	—	Reserved
<22>	UCIEM	R/W	1	1	Uncorrectable internal error mask
<21>	—	RAZ	—	—	Reserved
<20>	UREM	R/W	0	0	Unsupported request error mask
<19>	ECRCEM	R/W	0	0	ECRC error mask
<18>	MTLPM	R/W	0	0	Malformed TLP mask
<17>	ROM	R/W	0	0	Receiver overflow mask
<16>	UCM	R/W	0	0	Unexpected completion mask
<15>	CAM	R/W	0	0	Completer abort mask.
<14>	CTM	R/W	0	0	Completion timeout mask
<13>	FCPEM	R/W	0	0	Flow control protocol error mask
<12>	PTLPM	R/W	0	0	Poisoned TLP mask
<11:6>	—	RAZ	—	—	Reserved
<5>	SDES	RO	0	0	Surprise down error status (not supported)
<4>	DLPEM	R/W	0	0	Data link protocol error mask
<3:0>	—	RAZ	—	—	Reserved

Uncorrectable Error Severity Register PCIEEP(0..2)_CFG067

This register contains the sixty-eighth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:25>	—	RAZ	—	—	Reserved
<24>	UATOMBS	RO	0	0	Unsupported AtomicOp egress blocked severity
<23>	—	RAZ	0	0	Reserved
<22>	UCIES	R/W	1	1	Uncorrectable internal error severity
<21>	—	RAZ	0	0	Reserved
<20>	URES	R/W	0	0	Unsupported request error severity
<19>	ECRCES	R/W	0	0	ECRC error severity
<18>	MTLPS	R/W	1	1	Malformed TLP severity
<17>	ROS	R/W	1	1	Receiver overflow severity
<16>	UCS	R/W	0	0	Unexpected completion severity
<15>	CAS	R/W	0	0	Completer abort severity

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<14>	CTS	R/W	0	0	Completion timeout severity
<13>	FCPES	R/W	1	1	Flow control protocol error severity
<12>	PTLPS	R/W	0	0	Poisoned TLP severity
<11:6>	—	RO/H	—	—	Reserved
<5>	SDES	RO	1	1	Surprise down error status (not supported)
<4>	DLPES	R/W	1	1	Data link protocol error severity
<3:0>	—	RAZ	—	—	Reserved

Correctable Error Status Register PCIEEP(0..2)_CFG068

This register contains the sixty-ninth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:15>	—	RAZ	—	—	Reserved
<14>	CIES	R/W1C/H	0	0	Corrected internal error status
<13>	ANFES	R/W1C/H	0	0	Advisory non-fatal error status
<12>	RTTS	R/W1C/H	0	0	Replay timer timeout status
<11:9>	—	RAZ	—	—	Reserved
<8>	RNRS	R/W1C/H	0	0	REPLAY_NUM rollover status
<7>	BDLLPS	R/W1C/H	0	0	Bad DLLP status
<6>	BTLPS	R/W1C/H	0	0	Bad TLP status
<5:1>	—	RAZ	—	—	Reserved
<0>	RES	R/W1C/H	0	0	Receiver error status

Correctable Error Mask Register PCIEEP(0..2)_CFG069

This register contains the seventieth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:15>	—	RAZ	—	—	Reserved
<14>	CIEM	R/W	1	1	Corrected internal error mask
<13>	ANFEM	R/W	1	1	Advisory non-fatal error mask
<12>	RTTM	R/W	0	0	Replay timer timeout mask
<11:9>	—	RAZ	—	—	Reserved
<8>	RNRM	R/W	0	0	REPLAY_NUM rollover mask
<7>	BDLLPM	R/W	0	0	Bad DLLP mask
<6>	BTLPM	R/W	0	0	Bad TLP mask
<5:1>	—	RAZ	—	—	Reserved
<0>	REM	R/W	0	0	Receiver error mask

Cavium Confidential For Bowen Yang ALU 02/27/2014

Advanced Error Capabilities and Control Register PCIEEP(0..2)_CFG070

This register contains the seventy-first 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:9>	—	RAZ	—	—	Reserved
<8>	CE	R/W	0	0	ECRC check enable
<7>	CC	RO	1	1	ECRC check capable
<6>	GE	R/W	0	0	ECRC generation enable
<5>	GC	RO	1	1	ECRC generation capability
<4:0>	FEP	RO/H	0x0	0x0	First error pointer

Header Log Register 1 PCIEEP(0..2)_CFG071

This register contains the seventy-second 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DWORD1	RO/H	0x0	0x0	Header log register (first DWORD)

Header Log Register 2 PCIEEP(0..2)_CFG072

This register contains the seventy-third 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DWORD2	RO/H	0x0	0x0	Header log register (second DWORD)

Header Log Register 3 PCIEEP(0..2)_CFG073

This register contains the seventy-fourth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DWORD3	RO/H	0x0	0x0	Header log register (third DWORD)

Header Log Register 4 PCIEEP(0..2)_CFG074

This register contains the seventy-fifth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DWORD4	RO/H	0x0	0x0	Header log register (fourth DWORD)

Cavium Confidential For Bowen Yang ALU 02/27/2014

Resizable BAR (RBAR) Capability Header PCIEEP(0..2)_CFG082

This register contains the eighty-third 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	NCO	RO/H	0x0	0x0	Next capability offset
<19:16>	CV	RO	0x1	0x1	Capability version
<15:0>	PCIEEC	RO	0x15	0x15	PCI Express extended capability

Resizable BAR (RBAR) Capability Register PCIEEP(0..2)_CFG083

This register contains the eighty-fourth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:26>	—	RAZ	—	—	Reserved
<25:4>	SRS	RO	0xFFFFF	0xFFFFF	Supported resource sizes. The CN71XX advertises the maximum allowable BAR size (512GB – 0xF_FFFF) when the fus_bar2_size_conf is intact. When the fuse is blown, the CN71XX advertised a BAR size of 2TB (0x3F_FFFF). The BAR is disabled at runtime by writing all 0s to this field through PEM(0..2)_CFG_WR.
<3:0>	—	RAZ	—	—	Reserved

Resizable BAR (RBAR) Control Register PCIEEP(0..2)_CFG084

This register contains the eighty-fifth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:2>	—	RAZ	—	—	Reserved
<12:8>	RBARS	R/W	0x0	0x0	BAR size. The CN71XX advertises the minimum allowable BAR size of 0x0 (1MB) but will accept values as large as 0x15 (2TB).
<7:5>	NRBAR	RO	0x1	0x1	Number of resizable BARs.
<31:2>	—	RAZ	—	—	Reserved
<2:0>	RBARI	RO	0x4	0x4	Points to the BAR located at offset 0x18 (BAR2).

Cavium Confidential For Bowen Yang ALD 02/7/2014

Ack Latency Timer/Replay Timer Register PCIEEP(0..2)_CFG448

This register contains the four hundred forty-ninth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	RTL	R/W/H	0x308D	—	Replay time limit. The replay timer expires when it reaches this limit. The PCI Express bus initiates a replay upon reception of a nak or when the replay timer expires. This value is set correctly by the hardware out of reset or when the negotiated link width or payload size changes. If the user changes this value through a CSR write or by an EEPROM load, they should refer to the PCIe Specification for the correct value.
<15:0>	RTLTL	R/W/H	0x102F	—	Round trip latency time limit. The ack/nak latency timer expires when it reaches this limit. This value is set correctly by the hardware out of reset or when the negotiated link width or payload size changes. If the user changes this value through a CSR write or by an EEPROM load, they should refer to the PCIe Specification for the correct value.

Other Message Register PCIEEP(0..2)_CFG449

This register contains the four hundred fiftieth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	OMR	R/W	0xFFFFFFFF	—	Other message register. This register can be used for either of the following purposes: • To send a specific PCI Express message, the application writes the payload of the message into this register, then sets bit 0 of the Port Link Control Register to send the message. • To store a corruption pattern for corrupting the LCRC on all TLPs, the application places a 32-bit corruption pattern into this register and enables this function by setting bit 25 of the Port Link Control Register. When enabled, the transmit LCRC result is XORed with this pattern before inserting it into the packet.

Cavium Confidential For Bowen Yang @2014

Port Force Link Register
PCIEEP(0..2)_CFG450

This register contains the four hundred fifty-first 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	LPEC	R/W	0x7	0x7	Low power entrance count. The power management state waits this many clock cycles for the associated completion of a CfgWr to PCIE_CFG017 register, power state (PS) field register to go low-power. This register is intended for applications that do not let the PCI Express bus handle a completion for configuration request to the Power Management Control and Status (PCIE_CFG017) register.
<23:22>	—	RAZ	—	—	Reserved
<21:16>	LINK_STATE	R/W	0x0	0x0	Link state. The Link state that the PCIe bus is forced to when FORCE_LINK is set. State encoding: 0x0 = DETECT_QUIET 0x1 = DETECT_ACT 0x2 = POLL_ACTIVE 0x3 = POLL_COMPLIANCE 0x4 = POLL_CONFIG 0x5 = PRE_DETECT_QUIET 0x6 = DETECT_WAIT 0x7 = CFG_LINKWD_START 0x8 = CFG_LINKWD_ACEPT 0x9 = CFG_LANENUM_WAIT 0xA = CFG_LANENUM_ACEPT 0xB = CFG_COMPLETE 0xC = CFG_IDLE 0xD = RCVRY_LOCK 0xE = RCVRY_SPEED 0xF = RCVRY_RCVRCFG 0x10 = RCVRY_IDLE 0x11 = L0 0x12 = L0S 0x13 = L123_SEND_EIDLE 0x14 = L1_IDLE 0x15 = L2_IDLE 0x16 = L2_WAKE 0x17 = DISABLED_ENTRY 0x18 = DISABLED_IDLE 0x19 = DISABLED 0x1A = LPBK_ENTRY 0x1B = LPBK_ACTIVE 0x1C = LPBK_EXIT 0x1D = LPBK_EXIT_TIMEOUT 0x1E = HOT_RESET_ENTRY 0x1F = HOT_RESET
<15>	FORCE_LINK	R/W/H	0	0	Force Link. Forces the Link to the state specified by LINK_STATE. The Force Link pulse triggers Link renegotiation. **NOTE:** As the force Link is a pulse, writing a 1 to it does trigger the forced Link state event, even though reading it always returns a 0.
<14:12>	—	RAZ	—	—	Reserved
<11:8>	LINK_CMD	R/W	0x0	0x0	Link command. The Link command that the PCIe core is forced to transmit when FORCE_LINK = 1. Command encoding: 0x1 = PEM_SEND_IDLE 0x2 = PEM_SEND_EIDLE 0x3 = PEM_XMT_IN_EIDLE 0x4 = PEM_MOD_COMPL_PATTERN 0x5 = PEM_SEND_RCVR_DETECT_SEQ 0x6 = PEM_SEND_TS1 0x7 = PEM_SEND_TS2 0x8 = PEM_COMPLIANCE_PATTERN 0x9 = PEM_SEND_SDS 0xA = PEM_SEND_BEACON 0xB = PEM_SEND_N_FTS 0xC = PEM_NORM 0xD = PEM_SEND_SKP 0xE = PEM_SEND_EIES 0xF = PEM_SEND_EIES_SYM
<7:0>	LINK_NUM	RO/H	0x0	0x0	Link number. Not used for endpoint.

Cavium Confidential For Board Bring-up ALL 08/27/2014

Ack Frequency Register
PCIEEP(0..2)_CFG451

This register contains the four hundred fifty-second 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	—	RAZ	—	—	Reserved
<30>	EASPML1	R/W/H	0	0	Enter ASPM L1 without receive in L0s. Allow core to enter ASPM L1 even when link partner did not go to L0s (receive is not in L0s). When not set, core goes to ASPM L1 only after idle period, during which both receive and transmit are in L0s.
<29:27>	L1EL	R/W	0x3	0x3	L1 entrance latency. Values correspond to: 0x0 = 1 ms 0x4 = 16 ms 0x1 = 2 ms 0x5 = 32 ms 0x2 = 4 ms 0x6 = 64 ms 0x3 = 8 ms 0x7 = 64 ms
<26:24>	L0EL	R/W	0x3	0x3	L0s entrance latency. Values correspond to: 0x0 = 1 ms 0x4 = 5 ms 0x1 = 2 ms 0x5 = 6 ms 0x2 = 3 ms 0x6 = 7 ms 0x3 = 4 ms 0x7 = 7 ms
<23:16>	N_FTS_CC	R/W	0x30	0x30	N_FTS when common clock is used. The number of fast training sequence (FTS) ordered sets to be transmitted when transitioning from L0s to L0. The maximum number of FTS ordered sets that a component can request is 0xFF. **NOTE:** A value of 0x0 is not supported; a value of 0x0 can cause the LTSSM to go into the recovery state when exiting from L0s.
<15:8>	N_FTS	R/W	0x30	0x30	N_FTS. The number of fast training sequence (FTS) ordered sets to be transmitted when transitioning from L0s to L0. The maximum number of FTS ordered sets that a component can request is 0xFF. **NOTE:** A value of 0x0 is not supported; a value of 0x0 can cause the LTSSM to go into the recovery state when exiting from L0s.
<7:0>	ACK_FREQ	R/W	0x0	0x0	Ack frequency. The number of pending Acks specified here (up to 0xFF) before sending an Ack.

Cavium Confidential For Bowen Yang ALO 02/27/2014

Port Link Control Register
PCIEEP(0..2)_CFG452

This register contains the four hundred fifty-third 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:22>	—	RAZ	—	—	Reserved
<21:16>	LME	R/W	0x7	0x7	Link mode enable set as follows: 000001 (0x1): ×1 000011 (0x3): ×2 000111 (0x7): ×4 001111 (0xF): ×8 (not supported) 011111 (0x1F): ×16 (not supported) 111111 (0x3F): ×32 (not supported) This field indicates the maximum number of lanes supported by the PCIe port. The value can be set less than 0x7 to limit the number of lanes that the PCIe will attempt to use. If the value of 0x7 set by the hardware is not desired, this field can be programmed to a smaller value (i.e. EEPROM). See also PCIEEP(0..2)_CFG031[MLW]. **NOTE:** The value of this field does not indicate the number of lanes in use by the PCIe. This field sets the maximum number of lanes in the PCIe core that could be used. As per the PCIe specification, the PCIe core can negotiate a smaller link width, so all of ×4, ×2, and ×1 are supported when LME = 0xF, for example.
<15:12>	—	RAZ	—	—	Reserved
<11:8>	—	RO/H	0x1	0x1	Reserved
<7>	FLM	R/W/H	0	0	Fast link mode. Sets all internal timers to fast mode for simulation purposes. If during an EEPROM load, the first word loaded is 0xFFFFFFFF, the EEPROM load is terminated and this bit is set.
<6>	—	RAZ	—	—	Reserved
<5>	DLLLE	R/W	1	1	DLL link enable. Enables link initialization. If DLLLE = 0, the PCI Express bus does not transmit InitFC DLLPs and does not establish a link.
<4>	—	RAZ	—	—	Reserved
<3>	RA	R/W	0	0	Reset assert. Triggers a recovery and forces the LTSSM to the hot reset state (downstream port only).
<2>	LE	R/W	0	0	Loopback enable. Initiate loopback mode as a master. On a 0→1 transition, the PCIe core sends TS ordered sets with the loopback bit set to cause the link partner to enter into loopback mode as a slave. Normal transmission is not possible when LE = 1. To exit loopback mode, take the link through a reset sequence.
<1>	SD	R/W	0	0	Scramble disable. Setting this bit turns off data scrambling.
<0>	OMR	R/W	0	0	Other message request. When software writes a 1 to this bit, the PCI Express bus transmits the message contained in the Other Message register.

Cavium Confidential For Power Yang Only 2021

Lane Skew Register
PCIEEP(0..2)_CFG453

This register contains the four hundred fifty-fourth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	DLLD	R/W	0	0	Disable lane-to-lane deskew. Disables the internal lane-to-lane deskew logic.
<30:26>	—	RAZ	—	—	Reserved
<25>	ACK_NAK	R/W	0	0	Ack/Nak disable. Prevents the PCI Express bus from sending Ack and Nak DLLPs.
<24>	FCD	R/W	0	0	Flow control disable. Prevents the PCI Express bus from sending FC DLLPs.
<23:0>	ILST	R/W	0x0	0x0	Insert lane skew for transmit. Causes skew between lanes for test purposes. There are three bits per lane. The value is in units of one symbol time. For example, the value 0x2 for a lane forces a skew of two symbol times for that lane. The maximum skew value for any lane is 5 symbol times.

Symbol Number Register
PCIEEP(0..2)_CFG454

This register contains the four hundred fifty-fifth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	—	RAZ	—	—	Reserved
<23:19>	TMANLT	R/W	0x0	0x0	Timer modifier for Ack/Nak latency timer. Increases the timer value for the Ack/Nak latency timer, in increments of 64 clock cycles.
<18:14>	TMRT	R/W	0x8	0x8	Timer modifier for replay timer. Increases the timer value for the replay timer, in increments of 64 clock cycles.
<13:8>	—	RAZ	—	—	Reserved
<7:0>	MFUNCN	R/W	0x0	0x0	Max number of functions supported. Used for SR-IOV.

Symbol Timer/Filter Mask Register 1
PCIEEP(0..2)_CFG455

This register contains the four hundred fifty-sixth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	M_CFG0_FILT	R/W	0	0	Mask filtering of received configuration requests (RC mode only).
<30>	M_IO_FILT	R/W	0	0	Mask filtering of received I/O Requests (RC mode only).
<29>	MSG_CTRL	R/W	0	0	Message control. The application must not change this field.
<28>	M_CPL_ECRC_FILT	R/W	0	0	Mask ECRC error filtering for completions
<27>	M_ECRC_FILT	R/W	0	0	Mask ECRC error filtering
<26>	M_CPL_LEN_ERR	R/W	0	0	Mask length mismatch error for received completions
<25>	M_CPL_ATTR_ERR	R/W	0	0	Mask attributes mismatch error for received completions
<24>	M_CPL_TC_ERR	R/W	0	0	Mask traffic class mismatch error for received completions
<23>	M_CPL_FUN_ERR	R/W	0	0	Mask function mismatch error for received completions

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<22>	M_CPL_RID_ERR	R/W	0	0	Mask Requester ID mismatch error for received completions
<21>	M_CPL_TAG_ERR	R/W	0	0	Mask tag error rules for received completions
<20>	M_LK_FILT	R/W	0	0	Mask locked request filtering
<19>	M_CFG1_FILT	R/W	1	1	Mask Type 1 configuration request filtering.
<18>	M_BAR_MATCH	R/W	0	0	Mask BAR match filtering
<17>	M_POIS_FILT	R/W	1	1	Mask poisoned TLP filtering
<16>	M_FUN	R/W	0	0	Mask function
<15>	DFCWT	R/W	0	0	Disable FC watchdog timer
<14:11>	—	RAZ	—	—	Reserved
<10:0>	SKPIV	R/W	0x280	0x280	SKP interval value

Filter Mask Register 2
PCIEEP(0..2)_CFG456

This register contains the four hundred fifty-seventh 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:4>	—	RAZ	—	—	Reserved
<3>	M_HANDLE_FLUSH	R/W	0	0	Mask core filter to handle flush request
<2>	M_DABORT_4UCPL	R/W	0	0	Mask DLLP abort for unexpected CPL
<1>	M_VEND1_DRP	R/W	0	0	Mask vendor MSG Type 1 dropped silently
<0>	M_VEND0_DRP	R/W	0	0	Mask vendor MSG Type 0 dropped with UR error reporting

Debug Register 0
PCIEEP(0..2)_CFG458

This register contains the four hundred fifty-ninth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DBG_INFO_L32	RO/H	0x0	0x0	Debug info lower 32 bits

Debug Register 1
PCIEEP(0..2)_CFG459

This register contains the four hundred sixtieth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DBG_INFO_U32	RO/H	0x0	0x0	Debug info upper 32 bits

Cavium Confidential For Bowen Yang ALU 02/27/2014

Transmit Posted FC Credit Status Register PCIEEP(0..2)_CFG460

This register contains the four hundred sixty-first 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	—	RAZ	—	—	Reserved
<19:12>	TPHFCC	RO/H	0x0	0x0	Transmit posted header FC Credits. The posted header credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.
<11:0>	TPDFCC	RO/H	0x0	0x0	Transmit posted data FC credits. The posted data credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.

Transmit Non-posted FC Credit Status Register PCIEEP(0..2)_CFG461

This register contains the four hundred sixty-second 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	—	RAZ	—	—	Reserved
<19:12>	TCHFCC	RO/H	0x0	0x0	Transmit non-posted header FC credits. The non-posted header credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.
<11:0>	TCDFCC	RO/H	0x0	0x0	Transmit non-posted data FC credits. The non-posted data credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.

Transmit Completion FC Credit Status Register PCIEEP(0..2)_CFG462

This register contains the four hundred sixty-third 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	—	RAZ	—	—	Reserved
<19:12>	TCHFCC	RO/H	0x0	0x0	Transmit completion header FC credits. The completion header credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.
<11:0>	TCDFCC	RO/H	0x0	0x0	Transmit completion data FC credits. The completion data credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.

Cavium Confidential For Boven Yang A-L U 02/27/2014

Queue Status Register PCIEEP(0..2)_CFG463

This register contains the four hundred sixty-fourth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:3>	—	RAZ	—	—	Reserved
<2>	RQNE	RO/H	0	0	Received queue not empty. Indicates there is data in one or more of the receive buffers.
<1>	TRBNE	RO/H	0	0	Transmit retry buffer not empty. Indicates that there is data in the transmit retry buffer.
<0>	RTLPFCCNR	RO/H	0	0	Received TLP FC credits not returned. Indicates that the PCI Express bus has sent a TLP but has not yet received an UpdateFC DLLP indicating that the credits for that TLP have been restored by the receiver at the other end of the Link.

VC Transmit Arbitration Register 1 PCIEEP(0..2)_CFG464

This register contains the four hundred sixty-fifth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	WRR_VC3	RO	0x0	0x0	WRR weight for VC3
<23:16>	WRR_VC2	RO	0x0	0x0	WRR weight for VC2
<15:8>	WRR_VC1	RO	0x0	0x0	WRR weight for VC1
<7:0>	WRR_VC0	RO	0xF	0xF	WRR weight for VC0

VC Transmit Arbitration Register 2 PCIEEP(0..2)_CFG465

This register contains the four hundred sixty-sixth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	WRR_VC7	RO	0x0	0x0	WRR weight for VC7
<23:16>	WRR_VC6	RO	0x0	0x0	WRR weight for VC6
<15:8>	WRR_VC5	RO	0x0	0x0	WRR weight for VC5
<7:0>	WRR_VC4	RO	0x0	0x0	WRR weight for VC4

Cavium Confidential For Bowen Yang ALU 02/27/2014

VC0 Posted Receive Queue Control Register PCIEEP(0..2)_CFG466

This register contains the four hundred sixty-seventh 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	RX_QUEUE_ORDER	RO/WRSL	0	0	VC ordering for receive queues. Determines the VC ordering rule for the receive queues, used only in the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR: 0 = round robin. 1 = strict ordering, higher numbered VCs have higher priority. The application must not change this field.
<30>	TYPE_ORDERING	RO/WRSL	1	1	TLP type ordering for VC0. Determines the TLP type ordering rule for VC0 receive queues, used only in the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR: 0 = strict ordering for received TLPs: Posted, then completion, then non-posted. 1 = ordering of received TLPs follows the rules in PCI Express Base specification The application must not change this field.
<29:24>	—	RAZ	—	—	Reserved
<23:21>	QUEUE_MODE	RO/WRSL	0x2	0x2	VC0 posted TLP queue mode. The operating mode of the posted receive queue for VC0, used only in the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field. Only one bit can be set at a time: bit<23> = bypass bit<22> = cut-through bit<21> = store-and-forward
<20>	—	RAZ	—	—	Reserved
<19:12>	HEADER_CREDITS	RO/WRSL	0x20	0x20	VC0 posted header credits. The number of initial posted header credits for VC0, used for all receive queue buffer configurations. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<11:0>	DATA_CREDITS	RO/WRSL	0x40	0x40	VC0 posted data credits. The number of initial posted data credits for VC0, used for all receive queue buffer configurations. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Cavium Confidential For Bowen Yang ALL 0x2 0x64

VC0 Non-Posted Receive Queue Control Register PCIEEP(0..2)_CFG467

This register contains the four hundred sixty-eighth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	—	RAZ	—	—	Reserved
<23:21>	QUEUE_MODE	RO/WRSL	0x2	0x2	VC0 non-posted TLP queue mode. The operating mode of the non-posted receive queue for VC0, used only in the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. Only one bit can be set at a time: bit<23> = bypass bit<22> = cut-through bit<21> = store-and-forward The application must not change this field.
<20>	—	RAZ	—	—	Reserved
<19:12>	HEADER_CREDITS	RO/WRSL	0x1F	0x1F	VC0 non-posted header credits. The number of initial non-posted header credits for VC0, used for all receive queue buffer configurations. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<11:0>	DATA_CREDITS	RO/WRSL	0xD	0xD	VC0 non-posted data credits. The number of initial non-posted data credits for VC0, used for all receive queue buffer configurations. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

VC0 Completion Receive Queue Control Register PCIEEP(0..2)_CFG468

This register contains the four hundred sixty-ninth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	—	RAZ	—	—	Reserved
<23:21>	QUEUE_MODE	RO/WRSL	0x2	0x2	VC0 completion TLP queue mode. The operating mode of the completion receive queue for VC0, used only in the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. Only one bit can be set at a time: bit<23> = bypass bit<22> = cut-through bit<21> = store-and-forward The application must not change this field.
<20>	—	RAZ	—	—	Reserved
<19:12>	HEADER_CREDITS	RO/WRSL	0x0	0x0	VC0 completion header credits. The number of initial completion header credits for VC0, used for all receive queue buffer configurations. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Cavium Confidential For Bowen Yang ALU 12/7/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<11:0>	DATA_CREDITS	RO/WRSL	0x0	0x0	VC0 completion data credits. The number of initial completion data credits for VC0, used for all receive queue buffer configurations. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

VC0 Posted Buffer Depth Register PCIEEP(0..2)_CFG490

This register contains the four hundred ninety-first 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:26>	—	RAZ	—	—	Reserved
<25:16>	HEADER_DEPTH	RO/WRSL	0x26	0x26	VC0 posted header queue depth. Sets the number of entries in the posted header queue for VC0 when using the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<15:14>	—	RAZ	—	—	Reserved
<13:0>	DATA_DEPTH	RO/WRSL	0xC8	0xC8	VC0 posted data queue depth. Sets the number of entries in the posted data queue for VC0 when using the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

VC0 Non-Posted Buffer Depth Register PCIEEP(0..2)_CFG491

This register contains the four hundred ninety-second 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:26>	—	RAZ	—	—	Reserved
<25:16>	HEADER_DEPTH	RO/WRSL	0x25	0x25	VC0 non-posted header queue depth. Sets the number of entries in the non-posted header queue for VC0 when using the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<15:14>	—	RAZ	—	—	Reserved
<13:0>	DATA_DEPTH	RO/WRSL	0x61	0x61	VC0 non-posted data queue depth. Sets the number of entries in the non-posted data queue for VC0 when using the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Cavium Confidential For Bowen Yang ALU 12/27/2014

VC0 Completion Buffer Depth Register PCIEEP(0..2)_CFG492

This register contains the four hundred ninety-third 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:26>	—	RAZ	—	—	Reserved
<25:16>	HEADER_DEPTH	RO/WRSL	0x66	0x66	VC0 completion header queue depth. Sets the number of entries in the completion header queue for VC0 when using the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<15:14>	—	RAZ	—	—	Reserved
<13:0>	DATA_DEPTH	RO/WRSL	0x18E	0x18E	VC0 completion data queue depth. Sets the number of entries in the completion data queue for VC0 when using the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Port Logic Register (Gen2) PCIEEP(0..2)_CFG515

This register contains the five hundred sixteenth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:21>	—	RAZ	—	—	Reserved
<20>	S_D_E	R/W	0	0	SEL_DE_EMPHASIS. Used to set the de-emphasis level for upstream ports.
<19>	CTCRB	R/W	0	0	Configuration TX compliance receive bit. When set to 1, signals LTSSM to transmit TS ordered sets with the compliance receive bit assert (equal to 1).
<18>	CPYTS	R/W	0	0	Configuration PHY TX swing. Indicates the voltage level that the PHY should drive. When set to 1, indicates full swing. When set to 0, indicates low swing.
<17>	DSC	R/W/H	0	0	Directed speed change. Writing 1 initiates a speed change; always reads as 0.
<16:8>	LE	R/W	0x4	0x4	Lane enable. Indicates the number of lanes to check for exit from electrical idle in Polling.Active and Polling.Compliance. 0x1 = ×1, 0x2 = ×2, etc. Used to limit the maximum link width to ignore broken lanes that detect a receiver, but will not exit electrical idle and would otherwise prevent a valid link from being configured.
<7:0>	N_FTS	R/W	0x30	0x30	N_FTS. Sets the number of fast training sequences (N_FTS) that the core advertises as its N_FTS during GEN2 Link training. This value is used to inform the Link partner about the PHY's ability to recover synchronization after a low power state. **NOTE:** Do not set N_FTS to 0x0; doing so can cause the LTSSM to go into the recovery state when exiting from L0s.

Cavium Confidential For Bowen Yang 02/27/2015

PHY Status Register
PCIEEP(0..2)_CFG516

This register contains the five hundred seventeenth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	PHY_STAT	RO/H	0x0	0x0	PHY status

PHY Control Register
PCIEEP(0..2)_CFG517

This register contains the five hundred eighteenth 32-bits of type 0 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	PHY_CTRL	R/W	0x0	0x0	PHY control

BAR 0 Mask Register - Low
PCIEEP(0..2)_CFG004_MASK

This register is invisible to host software and not readable from the application. It is a write-only register and is accessible via the PEM(0..2)_CFG_WR registers. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:1>	LMASK	WORSL/H	0x1FFF	0x1FFF	BAR mask low.
<0>	ENB	WORSL/H	1	1	BAR enable: 0 = BAR 0 is disabled, 1 = BAR 0 is enabled. Bit<0> is interpreted as BAR enable when writing to the BAR mask register rather than as a mask bit because bit<0> of a BAR is always masked from writing by host software. Bit<0> must be written prior to writing the other mask bits.

BAR 0 Mask Register - High
PCIEEP(0..2)_CFG005_MASK

This register is invisible to host software and not readable from the application. It is a write-only register and is accessible via the PEM(0..2)_CFG_WR registers. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	UMASK	WORSL/H	0x0	0x0	BAR mask high.

Cavium Confidential For Bowen Yang ALI 02/27/2014

BAR 1 Mask Register - Low
PCIEEP(0..2)_CFG006_MASK

This register is invisible to host software and not readable from the application. It is a write-only register and is accessible via the PEM(0..2)_CFG_WR registers. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:1>	LMASK	WORSL/H	0x1FFFFFF	0x1FFFFFF	BAR mask low
<0>	ENB	WORSL/H	1	1	BAR enable: 0 = BAR 1 is disabled, 1 = BAR 1 is enabled. Bit<0> is interpreted as BAR enable when writing to the BAR mask register rather than as a mask bit because bit<0> of a BAR is always masked from writing by host software. Bit<0> must be written prior to writing the other mask bits.

BAR 1 Mask Register - High
PCIEEP(0..2)_CFG007_MASK

This register is invisible to host software and not readable from the application. It is a write-only register and is accessible via the PEM(0..2)_CFG_WR registers. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	UMASK	WORSL/H	0x0	0x0	BAR mask high

BAR 2 Mask Register - Low
PCIEEP(0..2)_CFG008_MASK

This register is invisible to host software and not readable from the application. It is a write-only register and is accessible via the PEM(0..2)_CFG_WR registers. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:1>	LMASK	WORSL/H	0x7FFFFFFF	0x7FFFFFFF	BAR mask low.
<0>	ENB	WORSL/H	1	1	BAR enable: 0 = BAR 2 is disabled, 1 = BAR 2 is enabled. Bit<0> is interpreted as BAR enable when writing to the BAR mask register rather than as a mask bit because bit<0> of a BAR is always masked from writing by host software. Bit<0> must be written prior to writing the other mask bits.

BAR 2 Mask Register - High
PCIEEP(0..2)_CFG009_MASK

This register is invisible to host software and not readable from the application. It is a write-only register and is accessible via the PEM(0..2)_CFG_WR registers. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	UMASK	WORSL/H	0x1FF	0x1FF	BAR mask high

Cavium Confidential For Bowen Yang 3/27/2014

Expansion ROM BAR Mask
PCIEEP(0..2)_CFG012_MASK

This register is invisible to host software and not readable from the application. It is a write-only register and is accessible via the PEM(0..2)_CFG_WR registers. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:1>	MASK	WORSL/H	0x7FFF	0x7FFF	BAR mask low
<0>	ENB	WORSL/H	1	1	BAR enable: 0 = BAR ROM is disabled; 1 = BAR ROM is enabled. Bit<0> is interpreted as BAR enable when writing to the BAR mask register rather than as a mask bit because bit<0> of a BAR is always masked from writing by host software. Bit<0> must be written prior to writing the other mask bits.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

22.7.2 RC Mode PCIe Configuration Registers

This section describes the PCIe configuration registers available in RC mode. Note that the PCIERC(0..2)_CFG* CSRs (PCICONFIGRC) can only be accessed via the PEM(0..2)_CFG_WR/PEM(0..2)_CFG_RD registers; they cannot be accessed via the PCIe port. The ADDR field of the PEM(0..2)_CFG_WR/PEM(0..2)_CFG_RD registers must match the address listed in Table 22–4.

Table 22–4 PCICONFIGRC(0..2) Registers

Register	Address (PEM*_CFG_RD/ PEM*_CFG_WR ADDR Field)	CSR Type	Detailed Description
PCIERC(0..2)_CFG000	0x0000	PCICONFIGRC	See page 960
PCIERC(0..2)_CFG001	0x0004	PCICONFIGRC	See page 960
PCIERC(0..2)_CFG002	0x0008	PCICONFIGRC	See page 961
PCIERC(0..2)_CFG003	0x000C	PCICONFIGRC	See page 961
PCIERC(0..2)_CFG004	0x0010	PCICONFIGRC	See page 962
PCIERC(0..2)_CFG005	0x0014	PCICONFIGRC	See page 962
PCIERC(0..2)_CFG006	0x0018	PCICONFIGRC	See page 962
PCIERC(0..2)_CFG007	0x001C	PCICONFIGRC	See page 962
PCIERC(0..2)_CFG008	0x0020	PCICONFIGRC	See page 963
PCIERC(0..2)_CFG009	0x0024	PCICONFIGRC	See page 963
PCIERC(0..2)_CFG010	0x0028	PCICONFIGRC	See page 964
PCIERC(0..2)_CFG011	0x002C	PCICONFIGRC	See page 964
PCIERC(0..2)_CFG012	0x0030	PCICONFIGRC	See page 964
PCIERC(0..2)_CFG013	0x0034	PCICONFIGRC	See page 964
PCIERC(0..2)_CFG014	0x0038	PCICONFIGRC	See page 965
PCIERC(0..2)_CFG015	0x003C	PCICONFIGRC	See page 965
PCIERC(0..2)_CFG016	0x0040	PCICONFIGRC	See page 966
PCIERC(0..2)_CFG017	0x0044	PCICONFIGRC	See page 967
PCIERC(0..2)_CFG020	0x0050	PCICONFIGRC	See page 967
PCIERC(0..2)_CFG021	0x0054	PCICONFIGRC	See page 968
PCIERC(0..2)_CFG022	0x0058	PCICONFIGRC	See page 968
PCIERC(0..2)_CFG023	0x005C	PCICONFIGRC	See page 968
PCIERC(0..2)_CFG028	0x0070	PCICONFIGRC	See page 968
PCIERC(0..2)_CFG029	0x0074	PCICONFIGRC	See page 969
PCIERC(0..2)_CFG030	0x0078	PCICONFIGRC	See page 969
PCIERC(0..2)_CFG031	0x007C	PCICONFIGRC	See page 970
PCIERC(0..2)_CFG032	0x0080	PCICONFIGRC	See page 971
PCIERC(0..2)_CFG033	0x0084	PCICONFIGRC	See page 972
PCIERC(0..2)_CFG034	0x0088	PCICONFIGRC	See page 973
PCIERC(0..2)_CFG035	0x008C	PCICONFIGRC	See page 974
PCIERC(0..2)_CFG036	0x0090	PCICONFIGRC	See page 974
PCIERC(0..2)_CFG037	0x0094	PCICONFIGRC	See page 974
PCIERC(0..2)_CFG038	0x0098	PCICONFIGRC	See page 975
PCIERC(0..2)_CFG039	0x009C	PCICONFIGRC	See page 975
PCIERC(0..2)_CFG040	0x00A0	PCICONFIGRC	See page 975
PCIERC(0..2)_CFG041	0x00A4	PCICONFIGRC	See page 977
PCIERC(0..2)_CFG042	0x00A8	PCICONFIGRC	See page 977
PCIERC(0..2)_CFG064	0x0100	PCICONFIGRC	See page 978
PCIERC(0..2)_CFG065	0x0104	PCICONFIGRC	See page 978
PCIERC(0..2)_CFG066	0x0108	PCICONFIGRC	See page 978
PCIERC(0..2)_CFG067	0x010C	PCICONFIGRC	See page 979

Cavium Confidential Bowen Yang ALU 02/27/2014

Table 22–4 PCICONFIGRC(0..2) Registers

Register	Address (PEM*_CFG_RD/ PEM*_CFG_WR ADDR Field)	CSR Type	Detailed Description
PCIERC(0..2)_CFG068	0x0110	PCICONFIGRC	See page 979
PCIERC(0..2)_CFG069	0x0114	PCICONFIGRC	See page 980
PCIERC(0..2)_CFG070	0x0118	PCICONFIGRC	See page 980
PCIERC(0..2)_CFG071	0x011C	PCICONFIGRC	See page 980
PCIERC(0..2)_CFG072	0x0120	PCICONFIGRC	See page 981
PCIERC(0..2)_CFG073	0x0124	PCICONFIGRC	See page 981
PCIERC(0..2)_CFG074	0x0128	PCICONFIGRC	See page 981
PCIERC(0..2)_CFG075	0x012C	PCICONFIGRC	See page 981
PCIERC(0..2)_CFG076	0x0130	PCICONFIGRC	See page 981
PCIERC(0..2)_CFG077	0x0134	PCICONFIGRC	See page 982
PCIERC(0..2)_CFG448	0x0700	PCICONFIGRC	See page 982
PCIERC(0..2)_CFG449	0x0704	PCICONFIGRC	See page 983
PCIERC(0..2)_CFG450	0x0708	PCICONFIGRC	See page 984
PCIERC(0..2)_CFG451	0x070C	PCICONFIGRC	See page 985
PCIERC(0..2)_CFG452	0x0710	PCICONFIGRC	See page 986
PCIERC(0..2)_CFG453	0x0714	PCICONFIGRC	See page 987
PCIERC(0..2)_CFG454	0x0718	PCICONFIGRC	See page 987
PCIERC(0..2)_CFG455	0x071C	PCICONFIGRC	See page 988
PCIERC(0..2)_CFG456	0x0720	PCICONFIGRC	See page 988
PCIERC(0..2)_CFG458	0x0728	PCICONFIGRC	See page 988
PCIERC(0..2)_CFG459	0x072C	PCICONFIGRC	See page 989
PCIERC(0..2)_CFG460	0x0730	PCICONFIGRC	See page 989
PCIERC(0..2)_CFG461	0x0734	PCICONFIGRC	See page 989
PCIERC(0..2)_CFG462	0x0738	PCICONFIGRC	See page 989
PCIERC(0..2)_CFG463	0x073C	PCICONFIGRC	See page 990
PCIERC(0..2)_CFG464	0x0740	PCICONFIGRC	See page 990
PCIERC(0..2)_CFG465	0x0744	PCICONFIGRC	See page 990
PCIERC(0..2)_CFG466	0x0748	PCICONFIGRC	See page 991
PCIERC(0..2)_CFG467	0x074C	PCICONFIGRC	See page 992
PCIERC(0..2)_CFG468	0x0750	PCICONFIGRC	See page 992
PCIERC(0..2)_CFG490	0x07A8	PCICONFIGRC	See page 993
PCIERC(0..2)_CFG491	0x07AC	PCICONFIGRC	See page 993
PCIERC(0..2)_CFG492	0x07B0	PCICONFIGRC	See page 993
PCIERC(0..2)_CFG515	0x080C	PCICONFIGRC	See page 994
PCIERC(0..2)_CFG516	0x0810	PCICONFIGRC	See page 994
PCIERC(0..2)_CFG517	0x0814	PCICONFIGRC	See page 995

Cavium Confidential For Bowen Yang ALU 02/27/2024

Device ID and Vendor ID Register PCIERC(0..2)_CFG000

This register contains the first 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	DEVID	R/W	0x96	0x96	Device ID for CN71XX, writable through PEM(0..2)_CFG_WR. However, the application must not change this field. The possible values are: CN63XX = 0x90 CN78XX = 0x95 CN66XX = 0x92 CN71/70XX = 0x96 CN68XX = 0x91 CN61XX = 0x93 CNF71XX = 0x94
<15:0>	VENDID	R/W	0x177D	0x177D	Vendor ID, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Command/Status Register PCIERC(0..2)_CFG001

This register contains the second 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	DPE	R/W1C/H	0	0	Detected parity error
<30>	SSE	R/W1C/H	0	0	Signaled system error
<29>	RMA	R/W1C/H	0	0	Received master abort
<28>	RTA	R/W1C/H	0	0	Received target abort
<27>	STA	R/W1C/H	0	0	Signaled target abort
<26:25>	DEVT	RO	0x0	0x0	DEVSEL timing. Not applicable for PCI Express. Hardwired to 0.
<24>	MDPE	R/W1C/H	0	0	Master data parity error
<23>	FBB	RO	0	0	Fast back-to-back capable. Not applicable for PCI Express. Hardwired to 0.
<22>	—	RO/H	—	—	Reserved
<21>	M66	RO	0	0	66 MHz capable. Not applicable for PCI Express. Hardwired to 0.
<20>	CL	RO	1	1	Capabilities list. Indicates presence of an extended capability item. Hardwired to 1.
<19>	I_STAT	RO	0	0	INT*x*# status
<18:11>	—	RAZ	—	—	Reserved
<10>	I_DIS	R/W	0	0	INT*x* assertion disable
<9>	FBBE	RO	0	0	Fast back-to-back transaction enable. Not applicable for PCI Express. Must be hardwired to 0.
<8>	SEE	R/W	0	0	SERR# enable
<7>	IDS_WCC	RO	0	0	IDSEL stepping/wait cycle control. Not applicable for PCI Express. Must be hardwired to 0.
<6>	PER	R/W	0	0	Parity error response
<5>	VPS	RO	0	0	VGA palette snoop. Not applicable for PCI Express. Must be hardwired to 0.
<4>	MWICE	RO	0	0	Memory write & invalidate command enable. Not applicable for PCI Express. Must be hardwired to 0.
<3>	SCSE	RO	0	0	Special cycle snooping enable. Not applicable for PCI Express. Must be hardwired to 0.

Cavium Confidential For Review Only Yang ALD 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<2>	ME	R/W	0	0	Bus master enable
<1>	MSAE	R/W	0	0	Memory space enable
<0>	ISAE	R/W	0	0	I/O space enable

Class Code/Revision ID Register PCIERC(0..2)_CFG002

This register contains the third 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	BCC	R/W	0xB	0xB	Base Class code, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<23:16>	SC	R/W	0x30	0x30	Subclass code, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<15:8>	PI	R/W	0x0	0x0	Programming interface, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<7:0>	RID	R/W	0x0	0x0	Revision ID, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

BIST, Header Type, Master Latency Timer, Cache Line Size Register PCIERC(0..2)_CFG003

This register contains the fourth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	BIST	RO	0x0	0x0	The BIST register functions are not supported. All 8 bits of the BIST register are hardwired to 0.
<23>	MFD	R/W	0	0	Multi function device. The multi function device bit is writable through PEM(0..2)_CFG_WR. However, this is a single function device. Therefore, the application must not write a 1 to this bit.
<22:16>	CHF	RO	0x1	0x1	Configuration header format. Hardwired to 0x1.
<15:8>	LT	RO	0x0	0x0	Master latency timer. Not applicable for PCI Express, hardwired to 0.
<7:0>	CLS	R/W	0x0	0x0	Cache line size. The cache line size register is R/W for legacy compatibility purposes and is not applicable to PCI Express device functionality.

Cavium Confidential For Bowen Yang ALU 04/28/2014

Base Address Register 0 Low
PCIERC(0..2)_CFG004

This register contains the fifth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	—	RO/H	—	—	Unused

Base Address Register 0 High
PCIERC(0..2)_CFG005

This register contains the sixth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	—	RO/H	—	—	Unused

Bus Number Register
PCIERC(0..2)_CFG006

This register contains the seventh 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	SLT	RO	0x0	0x0	Secondary latency timer. Not applicable to PCI Express, hardwired to 0x0.
<23:16>	SUBBNUM	R/W	0x0	0x0	Subordinate bus number
<15:8>	SBNUM	R/W	0x0	0x0	Secondary bus number
<7:0>	PBNUM	R/W	0x0	0x0	Primary bus number

I/O Base and I/O Limit/Secondary Status Register
PCIERC(0..2)_CFG007

This register contains the eighth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	DPE	R/W1C/H	0	0	Detected parity error
<30>	SSE	R/W1C/H	0	0	Signaled system error
<29>	RMA	R/W1C/H	0	0	Received master abort
<28>	RTA	R/W1C/H	0	0	Received target abort
<27>	STA	R/W1C/H	0	0	Signaled target abort
<26:25>	DEVT	RO	0x0	0x0	DEVSEL timing. Not applicable for PCI Express. Hardwired to 0.
<24>	MDPE	R/W1C/H	0	0	Master data parity error
<23>	FBB	RO	0	0	Fast back-to-back capable. Not applicable for PCI Express. Hardwired to 0.
<22>	—	RO/H	—	—	Reserved
<21>	M66	RO	0	0	66 MHz capable. Not applicable for PCI Express. Hardwired to 0.
<20:16>	—	RAZ	—	—	Reserved
<15:12>	LIO_LIMI	R/W	0x0	0x0	I/O space limit

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<11:9>	—	RAZ	—	—	Reserved
<8>	IO32B	RO	1	1	32-bit I/O space
<7:4>	LIO_BASE	R/W	0x0	0x0	I/O space base
<3:1>	—	RAZ	—	—	Reserved
<0>	IO32A	R/W	1	1	32-bit I/O space. 0 = 16-bit I/O addressing 1 = 32-bit I/O addressing This bit is writable through PEM(0..2)_CFG_WR. When the application writes to this bit through PEM(0..2)_CFG_WR, the same value is written to bit 8 of this register.

Memory Base and Memory Limit Register PCIERC(0..2)_CFG008

This register contains the ninth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	ML_ADDR	R/W	0x0	0x0	Memory limit address
<19:16>	—	RO/H	—	—	Reserved
<15:4>	MB_ADDR	R/W	0x0	0x0	Memory base address
<3:0>	—	RO/H	—	—	Reserved

Prefetchable Memory and Limit Register PCIERC(0..2)_CFG009

This register contains the tenth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	LMEM_LIMIT	R/W	0x0	0x0	Upper 12 bits of 32-bit prefetchable memory end address
<19:17>	—	RO/H	—	—	Reserved
<16>	MEM64B	RO	1	1	64-Bit memory addressing 0 = 32-bit memory addressing 1 = 64-bit memory addressing
<15:4>	LMEM_BASE	R/W	0x0	0x0	Upper 12 bits of 32-bit prefetchable memory start address
<3:1>	—	RO/H	—	—	Reserved
<0>	MEM64A	R/W	1	1	64-Bit memory addressing. 0 = 32-bit memory addressing 1 = 64-bit memory addressing This bit is writable through PEM(0..2)_CFG_WR. When the application writes to this bit through PEM(0..2)_CFG_WR, the same value is written to bit 16 of this register.

Cavium Confidential For Bowen Yang AT U 09/17/2014

Prefetchable Base Upper 32 Bits Register PCIERC(0..2)_CFG010

This register contains the eleventh 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	UMEM_BASE	R/W	0x0	0x0	Upper 32 bits of base address of prefetchable memory space. Used only when 64-bit prefetchable memory addressing is enabled.

Prefetchable Limit Upper 32 Bits Register PCIERC(0..2)_CFG011

This register contains the twelfth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	UMEM_LIMIT	R/W	0x0	0x0	Upper 32 bits of limit address of prefetchable memory space. Used only when 64-bit prefetchable memory addressing is enabled.

I/O Base and Limit Upper 16 Bits Register PCIERC(0..2)_CFG012

This register contains the thirteenth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	UIO_LIMIT	R/W	0x0	0x0	Upper 16 bits of I/O limit (if 32-bit I/O decoding is supported for devices on the secondary side).
<15:0>	UIO_BASE	R/W	0x0	0x0	Upper 16 bits of I/O base (if 32-bit I/O decoding is supported for devices on the secondary side).

Capability Pointer Register PCIERC(0..2)_CFG013

This register contains the fourteenth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:8>	—	RAZ	—	—	Reserved
<7:0>	CP	R/W	0x40	0x40	First capability pointer. Points to power management capability structure by default, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Cavium Confidential For Bowen Yang ALU 02/07/2014

Expansion ROM Base Address Register PCIERC(0..2)_CFG014

This register contains the fifteenth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	—	RAZ	—	—	Unused

Interrupt Line Register/Interrupt Pin/Bridge Control Register PCIERC(0..2)_CFG015

This register contains the sixteenth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<31:28>	—	RO/H	—	—	Reserved
<27>	DTSEES	RO	0	0	Discard timer SERR enable status. Not applicable to PCI Express, hardwired to 0.
<26>	DTS	RO	0	0	Discard timer status. Not applicable to PCI Express, hardwired to 0.
<25>	SDT	RO	0	0	Secondary discard timer. Not applicable to PCI Express, hardwired to 0.
<24>	PDT	RO	0	0	Primary discard timer. Not applicable to PCI Express, hardwired to 0.
<23>	FBBE	RO	0	0	Fast back-to-back transactions enable. Not applicable to PCI Express, hardwired to 0.
<22>	SBRST	R/W	0	0	Secondary bus reset. Hot reset. Causes TS1s with the hot reset bit to be sent to the link partner. When set, software should wait 2ms before clearing. The link partner normally responds by sending TS1s with the hot reset bit set, which will cause a link down event. Refer to “PCIe Link-Down Reset in RC Mode” section.
<21>	MAM	RO	0	0	Master abort mode. Not applicable to PCI Express, hardwired to 0.
<20>	VGA16D	R/W	0	0	VGA 16-bit decode
<19>	VGAE	R/W	0	0	VGA enable
<18>	ISAE	R/W	0	0	ISA enable
<17>	SEE	R/W	0	0	SERR enable
<16>	PERE	R/W	0	0	Parity error response enable
<15:8>	INTA	R/W	0x1	0x1	Interrupt pin. Identifies the legacy interrupt message that the device (or device function) uses. The Interrupt Pin register is writable through PEM(0..2)_CFG_WR. In a single-function configuration, only INTA is used. Therefore, the application must not change this field.
<7:0>	IL	R/W	0xFF	0xFF	Interrupt line

Cavium Confidential For Steven Yang ALU 2014

Power Management Capability ID/Power Management Next Item Pointer/Power Management Capabilities Register PCIERC(0..2)_CFG016

This register contains the seventeenth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:27>	PMES	R/W	0x0	0x0	PME_Support. Identifies the power states from which the PCI Express bus can generate PME Messages. A value of 0x0 for any bit indicates that the device (or function) is not capable of generating PME Messages while in that power state: Bit 11: If set, PME Messages can be generated from D0 Bit 12: If set, PME Messages can be generated from D1 Bit 13: If set, PME Messages can be generated from D2 Bit 14: If set, PME Messages can be generated from D3hot Bit 15: If set, PME Messages can be generated from D3cold The PME_Support field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<26>	D2S	R/W	0	0	D2 support, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<25>	D1S	R/W	0	0	D1 support, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<24:22>	AUXC	R/W	0x0	0x0	AUX current, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<21>	DSI	R/W	0	0	Device specific initialization (DSI), writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<20>	—	RAZ	—	—	Reserved
<19>	PME_CLOCK	RO	0	0	PME clock, hardwired to 0.
<18:16>	PMSV	R/W	0x3	0x3	Power management specification version, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<15:8>	NCP	R/W	0x50	0x50	Next capability pointer. Points to the MSI capabilities by default, writable through PEM(0..2)_CFG_WR.
<7:0>	PMCID	RO	0x1	0x1	Power management capability ID

Cavium Confidential Prel
Bowen Yang
ALU
02/27/2014

Power Management Control and Status Register PCIERC(0..2)_CFG017

This register contains the eighteenth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	PMDIA	RO	0x0	0x0	Data register for additional information (not supported)
<23>	BPCCEE	RO	0	0	Bus power/clock control enable, hardwired to 0.
<22>	BD3H	RO	0	0	B2/B3 support, hardwired to 0.
<21:16>	—	RAZ	—	—	Reserved
<15>	PMESS	R/W1C/H	0	0	PME status. Indicates whether or not a previously enabled PME event occurred.
<14:13>	PMEDSIA	RO	0x0	0x0	Data scale (not supported)
<12:9>	PMDS	RO	0x0	0x0	Data select (not supported)
<8>	PMEENS	R/W	0	0	PME enable. A value of 1 indicates that the device is enabled to generate PME.
<7:4>	—	RAZ	—	—	Reserved
<3>	NSR	R/W	0	0	No soft reset, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<2>	—	RAZ	—	—	Reserved
<1:0>	PS	R/W	0x0	0x0	Power state. Controls the device power state: 00: D0 01: D1 10: D2 11: D3 The written value is ignored if the specific state is not supported.

MSI Capability ID/MSI Next Item Pointer/MSI Control Register PCIERC(0..2)_CFG020

This register contains the twenty-first 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:25>	—	RAZ	—	—	Reserved
<24>	PVM	RO	0	0	Per-vector masking capable
<23>	M64	R/W	1	1	64-bit address capable, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<22:20>	MME	R/W	0x0	0x0	Multiple message enabled. Indicates that multiple message mode is enabled by system software. The number of messages enabled must be less than or equal to the multiple message capable (MMC) value.
<19:17>	MMC	R/W	0x0	0x0	Multiple message capable, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<16>	MSIEN	R/W	0	0	MSI enabled. When set, INTx must be disabled. This bit must never be set, as internal-MSI is not supported in RC mode. (Note that this has no effect on external MSI, which is commonly used in RC mode.)
<15:8>	NCP	R/W	0x70	0x70	Next capability pointer. Points to PCI Express capabilities by default, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<7:0>	MSICID	RO	0x5	0x5	MSI capability ID

Cavium Confidential For Bowen Yang ALU 6/27/2014

MSI Lower 32 Bits Address Register
PCIERC(0..2)_CFG021

This register contains the twenty-second 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:2>	LMSI	R/W	0x0	0x0	Lower 32-bit address
<1:0>	—	RAZ	—	—	Reserved

MSI Upper 32 Bits Address Register
PCIERC(0..2)_CFG022

This register contains the twenty-third 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	UMSI	R/W	0x0	0x0	Upper 32-bit address

MSI Data Register
PCIERC(0..2)_CFG023

This register contains the twenty-fourth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	—	RAZ	—	—	Reserved
<15:0>	MSIMD	R/W	0x0	0x0	MSI data. Pattern assigned by system software. Bits [4:0] are ORed with MSI_VECTOR to generate 32 MSI messages per function.

PCIe Capabilities/PCIe Capabilities List Register
PCIERC(0..2)_CFG028

This register contains the twenty-ninth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:30>	—	RAZ	—	—	Reserved
<29:25>	IMN	R/W	0x0	0x0	Interrupt message number. Updated by hardware, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<24>	SI	R/W	0	0	Slot implemented. This bit is writable through PEM(0..2)_CFG_WR. However, it must 0 for an endpoint device. Therefore, the application must not write a 1 to this bit.
<23:20>	DPT	RO	0x4	0x4	Device port type
<19:16>	PCIECV	RO	0x2	0x2	PCI Express capability version
<15:8>	NCP	RO/WRSL	0x0	0x0	Next capability pointer. Points to the MSI-X capability by default, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<7:0>	PCIEID	RO	0x10	0x10	PCI Express capability ID

Cavium Confidential For Bowen Yang ALU 02/27/2014

Device Capabilities Register PCIERC(0..2)_CFG029

This register contains the thirtieth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:28>	—	RAZ	—	—	Reserved
<27:26>	CSPLS	RO	0x0	0x0	Captured slot power limit scale. From message from RC, upstream port only.
<25:18>	CSPLV	RO	0x0	0x0	Captured slot power limit value. From message from RC, upstream port only.
<17:16>	—	RAZ	—	—	Reserved
<15>	RBER	R/W	1	1	Role-based error reporting, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<14>	—	RAZ	0	0	Undefined
<13:12>	—	RAZ	—	—	Undefined
<11:9>	EL1AL	R/W	0x0	0x0	Endpoint L1 acceptable latency, writable through PEM(0..2)_CFG_WR.
<8:6	EL0AL	R/W	0x0	0x0	Endpoint L0s acceptable latency, writable through PEM(0..2)_CFG_WR.
<5>	ETFS	R/W	0	0	Extended tag field supported. This bit is writable through PEM(0..2)_CFG_WR. However, the application must not write a 1 to this bit.
<4:3>	PFS	R/W	0x0	0x0	Phantom function supported. This field is writable through PEM(0..2)_CFG_WR. However, phantom function is not supported. Therefore, the application must not write any value other than 0x0 to this field.
<2:0>	MPSS	R/W	0x1	0x1	Max_Payload_Size supported, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Device Control/Device Status Register PCIERC(0..2)_CFG030

This register contains the thirty-first 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:22>	—	RAZ	—	—	Reserved
<21>	TP	RO	0	0	Transaction pending. Hard-wired to 0.
<20>	AP_D	RO	0	0	Aux power detected. Set to 1 if Aux power detected.
<19>	UR_D	R/W1C/H	0	0	Unsupported request detected. Errors are logged in this register regardless of whether or not error reporting is enabled in the device control register. UR_D occurs when we receive something unsupported. Unsupported requests are Nonfatal errors, so UR_D should cause NFE_D. Receiving a vendor-defined message should cause an unsupported request.
<18>	FE_D	R/W1C/H	0	0	Fatal error detected. Errors are logged in this register regardless of whether or not error reporting is enabled in the device control register. This field is set if we receive any of the errors in PCIERC(0..2)_CFG066 that has a severity set to Fatal. Malformed TLPs generally fit into this category.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<17>	NFE_D	R/W1C/H	0	0	Non-fatal error detected. Errors are logged in this register regardless of whether or not error reporting is enabled in the device control register. This field is set if we receive any of the errors in PCIERC(0..2)_CFG066 that has a severity set to Nonfatal and does NOT meet Advisory Nonfatal criteria, which most poisoned TLPs should.
<16>	CE_D	R/W1C/H	0	0	Correctable error detected. Errors are logged in this register regardless of whether or not error reporting is enabled in the device control register. This field is set if we receive any of the errors in PCIERC(0..2)_CFG068, for example, a Replay Timer Timeout. Also, it can be set if we get any of the errors in PCIERC(0..2)_CFG066 that has a severity set to Nonfatal and meets the Advisory Nonfatal criteria, which most ECRC errors should.
<15>	—	RAZ	—	—	Reserved
<14:12>	MRRS	R/W	0x2	0x2	Max read request size. 0x0 = 128 B 0x3 = 1024 B 0x1 = 256 B 0x4 = 2048 B 0x2 = 512 B 0x5 = 4096 B **NOTE:** SLI_S2M_PORT(0..2)_CTL[MRRS] and DPI_SLI_PRT(0..2)_CFG[MRRS] must also be set properly. SLI_S2M_PORT(0..2)_CTL[MRRS] and DPI_SLI_PRT(0..2)_CFG[MRRS] must not exceed the desired max read request size.
<11>	NS_EN	R/W	1	1	Enable no snoop
<10>	AP_EN	R/W	0	0	AUX power PM enable
<9>	PF_EN	R/W	0	0	Phantom function enable. This bit should never be set; PEM requests never use phantom functions.
<8>	ETF_EN	R/W	0	0	Extended tag field enable. This bit should never be set; PEM requests never use extended tags.
<7:5>	MPS	R/W	0x0	0x0	Max payload size. Legal values: 0x0 = 128 B, 0x1 = 256 B. Larger sizes are not supported. **NOTE:** Both PCI Express ports must be set to the same value for peer-to-peer to function properly. **NOTE:** DPI_SLI_PRT(0..2)_CFG[MPS] must be set to the same value as this field for proper functionality.
<4>	RO_EN	R/W	1	1	Enable relaxed ordering
<3>	UR_EN	R/W	0	0	Unsupported request reporting enable
<2>	FE_EN	R/W	0	0	Fatal error reporting enable
<1>	NFE_EN	R/W	0	0	Non-fatal error reporting enable
<0>	CE_EN	R/W	0	0	Correctable error reporting enable

Link Capabilities Register PCIERC(0..2)_CFG031

This register contains the thirty-second 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	PNUM	RO/WRSL	0x0	0x0	Port number, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<23>	—	RAZ	—	—	Reserved
<22>	ASPM	RO/WRSL	1	1	ASPM optionality compliance.
<21>	LBNC	RO/WRSL	1	1	Link bandwidth notification capability

Cavium Confidential For Review Only 6/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<20>	DLLARC	RO	1	1	Data Link Layer active reporting capable. Set to 1 for root complex devices and 0 for endpoint devices.
<19>	SDERC	RO	0	0	Surprise down error reporting capable. Not supported; hardwired to 0x0.
<18>	CPM	R/W	0	0	Clock power management. The default value is the value you specify during hardware configuration.
<17:15>	L1EL	R/W	0x1	0x1	L1 exit latency. The default value is the value you specify during hardware configuration.
<14:12>	L0EL	R/W	0x0	0x0	L0s exit latency. The default value is the value you specify during hardware configuration.
<11:10>	ASLPMS	R/W	0x3	0x3	Active state link PM support. The default value is the value you specify during hardware configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<9:4>	MLW	R/W/H	—	—	Maximum link width. The default value is the value you specify during hardware configuration (×1, ×2, or ×4), writable through PEM(0..2)_CFG_WR. PEM(0..2)_CFG.MD[1:0] RST_VALUE NOTE 00 0010b 2 lanes 01 0001b 1 lane 10 0100b 4 lanes
<3:0>	MLS	R/W/H	—	—	Maximum link speed. The reset value of this field is controlled by a value read from the MSB of PEM(0..2)_CFG[MD]. **MD[2] RST_VALUE NOTE** 1 0x1 2.5 GHz supported 0 0x2 5.0 GHz and 2.5 GHz supported This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Link Control/Link Status Register PCIERC(0..2)_CFG032

This register contains the thirty-third 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	LAB	R/W1C/H	0	0	Link autonomous bandwidth status. This bit is set to indicate that hardware has autonomously changed Link speed or width, without the port transitioning through DL_Down status, for reasons other than to attempt to correct unreliable Link operation.
<30>	LBM	R/W1C/H	0	0	Link bandwidth management status. This bit is set to indicate either of the following has occurred without the port transitioning through DL_DOWN status: • A link retraining has completed following a write of 1b to the Retrain Link bit • Hardware has changed the Link speed or width to attempt to correct unreliable Link operation, either through a LTSSM timeout of higher level process. This bit must be set if the physical layer reports a speed or width change was initiated by the downstream component that was not indicated as an autonomous change.
<29>	DLLA	RO/H	0	1	Data-link layer active

Cavium Confidential For Bowen Yang ALU 04/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<28>	SCC	R/W	1	0	Slot clock configuration. Indicates that the component uses the same physical reference clock that the platform provides on the connector. The default value is the value you select during hardware configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<27>	LT	RO/H	0	0	Link training
<26>	—	RAZ	—	—	Undefined
<25:20>	NLW	RO/H	0x1	0x4	Negotiated Link width. Set automatically by hardware after link initialization.
<19:16>	LS	RO/H	—	—	Current link speed. The encoded value specifies a bit location in the supported link speeds vector (in the Link Capabilities 2 register) that corresponds to the current link speed. 0x1: the negotiated link speed: 2.5 Gb/s 0x2: the negotiated link speed: 5.0 Gb/s 0x4: the negotiated link speed: 8.0 Gb/s (not supported)
<15:12>	—	RAZ	—	—	Reserved
<11>	LAB_INT_ENB	R/W	0	0	Link autonomous bandwidth interrupt enable. When set, enables the generation of an interrupt to indicate that the link autonomous bandwidth status bit has been set.
<10>	LBM_INT_ENB	R/W	0	0	Link bandwidth management interrupt enable.When set, enables the generation of an interrupt to indicate that the link bandwidth management status bit has been set.
<9>	HAWD	R/W	0	0	Hardware autonomous width disable (not supported).
<8>	ECPM	R/W	0	0	Enable clock power management. Hardwired to 0 if clock power management is disabled in the Link Capabilities register.
<7>	ES	R/W	0	0	Extended synch
<6>	CCC	R/W	0	0	Common clock configuration
<5>	RL	R/W	0	0	Retrain link
<4>	LD	R/W	0	0	Link disable
<3>	RCB	R/W	1	1	Read completion boundary (RCB), writable through PEM(0..2)_CFG_WR. However, the application must not change this field because an RCB of 64 bytes is not supported.
<2>	—	RAZ	—	—	Reserved
<1:0>	ASLPC	R/W	0x0	0x0	Active state Link PM control

Slot Capabilities Register PCIERC(0..2)_CFG033

This register contains the thirty-fourth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<31:19>	PS_NUM	R/W	0x0	0x0	Physical slot number, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<18>	NCCS	R/W	0	0	No command complete support, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<17>	EMIP	R/W	0	0	Electromechanical interlock present, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<16:15>	SP_LS	R/W	0x0	0x0	Slot power limit scale, writable through PEM(0..2)_CFG_WR.
<14:7>	SP_LV	R/W	0x0	0x0	Slot power limit value, writable through PEM(0..2)_CFG_WR.
<6>	HP_C	R/W	0	0	Hot-plug capable, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Cavium Confidential For Bowen Yang ALD 02/27/2014

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<5>	HP_S	R/W	0	0	Hot-plug surprise, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<4>	PIP	R/W	0	0	Power indicator present, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<3>	AIP	R/W	0	0	Attention indicator present, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<2>	MRLSP	R/W	0	0	MRL sensor present, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<1>	PCP	R/W	0	0	Power controller present, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<0>	ABP	R/W	0	0	Attention button present, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Slot Control/Slot Status Register PCIERC(0..2)_CFG034

This register contains the thirty-fifth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:25>	—	RAZ	—	—	Reserved
<24>	DLLS_C	R/W1C/H	0	0	Data link layer state changed
<23>	EMIS	RO	0	0	Electromechanical interlock status
<22>	PDS	RO	1	1	Presence detect state
<21>	MRLSS	RO	0	0	MRL sensor state
<20>	CCINT_D	R/W1C/H	0	0	Command completed
<19>	PD_C	R/W1C/H	0	0	Presence detect changed
<18>	MRLS_C	R/W1C/H	0	0	MRL sensor changed
<17>	PF_D	R/W1C/H	0	0	Power fault detected
<16>	ABP_D	R/W1C/H	0	0	Attention button pressed
<15:13>	—	RAZ	—	—	Reserved
<12>	DLLS_EN	R/W	0	0	Data link layer state changed enable.
<11>	EMIC	R/W	0	0	Electromechanical interlock control
<10>	PCC	R/W	0	0	Power controller control
<9:8>	PIC	R/W	0x3	0x3	Power indicator control
<7:6>	AIC	R/W	0x3	0x3	Attention indicator control
<5>	HPINT_EN	R/W	0	0	Hot-plug interrupt enable
<4>	CCINT_EN	R/W	0	0	Command completed interrupt enable
<3>	PD_EN	R/W	0	0	Presence detect changed enable
<2>	MRLS_EN	R/W	0	0	MRL sensor changed enable
<1>	PF_EN	R/W	0	0	Power fault detected enable
<0>	ABP_EN	R/W	0	0	Attention button pressed enable

Root Control/Root Capabilities Register

Cavium Confidential For Bowen Yang AU 01/27/2014

PCIERC(0..2)_CFG035

This register contains the thirty-sixth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:17>	—	RAZ	—	—	Reserved
<16>	CRSSV	RO	0	0	CRS software visibility. Not supported, hardwired to 0.
<15:5>	—	RAZ	—	—	Reserved
<4>	CRSSVE	RO	0	0	CRS software visibility enable. Not supported, hardwired to 0.
<3>	PMEIE	R/W	0	0	PME interrupt enable.
<2>	SEFEE	R/W	0	0	System error on fatal error enable.
<1>	SENFEE	R/W	0	0	System error on non-fatal error enable.
<0>	SECEE	R/W	0	0	System error on correctable error enable.

Root Status Register PCIERC(0..2)_CFG036

This register contains the thirty-seventh 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:18>	—	RAZ	—	—	Reserved
<17>	PME_PEND	RO	0	0	PME pending
<16>	PME_STAT	R/W1C/H	0	0	PME status
<15:0>	PME_RID	RO	0x0	0x0	PME requester ID

Device Capabilities Register 2 PCIERC(0..2)_CFG037

This register contains the thirty-eighth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:14>	—	RAZ	—	—	Reserved
<13:12>	TPH	RO	0x0	0x0	TPH completer supported (not supported)
<11>	LTRS	RO	0	0	Latency tolerance reporting (LTR) mechanism supported (not supported)
<10>	NOROPRPR	RO/H	1	1	No RO-enabled PR-PR passing. When set, the routing element never carries out the passing permitted in the relaxed ordering model.
<9>	ATOM128S	RO	0	0	128-bit AtomicOp supported (not supported)
<8>	ATOM64S	RO	0	0	64-bit AtomicOp supported
<7>	ATOM32S	RO	0	0	32-bit AtomicOp supported
<6>	ATOM_OPS	RO	0	0	AtomicOp routing supported
<5>	ARI_FW	RO	0	0	Alternate routing ID forwarding supported
<4>	CTDS	RO	1	1	Completion timeout disable supported
<3:0>	CTRS	RO/H	0xF	0xF	Completion timeout ranges supported

Device Control Register 2 Register

Cavium Confidential For Bowen Yang ALU 6/27/2014

PCIERC(0..2)_CFG038

This register contains the thirty-ninth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:10>	—	RAZ	—	—	Reserved
<9>	ID0_CP	RO	0	0	ID based ordering completion enable (not supported)
<8>	ID0_RQ	RO	0	0	ID based ordering request enable (not supported)
<7>	ATOM_OP_EB	RO	0	0	AtomicOp egress blocking (not supported)
<6>	ATOM_OP	RO	0	0	AtomicOp requester enable (not supported)
<5>	ARI	RO	0	0	Alternate routing ID forwarding supported (not supported)
<4>	CTD	R/W	0	0	Completion timeout disable
<3:0>	CTV	RO/H	0x0	0x0	Completion timeout value. 0x0 = 50 µS – 55 ms (default) 0x1 = 50 – 100 µs 0x2 = 1 – 10 ms 0x5 = 16 – 55 ms 0x6 = 65 – 210 ms 0x9 = 260 – 900 ms 0xA = 1 – 3.5 s 0xD = 4 – 13 s 0xE = 17 – 64 s Values not defined are reserved

Link Capabilities Register 2
PCIERC(0..2)_CFG039

This register contains the fortieth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:9>	—	RAZ	—	—	Reserved
<8>	CLS	RO	0	0	Crosslink supported
<7:1>	SLSV	RO/WRSL	—	—	Supported link speeds vector. Indicates the supported link speeds of the associated port. For each bit, a value of 1 indicates that the corresponding link speed is supported; otherwise, the link speed is not supported. Bit definitions are: Bit <1> = 2.5 GT/s Bit <2> = 5.0 GT/s Bit <3> = 8.0 GT/s (not supported) Bits <7:4> are reserved The reset value of this field is controlled by a value read from the MSB of the pi_select_mode pins. **pi_sel_md[2] / RST_VALUE / NOTE** 1 / 0x1 / 2.5 GHz supported 0 / 0x3 / 5.0 GHz and 2.5 GHz supported
<0>	—	RAZ	—	—	Reserved

Link Control Register 2/Link Status Register 2
PCIERC(0..2)_CFG040

This register contains the forty-first 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:17>	—	RAZ	—	—	Reserved

Cavium Confidential For Boyan Yang 02/1/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<16>	CDL	RO/H	1	1	Current de-emphasis level. When the link is operating at 5 GT/s speed, this bit reflects the level of de-emphasis. Encodings: 1 = –3.5 dB 0 = –6 dB **NOTE:** The value in this bit is undefined when the link is operating at 2.5 GT/s speed.
<15:13>	—	RAZ	—	—	Reserved
<12>	CDE	R/W	0	0	Compliance de-emphasis. This bit sets the de-emphasis level in Polling.Compliance state if the entry occurred due to the Tx compliance receive bit being 1. Encodings: 1 = –3.5 dB 0 = –6 dB **NOTE:** When the link is operating at 2.5 GT/s, the setting of this bit has no effect.
<11>	CSOS	R/W	0	0	Compliance SOS. When set to 1, the LTSSM is required to send SKP ordered sets periodically in between the (modified) compliance patterns. **NOTE:** When the link is operating at 2.5 GT/s, the setting of this bit has no effect.
<10>	EMC	R/W	0	0	Enter modified compliance. When this bit is set to 1, the device transmits a modified compliance pattern if the LTSSM enters Polling.Compliance state.
<9:7>	TM	R/W/H	0x0	0x0	Transmit margin. This field controls the value of the non-de-emphasized voltage level at the transmitter pins: 0x0 = 800–1200 mV for full swing 400–600 mV for half-swing 0x1–0x2 = values must be monotonic with a non-zero slope 0x3 = 200–400 mV for full-swing and 100–200 mV for halfswing 0x4–0x7 = reserved This field is reset to 0x0 on entry to the LTSSM Polling.Compliance substate. When operating in 5.0 GT/s mode with full swing, the de-emphasis ratio must be maintained within ± 1 dB from the specification-defined operational value either –3.5 or –6 dB.
<6>	SDE	R/W	0	0	Selectable de-emphasis. When the link is operating at 5.0 GT/s speed, selects the level of de-emphasis: 1 = –3.5 dB 0 = –6 dB When the link is operating at 2.5 GT/s speed, the setting of this bit has no effect.
<5>	HASD	R/W	0	0	Hardware autonomous speed disable. When asserted, the application must disable hardware from changing the link speed for device-specific reasons other than attempting to correct unreliable link operation by reducing link speed. Initial transition to the highest supported common link speed is not blocked by this signal.
<4>	EC	R/W	0	0	Enter compliance. Software is permitted to force a link to enter compliance mode at the speed indicated in the target link speed field by setting this bit to 1 in both components on a link and then initiating a hot reset on the link.

Cavium Confidential For Bowen Fang ASIC 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<3:0>	TLS	R/W	—	—	Target link speed. For downstream ports, this field sets an upper limit on link operational speed by restricting the values advertised by the upstream component in its training sequences: 0001 = 2.5Gb/s Target Link Speed 0010 = 5Gb/s Target Link Speed 0100 = 8Gb/s Target Link Speed All other encodings are reserved. If a value is written to this field that does not correspond to a speed included in the supported link speeds field, the result is undefined. For both upstream and downstream ports, this field is used to set the target compliance mode speed when software is using the enter compliance bit to force a link into compliance mode. The reset value of this field is controlled by a value read from the MSB of the pi_select_mode pins. **pi_sel_md[2] RST_VALUE NOTE** 1 0x1 2.5 GHz supported 0 0x2 5.0 GHz and 2.5 GHz supported

Slot Capabilities Register 2 PCIERC(0..2)_CFG041

This register contains the forty-second 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	—	RAZ	—	—	Reserved

Slot Control Register 2/Slot Status Register 2 PCIERC(0..2)_CFG042

This register contains the forty-third 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang ALU 02/27/2014

PCI Express Extended Capability Header Register PCIERC(0..2)_CFG064

This register contains the sixty-fifth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	NCO	RO	0x0	0x0	Next capability offset.
<19:16>	CV	RO	0x2	0x2	Capability version
<15:0>	PCIEEC	RO	0x1	0x1	PCI Express extended capability

Uncorrectable Error Status Register PCIERC(0..2)_CFG065

This register contains the sixty-sixth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:25>	—	RAZ	—	—	Reserved
<24>	UATOMBS	RO	0	0	Unsupported AtomicOp egress blocked status
<23>	—	RAZ	—	—	Reserved
<22>	UCIES	R/W1C/H	1	1	Uncorrectable internal error status
<21>	—	RAZ	—	—	Reserved
<20>	URES	R/W1C/H	0	0	Unsupported request error status
<19>	ECRCES	R/W1C/H	0	0	ECRC error status
<18>	MTLPS	R/W1C/H	0	0	Malformed TLP status
<17>	ROS	R/W1C/H	0	0	Receiver overflow status
<16>	UCS	R/W1C/H	0	0	Unexpected completion status
<15>	CAS	R/W1C/H	0	0	Completer abort status
<14>	CTS	R/W1C/H	0	0	Completion timeout status
<13>	FCPES	R/W1C/H	0	0	Flow control protocol error status
<12>	PTLPS	R/W1C/H	0	0	Poisoned TLP status
<11:6>	—	RAZ	—	—	Reserved
<5>	SDES	RO	0	0	Surprise down error status (not supported)
<4>	DLPES	R/W1C/H	0	0	Data Link Protocol error status
<3:0>	—	RAZ	—	—	Reserved

Uncorrectable Error Mask Register PCIERC(0..2)_CFG066

This register contains the sixty-seventh 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:25>	—	RAZ	—	—	Reserved
<24>	UATOMBM	RO	0	0	Unsupported AtomicOp egress blocked status.
<23>	—	RAZ	—	—	Reserved
<22>	UCIEM	R/W	1	1	Uncorrectable internal error mask
<21>	—	RAZ	—	—	Reserved
<20>	UREM	R/W	0	0	Unsupported request error mask
<19>	ECRCEM	R/W	0	0	ECRC error mask

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<18>	MTLPM	R/W	0	0	Malformed TLP mask
<17>	ROM	R/W	0	0	Receiver overflow mask
<16>	UCM	R/W	0	0	Unexpected completion mask
<15>	CAM	R/W	0	0	Completer abort mask
<14>	CTM	R/W	0	0	Completion timeout mask
<13>	FCPEM	R/W	0	0	Flow control protocol error mask
<12>	PTLPM	R/W	0	0	Poisoned TLP mask
<11:6>	—	RAZ	—	—	Reserved
<5>	SDEM	RO	0	0	Surprise down error mask (not supported)
<4>	DLPEM	R/W	0	0	Data link protocol error mask
<3:0>	—	RAZ	—	—	Reserved

Uncorrectable Error Severity Register PCIERC(0..2)_CFG067

This register contains the sixty-eighth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:25>	—	RAZ	—	—	Reserved
<24>	UATOMBS	RO	0	0	Unsupported AtomicOp egress blocked severity.
<23>	—	RAZ	—	—	Reserved
<22>	UCIES	R/W	1	1	Uncorrectable internal error severity
<21>	—	RAZ	—	—	Reserved
<20>	URES	R/W	0	0	Unsupported request error severity
<19>	ECRCES	R/W	0	0	ECRC error severity
<18>	MTLPS	R/W	1	1	Malformed TLP severity
<17>	ROS	R/W	1	1	Receiver overflow severity
<16>	UCS	R/W	0	0	Unexpected completion severity
<15>	CAS	R/W	0	0	Completer abort severity
<14>	CTS	R/W	0	0	Completion timeout severity
<13>	FCPES	R/W	1	1	Flow control protocol error severity
<12>	PTLPS	R/W	0	0	Poisoned TLP severity
<11:6>	—	RAZ	—	—	Reserved
<5>	SDES	RO	1	1	Surprise down error severity (not supported)
<4>	DLPES	R/W	1	1	Data link protocol error severity
<3:0>	—	RAZ	—	—	Reserved

Correctable Error Status Register PCIERC(0..2)_CFG068

This register contains the sixty-ninth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:15>	—	RAZ	—	—	Reserved
<14>	CIES	R/W1C	0	0	Corrected internal error status
<13>	ANFES	R/W1C/H	0	0	Advisory non-fatal error status
<12>	RTTS	R/W1C/H	0	0	Replay timer timeout status

Cavium Confidential For Power Yang 06/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<11:9>	—	RAZ	—	—	Reserved
<8>	RNRS	R/W1C/H	0	0	REPLAY_NUM rollover status
<7>	BDLLPS	R/W1C/H	0	0	Bad DLLP status
<6>	BTLPS	R/W1C/H	0	0	Bad TLP status
<5:1>	—	RAZ	—	—	Reserved
<0>	RES	R/W1C/H	0	0	Receiver error status

Correctable Error Mask Register PCIERC(0..2)_CFG069

This register contains the seventieth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<31:15>	—	RAZ	—	—	Reserved
<14>	CIEM	R/W	1	1	Corrected internal error mask
<13>	ANFEM	R/W	1	1	Advisory non-fatal error mask
<12>	RTTM	R/W	0	0	Replay timer timeout mask
<11:9>	—	RAZ	—	—	Reserved
<8>	RNRM	R/W	0	0	REPLAY_NUM rollover mask
<7>	BDLLPM	R/W	0	0	Bad DLLP mask
<6>	BTLPM	R/W	0	0	Bad TLP mask
<5:1>	—	RAZ	—	—	Reserved
<0>	REM	R/W	0	0	Receiver error mask

Advanced Capabilities and Control Register PCIERC(0..2)_CFG070

This register contains the seventy-first 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:9>	—	RAZ	—	—	Reserved
<8>	CE	R/W	0	0	ECRC check enable
<7>	CC	RO	1	1	ECRC check capable
<6>	GE	R/W	0	0	ECRC generation enable
<5>	GC	RO	1	1	ECRC generation capability
<4:0>	FEP	RO	0x0	0x0	First error pointer

Header Log Register 1 PCIERC(0..2)_CFG071

This register contains the seventy-second 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DWORD1	RO	0x0	0x0	Header log register (first DWORD)

Cavium Confidential For Bowen Yang ALU 02/27/2014

Header Log Register 2
PCIERC(0..2)_CFG072

This register contains the seventy-third 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DWORD2	RO	0x0	0x0	Header log register (second DWORD)

Header Log Register 3
PCIERC(0..2)_CFG073

This register contains the seventy-fourth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DWORD3	RO	0x0	0x0	Header log register (third DWORD)

Header Log Register 4
PCIERC(0..2)_CFG074

This register contains the seventy-fifth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DWORD4	RO	0x0	0x0	Header log register (fourth DWORD)

Root Error Command Register
PCIERC(0..2)_CFG075

This register contains the seventy-sixth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:3>	—	RAZ	—	—	Reserved
<2>	FERE	R/W	0	0	Fatal error reporting enable
<1>	NFERE	R/W	0	0	Non-fatal error reporting enable
<0>	CERE	R/W	0	0	Correctable error reporting enable

Root Error Status Register
PCIERC(0..2)_CFG076

This register contains the seventy-seventh 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:27>	AEIMN	R/W	0x0	0x0	Advanced error interrupt message number, writable through PEM(0..2)_CFG_WR.
<26:7>	—	RAZ	—	—	Reserved
<6>	FEMR	R/W1C/H	0	0	Fatal error messages received

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<5>	NFEMR	R/W1C/H	0	0	Non-fatal error messages received
<4>	FUF	R/W1C/H	0	0	First uncorrectable fatal
<3>	MULTI_EFNFR	R/W1C/H	0	0	Multiple ERR_FATAL/NONFATAL received
<2>	EFNFR	R/W1C/H	0	0	ERR_FATAL/NONFATAL received
<1>	MULTI_ECR	R/W1C/H	0	0	Multiple ERR_COR received
<0>	ECR	R/W1C/H	0	0	ERR_COR received

Error-Source Identification Register PCIERC(0..2)_CFG077

This register contains the seventy-eighth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	EFNFSI	RO	0x0	0x0	ERR_FATAL/NONFATAL source identification
<15:0>	ECSI	RO	0x0	0x0	ERR_COR source identification

Ack Latency Timer/Replay Timer Register PCIERC(0..2)_CFG448

This register contains the four hundred forty-ninth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	RTL	R/W/H	0x308D	—	Replay time limit. The replay timer expires when it reaches this limit. The PCI Express bus initiates a replay upon reception of a nak or when the replay timer expires. This value is set correctly by the hardware out of reset or when the negotiated link width or payload size changes. If the user changes this value through a CSR write or by an EEPROM load then they should refer to the PCIe Specification for the correct value.
<15:0>	RTLTL	R/W/H	0x102F	—	Round trip latency time limit. The ack/nak latency timer expires when it reaches this limit. This value is set correctly by the hardware out of reset or when the negotiated link width or payload size changes. If the user changes this value through a CSR write or by an EEPROM load, they should refer to the PCIe Specification for the correct value.

Cavium Confidential For Review Only 06/24/14

Other Message Register PCIERC(0..2)_CFG449

This register contains the four hundred fiftieth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	OMR	R/W	0xFFFFFFFF	—	Other message register. This register can be used for either of the following purposes: • To send a specific PCI Express message, the application writes the payload of the message into this register, then sets bit 0 of the Port Link Control Register to send the message. • To store a corruption pattern for corrupting the LCRC on all TLPs, the application places a 32-bit corruption pattern into this register and enables this function by setting bit 25 of the Port Link Control Register. When enabled, the transmit LCRC result is XORed with this pattern before inserting it into the packet.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Port Force Link Register
PCIERC(0..2)_CFG450

This register contains the four hundred fifty-first 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	LPEC	RO	0x7	0x7	Low power entrance count. The power management state waits this many clock cycles for the associated completion of a CfgWr to PCIE_CFG017 register, power state (PS) field register to go low-power. This register is intended for applications that do not let the PCI Express bus handle a completion for configuration request to the Power Management Control and Status (PCIE_CFG017) register.
<23:22>	—	RAZ	—	—	Reserved
<21:16>	LINK_STATE	R/W	0x0	0x0	Link state. The Link state that the PCI Express bus is forced to when bit 15 (Force Link) is set. Link state. State encoding: 0x0 = DETECT_QUIET 0x1 = DETECT_ACT 0x2 = POLL_ACTIVE 0x3 = POLL_COMPLIANCE 0x4 = POLL_CONFIG 0x5 = PRE_DETECT_QUIET 0x6 = DETECT_WAIT 0x7 = CFG_LINKWD_START 0x8 = CFG_LINKWD_ACEPT 0x9 = CFG_LANENUM_WAIT 0xA = CFG_LANENUM_ACEPT 0xB = CFG_COMPLETE 0xC = CFG_IDLE 0xD = RCVRY_LOCK 0xE = RCVRY_SPEED 0xF = RCVRY_RCVRCFG 0x10 = RCVRY_IDLE 0x11 = L0 0x12 = L0S 0x13 = L123_SEND_EIDLE 0x14 = L1_IDLE 0x15 = L2_IDLE 0x16 = L2_WAKE 0x17 = DISABLED_ENTRY 0x18 = DISABLED_IDLE 0x19 = DISABLED 0x1A = LPBK_ENTRY 0x1B = LPBK_ACTIVE 0x1C = LPBK_EXIT 0x1D = LPBK_EXIT_TIMEOUT 0x1E = HOT_RESET_ENTRY 0x1F = HOT_RESET
<15>	FORCE_LINK	R/W/H	0	0	Force link. Forces the Link to the state specified by the LINK_STATE field. The Force Link pulse triggers Link re-negotiation. **NOTE:** As the force Link is a pulse, writing a 1 to it does trigger the forced Link state event, even though reading it always returns a 0.
<14:12>	—	RAZ	—	—	Reserved
<11:8>	LINK_CMD	R/W	0x0	0x0	Link command. The Link command that the PCIe core is forced to transmit when FORCE_LINK = 1. Command encoding: 0x1 = PEM_SEND_IDLE 0x2 = PEM_SEND_EIDLE 0x3 = PEM_XMT_IN_EIDLE 0x4 = PEM_MOD_COMPL_PATTERN 0x5 = PEM_SEND_RCVR_DETECT_SEQ 0x6 = PEM_SEND_TS1 0x7 = PEM_SEND_TS2 0x8 = PEM_COMPLIANCE_PATTERN 0x9 = PEM_SEND_SDS 0xA = PEM_SEND_BEACON 0xB = PEM_SEND_N_FTS 0xC = PEM_NORM 0xD = PEM_SEND_SKP 0xE = PEM_SEND_EIES 0xF = PEM_SEND_EIES_SYM
<7:0>	LINK_NUM	R/W	0x4	0x4	Link number.

Cavium Confidential Ext
Bowen Ning ALU
2014

Ack Frequency Register
PCIERC(0..2)_CFG451

This register contains the four hundred fifty-second 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	—	RAZ	—	—	Reserved
<30>	EASPML1	R/W/H	0	0	Enter ASPM L1 without receive in L0s. Allow core to enter ASPM L1 even when link partner did not go to L0s (receive is not in L0s). When not set, core goes to ASPM L1 only after idle period during which both receive and transmit are in L0s.
<29:27>	L1EL	R/W	0x3	0x3	L1 entrance latency. Values correspond to: 000 = 1 ms 001 = 2 ms 010 = 4 ms 011 = 8 ms 100 = 16 ms 101 = 32 ms 110 or 111 = 64 ms
<26:24>	L0EL	R/W	0x3	0x3	L0s entrance latency. Values correspond to: 000 = 1 ms 001 = 2 ms 010 = 3 ms 011 = 4 ms 100 = 5 ms 101 = 6 ms 110 or 111 = 7 ms
<23:16>	N_FTS_CC	R/W	0x30	0x30	N_FTS when common clock is used. The number of fast training sequence (FTS) ordered sets to be transmitted when transitioning from L0s to L0. The maximum number of FTS ordered sets that a component can request is 0xFF (255). **NOTE:** The core does not support a value of 0x0; a value of 0x0 can cause the LTSSM to go into the recovery state when exiting from L0s.
<15:8>	N_FTS	R/W	0x30	0x30	N_FTS. The number of fast training sequence (FTS) ordered sets to be transmitted when transitioning from L0s to L0. The maximum number of FTS ordered sets that a component can request is 0xFF (255). **NOTE:** The core does not support a value of 0x0; a value of 0x0 can cause the LTSSM to go into the recovery state when exiting from L0s.
<7:0>	ACK_FREQ	R/W	0x0	0x0	Ack frequency. The number of pending Acks specified here (up to 0xFF) before sending an Ack.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Port Link Control Register
PCIERC(0..2)_CFG452

This register contains the four hundred fifty-third 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:22>	—	RAZ	—	—	Reserved
<21:16>	LME	R/W	0x7	0x7	Link mode enable set as follows: 000001 (0x1) = ×1 000011 (0x3) = ×2 000111 (0x7) = ×4 001111 (0xF) = ×8 (not supported) 011111 (0x1F) = ×16 (not supported) 111111 (0x3F) = ×32 (not supported) This field indicates the maximum number of lanes supported by the PCIe port. The value can be set less than 0xF to limit the number of lanes the PCIe will attempt to use. The programming of this field needs to be done by software before enabling the link. See also PCIERC(0..2)_CFG031[MLW]. **NOTE:** The value of this field does not indicate the number of lanes in use by the PCIe. This field sets the maximum number of lanes in the PCIe core that could be used. As per the PCIe specification, the PCIe core can negotiate a smaller link width, so all of ×4, ×2, and ×1 are supported when LME = 0xF, for example.
<15:12>	—	RAZ	—	—	Reserved
<11:8>	LINK_RATE	RO/H	0x1	0x1	Reserved
<7>	FLM	R/W/H	0	0	Fast link mode. Sets all internal timers to fast mode for simulation purposes.
<6>	—	RAZ	—	—	Reserved
<5>	DLLLE	R/W	1	1	DLL link enable. Enables link initialization. If DLL link enable = 0, the PCI Express bus does not transmit InitFC DLLPs and does not establish a Link.
<4>	—	RAZ	—	—	Reserved
<3>	RA	R/W	0	0	Reset assert. Triggers a recovery and forces the LTSSM to the hot reset state (downstream port only).
<2>	LE	R/W	0	0	Loopback enable. Initiate loopback mode as a master. On a 0→1 transition, the PCIe core sends TS ordered sets with the loopback bit set to cause the link partner to enter into loopback mode as a slave. Normal transmission is not possible when LE = 1. To exit loopback mode, take the link through a reset sequence.
<1>	SD	R/W	0	0	Scramble disable. Setting this bit turns off data scrambling.
<0>	OMR	R/W	0	0	Other message request. When software writes a 1 to this bit, the PCI Express bus transmits the Message contained in the Other Message register.

Cavium Confidential For Ruiwen Yang 08/23/24

Lane Skew Register
PCIERC(0..2)_CFG453

This register contains the four hundred fifty-fourth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	DLLD	R/W	0	0	Disable lane-to-lane deskew. Disables the internal lane-to-lane deskew logic.
<30:26>	—	RAZ	—	—	Reserved
<25>	ACK_NAK	R/W	0	0	Ack/Nak disable. Prevents the PCI Express bus from sending Ack and Nak DLLPs.
<24>	FCD	R/W	0	0	Flow control disable. Prevents the PCI Express bus from sending FC DLLPs.
<23:0>	ILST	R/W	0x0	0x0	Insert lane skew for transmit (not supported for ×16). Causes skew between lanes for test purposes. There are three bits per Lane. The value is in units of one symbol time. For example, the value 010b for a Lane forces a skew of two symbol times for that Lane. The maximum skew value for any Lane is 5 symbol times.

Symbol Number Register
PCIERC(0..2)_CFG454

This register contains the four hundred fifty-fifth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	—	RAZ	—	—	Reserved
<23:19>	TMANLT	R/W	0x0	0x0	Timer modifier for Ack/Nak latency timer. Increases the timer value for the Ack/Nak latency timer, in increments of 64 clock cycles.
<18:14>	TMRT	R/W	0x8	0x8	Timer modifier for replay timer. Increases the timer value for the replay timer, in increments of 64 clock cycles.
<13:8>	—	RAZ	—	—	Reserved
<7:0>	MFUNCN	R/W	0x0	0x0	Max number of functions supported.

Cavium Confidential For
Bowen Yang
ALL
02/27/2014

Symbol Timer/Filter Mask Register 1
PCIERC(0..2)_CFG455

This register contains the four hundred fifty-sixth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	M_CFG0_FILT	R/W	0	0	Mask filtering of received configuration requests (RC mode only)
<30>	M_IO_FILT	R/W	0	0	Mask filtering of received I/O Requests (RC mode only)
<29>	MSG_CTRL	R/W	0	0	Message control. The application must not change this field.
<28>	M_CPL_ECRC_FILT	R/W	0	0	Mask ECRC error filtering for completions
<27>	M_ECRC_FILT	R/W	0	0	Mask ECRC error filtering.
<26>	M_CPL_LEN_ERR	R/W	0	0	Mask length mismatch error for received completions
<25>	M_CPL_ATTR_ERR	R/W	0	0	Mask attributes mismatch error for received completions
<24>	M_CPL_TC_ERR	R/W	0	0	Mask traffic class mismatch error for received completions
<23>	M_CPL_FUN_ERR	R/W	0	0	Mask function mismatch error for received completions
<22>	M_CPL_RID_ERR	R/W	0	0	Mask Requester ID mismatch error for received completions
<21>	M_CPL_TAG_ERR	R/W	0	0	Mask tag error rules for received completions
<20>	M_LK_FILT	R/W	0	0	Mask locked request filtering
<19>	M_CFG1_FILT	R/W	0	0	Mask Type 1 Configuration request filtering
<18>	M_BAR_MATCH	R/W	0	0	Mask BAR match filtering
<17>	M_POIS_FILT	R/W	1	1	Mask poisoned TLP filtering
<16>	M_FUN	R/W	0	0	Mask function
<15>	DFCWT	R/W	0	0	Disable FC watchdog timer
<14:11>	—	RAZ	—	—	Reserved
<10:0>	SKPIV	R/W	0x280	0x280	SKP interval value

Filter Mask Register 2
PCIERC(0..2)_CFG456

This register contains the four hundred fifty-seventh 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:4>	—	RAZ	—	—	Reserved
<3>	M_HANDLE_FLUSH	R/W	0	0	Mask core filter to handle flush request
<2>	M_DABORT_4UCPL	R/W	0	0	Mask DLLP abort for unexpected CPL
<1>	M_VEND1_DRP	R/W	0	0	Mask vendor MSG Type 1 dropped silently
<0>	M_VEND0_DRP	R/W	0	0	Mask vendor MSG Type 0 dropped with UR error reporting

Debug Register 0
PCIERC(0..2)_CFG458

This register contains the four hundred fifty-ninth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DBG_INFO_L32	RO/H	0x0	0x0	The value on cxpl_debug_info[31:0].

Cavium Confidential Bowen Yang ALU 02/27/2014

Debug Register 1
PCIERC(0..2)_CFG459

This register contains the four hundred sixtieth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DBG_INFO_U32	RO/H	0x0	0x0	The value on cxpl_debug_info[63:32].

Transmit Posted FC Credit Status Register
PCIERC(0..2)_CFG460

This register contains the four hundred sixty-first 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	—	RAZ	—	—	Reserved
<19:12>	TPHFCC	RO/H	0x0	0x0	Transmit posted header FC Credits. The posted header credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.
<11:0>	TPDFCC	RO/H	0x0	0x0	Transmit posted data FC credits. The posted data credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.

Transmit Non-posted FC Credit Status Register
PCIERC(0..2)_CFG461

This register contains the four hundred sixty-second 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	—	RAZ	—	—	Reserved
<19:12>	TCHFCC	RO/H	0x0	0x0	Transmit non-posted header FC credits. The non-posted header credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.
<11:0>	TCDFCC	RO/H	0x0	0x0	Transmit non-posted data FC credits. The non-posted data credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.

Transmit Completion FC Credit Status Register
PCIERC(0..2)_CFG462

This register contains the four hundred sixty-third 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	—	RAZ	—	—	Reserved
<19:12>	TCHFCC	RO/H	0x0	0x0	Transmit completion header FC credits. The completion header credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.

Cavium Confidential For Bowen Tang 02/12/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<11:0>	TCDFCC	RO/H	0x0	0x0	Transmit completion data FC credits. The completion data credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.

Queue Status Register PCIERC(0..2)_CFG463

This register contains the four hundred sixty-fourth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:3>	—	RAZ	—	—	Reserved
<2>	RQNE	RO/H	0	0	Received queue not empty. Indicates there is data in one or more of the receive buffers.
<1>	TRBNE	RO/H	0	0	Transmit retry buffer not empty. Indicates that there is data in the transmit retry buffer.
<0>	RTLPFCCNR	RO/H	0	0	Received TLP FC credits not returned. Indicates that the PCI Express bus has sent a TLP but has not yet received an UpdateFC DLLP indicating that the credits for that TLP have been restored by the receiver at the other end of the Link.

VC Transmit Arbitration Register 1 PCIERC(0..2)_CFG464

This register contains the four hundred sixty-fifth 32-bits of type 0 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	WRR_VC3	RO	0x0	0x0	WRR Weight for VC3
<23:16>	WRR_VC2	RO	0x0	0x0	WRR Weight for VC2
<15:8>	WRR_VC1	RO	0x0	0x0	WRR Weight for VC1
<7:0>	WRR_VC0	RO	0xF	0xF	WRR Weight for VC0

VC Transmit Arbitration Register 2 PCIERC(0..2)_CFG465

This register contains the four hundred sixty-sixth 32-bits of type 0 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	WRR_VC7	RO	0x0	0x0	WRR Weight for VC7
<23:16>	WRR_VC6	RO	0x0	0x0	WRR Weight for VC6
<15:8>	WRR_VC5	RO	0x0	0x0	WRR Weight for VC5
<7:0>	WRR_VC4	RO	0xF	0xF	WRR Weight for VC4

Cavium Confidential For Bowen Yang ALU 02/27/2014

VC0 Posted Receive Queue Control Register PCIERC(0..2)_CFG466

This register contains the four hundred sixty-seventh 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	RX_QUEUE_ORDER	R/W	0	0	VC ordering for receive queues. Determines the VC ordering rule for the receive queues, used only in the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR: 0 = round-robin. 1 = strict ordering, higher numbered VCs have higher priority. The application must not change this field.
<30>	TYPE_ORDERING	R/W	1	1	TLP type ordering for VC0. Determines the TLP type ordering rule for VC0 receive queues, used only in the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR: 0 = strict ordering for received TLPs: Posted, then Completion, then Non-Posted. 1 = ordering of received TLPs follows the rules in PCI Express Base Specification. The application must not change this field.
<29:24>	—	RAZ	—	—	Reserved
<23:21>	QUEUE_MODE	R/W	0x2	0x2	VC0 posted TLP queue mode. The operating mode of the posted receive queue for VC0, used only in the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field. Only one bit can be set at a time: bit<23> = bypass bit<22> = cut-through bit<21> = store-and-forward
<20>	—	RAZ	—	—	Reserved
<19:12>	HEADER_CREDITS	R/W	0x20	0x20	VC0 posted header credits. The number of initial posted header credits for VC0, used for all receive queue buffer configurations. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<11:0>	DATA_CREDITS	R/W	0x40	0x40	VC0 posted data credits. The number of initial posted data credits for VC0, used for all receive queue buffer configurations. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Cavium Confidential For Bowen Yang 02/27/2014

VC0 Non-Posted Receive Queue Control Register PCIERC(0..2)_CFG467

This register contains the four hundred sixty-eighth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	—	RAZ	—	—	Reserved
<23:21>	QUEUE_MODE	R/W	0x2	0x2	VC0 non-posted TLP queue mode. The operating mode of the non-posted receive queue for VC0, used only in the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. Only one bit can be set at a time: bit<23> = bypass bit<22> = cut-through bit<21> = store-and-forward The application must not change this field.
<20>	—	RAZ	—	—	Reserved
<19:12>	HEADER_CREDITS	R/W	0x1F	0x1F	VC0 non-posted header credits. The number of initial non-posted header credits for VC0, used for all receive queue buffer configurations. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<11:0>	DATA_CREDITS	R/W	0xD	0xD	VC0 non-posted data credits. The number of initial non-posted data credits for VC0, used for all receive queue buffer configurations. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

VC0 Completion Receive Queue Control Register PCIERC(0..2)_CFG468

This register contains the four hundred sixty-ninth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	—	RAZ	—	—	Reserved
<23:21>	QUEUE_MODE	R/W	0x2	0x2	VC0 completion TLP queue mode. The operating mode of the completion receive queue for VC0, used only in the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. Only one bit can be set at a time: bit<23> = bypass bit<22> = cut-through bit<21> = store-and-forward The application must not change this field.
<20>	—	RAZ	—	—	Reserved
<19:12>	HEADER_CREDITS	R/W	0x60	0x60	VC0 completion header credits. The number of initial completion header credits for VC0, used for all receive queue buffer configurations. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Cavium Confidential For Bowen Yang ALU 2/17/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<11:0>	DATA_CREDITS	R/W	0x80	0x80	VC0 completion data credits. The number of initial completion data credits for VC0, used for all receive queue buffer configurations. This field is writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

VC0 Posted Buffer Depth Register PCIERC(0..2)_CFG490

This register contains the four hundred ninety-first 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:26>	—	RAZ	—	—	Reserved
<25:16>	HEADER_DEPTH	R/W	0x26	0x26	VC0 posted header queue depth. Sets the number of entries in the posted header queue for VC0 when using the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<15:14>	—	RAZ	—	—	Reserved
<13:0>	DATA_DEPTH	R/W	0xC8	0xC8	VC0 posted data queue depth. Sets the number of entries in the posted data queue for VC0 when using the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

VC0 Non-Posted Buffer Depth Register PCIERC(0..2)_CFG491

This register contains the four hundred ninety-second 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:26>	—	RAZ	—	—	Reserved
<25:16>	HEADER_DEPTH	R/W	0x25	0x25	VC0 non-posted header queue depth. Sets the number of entries in the non-posted header queue for VC0 when using the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<15:14>	—	RAZ	—	—	Reserved
<13:0>	DATA_DEPTH	R/W	0x61	0x61	VC0 non-posted data queue depth. Sets the number of entries in the non-posted data queue for VC0 when using the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

VC0 Completion Buffer Depth Register PCIERC(0..2)_CFG492

This register contains the four hundred ninety-third 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<31:26>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Kang ALU 02/26/2014

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<25:16>	HEADER_DEPTH	R/W	0x66	0x66	VC0 completion header queue depth. Sets the number of entries in the completion header queue for VC0 when using the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.
<15:14>	—	RAZ	—	—	Reserved
<13:0>	DATA_DEPTH	R/W	0x18E	0x18E	VC0 completion data queue depth. Sets the number of entries in the completion data queue for VC0 when using the segmented-buffer configuration, writable through PEM(0..2)_CFG_WR. However, the application must not change this field.

Port Logic Register (Gen2) PCIERC(0..2)_CFG515

This register contains the five hundred sixteenth 32-bits of type 1 PCIe configuration space. See Table 22–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:21>	—	RAZ	—	—	Reserved
<20>	S_D_E	R/W	0	0	SEL_DE_EMPHASIS. Used to set the de-emphasis level for upstream ports.
<19>	CTCRB	R/W	0	0	Config Tx compliance receive bit. When set to 1, signals LTSSM to transmit TS ordered sets with the compliance receive bit assert (equal to 1).
<18>	CPYTS	R/W	0	0	Config PHY Tx swing. Indicates the voltage level that the PHY should drive. When set to 1, indicates full swing. When set to 0, indicates low swing.
<17>	DSC	R/W/H	0	0	Directed speed change. A write of 1 initiates a speed change; always reads as 0.
<16:8>	LE	R/W	0x4	0x4	Lane enable. Indicates the number of lanes to check for exit from electrical idle in Polling.Active and Polling.Compliance. 0x1 = ×1, 0x2 = ×2, etc. Used to limit the maximum link width to ignore broken lanes that detect a receiver, but will not exit electrical idle and would otherwise prevent a valid link from being configured.
<7:0>	N_FTS	R/W	0x30	0x30	N_FTS. Sets the number of fast training sequences (N_FTS) that the core advertises as its N_FTS during GEN2 Link training. This value is used to inform the Link partner about the PHY's ability to recover synchronization after a low-power state. **NOTE:** Do not set N_FTS to zero; doing so can cause the LTSSM to go into the recovery state when exiting from L0s.

PHY Status Register PCIERC(0..2)_CFG516

This register contains the five hundred seventeenth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	PHY_STAT	RO/H	0x0	0x0	PHY status

Cavium Confidential For Power AE Lu Yang 2016/04/14

PHY Control Register
PCIERC(0..2)_CFG517

This register contains the five hundred eighteenth 32-bits of type 1 PCIe configuration space. See Table 22–4 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	PHY_CTRL	R/W	0x0	0x0	PHY control

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

22.8 PEM Registers

Table 22–5 lists the PEM registers.

Table 22–5 PEM Registers

	Address[1]	CSR Type[2]	Detailed Description
PEM(0..2)_CTL_STATUS	0x11800C0000000 + α*0x1000000	RSL	See page 997
PEM(0..2)_BIST_STATUS	0x11800C0000018 + α*0x1000000	RSL	See page 998
PEM(0..2)_DIAG_STATUS	0x11800C0000020 + α*0x1000000	RSL	See page 998
PEM(0..2)_CFG_WR	0x11800C0000028 + α*0x1000000	RSL	See page 998
PEM(0..2)_CFG_RD	0x11800C0000030 + α*0x1000000	RSL	See page 999
PEM(0..2)_TLP_CREDITS	0x11800C0000038 + α*0x1000000	RSL	See page 999
PEM(0..2)_P2N_BAR0_START	0x11800C0000080 + α*0x1000000	RSL	See page 1000
PEM(0..2)_P2N_BAR1_START	0x11800C0000088 + α*0x1000000	RSL	See page 1001
PEM(0..2)_P2N_BAR2_START	0x11800C0000090 + α*0x1000000	RSL	See page 1001
PEM(0..2)_CPL_LUT_VALID	0x11800C0000098 + α*0x1000000	RSL	See page 1001
PEM(0..2)_DBG_INFO_EN	0x11800C00000A0 + α*0x1000000	RSL	See page 1001
PEM(0..2)_BAR_CTL	0x11800C00000A8 + α*0x1000000	RSL	See page 1003
PEM(0..2)_BAR2_MASK	0x11800C00000B0 + α*0x1000000	RSL	See page 1003
PEM(0..2)_INB_READ_CREDITS	0x11800C00000B8 + α*0x1000000	RSL	See page 1003
PEM(0..2)_ECC_ENA	0x11800C00000C0 + α*0x1000000	RSL	See page 1004
PEM(0..2)_ECC_SYND_CTRL	0x11800C00000C8 + α*0x1000000	RSL	See page 1004
PEM(0..2)_DBG_INFO	0x11800C00000D0 + α*0x1000000	RSL	See page 1005
PEM(0..2)_BAR1_INDEX(0..15)	0x11800C0000100 + α*0x1000000 + β*0x8	RSL	See page 1006
PEM(0..2)_SPI_CTL	0x11800C0000180 + α*0x1000000	RSL	See page 1006
PEM(0..2)_SPI_DATA	0x11800C0000188 + α*0x1000000	RSL	See page 1007
PEM(0..2)_CLK_EN	0x11800C0000400 + α*0x1000000	RSL	See page 1007
PEM(0..2)_STRAP	0x11800C0000408 + α*0x1000000	RSL	See page 1007
PEM(0..2)_CFG	0x11800C0000410 + α*0x1000000	RSL	See page 1008
PEM(0..2)_ON	0x11800C0000420 + α*0x1000000	RSL	See page 1008
PEM(0..2)_INT_SUM	0x11800C0000428 + α*0x1000000	RSL	See page 1008
PEM(0..2)_INT_ENB	0x11800C0000430 + α*0x1000000	RSL	See page 1009
PEM(0..2)_INT_ENB_INT	0x11800C0000438 + α*0x1000000	RSL	See page 1010
PEM(0..2)_BIST_STATUS2	0x11800C0000440 + α*0x1000000	RSL	See page 1010

1. For registers with multiple versions, α represents the values within the specified range and β the second range.
2. RSL-type registers are accessed indirectly across the I/O Interconnect.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PEM Control Status Register
PEM(0..2)_CTL_STATUS

This is a general control and status register of the PEM. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:51>	—	RAZ	—	—	Reserved
<50>	INV_DPAR	R/W	0	0	Invert the generated parity to be written into the most significant data-queue buffer RAM block to force a parity error when it is later read.
<49>	INV_HPAR	R/W	0	0	Invert the generated parity to be written into the most significant header-queue buffer RAM block to force a parity error when it is later read.
<48>	INV_RPAR	R/W	0	0	Invert the generated parity to be written into the most significant retry buffer RAM block to force a parity error when it is later read.
<47>	AUTO_SD	RO	—	—	Link hardware autonomous speed disable
<46:42>	DNUM	RO	—	—	Primary bus device number
<41:34>	PBUS	RO	—	—	Primary bus number
<33:32>	—	RAZ	—	—	Reserved
<31:16>	CFG_RTRY	R/W	0x0	0x20	The time × 0x10000 in coprocessor-clock cycles to wait for a CPL to a configuration read that does not carry a retry status. Until such time that the timeout occurs and retry status is received for a configuration read, the read will be resent. A value of 0 disables retries and treats a CPL Retry as a CPL UR. When enabled, only one CFG RD may be issued until either successful completion or CPL UR.
<15:12>	—	RAZ	—	—	Reserved
<11>	PM_XTOFF	R/W/H	0	0	When set, a single cycle pulse is sent to the PCIe core pm_xmt_turnoff port. RC mode.
<10>	PM_XPME	R/W/H	0	0	When set, a single cycle pulse is sent to the PCIe core pm_xmt_pme port. EP mode.
<9>	OB_P_CMD	R/W/H	0	0	When set, a single cycle pulse is sent to the PCIe core outband_pwrup_cmd port. EP mode.
<8:6>	—	R/W	—	—	Reserved
<5>	DLY_ONE	R/W	0	0	When set the output client state machines will wait one cycle before starting a new TLP out.
<4>	LNK_ENB	R/W	0	0	When set, the link is enabled; when clear (0) the link is disabled. This bit only is active when in RC mode.
<3>	RO_CTLP	R/W	0	0	When set, C-TLPs that have the RO bit set will not wait for P-TLPs that are normally sent first.
<2>	FAST_LM	R/W	0	0	When set, forces fast link mode.
<1>	INV_ECRC	R/W	0	0	When set, causes the LSB of the ECRC to be inverted.
<0>	INV_LCRC	R/W	0	0	When set, causes the LSB of the LCRC to be inverted.

Cavium Confidential For Power ALU 04/27/2014

PEM BIST Status Register
PEM(0..2)_BIST_STATUS

This register contains the different interrupt summary bits of the PEM. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved.
<5>	RETRY	RO	0	0	Retry buffer.
<4>	SOT	RO	0	0	Start-of-transfer memory.
<3>	RQHDR0	RO	0	0	RX queue header 0.
<2>	RQHDR1	RO	0	0	RX queue header 1.
<1>	RQDATA0	RO	0	0	RX queue data memory 0.
<0>	RQDATA1	RO	0	0	RX queue data memory 1.

PEM Diagnostic Status Register
PEM(0..2)_DIAG_STATUS

This register contains selection control for the core diagnostic bus. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RO	—	—	Reserved.
<5:3>	PM_DST	RO	0x0	0x0	Current power management DSTATE.
<2>	PM_STAT	RO	0	0	Power management status.
<1>	PM_EN	RO	0	0	Power management event enable.
<0>	AUX_EN	RO	0	0	Auxiliary power enable.

PEM Configuration Write Register
PEM(0..2)_CFG_WR

This register allows write access to the configuration in the PCIe core. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	DATA	R/W	0x0	—	Data to write. Writing to this register starts a write operation.
<31:0>	ADDR	R/W	0x0	—	Address to write. Writing to this register starts a write operation. There are the subfields of ADDR: <31> When set, asserts dbi_cs2 at PCIe core. <30:12> Not used (when asserted, result is address out of bounds; write operation completes immediately). <11:0> The offset of the PCIe core configuration register being accessed.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PEM Configuration Read Register
PEM(0..2)_CFG_RD

This register allows read access to the configuration in the PCIe core. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	DATA	R/W/H	0x0	—	Data.
<31:0>	ADDR	R/W	0x0	—	Address to read. Writing to this register starts a read operation. There are the subfields of ADDR: <31:12> Not used (when asserted, result is address out of bounds; read operation returns all 1s). <11:0> The offset of the PCIe core configuration register being accessed.

PEM TLP Credits Register
PEM(0..2)_TLP_CREDITS

This register specifies the number of credits for use in moving TLPs. When this register is written, the credit values are reset to the register value. A write to this register should take place before the start of traffic flow. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:56>	—	RAZ	—	—	Reserved
<55:48>	PEAI_PPF	R/W/H	0x80	0x80	TLP credits for TLPs inbound for the PEM. The value in this register should not be changed. Values other than 0x80 can lead to unpredictable behavior.
<47:24>	—	RAZ	—	—	Reserved
<23:16>	SLI_CPL	R/W/H	0x80	0x80	TLP credits for completion TLPs in the SLI. Legal values are 0x24 to 0x80. When multiple PEMs are wire-ORed together, the sum of all PEMs' SLI_CPL fields must not exceed 0x80. The reset value for this register is based on the PCIE0_MODE<1:0> straps; 00 = 2 lanes = 64 (0x40) credits 01 = 1 lane = 42 (0x2A) credits 10, 11 = 4 lanes = 128 (0x80) credits. Software must reprogram this register for RC mode PEMs, since they always default to four lanes. Software might also need to reprogram this register for performance reasons (configured as the only PEM, but as two-lane: in this case, this PEM can take all of the credits).

Cavium Confidential For Bowen Yang 9/12/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<15:8>	SLI_NP	R/W	0x10	0x10	TLP credits for nonposted TLPs in the SLI. Legal values are 0x4 to 0x10. When multiple PEMs are wire-ORed together, the sum of all PEMs' SLI_CPL fields must not exceed 0x10. The reset value for this register is based on the PCIE0_MODE<1:0> straps; 00 = 2 lanes = 8 (0x8) credits 01 = 1 lane = 5 (0x5) credits 10, 11 = 4 lanes = 16 (0x10) credits. Software must reprogram this register for RC mode PEMs, since they always default to four lanes. Software might also need to reprogram this register for performance reasons (configured as the only PEM, but as two-lane: in this case, this PEM can take all of the credits).
<7:0>	SLI_P	R/W	0x80	0x80	TLP credits for posted TLPs in the SLI. Legal values are 0x24 to 0x80. When multiple PEMs are wire-ORed together, the sum of all PEMs' SLI_CPL fields must not exceed 0x80. The reset value for this register is based on the pi_select_mode[1:0] straps; 00 = 2 lanes = 64 (0x40) credits 01 = 1 lane = 42 (0x2A) credits 10, 11 = 4 lanes = 128 (0x80) credits. Software must reprogram this register for RC mode PEMs, since they always default to four lanes. Software might also need to reprogram this register for performance reasons (configured as the only PEM, but as two-lane: in this case, this PEM can take all of the credits). For PEM0 in EP mode, updating this field from the PCIe Port via a windowed write requires 3 SLI_P credits which are returned to the pool after the update. In addition a subsequent windowed write can cause an additional 3 SLI_P credits to be returned to the pool resulting in a total of 6 additional credits more than what this field is configured to be. Therefore any update to this register from the PCIe Port when PEM0 is in EP mode, software must account for potential 6 when updating this field. For example, for 4-lane mode where PEM0 gets all 128 credits, this field must be written with a value of 122 credits. This ensures that the resulting value can not end up more than the 128 credits maximum.

PEM PCIe to SLI BAR0 Start Register PEM(0..2)_P2N_BAR0_START

This register specifies the starting address for memory requests that are to be forwarded to the SLI in RC mode. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:14>	ADDR	R/W	0x0	0x0	The starting address of the 16KB BAR0 address space.
<13:0>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang ALD 10/27/2014

PEM PCIe to SLI BAR1 Start Register
PEM(0..2)_P2N_BAR1_START

This register specifies the starting address for memory requests that are to be forwarded to the SLI in RC mode. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:26>	ADDR	R/W	0x0	0x0	The starting address of the 64MB BAR1 address space.
<25:0>	—	RAZ	—	—	Reserved

PEM PCIe to SLI BAR2 Start Register
PEM(0..2)_P2N_BAR2_START

This register specifies the starting address for memory requests that are to be forwarded to the SLI in RC mode. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:41>	ADDR	R/W	0x0	0x0	The starting address of the 2^{41} BAR2 address space.
<40:0>	—	RAZ	—	—	Reserved

PEM Completion Lookup Table Valid Register
PEM(0..2)_CPL_LUT_VALID

This register specifies the bit set for an outstanding tag read. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	TAG	RO/H	0x0	0x0	Bit vector set corresponds to an outstanding tag.

PEM Debug Information Enable Register
PEM(0..2)_DBG_INFO_EN

This register allows PEM(0..2)_DBG_INFO to generate interrupts when the corresponding enable bit is set. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:46>	—	RAZ	—	—	Reserved
<45>	TPCDBE1	R/W	0	0	Allows PEM(0..2)_DBG_INFO[C_C_DBE] to generate an interrupt.
<44>	TPCSBE1	R/W	0	0	Allows PEM(0..2)_DBG_INFO[C_C_SBE] to generate an interrupt.
<43>	TPCSDE0	R/W	0	0	Allows PEM(0..2)_DBG_INFO[C_D_DBE] to generate an interrupt.
<42>	TPCSBE0	R/W	0	0	Allows PEM(0..2)_DBG_INFO[C_D_SBE] to generate an interrupt.
<41>	TNFDBE1	R/W	0	0	Allows PEM(0..2)_DBG_INFO[N_C_DBE] to generate an interrupt.
<40>	TNFSBE1	R/W	0	0	Allows PEM(0..2)_DBG_INFO[N_C_SBE] to generate an interrupt.
<39>	TNFDBE0	R/W	0	0	Allows PEM(0..2)_DBG_INFO[N_D_DBE] to generate an interrupt.
<38>	TNFSBE0	R/W	0	0	Allows PEM(0..2)_DBG_INFO[N_D_SBE] to generate an interrupt.
<37>	TPFDBE1	R/W	0	0	Allows PEM(0..2)_DBG_INFO[P_C_DBE] to generate an interrupt.
<36>	TPFSBE1	R/W	0	0	Allows PEM(0..2)_DBG_INFO[P_C_SBE] to generate an interrupt.
<35>	TPFDBE0	R/W	0	0	Allows PEM(0..2)_DBG_INFO[P_D_DBE] to generate an interrupt.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<34>	TPFSBE0	R/W	0	0	Allows PEM(0..2)_DBG_INFO[P_D_SBE] to generate an interrupt.
<33>	DATQ_PE	R/W	0	0	Allows PEM(0..2)_DBG_INFO[DATQ_PE] to generate an interrupt.
<32>	HDRQ_PE	R/W	0	0	Allows PEM(0..2)_DBG_INFO[HDRQ_PE] to generate an interrupt.
<31>	RTRY_PE	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RTRY_PE] to generate an interrupt.
<30>	ECRC_E	R/W	0	0	Allows PEM(0..2)_DBG_INFO[ECRC_E] to generate an interrupt.
<29>	RAWWPP	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RAWWPP] to generate an interrupt.
<28>	RACPP	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RACPP] to generate an interrupt.
<27>	RAMTLP	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RAMTLP] to generate an interrupt.
<26>	RARWDNS	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RARWDNS] to generate an interrupt.
<25>	CAAR	R/W	0	0	Allows PEM(0..2)_DBG_INFO[CAAR] to generate an interrupt.
<24>	RACCA	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RACCA] to generate an interrupt.
<23>	RACUR	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RACUR] to generate an interrupt.
<22>	RAUC	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RAUC] to generate an interrupt.
<21>	RQO	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RQO] to generate an interrupt.
<20>	FCUV	R/W	0	0	Allows PEM(0..2)_DBG_INFO[FCUV] to generate an interrupt.
<19>	RPE	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RPE] to generate an interrupt.
<18>	FCPVWT	R/W	0	0	Allows PEM(0..2)_DBG_INFO[FCPVWT] to generate an interrupt.
<17>	DPEOOSD	R/W	0	0	Allows PEM(0..2)_DBG_INFO[DPEOOSD] to generate an interrupt.
<16>	RTWDLE	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RTWDLE] to generate an interrupt.
<15>	RDWDLE	R/W	0	0	Allows PEM(0..2)_DBG_INFO[IRDWDLE] to generate an interrupt.
<14>	MRE	R/W	0	0	Allows PEM(0..2)_DBG_INFO[MRE] to generate an interrupt.
<13>	RTE	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RTE] to generate an interrupt.
<12>	ACTO	R/W	0	0	Allows PEM(0..2)_DBG_INFO[ACTO] to generate an interrupt.
<11>	RVDM	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RVDM] to generate an interrupt.
<10>	RUMEP	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RUMEP] to generate an interrupt.
<9>	RPTAMRC	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RPTAMRC] to generate an interrupt.
<8>	RPMERC	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RPMERC] to generate an interrupt.
<7>	RFEMRC	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RFEMRC] to generate an interrupt.
<6>	RNFEMRC	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RNFEMRC] to generate an interrupt.
<5>	RCEMRC	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RCEMRC] to generate an interrupt.
<4>	RPOISON	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RPOISON] to generate an interrupt.
<3>	RECRCE	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RECRCE] to generate an interrupt.
<2>	RTLPLLE	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RTLPLLE] to generate an interrupt.
<1>	RTLPMAL	R/W	0	0	Allows PEM(0..2)_DBG_INFO[RTLPMAL] to generate an interrupt.
<0>	SPOISON	R/W	0	0	Allows PEM(0..2)_DBG_INFO[SPOISON] to generate an interrupt.

Cavium Confidential For Board Use Only

PEM BAR Control Register
PEM(0..2)_BAR_CTL

This register contains control for BAR accesses. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved
<6:4>	BAR1_SIZ	R/W	0x1	0x1	PCIe Port 0 Bar1 Size. 0x0 = reserved 0x4 = 512MB 0x1 = 64MB 0x5 = 1024MB 0x2 = 128MB 0x6 = 2048MB 0x3 = 256MB 0x7 = reserved
<3>	BAR2_ENB	R/W	0	1	When set to 1, BAR2 is enabled and will respond; when clear, BAR2 access will cause UR responses.
<2:1>	BAR2_ESX	R/W	0x0	—	Value is XORed with PCIe address [45:44] to determine the endian swap mode.
<0>	BAR2_CAX	R/W	0	0	Value is XORed with PCIe address [46] to determine the L2 cache attribute. Not cached in L2 if XOR result is 1.

PEM BAR2 Mask Register
PEM(0..2)_BAR2_MASK

This register contains the mask pattern that is ANDed with the address from the PCIe core for BAR2 hits. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:38>	—	RAZ	—	—	Reserved
<37:3>	MASK	R/W	0x7FFFFFFFF	0x7FFFFFFFF	The value to be ANDed with the address sent to the CN71XX memory.
<2:0>	—	RAZ	—	—	Reserved

PEM In-flight Read Credits Register
PEM(0..2)_INB_READ_CREDITS

This register contains the number of in-flight read operations from PCIe core to SLI. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	NUM	R/W	0x20	0x20	The number of reads that may be in flight from the PCIe core to the SLI. Minimum number is 0x2; maximum number is 0x20.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PEM ECC Enable Register
PEM(0..2)_ECC_ENA

This register contains enable signals for ECC RAMs. See See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved.
<5>	TLP_NC_ENA	R/W	1	1	ECC enable signal for TLP non-posted control FIFO.
<4>	TLP_ND_ENA	R/W	1	1	ECC enable signal for TLP non-posted data FIFO.
<3>	TLP_PC_ENA	R/W	1	1	ECC enable signal for TLP posted control FIFO.
<2>	TLP_PD_ENA	R/W	1	1	ECC enable signal for TLP posted data FIFO.
<1>	TLP_CC_ENA	R/W	1	1	ECC enable signal for TLP CPL control FIFO.
<0>	TLP_CD_ENA	R/W	1	1	ECC enable signal for TLP CPL data FIFO.

PEM Syndrome Control Register
PEM(0..2)_ECC_SYND_CTRL

This register contains syndrome control signals for ECC RAMs. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved.
<11:10>	TLP_NC_SYN	R/W	0x0	0x0	Syndrome flip bits for TLP non-posted control FIFO.
<9:8>	TLP_ND_SYN	R/W	0x0	0x0	Syndrome flip bits for TLP non-posted data FIFO
<7:6>	TLP_PC_SYN	R/W	0x0	0x0	Syndrome flip bits for TLP posted control FIFO.
<5:4>	TLP_PD_SYN	R/W	0x0	0x0	Syndrome flip bits for TLP posted data FIFO.
<3:2>	TLP_CC_SYN	R/W	0x0	0x0	Syndrome flip bits for TLP CPL control FIFO.
<1:0>	TLP_CD_SYN	R/W	0x0	0x0	Syndrome flip bits for TLP CPL data FIFO.

Cavium Confidential For Bowen Yang ALL 02/27/2014

PEM Debug Information Register
PEM(0..2)_DBG_INFO

This is a debug information register of the PEM. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:46>	—	RAZ	—	—	Reserved
<45>	C_C_DBE	R/W1C/H	0	0	Detected a TLP CPL FIFO control double-bit error.
<44>	C_C_SBE	R/W1C/H	0	0	Detected a TLP CPL FIFO control single-bit error.
<43>	C_D_DBE	R/W1C/H	0	0	Detected a TLP CPL FIFO data double-bit error.
<42>	C_D_SBE	R/W1C/H	0	0	Detected a TLP CPL FIFO data single-bit error.
<41>	N_C_DBE	R/W1C/H	0	0	Detected a TLP non-posted FIFO control double-bit error.
<40>	N_C_SBE	R/W1C/H	0	0	Detected a TLP non-posted FIFO control single-bit error.
<39>	N_D_DBE	R/W1C/H	0	0	Detected a TLP non-posted FIFO data double-bit error.
<38>	N_D_SBE	R/W1C/H	0	0	Detected a TLP non-posted FIFO data single-bit error.
<37>	P_C_DBE	R/W1C/H	0	0	Detected a TLP posted FIFO control double-bit error.
<36>	P_C_SBE	R/W1C/H	0	0	Detected a TLP posted FIFO control single-bit error.
<35>	P_D_DBE	R/W1C/H	0	0	Detected a TLP posted FIFO data double-bit error.
<34>	P_D_SBE	R/W1C/H	0	0	Detected a TLP posted FIFO data single-bit error.
<33>	DATQ_PE	R/W1C/H	0	0	Detected a data-queue-RAM parity error.
<32>	HDRQ_PE	R/W1C/H	0	0	Detected a header-queue-RAM parity error.
<31>	RTRY_PE	R/W1C/H	0	0	Detected a retry-RAM parity error.
<30>	ECRC_E	R/W1C/H	0	0	Received an ECRC error.
<29>	RAWWPP	R/W1C/H	0	0	Received a write with poisoned payload.
<28>	RACPP	R/W1C/H	0	0	Received a completion with poisoned payload.
<27>	RAMTLP	R/W1C/H	0	0	Received a malformed TLP.
<26>	RARWDNS	R/W1C/H	0	0	Received a request that device does not support.
<25>	CAAR	R/W1C/H	0	0	Completer aborted a request. This bit is never set because PEM does not generate completer aborts.
<24>	RACCA	R/W1C/H	0	0	Received a completion with CA status.
<23>	RACUR	R/W1C/H	0	0	Received a completion with UR status.
<22>	RAUC	R/W1C/H	0	0	Received an unexpected completion.
<21>	RQO	R/W1C/H	0	0	Receive queue overflow. Normally happens only when flow control advertisements are ignored.
<20>	FCUV	R/W1C/H	0	0	Flow control update violation.
<19>	RPE	R/W1C/H	0	0	PHY reported an 8B/10B decode error (RxStatus = 0x4) or disparity error (RxStatus = 0x7).
<18>	FCPVWT	R/W1C/H	0	0	Flow control protocol violation (watchdog timer).
<17>	DPEOOSD	R/W1C/H	0	0	DLLP protocol error (out of sequence DLLP).
<16>	RTWDLE	R/W1C/H	0	0	Received TLP with datalink layer error.
<15>	RDWDLE	R/W1C/H	0	0	Received DLLP with datalink layer error.
<14>	MRE	R/W1C/H	0	0	Maximum number of retries exceeded.
<13>	RTE	R/W1C/H	0	0	Replay timer expired. This bit is set when the replay timer expires in the PCIe core. The probability of this bit being set increases with the traffic load.
<12>	ACTO	R/W1C/H	0	0	A completion timeout occurred.
<11>	RVDM	R/W1C/H	0	0	Received vendor-defined message.

Cavium Confidential For Bowen Yang @ 2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<10>	RUMEP	R/W1C/H	0	0	Received unlock message (EP mode only).
<9>	RPTAMRC	R/W1C/H	0	0	Received PME turnoff acknowledge message (RC mode only).
<8>	RPMERC	R/W1C/H	0	0	Received PME message (RC mode only).
<7>	RFEMRC	R/W1C/H	0	0	Received fatal-error message (RC mode only). This bit is set when a message with ERR_FATAL is set.
<6>	RNFEMRC	R/W1C/H	0	0	Received non-fatal error message (RC mode only).
<5>	RCEMRC	R/W1C/H	0	0	Received correctable error message (RC mode only).
<4>	RPOISON	R/W1C/H	0	0	Received poisoned TLP.
<3>	RECRCE	R/W1C/H	0	0	Received ECRC error.
<2>	RTLPLLE	R/W1C/H	0	0	Received TLP has link layer error.
<1>	RTLPMAL	R/W1C/H	0	0	Received TLP is malformed or a vendor message is received. If the core receives a MSG (or vendor message), this bit is set.
<0>	SPOISON	R/W1C/H	0	0	Poisoned TLP sent.

PEM BAR1 Index 0–15 Register
PEM(0..2)_BAR1_INDEX(0..15)

This register contains the address index and control bits for access to memory ranges of BAR1. The index is built from supplied address [35:22]. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:24>	—	RAZ	—	—	Reserved
<23:4>	ADDR_IDX	R/W	0x0	—	Address index. Address bits [41:22] sent to L2C.
<3>	CA	R/W	0	0	Cached. Set to 1 when access is not to be cached in L2.
<2:1>	END_SWP	R/W	0x0	—	Endian-swap mode. 0x0 = pass-through mode (no swap) 0x1 = 64-bit byte-swap mode [ABCD_EFGH] → [HGFE_DCBA] 0x2 = 32-bit byte-swap mode [ABCD_EFGH] → [DCBA_HGFE] 0x3 = 32-bit exchange mode [ABCD_EFGH] → [EFGH_ABCD]
<0>	ADDR_V	R/W	0	—	Address valid. Set to 1 when the selected address range is valid.

PEM SPI Control Register
PEM(0..2)_SPI_CTL

See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved.
<13>	START_BUSY	R/W	0	0	Start/Busy status. When written, starts SPI xctn. When read: 1 = EEPROM busy, 0 = complete.
<12>	TVALID	R/W	0	0	Table valid. Reads 1 if at least one valid entry was read from EEPROM and written to a register. Write to clear status.

Cavium Confidential For Bowen Yang ALI 02/27/2015

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<11:9>	CMD	R/W	0x0	0x0	SPI commands: 0x0 = reserved 0x1 = WRSR 0x2 = WRITE 0x3 = READ 0x4 = WRDI 0x5 = RDSR 0x6 = WREN 0x7 = reserved Refer to Section 22.4.1.1.
<8:0>	ADR	R/W	0x0	0x0	EEPROM read/write address.

PEM SPI Data Register
PEM(0..2)_SPI_DATA

This register contains most recently read or written data. It is unpredictable at power-up. See See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	PREAMBLE	R/W/H	—	—	PREAMBLE. Values can be: 0x9DA1 = valid entry 0x6A5D = end-of-table 0xFFFF = null
<47:45>	—	RAZ	—	—	Reserved.
<44>	CS2	R/W/H	—	—	CS2 bit. Allows access to write protected configuration registers.
<43:32>	ADR	R/W/H	—	—	EEPROM ADR offset.
<31:0>	DATA	R/W/H	—	—	Read or write data.

PEM Clock Enable Register
PEM(0..2)_CLK_EN

This register provides clock-enable signals for the coprocessor and PCIe clocks. See See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved.
<1>	PCECLK_GATE	R/W/H	0x0	0x0	When set to 1, the PCIe clock is gated off. When clear, the PCIe clock is enabled.
<0>	CSCLK_GATE	R/W/H	0x0	0x0	When set to 1, the coprocessor clock is gated off. When clear, the coprocessor clock is enabled.

PEM Pin Strapping Register
PEM(0..2)_STRAP

This register shows the value of the input strapping pins. See See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved.
<3>	PILANESWAP	RO/H	—	—	Shows the value of the pi_select_laneswap pin.
<2:0>	PIMODE	RO/H	—	—	Shows the value of the pi_select_mode[2:0] pins.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PEM Application Configuration Register
PEM(0..2)_CFG

This register provides control signals for the configuration of the PCIe application. See See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved.
<4>	LANESWAP	R/W/H	—	—	Lane swap. This field overwrites the signal setting for lane swapping. 1 = lane swapping is performed to/from the SerDes. 0 = no lane swapping is performed.
<3>	HOSTMD	R/W/H	—	—	Host mode. This field overwrites the signal settings for host mode. 1 = the PEM is configured to be a root-complex device. 0 = the PEM is configured to be an end-point device
<2:0>	MD	R/W/H	—	—	Mode. This field overwrites the PCIE0_MODE<2:0> signal settings. This value is used to set the maximum link width field in the core's Link Capabilities Register (CFG031) to indicate the maximum number of lanes supported. Note that less lanes than the specified maximum can be configured for use via the core's Link Control Register (CFG032) negotiated link width field. 0x0 = Gen2, 2-lanes 0x4 = Gen1, 2-lanes 0x1 = Gen2, 1-lane 0x5 = Gen1, 1-lane 0x2 = Gen2, 4-lanes 0x6 = Gen1, 4-lanes 0x3 = reserved 0x7 = reserved **NOTE:** The lower two bits of the MD field (i.e. [1:0]) must be the same across all configured PEMs.

PEM ON Status Register
PEM(0..2)_ON

This register indicates whether the PEM is configured and ready. See See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved.
<1>	PEMOOR	RO/H	—	—	Out-of-reset. Indication to software that the PEM has been taken out of reset (i.e. BIST is done) and it is safe to configure the core CSRs.
<0>	PEMON	R/W/H	—	—	On. Indication to the DLM that the PEM is out of reset, configured, and ready to send/receive traffic. Setting this bit takes the configured PIPE out of reset.

PEM Interrupt Summary Register
PEM(0..2)_INT_SUM

This register contains the different interrupt summary bits of the PEM. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved.
<13>	CRS_DR	R/W1C/H	0	0	Had a CRS timeout when retries were disabled.
<12>	CRS_ER	R/W1C/H	0	0	Had a CRS timeout when retries were enabled.
<11>	RDLK	R/W1C/H	0	0	Received read lock TLP.

Cavium Confidential For Bowen Yang ALU 04-2-2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<10>	EXC	RO/H	0	0	Set when the PEM_DBG_INFO register has a bit set and its corresponding PEM_DBG_INFO_EN bit is set.
<9>	UN_BX	R/W1C/H	0	0	Received N-TLP for unknown BAR.
<8>	UN_B2	R/W1C/H	0	0	Received N-TLP for BAR2 when BAR2 is disabled.
<7>	UN_B1	R/W1C/H	0	0	Received N-TLP for BAR1 when BAR1 index valid is not set.
<6>	UP_BX	R/W1C/H	0	0	Received P-TLP for an unknown BAR.
<5>	UP_B2	R/W1C/H	0	0	Received P-TLP for BAR2 when BAR2 is disabled.
<4>	UP_B1	R/W1C/H	0	0	Received P-TLP for BAR1 when BAR1 index valid is not set.
<3>	PMEM	RO/H	0	0	Received PME MSG.
<2>	PMEI	RO/H	0	0	PME interrupt.
<1>	SE	R/W1C/H	0	0	System error, RC mode only.
<0>	AERI	RO/H	0	0	Advanced error reporting interrupt, RC mode only.

PEM Interrupt Enable Register
PEM(0..2)_INT_ENB

This register enables interrupt conditions for the PEM to generate an RSL interrupt. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved
<13>	CRS_DR	R/W	0	1	Enables PEM(0..2)_INT_SUM[CRS_DR] to generate an interrupt to the MIO.
<12>	CRS_ER	R/W	0	1	Enables PEM(0..2)_INT_SUM[CRS_ER] to generate an interrupt to the MIO.
<11>	RDLK	R/W	0	1	Enables PEM(0..2)_INT_SUM[RDLK] to generate an interrupt to the MIO.
<10>	EXC	R/W	0	1	Enables PEM(0..2)_INT_SUM[EXC] to generate an interrupt to the MIO.
<9>	UN_BX	R/W	0	1	Enables PEM(0..2)_INT_SUM[UN_BX] to generate an interrupt to the MIO.
<8>	UN_B2	R/W	0	1	Enables PEM(0..2)_INT_SUM[UN_B2] to generate an interrupt to the MIO.
<7>	UN_B1	R/W	0	1	Enables PEM(0..2)_INT_SUM[UN_B1] to generate an interrupt to the MIO.
<6>	UP_BX	R/W	0	1	Enables PEM(0..2)_INT_SUM[UP_BX] to generate an interrupt to the MIO.
<5>	UP_B2	R/W	0	1	Enables PEM(0..2)_INT_SUM[UP_B2] to generate an interrupt to the MIO.
<4>	UP_B1	R/W	0	1	Enables PEM(0..2)_INT_SUM[UP_B1] to generate an interrupt to the MIO.
<3>	PMEM	R/W	0	1	Enables PEM(0..2)_INT_SUM[PMEM] to generate an interrupt to the MIO.
<2>	PMEI	R/W	0	1	Enables PEM(0..2)_INT_SUM[PMEI] to generate an interrupt to the MIO.
<1>	SE	R/W	0	1	Enables PEM(0..2)_INT_SUM[SE] to generate an interrupt to the MIO.
<0>	AERI	R/W	0	1	Enables PEM(0..2)_INT_SUM[AERI] to generate an interrupt to the MIO

Cavium Confidential For Boyuan Yang 2020-04

PEM Interrupt Enable Register
PEM(0..2)_INT_ENB_INT

This register enables interrupt conditions for the PEM to generate an RSL interrupt. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved
<13>	CRS_DR	R/W	0	1	Enables PEM(0..2)_INT_SUM[CRS_DR] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].
<12>	CRS_ER	R/W	0	1	Enables PEM(0..2)_INT_SUM[CRS_ER] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].
<11>	RDLK	R/W	0	1	Enables PEM(0..2)_INT_SUM[RDLK] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].
<10>	EXC	R/W	0	1	Enables PEM(0..2)_INT_SUM[EXC] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].
<9>	UN_BX	R/W	0	1	Enables PEM(0..2)_INT_SUM[UN_BX] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].
<8>	UN_B2	R/W	0	1	Enables PEM(0..2)_INT_SUM[UN_B2] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].
<7>	UN_B1	R/W	0	1	Enables PEM(0..2)_INT_SUM[UN_B1] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].
<6>	UP_BX	R/W	0	1	Enables PEM(0..2)_INT_SUM[UP_BX] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].
<5>	UP_B2	R/W	0	1	Enables PEM(0..2)_INT_SUM[UP_B2] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].
<4>	UP_B1	R/W	0	1	Enables PEM(0..2)_INT_SUM[UP_B1] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].
<3>	PMEM	R/W	0	1	Enables PEM(0..2)_INT_SUM[PMEM] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].
<2>	PMEI	R/W	0	1	Enables PEM(0..2)_INT_SUM[PMEI] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].
<1>	SE	R/W	0	1	Enables PEM(0..2)_INT_SUM[SE] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].
<0>	AERI	R/W	0	1	Enables PEM(0..2)_INT_SUM[AERI] to generate an interrupt to the SLI as SLI_INT_SUM[MAC1_INT].

PEM BIST Status2 Register
PEM(0..2)_BIST_STATUS2

This register contains results from BIST runs of PEM's memories. See Table 22–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved
<13>	PEAI_P2E	RO	0	0	BIST status for the peai_pesc_fifo.
<12>	TLPN_D	RO	0	0	BIST status for the tlp_n_fifo_data.
<11>	TLPN_CTL	RO	0	0	BIST status for the tlp_n_fifo_ctl.
<10>	TLPP_D	RO	0	0	BIST status for the tlp_p_fifo_data.
<9>	TLPP_CTL	RO	0	0	BIST status for the tlp_p_fifo_ctl.

Cavium Confidential For Beta Use 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<8>	TLPC_D	RO	0	0	BIST status for the tlp_c_fifo_data.
<7>	TLPC_CTL	RO	0	0	BIST status for the tlp_c_fifo_ctl.
<6>	TLPAN_D	RO	0	0	BIST status for the tlp_n_afifo_data.
<5>	TLPAN_CTL	RO	0	0	BIST status for the tlp_n_afifo_ctl.
<4>	TLPAP_D	RO	0	0	BIST status for the tlp_p_afifo_data.
<3>	TLPAP_CTL	RO	0	0	BIST status for the tlp_p_afifo_ctl.
<2>	TLPAC_D	RO	0	0	BIST status for the tlp_c_afifo_data.
<1>	TLPAC_CTL	RO	0	0	BIST status for the .tlp_c_afifo_ctl
<0>	M2S	RO	0	0	BIST status for the m2s_fifo.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 23

SGMII / 1000BASE-X / QSGMII Interfaces

This chapter contains the following subjects:

- Introduction
- GMX Packet-Interface Buffering
- PCS Autonegotiation, Mode, and GMX Configuration
- SGMII/1000BASE-X/QSGMII Features
- Wake-on-LAN (WOL)
- Errors/Exceptions
- Reset, Autonegotiation, Initialization, and Link Status Changes
- Statistics
- Loopback
- Enumerations
- GMX Registers
- PCS Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

CN71XX implements up to two Serial Gigabit Media-Independent Interface (SGMII)/ Quad Serial Gigabit Media-Independent Interface (QSGMII)/1000BASE-X SerDes interfaces through a single-port/single-lane packet interface.

A full-duplex port/lane consists of four external pins in total: a differential output pair and a differential input pair.

CN71XX couples logic that implements the SGMII, QSGMII, and/or 1000BASE-X protocols on SerDes lanes with a 10/100/1000 802.3 MAC.

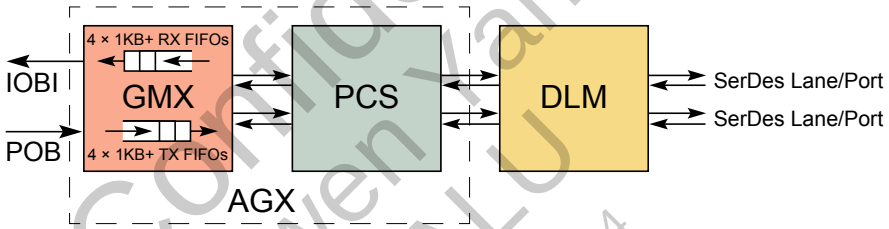
The remainder of this chapter assumes GMX(0..1)_INF_MODE[TYPE] = SGMII/ 1000BASE-X/QSGMII. If this is not the case, the CN71XX SGMII/1000BASE-X SerDes logic is not operational.

23.1 Introduction

Figure 23–1 is a block-diagram showing the CN71XX SGMII interface.

- **GMX** implements the 10/100/1000 802.3 MAC and includes 4KB receive buffers and 4KB transmit buffers. GMX sends packet-input data to PIP/IPD (refer to Chapter 9) and receives packet-output data from PKO (refer to Chapter 10).
- **PCS** implements the SGMII / 1000BASE-X protocols. PCS translates between the MAC and the SerDes lanes.
- CN71XX packages two SerDes lanes into a **dual-lane module (DLM)**.

Figure 23–1 CN71XX SGMII Interface Block Diagram



1000BASE-X is a generic physical-layer protocol designed for full-duplex serial physical media at 1000Mbps operation. It is defined in Clause 36 (PCS and PMA layers) and the optional Clause 37 (Autonegotiation) in the IEEE 802.3-2005 specification. 1000BASE-X converts the parallel GMII interface (defined in Clause 35) required by the 802.3 MAC to a 8b/10b serial code. The CN71XX PCS logic implements the 1000BASE-X protocol on one full-duplex SerDes lane when PCS(0..1)_MISC(0..3)_CTL_REG[MODE] = 1.

The Serial Gigabit Media Independent Interface (SGMII) protocol is a serial protocol that highly leverages 1000BASE-X. SGMII adds/modifies the following features of/to 1000BASE-X:

- SGMII defines a particular physical-layer implementation. CN71XX does not provide a transmit clock and does not require a clock at the receiver. CN71XX implements this clock-less SGMII on one full-duplex SerDes lane.
- SGMII implements 10Mbps and 100Mbps as well as 1000Mbps full- and half-duplex on the 1.25 Gbaud full-duplex lanes. It implements 10/100 over 1.25 Gbaud physical channel by duplicating/sampling above the PCS layer.

Cavium Confidential For Bowen Yang ALU 8/4/2014

- SGMII modifies the optional autonegotiation for the 10/100/1000 PHY to directly communicate the line status to the MAC.

The Quad Serial Gigabit Media Independent Interface (QSGMII) is a method of combining four SGMII interfaces onto a single SerDes lane. Each port has the ability to operate at independent 10/100/1000Mbs speeds and duplex. Refer to QSGMII Specification, Revision 1.3 (February 3, 2011).

The CN71XX PCS logic implements the SGMII/QSGMII protocol when PCS(0..1)_MISC*n*_CTL_REG[MODE] = 0. CN71XX is typically a MAC when it implements the SGMII interface, but can also optionally act as a PHY.

CN71XX implements one SerDes DLM containing two full-duplex SerDes channels.

23.2 GMX Packet-Interface Buffering

The CN71XX packet interface contains a 4KB receive-staging buffer and a 4KB transmit-staging buffer. In SGMII/1000BASE-X mode, CN71XX allocates all the buffer space to the single port. In QSGMII mode (1–4 ports), CN71XX partitions these buffers evenly across the ports, so that each port has at least 1KB receive-staging buffer and 1KB transmit-staging buffer.

- If two ports are used (i.e. if GMX(0..1)_RX_PRTS[PRTS] = GMX(0..1)_TX_PRTS[PRTS] = 0x2), then each port has a 2KB receive FIFO and a 2KB transmit FIFO.
- If all four ports are used (i.e. if GMX(0..1)_RX_PRTS[PRTS] = GMX(0..1)_TX_PRTS[PRTS] = 0x4), then each port has a 1KB receive FIFO and a 1KB transmit FIFO.

In the output case, PKO (described in Chapter 10) has much buffering to add to this packet-interface buffering. More importantly, the CN71XX hardware buffers packet data in L2/DRAM, and this buffering can be many times larger than the packet interface and PKO buffering combined.

In the input case, it is possible for PIP/IPD (described in Chapter 9) to exhaust all of the available L2/DRAM buffering, leaving the packet interface as the only buffering option for input packets. But most CN71XX applications use packet drop and/or per-port backpressure to prevent buffer exhaustion (refer to Sections 9.6 and 9.7). CN71XX can backpressure (via pause packets in full-duplex mode or forced collisions in half-duplex mode) to avoid overflow of the input buffer when its usage exceeds programmable on/off thresholds (refer to Section 23.4.3.2).

23.3 PCS Autonegotiation, Mode, and GMX Configuration

This section covers PCS autonegotiation, the PCS mode, and GMX configuration based on the PCS mode.

Cavium Confidential For bowen yang @ATU 6/27/2014

23.3.1 Autonegotiation

Autonegotiation is described in Clause 37 of the IEEE 802.3-2005 specification. CN71XX supports only base pages, and does not support any next pages. Autonegotiation is enabled on CN71XX by PCS(0..1)_MR(0..3)_CONTROL_REG[AN_EN].

CN71XX starts the autonegotiation state machine in any of the following situations (note that the autonegotiation state machine "starts" even when PCS(0..1)_MR(0..3)_CONTROL_REG[AN_EN] = 0, disabling autonegotiation):

- After PCS reset deasserts. PCS is in reset when PCS(0..1)_MR(0..3)_CONTROL_REG[PWR_DN] = 1 or when PCS(0..1)_MR(0..3)_CONTROL_REG[RESET] was recently set to 1.
- When PCS(0..1)_MR(0..3)_CONTROL_REG[AN_EN] changes value when PCS is not in reset, or
- When PCS(0..1)_MR(0..3)_CONTROL_REG[RST_AN] changes 0→1 when PCS is not in reset, or
- When the PCS receive state machine loses synchronization (i.e. PCS(0..1)_RX(0..3)_SYNC_REG[SYNC] = 0) for an interval defined by the link timer (PCS(0..1)_LINK(0..3)_TIMER_COUNT_REG[COUNT]), or
- When autonegotiation state machine finds an error, as per Clause 36 and 37 of the 802.3 specifications.
- When autonegotiation is enabled and the remote device reinitiates autonegotiation, as per Clause 37 of the 802.3 specifications.

CN71XX disables PCS autonegotiation out of any chip reset sequence (i.e. the reset value of PCS(0..1)_MR(0..3)_CONTROL_REG[AN_EN] is 0). When autonegotiation is disabled, as it is following any chip reset, most of the autonegotiation sequence is avoided, as Clause 37 in the 802.3 specification describes.

CN71XX implements one of three different autonegotiation protocols when autonegotiation is enabled:

- In 1000BASE-X mode (i.e. PCS(0..1)_MISC(0..3)_CTL_REG[MODE] = 1), CN71XX implements the 1000BASE-X autonegotiation protocol
- In SGMII or QSGMII mode (e.g. PCS(0..1)_MISC(0..3)_CTL_REG[MODE] = 0), CN71XX implements the modified autonegotiation protocol:
  - If PCS(0..1)_MISC(0..3)_CTL_REG[MAC_PHY] = 0, CN71XX acts as a MAC.
  - If PCS(0..1)_MISC(0..3)_CTL_REG[MAC_PHY] = 1, CN71XX acts as a PHY.

CN71XX clears PCS(0..1)_MR(0..3)_STATUS_REG[AN_CPT] whenever it starts autonegotiation, then later sets it when the autonegotiation completes.

After autonegotiation completes when autonegotiation is enabled, PCS normally uses the autonegotiation result (i.e. PCS(0..1)_AN(0..3)_RESULTS_REG[DUP,SPD]) for its current mode. PCS(0..1)_MISC(0..3)_CTL_REG[AN_OVRD] optionally switches the CN71XX PCS to use PCS(0..1)_MR(0..3)_CONTROL_REG[DUP,SPDLSB,SPDMSB] instead.

Autonegotiation should not be enabled when PCS(0..1)_MR(0..3)_CONTROL_REG[UNI] = 1.

Cavium Confidential For Bowen Yang ALU 02/27/2017

Section 23.6.4 and Section 23.7.5 describe the interrupts that can occur during/after autonegotiation. Section 23.7.2 describes the software sequence that should be used to execute autonegotiation.

23.3.2 PCS Mode Determination

The current mode of the PCS logic is described by the speed of the interface (1000, 100, 10) and the duplex status (full-duplex or half-duplex).

The current mode of the PCS logic depends on whether autonegotiation is enabled and its status.

If autonegotiation is disabled (i.e. PCS(0..1)_MR*n*_CONTROL_REG[AN_EN] = 0) or if the autonegotiation result is being over-ridden (i.e. PCS(0..1)_MISC*n*_CTL_REG[AN_OVRD] = 1), then PCS(0..1)_MR*n*_CONTROL_REG[DUP,SPDLSB,SPDMSB] indicate the current operating mode of the PCS logic.

If autonegotiation is enabled (i.e. PCS(0..1)_MR*n*_CONTROL_REG[AN_EN] = 1) and the autonegotiation result is not being over-ridden (i.e. PCS(0..1)_MISC*n*_CTL_REG[AN_OVRD] = 0) and autonegotiation is complete (i.e. after PCS(0..1)_MR*n*_STATUS_REG[AN_CPT] = 1), then PCS(0..1)_AN*n*_RESULT_REG[DUP,SPD] indicate the current operating mode of the PCS logic.

Section 23.4.5.1 describes more about the PCS receive byte reduction logic when speed is 100 or 10 (in SGMII mode). PCS(0..1)_MISC*n*_CTL_REG[SAMP_PT] needs to be selected appropriately for these modes. Recommended PCS(0..1)_MISC*n*_CTL_REG[SAMP_PT] values are 5 and 50 for 100 and 10, respectively.

Software can read PCS(0..1)_MR*n*_STATUS_REG[LNK_ST] at any time to determine the Hardware view of the reliability of the channel. When autonegotiation is used, PCS(0..1)_MR*n*_STATUS_REG[LNK_ST] takes into account the health of both the transmit and receive channels, assuming the link partner is following the standard autonegotiation rules. When autonegotiation is not used, PCS(0..1)_MR*n*_STATUS_REG[LNK_ST] is only an indication of the health of the receive channel.

Section 23.7.5 describes how PCS mode status changes can be communicated to software.

Cavium Confidential For Bowen Kang ATD 02/27/2014

23.3.3 GMX Configuration Based on PCS Mode

GMX has several configuration registers that should be set depending on the PCS mode.

Table 23–1 lists the recommended settings for each mode.

Table 23–1 Recommended GMX Configuration Settings

CSR/Field	DUPLEX	1000	100	10
GMX(0..1)_PRT*n*_CFG[DUPLEX]	Full	1	1	1
GMX(0..1)_PRT*n*_CFG[DUPLEX]	Half	0	0	0
GMX(0..1)_PRT*n*_CFG[SPEED]	Full+Half	1	0	0
GMX(0..1)_PRT*n*_CFG[SPPED_MSB]	Full+Half	0	0	1
GMX(0..1)_PRT*n*_CFG[SLOTTIME]	Full+Half	1	0	0
GMX(0..1)_TX*n*_SLOT[SLOT]	Full+Half	512	64	64
GMX(0..1)_TX*n*_BURST[BURST]	Full	0	0	0
GMX(0..1)_TX*n*_BURST[BURST]	Half	8192	0	0

Note that 100 and 10 modes are only relevant in SGMII mode, as per Table 23–1 in Section 23.3.2.

23.4 SGMII/1000BASE-X/QSGMII Features

This section discusses notable features provided by the CN71XX GMX/PCS architecture, some of which being outside the IEEE 802.3 specifications.

23.4.1 QSGMII Mode

QSGMII allows the hardware to extend the SGMII/1000Base-X interface to four ports. To enable QSGMII mode, set GMX(0..1)_INF_MODE[MODE] = QSGMII.

QSGMII combines four ports onto a single SerDes lane. Ports1-3 in each GMX are only available when operating in QSGMII mode. Each port can configure speed (10/100/1000Mbs), duplex (half-duplex/full-duplex), and autonegotiation capabilities independently.

23.4.2 SerDes Lane Polarity

CN71XX PCS logic can control and optionally autodetect the SerDes polarity. Board routing can be simplified in some cases if the two wires that implement a differential pair are swapped.

On the transmit side, PCS(0..1)_TX_RX*n*_POLARITY_REG[TXPLRT] optionally inverts the corresponding SerDes differential output pair.

On the receive side, the PCS receive logic attempts to dynamically determine the polarity as a part of code group synchronization. Each time that code group synchronization occurs (i.e. whenever PCS(0..1)_RX*n*_SYNC_REG[SYNC] transitions from 0 to 1), PCS(0..1)_TX_RX*n*_POLARITY_REG[AUTORXPL] indicates the polarity calculated during that code group synchronization. This result should normally be used, but PCS(0..1)_TX_RX*n*_POLARITY_REG[RXOVRD] can alternatively select the pre-configured receive polarity PCS(0..1)_TX_RX*n*_POLARITY_REG[RXPLRT] instead of the autocalculated

Cavium Confidential For Bowen Yang ALD 02/27/2013

PCS(0..1)_TX_RX*n*_POLARITY_REG[AUTORXPL]. This over-ride may be required in circumstances where the transmitter does not entirely follow the Clause 36 802.3 specifications.

The autocalculated polarity should not be used when any loopback option is enabled, so PCS(0..1)_TX_RX*n*_POLARITY_REG[RXOVRD] should be set to one in that case. Section 23.9 describes the loopback modes.

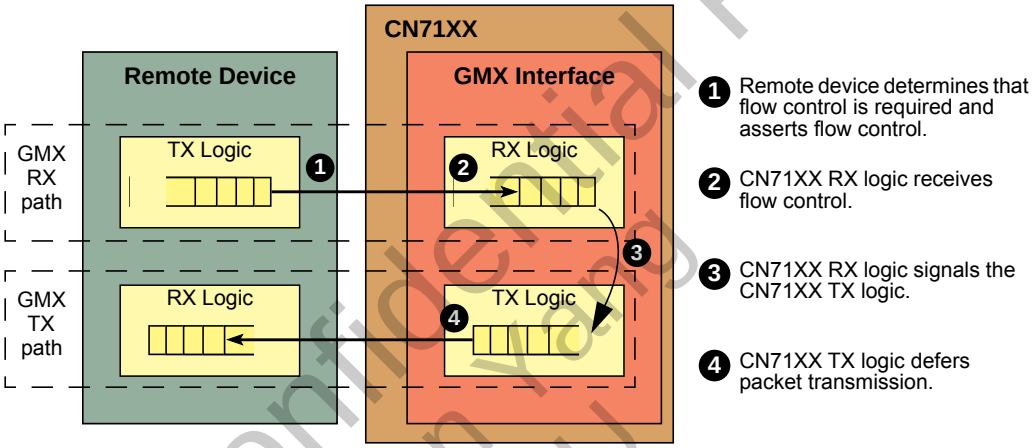
23.4.3 Flow Control

The following subsections discuss how the CN71XX GMX handles flow control for the receive and transmit paths. A remote device's ability to control data flow to the CN71XX is described in Section 23.4.3.1, and CN71XX's ability to control data flow to the a remote device is described in Section 23.4.3.2.

23.4.3.1 Remote-Device Flow Control

Remote-device flow control is shown in Figure 23–2.

Figure 23–2 Remote-Device Flow Control



CN71XX supports flow control for both half- and full-duplex modes of operation. When the receive interface in full-duplex mode, the CN71XX can be configured to receive pause control packets as defined in Annexes 31A and 31B of the 802.3 specification based on the GMX(0..1)_RX(0..3)_FRM_CTL[CTL_BCK/CTL_DRP] registers. Common modes of operation are:

CTL_BCK = 1, CTL_DRP = 1	CN71XX inspects pause packets, sets the transmit defer counter to the TIME parameter in the packets, and then drops the pause packets. CN71XX then defers the transmission of new packets until the timer expires.
CTL_BCK = 0, CTL_DRP = 0	CN71XX processes pause packets like normal packets, so software will see them. Software can optionally implement a backpressure scheme. Software can also set/reset the defer counter by writing to GMX(0..1)_TX(0..3)_SOFT_PAUSE[TIME]. Like pause frames, software must periodically write to GMX(0..1)_TX(0..3)_SOFT_PAUSE[TIME] if the flow condition extends beyond a TIME interval. CN71XX asserts GMX(0..1)_RX(0..3)_INT_REG[PAUSE_DRP] whenever it drops a pause packet in this mode.

Cavium Confidential For Bowen Yang NDA 3/2/2015

CTL_BCK = 0, CTL_DRP = 1	All pause frames are dropped and completely ignored by both hardware and software.
CTL_BCK = 1, CTL_DRP = 0	CN71XX sends pause packets to software like normal packets, and sets the transmit-defer counter to the TIME parameter in the packet. CN71XX then defers the transmission of new outbound packets until the timer expires. As software can see all pause frames, it can optionally implement a backpressure scheme. Software can also set/reset the defer counter by writing to GMX(0..1)_TX(0..3)_SOFT_PAUSE[TIME]. Like pause frames, software must periodically write to GMX(0..1)_TX(0..3)_SOFT_PAUSE[TIME] if the flow condition extends beyond a TIME interval. CN71XX asserts GMX(0..1)_RX(0..3)_INT_REG[PAUSE_DRP] whenever it drops a pause packet in this mode.

Note that the transmit defer counter is not set if either of the following is true:

- minimum packet-length checking is enabled in GMX (i.e. GMX(0..1)_RX(0/1..3)_FRM_CHK[MINERR] = 1) and the packet size is less than minFrameSize (minFrameSize = 60 + 4 × GMX(0..1)_RX(0/1..3)_FRM_CHK[FCSERR])
- FCS checking is enabled (GMX(0..1)_RX(0/1..3)_FRM_CHK[FCSERR] = 1) and the pause packet failed the FCS/CRC check.

Both CTL_BCK and CTL_DRP should be set to 0s in half-duplex mode (pause packets should not be present in half-duplex mode).

If PREAMBLE/SFD is present (i.e. GMX(0..1)_RX(0..3)_FRM_CTL[PRE_CHK] = 1, refer to Section 23.4.5.1 below) and/or if user-defined data is present (i.e. GMX(0..1)_RX(0..3)_UDD_SKP[LEN] ≠ 0, refer to Sections 23.4.5.2 and 23.4.5.3 below) in a received packet, CN71XX discards them before it determines whether a packet is a pause frame. CN71XX recognizes the remaining packet bytes as a pause frame if all these conditions are true:

- Either CTL_BCK or CTL_DRP are set
- The first 6 bytes of the packet (i.e. the L2 destination address) match one of these two enabled comparisons:
  - GMX(0..1)_RX(0..3)_FRM_CTL[CTL_MCST] = 1 and the first 6 bytes equal the standard multicast address 0x01-80-C2-00-00-01
  - GMX(0..1)_RX(0..3)_FRM_CTL[CTL_SMAC] = 1 and the first 6 bytes equal GMX(0..1)_SMAC(0..3)[SMAC]
- The 13th and 14th bytes (i.e. the L2 type field) equal 88-08 (MAC control), and
- The 15th and 16th bytes (i.e. the MAC control opcode) equal 00-01 (PAUSE)

Note that PREAMBLE/SFD should normally be present in received pause frames (and GMX(0..1)_RX(0..3)_FRM_CTL[PRE_CHK] should be set) whenever either CTL_BCK or CTL_DRP are set.

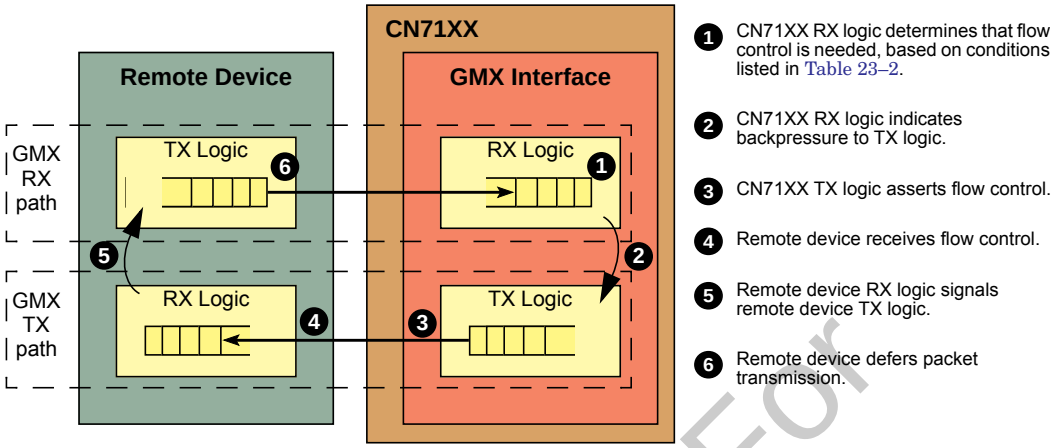
In half-duplex mode, the remote device creates collisions to flow control. GMX views these collisions as normal collisions.

Cavium Confidential For Bowen Yang ALU 02/21/2014

23.4.3.2 CN71XX Flow Control

CN71XX flow control is shown in Figure 23–3.

Figure 23–3 CN71XX Flow Control



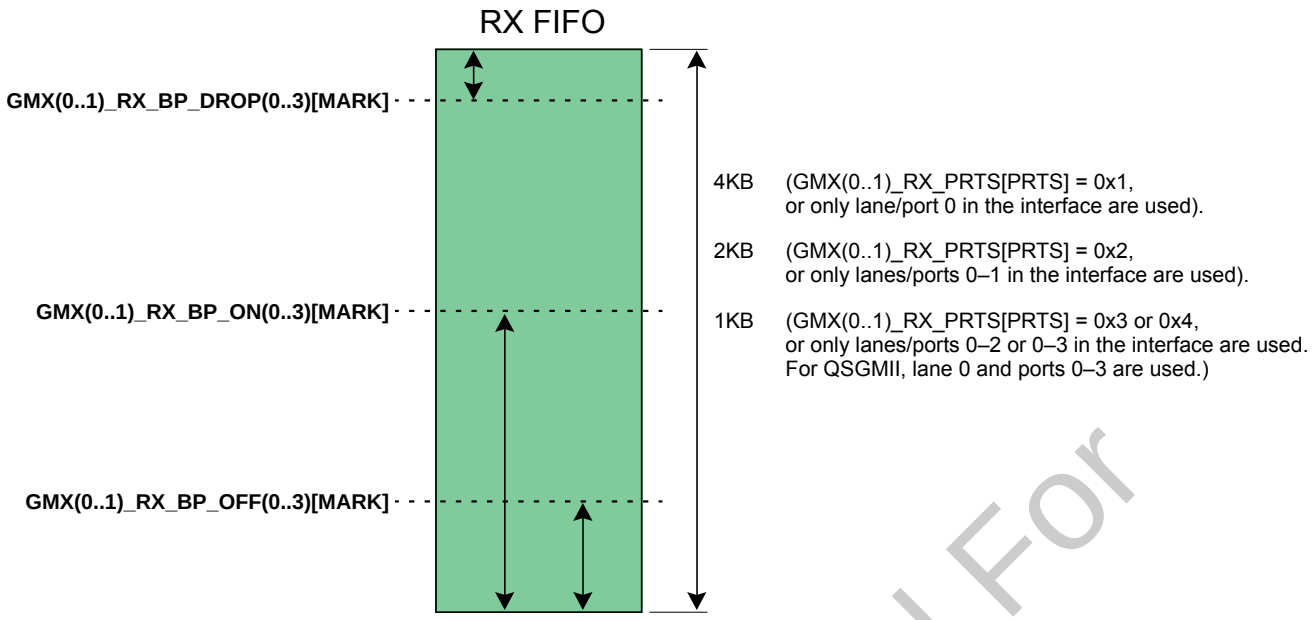
CN71XX controls the flow of packets from the transmitting device under the conditions specified in Table 23–2.

Table 23–2 Flow Control Conditions

Condition	Description
Receive FIFO Flow Control	The receive FIFO can control the flow as it fills up. Each port/lane has a dedicated receive FIFO with dedicated controls. Figure 23–4 shows the relative FIFO depths for each programmable parameter. CN71XX begins to flow control to port when the FIFO accumulates GMX(0..1)_RX_BP_ON(0..3) data. Flow control is removed once the FIFO drops below GMX(0..1)_RX_BP_OFF(0..3) data.
Per-Port Flow Control	PIP/IPD can also communicate per-port flow control. This flow control is exhibited the same way as receive FIFO flow control. Refer to Sections 9.6 and 9.7 for more detailed information.
Software Flow Control	Software can use GMX(0..1)_TX_OVR_BP[EN/BP] to force or ignore the flow-control state as desired. If GMX(0..1)_TX_OVR_BP[EN*n*] = 1 for a port, the flow-control state will be GMX(0..1)_TX_OVR_BP[BP*n*]. This feature is intended for use only in system testing.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 23–4 Receive FIFO Parameters

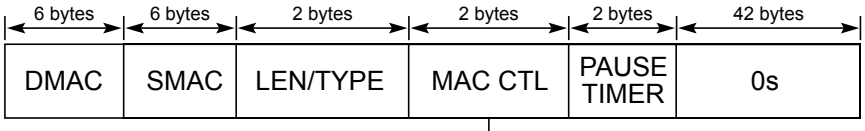


In half-duplex mode when flow control is active for the port, CN71XX forces collisions by sending JAM packets.

In full-duplex mode, CN71XX can automatically generate pause packets. Figure 23–5 shows CN71XX's autogenerated pause packets. Much of the packet is programmable, and can differ for different ports. CN71XX sends these 60-byte packets out the SerDes interface similar to normal packets — refer to Figure 23–11 and Section 23.4.6. The only processing difference is that FCS can be added to autogenerated pause packets in cases when it isn't added to normal packets generated by PKO. Section 23.4.6.4 below describes the FCS processing differences. When flow control is active, CN71XX autogenerates pause packets at a frequency indicated by GMX(0..1)_TX(0..3)_PAUSE_PKT_INTERVAL[INTERVAL]. If GMX(0..1)_TX(0..3)_PAUSE_ZERO[SEND] = 1, CN71XX autogenerates a pause packet with a pause time of 0 when flow-control deasserts, which can allow the remote transmitter to restart more quickly.

Figure 23–5 Generated Pause Packet Format

DMAC	=	GMX(0..1)_TX_PAUSE_PKT_DMAC[SMAC]
SMAC	=	GMX(0..1)_SMAC*n*[SMAC]
LEN/TYPE	=	GMX(0..1)_TX_PAUSE_PKT_TYPE[TYPE]
MAC CTL	=	00–01
PAUSE TIMER	=	GMX(0..1)_TX*n*_PAUSE_PKT_TIME[TIME]



CN71XX converts this packet the same way as the **Initial Packet Data (from PKO)** shown in Figure 23–11.

Cavium Confidential For Bowen Liang ALU 9/27/2014

Note that CN71XX should likely only autogenerate pause packets (in full-duplex mode) when it automatically inserts PREAMBLE/SFD (i.e. when GMX(0..1)_TX(0..3)_APPEND[PREAMBLE] = 1), and when there is no UDD present in out-going packets. Section 23.4.6.4 describes PREAMBLE/SFD autoinsertion. Section 23.4.5.3 describes the UDD in received packets. The CN71XX GMX transmit logic is not aware of any UDD in transmitted packets. In cases where PREAMBLE/SFD is not automatically generated or when there is UDD present in out-going packets, software should likely generate outgoing pause packets.

23.4.4 Receive Packet Dropping

There are several mechanisms within CN71XX that will drop packets:

- preamble errors
- receive-FIFO packet dropping
- pause packet drops
- DMAC filters
- PIP/IPD per-QOS or per-port admission control
- Receive collisions

These mechanisms are described in the following subsections.

23.4.4.1 PREAMBLE Errors

If preamble checking is enabled (GMX(0..1)_RX*n*_FRM_CTL[PRE_CHK] = 1) and the packet does not send a valid PREAMBLE followed by SFD, the packet and all subsequent data until the next SGMII idle cycle is dropped. Normally only a single packet is dropped, but if the error occurs at the beginning of a burst (in 1000 half-duplex mode), then multiple packets are dropped.

A valid PREAMBLE must include the following elements:

- First, an SPD (/S/) code group (CN71XX PCS logic converts this to a PREAMBLE (0x55) for the GMX logic)
- Last, a data code group converting to SFD (0xD5)
- In between these two code groups, there may be additional data code groups that do not convert to SFD.

23.4.4.2 Receive-FIFO Packet Dropping

When the receive FIFO exceeds the GMX(0..1)_RX_BP_DROP*n* (as shown in Figure 23–4), the packet cannot be buffered and must be dropped. If there is room in the FIFO for a partial portion of the packet, the beginning of the packet is buffered but is sent down the IOBI with the PARTIAL opcode (indicating a drop). As described is Section 9.5, partial packets can be recognized by software when WORD2[RE] = 1 and WORD2[OPCODE] = 1 (partial error). All other packets are completely dropped and are not sent on the IOBI.

Both partial drops and complete drops set the CIU_INT*n*_SUM0[GMX_DRP] bit in CIU and can raise an interrupt based on CIU_INT*n*_EN0[GMX_DRP].

In most configurations, GMX receive-FIFO packet drops should be non-existent or rare. If drops are required, PIP/IPD packet drops, described in Section 23.4.4.5 below, should be used instead in most cases.

Cavium Confidential For Bowen yang ALU 06/27/2014

23.4.4.3 Pause Packet Drops

CN71XX drops any incoming pause packet, when GMX(0..1)_RX*n*_FRM_CTL[CTL_DRP] = 1. See Section 23.4.3.1 for additional details. In addition, pause packets can be dropped if GMX(0..1)_RX*n*_FRM_CTL[CTL_DRP] = 0 and the receive-FIFO is full. See Section 23.4.4.2 for additional details.

CN71XX sets GMX(0..1)_RX*n*_INT_REG[PAUSE_DRP] = 1 and GMX_RX_PAUSE_DROP_TIME[STATUS] = pause_time extracted from the last pause packet that is dropped when GMX(0..1)_RX*n*_FRM_CTL[CTL_DRP]=0 and a pause packet cannot be buffered in the Receive FIFO. The hardware can optionally raise an interrupt if GMX(0..1)_RX*n*_INT_EN[PAUSE_DRP] = 1.

23.4.4.4 DMAC Filter

Each port/lane can store up to 32 full MAC addresses to match against (GMX(0..1)_RX*n*_ADR_CAM_ALL_EN and GMX_RX_ADR_CAM*n*). (CN71XX can also operate in a legacy mode in which each port/lane has access to 8 MAC addresses using GMX(0..1)_RXn_ADR_CAM_ALL_EN.) CN71XX can be configured to accept or drop packets based on this address match (GMX(0..1)_RX*n*_ADR_CTL[CAM_MODE]). In addition, CN71XX can be configured to accept or reject multicast or broadcast packets (GMX(0..1)_RX*n*_ADR_CTL[MCST/BCST]).

The full algorithm is shown in Figure 23–6.

Figure 23–6 DMAC Filter Algorithm

```
bool dmac_addr_filter(uint8 prt, uint48 dmac) {
  ASSERT(prt >= 0 && prt <= 3);
  if (is_bcst(dmac))                                  //broadcast accept
    return (GMX_RX{prt}_ADR_CTL[BCST] ? ACCEPT : REJECT);
  if (is_mcst(dmac) & GMX_RX{prt}_ADR_CTL[MCST] == 1)     //multicast reject
    return REJECT;
  if (is_mcst(dmac) & GMX_RX{prt}_ADR_CTL[MCST] == 2)     //multicast accept
    return ACCEPT;
  cam_hit = 0;
  for (i=0; i<32; i++) {
    if (GMX_RX{prt}_ADR_CAM_ALL_EN[EN<i>] == 0)
       continue;
    uint48 unswizzled_mac_adr = 0x0;
    for (j=5; j>=0; j--) {
       unswizzled_mac_adr = (unswizzled_mac_adr << 8) | GMX_RX{i>>3}_ADR_CAM{j}[ADR<(i&7)*8+7:(i&7)*8>];
    }
    if (unswizzled_mac_adr == dmac) {
       cam_hit = 1;
       break;
    }
  }
  if (cam_hit)
    return (GMX_RX{prt}_ADR_CTL[CAM_MODE] ? ACCEPT : REJECT);
  else
    return (GMX_RX{prt}_ADR_CTL[CAM_MODE] ? REJECT : ACCEPT);
}
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

23.4.4.5 PIP/IPD Per-QOS and Per-Port Admission Control

Sections 9.6 and 9.7 discuss PIP/IPD's QOS and per-port admission control. Packets can optionally be dropped when they fail QOS or per-port threshold checks.

23.4.4.6 Receive Collisions

When operating in half-duplex mode (GMX(0..1)_PRT*n*_CFG[DUPLEX] = 0), the receiver drops any packets in which the slottime was not satisfied. Note that late collisions are not normally dropped since they satisfy the slottime. Late collisions can be dropped due to any of the other reasons above.

23.4.5 Receive Packet Processing and Format

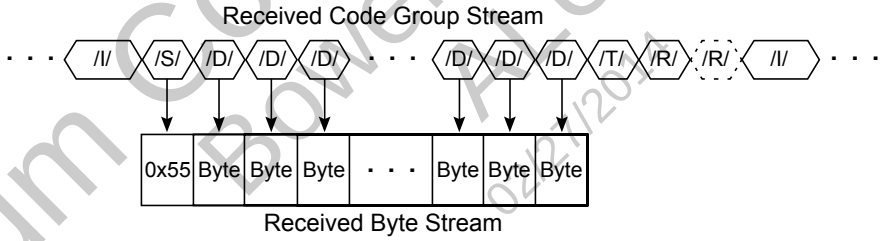
The following subsections describe the processing and format of received packets.

23.4.5.1 Code Group Conversion to Packet Bytes

After the CN71XX SGMII/1000BASE-X/QSGMII receive logic obtains bit lock and code group synchronization (i.e. after PCS(0..1)_RX*n*_SYNC_REG[BIT_LOCK,SYNC] are set to 1), it then reliably converts received packets from a sequence of code groups to a byte stream.

Figure 23–7 depicts a typical received packet and the byte stream that CN71XX first converts it to. The received code group stream consists of a Start_of_Packet Delimiter (SPD, /S/) followed by the packet data, followed by an End_of_Packet Delimiter (EPD, /T//R/ or /T//R//R/). The second carrier-extend (/R/) is present, if necessary, to make the packet (everything excluding the idle ordered sets) an even number of code groups. The CN71XX first converts the packet into the indicated byte stream. CN71XX translates the SPD code group into 0x55 (typically considered a PREAMBLE byte), and the remaining code groups according to the data code group rules specified in Clause 36 of the IEEE 802.3 specifications.

Figure 23–7 Converting Code Groups to Bytes



The code group stream in Figure 23–7 depicts a typical packet, not all possible packets. Another notable possibility is the 1000 Half-Duplex case, where the packet may be followed by carrier-extends (i.e. more /R/'s) or may be in the middle of a burst surrounded by carrier-extends.

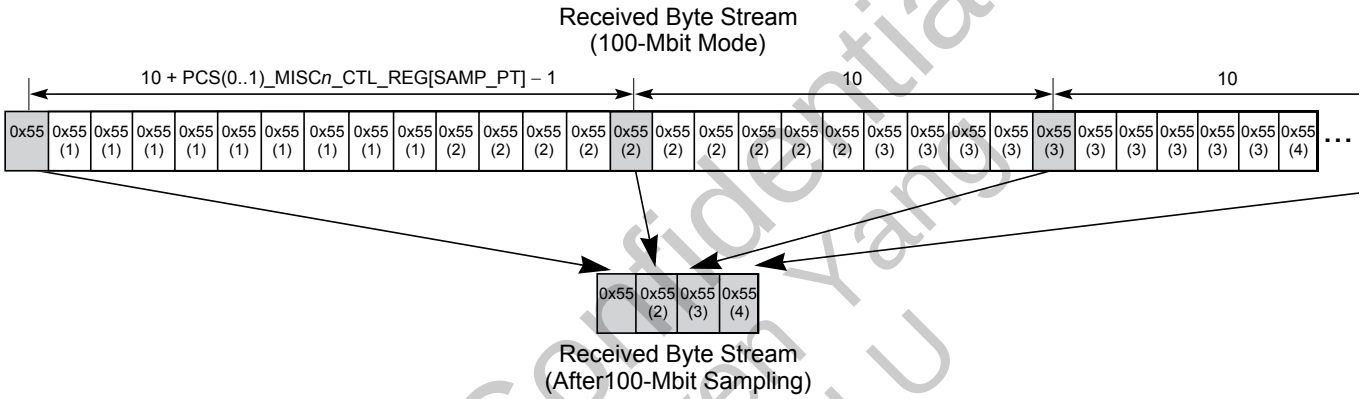
Cavium Confidential For Bowen Zhang 02/27/2015

When the CN71XX SGMII/1000BASE-X/QSGMII logic is in 10 or 100 modes, it next reduces the duplicated received bytes. Let the duplication factor be X, where X is 10 for 100 Mbit, and 100 for 10 Mbit. CN71XX PCS configuration reduces the received packet by selecting programmable samples from replicated byte stream, according to the following rules:

- The SPD (/S/) code group and the first received byte is the reference point. CN71XX PCS logic always converts the SPD code group into the byte 0x55 (i.e. a PREAMBLE byte) and makes it the first byte of the reduced packet.
- Next, CN71XX PCS logic skips over the next X + PCS(0..1)_MISC*n*_CTL_REG[SAMP_PT]–2 received bytes, and samples the X + PCS(0..1)_MISC*n*_CTL_REG[SAMP_PT]–1th byte as the next byte of the reduced packet.
- After that, CN71XX continually skips over X–1 bytes and samples each Xth byte until it observes the EPD.

Figure 23–8 depicts an example of this process for 100 Mbit mode when PCS(0..1)_MISC*n*_CTL_REG[SAMP_PT] = 0x5, which is the recommended value for 100 mode (where X = 10). The recommended PCS(0..1)_MISC*n*_CTL_REG[SAMP_PT] setting is X/2.

Figure 23–8 100 Mbit Byte Stream Reduction Example



The Figure 23–8 example shows a case where the initial byte is duplicated X times in the packet. (X is 10 in this example case because the example is 100 Mbit. The 10 “copies” include the initial “0x55” (from the SPD) plus nine copies of “0x55 (1).”) This is not the only legal possibility in SGMII mode. It is also legal for the first byte to be duplicated X–1 rather than X times. But each subsequent byte must be duplicated exactly X times, as in Figure 23–8. In either case (X or X–1 duplication of the first byte), the recommended PCS(0..1)_MISC*n*_CTL_REG[SAMP_PT] settings will reduce the byte stream correctly.

The CN71XX SGMII/1000BASE-X/QSGMII logic next converts the byte stream into packet bytes. Figure 23–9 shows this. This step processes the PREAMBLE/SFD that is typically in the received packet. CN71XX assumes that a PREAMBLE/SFD is present in the received packet when GMX(0..1)_RX*n*_FRM_CTL[PRE_CHK] = 1. When present, the PREAMBLE/SFD is typically discarded. But CN71XX can also optionally retain all the received bytes as packet bytes when GMX(0..1)_RX*n*_FRM_CTL[PRE_STRP] = 0 in full-duplex mode. This may be useful for debug or when CN71XX communicates with a device other than a PHY.

Cavium Confidential For Bowen Yang ALU 02/26/2014

Figure 23–9 Conversion from Byte Stream to Packet Bytes

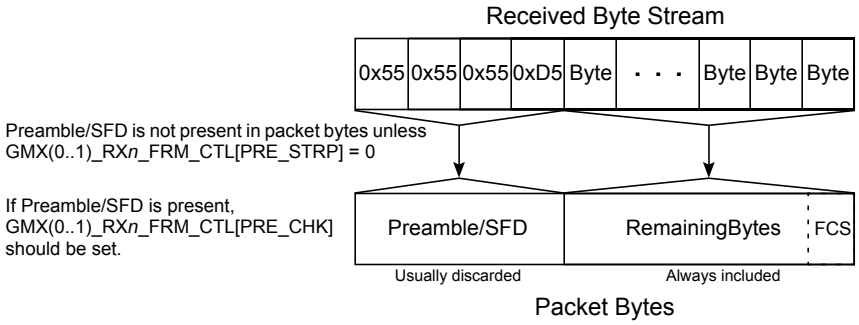


Figure 23–9 shows a case where there are three PREAMBLE bytes followed by the SFD byte. More commonly, the byte stream will contain either 6 or 7 PREAMBLE bytes followed by the SFD byte. (Equivalently, when in 1000 mode, the packet will typically contain SPD code followed by 5 or 6 PREAMBLE data codes, followed by an SFD data code.)

NOTE: Per the 1000BASE-X standard described in Clause 36 of the 802.3 standard, transmitting PCS logic may replace either the first one or two bytes (normally, both PREAMBLE bytes) of outgoing packets with the single SPD code group, depending on when a packet starts in an odd/even sequence, which could be unpredictable. (This is why typical received packets may contain an SPD code followed by either 5 or 6 PREAMBLE code groups, rather than a fixed number of PREAMBLEs.) Receivers typically discard the PREAMBLE/SFD and recover the actual packet data bytes, so this randomness is not an issue. If, however, CN71XX does not strip the PREAMBLE/SFD (i.e. if GMX(0..1)_RX*n*_FRM_CTL[PRE_STRP] = 0), then subsequent hardware/software may be confused by this unpredictability. In general, care must be taken to avoid system issues like this. For, example, a CN71XX transmitter can enable ALIGN mode to eliminate this unpredictability, as Section 23.4.6.6 describes below.

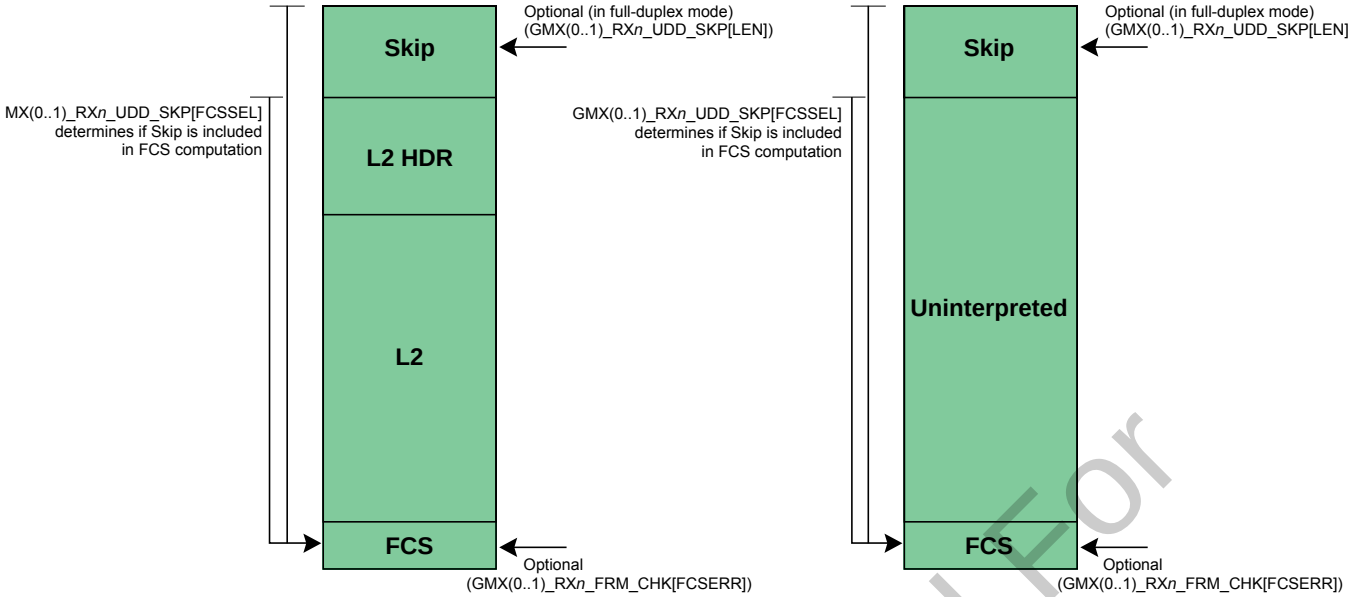
Unless GMX drops a received packet or packet bytes (refer to Section 23.4.4), PIP/IPD receives the packet data bytes for further processing via the IOBI shown in Figure 23–1. Note that the received packet data may contain an FCS at the end, as Figure 23–9 shows. Section 23.4.5.6 below describes the GMX FCS processing.

23.4.5.2 Receive Packet Formats

Figure 23–10 shows the supported receive-packet formats, which differ from those described in Section 9.2.

Cavium Confidential For ALL Bowen Yang 02/27/2014

Figure 23–10 Packet Formats



The format can include the optional skip/UDD field, the L2 header, the remainder of the L2 packet, and the optional standard FCS. The next sections discuss these components.

CN71XX normally inspects the packet bytes to extract the L2 parameters that are required for various tasks and checks.

23.4.5.3 Receive User-Defined Data

The packet parser must be able to determine where the L2 header is in the packet to extract the correct data. Normally, the L2 header will be at the start of the packet bytes. In full-duplex mode only, CN71XX also supports user-defined data (UDD) prior to the L2 header. CN71XX can support 0–64 bytes of UDD data based on GMX(0..1)_RX*n*_UDD_SKP[LEN]. Note that this is a static per-port configuration that cannot change on a per-packet basis. CN71XX can also be configured to compute the FCS over the UDD data or just the L2 header and L2 payload based on GMX(0..1)_RX*n*_UDD_SKP[FCSSEL].

Systems have two choices when skip bytes exist in packet bytes (followed by an ordinary ethernet L2 header):

1. If the skip amount is constant for all packets, set GMX(0..1)_RX*n*_UDD_SKP[LEN] to the total SKIP (Skip I + PKT_INST_HDR + Skip II, refer to Section 9.2.)

2. If the amount of total skip data before the L2 HDR is variable, the L2 HDR inspection must be disabled (see Section 23.4.5.4 for more details).

Note that this discussion applies only to the receive path. GMX is not aware of any UDD data present in out-going packets. If UDD does need to be present in out-going packets, pause packet autogeneration should likely be disabled, as the autogenerated packets cannot contain any UDD. Refer to Section 23.4.3.2 above. Outbound FCS is always calculated over all bytes in an outbound packet. If UDD-like bytes should not be included in the FCS for outbound packets, software should generate the FCS for the packets. Section 23.4.6.3 below describes outbound FCS calculations.

Cavium Confidential For Bowen Yang ALU 02/27/2014

23.4.5.4 Receive L2 Headers

The packet interface supports all six of the L2 header types as defined in Figure 9–7 in Chapter 9, Packet Input Processing/Input Packet Data Unit (PIP/IPD).

Some systems may not use the standard Ethernet L2 HDR header or may develop custom headers. Normal L2 HDR inspection can be disabled on a per-port/lane basis. If the L2 HDR is not present in the packet, the L2 HDR inspection must be disabled. See Section 23.4.5.5 for more details.

23.4.5.5 Disabling Receive L2 Header Inspection

If the L2 header is not present or is not at a fixed offset in the packet, L2 header inspection should be disabled as follows:

- **DMAC filtering** – the L2 DMAC field cannot be extracted from the packet. The DMAC filter can be put into promiscuous mode by setting the following parameters:

  GMX(0..1)_RX*n*_ADR_CTL[CAM_MODE] = 0
  GMX(0..1)_RX*n*_ADR_CTL[MCST] = 0
  GMX(0..1)_RX*n*_ADR_CTL[BCST] = 1
  GMX(0..1)_RX*n*_ADR_CAM_EN[EN] = 0.

- **Pause-packet recognition** – the L2 DMAC and type fields cannot be extracted from the packet. Pause-packet recognition can be disabled by setting the following parameters:

  GMX(0..1)_RX*n*_FRM_CTL[CTL_BCK] = 0
  GMX(0..1)_RX*n*_FRM_CTL[CTL_DRP] = 0

23.4.5.6 Receive FCS

This optional FCS shown in Figure 23–10 can optionally be checked by GMX but GMX always forwards the FCS bytes to PIP/IPD. PIP/IPD then optionally discards the FCS bytes. Refer to Section 9.2.5.

CN71XX checks the standard four-byte CRC-32 FCS field in the packet when GMX(0..1)_RX*n*_FRM_CHK[FCSERR] = 1. Section 23.6.1 describes the CN71XX response when the FCS check is enabled and fails.

The CRC check normally covers all prior packet bytes. CN71XX optionally excludes the UDD/skip bytes. When GMX(0..1)_RX*n*_UDD_SKP[FCSSEL] = 1, CN71XX excludes the first GMX(0..1)_RX*n*_UDD_SKP[LEN] bytes from the FCS calculation.

23.4.6 Transmit Packet Processing

CN71XX SGMII/1000BASE-X/QSGMII transmit logic can autogenerate pause frames (refer to Section 23.4.3.2 and Figure 23–4). But most packet transmissions occur as packets arrive at the packet interface from PKO via the POB shown in Figure 23–1. This section describes the CN71XX transmit logic for those outbound packets. The processing of the autogenerated packets and PKO packets is identical except for FCS. Section 23.4.6.4 describes the differences.

23.4.6.1 Transmit Interpacket Gap (IPG)

CN71XX SGMII/1000BASE-X/QSGMII transmit logic enforces minimum interpacket gaps (IPGs) between all transmitted packets as per the 802.3 specifications. (IPG is also known as interFrameSpacing or interFrameGap.) CN71XX enforces this gap

Cavium Confidential For Bowen Yang AU 6/17/2014

between all packets that it transmits, which includes both the usual PKO-generated packets and any autogenerated pause packets. (Section 23.4.3.2 describes pause packet autogeneration.) Most IPGs include idle ordered_sets to separate the packets. Each idle ordered_set is two code groups and must be aligned to even cycles. GMX(0..1)_TX*n*_SGMII_CTL[ALIGN] controls the ALIGN mode of the link which can have bandwidth and data transmission implications. See Section 23.4.6.7, Transmit Conversion to Code Groups, for further details.

CN71XX splits the IPG into the first 2/3 (GMX(0..1)_TX_IFG[IFG1]) and the last 1/3 (GMX(0..1)_TX_IFG[IFG2]) for proper half-duplex deference, as per Section 4.2.3.2.1 in the IEEE 802.3-2005 specification.

In 10 and 100 modes, CN71XX guarantees that at least (GMX(0..1)_TX_IFG[IFG1] + GMX(0..1)_TX_IFG[IFG2])×X/2 – 1 idle ordered_sets exist between any two packets, where X is the replication factor.

When idle ordered sets are used in the 1000 mode IPG, as is always the case in full-duplex 1000 mode, and sometimes the case in half-duplex 1000 mode, CN71XX ensures that at least (GMX(0..1)_TX_IFG[IFG1] + GMX(0..1)_TX_IFG[IFG2])/2 – 1 idle ordered_sets (i.e. /I/) are present between any two transmitted packets.

When bursting is enabled in 1000 half-duplex mode on CN71XX, some IPGs are carrier-extends (/R/) rather than idle ordered_sets. Carrier-extends consume only one code group rather than two. When carrier-extends are used, CN71XX ensures that at least GMX(0..1)_TX_IFG[IFG1] + GMX(0..1)_TX_IFG[IFG2] – 1 carrier-extends exist between any two transmitted packets. Note that this count includes the one or two carrier-extends that are part of the EPD sequence.

23.4.6.2 Packet Transmission

The transmit FIFO is 4KB per interface and is partitioned into logical FIFOs, one per output port/lane. The FIFOs always allocate memory on power-of-2 ports as shown in Table 23–3.

Table 23–3 FIFO Memory Allocation

Number of Ports	GMX(0..1)_TX_PRTS [PRTS]	PKO_REG_GMX_PORT_MODE[MODE]	Allocated Memory (per port)
1	0x1	0x4	4KB
2	0x2	0x3	2KB

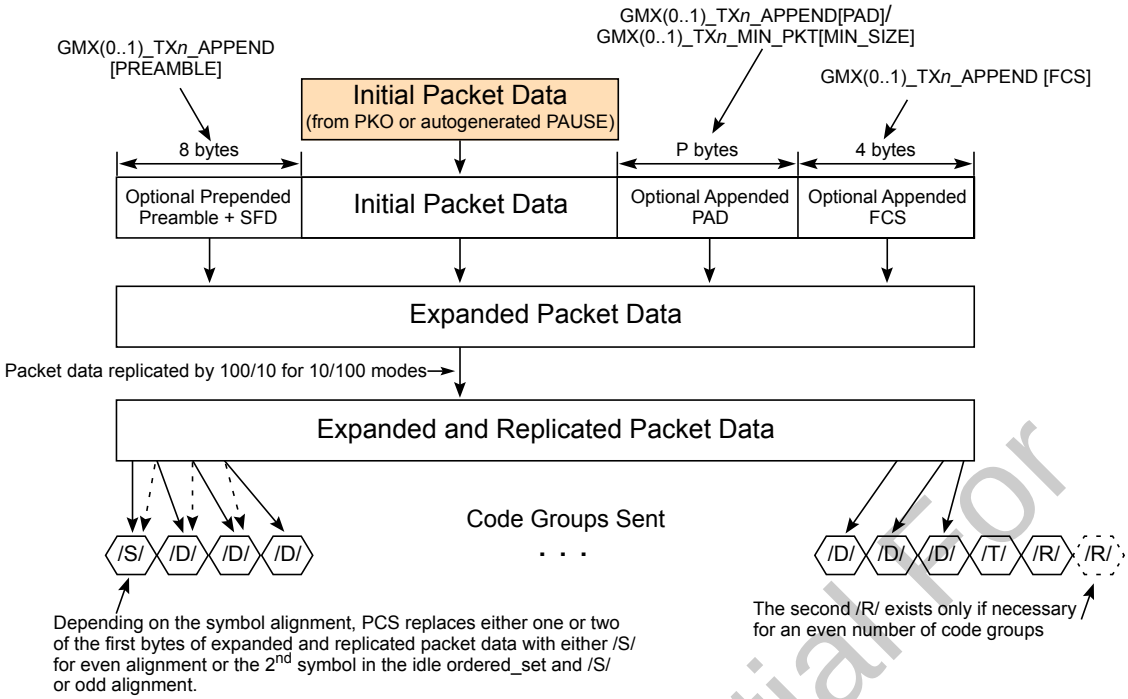
Software must correctly configure PKO based on the transmit FIFO size (i.e. PKO_REG_GMX_PORT_MODE[MODE*] must be set as indicated by the table above).

The GMX transmit state machine begins packet transmission when the FIFO contains a complete packet or if the FIFO length exceeds a programmed number of bytes (GMX(0..1)_TX*n*_THRESH). The threshold value should be large enough to avoid underflow and small enough to avoid excessive latencies. The reset value of the register is recommended for most configurations. For additional protection against packet underflow, PKO engines can be put into threshold mode, where the engines will hold onto packets as long as possible before sending. Refer to PKO_REG_ENGINE_THRESH and Chapter 10.

Figure 23–11 depicts the CN71XX SGMII/1000BASE-X/QSGMII logic transmit packet translation from PKO/pause bytes to code groups on the outbound SerDes. The next sections describe this translation in detail.

Cavium Confidential For Bowen Yang ALU 02/7/2014

Figure 23–11 Outbound Conversion to Code Groups



Note that Figure 23–11 only shows a typical case, and does not depict all possibilities. A notable exception includes the 1000 Half-Duplex burst case, where the packet may be followed by carrier-extends (i.e. more /R/'s) or may be in the middle of a burst surrounded by carrier-extends. Section 23.4.6.8 describes more about CN71XX's burst processing.

23.4.6.3 Optional Transmit PAD Append

First, the CN71XX SGMII/1000BASE-X/QSGMII transmit logic optionally appends zeroes to make all packets satisfy a minimum size requirement. When GMX(0..1)_TX*n*_APPEND[PAD] = 1, this PAD append forces all packets to be at least GMX(0..1)_TX*n*_MIN_PKT[MIN_SIZE]+1 bytes. In Figure 23–11, P is either zero bytes if the packet is large enough, else

P = GMX(0..1)_TX*n*_MIN_PKT[MIN_SIZE] + 1 – (# of bytes from PKO/pause)

Note that CRC is normally generated by the CN71XX SGMII/1000BASE-X/QSGMII logic following the append. If so, then all packets become at least GMX(0..1)_TX*n*_MIN_PKT[MIN_SIZE]+5 bytes. In that typical case, GMX(0..1)_TX*n*_MIN_PKT[MIN_SIZE] = 59 produces the usual minimum packet of 64 bytes.

Cavium Confidential For Bowen Yang 02/27/2014

23.4.6.4 Optional Transmit FCS Append

Second, the CN71XX SGMII/1000BASE-X/QSGMII transmit logic optionally calculates and appends the standard 4-byte CRC-32 FCS. When GMX(0..1)_TX*n*_APPEND[FCS] = 1, CN71XX appends the FCS, as Figure 23–11 shows.

Note that the FCS calculated for outgoing packets always covers all bytes in outgoing packets. This is unlike the FCS calculations described by Figure 23–10 and Section 23.4.5.6, where the UDD bytes can optionally be excluded from the FCS calculation for inbound packets. If some bytes should be excluded from the FCS for outbound packets, the FCS must be generated by software.

When GMX(0..1)_TX*n*_APPEND[FCS] = 1, GMX transmit FCS processing is identical for both PKO-generated and autogenerated packets. When GMX(0..1)_TX*n*_APPEND[FCS] = 0 and GMX(0..1)_TX*n*_MIN_PKT[MIN_SIZE] ≤ 59 (i.e. when FCS is not appended for normal PKO-generated packets and no pad is added to autogenerated pause packets), GMX(0..1)_TX*n*_APPEND[FORCE_FCS] can be set to one. When GMX(0..1)_TX*n*_APPEND[FORCE_FCS] is set, CN71XX generates FCS only for the autogenerated outbound packets, and does not generate FCS for normal PKO-generated packets. This may be useful when software generates the FCS for out-going packets.

23.4.6.5 Optional Transmit PREAMBLE/SFD Prepend

Third, the CN71XX SGMII/1000BASE-X/QSGMII transmit logic optionally prepends the standard seven bytes of PREAMBLE (0x55) followed by SFD (0xD5). When GMX(0..1)_TX*n*_APPEND[PREAMBLE] = 1, CN71XX prepends the PREAMBLE+SFD, as Figure 23–11 shows.

The CN71XX always prepends the full PREAMBLE+SFD when GMX(0..1)_TX*n*_APPEND[PREAMBLE]=1. Section 23.4.6.7, Transmit Conversion to Code Group, describes transformations by the PCS layer which can reduce the final transmitted PREAMBLE length to achieve maximum performance. It is recommended that GMX(0..1)_TX*n*_SGMII_CTL[ALIGN] = 0 when GMX(0..1)_TX*n*_APPEND[PREAMBLE]=1 for maximum performance, which can reduce the PREAMBLE+SFD length from 8 to 7 bytes.

If the PREAMBLE+SFD is not automatically generated by CN71XX (i.e. if GMX(0..1)_TX*n*_APPEND[PREAMBLE] = 0), as might be appropriate when CN71XX is not communicating with a standard PHY, there are several things to note:

(1) It is possible for the packet bytes from PKO to include any form/size PREAMBLE/SFD. Care must be taken regarding CRC generation, though, if some form of PREAMBLE/SFD is included in the PKO bytes, as these bytes will be included in any hardware-generated CRC calculation. Refer to Section 23.4.6.4.

(2) In its conversion from expanded packet data to code groups (refer to Section 23.4.6.7), CN71XX replaces either one or two of the first bytes of expanded and replicated packet data. These bytes are always recovered as one 0x55 (i.e. one PREAMBLE byte) at the receiver, regardless of their original value. It is recommended that the replaced bytes always be 0x55.

Cavium Confidential Beyond Ying 02/27/2014

(3) Whether CN71XX replaces one or two bytes is unpredictable in 1000 mode unless ALIGN mode is enabled, as Section 23.4.6.7 describes below. When ALIGN mode is not enabled and the packet does not include a PREAMBLE+SFD, it may be difficult for the receiver to properly recover the packet data. See Section 23.4.6.7, Transmit Conversion to Code Group for further details.

(4) CN71XX likely cannot autogenerate pause packets since the remote receiver will likely not be able to interpret the pause packet. Section 23.4.3.2 and Figure 23–5 describe pause packets and their autogeneration.

23.4.6.6 Transmit Replication for 10 and 100 Modes

Fourth, the CN71XX SGMII/1000BASE-X/QSGMII transmit logic replicates the expanded packet data when necessary before it converts the packet to code groups, as Figure 23–11 shows. The replication factor is 100 and 10 for 10 and 100 modes, respectively. CN71XX replicates each byte of the expanded packet data independently. There is no replication in 1000 mode.

CN71XX also replicates all autogenerated pause packets by the same factor as all packet data in 10 and 100 modes. Section 23.4.3.2 describes the autogenerated pause packets.

Section 23.4.6.1 describes the affect of replication on IPG.

23.4.6.7 Transmit Conversion to Code Groups

Fifth, the CN71XX SGMII/1000BASE-X/QSGMII transmit logic converts the expanded and replicated packet data into code groups, as Figure 23–11 shows.

CN71XX replaces either one or two of the first bytes of the expanded and replicated packet data with the SPD (/S/). In 10 and 100 modes, CN71XX always replaces exactly one byte with the SPD code group. In 1000 mode, it is normally unpredictable whether the SPD code group replaces one or two bytes. CN71XX also contains ALIGN mode to force predictability in this translation in 1000 mode. ALIGN mode is enabled when GMX(0..1)_TX*n*_SGMII_CTL[ALIGN] = 1. When ALIGN mode is enabled, the SPD always replaces exactly one byte of the expanded and replicated packet data. ALIGN mode has no effect in 10 and 100 modes. CN71XX ALIGN mode also applies to the autogenerated pause packets. Section 23.4.3.2 describes the autogenerated pause packets.

This unpredictability is due to the specifics of the PCS TX state machine. If the CN71XX decides to schedule a packet for transmission in an odd cycle, the PCS TX state machine is not able to send the /S/, as the idle ordered_set must first be completed. If GMX(0..1)_TX*n*_SGMII_CTL[ALIGN] = 1, the packet is stalled, increasing the effective IPG which reduces the link bandwidth, but preserving the packet in its entirety. If GMX(0..1)_TX*n*_SGMII_CTL[ALIGN] = 0, the PCS layer converts the first byte of PREAMBLE into the 2nd symbol in the idle ordered_set and replaces the second byte of the PREAMBLE with /S/. This mode preserves the link bandwidth at the cost of reducing the PREAMBLE length. A standard Ethernet link in which PREAMBLE is prepended to the frame should normally choose GMX(0..1)_TX*n*_SGMII_CTL[ALIGN] = 0, maintaining bandwidth. If the frames being sent over the link do not have a standard PREAMBLE, it is recommended to operate the link with GMX(0..1)_TX*n*_SGMII_CTL[ALIGN] = 1 so that meaningful frame data is not lost.

Cavium Confidential For Bowen Yang 02/27/20

When GMX(0..1)_TX*n*_SGMII_CTL[ALIGN] = 1, the SPD always replaces exactly one byte of the expanded and replicated packet data.
GMX(0..1)_TX*n*_SGMII_CTL[ALIGN] has no effect in 10 and 100 modes.
GMX(0..1)_TX*n*_SGMII_CTL[ALIGN] also applies to the autogenerated pause packets. Section 23.4.3.2 describes the autogenerated pause packets.

As Figure 23–11 shows, CN71XX terminates each packet with the EPD of either /T//R/ or /T//R//R/ code groups, whichever is needed so that the packet consumes an even number of code groups.

23.4.6.8 Transmit Collisions

CN71XX fully supports half-duplex CSMA/CD operation in 10/100/1000 mbs modes. In the event of a detected collision, CN71XX takes the following steps:

- CN71XX bursts the JAM pattern, backs off the bus, and retransmits the packet later.
- CN71XX carrier extends to meet the higher slottime of 1000Mbs operation.
- CN71XX attempts retransmission GMX(0..1)_TX_COL_ATTEMPT[LIMIT] times before giving up. If the retransmission limit is reached, it is considered an excessive collision (refer to XSCOL in Table 23–5). The packet is drained and the transmit machines resets the collision count for the next packet.

Late collisions (i.e. collisions that occur after the slot time is satisfied) are considered an error and the packet is not retransmitted. Refer to Section 23.6.2.

23.4.6.9 Transmit Bursts

CN71XX supports packet bursting on half-duplex links in 1000 Mbs mode. CN71XX continues sending packets (if packets are available to send) up until the burst limit (GMX(0..1)_TX*n*_BURST[BURST]). CN71XX fills IPG cycles with carrier extends. The first packet in a burst carrier-extends until the slot time is covered, as per the 802.3 specifications.

23.4.7 Unidirectional Transport

CN71XX implements the PCS modifications described in Clause 66.2 of the IEEE 802.3-2005 specification. When PCS(0..1)_MR*n*_CONTROL_REG[UNI] = 1, the autonegotiation process transmit flag always takes the value DATA and the autonegotiation process is never invoked.

23.5 Wake-on-LAN (WOL)

CN71XX supports power-savings features in which the network can wake the core from a power-savings state. This allows the core to be placed in a low-power state, but still allow remote and networked administration of the device. The CN71XX core can be awakened either by directed packets or by magic packets.

23.5.1 Packet Detection

AGX supports WOL via four types of packets:

- directed exact match
- directed multicast match
- directed broadcast match

Cavium Confidential For Bowen Yang ALU 02/27/2014

- magic packet match

When operating in wake-on-LAN mode, the configured MAC port unconditionally drops all received packets. The MAC will raise an interrupt to indicate that either a directed or magic packet has been detected and the system is requested to exit the low-power state.

23.5.1.1 Directed-Packet Match

When a MAC port is configured for direct-packet-match mode, all received packets are applied to the DMAC address filter as defined by dmac_addr_filter function as described in Section 23.4.4.4.

- When the function returns ACCEPT, an interrupt is asserted.
- No action is taken upon REJECT.

Care must be taken when GMX(0..1)_RX(0..3)_ADR_CTL[CAM_MODE] = 0, as the DMAC filter is operating in inversion mode. AGX has 32 DMAC registers (shared among the four ports) to use for exact address matches for either directed or directed multicast DMAC.

23.5.1.2 Magic-Packet Match

When a MAC port is configured for magic-packet-match mode, all received packets are parsed for the magic-data pattern. The pattern consists of six synchronization bytes of 0xFF followed by the GMX(0..1)_SMAC(0..3)[SMAC<47:0>] repeated 16 times.

The pattern can be located anywhere in the packet once the DMAC, SMAC, and TYPE are parsed. The pattern must not be interrupted for magic-packet detection to occur. An example magic packet might look like the following:

```
UDD_SKIP
DMAC
SMAC
TYPE
[optional bytes]
FF FF FF FF FF FF   // Synchronization
SMAC0
SMAC1
SMAC2
...
SMAC15
[optional bytes]
FCS
```

Magic packets are not required to have a good FCS to have a match.

Cavium Confidential For Bowen Yang ALU 02/27/2014

23.5.2 Operation

When software determines it wants to enter WOL mode, it sets GMX(0..1)_WOL_CTL[EN] = 1.

This places the MAC in WOL mode, which causes the MAC to drop incoming packets. Packets in that have been partially buffered in the RX FIFO when WOL mode is entered are terminated with the PARTIAL opcode error. All subsequent incoming packets will be completely dropped. Packets will continue to be dropped for the duration of the WOL event (i.e. while GMX(0..1)_WOL_CTL[EN] = 1).

Once GMX(0..1)_WOL_CTL[EN] is set, software can reconfigure the DMAC filtering, UDD_SKIP, and MAGIC packet control to achieve the desired packet matching. Systems can choose to use either directed-packet match, magic-packet match, or both. Mulitple MACs can be enabled for WOL matching simultaneously and have distinct matching modes and criteria.

Software can then begin WOL-event detection on a individual port by setting GMX(0..1)_WOL_CTL[DIRECT_EN<prt>/MAGIC_EN<prt>] = 1 .

When a WOL event occurs, it asserts the interrupt bit, GMX(0..1)_RX(0..3)_INT_REG[WOL] and notifies CIU if the corresponding enable bit is set (GMX(0..1)_RX(0/1..3)_INT_EN[WOL].

The MAC will continue to drop packets. The interrupt handler can put the MACs back into normal operation and deassert GMX(0..1)_WOL_CTL[EN]; packets will then begin flowing in.

The CAMs can be disabled in directed-packet mode by configuring the DCAM CAM properly.

GMX(0..1)_RX(0..3)_ADR_CTL[CAM_MODE] = 1
GMX(0..1)_RX(0..3)_ADR_CTL[MCST] = 1
GMX(0..1)_RX(0..3)_ADR_CTL[BCST] = 0
GMX(0..1)_RX(0..3)_ADR_CAM_ALL_EN[EN] = 0

23.6 Errors/Exceptions

CN71XX implements many error and exception checks on the SGMII/1000BASE-X/QSGMII interfaces. The following subsections describe the exception checkers and how the exception is communicated to software.

Many of these exceptions can cause interrupts:

- GMX(0..1)_RX*n*_INT_REG interrupts can optionally be enabled via corresponding bits in GMX(0..1)_RX*n*_INT_EN.
- PCS(0..1)_INT*n*_REG interrupts can optionally be enabled via corresponding bits in PCS(0..1)_INT*n*_EN_REG.
- GMX(0..1)_TX_INT_REG interrupts can optionally be enabled via corresponding bits in GMX(0..1)_TX_INT_EN.

Chapter 13 covers how these interrupts can be distributed system-wide.

Cavium Confidential For Bowen Yang ALU 02/27/2014

23.6.1 Receive Error/Exception Checks

Many of the exceptions can be enabled by setting the appropriate bit in GMX(0..1)_RX*n*_FRM_CHK. If the bit is set, the exception is logged in GMX(0..1)_RX*n*_INT_REG. In addition, packets sent into CN71XX arrive with a receive error (work-queue entry WORD2[RE] = 1) and the opcode (WORD2[Opcode]) is set to the exception in the work-queue entry. Chapter 9 and Section 9.5 describe the work queue entries that CN71XX creates for arriving packets.

PCS(0..1)_INT*n*_REG contains a number of additional error interrupts, some of which should never assert in normal operation.

Table 23–4 lists the receive exceptions, their causes, and how they are handled.

Table 23–4 Receive Errors/Exceptions

Exception	Cause	Notification
CAREXT	A packet was received with one or more carrier extend errors.	GMX(0..1)_RX*n*_FRM_CHK[CAREXT] enables the check. If enabled, GMX(0..1)_RX*n*_INT_REG[CAREXT] is set to 1 and work-queue entry WORD2[OPCODE] is set to 0x9.
JABBER	A packet was received with length > GMX(0..1)_RX*n*_JABBER bytes.	The packet is unconditionally truncated at GMX(0..1)_RX*n*_JABBER bytes. GMX(0..1)_RX*n*_FRM_CHK[JABBER] enables the check. If enabled, GMX(0..1)_RX*n*_INT_REG[JABBER] is set to 1 and work-queue entry WORD2[OPCODE] is set to 0x2.
FCSERR	A packet was received that failed the FCS/CRC check. This error is likely to occur when a RCVERR occurs.	GMX(0..1)_RX*n*_FRM_CHK[FCSERR] enables the check. If enabled, GMX(0..1)_RX*n*_INT_REG[FCSERR] is set to 1 and work-queue entry WORD2[OPCODE] is set to 0x3, 0x6, or 0x7 depending on the length and alignment checks.
MINERR	A packet was received with length < GMX(0..1)_RX*n*_FRM_MIN bytes.	GMX(0..1)_RX*n*_FRM_CHK[MINERR] enables the check. If enabled, GMX(0..1)_RX*n*_INT_REG[MINERR] is set to 1 and work-queue entry WORD2[OPCODE] is set to 0x6 (if packet has bad FCS) or 0x8 (if packet has good FCS).
RCVERR	A packet was received with one more cycles of data-reception error indicated on the internal GMII bus (RX_ER). This can result from loss of synchronization on the SerDes interface or a bad received code group while receiving a packet (refer to RXERR and RXLOCK below in this table), or if the remote transmitter sends a error propagation ordered set.	GMX(0..1)_RX*n*_FRM_CHK[RCVERR] enables the check. If enabled, GMX(0..1)_RX*n*_INT_REG[RCVERR] is set to 1 and work-queue entry WORD2[OPCODE] is set to 0xB.
SKPERR	A packet was received in which the length < GMX(0..1)_RX*n*_UDD_SKP[LEN] bytes. This means that there was not enough packet data in order to get past the Skip field and into the L2 HDR.	GMX(0..1)_RX*n*_FRM_CHK[SKPERR] enables the check. If enabled, GMX(0..1)_RX*n*_INT_REG[SKPERR] is set to 1 and work-queue entry WORD2[OPCODE] is set to 0xC.
OVRERR	Indicates that SGMII/1000BASE-X/QSGMII data arrived too quickly. This can only occur when SGMII/1000BASE-X/QSGMII and the CN71XX coprocessor-clock frequencies are mismatched. coprocessor-clock rate ≤ 0.1 × (baud rate) This should never happen during normal operation.	This check is always enabled and sets GMX(0..1)_RX*n*_INT_REG[OVRERR] to 1.
PCTERR	A packet with a bad preamble was received. Refer to Section 23.4.4.1. CN71XX drops the packet.	GMX(0..1)_RX*n*_FRM_CTL[PRE_CHK] enables the preamble checker. If enabled, GMX(0..1)_RX*n*_INT_REG[PCTERR] is set to 1. The packet is dropped.

Cavium Confidential
Bowen Zhao
ALU
05/07/2014

Table 23–4 Receive Errors/Exceptions (Continued)

Exception	Cause	Notification
RSVERR	CN71XX detected a reserved opcode on the internal GMII interface. This error should never occur.	This check is always enabled and sets GMX(0..1)_RX*n*_INT_REG[RSVERR] to 1.
FALERR	CN71XX detected a false carrier on the SGMII/1000BASE-X/QSGMII interface.	This check is always enabled and sets GMX(0..1)_RX*n*_INT_REG[FALERR] to 1.
COLDET	In half-duplex mode, CN71XX detected a collision. This can and will occur under normal operation in half-duplex mode.	This check is always enabled and sets GMX(0..1)_RX*n*_INT_REG[COLDET] to 1.
IFGERR	CN71XX detected an interframe gap smaller than GMX(0..1)_RX*n*_IFG[IFG]. This exception does not necessarily indicate a system failure.	This check is always enabled and sets GMX(0..1)_RX*n*_INT_REG[IFGERR] to 1.
PAUSE_DRP	A pause packet was dropped due to full GMX RX FIFO.	This check is always enabled and sets GMX(0..1)_RX*n*_INT_REG[PAUSE_DRP] to 1. It also loads GMX(0..1)_RX*n*_PAUSE_DROP_TIME[STATUS] with pause_time of the pause packet.
RXERR	CN71XX received a bad (10-bit) code group. A bad code group can lead to other errors. The CN71XX GMX logic may also generate an RXERR when PCS is disabled in the middle of a received packet. Refer to Section 23.7.1.	This check is always enabled and sets PCS(0..1)_INT*n*_REG[RXERR] to 1.
RXLOCK	CN71XX receive logic lost code group or bit lock synchronization (i.e. one of PCS(0..1)_RX*n*_SYNC_REG[SYNC,BIT_LOCK] asserts). Loss of synchronization can lead to many other errors.	This check is always enabled and sets PCS(0..1)_INT*n*_REG[RXLOCK] to 1.
SYNC_BAD RXBAD	Internal errors that should never occur during normal operation.	These checks are always enabled and set corresponding PCS(0..1)_INT*n*_REG[SYNC_BAD,RXBAD] bits to 1.

Cavium Confidential For Bowen Yang ALU 02/27/2014

23.6.2 Transmit Error/Exception Checks

Many of the transmit errors are logged in GMX(0..1)_TX_INT_REG. PCS(0..1)_INT*n*_REG contains a number of additional error interrupts, some of which should never assert in normal operation.

Additionally, GMX(0..1)_BAD_REG contains a number of error conditions that should not happen during normal operation.

Table 23–5 lists the exceptions, their causes, and how they are handled.

Table 23–5 Transmit Errors/Exceptions

Exception	Cause	Notification
PKO_NXA	The SGMII/1000BASE-X/QSGMII transmitter received a request to send data out a port not enabled by GMX(0..1)_TX_PRTS[PRTS].	CN71XX asserts GMX(0..1)_TX*n*_INT_REG[PKO_NXA*n*] and logs the offending port in GMX(0..1)_NXA_ADR.
UNDFLW	In the unlikely event that PKO cannot keep the GMX TX FIFO full, the SGMII/1000BASE-X/ QSGMII packet transfer will underflow. This should be detected by the receiving device as an FCS error. Internally, the packet is drained and lost.	CN71XX asserts GMX(0..1)_TX*n*_INT_REG[UNDFLW*n*] and drains the packet.
LATE_COL	A late collision occurs in half-duplex mode if GMX detects a collision after the slot time for the packet. GMX discards the packet and receivers are likely to encounter an FCS error.	CN71XX asserts GMX(0..1)_TX_INT_REG[LATE_COL*n*] and drains the packet.
XSCOL	An excessive collision occurs in half-duplex mode if CN71XX detects a collision during each of GMX(0..1)_TX*n*_COL_ATTEMPT[LIMIT] attempts to send a packet.	CN71XX asserts GMX(0..1)_TX*n*_INT_REG[XSCOL*n*] and the packet is drained and lost.
XSDEF	An excessive deferral occurs in half-duplex mode if CN71XX detects that a packet cannot send on the transmit interface for maxDeferTime (as defined by IEEE 802.3-2002 specification) time. (It is approximately the time for 3KB and 19KB packets in 10/100 and 1000 modes, respectively.)	CN71XX asserts GMX(0..1)_TX*n*_INT_REG[XSDEF*n*] and continues to attempt to transmit the packet.
OUT_OVR LOSTSTAT STATOVR INB_NXA	The GMX transmitter has several internal error checks. These checks never assert in a correctly configured system. Any assertion indicates that an internal problem has occurred.	CN71XX asserts corresponding bit in GMX(0..1)_BAD_REG.
TXBAD TXFIFO TXFIFU	The PCS transmitter has several internal error checks. These checks never assert in a correctly configured system. Any assertion indicates that an internal problem has occurred.	CN71XX asserts corresponding bit in PCS(0..1)_INT*n*_REG.

23.6.3 Transmit Error Propagation

If PKO detects an error when reading packet data, the error is passed to the SGMII/1000BASE-X/QSGMII transmitter. If GMX(0..1)_TX_CORRUPT[CORRUPTn] is set, CN71XX corrupts the FCS of the packet in order to notify the receiving device that the packet is corrupted.

Software can detect that PKO detected an error through the PKO_REG_ERROR[PARITY] bit.

Cavium Confidential For Bowen Yang ALU 02/27/2014

23.6.4 PCS Error/Exception Checks

Table 23–6 covers all of the errors that are not directly related to either receive or transmit. All of these errors come from the PCS block, and all PCS errors are logged in PCS(0..1)_INT*n*_REG.

Table 23–6 PCS Errors/Exceptions

Exception	Cause	Notification
AN_ERR	Autonegotiation completed and the autonegotiation resolution function could not find a common operating mode. This error can only happen in 1000BASE-X mode	CN71XX sets PCS(0..1)_INT*n*_REG[AN_ERR] after completing the autonegotiation.
REM_FLT	CN71XX received a non-zero remote fault encoding from the link partner as part of autonegotiation.	CN71XX sets PCS(0..1)_MR*n*_STATUS_REG[RM_FLT] to 1. PCS(0..1)_AN*n*_LP_ABIL_REG[REM_FLT] contains the particular code in 1000BASE-X mode. PCS(0..1)_SGM*n*_LP_ADV_REG[LINK] will be 0 in SGMII mode.
XMIT	The xmit variable changes. This is due to CN71XX or the link partner initiating autonegotiation. The data received or transmitted on the link is discarded as autonegotiation takes over the link any time that the xmit variable is not data. Clauses 36 and 37 in the 802.3-2005 specification describe the xmit variable. (Figure 37-6, in particular, describes xmit and when it changes value.) Note that PCS(0..1)_INT*n*_REG[XMIT] will set during any autonegotiation and after every reset, except when PCS(0..1)_MR*n*_CONTROL_REG[UNI] is set. When autonegotiation is disabled and PCS(0..1)_MR*n*_CONTROL_REG[UNI] = 0, xmit variable changes are an indication of lack of code group synchronization by the receive logic.	CN71XX sets PCS(0..1)_INT*n*_REG[XMIT] to 1.
LNKSPD DUP	The operational link speed or duplex mode changes. These bits can assert in the following situations: • When PCS(0..1)_MR*n*_CONTROL_REG[DUP, SPDLSB,MSB] is written producing a different mode when autonegotiation is disabled or the autonegotiation result is being over-ridden. • When autonegotiation completes, is enabled, is not being over-ridden, and produces a different result. • When autonegotiation is disabled.	CN71XX sets PCS(0..1)_INT*n*_REG[LNKSPD] or PCS(0..1)_INT*n*_REG[DUP].
AN_BAD	The hardware autonegotiation state machine reached a bad state. This should never happen during normal operation.	CN71XX sets PCS(0..1)_INT*n*_REG[AN_BAD] to 1.

Cavium Confidential For Bowen Yang ALU 02/27/2014

23.7 Reset, Autonegotiation, Initialization, and Link Status Changes

23.7.1 GMX/PCS Reset and Enables

In addition to chip reset, the PCS state machines are in reset:

- Whenever PCS(0..1)_MR*n*_CONTROL_REG[PWR_DN] = 1, or
- For a brief amount of time after software writes PCS(0..1)_MR*n*_CONTROL_REG[RESET] = 1. (Software should read PCS(0..1)_MR*n*_CONTROL_REG[RESET] until it is 0 to determine when the reset sequence is complete.)

GMX cannot receive any new input packet data on a port nor transfer packet data out a port when PCS is in reset. If GMX is in the middle of receiving an inbound packet when PCS enters reset, then GMX will force an RXERR to close the packet. Section 23.6.1 describes the RXERR error.

The GMX state machines are disabled when GMX(0..1)_PRT*n*_CFG[EN] = 0. When a GMX port is disabled by clearing GMX(0..1)_PRT*n*_CFG[EN] = 0, CN71XX will stop sending new packets out the port. Subsequent outbound packets will remain enqueued in the transmit FIFO for the port when a GMX port is disabled this way. (These enqueued packets will be sent out later if GMX is later re-enabled.) If CN71XX is in the middle of sending out a packet when GMX(0..1)_PRT*n*_CFG[EN] is cleared to 0, that packet will be completed in entirety.

When a GMX port is disabled by clearing GMX(0..1)_PRT*n*_CFG[EN] = 0, the GMX receive path stops putting packets into the receive FIFO. If CN71XX is in the middle of receiving a packet when GMX(0..1)_PRT*n*_CFG[EN] is cleared to 0, the packet will be received in entirety (assuming it is not dropped). New packets are always ignored when GMX(0..1)_PRT*n*_CFG[EN] = 0.

When PCS(0..1)_MISC*n*_CTL_REG[GMXENO] = 1, the effect is the same on the receive path as when GMX(0..1)_PRT*n*_CFG[EN] = 0: the GMX receive path also stops putting received packets into the receive FIFO. If CN71XX is in the middle of receiving a packet when PCS(0..1)_MISC*n*_CTL_REG[GMXENO] is set to 1, the packet will be received in entirety (assuming it is not dropped). New packets are always ignored after PCS(0..1)_MISC*n*_CTL_REG[GMXENO] = 1. Note that PCS(0..1)_MISC*n*_CTL_REG[GMXENO] has no effect on transmitted packets. It is recommended for PCS(0..1)_MISC*n*_CTL_REG[GMXENO] to be set as PCS logic exits reset, up until the point when the link status appears good (i.e. PCS(0..1)_MR*n*_STATUS_REG[LNK_ST] = 1). Section 23.7.3 describes more about link bring-up.

Table 23–7 describes the effect of various enables on CN71XX behavior.

Table 23–7 Description of Different Enable Modes

GMX(0..1)_PRT*n*_CFG [EN]	PCS(0..1)_MISC*n*_CTL_REG [GMXENO]	Description
0	X	CN71XX will not put new packets into the receive FIFO and will not transmit any outbound packets.
1	0	Normal operating mode. CN71XX puts received packets into the receive FIFO and transmits outbound packets that PKO puts in the transmit FIFO or are autogenerated.
1	1	CN71XX will not put new packets into the receive FIFO, but will transmit outbound packets that PKO puts in the transmit FIFO or are autogenerated. Carrier-sense and collision are ignored in half-duplex mode. As much as possible, this will drain the GMX FIFOs.

Cavium Confidential For Edwin Kang 02-27-2014

23.7.2 Autonegotiation Execution

Section 23.3.1 describes PCS autonegotiation.

Software should do the following to execute this autonegotiation when autonegotiation is desired:

1. Enable the interface. Set GMX(0..1)_INF_MODE[EN] = 1 if not already set.
2. Disable GMX. Set PCS(0..1)_MISC*n*_CTL_REG[GMXENO] = 1. Clear GMX(0..1)_PRT*n*_CFG[EN] = 0 if GMX should not be drained. Refer to Section 23.7.1 above.
3. Write PCS(0..1)_MISC*n*_CTL_REG[MODE,AN_OVRD,MAC_PHY,SAMP_PT] to the selected values. If the final speed (after autonegotiation) is known to be 10Mbit, PCS(0..1)_MISC*n*_CTL_REG[SAMP_PT] should be written to the value appropriate for 10Mbit (e.g. 50). Otherwise, PCS(0..1)_MISC*n*_CTL_REG[SAMP_PT] should be written to the value appropriate for 100Mbit (e.g., 5).
4. If desired, set the polarity of the SerDes link. Refer to Section 23.4.2 and PCS(0..1)_TX_RX*n*_POLARITY_REG.
5. Write PCS(0..1)_LINK*n*_TIMER_COUNT_REG[COUNT] with the appropriate value.
   - 1000BASE-X specifies a 10ms interval
   - SGMII specifies a 1.6ms interval
6. Write the advertisement register to be used as the tx_Config_Reg<D15:D0> of the autonegotiation.
   - In 1000BASE-X mode, tx_Config_Reg<D15:D0> is PCS(0..1)_AN*n*_ADV_REG.
   - In SGMII PHY mode, tx_Config_Reg<D15:D0> is PCS(0..1)_SGM*n*_AN_ADV_REG
   - In SGMII MAC mode, tx_Config_Reg<D15:D0> is the fixed value 0x4001, so this step can be skipped.
7. Preload PCS(0..1)_MR*n*_CONTROL_REG
   - PCS(0..1)_MR*n*_CONTROL_REG[DUP,SPDLSB,SPDMSB] should be written to appropriate values if PCS(0..1)_MISC*n*_CTL_REG[AN_OVRD] = 1.
   - Set PCS(0..1)_MR*n*_CONTROL_REG[AN_EN] = 1 to enable autonegotiation.
8. Take PCS through a reset sequence. PCS(0..1)_MR*n*_CONTROL_REG[PWR_DN] should be cleared to 0. Write PCS(0..1)_MR*n*_CONTROL_REG[RESET] = 1 (while not changing the value of the other PCS(0..1)_MR*n*_CONTROL_REG bits). Read PCS(0..1)_MR*n*_CONTROL_REG[RESET] until it changes value to 0.
9. Write PCS(0..1)_MR*n*_CONTROL_REG[RST_AN] = 1 to ensure a fresh autonegotiation starts.
10. Wait for PCS(0..1)_MR*n*_STATUS_REG[AN_CPT] to be set, indicating that autonegotiation is complete.
11. The link-partner's advertisement can be observed at this point.
    - In 1000BASE-X mode, rx_Config_Reg<D15:D0> is PCS(0..1)_AN*n*_LP_ABIL_REG.
    - In SGMII PHY mode, rx_Config_Reg<D15:D0> should be the constant 0x4001, and need not be observed.

Cavium Confidential For Boren Yang 2018/10/31

- In SGMII MAC mode, rx_Config_Reg<D15:D0> is PCS(0..1)_SGM*n*_LP_ADV_REG.
- In QSGMII mode, TBD.

12. The final mode can be determined. Section 23.3.2 describes how to do this. If the final selected speed is 10Mbit, PCS(0..1)_MISC*n*_CTL_REG[SAMP_PT] may be changed to the value most appropriate for 10Mbit operation at this point.

23.7.3 Link Bring-Up

This sequence should be used to bring up a CN71XX SGMII/1000BASE-X/QSGMII SerDes interface:

1. Set PCS(0..1)_MR*n*_CONTROL_REG[PWR_DN] = 1 (if not already set) to hold PCS in reset.

2. Enable the interface. Set GMX(0..1)_INF_MODE[EN] = 1 if not already set.

3. Disable GMX. Set PCS(0..1)_MISC*n*_CTL_REG[GMXENO] = 1 to disable reception. Clear GMX(0..1)_PRT*n*_CFG[EN] = 0 if GMX should not be drained. Refer to Section 23.7.1.

4. Next, the PCS should be brought up:

4a. If autonegotiation is desired, it should be executed as Section 23.7.2 describes.

4b. If autonegotiation is not desired, program the following sequence:

- Set PCS(0..1)_MISC*n*_CTL_REG[MODE,MAC_PHY,SAMP_PT] to appropriate values.
- If desired, set the polarity of the SerDes link. Refer to Section 23.4.2 and PCS(0..1)_TX_RX*n*_POLARITY_REG.
- Set PCS(0..1)_LINK*n*_TIMER_COUNT_REG[COUNT] to the appropriate value.
  - 1000BASE-X specifies a 10ms interval
  - SGMII specifies a 1.6ms interval
  - QSGMII specifies a TBD interval
- Set PCS(0..1)_MR*n*_CONTROL_REG[DUP,SPDMSB,SPDLSB] to appropriate values.
- Clear PCS(0..1)_MR*n*_CONTROL_REG[LOOPBACK,PWR_DN] = 0.
- Take PCS through a reset sequence. Write PCS(0..1)_MR*n*_CONTROL_REG[RESET] = 1 (while not changing the value of the other PCS(0..1)_MR*n*_CONTROL_REG bits). Read PCS(0..1)_MR*n*_CONTROL_REG[RESET] until it changes value to 0.

5. Check to make sure that the link appears up and stable. The link appears up and stable when PCS(0..1)_MR*n*_STATUS_REG[LNK_ST] is set. Section 23.3.2 discusses more about PCS(0..1)_MR*n*_STATUS_REG[LNK_ST].

6. Configure GMX.

6a. If not done already, clear GMX(0..1)_PRT*n*_CFG[EN] = 0.

6b. Wait for GMX(0..1)_PRT*n*_CFG[RX_IDLE,TX_IDLE] to both become set, indicating that GMX is idle.

At this point, GMX will neither be sending nor receiving packets.

6c. Program all relevant GMX configuration while GMX(0..1)_PRT*n*_CFG[EN] = 0. This includes all things described in this chapter. Most importantly, Section 23.3.3 contains the different parameters needed for the different link

Cavium Confidential For Boyan Yang ALU 02/27/2014

modes. Section 23.3.2 describes how to determine the PCS mode. The GMX mode should always be set to match the PCS mode.

6d. Set GMX(0..1)_PRT*n*_CFG[EN] = 1 to enable GMX. At this point, GMX can transmit packets, but cannot receive packets since PCS(0..1)_MISC*n*_CTL_REG[GMXENO] = 1.

7. Enable packet receive by clearing PCS(0..1)_MISC*n*_CTL_REG[GMXENO] = 0. This step should only be executed after PIP/IPD have been configured, as packets should not arrive at PIP/IPD until they are fully configured. Refer to Chapter 9.

23.7.4 Link Bring-Down

When bringing the link down, software should immediately set PCS(0..1)_MISC*n*_CTL_REG[GMXENO] = 1 to prevent extraneous packets from arriving. Clear GMX(0..1)_PRT*n*_CFG[EN] = 0 if GMX should not be drained. Section 23.7.1 above describes the affect of these enables on behavior.

If GMX is being drained (i.e. if GMX(0..1)_PRT*n*_CFG[EN] remains asserted), it may also be appropriate to forcibly disable pause frame generation by CN71XX by setting GMX(0..1)_TX_OVR_BP[EN<prt>] = 1, and clearing GMX(0..1)_TX_OVR_BP[BP<prt>] = 0.

It may also be appropriate to reset the PCS logic. Section 23.7.1 also describes how to do this.

23.7.5 Link Status and Changes

The link configuration/status includes the CN71XX PCS mode, the link partner's PCS mode, and the GMX mode.

The CN71XX PCS mode should always match the PCS mode of the link partner, and the GMX mode should also match the PCS mode. Section 23.3.2 describes how to determine the local PCS mode. Section 23.3.3 describes some key parameters of the GMX mode related to the PCS mode.

Out-of-band means may be used to achieve a consistent link status. It is possible that the link status will be known beforehand. If the link status is not known beforehand, the mode of the link partner could be determined through these and other ways:

- Software can poll the appropriate registers in the link partner looking for a status change. This might be done using the CN71XX SMI/MDIO interface. Chapter 32 describes the CN71XX SMI/MDIO interface.
- If the link-partner supports link-status-change interrupts, the board can route the interrupt to CN71XX.

When autonegotiation is enabled, autonegotiation can ensure that the PCS modes are consistent. In that case, the following in-band notifications are useful indications of PCS mode changes:

- Software can receive an interrupt every time when the xmit variable changes value. Section 23.6.4 describes the XMIT exception. Xmit variable changes cause major disruptions, and can occur whenever PCS(0..1)_MR*n*_CONTROL_REG[UNI] = 0, both when autonegotiation enabled or disabled. When autonegotiation is enabled, xmit variable changes may be initiated locally or remotely. When autonegotiation is disabled, xmit variable changes are only an indication of lack of code group synchronization by the PCS receive logic.

Cavium Confidential For Bowen Zhang ADU 02-27-2014

- If autonegotiation is enabled, Software may use the LNKSPD,DUP interrupts described in Section 23.6.4. CN71XX asserts the PCS(0..1)_INT*n*_REG[LNKSPD,LNKDUP] interrupt status bits anytime the operational mode changes.

Chapter 13 indicates how these interrupts can be distributed on CN71XX.

Any PCS mode change may imply that the current GMX mode does not match the current PCS mode, and that should be immediately corrected. The link bring-down sequence described in Section 23.7.4 should be used. Following that, the link bring-up sequence in Section 23.7.2 can be used to bring the link back up in the new mode.

23.8 Statistics

CN71XX supports a rich statistics gathering, described in Table 23–8.

Table 23–8 CN71XX Statistics Gathering

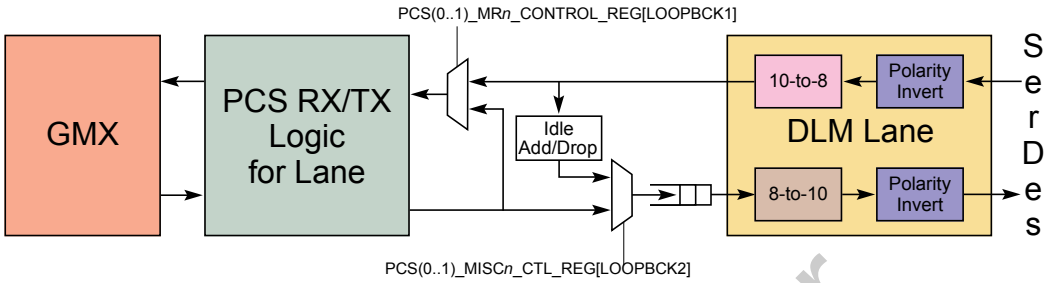
CSR Field	Statistic
RX	
GMX(0..1)_RX*n*_STATS_PKTS[CNT]	32-bit count of all good packets
GMX(0..1)_RX*n*_STATS_OCTS[CNT]	48-bit count of all bytes from good packets
GMX(0..1)_RX*n*_STATS_PKTS_CTL[CNT]	32-bit count of all control/pause packets
GMX(0..1)_RX*n*_STATS_OCTS_CTL[CNT]	48-bit count of all bytes from control/pause packets
GMX(0..1)_RX*n*_STATS_PKTS_DMAC[CNT]	32-bit count of all DMAC filtered packets
GMX(0..1)_RX*n*_STATS_OCTS_DMAC[CNT	48-bit count of all bytes from DMAC filtered packets
GMX(0..1)_RX*n*_STATS_PKTS_DRP[CNT]	32-bit count of all packets dropped due to RX FIFO full
GMX(0..1)_RX*n*_STATS_OCTS_DRP[CNT]	48-bit count of all bytes from packets dropped due to RX FIFO full
GMX(0..1)_RX*n*_STATS_PKTS_BAD[CNT]	32-bit count of all bad packets
TX	
GMX(0..1)_TX*n*_STAT0[XSCOL]	32-bit count of packets dropped due to excessive collisions
GMX(0..1)_TX*n*_STAT0[XSDEF]	32-bit count of packets dropped due to excessive deferral
GMX(0..1)_TX*n*_STAT1[MCOL]	32-bit count of packets sent that experienced multiple collisions before successful transmission
GMX(0..1)_TX*n*_STAT1[SCOL]	32-bit count of packets sent that experienced a single collision before successful transmission
GMX(0..1)_TX*n*_STAT2[OCTS]	48-bit count of all bytes sent
GMX(0..1)_TX*n*_STAT3[PKTS]	32-bit count of all packets sent
GMX(0..1)_TX*n*_STAT4[HIST0]	32-bit count of packets sent with an octet count < 64
GMX(0..1)_TX*n*_STAT4[HIST1]	32-bit count of packets sent with an octet count == 64
GMX(0..1)_TX*n*_STAT5[HIST2]	32-bit count of packets sent with an octet count of 65–127
GMX(0..1)_TX*n*_STAT5[HIST3]	32-bit count of packets sent with an octet count of 128–255
GMX(0..1)_TX*n*_STAT6[HIST4]	32-bit count of packets sent with an octet count of 256–511
GMX(0..1)_TX*n*_STAT6[HIST5]	32-bit count of packets sent with an octet count of 512–1023
GMX(0..1)_TX*n*_STAT7[HIST6]	32-bit count of packets sent with an octet count of 1024–1518
GMX(0..1)_TX*n*_STAT7[HIST7]	32-bit count of packets sent with an octet count of > 1518
GMX(0..1)_TX*n*_STAT8[BCST]	32-bit count of packets sent to a multicast DMAC
GMX(0..1)_TX*n*_STAT8[MCST]	32-bit count of packets sent to the broadcast DMAC
GMX(0..1)_TX*n*_STAT9[CTL]	32-bit count of control/pause packets sent
GMX(0..1)_TX*n*_STAT9[UNDFLW]	32-bit count of packets sent that experienced a transmit underflow and were truncated

Cavium Confidential For Bowen Yang 04/17/2014

23.9 Loopback

CN71XX PCS logic supports both internal- and external-loopback modes. These modes cannot be used simultaneously. Figure 23–12 shows the loopback architecture.

Figure 23–12 PCS Loopback



When PCS(0..1)_MR*n*_CONTROL_REG[LOOPBCK1] = 1 and GMX is fully enabled (i.e. GMX(0..1)_PRT*n*_CFG[EN] = 1 and PCS(0..1)_MISC*n*_CTL_REG[GMXENO] = 0), the PCS logic implements internal loopback. In this mode, packets that are sent out the GMX transmit FIFO arrive at the GMX receive FIFO. As Figure 23–12 indicates, the internally-looped packet data traverses all PCS transmit and receive state machines. The packet data is received at the same rate as transmitted. Note that PCS internal loopback is not functional when the DLM is not being clocked, as may be the case if the DLM is not used.

When PCS(0..1)_MISC*n*_CTL_REG[LOOPBCK2] = 1, the PCS logic loops all packets received by the SerDes port/lane out the corresponding SerDes output port/lane. As Figure 23–12 indicates, the external loopback may not transmit the identical code group stream as was received because preserving idle ordered_sets (i.e. /I2/s) may be added or dropped. This is necessary, as the transmit clock may differ slightly from the receive clock. The CN71XX external loopback process is simple. The DLM receive logic obtains bit lock (indicated by PCS(0..1)_RX*n*_SYNC_REG[BIT_LOCK]), and performs polarity inversion and 10-to-8 decode. The PCS idle add/drop state machine pattern-matches for /I2/'s, and either duplicates or removes each /I2/ based on high- and low-water marks on the output FIFO displayed in Figure 23–12. The DLM transmit logic converts 8-to-10 (recalculating disparity) and inverts polarity to complete the external loopback.

A PCS reset sequence should always be used whenever the operating mode of the PCS logic changes, including either of the following loopback modes. Section 23.7.1 describes how to reset the PCS logic.

- In the internal-loopback case, set PCS(0..1)_MR*n*_CONTROL_REG[LOOPBCK1] = 1 coincident with setting PCS(0..1)_MR*n*_CONTROL_REG[RESET] = 1 to reset PCS.
- In the external-loopback case, set PCS(0..1)_MISC*n*_CTL_REG[LOOPBCK2] = 1 first, followed by a PCS reset. PCS behavior may be unpredictable between setting LOOPBCK2 to 1 and the reset. After reset, external-loopback transmit may be garbled until bit lock is achieved, the output FIFO displayed in Figure 23–12 fills, and a legal sequence of idles is received.

The autocalculated SerDes polarity should not be used when any loopback option is enabled. Refer to Section 23.4.2.

Cavium Confidential For ACD Bowen Yang 03/27/2015

23.10 Enumerations

GMX Interface Mode
GMX_INF_MODE_E

Enumerates the values for GMX(0..1)_INF_MODE[MODE].

Value	Mnemonic	Description
0x0	DISABLED	The interface is disabled. When GMX0 is configured for RXAUI mode, GMX1 must be disabled.
0x1	SGMII	The interface is in SGMII mode.
0x2	QSGMII	The interface is in QSGMII mode.
0x3	RXAUI	The interface is in RXAUI mode. When GMX0 is configured for RXAUI mode, GMX1 must be disabled. This mode is not a valid operating mode for GMX1.

23.11 GMX Registers

The GMX registers are listed in Table 23–9.

Table 23–9 GMX Registers

Register	Address[1]	CSR Type[2]	Detailed Description
GMX(0..1)_RX(0..3)_INT_REG	0x1180008000000 + α*0x8000000 + β*0x800 + α*0x1000000 + β*0x8	RSL	See page 1050
GMX(0..1)_RX(0/1..3)_INT_EN	0x1180008000008 + α*0x8000000 + β*0x800	RSL	See page 1051
GMX(0..1)_PRT(0..3)_CFG	0x1180008000010 + α*0x8000000 + β*0x800	RSL	See page 1052
GMX(0..1)_RX(0..3)_FRM_CTL	0x1180008000018 + α*0x8000000 + β*0x800	RSL	See page 1053
GMX(0..1)_RX(0/1..3)_FRM_CHK	0x1180008000020 + α*0x8000000 + β*0x800	RSL	See page 1054
GMX(0..1)_RX(0..3)_JABBER	0x1180008000038 + α*0x8000000 + β*0x800	RSL	See page 1055
GMX(0..1)_RX(0..3)_DECISION	0x1180008000040 + α*0x8000000 + β*0x800	RSL	See page 1055
GMX(0..1)_RX(0..3)_UDD_SKP	0x1180008000048 + α*0x8000000 + β*0x800	RSL	See page 1056
GMX(0..1)_RX(0..3)_STATS_CTL	0x1180008000050 + α*0x8000000 + β*0x800	RSL	See page 1056
GMX(0..1)_RX(0..3)_IFG	0x1180008000058 + α*0x8000000 + β*0x800	RSL	See page 1057
GMX(0..1)_RX(0..3)_PAUSE_DROP_TIME	0x1180008000068 + α*0x8000000 + β*0x800	RSL	See page 1057
GMX(0..1)_RX(0..3)_STATS_PKTS	0x1180008000080 + α*0x8000000 + β*0x800	RSL	See page 1057
GMX(0..1)_RX(0..3)_STATS_OCTS	0x1180008000088 + α*0x8000000 + β*0x800	RSL	See page 1058
GMX(0..1)_RX(0..3)_STATS_PKTS_CTL	0x1180008000090 + α*0x8000000 + β*0x800	RSL	See page 1058
GMX(0..1)_RX(0..3)_STATS_OCTS_CTL	0x1180008000098 + α*0x8000000 + β*0x800	RSL	See page 1058
GMX(0..1)_RX(0..3)_STATS_PKTS_DMAC	0x11800080000A0 + α*0x8000000 + β*0x800	RSL	See page 1059
GMX(0..1)_RX(0..3)_STATS_OCTS_DMAC	0x11800080000A8 + α*0x8000000 + β*0x800	RSL	See page 1059
GMX(0..1)_RX(0..3)_STATS_PKTS_DRP	0x11800080000B0 + α*0x8000000 + β*0x800	RSL	See page 1059
GMX(0..1)_RX(0..3)_STATS_OCTS_DRP	0x11800080000B8 + α*0x8000000 + β*0x800	RSL	See page 1060
GMX(0..1)_RX(0..3)_STATS_PKTS_BAD	0x11800080000C0 + α*0x8000000 + β*0x800	RSL	See page 1060
GMX(0..1)_RX(0..3)_ADR_CTL	0x1180008000100 + α*0x8000000 + β*0x800	RSL	See page 1061
GMX(0..1)_RX(0..3)_ADR_CAM_EN	0x1180008000108 + α*0x8000000 + β*0x800	RSL	See page 1062
GMX(0..1)_RX(0..3)_ADR_CAM_ALL_EN	0x1180008000110 + α*0x8000000 + β*0x800	RSL	See page 1063
GMX(0..1)_RX(0..3)_ADR_CAM0	0x1180008000180 + α*0x8000000 + β*0x800	RSL	See page 1063

Cavium Confidential For Boyen Long 02/7/2014

Table 23–9 GMX Registers (Continued)

Register	Address[1]	CSR Type[2]	Detailed Description
GMX(0..1)_RX(0..3)_ADR_CAM1	0x1180008000188 + α*0x8000000 + β*0x800	RSL	See page 1063
GMX(0..1)_RX(0..3)_ADR_CAM2	0x1180008000190 + α*0x8000000 + β*0x800	RSL	See page 1063
GMX(0..1)_RX(0..3)_ADR_CAM3	0x1180008000198 + α*0x8000000 + β*0x800	RSL	See page 1063
GMX(0..1)_RX(0..3)_ADR_CAM4	0x11800080001A0 + α*0x8000000 + β*0x800	RSL	See page 1063
GMX(0..1)_RX(0..3)_ADR_CAM5	0x11800080001A8 + α*0x8000000 + β*0x800	RSL	See page 1063
GMX(0..1)_TX(0..3)_THRESH	0x1180008000210 + α*0x8000000 + β*0x800	RSL	See page 1065
GMX(0..1)_TX(0..3)_APPEND	0x1180008000218 + α*0x8000000 + β*0x800	RSL	See page 1066
GMX(0..1)_TX(0..3)_SLOT	0x1180008000220 + α*0x8000000 + β*0x800	RSL	See page 1066
GMX(0..1)_TX(0..3)_BURST	0x1180008000228 + α*0x8000000 + β*0x800	RSL	See page 1066
GMX(0..1)_SMAC(0..3)	0x1180008000230 + α*0x8000000 + β*0x800	RSL	See page 1066
GMX(0..1)_TX(0..3)_PAUSE_PKT_TIME	0x1180008000238 + α*0x8000000 + β*0x800	RSL	See page 1067
GMX(0..1)_TX(0..3)_MIN_PKT	0x1180008000240 + α*0x8000000 + β*0x800	RSL	See page 1067
GMX(0..1)_TX(0..3)_PAUSE_PKT_INTERVAL	0x1180008000248 + α*0x8000000 + β*0x800	RSL	See page 1068
GMX(0..1)_TX(0..3)_SOFT_PAUSE	0x1180008000250 + α*0x8000000 + β*0x800	RSL	See page 1068
GMX(0..1)_TX(0..3)_PAUSE_TOGO	0x1180008000258 + α*0x8000000 + β*0x800	RSL	See page 1069
GMX(0..1)_TX(0..3)_PAUSE_ZERO	0x1180008000260 + α*0x8000000 + β*0x800	RSL	See page 1069
GMX(0..1)_TX(0..3)_STATS_CTL	0x1180008000268 + α*0x8000000 + β*0x800	RSL	See page 1069
GMX(0..1)_TX(0..3)_CTL	0x1180008000270 + α*0x8000000 + β*0x800	RSL	See page 1069
GMX(0..1)_TX(0..3)_STAT0	0x1180008000280 + α*0x8000000 + β*0x800	RSL	See page 1070
GMX(0..1)_TX(0..3)_STAT1	0x1180008000288 + α*0x8000000 + β*0x800	RSL	See page 1070
GMX(0..1)_TX(0..3)_STAT2	0x1180008000290 + α*0x8000000 + β*0x800	RSL	See page 1070
GMX(0..1)_TX(0..3)_STAT3	0x1180008000298 + α*0x8000000 + β*0x800	RSL	See page 1071
GMX(0..1)_TX(0..3)_STAT4	0x11800080002A0 + α*0x8000000 + β*0x800	RSL	See page 1071
GMX(0..1)_TX(0..3)_STAT5	0x11800080002A8 + α*0x8000000 + β*0x800	RSL	See page 1071
GMX(0..1)_TX(0..3)_STAT6	0x11800080002B0 + α*0x8000000 + β*0x800	RSL	See page 1072
GMX(0..1)_TX(0..3)_STAT7	0x11800080002B8 + α*0x8000000 + β*0x800	RSL	See page 1072
GMX(0..1)_TX(0..3)_STAT8	0x11800080002C0 + α*0x8000000 + β*0x800	RSL	See page 1072
GMX(0..1)_TX(0..3)_STAT9	0x11800080002C8 + α*0x8000000 + β*0x800	RSL	See page 1073
GMX(0..1)_TX(0..3)_SGMII_CTL	0x1180008000300 + α*0x8000000 + β*0x800	RSL	See page 1073
GMX(0..1)_TX(0..3)_JAM_MODE	0x1180008000380 + α*0x8000000 + β*0x800	RSL	See page 1073
GMX(0..1)_BIST	0x1180008000400 + α*0x8000000	RSL	See page 1074
GMX(0..1)_RX_PRTS	0x1180008000410 + α*0x8000000	RSL	See page 1074
GMX(0..1)_RX_BP_DROP(0..3)	0x1180008000420 + α*0x8000000 + β*0x8	RSL	See page 1075
GMX(0..1)_RX_BP_ON(0..3)	0x1180008000440 + α*0x8000000 + β*0x8	RSL	See page 1075
GMX(0..1)_RX_BP_OFF(0..3)	0x1180008000460 + α*0x8000000 + β*0x8	RSL	See page 1076
GMX(0..1)_TX_PRTS	0x1180008000480 + α*0x8000000	RSL	See page 1076
GMX(0..1)_TX_IFG	0x1180008000488 + α*0x8000000	RSL	See page 1076
GMX(0..1)_TX_JAM	0x1180008000490 + α*0x8000000	RSL	See page 1077
GMX(0..1)_TX_COL_ATTEMPT	0x1180008000498 + α*0x8000000	RSL	See page 1077
GMX(0..1)_TX_PAUSE_PKT_DMAC	0x11800080004A0 + α*0x8000000	RSL	See page 1077
GMX(0..1)_TX_PAUSE_PKT_TYPE	0x11800080004A8 + α*0x8000000	RSL	See page 1077
GMX(0..1)_TX_OVR_BP	0x11800080004C8 + α*0x8000000	RSL	See page 1078

Cavium Confidential For Bowen Yang @2021/2/1

Table 23–9 GMX Registers (Continued)

Register	Address[1]	CSR Type[2]	Detailed Description
GMX(0..1)_TX_BP	0x11800080004D0 + α*0x8000000	RSL	See page 1078
GMX(0..1)_TX_CORRUPT	0x11800080004D8 + α*0x8000000	RSL	See page 1079
GMX(0..1)_RX_PRT_INFO	0x11800080004E8 + α*0x8000000	RSL	See page 1079
GMX(0..1)_TX_LFSR	0x11800080004F8 + α*0x8000000	RSL	See page 1079
GMX(0..1)_TX_INT_REG	0x1180008000500 + α*0x8000000	RSL	See page 1080
GMX(0..1)_TX_INT_EN	0x1180008000508 + α*0x8000000	RSL	See page 1080
GMX(0..1)_NXA_ADR	0x1180008000510 + α*0x8000000	RSL	See page 1081
GMX(0..1)_BAD_REG	0x1180008000518 + α*0x8000000	RSL	See page 1081
GMX(0..1)_STAT_BP	0x1180008000520 + α*0x8000000	RSL	See page 1081
GMX(0..1)_QSGMII_CTL	0x1180008000760 + α*0x8000000	RSL	See page 1082
GMX(0..1)_WOL_CTL	0x1180008000780 + α*0x8000000	RSL	See page 1082
GMX(0..1)_INF_MODE	0x11800080007F8 + α*0x8000000	RSL	See page 1082

1. For registers with multiple versions, α represents the values within the specified range and β the second range.
2. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

GMX Interrupt Registers
GMX(0..1)_RX(0..3)_INT_REG

This register allows interrupts to be sent to the control processor. If an error is noted in this register, and the corresponding bit in GMX(0..1)_RX(0/1..3)_INT_EN is set, an exception is raised. In RXAUI mode, port 0 is used for interrupt logging.

- Exception conditions <10:0> can also set the rcv/opcode in the received packet's workQ entry. GMX(0..1)_RX(0/1..3)_FRM_CHK provides a bit mask for configuring which conditions set the error.
- In half duplex operation, the expectation is that collisions will appear as either MINERR or CAREXT errors.

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:30>	—	RAZ	—	—	Reserved
<29>	WOL	R/W1C/H	0	0	A wake-on-LAN event has occurred.
<28>	HG2CC	R/W1C/H	0	0	HiGig2 received message CRC or control character error. Set when either a CRC8 error is detected or when a control character is found in the message bytes after the K.SOM. **NOTE:** HG2CC has higher priority than HG2FLD, i.e. a HiGig2 message that results in HG2CC getting set never sets HG2FLD.
<27>	HG2FLD	R/W1C/H	0	0	HiGig2 received message field error, as follows: **1.** MSG_TYPE field is not 0x00, i.e. it is not a FLOW CONTROL message, which is the only defined type for HiGig2. **2.** FWD_TYPE field is not 0x0, i.e. Link Level message, which is the only defined type for HiGig2. **3.** FC_OBJECT field is neither 0x0 for Physical Link nor 0x2 for Logical Link, which are the only two defined types in HiGig2.
<26>	UNDAT	R/W1C /H	0	0	Unexpected data. (XAUI mode only.)
<25>	UNEOP	R/W1C /H	0	0	Unexpected EOP. (XAUI mode only.)
<24>	UNSOP	R/W1C/H	0	0	Unexpected SOP. (XAUI mode only.)
<23>	BAD_TERM	R/W1C /H	0	0	Frame is terminated by a control character other than /T/. The error propagation control character /E/ is included as part of the frame and does not cause a frame termination. (XAUI mode only.)
<22>	BAD_SEQ	R/W1C /H	0	0	Reserved sequence detected. (XAUI mode only.)
<21>	REM_FAULT	R/W1C /H	0	0	Remote fault sequence detected. (XAUI mode only.)
<20>	LOC_FAULT	R/W1C /H	0	0	Local fault sequence detected. (XAUI mode only.)
<19>	PAUSE_DRP	R/W1C/H	0	0	Pause packet was dropped due to full GMX RX FIFO.
<18:16>	—	RAZ	—	—	Reserved
<15>	IFGERR	R/W1C/H	0	0	Interframe gap violation. Does not necessarily indicate a failure. (SGMII/QSGMII/1000BASE-X only.)
<14>	COLDET	R/W1C/H	0	0	Collision detection. Collisions can only occur in half-duplex mode. A collision is assumed by the receiver when the slottime (GMX(0..1)_PRT(0..3)_CFG[SLOTTIME]) is not satisfied. In 10/100 mode, this will result in a frame < SLOTTIME. In 1000 mode, it could result either in frame < SLOTTIME or a carrier extend error with the SLOTTIME. These conditions are visible by 1) transfer ended before slottime – COLDET or 2) carrier extend error – CAREXT. **NOTE:** RXAUI interfaces are inherently full-duplex so collisions cannot occur.
<13>	FALERR	R/W1C/H	0	0	False-carrier error, or carrier-extend error after slottime is satisfied. (SGMII/QSGMII/1000BASE-X only.)

Cavium Confidential For Bowei Yan 04/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<12>	RSVERR	R/W1C/H	0	0	Reserved opcode error.
<11>	PCTERR	R/W1C/H	0	0	Bad preamble / protocol error. Checks that the frame begins with a valid PREAMBLE sequence. Does not check the number of PREAMBLE cycles. In RXAUI mode, the column of data that was bad is logged in GMX(0..1)_RX_XAUI_BAD_COL.
<10>	OVRERR	R/W1C/H	0	0	Internal data aggregation overflow error. This interrupt should never assert. (SGMII/QSGMII/1000BASE-X only.)
<9>	—	RAZ	—	—	Reserved
<8>	SKPERR	R/W1C/H	0	0	Skipper error.
<7>	RCVERR	R/W1C/H	0	0	Data-reception error. Frame was received with data-reception error
<6:5>	—	RAZ	—	—	Reserved
<4>	FCSERR	R/W1C/H	0	0	FCS/CRC error. Frame was received with FCS/CRC error
<3>	JABBER	R/W1C/H	0	0	System-length error: frame was received with length > sys_length. An RX Jabber error indicates that a packet was received which is longer than the maximum allowed packet as defined by the system. GMX truncates the packet at the JABBER count. Failure to do so could lead to system instability.
<2>	—	RAZ	—	—	Reserved
<1>	CAREXT	R/W1C/H	0	0	Carrier-extend error. (SGMII/QSGMII/1000BASE-X only.)
<0>	MINERR	R/W1C	0	—	Pause frame was received with length < minFrameSize. Frame length checks are typically handled in PIP (PIP_INT_REG[MINERR]), but pause frames are normally discarded before being inspected by PIP. Total frame DA+SA+TL+DATA+PAD+FCS < 64.

GMX Interrupt-Enable Registers
GMX(0..1)_RX(0/1..3)_INT_EN

This register provides the interrupt-enable bits for the corresponding error in GMX(0..1)_RX(0..3)_INT_REG. In RXAUI mode, port 0 is used for checking. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:30>	—	RAZ	—	—	Reserved
<29>	WOL	R/W/H	0	0	A wake-on-LAN event has occurred.
<28>	HG2CC	R/W	0	0	HiGig2 CRC8 or control-charactererror interrupt enable.
<27>	HG2FLD	R/W	0	0	HiGig2 badfielderror interrupt enable.
<26>	UNDAT	R/W	0	0	Unexpected data. (RXAUI mode only.)
<25>	UNEOP	R/W	0	0	Unexpected EOP. (RXAUI mode only.)
<24>	UNSOP	R/W	0	0	Unexpected SOP. (RXAUI mode only.)
<23>	BAD_TERM	R/W	0	0	Frame is terminated by a control character other than /T/. The error propagation control character /E/ is included as part of the frame and does not cause a frame termination. (RXAUI mode only.)
<22>	BAD_SEQ	R/W	0	0	Reserved sequence detected. (RXAUI mode only.)
<21>	REM_FAULT	R/W	0	0	Remote fault sequence detected. (XAUI mode only.)
<20>	LOC_FAULT	R/W	0	0	Local fault sequence detected. (RXAUI mode only.)
<19>	PAUSE_DRP	R/W	0	0	Pause packet was dropped due to full GMX RX FIFO.
<18:16>	—	RAZ	—	—	Reserved
<15>	IFGERR	R/W	0x0	0	Interframe-gap violation. (SGMII/QSGMII/1000BASE-X only.)
<14>	COLDET	R/W	0x0	0	Collision detection. (SGMII/QSGMII/1000BASE-X only.)
<13>	FALERR	R/W	0x0	0	False-carrier error, or carrier-extend error after slottime is satisfied. (SGMII/QSGMII/1000BASE-X only.)

Cavium Confidential For Bowen Yang ALU 02/7/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<12>	RSVERR	R/W	0x0	0	Reserved opcodes.
<11>	PCTERR	R/W	0x0	0	Bad preamble / protocol.
<10>	OVRERR	R/W	0x0	0	Internal data aggregation overflow error. (SGMII/QSGMII/1000BASE-X only.)
<9>	—	RAZ	—	—	Reserved
<8>	SKPERR	R/W	0x0	0	Skipper error.
<7>	RCVERR	R/W	0x0	0	Frame was received with data reception error.
<6:5>	—	RAZ	—	—	Reserved
<4>	FCSERR	R/W	0x0	0	Frame was received with FCS/CRC error.
<3>	JABBER	R/W	0x0	0	Frame was received with length > sys_length.
<2>	—	RAZ	—	—	Reserved
<1>	CAREXT	R/W	0x0	0	Carrier extend error. (SGMII/QSGMII/1000BASE-X only.)
<0>	MINERR	R/W	0	0	Pause frame was received with length < minFrameSize.

GMX Port Configuration Registers
GMX(0..1)_PRT(0..3)_CFG

This register controls the configuration of the port. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved
<13>	TX_IDLE	RO/H	1	—	TX machine is idle.
<12>	RX_IDLE	RO/H	1	—	RX machine is idle.
<11:9>	—	RAZ	—	—	Reserved
<8>	SPEED_MSB	R/W	0	—	Link speed MSB [SPEED_MSB:SPEED] 10 = 10 Mbs operation 00 = 100 Mbs operation 01 = 1000 Mbs operation 11 = Reserved (SGMII/QSGMII/1000BASE-X only.)
<7:4>	—	RAZ	—	—	Reserved
<3>	SLOTTIME	R/W	1	—	Slot time for half-duplex operation 0 = 512 bit times (10/100 Mbs operation) 1 = 4096 bit times (1000 Mbs operation) (SGMII/QSGMII/1000BASE-X only.)
<2>	DUPLEX	R/W	1	—	Duplex mode: 0 = half-duplex (collisions/extensions/bursts), 1 = full-duplex (SGMII/QSGMII/1000BASE-X only.)
<1>	SPEED	R/W	1	—	Link Speed LSB [SPEED_MSB:SPEED] 10 = 10 Mbs operation 00 = 100 Mbs operation 01 = 1000 Mbs operation 11 = Reserved (SGMII/QSGMII/1000BASE-X only.)
<0>	EN	R/W	0	—	Link enable. When EN is clear, packets are not received or transmitted (including PAUSE and JAM packets). If EN is cleared while a packet is currently being received or transmitted, the packet is allowed to complete before the bus is idled. On the RX side, subsequent packets in a burst are ignored.

Cavium Confidential For Boyue Yang 2014

Frame Control Registers
GMX(0..1)_RX(0..3)_FRM_CTL

This register controls the handling of the frames.

The CTL_BCK/CTL_DRP bits control how the hardware handles incoming PAUSE packets. The most common modes of operation:

- CTL_BCK = 1, CTL_DRP = 1: hardware handles everything
- CTL_BCK = 0, CTL_DRP = 0: software sees all pause frames
- CTL_BCK = 0, CTL_DRP = 1: all pause frames are completely ignored

These control bits should be set to CTL_BCK = 0,CTL_DRP = 0 in half-duplex mode. Since PAUSE packets only apply to full duplex operation, any PAUSE packet would constitute an exception which should be handled by the processing cores. PAUSE packets should not be forwarded.

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	—	—	Reserved
<12>	PTP_MODE	R/W	0	0	Timestamp mode. When PTP_MODE is set, a 64-bit timestamp is prepended to every incoming packet. See Section 14.4. The timestamp bytes are added to the packet in such a way as to not modify the packet's receive byte count. This implies that the GMX(0..1)_RX(0..3)_JABBER, MINERR, GMX(0..1)_RX(0..3)_DECISION, GMX(0..1)_RX(0..3)_UDD_SKP, and the GMX_RX_STATS_* do not require any adjustment as they operate on the received packet size. When the packet reaches PIP, its size will reflect the additional bytes and is subject to the restrictions below. If PTP_MODE = 1 and PRE_CHK = 1, PRE_STRP must be 1. If PTP_MODE = 1, PIP_PRT_CFG(0/1..3,16..19,32..3943)[SKIP] should be increased by 8, PIP_PRT_CFG(0/1..3,16..19,32..3943)[HIGIG_EN] should be 0, PIP_FRM_LEN_CHK0[MAXLEN] should be increased by 8, PIP_FRM_LEN_CHK0[MINLEN] should be increased by 8, PIP_TAG_INC(0..63)[EN] should be adjusted, and PIP_PRT_CFG(0..3,16..19,24,32..39)[ALT_SKP_EN] should be 0.
<11>	—	RAZ	—	—	Reserved
<10>	NULL_DIS	R/W	0	0	When set, do not modify the MOD bits on NULL ticks due to partial packets.
<9>	PRE_ALIGN	R/W	0	0	When set, PREAMBLE parser aligns the SFD byte regardless of the number of previous PREAMBLE nibbles. In this mode, [PRE_STRP] should be set to account for the variable nature of the PREAMBLE. [PRE_CHK] must be set to enable this and all PREAMBLE features. (SGMII at 10/100Mbs only)
<8:7>	—	RAZ	—	—	Reserved
<6>	PRE_FREE	RO	1	1	When set, PREAMBLE checking is less strict. GMX will begin the frame at the first SFD. [PRE_CHK] must be set to enable this and all PREAMBLE features. (SGMII/QSGMII/1000BASE-X only.)
<5>	CTL_SMAC	R/W	0	0	Control pause frames can match station SMAC.
<4>	CTL_MCST	R/W	1	1	Control pause frames can match globally assign multicast address.
<3>	CTL_BCK	R/W	1	1	Forward pause information to TX block.
<2>	CTL_DRP	R/W	1	1	Drop control-pause frames.

Cavium Confidential For Power Card ATD 12/20/2012

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<1>	PRE_STRP	R/W	1	1	Strip off the preamble (when present). 0 = PREAMBLE + SFD is sent to core as part of frame 1 = PREAMBLE + SFD is dropped [PRE_CHK] must be set to enable this and all PREAMBLE features. If PTP_MODE=1 and PRE_CHK=1, PRE_STRP must be 1. When PRE_CHK is set (indicating that the PREAMBLE will be sent), PRE_STRP determines if the PREAMBLE+SFD bytes are thrown away or sent to the core as part of the packet. In either mode, the PREAMBLE+SFD bytes are not counted toward the packet size when checking against the MIN and MAX bounds. Furthermore, the bytes are skipped when locating the start of the L2 header for DMAC and Control frame recognition.
<0>	PRE_CHK	R/W	1	1	Check the preamble for correctness. This port is configured to send a valid 802.3 PREAMBLE to begin every frame. GMX checks that a valid PREAMBLE is received (based on PRE_FREE). When a problem does occur within the PREAMBLE sequence, the frame is marked as bad and not sent into the core. The GMX(0..1)_RX(0..3)_INT_REG[PCTERR] interrupt is also raised. When GMX0_TX_XAUI_CTL[HG_EN] is set, PRE_CHK must be zero. If PTP_MODE=1 and PRE_CHK=1, PRE_STRP must be 1.

Frame Check Registers

GMX(0..1)_RX(0/1..3)_FRM_CHK

This register indicates which frame errors will set the ERR bit of the frame. In RXAUI mode, port 0 is used for checking. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8>	SKPERR	R/W	1	1	Skipper error.
<7>	RCVERR	R/W	1	1	Frame was received with data-reception error.
<6:5>	—	RAZ	—	—	Reserved
<4>	FCSERR	R/W	1	1	Frame was received with FCS/CRC error.
<3>	JABBER	R/W	1	1	Frame was received with length > sys_length.
<2>	—	RAZ	—	—	Reserved
<1>	CAREXT	R/W	1	1	Carrier extend error. (SGMII/QSGMII/1000BASE-X only.)
<0>	MINERR	R/W	1	1	Pause frame was received with length < minFrameSize.

Cavium Confidential For Bowen Yang ALU 02/17/2011

GMX Maximum Packet-Size Registers
GMX(0..1)_RX(0..3)_JABBER

This register specifies the maximum size for packets, beyond which the GMX truncates. In RXAUI mode, port 0 is used for checking. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	CNT	R/W	0x2800	0x2800	Byte count for jabber check. Failing packets set the JABBER interrupt and are optionally sent with opcode = JABBER. GMX truncates the packet to CNT bytes. CNT must be 8-byte aligned such that CNT[2:0] = 000.

Note:

The packet that is sent to the packet-input logic will have an additional eight bytes if GMX(0..1)_RX(0..3)_FRM_CTL[PRE_CHK] = 1 and GMX(0..1)_RX(0..3)_FRM_CTL[PRE_STRP] = 0. The maximum packet that will be sent is defined as:

```
max_sized_packet =
GMX(0..1)_RXn_JABBER[CNT]+
((GMX(0..1)_RXn_FRM_CTL[PRE_CHK] &
!GMX(0..1)_RXn_FRM_CTL[PRE_STRP]) × 8)
```

GMX Packet Decision Registers
GMX(0..1)_RX(0..3)_DECISION

This register specifies the byte count used to determine when to accept or to filter a packet. As each byte in a packet is received by GMX, the L2 byte count (i.e. the number of bytes from the beginning of the L2 header (DMAC)) is compared against CNT. In normal operation, the L2 header begins after the PREAMBLE + SFD (GMX(0..1)_RX(0..3)_FRM_CTL[PRE_CHK] = 1) and any optional UDD skip data (GMX(0..1)_RX(0..3)_UDD_SKP[LEN]). See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved
<4:0>	CNT	R/W	0x18	0x18	The byte count used to decide when to accept or filter a packet. Refer to Table 23–10.

Notes:

When GMX(0..1)_RX(0..3)_FRM_CTL[PRE_CHK] = 0, PREAMBLE + SFD are prepended to the packet and require UDD skip length to account for them.

Table 23–10 GMX Decisions

Port Mode	L2 Size[1]	
	<GMX_RX_DECISION bytes (default = 0x24)	≥GMX_RX_DECISION bytes (default = 0x24)
Full Duplex	Accept packet No filtering is applied	Apply filters Accept packet based on DMAC and PAUSE packet filters
Half Duplex	Drop packet Packet is unconditionally dropped	Apply filters Accept packet based on DMAC

1. where L2_size = MAX(0, total_packet_size – GMX(0..1)_RX(0..3)_UDD_SKP[LEN] – ((GMX(0..1)_RX(0..3)_FRM_CTL[PRE_CHK] = 1) × 8)

Cavium Confidential For Power Kang ALU 02/27/2014

GMX User-Defined Data Skip Registers
GMX(0..1)_RX(0..3)_UDD_SKP

This register specifies the amount of user-defined data (UDD) added before the start of the L2C data. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8>	FCSSEL	R/W	0	0	Include the skip bytes in the FCS calculation 0 = all skip bytes are included in FCS 1 = the skip bytes are not included in FCS When GMX0_TX_XAUI_CTL[HG_EN] is set, this field must be zero. The skip bytes are part of the packet and are sent through the IOI packet interface and are handled by PIP. The system can determine if the UDD bytes are included in the FCS check by using the FCSSEL field, if the FCS check is enabled.
<7>	—	RAZ	—	—	Reserved
<6:0>	LEN	R/W	0x0	0x0	Amount of user-defined data before the start of the L2C data, in bytes. Setting to 0 means L2C comes first; maximum value is 64. LEN must be 0x0 in half-duplex operation. When GMX0_TX_XAUI_CTL[HG_EN] is set, this field must be set to 12 or 16 (depending on HiGig header size) to account for the HiGig header. LEN = 12 selects HiGig/HiGig+; LEN = 16 selects HiGig2.

Notes:

Assume that the PREAMBLE/SFD is always at the start of the frame, even before UDD bytes. In most cases, there will be no preamble in these cases, since it will be packet interface in direct communication with another packet interface (MAC to MAC), without a PHY involved.

In all cases, the UDD bytes are sent through the packet interface as part of the packet. The UDD bytes are never stripped from the actual packet.

If LEN ≠ 0 (i.e. there is some UDD), then length checking should be disabled (i.e. GMX(0..1)_RX(0/1..3)_FRM_CHK[LENERR] should be cleared and GMX(0..1)_RX(0..3)_INT_REG[LENERR] set to 0.

Address filtering and control-packet filtering can be performed as needed.

GMX RX Statistics Control Registers
GMX(0..1)_RX(0..3)_STATS_CTL

This register controls the receive statistics registers. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	RD_CLR	R/W	0	0	Clear on read. When this bit is set, any receive statistics registers are cleared when read.

Cavium Confidential For Bowman ALU 02/27/2014

GMX Minimum Interframe-Gap Cycles Registers
GMX(0..1)_RX(0..3)_IFG

This register specifies the minimum number of interframe-gap (IFG) cycles between packets. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	IFG	R/W	0x8	0x8	Min IFG (in IFG×8 bits) between packets used to determine IFGERR. Normally IFG is 96 bits. **NOTE:** Note in some operating modes, IFG cycles can be inserted or removed in order to achieve clock rate adaptation. For these reasons, the default value is slightly conservative and does not check up to the full 96 bits of IFG. (SGMII/QSGMII/1000BASE-X only.)

GMX RX Pause-Packet Drop Time Registers
GMX(0..1)_RX(0..3)_PAUSE_DROP_TIME

This register extracts the [TIME] field in a PAUSE packet that was dropped due to a GMX RX FIFO full condition. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	STATUS	R/W1C/H	0x0	—	Time extracted from the dropped PAUSE packet.

GMX RX Good Packets Registers
GMX(0..1)_RX(0..3)_STATS_PKTS

This register provides a count of good received packets. The following are examples of "not good" packets:

- packets recognized as PAUSE packets
- packets dropped due to the DMAC filter
- packets dropped due to FIFO full status
- packets that have opcodes ≠ 0 (FCS, Length, etc.).

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of received good packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/21/2014

GMX RX Good Packets Octet Registers
GMX(0..1)_RX(0..3)_STATS_OCTS

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	CNT	RC/W/H	0x0	—	Octet count of received good packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

GMX RX Pause Packets Registers
GMX(0..1)_RX(0..3)_STATS_PKTS_CTL

This register provides a count of all packets received that were recognized as flow-control or pause packets. Note that pause packets with any kind of error are also counted in GMX(0..1)_RX(0..3)_STATS_PKTS_BAD.

- Pause packets can be optionally dropped or forwarded based on GMX(0..1)_RX(0..3)_FRM_CTL[CTL_DRP]. This count increments regardless of whether the packet is dropped.
- Pause packets are never counted in GMX(0..1)_RX(0..3)_STATS_PKTS_BAD.
- Packets dropped due to the DMAC filter are counted in GMX(0..1)_RX(0..3)_STATS_PKTS_DMAC and not here.

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of received flow-control or pause packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

GMX RX Pause Packets Octet Registers
GMX(0..1)_RX(0..3)_STATS_OCTS_CTL

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	CNT	RC/W/H	0x0	—	Octet count of received flow-control or pause packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GMX RX DMAC Packets Registers
GMX(0..1)_RX(0..3)_STATS_PKTS_DMAC

This register provides a count of all packets received that were dropped by the DMAC filter. Packets that match the DMAC are dropped and counted here regardless of whether they were bad packets. These packets are never counted in GMX(0..1)_RX(0..3)_STATS_PKTS. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of filtered DMAC packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

GMX RX DMAC Packets Octet Registers
GMX(0..1)_RX(0..3)_STATS_OCTS_DMAC

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	CNT	RC/W/H	0x0	—	Octet count of filtered DMAC packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

GMX RX Overflow Packets Registers
GMX(0..1)_RX(0..3)_STATS_PKTS_DRP

This register provides a count of all packets received that were dropped due to a full receive FIFO. It counts both partial packets in which there was enough space in the RX FIFO to begin to buffer, and the packet and total drops in which no packet was sent to PIP/IPD. It counts all packets dropped by the FIFO, both good and bad packets received. It does not count packets dropped by the DMAC or PAUSE-packet filters.

Some packets that were not able to satisfy the DECISION_CNT may not actually be dropped by CN71XX, but they will be counted here as if they were dropped.

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of dropped packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang 02/27/2014

GMX RX Overflow Packets Octet Registers GMX(0..1)_RX(0..3)_STATS_OCTS_DRP

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	CNT	RC/W/H	0x0	—	Octet count of dropped packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

GMX RX Bad Packets Registers GMX(0..1)_RX(0..3)_STATS_PKTS_BAD

This register provides a count of all packets received with some error that were not dropped either due to the DMAC filter or lack of room in the receive FIFO. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of received packets with opcode ≠ 0

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Address-Filtering Control Registers
GMX(0..1)_RX(0..3)_ADR_CTL

See Table 23–9 for the address. In RXAUI mode, only GMX_RX0_ADR_CTL is used. GMX_RX(1..3)_ADR_CTL should not be used.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	CAM_MODE	R/W	0	—	Allow or deny DMAC address filter 0 = reject the packet on DMAC address match 1 = accept the packet on DMAC address match
<2:1>	MCST	R/W	0x0	—	Multicast mode: 0x0 = Use the address-filter CAM 0x1 = Force reject all multicast packets 0x2 = Force accept all multicast packets 0x3 = Reserved
<0>	BCST	R/W	1	—	Accept all broadcast packets.

Algorithm:

Here is some pseudo-code that represents the address-filter behavior.

```
bool dmac_addr_filter(uint8 prt, uint48 dmac) {
  ASSERT(prt >= 0 && prt <= 3);
  if (is_bcst(dmac))                              // broadcast accept
    return (GMX_RX{prt}_ADR_CTL[BCST] ? ACCEPT : REJECT);
  if (is_mcst(dmac) & GMX_RX{prt}_ADR_CTL[MCST] == 1)   // multicast reject
    return REJECT;
  if (is_mcst(dmac) & GMX_RX{prt}_ADR_CTL[MCST] == 2)   // multicast accept
    return ACCEPT;

  cam_hit = 0;

  for (i=0; i<32; i++) {
    if (GMX_RX{prt}_ADR_CAM_ALL_EN[EN<i>] == 0)
      continue;
    uint48 unswizzled_mac_adr = 0x0;
    for (j=5; j>=0; j--) {
       unswizzled_mac_adr = (unswizzled_mac_adr << 8) |
                            GMX_RX{i>>3}_ADR_CAM{j}[ADR<(i&7)*8+7:(i&7)*8>];
    }
    if (unswizzled_mac_adr == dmac) {
      cam_hit = 1;
      break;
    }
  }

  if (cam_hit)
    return (GMX_RX{prt}_ADR_CTL[CAM_MODE] ? ACCEPT : REJECT);
  else
    return (GMX_RX{prt}_ADR_CTL[CAM_MODE] ? REJECT : ACCEPT);
}
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

Address-Filtering Control Enable Registers
GMX(0..1)_RX(0..3)_ADR_CAM_EN

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	EN	R/W	0x0	—	CAM-entry enable bits. GMX has 32 DMAC entries that can be accessed with the GMX(0..1)_RX(0..3)_ADR_CAM(0..5) CSRs. These 32 DMAC entries can be used by any of the two SGMII/QSGMII MACs or the RXAUI MAC. Each port has independent control of which of the 32 DMAC entries to include in the CAM lookup. Legacy GMX implementations used eight DMAC entries while current implementations use 32 common entries. This register is intended for legacy applications that only require eight DMAC CAM entries per MAC. New applications may choose to ignore GMX(0..1)_RX*n*_ADR_CAM_EN using GMX(0..1)_RX*n*_ADR_CAM_ALL_EN instead. EN controls the enable bits for the eight legacy CAM entries as follows: port0, EN = GMX_RX0_ADR_CAM_ALL_EN[EN<7:0>] port1, EN = GMX_RX1_ADR_CAM_ALL_EN[EN<15:8>] port2, EN = GMX_RX2_ADR_CAM_ALL_EN[EN<23:16>] port3, EN = GMX_RX3_ADR_CAM_ALL_EN[EN<31:24>] The full 32 independent per-MAC enable bits are in GMX(0..1)_RX*n*_ADR_CAM_ALL_EN. Therefore, write operations to GMX(0..1)_RX*n*_ADR_CAM_ALL_EN[EN] are reflected in EN and writes to EN are reflected in GMX(0..1)_RXX_ADR_CAM_ALL_EN[EN]. In RXAUI mode, only GMX(0..1)_RX0_ADR_CAM_EN is used; GMX_RX(1..3)_ADR_CAM_EN should not be used.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Address-Filtering Control Enable Registers GMX(0..1)_RX(0..3)_ADR_CAM_ALL_EN

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	EN	R/W	0x0	—	CAM-entry enable bits. GMX has 32 DMAC entries that can be accessed with the GMX(0..1)_RX(0..3)_ADR_CAM(0..5) CSRs. These 32 DMAC entries can be used by any of the two SGMII/QSGMII MACs or the RXAUI MAC. Each port has independent control of which of the 32 DMAC entries to include in the CAM lookup. GMX(0..1)_RX*n*_ADR_CAM_ALL_EN was not present in legacy GMX implementations which had only eight DMAC CAM entries. New applications may choose to ignore GMX(0..1)_RX*n*_ADR_CAM_EN using GMX(0..1)_RX*n*_ADR_CAM_ALL_EN instead. EN represents the full 32 independent per-MAC enable bits. Writes to EN are reflected in GMX(0..1)_RX*n*_ADR_CAM_EN[EN] and writes to GMX(0..1)_RX*n*_ADR_CAM_EN[EN] are reflected in EN. Refer to GMX(0..1)_RX*n*_ADR_CAM_EN for the CSR mapping. In RXAUI mode, only GMX(0..1)_RX0_ADR_CAM_ALL_EN is used; GMX(0..1)_RX(1..3)_ADR_CAM_ALL_EN should not be used.

Address-Filtering CAM Control Registers GMX(0..1)_RX(0..3)_ADR_CAM0

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	ADR	R/W	0x0	—	The DMAC address to match on. Each entry contributes eight bits to one of eight matchers. The CAM matches against unicast or multicast DMAC addresses. All GMX(0..1)_RX(0..3)_ADR_CAM(0..5) registers can be used in either SGMII/QSGMII or RXAUI mode such that any GMX MAC can use any of the 32 common DMAC entries. GMX(0..1)_RX(1..3)_ADR_CAM(0..5) are the only non-port0 registers used in RXAUI mode.

Cavium Confidential For Peter Yang 02/27/2014

Address-Filtering CAM Control Registers GMX(0..1)_RX(0..3)_ADR_CAM1

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	ADR	R/W	0x0	—	The DMAC address to match on. Each entry contributes eight bits to one of eight matchers. The CAM matches against unicast or multicast DMAC addresses. All GMX(0..1)_RX(0..3)_ADR_CAM(0..5) registers can be used in either SGMII/QSGMII or RXAUI mode such that any GMX MAC can use any of the 32 common DMAC entries. GMX(0..1)_RX(1..3)_ADR_CAM(0..5) are the only non-port0 registers used in RXAUI mode.

Address-Filtering CAM Control Registers GMX(0..1)_RX(0..3)_ADR_CAM2

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	ADR	R/W	0x0	—	The DMAC address to match on. Each entry contributes eight bits to one of eight matchers. The CAM matches against unicast or multicast DMAC addresses. All GMX(0..1)_RX(0..3)_ADR_CAM(0..5) registers can be used in either SGMII/QSGMII or RXAUI mode such that any GMX MAC can use any of the 32 common DMAC entries. GMX(0..1)_RX(1..3)_ADR_CAM(0..5) are the only non-port0 registers used in RXAUI mode.

Address-Filtering CAM Control Registers GMX(0..1)_RX(0..3)_ADR_CAM3

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	ADR	R/W	0x0	—	The DMAC address to match on. Each entry contributes eight bits to one of eight matchers. The CAM matches against unicast or multicast DMAC addresses. All GMX(0..1)_RX(0..3)_ADR_CAM(0..5) registers can be used in either SGMII/QSGMII or RXAUI mode such that any GMX MAC can use any of the 32 common DMAC entries. GMX(0..1)_RX(1..3)_ADR_CAM(0..5) are the only non-port0 registers used in RXAUI mode.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Address-Filtering CAM Control Registers GMX(0..1)_RX(0..3)_ADR_CAM4

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	ADR	R/W	0x0	—	The DMAC address to match on. Each entry contributes eight bits to one of eight matchers. The CAM matches against unicast or multicast DMAC addresses. All GMX(0..1)_RX(0..3)_ADR_CAM(0..5) registers can be used in either SGMII/QSGMII or RXAUI mode such that any GMX MAC can use any of the 32 common DMAC entries. GMX(0..1)_RX(1..3)_ADR_CAM(0..5) are the only non-port0 registers used in RXAUI mode.

Address-Filtering CAM Control Registers GMX(0..1)_RX(0..3)_ADR_CAM5

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	ADR	R/W	0x0	—	The DMAC address to match on. Each entry contributes eight bits to one of eight matchers. The CAM matches against unicast or multicast DMAC addresses. All GMX(0..1)_RX(0..3)_ADR_CAM(0..5) registers can be used in either SGMII/QSGMII or RXAUI mode such that any GMX MAC can use any of the 32 common DMAC entries. GMX(0..1)_RX(1..3)_ADR_CAM(0..5) are the only non-port0 registers used in RXAUI mode.

Packet TX Threshold Registers GMX(0..1)_TX(0..3)_THRESH

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8:0>	CNT	R/W	0x20	0x20	Number of 16-byte ticks to accumulate in the TX FIFO before sending on the packet interface. This field should be large enough to prevent underflow on the packet interface and must never be set to 0x0. This register cannot exceed the TX FIFO depth as follows. GMX_TX_PRTS = 0,1: CNT maximum = 0x100 GMX_TX_PRTS = 2: CNT maximum = 0x080 GMX_TX_PRTS = 3,4: CNT maximum = 0x040 **NOTE:** In RXAUI mode, port 0 is used for checking. Since RXAUI mode uses a single TX FIFO and is a higher data rate, Cavium recommends a value of 0x80.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Packet TX Append Control Registers GMX(0..1)_TX(0..3)_APPEND

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	FORCE_FCS	R/W	1	1	Append the ethernet FCS on each pause packet when FCS = 0. Pause packets are normally padded to 60 bytes. If GMX(0..1)_TX(0..3)_MIN_PKT[MIN_SIZE] exceeds 59, then FORCE_FCS is not used.
<2>	FCS	R/W	1	1	Append the Ethernet FCS on each packet.
<1>	PAD	R/W	1	1	Append PAD bytes such that a minimum-sized packet is transmitted.
<0>	PREAMBLE	R/W	1	1	Prepend the Ethernet preamble on each transfer. When GMX0_TX_XAUI_CTL[HG_EN] is set, PREAMBLE must be zero.

Packet TX Slottime Counter Registers GMX(0..1)_TX(0..3)_SLOT

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved
<9:0>	SLOT	R/W	0x200	0x200	Slottime (refer to 802.3 to set correctly) (SGMII/SGMII/1000BASE-X only.) 10/100Mbs: set SLOT to 0x40 (corresponds to 512 bit times) 1000Mbs: set SLOT to 0x200 (corresponds to 4096 bitimes)

Packet TX Burst-Counter Registers GMX(0..1)_TX(0..3)_BURST

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	BURST	R/W	0x2000	0x2000	Burst (refer to 802.3 to set correctly). (SGMII/QSGMII/1000BASE-X only.) Only valid for 1000Mb/s half-duplex operation as follows: half duplex/1000Mb/s: 0x2000 all other modes: 0x0

Packet SMAC Registers GMX(0..1)_SMAC(0..3)

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	SMAC	R/W	0x0	—	The SMAC field is used for generating and accepting control pause packets.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Packet TX Pause Packet Pause-Time Registers GMX(0..1)_TX(0..3)_PAUSE_PKT_TIME

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TIME	R/W	0x60	—	Provides the pause_time field placed in outbound 802.3 PAUSE packets, HiGig2 messages, or PFC/CBFC pause packets in 512 bit-times. Normally, TIME > GMX(0..1)_TX(0..3)_PAUSE_PKT_INTERVAL[INTERVAL].

Notes:

Choosing proper values of GMX(0..1)_TX(0..3)_PAUSE_PKT_TIME[TIME] and GMX(0..1)_TX(0..3)_PAUSE_PKT_INTERVAL[INTERVAL] can be challenging. Cavium suggests that TIME be much greater than INTERVAL and GMX(0..1)_TX(0..3)_PAUSE_ZERO[SEND] be set to 1. This allows a periodic refresh of the PAUSE count and then when the backpressure condition is lifted, a PAUSE packet with TIME = 0 is sent indicating that CN71XX is ready for additional data.

If the system chooses to not set GMX(0..1)_TX(0..3)_PAUSE_ZERO[SEND], then Cavium suggests that TIME and INTERVAL are programmed such that they satisfy the following rule:

INTERVAL ≤ TIME – (largest_pkt_size + IFG + pause_pkt_size)

where:

largest_pkt_size = the largest packet that the system can send (normally 1518 bytes)

IFG = the interframe gap

pause_pkt_size = the size of the PAUSE packet (normally 64 bytes).

Packet TX Minimum-Size-Packet Registers GMX(0..1)_TX(0..3)_MIN_PKT

This register specifies the minimum size a packet should be. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	MIN_SIZE	R/W	0x3B	0x3B	Minimum frame size in bytes before the FCS is applied. Padding is only appended when GMX(0..1)_TX(0..3)_APPEND[PAD] for the corresponding port is set. In SGMII mode, packets are padded to MIN_SIZE+1. The reset value pads to 60 bytes. In RXAUI mode, packets are padded to MIN(252,(MIN_SIZE+1 & ~0x3)). When GMX0_TX_XAUI_CTL[HG_EN] is set, the HiGig header (12B or 16B) is normally added to the packet, so MIN_SIZE should be 59 + 12 = 71B for HiGig or 59 + 16 = 75B for HiGig2.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Packet TX Pause-Packet Transmission-Interval Registers GMX(0..1)_TX(0..3)_PAUSE_PKT_INTERVAL

This register specifies how often Pause packets are sent. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	INTERVAL	R/W	0x10	—	Arbitrate for a 802.3 Pause packet, HiGig2 message, or PFC/CBFC pause packet every (INTERVAL × 512) bit-times. Normally, 0 < INTERVAL < GMX(0..1)_TX(0..3)_PAUSE_PKT_TIME[TIME]. INTERVAL = 0 only send a single PAUSE packet for each backpressure event.

Notes:

Choosing proper values of GMX(0..1)_TX(0..3)_PAUSE_PKT_TIME[TIME] and GMX(0..1)_TX(0..3)_PAUSE_PKT_INTERVAL[INTERVAL] can be challenging. Cavium suggests that TIME be much greater than INTERVAL and GMX(0..1)_TX(0..3)_PAUSE_ZERO[SEND] be set to 1. This allows a periodic refresh of the PAUSE count and then when the backpressure condition is lifted, a PAUSE packet with TIME = 0 is sent indicating that CN71XX is ready for additional data.

If the system chooses to not set GMX(0..1)_TX(0..3)_PAUSE_ZERO[SEND], then Cavium suggests that TIME and INTERVAL are programmed such that they satisfy the following rule:

INTERVAL <= TIME – (largest_pkt_size + IFG + pause_pkt_size)

where:

largest_pkt_size = the largest packet that the system can send (normally 1518 bytes)

IFG = the interframe gap

pause_pkt_size = the size of the PAUSE packet (normally 64 bytes).

Packet TX Software-Pause Registers GMX(0..1)_TX(0..3)_SOFT_PAUSE

This register specifies the pause time. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TIME	R/W	0x0	—	Back off the TX bus for (TIME × 512) bit-times

Cavium Confidential For Bowen Yang ALU 8/27/2014

Packet TX Time-to-Backpressure Registers
GMX(0..1)_TX(0..3)_PAUSE_TOGO

This register is the pause-deferral counter. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:16>	MSG_TIME	RO/H	—	—	Amount of time remaining to backpressure, from the HiGig2 physical message pause timer (only valid on port0).
<15:0>	TIME	RO/H	—	—	Amount of time remaining to backpressure, from the standard 802.3 pause timer.

Packet TX Pause-Zero-Enable Registers
GMX(0..1)_TX(0..3)_PAUSE_ZERO

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	SEND	R/W	1	1	Send pause-zero enable.When this bit is set, and the backpressure condition is clear, it allows sending a pause packet with pause_time of 0 to enable the channel.

GMX TX Statistics Control Registers
GMX(0..1)_TX(0..3)_STATS_CTL

This register controls the transmit statistics registers. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	RD_CLR	R/W	0	0	Clear on read. When this bit is set, any transmit statistics registers are cleared when read.

GMX Transmit Control Registers
GMX(0..1)_TX(0..3)_CTL

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	XSDEF_EN	R/W	1	1	Enables the excessive-deferral check for statistics and interrupts. (SGMII/QSGMII/1000BASE-X half-duplex only.)
<0>	XSCOL_EN	R/W	1	1	Enables the excessive-collision check for statistics and interrupts. (SGMII/QSGMII/1000BASE-X half-duplex only.)

Cavium Confidential For Bowen Yang ALU 02/27/2014

Transmit Statistics Registers 0
GMX(0..1)_TX(0..3)_STAT0

This register maintains statistics on excessive deferrals and excessive collisions. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	XSDEF	RC/W/H	0x0	—	Number of packets dropped (never successfully sent) due to excessive deferrals. (SGMII/QSGMII/1000BASE-X half-duplex only.)
<31:0>	XSCOL	RC/W/H	0x0	—	Number of packets dropped (never successfully sent) due to excessive collision. Defined by GMX(0..1)_TX_COL_ATTEMPT[LIMIT]. (SGMII/QSGMII/1000BASE-X half-duplex only.)

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Registers 1
GMX(0..1)_TX(0..3)_STAT1

This register maintains statistics on single collisions and multiple collisions. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	SCOL	RC/W/H	0x0	—	Number of packets sent with a single collision. (SGMII/QSGMII/1000BASE-X half-duplex only.)
<31:0>	MCOL	RC/W/H	0x0	—	Number of packets sent with multiple collisions but less than GMX(0..1)_TX_COL_ATTEMPT[LIMIT]. (SGMII/QSGMII/1000BASE-X half-duplex only.)

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Registers 2
GMX(0..1)_TX(0..3)_STAT2

This register maintains statistics on the number of octets sent on the interface. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	OCTS	RC/W/H	0x0	—	Number of total octets sent on the interface. Does not count octets from frames that were truncated due to collisions in half-duplex mode.

NOTE:
- Octet counts are the sum of all data transmitted on the wire, including packet data, pad bytes, FCS bytes, Pause bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when GMX(0..1)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen yard ALU 02/27/2014

Transmit Statistics Registers 3
GMX(0..1)_TX(0..3)_STAT3

Used for GMX_TX_STATS_PKTS.

This register maintains statistics on the number of frames sent on the interface. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	PKTS	RC/W/H	0x0	—	Number of total frames sent on the interface. Does not count frames that were truncated due to collisions in half-duplex mode.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Registers 4
GMX(0..1)_TX(0..3)_STAT4

This register maintains statistics on the number of packets sent on the interface with octet counts of 64 and less than 64. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	HIST1	RC/W/H	0x0	—	Number of packets sent with an octet count of 64.
<31:0>	HIST0	RC/W/H	0x0	—	Number of packets sent with an octet count < 64.

NOTE:
- Packet length is the sum of all data transmitted on the wire for the given packet, including packet data, pad bytes, FCS bytes, PAUSE bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when GMX(0..1)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Registers 5
GMX(0..1)_TX(0..3)_STAT5

This register maintains statistics on the number of packets sent on the interface with octet counts between 65 and 127 and between 128 and 255. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	HIST3	RC/W/H	0x0	—	Number of packets sent with an octet count of 128 – 255.
<31:0>	HIST2	RC/W/H	0x0	—	Number of packets sent with an octet count of 65 – 127.

NOTE:
- Packet length is the sum of all data transmitted on the wire for the given packet, including packet data, pad bytes, FCS bytes, PAUSE bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when GMX(0..1)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/17/2014

Transmit Statistics Registers 6
GMX(0..1)_TX(0..3)_STAT6

This register maintains statistics on the number of packets sent on the interface with octet counts between 256 and 511 and between 512 and 1023. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	HIST5	RC/W/H	0x0	—	Number of packets sent with an octet count of 512 – 1023.
<31:0>	HIST4	RC/W/H	0x0	—	Number of packets sent with an octet count of 256 – 511.

NOTE:
- Packet length is the sum of all data transmitted on the wire for the given packet, including packet data, pad bytes, FCS bytes, PAUSE bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when GMX(0..1)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Registers 7
GMX(0..1)_TX(0..3)_STAT7

This register maintains statistics on the number of packets sent on the interface with octet counts between 1024 and 1518 and greater than 1518. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	HIST7	RC/W/H	0x0	—	Number of packets sent with an octet count of > 1518.
<31:0>	HIST6	RC/W/H	0x0	—	Number of packets sent with an octet count of 1024 – 1518.

NOTE:
- Packet length is the sum of all data transmitted on the wire for the given packet, including packet data, pad bytes, FCS bytes, PAUSE bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when GMX(0..1)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Registers 8
GMX(0..1)_TX(0..3)_STAT8

This register maintains statistics on the number of packets sent to broadcast DMAC and to multicast DMAC. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	MCST	RC/W/H	0x0	—	Number of packets sent to multicast DMAC. Does not include BCST packets.
<31:0>	BCST	RC/W/H	0x0	—	Number of packets sent to broadcast DMAC. Does not include MCST packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.
- Note that the GMX determines if the packet is MCST or BCST from the DMAC of the packet. GMX assumes that the DMAC lies in the first six bytes of the packet as per the 802.3 frame definition. If the system requires additional data before the L2 header, then the MCST and BCST counters may not reflect reality and should be ignored by software.

Cavium Confidential For Bowen Yang ALU 04/27/2014

Transmit Statistics Registers 9
GMX(0..1)_TX(0..3)_STAT9

This register maintains statistics on the number of underflow packets and control packets. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	UNDFLW	RC/W/H	0x0	—	Number of underflow packets.
<31:0>	CTL	RC/W/H	0x0	—	Number of Control packets (PAUSE flow control) generated by GMX. It does not include control packets forwarded or generated by the cores. CTL counts the number of generated PFC frames; CTL does not track the number of generated HG2 messages.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0..1)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

SGMII Control Register
GMX(0..1)_TX(0..3)_SGMII_CTL

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	ALIGN	R/W	1	0	Align the transmission to even cycles: Recommended value is: ALIGN = GMX(0..1)_TX(0..3)_APPEND[PREAMBLE]. (See Section 23.4.6.7, Transmit Conversion to Code Groups for a complete discussion.) 0 = Data can be sent on any cycle. In this mode, the interface functions at maximum bandwidth. It is possible for the TX PCS machine to drop the first byte of the TX frame. When GMX(0..1)_TX(0..3)_APPEND[PREAMBLE] is set, the first byte is a preamble byte, which can be dropped to compensate for an extended IPG. 1 = Data is only sent on even cycles. In this mode, there can be bandwidth implications when sending odd-byte packets as the IPG can extend an extra cycle. There will be no loss of data. (SGMII/QSGMII/1000BASE-X only)

Transmit JAM Mode Registers
GMX(0..1)_TX(0..3)_JAM_MODE

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	MODE	R/W	0	0	When forming half-duplex JAM packets, use either the incoming SOP or CRS event to begin packet creation: 0 = SOP event, 1 = CRS event. (SGMII/QSGMII/1000BASE-X half-duplex only)

Cavium Confidential For Beta Yard 02 Dec 14

GMX BIST Results Register
GMX(0..1)_BIST

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved.
<24:0>	STATUS	RO	0x0	0x0	BIST results. 0 = pass, 1 = fail 0: gmx#.inb.fif_bnk0 1: gmx#.inb.fif_bnk1 2: gmx#.inb.fif_bnk2 3: gmx#.inb.fif_bnk3 4: gmx#.inb.fif_bnk_ext0 5: gmx#.inb.fif_bnk_ext1 6: gmx#.inb.fif_bnk_ext2 7: gmx#.inb.fif_bnk_ext3 8: gmx#.outb.fif.fif_bnk0 9: gmx#.outb.fif.fif_bnk1 10: gmx#.outb.fif.fif_bnk2 11: gmx#.outb.fif.fif_bnk3 12: gmx#.outb.fif.fif_bnk_ext0 13: gmx#.outb.fif.fif_bnk_ext1 14: gmx#.outb.fif.fif_bnk_ext2 15: gmx#.outb.fif.fif_bnk_ext3 16: gmx#.csr.gmi0.srf8x64m1_bist 17: gmx#.csr.gmi1.srf8x64m1_bist 18: gmx#.csr.gmi2.srf8x64m1_bist 19: gmx#.csr.gmi3.srf8x64m1_bist 20: gmx#.csr.drf20x32m2_bist 21: gmx#.csr.drf20x48m2_bist 22: gmx#.outb.stat.drf16x27m1_bist 23: gmx#.outb.stat.drf40x64m1_bist 24: xgmii.tx.drf16x38m1_async_bist

RX Ports Register
GMX(0..1)_RX_PRTS

This register specifies the number of ports, which indicates the number of FIFOs that make up the RX buffer. See Table 23–9 for the address.

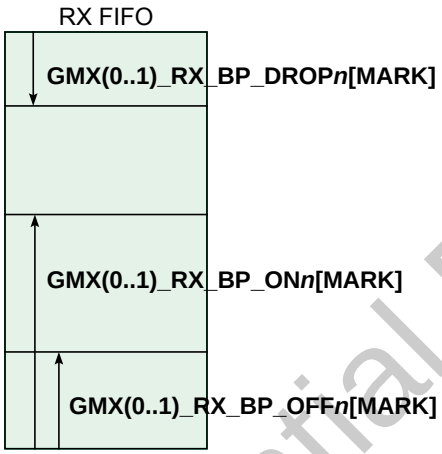
Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved
<2:0>	PRTS	R/W	0x4	0x4	In SGMII/QSGMII/1000BASE-X mode, the RX buffer can be carved into several logical buffers depending on the number or implemented ports. 0 or 1 port = 512 ticks / 4096 bytes 2 ports = 256 ticks / 2048 bytes 3 or 4 ports = 128 ticks / 1024 bytes This field must be set to 0x1 in XAUI mode.

Cavium Confidential File
Bowen Yang
ALL
02/27/2014

RX FIFO Packet-Drop Registers GMX(0..1)_RX_BP_DROP(0..3)

This register provides the mark for an RX FIFO beyond which packets are dropped and not buffered. The actual watermark is dynamic with respect to the GMX(0..1)_RX_PRTS register. GMX(0..1)_RX_PRTS controls the depth of the port's FIFO, so as ports are added or removed, the drop point may change. The relationship of the values in GMX(0..1)_RX_BP_DROP(0..3), GMX(0..1)_RX_BP_ON(0..3), and GMX(0..1)_RX_BP_OFF(0..3) is shown Figure 23–13.

Figure 23–13 RX FIFO Values



In RXAUI mode, port 0 is used for checking. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	MARK	R/W	0x2	0x2	Number of eight-byte ticks to reserve in the RX FIFO. When the FIFO exceeds this count, packets are dropped and not buffered. MARK should typically be programmed to the number of ports + 1. Failure to program correctly can lead to system instability.

RX Backpressure On Registers GMX(0..1)_RX_BP_ON(0..3)

This register provides the high watermark for port/interface backpressure, the FIFO-full level at which backpressure is asserted. In RXAUI mode, port 0 is used for checking. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8:0>	MARK	R/W	0x40	0x40	High watermark (in eight-byte ticks) for asserting backpressure. Each register is for an individual port. In RXAUI mode, port 0 is used for the unified RX FIFO. GMX_RX_BP_ON must satisfy: BP_OFF[MARK] ≤ BP_ON[MARK] < (FIFO_SIZE – BP_DROP[MARK]) A value of 0x0 immediately asserts backpressure.

Cavium Confidential For Bowen Yang ALL 6/27/2014

RX Backpressure Off Registers
GMX(0..1)_RX_BP_OFF(0..3)

This register provides the low watermark for port/interface backpressure, the FIFO-full level at which backpressure can be deasserted. In RXAUI mode, port 0 is used for checking. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	MARK	R/W	0x10	0x10	Low-water mark (in eight-byte ticks) to deassert backpressure.

TX Ports Register
GMX(0..1)_TX_PRTS

This register specifies the number of ports allowed on the interface, which is one higher than highest-numbered port (e.g. if port 3 is the highest port number, this register specifies 4). See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved
<4:0>	PRTS	R/W	0x4	—	Number of ports allowed on the interface. (SGMII/QSGMII/1000BASE-X only.)

Packet TX Interframe Gap Register
GMX(0..1)_TX_IFG

Consider the following when programming IFG1 and IFG2:

- For 10/100/1000 Mb/s half-duplex systems that require IEEE 802.3 compatibility, IFG1 must be in the range of 1–8, IFG2 must be in the range of 4–12, and the IFG1 + IFG2 sum must be 12.
- For 10/100/1000 Mb/s full-duplex systems that require IEEE 802.3 compatibility, IFG1 must be in the range of 1–11, IFG2 must be in the range of 1–11, and the IFG1 + IFG2 sum must be 12.
- For XAUI/10Gb/s systems that require IEEE 802.3 compatibility, the IFG1 + IFG2 sum must be 12. IFG1[1:0] and IFG2[1:0] must be zero.
- For all other systems, IFG1 and IFG2 can be any value in the range of 1–15, allowing for a total possible IFG sum of 2–30.

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:4>	IFG2	R/W	0x4	—	Remainder of interFrameGap timing, equal to interFrameGap – IFG1 (in IFG2 × 8 bits). If CRS is detected during IFG2, the interFrameSpacing timer is not reset and a frame is transmitted once the timer expires.
<3:0>	IFG1	R/W	0x8	—	First portion of interFrameGap timing, in the range of 0 to 2/3 (in IFG2 × 8 bits). If CRS is detected during IFG1, the interFrameSpacing timer is reset and a frame is not transmitted.

Cavium Confidential For Bowen Yang ALU 6/23/2014

Packet TX JAM Pattern Register
GMX(0..1)_TX_JAM

This register provides the pattern used in JAM bytes. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	JAM	R/W	0xEE	—	JAM pattern. (SGMII/1000Base-X half-duplex only.)

Packet TX Collision Attempts Before Dropping Frame Register
GMX(0..1)_TX_COL_ATTEMPT

This register provides the number of collision attempts allowed before dropping the frame. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved
<4:0>	LIMIT	R/W	0x10	0x10	Number of collision attempts allowed. (SGMII/1000Base-X half-duplex only.)

Packet TX Pause-Packet DMAC-Field Register
GMX(0..1)_TX_PAUSE_PKT_DMAC

This register provides the DMAC value that is placed in outbound pause packets. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	DMAC	R/W	0x0180C2000001	0x0180C2000001	The DMAC field, which is placed is outbound pause packets.

Packet TX Pause Packet TYPE Field Register
GMX(0..1)_TX_PAUSE_PKT_TYPE

This register provides the Type field that is placed in outbound pause packets. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TYPE	R/W	0x8808	0x8808	The Type field placed is outbound pause packets.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Packet Interface TX Override Backpressure Register GMX(0..1)_TX_OVR_BP

This register provides the ability to assert per-port backpressure, as well as the ability to override the assertion of backpressure for SGMII / 1000BASE-X and XAUI ports.

In RXAUI mode, only the least significant bit (corresponding to port 0) of EN, BP, and IGN_FULL are used. GMX(0..1)_TX_OVR_BP[EN<0>] must be set to 1 and GMX(0..1)_TX_OVR_BP[BP<0>] must be cleared to 0 to forcibly disable hardware-automatic 802.3 pause packet generation with the HiGig2 Protocol when GMX(0..1)_HG2_CONTROL[HG2TX_EN] = 0. (The HiGig2 protocol is indicated by GMX(0..1)_TX_XAUI_CTL[HG_EN] = 1 and GMX(0..1)_RX0_UDD_SKP[LEN] = 16.) Hardware can only autogenerate backpressure through HiGig2 messages (optionally, when GMX(0..1)_HG2_CONTROL[HG2TX_EN] = 1) with the HiGig2 protocol.

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:32>	TX_PRT_BP	R/W	0x0	0x0	Per-port backpressure sent to PKO: 0 = port is available. 1 = port should be backpressured. TX_PRT_BP should not be set until GMX_INF_MODE[EN] has been enabled
<31:12>	—	RAZ	—	—	Reserved
<11:8>	EN	R/W	0x0	0x0	Per-port backpressure enable/override (<8> for port 0, ..., <11> for port 3). Bits in this field must be set to allow backpressure to be asserted for the corresponding port. Ports that don't have the corresponding bit set are not able to assert backpressure.
<7:4>	BP	R/W	0x0	0x0	Per-port backpressure status to use: (<4> for port 0, ..., <7> for port 3) 0 = port is available 1 = port should be backpressured
<3:0>	IGN_FULL	R/W	0x0	0x0	Ignore the RX FIFO full when computing backpressure (<0> for port 0, ..., <3> for port 3).

Packet Interface TX Backpressure Status Register GMX(0..1)_TX_BP

This register shows the status of backpressure for SGMII/QSGMII/1000BASE-X and RXAUI ports. In RXAUI mode, only the least significant bit (corresponding to port 0) of EN, BP, and IGN_FULL are used. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	BP	RO	0x0	0x0	Per-port backpressure status: (<0> for port 0, ..., <3> for port 3) 0 = port is available, 1 = port should be backpressured

Cavium Confidential For Bowen Yang ALU 02/27/2014

TX Corrupt Packets Register
GMX(0..1)_TX_CORRUPT

This register enables the STX to corrupt a packet under certain conditions. Packets sent from PKO with the ERR wire asserted are corrupted by the transmitter if CORRUPT[port#] is set (RXAUI uses port# 0).

Corruption means that GMX sends a bad FCS value. If GMX(0..1)_TX(0..3)_APPEND[FCS] is clear, then no FCS is sent and the GMX cannot corrupt it. The corrupt FCS value is 0xEEEEEEEE for SGMII/QSGMII/1000BASE-X and four bytes of the error propagation code in RXAUI mode. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	CORRUPT	R/W	0xF	0xF	Per-port error propagation. 0 = Never corrupt packets 1 = Corrupt packets with ERR

RX Port State Information Register
GMX(0..1)_RX_PRT_INFO

This register reports the RX status for the port. In RXAUI mode, only the least significant bit (corresponding to port 0) of DROP and COMMIT are used. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:20>	—	RAZ	—	—	Reserved
<19:16>	DROP	RO/H	0x0	0x0	Per-port indication that data was dropped.
<15:4>	—	RAZ	—	—	Reserved
<3:0>	COMMIT	RO/H	0x0	0x0	Per-port indication that SOP was accepted.

TX LFSR Register
GMX(0..1)_TX_LFSR

This register shows the contents of the linear feedback shift register (LFSR), which is used to implement truncated binary exponential backoff. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	LFSR	R/W/H	0xFFFF	—	Contains the current state of the LFSR, which is used to feed random numbers to compute truncated binary exponential backoff. (SGMII/1000Base-X half-duplex only.)

Cavium Confidential For Bowen Yang ALU 02/27/2014

TX Interrupt Register
GMX(0..1)_TX_INT_REG

This register allows per-port interrupts to be sent to the control processor. If an error is noted in this register, and the corresponding bit in GMX(0..1)_TX_INT_EN is set, an exception is raised. In RXAUI mode, only the least significant bit (corresponding to port 0) of UNDFLW is used. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved
<24>	XCHANGE	R/W1C/H	0	0	RXAUI link status changed. This denotes a change to GMX_RX_XAUI_CTL[STATUS] (XAUI mode only).
<23:20>	PTP_LOST	R/W1C/H	0x0	0x0	A packet with a PTP request could not be sent due to XSCOL.
<19:16>	LATE_COL	R/W1C/H	0x0	0x0	TX late collision. (SGMII/1000BASE-X half-duplex only.)
<15:12>	XSDEF	R/W1C/H	0x0	0x0	TX excessive deferrals. (SGMII/1000BASE-X half-duplex only.)
<11:8>	XSCOL	R/W1C/H	0x0	0x0	TX excessive collisions. (SGMII/1000BASE-X half-duplex only.)
<7:6>	SPARE	R/W1C/H	0x0	0x0	Spare bits
<5:2>	UNDFLW	R/W1C/H	0x0	0x0	TX underflow
<1>	—	RAZ	—	—	Reserved
<0>	PKO_NXA	R/W1C/H	0	0	Port address out-of-range from PKO interface.

TX Interrupt-Enable Register
GMX(0..1)_TX_INT_EN

This register provides the interrupt-enable bits for the corresponding error in GMX(0..1)_TX_INT_REG. In RXAUI mode, only the least significant bit (corresponding to port 0) of UNDFLW is used. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved
<24>	XCHANGE	R/W	0	0	RXAUI link status changed. This denotes a change to GMX_RX_XAUI_CTL[STATUS] (XAUI mode only).
<23:20>	PTP_LOST	R/W	0x0	0x0	A packet with a PTP request could not be sent due to XSCOL.
<19:16>	LATE_COL	R/W	0x0	0x0	TX late collision. (SGMII/QSGMII/1000BASE-X only.)
<15:12>	XSDEF	R/W	0x0	0x0	TX excessive deferrals. (SGMII/QSGMII/1000BASE-X half-duplex only.)
<11:8>	XSCOL	R/W	0x0	0x0	TX excessive collisions. (SGMII/QSGMII/1000BASE-X half-duplex only.)
<7:6>	SPARE	R/W	0x0	0x0	Spare bits.
<5:2>	UNDFLW	R/W	0x0	0x0	TX Underflow
<1>	—	RAZ	—	—	Reserved
<0>	PKO_NXA	R/W	0	0	Port address out-of-range from PKO Interface

Cavium Confidential For Bowen Yang 02/12/2016

Address-out-of-Range Error Register GMX(0..1)_NXA_ADR

This register shows the logged address that caused an address-out-of-range error. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	PRT	RO/H	0x0	—	Logged address for address-out-of-range exceptions. The logged address is from the first exception that caused the problem. Ports 3:0 are valid.

GMX Miscellaneous Error Register GMX(0..1)_BAD_REG

This register contains a variety of error conditions that were not covered in other error registers. In RXAUI mode, only the least significant bit (corresponding to port 0) of INB_NXA, LOSTSTAT, OUT_OVR, are used. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:31>	—	RAZ	—	—	Reserved
<30:27>	INB_NXA	R/W1C/H	0x0	0x0	Inbound port > GMX(0..1)_RX_PRTS
<26>	STATOVR	R/W1C/H	0	0	TX statistics overflow. The common FIFO to SGMII/QSGMII and RXAUI had an overflow. TX statistics are corrupted.
<25:22>	LOSTSTAT	R/W1C/H	0x0	0x0	TX statistics data was over-written. In SGMII mode, one bit per port. In RXAUI mode, only port 0 is used. TX statistics are corrupted.
<21:6>	—	RAZ	0x0	0x0	Reserved
<5:2>	OUT_OVR	R/W1C/H	0x0	0x0	Outbound data FIFO overflow (per port).
<1:0>	—	RAZ	—	—	Reserved

GMX Backpressure Statistics Register GMX(0..1)_STAT_BP

This register indicates the number of cycles that the TX/Stats block has held up operation. See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:17>	—	RAZ	—	—	Reserved
<16>	BP	RO	0	0	When the TX stats machine cannot update the stats registers quickly enough, the machine has the ability to backpressure TX datapath. This is a rare event and will not occur in normal operation. 0 = No backpressure is applied. 1 = Backpressure is applied to TX datapath to allow stat update operations to complete.
<15:0>	CNT	R/W1C	0x0	0x0	Number of cycles that backpressure has been asserted, saturating counter.

Cavium Confidential For Bowen Yang ALU 02/27/2014

QSGMII Control Register
GMX(0..1)_QSGMII_CTL

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	DISPARITY	R/W	0	0	When set, enables the running disparity check at the receiver. Due to the nature of QSGMII, bit errors on the link may cause a running disparity error to propagate across ports. Disabling running disparity checking at the receiver prevents error propagation to other ports.

Wake-On-LAN Control Register
GMX(0..1)_WOL_CTL

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:36>	—	RAZ	—	—	Reserved.
<35:32>	MAGIC_EN	R/W	0x0	—	Port enable for magic-packet WOL event detection.
<31:20>	—	RAZ	—	—	Reserved.
<19:16>	DIRECT_EN	R/W	0x0	—	Port enable for directed-packet WOL event detection.
<15:1>	—	RAZ	—	—	Reserved.
<0>	EN	R/W	0	—	Enter WOL mode.

Interface Mode Register
GMX(0..1)_INF_MODE

See Table 23–9 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:4>	MODE	R/W	0	—	Interface operating mode. Enumerated by GMX_INF_MODE_E. This field should be written before any of the DLM registers are written and the EN is asserted.
<3:2>	—	RAZ	—	—	Reserved
<1>	EN	R/W	0	—	Interface enable. This field must be set to enable the packet interface. It should be enabled before any other requests to GMX, including enabling port backpressure with IPD_CTL_STATUS[PBP_EN].
<0>	—	RAZ	—	—	Reserved

Cavium Confidential For Nowen Yang ALU 02/27/2014

23.12 PCS Registers

The PCS registers are listed in Table 23–11.

Table 23–11 PCS Registers

Register	Address[1]	CSR Type[2]	Detailed Description
PCS(0..1)_MR(0..3)_CONTROL_REG	0x11800B0001000 + α*0x8000000 + β*0x400	RSL	See page 1084
PCS(0..1)_MR(0..3)_STATUS_REG	0x11800B0001008 + α*0x8000000 + β*0x400	RSL	See page 1085
PCS(0..1)_AN(0..3)_ADV_REG	0x11800B0001010 + α*0x8000000 + β*0x400	RSL	See page 1086
PCS(0..1)_AN(0..3)_LP_ABIL_REG	0x11800B0001018 + α*0x8000000 + β*0x400	RSL	See page 1086
PCS(0..1)_AN(0..3)_RESULTS_REG	0x11800B0001020 + α*0x8000000 + β*0x400	RSL	See page 1087
PCS(0..1)_AN(0..3)_EXT_ST_REG	0x11800B0001028 + α*0x8000000 + β*0x400	RSL	See page 1087
PCS(0..1)_LINK(0..3)_TIMER_COUNT_REG	0x11800B0001040 + α*0x8000000 + β*0x400	RSL	See page 1087
PCS(0..1)_TX_RX(0..3)_POLARITY_REG	0x11800B0001048 + α*0x8000000 + β*0x400	RSL	See page 1088
PCS(0..1)_RX(0..3)_SYNC_REG	0x11800B0001050 + α*0x8000000 + β*0x400	RSL	See page 1088
PCS(0..1)_RX(0..3)_STATES_REG	0x11800B0001058 + α*0x8000000 + β*0x400	RSL	See page 1088
PCS(0..1)_TX(0..3)_STATES_REG	0x11800B0001060 + α*0x8000000 + β*0x400	RSL	See page 1089
PCS(0..1)_SGM(0..3)_AN_ADV_REG	0x11800B0001068 + α*0x8000000 + β*0x400	RSL	See page 1089
PCS(0..1)_SGM(0..3)_LP_ADV_REG	0x11800B0001070 + α*0x8000000 + β*0x400	RSL	See page 1090
PCS(0..1)_MISC(0..3)_CTL_REG	0x11800B0001078 + α*0x8000000 + β*0x400	RSL	See page 1090
PCS(0..1)_INT(0..3)_REG	0x11800B0001080 + α*0x8000000 + β*0x400	RSL	See page 1091
PCS(0..1)_INT(0..3)_EN_REG	0x11800B0001088 + α*0x8000000 + β*0x400	RSL	See page 1092
PCS(0..1)_LOG_ANL(0..3)_REG	0x11800B0001090 + α*0x8000000 + β*0x400	RSL	See page 1092
PCS(0..1)_SERDES_CRDT_CNT(0..3)_REG	0x11800B00010A0 + α*0x8000000 + β*0x400	RSL	See page 1092
PCS(0..1)_MAC_CRDT_CNT(0..3)_REG	0x11800B00010B0 + α*0x8000000 + β*0x400	RSL	See page 1092

1. For registers with multiple versions, α represents the values within the specified range and β the second range.
2. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang ALU 6/21/2014

PCS Control Register 0
PCS(0..1)_MR(0..3)_CONTROL_REG

When PCS(0..1)_MR(0..3)_CONTROL_REG[AN_EN] is set, autonegotiation is allowed to happen. The results of the autonegotiation process set the fields in PCS(0..1)_AN(0..3)_RESULTS_REG. When [AN_EN] is not set, PCS(0..1)_AN(0..3)_RESULTS_REG is not valid. The values in the fields [SPDLSB], [DUP], and [SPDLSB] are from PCS(0..1)_MR(0..3)_CONTROL_REG.

See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	RESET	R/W/H	0	0	Set to initiate software reset. This bit returns to 0 after PCS has been reset. Takes 32 coprocessor-clock cycles to reset PCS.
<14>	LOOPBACK1	R/W	0	0	Set to enable loopback. 0 = normal operation, 1 = loopback. Loopback mode returns (loopback) TX data from GMII TX back to the GMII RX interface. The loopback happens in the PCS module. Autonegotiation is disabled during loopback, even if the AN_EN bit is set.
<13>	SPDLSB	R/W	0	0	Link speed effective only if autonegotiation is disabled as follows: 00 = 10Mb/s 10 = 1000Mb/s 01 = 100Mb/s 11 = reserved
<12>	AN_EN	R/W	0	0	Enable autonegotiation. 1 = enable, 0 = disable.
<11>	PWR_DN	R/W	1	0	Power down: 1 = power down (hardware reset), 0 = normal operation.
<10>	—	RAZ	—	—	Reserved
<9>	RST_AN	R/W/H	0	0	Autonegotiation restart. If bit <12> = 1 and PCS(0..1)_MR(0..3)_STATUS_REG[AN_ABIL] = 1, autonegotiation begins. Otherwise, software writes are ignored and this bit remains at 0. This bit clears itself to 0 when AN starts.
<8>	DUP	R/W	1	1	This field is effective only if autonegotiation is disabled. 0 = half duplex 1 = full duplex If PCS(0..1)_MR(0..3)_STATUS_REG bits <15:9> and PCS(0..1)_MR(0..3)_EXT_ST_REG bits <15:12> allow only one duplex mode, this bit corresponds to that value and any attempt to write are ignored.
<7>	COLTST	R/W	0	0	COL test enable. 0 = disable COL signal test, 1 = enable COL signal test. During COL test, the COL signal reflects the GMII TX_EN signal with less than 16BT delay.
<6>	SPDMSB	R/W	1	1	Link speed effective only if autonegotiation is disabled as follows: 00 = 10Mb/s 10 = 1000Mb/s 01 = 100Mb/s 11 = reserved
<5>	UNI	R/W	0	0	Unidirectional. (IEEE Std 802.3-2005, Clause 66.2) When set, this bit overrides the AN_EN bit and disables the autonegotiation variable mr_an_enable. Used in 1000BASE-X, SGMII, and QSGMII modes.
<4:0>	—	RAZ	—	—	Reserved

Cavium Confidential For ALE Power Yang 02/12/2014

PCS Status Register 1
PCS(0..1)_MR(0..3)_STATUS_REG

Bits <15:9> in this register indicate the ability to operate when PCS(0..1)_MISC(0..3)_CTL_REG[MAC_PHY] is set to MAC mode. Bits <15:9> are always read as 0, indicating that the chip cannot operate in the corresponding modes. The field [RM_FLT] is a "don't care" when the selected mode is SGMII or QSGMII. See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	HUN_T4	RO	0	0	When set, 100 BASE-T4 capable.
<14>	HUN_XFD	RO	0	0	When set, 100 BASE-X Full Duplex.
<13>	HUN_XHD	RO	0	0	When set, 100 BASE-X Half Duplex.
<12>	TEN_FD	RO	0	0	When set, 10 Mb/s Full Duplex.
<11>	TEN_HD	RO	0	0	When set, 10 Mb/s Half Duplex.
<10>	HUN_T2FD	RO	0	0	When set, 100 BASE-T2 Full Duplex.
<9>	HUN_T2HD	RO	0	0	When set, 100 BASE-T2 Half Duplex.
<8>	EXT_ST	RO	1	1	When set, extended status info in reg15.
<7>	—	RAZ	—	—	Reserved
<6>	PRB_SUP	RO	1	1	Preamble suppression. 0 = unable to accept frames without preamble bytes preceding them. 1 = able to work without preamble bytes at the beginning of frames.
<5>	AN_CPT	RO	0	1	When set, autonegotiation is complete and the contents of PCS(0..1)_AN(0..3)_RESULTS_REG are valid.
<4>	RM_FLT	RO	0	0	When set, remote fault condition detected. This bit implements a latching-high behavior. It is cleared by a software read operation of this register or when reset PCS(0..1)_MR(0..3)_CONTROL_REG[RESET] is asserted. PCS(0..1)_AN(0..3)_ADV_REG[PAUSE] lists the fault conditions.
<3>	AN_ABIL	RO	1	1	When set, autonegotiation capable.
<2>	LNK_ST	RO	0	1	Link state as follows: 0 = link down, 1 = link up. Set during autonegotiation process. Set whenever XMIT=DATA. Latching Lo behavior when link goes down. Link down value of the bit stays low until software reads the register.
<1>	—	RAZ	—	—	Reserved
<0>	EXTND	RO	0	0	This field is always 0, extended capability registers not present.

Cavium Confidential For Bowen Yang ALD 02/27/2014

PCS Autonegotiation Advertisement Register PCS(0..1)_AN(0..3)_ADV_REG

See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	NP	RO/H	0	0	Next page capable. This feature is not supported; this field is always 0.
<14>	—	RAZ	—	—	Reserved
<13:12>	REM_FLT	R/W	0x0	0x0	00 = link OK, XMIT=DATA 01 = link failure (loss of sync, XMIT≠DATA) 10 = local device offline 11 = autonegotiation error; failure to complete AN. AN error is set if resolution function precludes operation with link partner.
<11:9>	—	RAZ	—	—	Reserved
<8:7>	PAUSE	R/W	0x0	0x0	Pause frame flow capability across link, exchanged during autonegotiation as follows: 00 = no pause. 01 = symmetric pause. 10 = asymmetric pause. 11 = both symmetric and asymmetric pause to local device.
<6>	HFD	R/W	1	1	When set, local device is half-duplex capable.
<5>	FD	R/W	1	1	When set, local device is full-duplex capable.
<4:0>	—	RAZ	—	—	Reserved

PCS Autonegotiation Link-Partner Ability Register PCS(0..1)_AN(0..3)_LP_ABIL_REG

This is the autonegotiation Link partner ability register 5 as per IEEE 802.3, Clause 37. See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	NP	RO	0	0	Next page capable. 1 = link partner is next page capable. 0 = link partner not next page capable.
<14>	ACK	RO	0	—	When set, indicates acknowledgement received.
<13:12>	REM_FLT	RO	0x0	0x0	Link partner's link status as follows: 00 = link OK 10 = link failure 01 = offline 11 = autonegotiation error
<11:9>	—	RAZ	—	—	Reserved
<8:7>	PAUSE	RO	0x0	0x0	Link partner pause setting as follows: 00 = no pause. 01 = symmetric pause. 10 = asymmetric pause. 11 = both symm and asymm pause to local device.
<6>	HFD	RO	0	0	When set, link partner is half-duplex capable.
<5>	FD	RO	0	0	When set, link partner is full-duplex capable.
<4:0>	—	RAZ	—	—	Reserved

Cavium Confidential For Bowen Yang ALU 2014

PCS Autonegotiation Results Register
PCS(0..1)_AN(0..3)_RESULTS_REG

This register is not valid when PCS(0..1)_MR(0..3)_CONTROL_REG[AN_OVRD] is set to 1. If PCS(0..1)_MR(0..3)_CONTROL_REG[AN_OVRD] is set to 0 and PCS(0..1)_AN(0..3)_RESULTS_REG[AN_CPT] is set to 1, this register is valid.

See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved
<6:5>	PAUSE	RO	0x0	0x0	Pause selection (“don’t care” for SGMII/QSGMII) as follows: 00 = disable pause, both TX and RX 01 = enable pause frames, RX only. 10 = enable pause frames, TX only. 11 = enable pause frames, TX and RX.
<4:3>	SPD	RO	0x0	0x0	Link speed selection as follows: 00 = 10 Mb/s 10 = 1000 Mb/s 01 = 100 Mb/s 11 = reserved
<2>	AN_CPT	RO	0	1	Autonegotiation completed. 0 = AN not completed or failed, 1 = AN completed.
<1>	DUP	RO	0	0	Duplex mode. 0=half duplex, 1=full duplex.
<0>	LINK_OK	RO	0	0	0 = link down, 1 = link up (OK).

PCS Autonegotiation Extended Status Register 15
PCS(0..1)_AN(0..3)_EXT_ST_REG

This is the autonegotiation extended status register 15 as per IEEE 802.3, Clause 22. See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	THOU_XFD	RO/H	1	1	When set, PHY is 1000 BASE-X full duplex capable.
<14>	THOU_XHD	RO/H	1	1	When set, PHY is 1000 BASE-X half duplex capable.
<13>	THOU_TFD	RO/H	0	0	When set, PHY is 1000 BASE-T full duplex capable.
<12>	THOU_THD	RO/H	0	0	When set, PHY is 1000 BASE-T half duplex capable.
<11:0>	—	RAZ	—	—	Reserved

PCS Link Timer Register
PCS(0..1)_LINK(0..3)_TIMER_COUNT_REG

This is the 1.6 ms nominal Link timer register. See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	COUNT	R/W	0x446	—	(Coprocessor clock period × 1024) × COUNT should be 1.6 ms for SGMII/QSGMII) and 10 ms otherwise, which is the link timer used in autonegotiation. Reset assumes a 700 MHz coprocessor clock for 1.6 ms link timer.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PCS TX_RX Polarity Register
PCS(0..1)_TX_RX(0..3)_POLARITY_REG

See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	RXOVRD	R/W	0	0	0 = AUTORXPL determines polarity 1 = RXPLRT determines polarity
<2>	AUTORXPL	RO/H	0	0	Auto RX polarity detected: 0 = normal 1 = inverted This bit always represents the correct RX polarity setting needed for successful RX path operation, once a successful code group sync is obtained.
<1>	RXPLRT	R/W	0	0	0 = normal polarity. 1 = inverted polarity.
<0>	TXPLRT	R/W	0	0	0 = normal polarity. 1 = inverted polarity.

PCS Code Group Synchronization Register
PCS(0..1)_RX(0..3)_SYNC_REG

See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	SYNC	RO	0	0	When set, code group synchronization achieved
<0>	BIT_LOCK	RO	0	0	When set, bit lock achieved.

PCS RX State Machines States Register
PCS(0..1)_RX(0..3)_STATES_REG

See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	RX_BAD	RO	0	0	Receive state machine is in an illegal state.
<14:10>	RX_ST	RO/H	0x0	0x0	Receive state machine state.
<9>	SYNC_BAD	RO	0	0	Receive synchronization state machine is in an illegal state.
<8:5>	SYNC	RO/H	0x0	0x0	Receive synchronization state machine state.
<4>	AN_BAD	RO	0	0	Autonegotiation state machine is in an illegal state.
<3:0>	AN_ST	RO/H	0x0	0x0	Autonegotiation state machine state.

Cavium Confidential For Bowen Yang ALU 2/27/2014

PCS TX State Machines States Register PCS(0..1)_TX(0..3)_STATES_REG

See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved.
<6:5>	XMIT	RO/H	0x0	—	0x0 = undefined 0x2 = idle 0x1 = config 0x3 = data
<4>	TX_BAD	RO	0	0	Transmit state machine in an illegal state.
<3:0>	ORD_ST	RO/H	0x0	0x0	Transmit ordered set state machine state.

PCS SGMII Autonegotiation Advertisement Register PCS(0..1)_SGM(0..3)_AN_ADV_REG

This is the SGMII/QSGMII autonegotiation advertisement register (sent out as TX_CONFIG_REG). This register is sent during autonegotiation if PCS(0..1)_MISC(0..3)_CTL_REG[MAC_PHY] is set to 1 (PHY mode). If the bit is not set (0 = MAC mode), the tx_Config_Reg[14] becomes ACK bit and [0] is always 1. All other bits in tx_Config_Reg sent will be 0. The PHY dictates the autonegotiation results. See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	LINK	R/W	0	1	Link status: 0 = link down. 1 = link up.
<14>	ACK	RO	0	0	Autonegotiation acknowledgement.
<13>	—	RAZ	—	—	Reserved
<12>	DUP	R/W	1	1	Duplex mode: 0 = half duplex. 1 = full duplex.
<11:10>	SPEED	R/W	0x2	0x2	Link speed: 00 = 10 Mb/s 10 = 1000 Mb/s 01 = 100 Mb/s 11 = reserved
<9:1>	—	RAZ	—	—	Reserved
<0>	ONE	RO	1	1	Always set to match TX_CONFIG_REG<0>

Cavium Confidential For Bowen Yang ALU 02/27/2014

PCS SGMII Link-Partner-Advertisement Register PCS(0..1)_SGM(0..3)_LP_ADV_REG

This is the SGMII Link partner advertisement register (received as RX_CONFIG_REG). See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	LINK	RO	0	1	Link status: 0 = link down. 1 = link up.
<14>	—	RO	0	—	Reserved for autonegotiation ACK.
<13>	—	RAZ	0	—	Reserved
<12>	DUP	RO	0	1	Duplex mode: 0 = half duplex. 1 = full duplex.
<11:10>	SPEED	RO	0x0	0x2	Link speed: 00 = 10 Mb/s 10 = 1000 Mb/s 01 = 100 Mb/s 11 = reserved
<9:1>	—	RAZ	—	—	Reserved
<0>	ONE	RO	1	1	Always set to match TX_CONFIG_REG<0>

SGMII Miscellaneous Control Register PCS(0..1)_MISC(0..3)_CTL_REG

See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved
<11>	GMXENO	R/W	0	0	GMX enable override. When set to 1, forces GMX to appear disabled. The enable/disable status of GMX is checked only at SOP of every packet.
<10>	LOOPBCK2	R/W	0	0	Sets external loopback mode to return RX data back out via the TX data path. 0 = no loopback, 1 = loopback. **NOTE:** LOOPBCK1 and LOOPBCK2 modes may not be supported simultaneously.
<9>	MAC_PHY	R/W	0	0	MAC/PHY. 0 = MAC, 1 = PHY decides the TX_CONFIG_REG value to be sent during autonegotiation.
<8>	MODE	R/W	0	0	Mode bit. 0 = SGMII/QSGMII, 1 = 1000BASE-X. When MODE = 1, 1000BASE-X mode is selected. Autonegotiation follows IEEE 802.3 clause 37. When MODE = 0, SGMII/QSGMII mode is selected and the following note applies. **NOTE:** The SGMII/QSGMII AN advertisement register (PCS(0..1)_SGM(0..3)_AN_ADV_REG) is sent during autonegotiation if the PCS(0..1)_MISC(0..3)_CTL_REG[MAC_PHY] is set (1 = PHY mode). If the bit is not set (0 = MAC mode), the TX_CONFIG_REG[14] becomes ACK bit and [0] is always 1. All other bits in TX_CONFIG_REG sent are 0. The PHY dictates the autonegotiation results.
<7>	AN_OVRD	R/W	0x0	0x0	Autonegotiation override. 0 = disable, 1 = enable override AN results. Autonegotiation is allowed to happen but the results are ignored when set. Duplex and Link speed values are set from PCS(0..1)_MR(0..3)_CONTROL_REG.

Cavium Confidential For Bowen Yang 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<6:0>	SAMP_PT	R/W	0x1	—	Byte number in elongated frames for 10/100Mb/s operation for data sampling on RX side in PCS. Recommended values are 0x5 for 100Mb/s operation and 0x32 for 10Mb/s operation. For 10Mb/s operation, this field should be set to a value less than 99 and greater than 0. If set out of this range, a value of 50 is used for actual sampling internally without affecting the CSR field. For 100Mb/s operation this field should be set to a value less than 9 and greater than 0. If set out of this range, a value of 5 is used for actual sampling internally without affecting the CSR field.

PCS Interrupt Register
PCS(0..1)_INT(0..3)_REG

See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	—	—	Reserved
<12>	DBG_SYNC	R/W1C/H	0	0	Code group sync failure debug help. DBG_SYNC interrupt fires when code group synchronization state machine makes a transition from SYNC_ACQUIRED_1 state to SYNC_ACQUIRED_2 state. (See IEEE 802.3-2005, figure 37–9). It indicates that a bad code group was received after code group synchronization was achieved. This interrupt should be disabled during normal link operation. Use it as a debug help feature only.
<11>	DUP	R/W1C/H	0	0	Set whenever duplex mode changes on the link.
<10>	SYNC_BAD	R/W1C/H	0	0	Set by hardware whenever RX sync state machine reaches a bad state. Should never be set during normal operation.
<9>	AN_BAD	R/W1C/H	0	0	Set by hardware whenever autonegotiation state machine reaches a bad state. Should never be set during normal operation.
<8>	RXLOCK	R/W1C/H	0	0	Set by hardware whenever code group sync or bit lock failure occurs. Cannot fire in loopback1 mode.
<7>	RXBAD	R/W1C/H	0	0	Set by hardware whenever RX state machine reaches a bad state. Should never be set during normal operation.
<6>	RXERR	R/W1C/H	0	0	Set whenever RX receives a code group error in 10-bit to 8-bit decode logic. Cannot fire in loopback1 mode.
<5>	TXBAD	R/W1C/H	0	0	Set by hardware whenever TX state machine reaches a bad state. Should never be set during normal operation.
<4>	TXFIFO	R/W1C/H	0	0	Set whenever hardware detects a TX FIFO overflow condition.
<3>	TXFIFU	R/W1C/H	0	0	Set whenever hardware detects a TX FIFO underflow condition.
<2>	AN_ERR	R/W1C/H	0	0	Autonegotiation error; AN resolution function failed.
<1>	XMIT	R/W1C/H	0	0	Set whenever hardware detects a change in the XMIT variable. XMIT variable states are IDLE, CONFIG and DATA.
<0>	LNKSPD	R/W1C/H	0	0	Set by hardware whenever link speed has changed.

Cavium Confidential For Power Xaui 9.1 Pcs4

PCS Interrupt Enable Register
PCS(0..1)_INT(0..3)_EN_REG

See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	—	—	Reserved
<12>	DBG_SYNC_EN	R/W	0	0	Enable code group sync failure debug help.
<11>	DUP	R/W	0	1	Enable duplex mode changed interrupt.
<10>	SYNC_BAD_EN	R/W	0	1	Enable RX sync state machine in bad state interrupt.
<9>	AN_BAD_EN	R/W	0	1	Enable autonegotiation state machine bad state interrupt.
<8>	RXLOCK_EN	R/W	0	1	Enable RX code group sync/bit lock failure interrupt.
<7>	RXBAD_EN	R/W	0	1	Enable RX state machine in bad state interrupt.
<6>	RXERR_EN	R/W	0	1	Enable RX error condition interrupt.
<5>	TXBAD_EN	R/W	0	1	Enable TX state machine in bad state interrupt.
<4>	TXFIFO_EN	R/W	0	1	Enable TX FIFO overflow condition interrupt.
<3>	TXFIFU_EN	R/W	0	1	Enable TX FIFO underflow condition interrupt.
<2>	AN_ERR_EN	R/W	0	1	Enable autonegotiation error condition interrupt.
<1>	XMIT_EN	R/W	0	1	Enable XMIT variable state change interrupt.
<0>	LNKSPD_EN	R/W	0	1	Enable link speed has changed interrupt.

PCS Logic Analyzer Register
PCS(0..1)_LOG_ANL(0..3)_REG

The logic analyzer is enabled with PCS(0..1)_LOG_ANL(0..3)_REG[LA_EN] for the specified PCS lane only. PCS(0..1)_LOG_ANL(0..3)_REG[PKT_SZ] is effective only when [LA_EN] = 1. For normal operation (SGMII, QSGMII, or 1000BASE-X), this bit must be 0. See also PCSX0_LOG_ANL_REG. See Table 23–11 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	LAFIFOVFL	R/W1C/H	0	0	Logic analyzer FIFO overflow. 1=logic analyses FIFO overflowed during packetization. Write 1 to clear this bit.
<2>	LA_EN	R/W/H	0	0	Logic analyzer enable. 1= Logic analyzer enabled, 0=Logic analyzer disabled.
<1:0>	PKT_SZ	R/W/H	0x0	0x0	Logic analyzer packet size. Bit <1>, Bit <0> 0 0 packet size 1k bytes 0 1 packet size 4k bytes 1 0 packet size 8k bytes 1 1 packet size 16k bytes.

Cavium Confidential For Bowen Yang ALU 01/27/2014

Chapter 24

Reduced 10 Gigabit Ethernet Attachment Unit Interface (RXAUI)

This chapter contains the following subjects:

- Introduction
- GMX Packet Interface Buffering
- RXAUI Features
- Errors/Exceptions
- Reset, Initialization, and Link Status Changes
- Statistics
- Loopback
- Enumerations
- GMX Registers
- PCSX Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

CN71XX implements a single reduced XAUI (RXAUI) SerDes interface in a two-lane packet interface. The interface consists of eight external pins in total: two differential output pairs plus two differential input pairs.

CN71XX couples logic that implements the RXAUI interface/protocols on SerDes lanes with an IEEE 802.3-2005 MAC.

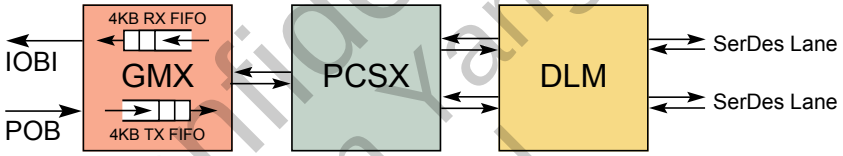
The remainder of this chapter assumes GMX(0)_INF_MODE[MODE] = RXAUI, indicating that the interface is in RXAUI mode. If GMX(0)_INF_MODE[MODE] ≠ RXAUI, the CN71XX RXAUI SerDes logic is not operational.

24.1 Introduction

Figure 24–1 is a block-diagram showing the CN71XX packet interface.

- **GMX** implements the 802.3 MAC and includes 4KB receive buffers and 4KB transmit buffers. GMX sends packet input data to PIP/IPD (Chapter 9) and receives packet output data from PKO (Chapter 10). GMX also implements the reconciliation sublayer described in Clause 46 of the IEEE 802.3-2005 specification.
- **PCSX** implements the XGXS/PCS sublayer described in Clause 47 and 48 of the IEEE 802.3-2005 specification. PCSX translates between the MAC and the SerDes lanes.
- CN71XX packages two SerDes lanes into a **dual-lane module** (**DLM**).

Figure 24–1 CN71XX Packet Interfaces Block Diagram



CN71XX contains one SerDes DLM, containing two full-duplex SerDes channels.

24.2 GMX Packet Interface Buffering

The CN71XX packet interface contains a 4KB receive and a 4KB transmit staging buffer.

In the output case, PKO (described in Chapter 10) has much buffering to add to this packet-interface buffering. More importantly, the CN71XX hardware buffers packet data in L2/DRAM, and this buffering can be many times larger than the packet interface and PKO buffering combined.

In the input case, it is possible for PIP/IPD (described in Chapter 9) to exhaust all of the available L2/DRAM buffering, leaving the packet interface as the only buffering option for input packets. But most CN71XX applications use packet drop and/or per-port backpressure to prevent buffer exhaustion (refer to Sections 9.6 and 9.7). CN71XX can backpressure (via pause packets) to avoid overflow of the input buffer when its usage exceeds programmable on/off thresholds (refer to Section 24.3.2.4).

Cavium Confidential For Bowen Yang ALU 02/27/2014

24.3 RXAUI Features

This section discusses notable features provided by the CN71XX GMX/PCSX architecture, some of which being outside the IEEE 802.3 specifications.

24.3.1 RXAUI Lane Polarity and Swapping

CN71XX PCSX logic can swap the SerDes lanes and control the SerDes polarity. Board routing can be simplified in some cases if the four lanes that implement RXAUI and/or two wires that implement a differential pair are swapped. The lane swap switches lane 0 with lane 3, and lane 1 with lane 2.

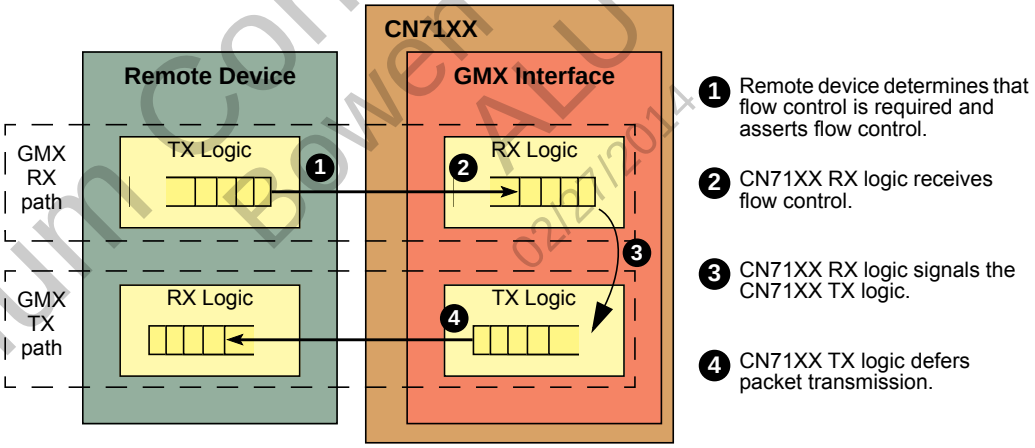
- On the transmit side, PCSX(0)_MISC_CTL_REG[TX_SWAP] optionally swaps the SerDes lanes, and PCSX(0)_TX_RX_POLARITY_REG[TXPLRT, XOR_TXPLRT] optionally inverts the SerDes differential output pairs. CN71XX inverts the polarity of pair/lane *i* when PCSX(0)_TX_RX_POLARITY_REG[TXPLRT] ⊕ PCSX(0)_TX_RX_POLARITY_REG[XOR_TXPLRT<*i*>] = 1.
- On the receive side, PCSX(0)_MISC_CTL_REG[RX_SWAP] optionally swaps the SerDes lanes, and PCSX(0)_TX_RX_POLARITY_REG[RXPLRT, XOR_RXPLRT] optionally inverts the SerDes differential input pairs. CN71XX inverts the polarity of pair/lane *i* when PCSX(0)_TX_RX_POLARITY_REG[RXPLRT] ⊕ PCSX(0)_TX_RX_POLARITY_REG[XOR_RXPLRT<*i*>] = 1.

24.3.2 Flow Control

The following subsections discuss how the CN71XX receives and transmits flow control.

Remote-device flow control is shown in Figure 23–2.

Figure 24–2 Remote-Device Flow Control



Cavium Confidential For Bowen Yang ALU 02/27/2014

24.3.2.1 Flow Control From Received IEEE 802.3 Pause Packets

When the receive interface in full-duplex mode, the CN71XX can be configured to receive pause control packets as defined in Annexes 31A and 31B of the 802.3 specification based on the GMX(0)_RX(0..3)_FRM_CTL[CTL_BCK/CTL_DRP] registers. Common modes of operation are:

[CTL_BCK] = 1, [CTL_DRP] = 1 — CN71XX inspects pause packets, sets the transmit defer counter to the TIME parameter in the packets, and then drops the pause packets. CN71XX then defers the transmission of new packets until the timer expires.

[CTL_BCK] = 0, [CTL_DRP] = 0 — CN71XX processes pause packets like normal packets, so software will see them. Software can optionally implement a backpressure scheme. Software can also set/reset the defer counter by writing to GMX(0)_TX(0..3)_SOFT_PAUSE[TIME]. Like pause frames, software must periodically write to GMX(0)_TX(0..3)_SOFT_PAUSE[TIME] if the flow condition extends beyond a TIME interval.
CN71XX asserts GMX(0)_RX(0..3)_INT_REG[PAUSE_DRP] whenever it drops a pause packet due to filling the receive FIFO in this mode.

[CTL_BCK] = 0, [CTL_DRP] = 1 — All pause frames are dropped and completely ignored by both hardware and software.

[CTL_BCK] = 1, [CTL_DRP] = 0 — CN71XX sends pause packets to software like normal packets, and sets the transmit-defer counter to the TIME parameter in the packet. CN71XX then defers the transmission of new outbound packets until the timer expires. As software can see all pause frames, it can optionally implement a backpressure scheme. Software can also set/reset the defer counter by writing to GMX(0)_TX(0..3)_SOFT_PAUSE[TIME]. Like pause frames, software must periodically write to GMX(0)_TX(0..3)_SOFT_PAUSE[TIME] if the flow condition extends beyond a TIME interval.
CN71XX asserts GMX(0)_RX(0..3)_INT_REG[PAUSE_DRP] whenever it drops a pause packet due to filling the receive FIFO in this mode.

Note that the transmit defer counter is not set if FCS checking is enabled (GMX(0)_RX(0..3)_FRM_CHK[FCSERR]) and the pause packet failed the FCS/CRC check.

If PREAMBLE/SFD is present (i.e. GMX(0)_RX(0..3)_FRM_CTL[PRE_CHK] = 1, refer to Section 24.3.4.1) and/or if user-defined data is present (i.e. GMX(0)_RX(0..3)_UDD_SKP[LEN] ≠ 0, refer to Sections 24.3.4.2 and 24.3.4.3) in a received packet, CN71XX discards them before it determines whether a packet is a pause frame. CN71XX recognizes the remaining packet bytes as a pause frame if all these conditions are true:

- either CTL_BCK or CTL_DRP are set
- the first six bytes of the packet (i.e. the L2 destination address) match one of these two enabled comparisons:
  - GMX(0)_RX(0..3)_FRM_CTL[CTL_MCST] = 1 and the first 6 bytes equal the standard multicast address 0x01-80-C2-00-00-01

Cavium Confidential For Broadcom Use Only 02/27/2014

- GMX(0)_RX(0..3)_FRM_CTL[CTL_SMAC] = 1 and the first six bytes equal GMX(0)_SMAC0[SMAC]

- the 13th and 14th bytes (i.e. the L2 type field) equal 0x88-08 (MAC control), and
- the 15th and 16th bytes (i.e. the MAC control opcode) equal 0x00-01 (PAUSE)

Note that PREAMBLE/SFD should normally be present in received pause frames (and GMX(0)_RX(0..3)_FRM_CTL[PRE_CHK] should be set) whenever either CTL_BCK or CTL_DRP are set.

24.3.2.2 Flow Control From Received PFC Packets

There is a proposed specification for priority-based flow control (PFC), formerly known as class-based flow control (CBFC). (The current revision, titled "Proposal for Priority Based Flow Control," is in revision *bb-pelissier-pfc-proposal-0508.pdf*, and is authored by Hugh Barrass from Cisco Systems, et al. It is currently located at http://www.ieee802.org/1/files/public/docs2008/bb-pelissier-pfc-proposal-0508.pdf.) CN71XX supports this specification in RXAUI mode. Furthermore, CN71XX extends the PFC definition. The current specification revision supports only an eight-bit class-enable vector (though the vector field is 16 bits) and eight pause timers (though the packet can easily accommodate 16 pause timers). CN71XX supports the full 16-bit class-enable vector and 16 corresponding pause timers.

When GMX(0)_PRT0_CBFC_CTL[RX_EN] = 1, CN71XX recognizes and can respond to (and accumulate statistics for) PFC packets. When GMX(0)_PRT0_CBFC_CTL[RX_EN] = 0, CN71XX never treats PFC packets as control/pause packets. (HiGig2 messages should not be simultaneously present with PFC frames.)

When CN71XX recognizes PFC packets, GMX(0)_PRT0_CBFC_CTL[BCK_EN,DRP_EN] control how CN71XX handles PFC packets:

[BCK_EN] = 1, [DRP_EN] = 1	CN71XX inspects PFC packets, updates backpressure state, and then drops the pause packets. CN71XX then defers the transmission of new packets based on the backpressure state.
[BCK_EN] = 0, [DRP_EN] = 0	CN71XX processes PFC packets like normal packets, so software will see them. Software can optionally implement a backpressure scheme using either GMX(0)_TX_OVR_BP[TX_PRT_BP] or PKO_MEM_QUEUE_QOS[QOS_MASK]. CN71XX asserts GMX(0)_RX(0..3)_INT_REG[PAUSE_DRP] whenever it drops a pause packet due to filling the receive FIFO in this mode.
[BCK_EN] = 0, [DRP_EN] = 1	All PFC frames are dropped and completely ignored by both hardware and software.

Cavium Confidential For Bowen Yang @ALU 2014/12/14

[BCK_EN] = 1, [DRP_EN] = 0	CN71XX sends pause packets to software like normal packets and updates backpressure state. CN71XX then defers the transmission of new packets based on the backpressure state. Software can also force additional backpressure via either GMX(0)_TX_OVR_BP[TX_PRT_BP] or PKO_MEM_QUEUE_QOS[QOS_MASK]. CN71XX asserts GMX(0)_RX(0..3)_INT_REG[PAUSE_DRP] whenever it drops a pause packet due to filling the receive FIFO in this mode.

CN71XX implements PFC from received packets using a single transmit defer counter (GMX(0)_RX_HG2_STATUS[LGTIM2GO]) rather than 8/16 separate counters. CN71XX sets this defer counter with a large enough value to cover the maximum possible interval. This means that CN71XX may backpressure a given class/port longer than the minimum requested by the PFC packet. GMX(0)_RX_HG2_STATUS[XOF] indicates which classes/ports are currently backpressured due to the defer counter.

In detail, CN71XX updates GMX(0)_RX_HG2_STATUS[LGTIM2GO] and GMX(0)_RX_HG2_STATUS[XOF] upon receipt of a PFC packet. Let CEV be the received class, and PT[*i*] be the ith received pause timer, for i = 0 .. 15.

When CEV ≠ 0 in a received PFC packet, CN71XX updates GMX(0)_RX_HG2_STATUS[LGTIM2GO] as follows:

TBD

CN71XX sets GMX(0)_RX_HG2_STATUS[XOF<*i*>] whenever CEV<*i*> is set to 1 and PT[*i*] ≠ 0 in a received PFC packet.

CN71XX clears GMX(0)_RX_HG2_STATUS[XOF<i>] whenever either of the two following conditions is true:

- GMX(0)_RX_HG2_STATUS[LGTIM2GO] = 0
- CEV<*i*> is set to 1 and (PT[*i*] = 0) in a received PFC packet

In the discussions below, we refer to this equation:

FINAL_XOF<15:0> = (GMX(0)_RX_HG2_STATUS[XOF<15:0>] & GMX(0)_HG2_CONTROL[LOGL_EN<15:0>]) | GMX(0)_TX_OVR_BP[TX_PRT_BP<15:0>]

GMX(0)_HG2_CONTROL[LOGL_EN] allows software to ignore the flow control received from the PFC packets. GMX(0)_TX_OVR_BP[TX_PRT_BP] allows software to emulate the effects of PFC backpressure.

Note that the backpressure state is not changed if FCS checking is enabled (GMX(0)_RX(0..3)_FRM_CHK[FCSERR]) and the PFC packet failed the FCS/CRC check.

If PREAMBLE/SFD is present (i.e. GMX(0)_RX(0..3)_FRM_CTL[PRE_CHK] = 1, refer to Section 24.3.4.1) and/or if user-defined data is present (i.e. GMX(0)_RX(0..3)_UDD_SKP[LEN] ≠ 0, refer to Sections 24.3.4.2 and 24.3.4.3) in a received packet, CN71XX discards them before it determines whether a packet is a PFC frame.

Cavium Confidential For Power Yang ALU 07/07/2014

CN71XX recognizes the remaining packet bytes as a PFC frame if all the following conditions are true:

- GMX(0)_PRT0_CBFC_CTL[RX_EN] = 1.
- The first six bytes of the packet (i.e. the L2 destination address) match one of these two enabled comparisons:
  - GMX(0)_RX(0..3)_FRM_CTL[CTL_MCST] = 1 and the first six bytes equal the standard multicast address 0x01-80-C2-00-00-01.
  - GMX(0)_RX(0..3)_FRM_CTL[CTL_SMAC] = 1 and the first six bytes equal GMX(0)_SMAC0[SMAC].
- The 13th and 14th bytes (i.e. the L2 type field) equal 0x88–08 (MAC control).
- The 15th and 16th bytes (i.e. the MAC control opcode) equal 0x01–01.
- The packet is large enough to contain all 16 time counts.
- The packet is free of lower-layer errors (e.g. FCS, RSVERR).

Note that PREAMBLE/SFD should normally be present in received PFC frames (and GMX(0)_RX(0..3)_FRM_CTL[PRE_CHK] should be set) whenever either BCK_EN or DRP_EN are set.

CN71XX generates two forms of flow control from received PFC packets:

- Physical-link flow-control (see Section 24.3.2.2.1)
- Logical-link flow-control (see Section 24.3.2.2.2)

24.3.2.2.1 Physical-Link Flow Control From Received PFC Packets and CSRs

CN71XX's PFC physical-link flow-control response is similar to its response to IEEE 802.3 pause frames: it stops transmission of all new packets (from all classes) on the physical link.

CN71XX PFC physical-link flow control is active whenever all the following conditions are met:

- GMX(0)_PRT0_CBFC_CTL[RX_EN] is set
- FINAL_XOF<15:0> is non-zero
- all bits set in GMX(0)_PRT0_CBFC_CTL[PHYS_BP] are also set in FINAL_XOF<15:0>

  (FINAL_XOF is defined in Section 24.3.2.2.)

Typically, GMX(0)_PRT0_CBFC_CTL[PHYS_BP<15:0>] will indicate which of the 16 possible classes/ports are used, so that CN71XX applies physical-link flow control whenever all classes/ports are flow-controlled. When eight classes/ports are used, GMX(0)_PRT0_CBFC_CTL[PHYS_BP<15:0>] would typically be set to value 0x00FF, for example.

24.3.2.2.2 PKO Flow Control From Received PFC Packets and CSRs

Aside from physical-link flow control, the CN71XX RXAUI logic does not directly obey the PFC backpressure. Instead, it forwards this backpressure to the CN71XX PKO hardware which may use it to stop packet transmit.

FINAL_XOF<15:0>, defined in Section 24.3.2.2, is the backpressure passed to PKO. How the PKO hardware uses this logical-link backpressure to backpressure its internal ports is described in Section 10.8.

Cavium Confidential For Bowen Yang ALU 02/27/2014

24.3.2.3 Physical-/Logical-Link Flow Control and Received HiGig2 Messages

The HiGig2 protocol allows a HiGig2 transmitter to transmit HiGig2 messages that can supply backpressure information to a receiver. (HiGig2 is indicated by GMX(0)_TX_XAUI_CTL[HG_EN] = 1 and GMX(0)_RX(0..3)_UDD_SKP[LEN] = 0x10.)

HiGig2 messages may be either physical link or logical link. Physical-link backpressure HiGig2 messages are like IEEE 802.3 pause frames and are covered in Section 24.3.2.3.1. Logical-link backpressure HiGig2 messages are unlike IEEE 802.3 pause frames, and are covered in Section 24.3.2.3.2.

Whenever any HiGig2 messages are transmitted by the attached HiGig2 device and received by CN71XX, GMX(0)_HG2_CONTROL[HG2RX_EN] must be set to one so that the RXAUI hardware will remove the HiGig2 messages from the receive stream. HiGig2 messages can never be forwarded to software as a normal packet; they must and can only be handled by the RXAUI hardware. CN71XX can receive and tolerate HiGig2 messages at all conditions allowed by the HiGig2 specifications.

Note that any received HiGig2 message is ignored (and GMX(0)_RX(0..3)_INT_REG[HG2CC] is set) when a received HiGig2 message fails the FCS/CRC check.

24.3.2.3.1 Flow Control From Received Physical-Link HiGig2 Messages

The RXAUI hardware responds to physical-link backpressure HiGig2 messages similar to the way it responds to IEEE 802.3 pause frames (refer to Section) — it sets a time counter (GMX(0)_RX_HG2_STATUS[PHTIM2GO]) that can defer transmit of packets on the interface:

- When GMX(0)_HG2_CONTROL[PHYS_EN] is set to 1, the RXAUI hardware does not start transmitting a new packet on the interface whenever GMX(0)_RX_HG2_STATUS[PHTIM2GO] ≠ 0. (As with 802.3 pause frames, CN71XX only defers transmission of nonflow-control packets, and only defers on packet boundaries.)
- When GMX(0)_HG2_CONTROL[PHYS_EN] is cleared to 0, the RXAUI hardware never stops packet transmit due to HiGig2 message physical port backpressure.

When GMX(0)_RX_HG2_STATUS[PHTIM2GO] ≠ 0, it is a direct indication that (GMX(0)_HG2_CONTROL[HG2RX_EN] was/is set to 1 and) the RXAUI hardware recently received a message containing physical-link backpressure. GMX(0)_HG2_CONTROL[PHYS_EN] allows software to ignore the physical-link backpressure received from HiGig2 messages.

24.3.2.3.2 PKO Flow Control From Received Logical-Link HiGig2 Messages and CSRs

As with physical-link backpressure, the RXAUI hardware responds to logical-link backpressure HiGig2 messages by setting a time counter (GMX(0)_RX_HG2_STATUS[LGTIM2GO]), and when the counter is nonzero, backpressure asserts. But there are several differences between logical-link backpressure and physical-link backpressure:

1. The RXAUI hardware does not directly obey the backpressure. Instead, it forwards the logical-link backpressure to the PKO hardware which may use it to stop packet transmit. A description of how the PKO hardware uses the logical-link backpressure to backpressure its internal ports is in Section 10.8.

2. There is one logical-link counter that covers 16 channels of different logical-link backpressure (GMX(0)_RX_HG2_STATUS[XOF]). The RXAUI hardware maps

Cavium Confidential For Review Only 09/12/14

backpressure from the 16 channels on to the pipe space via GMX(0)_TX_(0..3)_PIPE and forwards the pipe state to PKO.

It is possible for software to emulate the effects of logical-link HiGig2 backpressure by setting GMX(0)_TX_OVR_BP[TX_PRT_BP]. Let FINAL_XOF<15:0> be defined as:

FINAL_XOF<15:0> = (GMX(0)_RX_HG2_STATUS[XOF<15:0>] & GMX(0)_HG2_CONTROL[LOGL_EN<15:0>]) | GMX(0)_TX_OVR_BP[TX_PRT_BP<15:0>]

The RXAUI hardware asserts logical-link backpressure to PKO for port *i* whenever FINAL_XOF<*i*> true.

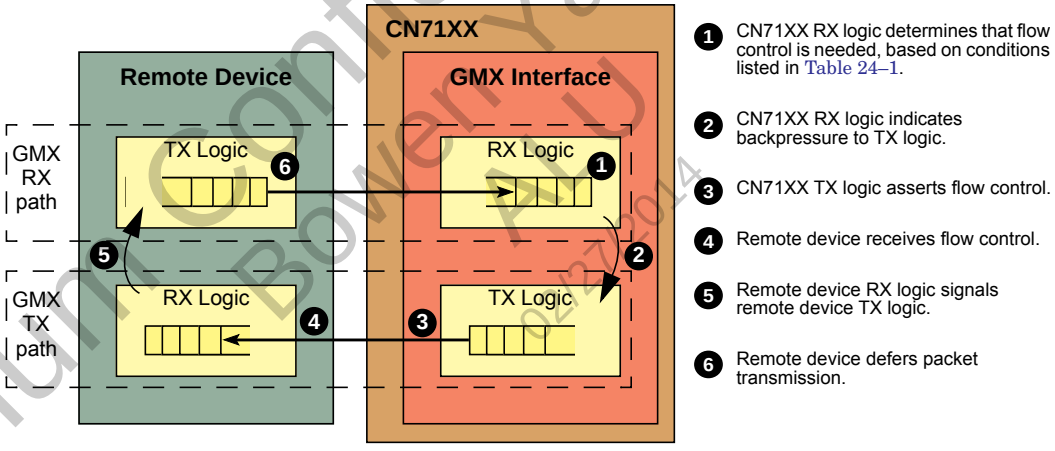
GMX(0)_HG2_CONTROL[LOGL_EN] allows software to ignore the logical-link backpressure received from HiGig2 messages on a port-by-port basis. GMX(0)_TX_OVR_BP[TX_PRT_BP] allows software to emulate the effects of PFC backpressure.

When GMX(0)_RX_HG2_STATUS[XOF] ≠ 0, it is a direct indication that GMX(0)_HG2_CONTROL[HG2RX_EN] was/is set to 1 and the RXAUI hardware recently received a logical-link HiGig2 message that asserted backpressure for one or more of the logical-link ports. The same is true for GMX(0)_RX_HG2_STATUS[LGTIM2GO]: any time GMX(0)_RX_HG2_STATUS[LGTIM2GO] ≠ 0, it indicates that the RXAUI hardware recently received a logical-link HiGig2 message.

24.3.2.4 Flow Control Transmitted via Autogenerated IEEE 802.3 Pause Packets

CN71XX flow control is shown in Figure 24–3.

Figure 24–3 CN71XX Flow Control



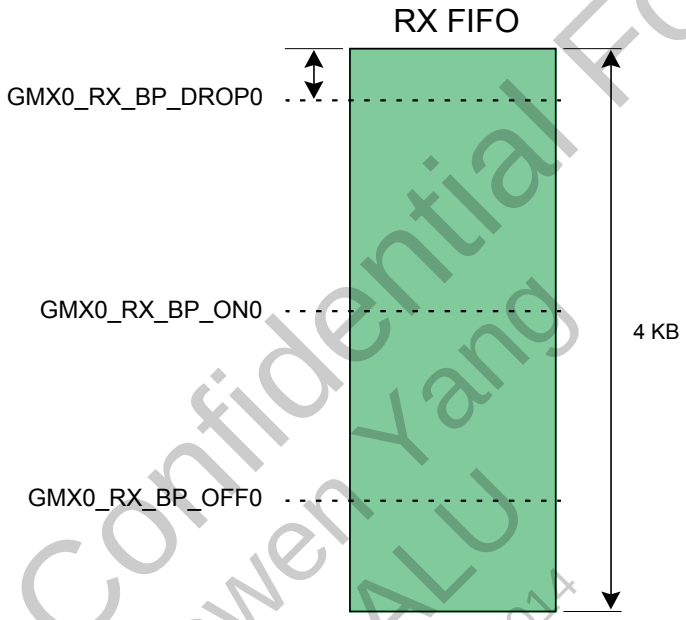
Cavium Confidential For Bowen Yang AU 02-27-2014

CN71XX controls the flow of packets from the transmitting device under the conditions specified in Table 24–1.

Table 24–1 Flow Control Conditions

Condition	Description
Receive FIFO Flow Control	The receive FIFO can control the flow as it fills up. Figure 24–4 shows the relative FIFO depths for each programmable parameter. CN71XX begins to flow control to port when the FIFO accumulates GMX(0)_RX_BP_ON0 data. Flow control is removed once the FIFO drops below GMX(0)_RX_BP_OFF0 data.
Per-Port Flow Control	PIP/IPD can also communicate per-port flow control. This flow control is exhibited the same way as receive FIFO flow control. Refer to Sections 9.6 and 9.7 for more detailed information.
Software Flow Control	Software can use GMX(0)_TX_OVR_BP[EN<0>/BP<0>] to force or ignore the flow-control state as desired. If GMX(0)_TX_OVR_BP[EN<0>] = 1 for a port, the flow-control state is GMX(0)_TX_OVR_BP[BP<0>].
PFC Flow Control	TBD

Figure 24–4 Receive FIFO Parameters



CN71XX can automatically generate IEEE 802.3 pause packets when neither PFC frames (refer to Section 24.3.2.5) nor HiGig2 messages (refer to Section 24.3.2.6) are enabled for autogeneration (i.e. CN71XX can automatically generate IEEE 802.3 pause packets when GMX(0)_PRT0_CBFC_CTL[TX_EN] = 0 and GMX(0)_HG2_CONTROL[HG2TX_EN] = 0).

Figure 24–5 shows CN71XX's autogenerated IEEE 802.3 pause packets. Much of the packet is programmable, and can differ for different ports. CN71XX sends these 60 byte packets out the SerDes interface similar to normal packets — refer to Figure 24–10 and Section 24.3.5. The only processing differences from normal PKO-generated packets are with regard to transmit PREAMBLE/SFD prepend and FCS append. Sections 24.3.5.4 and 24.3.5.5 below describe the differences. When flow control is active, CN71XX autogenerates pause packets at a frequency indicated by GMX(0)_TX(0..3)_PAUSE_PKT_INTERVAL[INTERVAL]. If GMX(0)_TX(0..3)_PAUSE_ZERO[SEND] = 1, CN71XX autogenerates a pause packet with a pause time of 0 when flow-control deasserts, which can allow the remote transmitter to restart more quickly.

Cavium Confidential For Bowen Yang ALU 12/27/2014

Figure 24–5 Generated Pause Packet Format

HG_HDR =	GMX(0)_TX_XAUI_CTL[HG_PAUSE_HGI]	HG_HDR value
	0	0xFB0000000000
	1	0xFB4000000000
	2	0xFB8000000000
	3	0xFBC000000000

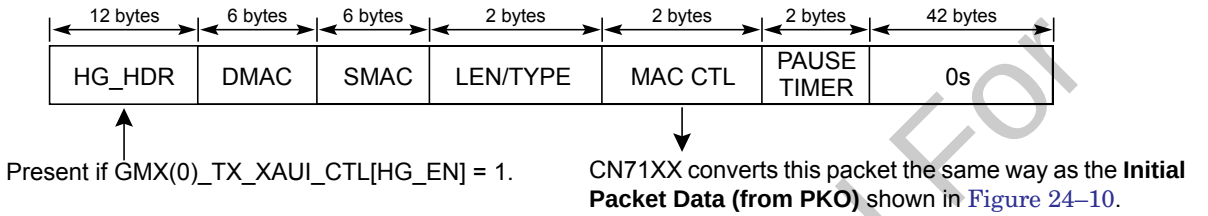
DMAC = GMX(0)_TX_PAUSE_PKT_DMAC[SMAC]

SMAC = GMX(0)_TX_SMAC0[SMAC]

LEN/TYPE = GMX(0)_TX_TX_PAUSE_PKT_TYPE[TYPE]

MAC CTL = 00–01

PAUSE TIMER = GMX(0)_TX*n*_PAUSE_PKT_TIME[TIME]



When HiGig is enabled (i.e. GMX(0)_TX_XAUI_CTL[HG_EN] = 1), CN71XX includes the HG_HDR field for outgoing pause packets, which can serve as the 12-byte header/UDD for the outgoing packet. Note that pause packet autogeneration can only be used with HiGig variants that have a 12-byte header (i.e. HiGig and HiGig+).

Note that the RXAUI hardware cannot autogenerate proper IEEE 802.3 pause packets with the HiGig2 protocol. (HiGig2 is indicated by GMX(0)_TX_XAUI_CTL[HG_EN] = 1 and GMX(0)_RX(0..3)_UDD_SKP[LEN] = 0x10.) Refer to Section 24.3.2.6.

- When GMX(0)_HG2_CONTROL[HG2TX_EN] = 1, the hardware autogenerates physical-link HiGig2 messages, and never autogenerates 802.3 pause frames.
- When GMX(0)_HG2_CONTROL[HG2TX_EN] = 0 with a HiGig2 link, the hardware does not constrain 802.3 pause packet generation, so software must forcibly disable 802.3 pause packets by setting GMX(0)_TX_OVR_BP[EN<0>] to 1 and clearing GMX(0)_TX_OVR_BP[BP<0>] to 0.

Otherwise, if HiGig is not enabled (i.e. GMX(0)_TX_XAUI_CTL[HG_EN] = 0), then note the following regarding PREAMBLE/SFD and UDD:

- When CN71XX autogenerates pause packets in RXAUI mode, it also always automatically inserts PREAMBLE/SFD. This likely means that CN71XX should likely only autogenerate pause packets when there is PREAMBLE/SFD present in out-going packets. Section 24.3.5.5 describes PREAMBLE/SFD autoinsertion.
- CN71XX should likely only autogenerate pause packets when there is no UDD present in out-going packets. Section 24.3.4.3 describes the UDD in received packets. The CN71XX GMX transmit logic is not aware of any UDD in transmitted packets. It also does not add any UDD in the pause packets it autogenerates. In cases when there is UDD present in out-going packets, software should likely generate all outgoing pause packets.

Cavium Confidential For Bowen Yang 02/7/2013

24.3.2.5 Flow Control Transmitted via Autogenerated PFC Messages

As described in Section 24.3.2.2, CN71XX implements the proposed specification for priority-based flow control (PFC). CN71XX can transmit autogenerated PFC frames when GMX(0)_PRT0_CBFC_CTL[TX_EN] = 1 and GMX(0)_HG2_CONTROL[HG2TX_EN] = 0.

Figure 24–6 shows CN71XX's autogenerated PFC frames.

Figure 24–6 Generated PFC Frame Format

HG_HDR =

GMX(0)_TX_XAUI_CTL[HG_PAUSE_HGI]	HG_HDR value
0	0xFB0000000000
1	0xFB4000000000
2	0xFB8000000000
3	0xFBC000000000

DMAC = GMX(0)_TX_PAUSE_PKT_DMAC[SMAC]

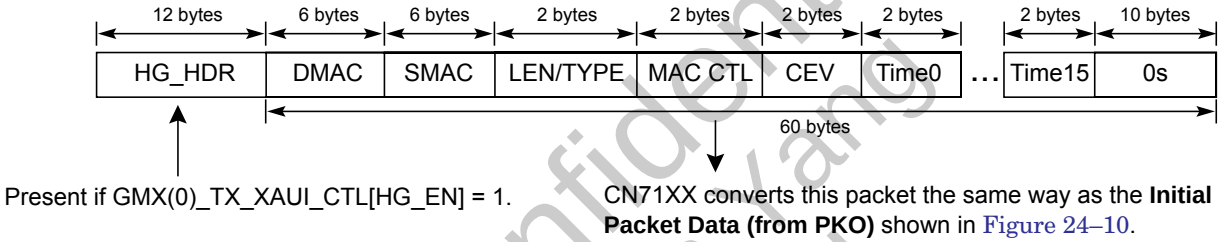
SMAC = GMX(0)_TX_SMAC0[SMAC]

LEN/TYPE = GMX(0)_TX_TX_PAUSE_PKT_TYPE[TYPE]

MAC CTL = 01–01

CEV = GMX(0)_PRT0_CBFC_CTL[PHYS_EN] | GMX(0)_PRT0_CBFC_CTL[LOGL_EN]

Time0..Time15 = GMX(0)_TX(0..3)_PAUSE_PKT_TIME[TIME] or 0



Much of the packet is programmable. CN71XX sends these 60/72-byte packets out through the SerDes interface similar to the way it sends normal packets — PREAMBLE is always and FCS is typically added (refer to Figure 24–10 and Section 24.3.5). The only processing differences from normal PKO-generated packets are with regard to transmit PREAMBLE/SFD prepend and FCS append. Sections 24.3.5.4 and 24.3.5.5 describe the differences.

Similar to autogenerated 802.3 pause packets, GMX(0)_TX(0..3)_PAUSE_PKT_TIME[TIME] or 0 is the time included in the 16 time fields of all classes/ports in every autogenerated PFC frame. When flow control is active for one or more classes/ports, CN71XX autogenerates PFC frames at a frequency indicated by GMX(0)_TX(0..3)_PAUSE_PKT_INTERVAL[INTERVAL]. If GMX(0)_TX(0..3)_PAUSE_ZERO[SEND] = 1, CN71XX autogenerates PFC frames whenever flow control first becomes inactive for all classes/ports, which can allow the remote transmitter to restart more quickly.

CN71XX considers flow-control to be active for a class/port whenever there is either physical-link backpressure or logical-link backpressure for the class/port. CN71XX's creation of physical-link backpressure is described in Section 24.3.2.5.1 and CN71XX's creation of logical-link backpressure is described in Section 24.3.2.5.2.

Cavium Confidential For Chen Yang 8/17/2021

When HiGig is enabled (i.e. GMX(0)_TX_XAUI_CTL[HG_EN] = 1), CN71XX includes the HG_HDR field for outgoing PFC frames, which can serve as the 12-byte header/UDD for the outgoing packet. Note that PFC autogeneration can only be used with HiGig variants that have a 12-byte header (i.e. HiGig and HiGig+).

NOTE: The RXAUI hardware cannot autogenerate proper PFC frames with the HiGig2 protocol. HiGig2 is indicated when GMX(0)_TX_XAUI_CTL[HG_EN] = 1 and GMX(0)_RX(0..3)_UDD_SKP[LEN] = 0x16 (refer to Section 24.3.2.6).

- When GMX(0)_HG2_CONTROL[HG2TX_EN] = 1, the hardware autogenerates physical-link HiGig2 messages, and never autogenerates PFC frames.
- When GMX(0)_HG2_CONTROL[HG2TX_EN] = 0 with a HiGig2 link, the hardware does not constrain PFC frame generation, so software must forcibly disable both PFC and pause frames with the following steps:
  - clearing GMX(0)_PRT0_CBFC_CTL[TX_EN] to 0
  - setting GMX(0)_TX_OVR_BP[EN<0>] to 1
  - clearing GMX(0)_TX_OVR_BP[BP<0>] to 0.

Otherwise, if HiGig is not enabled (i.e. GMX(0)_TX_XAUI_CTL[HG_EN] = 0), then note the following regarding PREAMBLE/SFD and UDD:

- When CN71XX autogenerates PFC frames in RXAUI mode, it also always automatically inserts PREAMBLE/SFD. This likely means that CN71XX should only autogenerate PFC frames when there is PREAMBLE/SFD present in out-going packets. Section 24.3.5.5 describes PREAMBLE/SFD autoinsertion.
- CN71XX should likely autogenerate PFC frames only when there is no UDD present in out-going packets. Section 24.3.4.3 describes the UDD in received packets. The CN71XX GMX transmit logic is not aware of any UDD in transmitted packets. It also does not add any UDD in the PFC frames it autogenerates. In cases when there is UDD present in out-going packets, software should likely generate all outgoing PFC frames.

24.3.2.5.1 Physical-Link Backpressure Via Autogenerated PFC Messages

When CN71XX can transmit autogenerated PFC frames, CN71XX generates physical-link PFC backpressure under the exact same conditions as shown in Section 24.3.2.4 for autogenerated IEEE 802.3 pause packets. PFC physical-link backpressure can serve the same purpose as IEEE 802.3 pause frames (refer to Section 24.3.2.4, Table 24–1, and Figure 24–4).

CN71XX applies physical-link backpressure by asserting flow control to every class/port <*i*> where GMX(0)_PRT0_CBFC_CTL[PHYS_EN<*i*>] is set. Typically, GMX(0)_PRT0_CBFC_CTL[PHYS_EN] should select all classes/ports being used. For example, with eight classes/ports, GMX(0)_PRT0_CBFC_CTL[PHYS_EN] should typically be set to 0xFF.

CN71XX can initiate new PFC frames when physical-link backpressure asserts or when physical-link backpressure deasserts and logical backpressure is still enabled for some classes/ports (or no backpressure is enabled and GMX(0)_TX(0..3)_PAUSE_ZERO[SEND] is set to 1).

Cavium Confidential For Bowen Yang ALD 01/7/2014

24.3.2.5.2 Logical-Link Backpressure Via Autogenerated PFC Messages

When CN71XX can transmit autogenerated PFC frames, the RXAUI hardware generates logical-link flow control when software sets GMX(0)_TX0_CBFC_XOFF[XOFF] to a nonzero value. CN71XX generates logical-link flow control for class/port *i* whenever both GMX(0)_TX0_CBFC_XOFF[XOFF<*i*>] and GMX(0)_PRT0_CBFC_CTL[LOGL_EN<*i*>] are set. Both XOFF and LOGL_EN are 16-bit fields – one bit per possible class/port.

GMX(0)_TX0_CBFC_XOFF[XOFF] is conveniently structured for lock-free software access to the individual bits/ports. Writing 1s to GMX(0)_TX0_CBFC_XOFF[XOFF] cause corresponding GMX(0)_TX0_CBFC_XOFF[XOFF] bits to be set. Writing 1s to GMX(0)_TX0_CBFC_XON[XON] cause corresponding GMX(0)_TX0_CBFC_XOFF[XOFF] bits to be cleared. With this, different CPU cores can easily simultaneously set/clear different GMX(0)_TX0_CBFC_XOFF[XOFF] bits without interference.

CN71XX can initiate new PFC frames when software writes GMX(0)_TX0_CBFC_XOFF[XOFF] with a non-zero value or software writes GMX(0)_TX0_CBFC_XON[XON] with a non-zero value such that the logical backpressure changes.

24.3.2.6 Flow Control Transmitted via Autogenerated HiGig2 Messages

The HiGig2 protocol allows the RXAUI hardware to transmit HiGig2 messages to supply backpressure information to a remote receiver. Section 24.3.2.6.1 covers the generation of physical-link backpressure HiGig2 messages. Section 24.3.2.6.2 discusses the transmit of logical-link HiGig2 messages.

The RXAUI hardware is enabled to transmit HiGig2 messages when GMX(0)_HG2_CONTROL[HG2TX_EN] is set to 1. This enable should only ever be set when CN71XX connects to a device that implements HiGig2 and is expecting to receive HiGig2 messages. GMX(0)_HG2_CONTROL[HG2TX_EN] enables both physical-link and logical-link HiGig2 messages.

The RXAUI hardware can only generate HiGig2 messages as a normal packet between other packets, like IEEE 802.3 pause frames.

As with autogenerated 802.3 pause packets, GMX(0)_TX(0..3)_PAUSE_PKT_TIME[TIME] is the time included in every autogenerated HiGig2 message. When physical-link (virtual link) flow control is active, CN71XX autogenerates physical-link (virtual-link) HiGig2 messages at a frequency indicated by GMX(0)_TX(0..3)_PAUSE_PKT_INTERVAL[INTERVAL].

NOTE: When CN71XX is attached to a HiGig2 device and GMX(0)_HG2_CONTROL[HG2TX_EN] is cleared to 0 (to disable HiGig2 message generation), both IEEE 802.3 pause-packet and PFC frame generation must be forcibly disabled with the following steps:

- clear GMX(0)_PRT0_CBFC_CTL[TX_EN] to 0
- set GMX(0)_TX_OVR_BP[EN<0>] to 1
- clear GMX(0)_TX_OVR_BP[BP<0>] to 0.

This is done because CN71XX can generate neither 802.3 pause packets nor PFC frames in HiGig2 mode.

Cavium Confidential For Power Yang AU 01/27/2014

24.3.2.6.1 Physical-Link Flow Control Transmitted via Autogenerated HiGig2 Messages

When GMX(0)_HG2_CONTROL[HG2TX_EN] is set to 1, the RXAUI hardware generates physical-link HiGig2 messages when there is backpressure generated by the exact same conditions as shown in Section 24.3.2.4 for autogenerated IEEE 802.3 pause packets. Physical-link backpressure via HiGig2 messages serves the same purpose as IEEE 802.3 pause frames (refer to Section 24.3.2.4, Table 24–1, and Figure 24–4).

When physical-link flow control is active, CN71XX autogenerates physical-link HiGig2 messages at a frequency indicated by GMX(0)_TX(0..3)_PAUSE_PKT_INTERVAL[INTERVAL]. If GMX(0)_TX(0..3)_PAUSE_ZERO[SEND] = 1, CN71XX autogenerates a physical-link HiGig2 message with a pause time of 0 when the physical-link backpressure deasserts, which can allow the remote transmitter to restart more quickly.

24.3.2.6.2 Logical-Link Flow Control Transmitted via Autogenerated HiGig2 Messages

When GMX(0)_HG2_CONTROL[HG2TX_EN] is set to 1, the RXAUI hardware generates logical-link HiGig2 messages whenever software sets GMX(0)_TX_HG2_REG1[TX_XOF] to a nonzero value. GMX(0)_TX_HG2_REG1[TX_XOF] is a 16-bit field — one bit for each of the 16 possible logical-link ports.

GMX(0)_TX_HG2_REG1[TX_XOF] is conveniently structured for lock-free software access to the individual bits/ports. Writing 1s to GMX(0)_TX_HG2_REG1[TX_XOF] cause corresponding GMX(0)_TX_HG2_REG1[TX_XOF] bits to be set. Writing 1s to GMX(0)_TX_HG2_REG2[TX_XON] cause corresponding GMX(0)_TX_HG2_REG1[TX_XOF] bits to be cleared. With this, different CPU cores can easily simultaneously set/clear different GMX(0)_TX_HG2_REG1[TX_XOF] bits without interference.

When GMX(0)_TX_HG2_REG1[TX_XOF] ≠ 0, CN71XX autogenerates logical-link HiGig2 messages at a frequency indicated by GMX(0)_TX(0..3)_PAUSE_PKT_INTERVAL[INTERVAL]. CN71XX also autogenerates logical-link HiGig2 messages any time software updates GMX(0)_TX_HG2_REG1[TX_XOF] to a nonzero value. If GMX(0)_TX(0..3)_PAUSE_ZERO[SEND] = 1, CN71XX autogenerates a physical-link HiGig2 message with a pause time of 0 when software changes GMX(0)_TX_HG2_REG1[TX_XOF] from a nonzero value to zero, which can allow the remote transmitter to restart more quickly.

24.3.3 Receive Packet Dropping

There are several mechanisms within CN71XX that will drop packets:

- preamble errors
- receive-FIFO packet dropping
- pause packet drops
- DMAC filters
- PIP/IPD per-QOS or per-port admission control

These mechanisms are described in the following subsections.

Cavium Confidential For Bosen Yang 02/27/2015

24.3.3.1 PREAMBLE Errors

If preamble checking is enabled (GMX(0)_RX(0..3)_FRM_CTL[PRE_CHK] = 1) and the packet does not send a valid PREAMBLE followed by SFD, the packet is dropped.

A valid PREAMBLE/SFD must include:

- A start (/S/) code group in lane 0 (CN71XX PCSX logic converts this to a 0xFB for the GMX logic), followed by
- Exactly six data code groups converting to PREAMBLE (0x55), followed by
- A data code group converting to SFD (0xD5) in lane 3

Figure 24–8 shows an example of a valid PREAMBLE/SFD in the byte stream for a packet.

Note that PREAMBLE/SFD must be present in packets (refer to Section 24.3.4.1) for preamble checking to be enabled. Note also that GMX(0)_RX(0..3)_FRM_CTL[PRE_CHK] must be 0 when HiGig is enabled (i.e. when GMX(0)_TX_XAUI_CTL[HG_EN] = 1), and PREAMBLE/SFD is never present.

24.3.3.2 Receive-FIFO Packet Dropping

When the receive FIFO exceeds the GMX(0)_RX_BP_DROP0 (as shown in Figure 24–4), the packet cannot be buffered and must be dropped. If there is room in the FIFO for a partial portion of the packet, the beginning of the packet is buffered but is sent down the IOBI with the PARTIAL opcode (indicating a drop). As described is Section 9.5, partial packets can be recognized by software when WORD2[RE] = 1 and WORD2[OPCODE] = 1 (partial error). All other packets are completely dropped and are not sent on the IOBI.

Both partial drops and complete drops set the CIU_INT*n*_SUM0[GMX_DRP*m*] bit in CIU and can raise an interrupt based on CIU_INT*n*_EN0[GMX_DRP*m*].

In most configurations, GMX receive-FIFO packet drops should be non-existent or rare. If drops are required, PIP/IPD packet drops, described in Section 24.3.3.5, should be used instead in most cases.

24.3.3.3 Pause Packet Drops

CN71XX drops any incoming pause packet, when GMX(0)_RX(0..3)_FRM_CTL[CTL_DRP] = 1. See Section for additional details. In addition, pause packets can be dropped if GMX(0)_RX(0..3)_FRM_CTL[CTL_DRP] = 0 and the receive-FIFO is full. See Section 24.3.3.2 for additional details.

CN71XX sets GMX(0)_RX(0..3)_INT_REG[PAUSE_DRP] = 1 and GMX(0)_RX(0..3)_PAUSE_DROP_TIME[STATUS] = pause_time extracted from the last pause packet that is dropped when GMX(0)_RX(0..3)_FRM_CTL[CTL_DRP] = 0 and a pause packet cannot be buffered in the receive FIFO. The hardware can optionally raise an interrupt if GMX(0)_RX(0..3)_INT_EN[PAUSE_DRP] = 1.

24.3.3.4 DMAC Filter

CN71XX can store up to eight full MAC addresses to match against (GMX(0)_RX(0..3)_ADR_CAM_EN and GMX(0)_RX(0..3)_ADR_CAM*). CN71XX can be configured to accept or drop packets based on this address match

Cavium Confidential Por
Bowen Yang
ALU
5/27/2014

(GMX(0)_RX(0..3)_ADR_CTL[CAM_MODE]). In addition, CN71XX can be configured to accept or reject multicast or broadcast packets (GMX(0)_RX(0..3)_ADR_CTL[MCST/ BCST]).

The full algorithm is shown below:

```
bool dmac_addr_filter(uint48 dmac) {
  if (is_bcst(dmac))                                    //broadcast accept
    return (GMX_RXn_ADR_CTL[BCST] ? ACCEPT : REJECT);
  if (is_mcst(dmac) & GMX_RXn_ADR_CTL[MCST] == 1)     //multicast reject
    return REJECT;
  if (is_mcst(dmac) & GMX_RXn_ADR_CTL[MCST] == 2)     //multicast accept
    return ACCEPT;
  cam_en = 0;
  cam_hit = 0;
  for (i=0; i<8; i++) {
    if (GMX_RXn_ADR_CAM_EN[EN<i>] == 0)
       continue;
    cam_en++;
    uint48 unswizzled_mac_adr = 0x0;
    for (j=5; j>=0; j--) {
       cam_byte = GMX_RXn_ADR_CAM{j}[ADR<i*8+7:i*8>];
       unswizzled_mac_adr = (unswizzled_mac_adr << 8) | cam_byte;
    }
    if (unswizzled_mac_adr == dmac) {
       cam_hit = 1;
       break;
    }
  }
  if (cam_hit)
    return (GMX_RXn_ADR_CTL[CAM_MODE] ? ACCEPT : REJECT);
  else
    return (GMX_RXn_ADR_CTL[CAM_MODE] ? REJECT : ACCEPT);
}
```

24.3.3.5 PIP/IPD Per-QOS and Per-Port Admission Control

Sections 9.6 and 9.7 discuss PIP/IPD's QOS and per-port admission control. Packets can optionally be dropped when they fail QOS or per-port threshold checks.

24.3.4 Receive Packet Processing and Format

The following subsections describe the processing and format of received packets.

Note that this section does not cover HiGig2 message receipt. When attached to a HiGig2 device, CN71XX can receive HiGig2 messages according to the rules described in the HiGig2 specifications.

24.3.4.1 Code Group Conversion to Packet Bytes

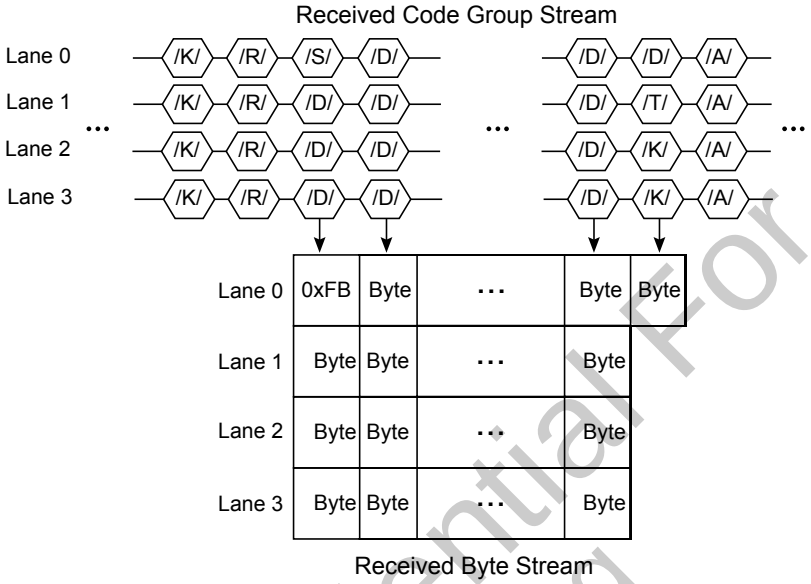
After the CN71XX PCSX receive logic obtains bit lock, code group synchronization, and alignment (i.e. after PCSX(0)_BIT_LOCK_STATUS_REG[BIT_LOCK*] = 1 and PCSX(0)_10GBX_STATUS_REG[L*SYNC,ALIGND] = 1), and if the CN71XX reconciliation layer obtains link_fault = OK (i.e. if GMX(0)_RX_XAUI_CTL[STATUS] = 0), CN71XX then reliably converts received packets from a sequence of code groups on four parallel lanes to a byte stream.

Figure 24–7 shows a typical received RXAUI packet and the byte stream that CN71XX first converts it to. Time progresses from left to right. Amongst the four code groups/bytes that arrive together, lane 0 contains the first byte and lane 3 contains

Cavium Confidential For Bowen Yang ALU 02/27/2014

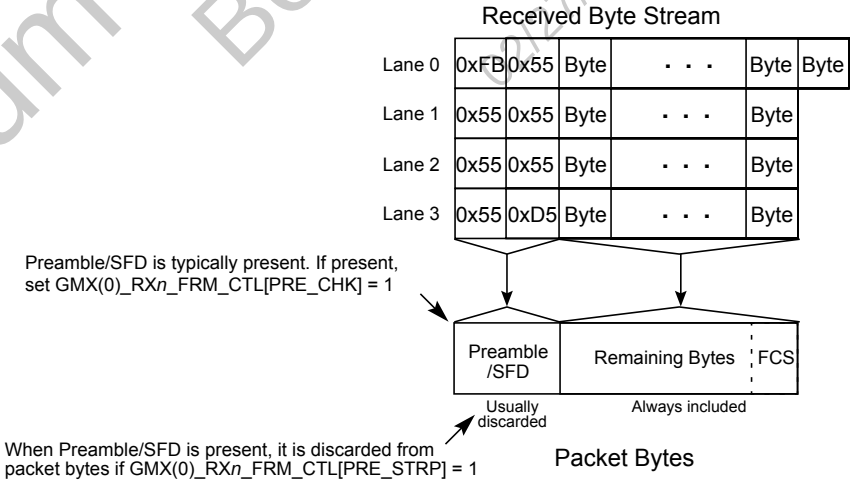
the last byte. The received code group stream for the packet consists of a Start code group (/S/) followed by the packet data, followed by a Terminate code group (/T/). Idle columns (i.e. columns of /K/s, /R/s, or /A/s) may occur between consecutive packets. CN71XX first converts the packet into the indicated byte stream. CN71XX translates the Start code group into 0xFB (typically considered a PREAMBLE byte), and the subsequent code groups prior to the Terminate code group according to the data code group rules specified in Clause 46, 47, and 48 of the IEEE 802.3 specifications.

Figure 24–7 Converting Code Groups to Bytes



CN71XX RXAUI logic next converts the byte stream into packet bytes. Figure 24–8 shows this. This step processes the PREAMBLE/SFD that is typically in the received packet. CN71XX assumes that a PREAMBLE/SFD is present in the received packet when GMX(0)_RX(0..3)_FRM_CTL[PRE_CHK] = 1. When present, the PREAMBLE/SFD is typically discarded. But CN71XX can also optionally retain all the received bytes as packet bytes when GMX(0)_RX(0..3)_FRM_CTL[PRE_STRP] = 0. This may be useful for debug or when CN71XX communicates with a device other than a PHY.

Figure 24–8 Conversion from Byte Stream to Packet Bytes



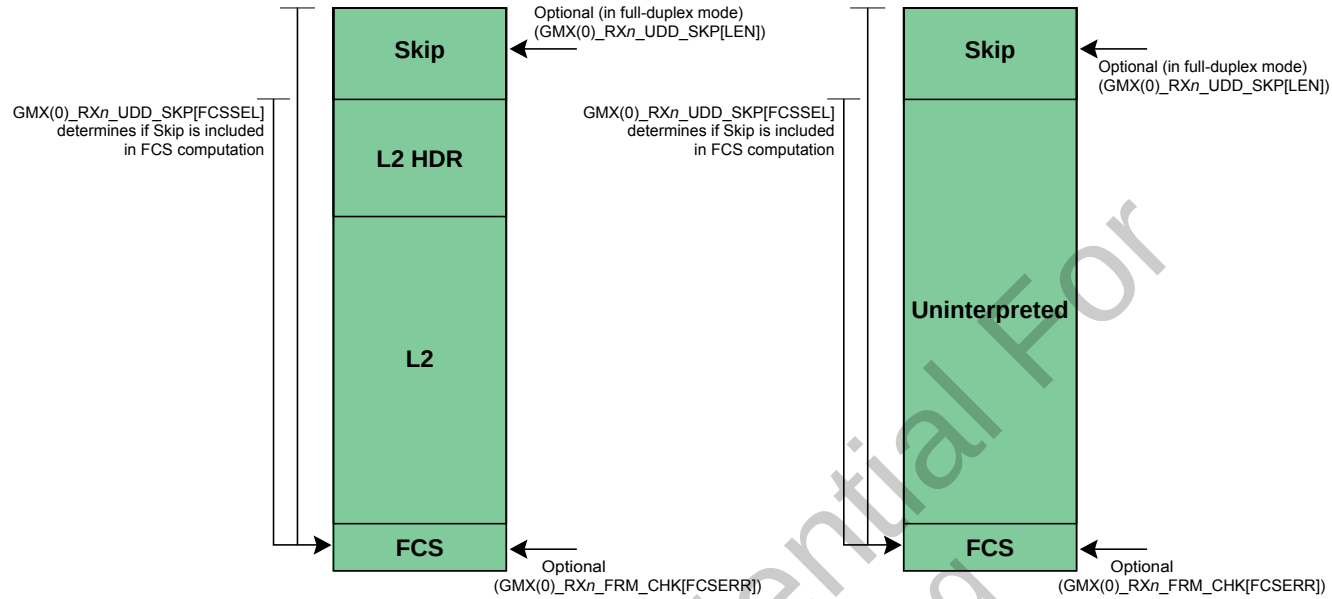
Cavium Confidential For Bowen Ang 02/7/19

Note that a legal preamble must look exactly like that shown in Figure 24–8. Refer to Section 24.3.3.1.

24.3.4.2 Receive Packet Formats

Figure 24–9 shows the supported receive-packet formats, which differ from those described in Section 9.2.

Figure 24–9 Packet Formats



The format can include the optional skip/UDD field, the L2 header, the remainder of the L2 packet, and the optional standard FCS. The next sections discuss these components.

CN71XX normally inspects the packet bytes to extract the L2 parameters that are required for various tasks and checks.

24.3.4.3 Receive User-Defined Data

The packet parser must be able to determine where the L2 header is in the packet to extract the correct data. Normally, the L2 header will be at the start of the packet bytes. CN71XX also supports user-defined data (UDD) prior to the L2 header. CN71XX can support 0–64 bytes of UDD data based on GMX(0)_RX(0..3)_UDD_SKP[LEN]. Note that this is a static per-port configuration that cannot change on a per-packet basis. CN71XX can also be configured to either compute the FCS over the UDD data or just the L2 header and L2 payload based on GMX(0)_RX(0..3)_UDD_SKP[FCSSEL].

When HiGig mode is enabled (i.e. when GMX(0)_TX_XAUI_CTL[HG_EN] = 1), the UDD corresponds to the HiGig header, so GMX(0)_RX(0..3)_UDD_SKP[LEN] should be set to the size of the header, and GMX(0)_RX(0..3)_UDD_SKP[FCSSEL] must also be set in that case.

Systems have two choices when skip bytes exist in packet bytes (followed by an ordinary Ethernet L2 header):

1. If the skip amount is constant for all packets, set GMX(0)_RX(0..3)_UDD_SKP[LEN] to the total SKIP (Skip I + PKT_INST_HDR + Skip II, refer to Section 9.2.)

Cavium Confidential For Power Yang ALU 6/27/2014

2. If the amount of total skip data before the L2 HDR is variable, the L2 HDR inspection must be disabled (see Section 24.3.4.4 for more details).

Note that this discussion applies only to the receive path. GMX is not aware of any UDD data present in out-going packets. If UDD does need to be present in out-going packets when HiGig mode is not enabled (i.e. when GMX(0)_TX_XAUI_CTL[HG_EN] = 0), pause packet autogeneration should likely be disabled, as the autogenerated packets cannot contain any UDD. Refer to Section Section 24.3.2.4. Outbound FCS is always calculated over all bytes in an outbound packet. If UDD-like bytes should not be included in the FCS for outbound packets, software should generate the FCS for the packets. Section 24.3.5.4 describes outbound FCS calculations.

24.3.4.4 Receive L2 Headers

The packet interface supports all six of the L2 header types as defined in Figure 9–9 in Chapter 9, Packet Input Processing/Input Packet Data Unit (PIP/IPD).

Some systems may not use the standard Ethernet L2 HDR header or may develop custom headers. Normal L2 HDR inspection can be disabled. If the L2 HDR is not present in the packet, the L2 HDR inspection must be disabled. See Section 24.3.4.5 for more details.

24.3.4.5 Disabling Receive L2 Header Inspection

If the L2 header is not present or is not at a fixed offset in the packet, L2 header inspection should be disabled as follows:

- DMAC filtering – the L2 DMAC field cannot be extracted from the packet. The DMAC filter can be put into promiscuous mode by setting the following parameters:

  GMX(0)_RX(0..3)_ADR_CTL[CAM_MODE] = 0
  GMX(0)_RX(0..3)_ADR_CTL[MCST] = 0
  GMX(0)_RX(0..3)_ADR_CTL[BCST] = 1
  GMX(0)_RX(0..3)_ADR_CAM_EN[EN] = 0.

  Refer to Section 24.3.3.4.

- Pause-packet recognition – the L2 DMAC and type fields cannot be extracted from the packet. Pause-packet recognition can be disabled by setting the following parameters:

  GMX(0)_RX(0..3)_FRM_CTL[CTL_BCK] = 0
  GMX(0)_RX(0..3)_FRM_CTL[CTL_DRP] = 0

  Refer to Section .

24.3.4.6 Receive FCS

This optional FCS shown in Figure 24–11 can optionally be checked by GMX but GMX always forwards the FCS bytes to PIP/IPD. PIP/IPD then optionally discards the FCS bytes. Refer to Section 9.2.6.

CN71XX checks the standard four-byte CRC-32 FCS field in the packet when GMX(0)_RX(0..3)_FRM_CHK[FCSERR] = 1. Section 24.5.1 describes the CN71XX response when the FCS check is enabled and fails.

The CRC check normally covers all prior packet bytes. CN71XX optionally excludes the UDD/skip bytes. When GMX(0)_RX(0..3)_UDD_SKP[FCSSEL] = 1, CN71XX excludes the first GMX(0)_RX(0..3)_UDD_SKP[LEN] bytes from the FCS calculation.

Cavium Confidential For Bowen Yang ALU 04/27/2014

Note that GMX(0)_RX(0..3)_UDD_SKP[FCSSEL] must be 0 in HiGig mode (i.e. when GMX(0)_TX_XAUI_CTL[HG_EN] = 1) to indicate that FCS must be calculated on the UDD/Skip data.

Note that when PREAMBLE/SFD is not present (i.e. when GMX(0)_RX*n*_FRM_CTL[PRE_CHK] = 0) and there is either no UDD or FCS is calculated on the UDD, the first byte of the received packet will always be 0xFB, and this and all packet bytes are included in the FCS calculation. Refer to Figure 24–8.

24.3.5 Transmit Packet Processing

CN71XX RXAUI transmit logic can autogenerate pause frames (refer to Section 24.3.2.4 and Figure 24–5). But most packet transmissions occur as packets arrive at the packet interface from PKO via the POB bus depicted in Figure 24–1. This section describes the CN71XX transmit logic for those outbound packets. The processing of the autogenerated packets and PKO packets is identical except for PREAMBLE/SFD and FCS. Sections 24.3.5.4 and 24.3.5.5 describe the differences.

Note that this section does not cover HiGig2 message transmit. When attached to a HiGig2 device, CN71XX can transmit HiGig2 messages according to the rules described in the HiGig2 specifications and in Section 24.3.2.6 above.

24.3.5.1 Transmit Interpacket Gap (IPG)

CN71XX RXAUI transmit logic enforces interpacket gaps (IPGs) between all transmitted packets as per the 802.3 specifications. (IPG is also known as interFrameSpacing or interFrameGap.) CN71XX enforces this gap between all packets that it transmits, which includes both the usual PKO-generated packets and any autogenerated pause packets. (Section 24.3.2.4 describes pause packet autogeneration.)

The IPG includes the /T/ that terminated the prior packet as well as all /A/s, /K/s, and /R/s before the /S/ that starts the next packet.

(GMX(0)_TX_IFG[IFG1] + GMX(0)_TX_IFG[IFG2]) is the configured minimum IPG for the MAC. Note that this must always be a multiple of four bytes/code groups.

CN71XX implements the deficit idle counter feature described in Clause 46 of the 802.3-2005 specifications when GMX(0)_TX_XAUI_CTL[DIC_EN] = 1. If this deficit idle counter feature is not enabled, CN71XX always generates more than (GMX(0)_TX_IFG[IFG1] + GMX(0)_TX_IFG[IFG2]) IPG between any two packets whenever the length of the prior packet is not a multiple of four. When GMX(0)_TX_XAUI_CTL[DIC_EN] = 1, the GMX transmit reconciliation layer logic introduces or removes idle columns as necessary so the minimum IPG equals (GMX(0)_TX_IFG[IFG1] + GMX(0)_TX_IFG[IFG2]) on average. The IPG between any two packets may fall below (GMX(0)_TX_IFG[IFG1] + GMX(0)_TX_IFG[IFG2]), but the “average minimum” equals (GMX(0)_TX_IFG[IFG1] + GMX(0)_TX_IFG[IFG2]).

CN71XX creates columns of /A/, /K/, or /R/ idles to fill the idle columns as specified in Clause 48 of 802.3-2005. (CN71XX implements the $X^7 + X^6 + 1$ PRBS.)

Cavium Confidential For Bowen Yang @ ALU 01/20/2014

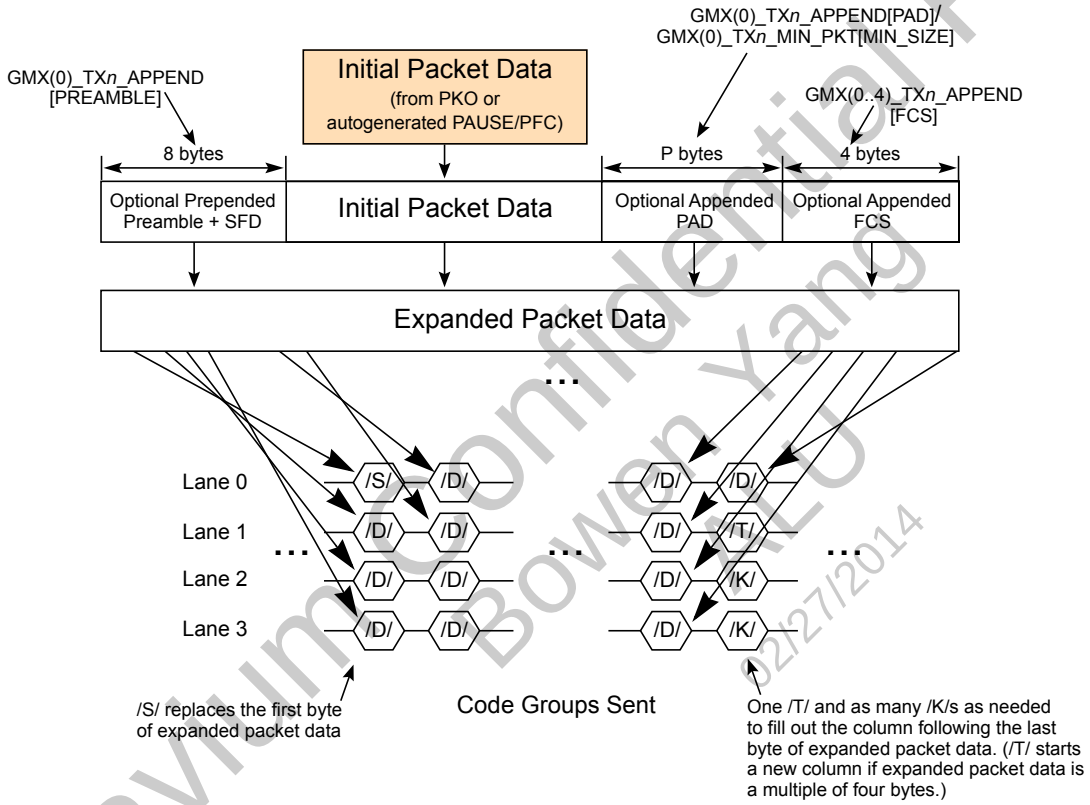
24.3.5.2 Packet Transmission

As Section 24.2 above indicates, the transmit FIFO is 4KB in RXAUI mode. Note that PKO_REG_GMX_PORT_MODE[MODE*] must be set to 0x4 when the packet interface is in RXAUI mode.

The GMX transmit state machine begins packet transmission when the FIFO contains a complete packet or if the FIFO length exceeds a programmed number of bytes (GMX(0)_TX(0..3)_THRESH). The threshold value should be large enough to avoid underflow and small enough to avoid excessive latencies. The recommended threshold value for most RXAUI applications is GMX(0)_TX(0..3)_THRESH[CNT] = 0x80 (2KB). For additional protection against packet underflow, PKO engines can be put into threshold mode, where the engines will hold onto packets as long as possible before sending. Refer to PKO_REG_ENGINE_THRESH and Chapter 10.

Figure 24–10 shows the CN71XX RXAUI logic transmit-packet translation from PKO/pause bytes to code groups on the outbound SerDes. The next sections describe this translation in detail.

Figure 24–10 Outbound Conversion to Code Groups



Cavium Confidential For Bowen Yang 02/27/2014

24.3.5.3 Optional Transmit PAD Append

First, the CN71XX RXAUI transmit logic optionally appends 0s to make all packets satisfy a minimum size requirement. When GMX(0)_TX(0..3)_APPEND[PAD] = 1, this PAD append forces all packets to be at least

4 × ((GMX(0)_TX(0..3)_MIN_PKT[MIN_SIZE] + 1) >>2) bytes.

In Figure 24–10, P is either zero bytes if the packet is large enough, else

P = 4 × ((GMX(0)_TX0_MIN_PKT[MIN_SIZE] + 1) >>2) – (# of bytes from PKO/pause)

Note that CRC is normally generated by the CN71XX RXAUI logic following the append. If so, then all packets become at least

4 × ((GMX(0)_TX(0..3)_MIN_PKT[MIN_SIZE] + 1) >>2) + 4 bytes.

In that typical case, GMX(0)_TX(0..3)_MIN_PKT[MIN_SIZE] = 59 produces the usual minimum packet of 64 bytes.

24.3.5.4 Optional Transmit FCS Append

Second, the CN71XX RXAUI transmit logic optionally calculates and appends the standard 4-byte CRC-32 FCS. When GMX(0)_TX(0..3)_APPEND[FCS] = 1, CN71XX appends the FCS, as Figure 24–10 shows.

Note that the FCS calculated for outgoing packets always covers all bytes in outgoing packets, except for autogenerated PREAMBLE/SFD. This is unlike the receive FCS calculations described by Figure 24–11 and Section 24.3.4.6, where the UDD bytes can optionally be excluded from the FCS calculation for inbound packets. If some bytes should be excluded from the FCS for outbound packets, the FCS must be generated by software.

Note that when PREAMBLE/SFD is not autogenerated by CN71XX (i.e. when GMX(0)_TX(0..3)_APPEND[PREAMBLE] = 0), CN71XX converts the first byte of packet data into a /S/. (Refer to Figure 24–10 and Section 24.3.5.6.) The CN71XX transmit FCS generation logic forces the first byte of packet data to be 0xFB when it calculates CRC. This means that the remote receiver must then convert the /S/ back into a 0xFB and calculate FCS including it to check FCS correctly when GMX(0)_TX(0..3)_APPEND[PREAMBLE]=0. (Figure 24–8 and Section 24.3.4.1 describe how a CN71XX receiver converts the /S/ to 0xFB. Section 24.3.4.6 describes CN71XX's receive FCS calculation.)

When GMX(0)_TX(0..3)_APPEND[FCS] = 1, GMX transmit FCS processing is identical for both PKO-generated and autogenerated packets. When FCS is not appended for normal PKO-generated packets (i.e. when GMX(0)_TX(0..3)_APPEND[FCS] = 0) and when no pad is added to autogenerated pause/PFC packets (i.e. when GMX(0)_TX(0..3)_MIN_PKT[MIN_SIZE] = 59 or 71, depending on whether HiGig is enabled), GMX(0)_TX(0..3)_APPEND[FORCE_FCS] can be set to 1. When GMX(0)_TX(0..3)_APPEND[FORCE_FCS] is set, CN71XX generates FCS only for the autogenerated outbound packets, and does not generate FCS for normal PKO-generated packets. This may be useful when software generates the FCS for out-going packets.

Cavium Confidential For Boren Yang 02/26/14

24.3.5.5 Optional Transmit PREAMBLE/SFD Prepend

Third, the CN71XX RXAUI transmit logic optionally prepends the standard seven bytes of PREAMBLE (0x55) followed by SFD (0xD5). When GMX(0)_TX(0..3)_APPEND[PREAMBLE] = 1, CN71XX prepends the PREAMBLE+SFD, as shown in Figure 24–10.

Note that CN71XX always autogenerates PREAMBLE/SFD for all autogenerated pause/PFC packets in RXAUI mode.

If the PREAMBLE+SFD is not automatically generated by CN71XX (i.e. if GMX(0)_TX(0..3)_APPEND[PREAMBLE] = 0), as might be appropriate when CN71XX is not communicating with a standard PHY, there are several things to note:

1. It is possible for the packet bytes from PKO to include any form/size PREAMBLE/SFD. Care must be taken regarding CRC generation, though, if some form of PREAMBLE/SFD is included in the PKO bytes, as these bytes will be included in any hardware-generated CRC calculation. Refer to Section 24.3.5.4.
2. In its conversion from expanded packet data to code groups (refer to Section 24.3.5.6), CN71XX replaces the first byte of expanded packet data with an /S/. The remote receiver may replace the /S/ with a packet data byte. (Figure 24–8 and Section 24.3.4.1 describe how CN71XX receive logic converts the /S/ to 0xFB.) Section 24.3.5.4 covers the FCS generation issues that arise in this situation.
3. If HiGig is not enabled (i.e. GMX(0)_TX_XAUI_CTL[HG_EN] = 0), CN71XX may not be able to autogenerate pause/PFC packets since it always autogenerates PREAMBLE/SFD for all pause/PFC packets it autogenerates in this case. The remote receiver may not be able to interpret the pause/PFC packet if it is not expecting PREAMBLE/SFD. Pause/PFC packets and their autogeneration are described in Sections 24.3.2.4 and 24.3.2.5 and Figures 24–5 and 24–6.

24.3.5.6 Transmit Conversion to Code Groups

Fourth, the CN71XX RXAUI transmit logic converts the expanded packet data into code groups, as Figure 24–10 shows.

CN71XX replaces the first byte of the expanded and packet data with the start code group (/S/).

As Figure 24–10 shows, CN71XX follows each packet with a terminate code group (/T/), and fills out the remainder of the column with idles (always /K/s). If the expanded packet data is a multiple of 4 bytes, the terminate and idle code group all exist in a following column.

24.3.6 Unidirectional Transport

CN71XX implements the reconciliation layer modifications described in clause 66.3 of the IEEE 802.3-2005 specification. When GMX(0)_TX_XAUI_CTL[UNI_EN] = 1, unidirectional capability is enabled, allowing CN71XX to transmit packets independent of the receive link status.

Cavium Confidential For Bowen Yang ALU 09/17/2014

24.4 Wake-on-LAN (WOL)

CN71XX supports power-savings features in which the network can wake the core from a power-savings state. This allows the core to be placed in a low-power state, but still allow remote and networked administration of the device. The CN71XX core can be awakened either by directed packets or by magic packets.

24.4.1 Packet Detection

AGX supports WOL via four types of packets:

- directed exact match
- directed multicast match
- directed broadcast match
- magic packet match

When operating in wake-on-LAN mode, the configured MAC port unconditionally drops all received packets. The MAC will raise an interrupt to indicate that either a directed or magic packet has been detected and the system is requested to exit the low-power state.

24.4.1.1 Directed-Packet Match

When a MAC port is configured for direct-packet-match mode, all received packets are applied to the DMAC address filter as defined by dmac_addr_filter function.

- When the function returns ACCEPT, an interrupt is asserted.
- No action is taken upon REJECT.

Care must be taken when GMX(0)_RX(0..3)_ADR_CTL[CAM_MODE] = 0, as the DMAC filter is operating in inversion mode. AGX has 32 DMAC registers (shared among the four ports) to use for exact address matches for either directed or directed multicast DMAC.

24.4.1.2 Magic-Packet Match

When a MAC port is configured for magic-packet-match mode, all received packets are parsed for the magic-data pattern. The pattern consists of six synchronization bytes of 0xFF followed by the GMX(0)_SMAC(0..3)[SMAC<47:0>] repeated 16 times.

The pattern can be located anywhere in the packet once the DMAC, SMAC, and TYPE are parsed. The pattern must not be interrupted for magic-packet detection to occur. An example magic packet might look like the following:

```
UDD_SKIP
DMAC
SMAC
TYPE
[optional bytes]
FF FF FF FF FF FF   // Synchronization
SMAC0
SMAC1
SMAC2
...
SMAC15
[optional bytes]
FCS
```

Magic packets are not required to have a good FCS to have a match.

Cavium Confidential For Bowen Yang 02/7/2014

24.4.2 Operation

When software determines it wants to enter WOL mode, it sets GMX(0)_WOL_CTL[EN] = 1.

This places the MAC in WOL mode, which causes the MAC to drop incoming packets. The drop mechanism uses the RX FIFO full path. Packets in that have been partially buffered in the RX FIFO when WOL mode is entered are terminated with the PARTIAL opcode error. All subsequent incoming packets will be completely dropped. Packets will continue to be dropped for the duration of the WOL event (i.e. while GMX(0)_WOL_CTL[EN] = 1).

Once GMX(0)_WOL_CTL[EN] is set, software can reconfigure the DMAC filtering, UDD_SKIP, and MAGIC packet control to achieve the desired packet matching. Systems can choose to use either directed-packet match, magic-packet match, or both. Multiple MACs can be enabled for WOL matching simultaneously and have distinct matching modes and criteria.

Software can then begin WOL-event detection on a individual port by setting GMX(0)_WOL_CTL[DIRECT_EN<prt>/MAGIC_EN<prt>] = 1 .

When a WOL event occurs, it asserts the interrupt bit, GMX(0)_RX(0..3)_INT_REG[WOL] and notifies CIU if the corresponding enable bit is set (GMX(0)_RX(0..3)_INT_EN[WOL].

The MAC will continue to drop packets. The interrupt handler can put the MACs back into normal operation and deassert GMX(0)_WOL_CTL[EN]; packets will then begin flowing in.

The CAMs can be disabled in directed-packet mode by configuring the DCAM CAM properly.

GMX(0)_RX(0..3)_ADR_CTL[CAM_MODE] = 1
GMX(0)_RX(0..3)_ADR_CTL[MCST] = 1
GMX(0)_RX(0..3)_ADR_CTL[BCST] = 0
GMX(0)_RX(0..3)_ADR_CAM_ALL_EN[EN] = 0

Cavium Confidential For Bowen Yang ALU 02/27/2014

24.5 Errors/Exceptions

CN71XX implements many error and exception checks on the RXAUI interfaces. The following subsections describe the exception checkers and how the exception is communicated to software.

Many of these exceptions can cause interrupts:

- GMX(0)_RX(0..3)_INT_REG interrupts can optionally be enabled via corresponding bits in GMX(0)_RX(0..3)_INT_EN.
- PCSX(0)_INT_REG interrupts can optionally be enabled via corresponding bits in PCSX(0)_INT_EN_REG.
- GMX(0)_TX_INT_REG interrupts can optionally be enabled via corresponding bits in GMX(0)_TX_INT_EN.

Chapter 13 covers how these interrupts can be distributed system-wide.

24.5.1 Receive Errors/Exception Checks

Many of the exceptions can be enabled by setting the appropriate bit in GMX(0)_RX(0..3)_FRM_CHK. If the bit is set, the exception is logged in GMX(0)_RX(0..3)_INT_REG. In addition, receive errors can cause the work-queue entry associated with received packets to indicate a receive error (work-queue-entry WORD2[RE] = 1) with the opcode (WORD2[Opcode]) set to indicate the relevant exception. Chapter 9 and Section 9.5 describe the work queue entries that CN71XX creates for arriving packets.

PCSX(0)_INT_REG contains a number of additional error interrupts, some of which should never assert in normal operation.

Table 24–2 lists the receive exceptions, their causes, and how they are handled.

Table 24–2 Receive Errors/Exceptions

Exception	Cause	Notification
JABBER	A packet was received with length > GMX(0)_RX(0..3)_JABBER bytes.	The packet is unconditionally truncated at GMX(0)_RX(0..3)_JABBER bytes. GMX(0)_RX(0..3)_FRM_CHK[JABBER] enables the check. If enabled, GMX(0)_RX(0..3)_INT_REG[JABBER] is set to 1 and work-queue-entry WORD2[OPCODE] is set to 0x2.
RCVERR	A packet was received by the MAC with one or more error control characters. This can result from loss of synchronization or alignment on the SerDes interface or a bad received code group while receiving a packet (refer to ALGNLOS, SYNLOS, and BITLCKLOS below in this table), the packet was improperly terminated, or if the remote transmitter propagates a transmit error.	GMX(0)_RX(0..3)_FRM_CHK[RCVERR] enables the check. If enabled, GMX(0)_RX(0..3)_INT_REG[RCVERR] is set to 1 and work-queue-entry WORD2[OPCODE] is set to 0xB.
FCSERR	A packet was received that failed the FCS/CRC check. This error is likely to occur when a RCVERR occurs.	GMX(0)_RX(0..3)_FRM_CHK[FCSERR] enables the check. If enabled, GMX(0)_RX(0..3)_INT_REG[FCSERR] is set to 1 and work-queue-entry WORD2[OPCODE] is set to 0x3, 0x6 or 0x7.

Cavium Confidential For Bowen Yang only

Table 24–2 Receive Errors/Exceptions (Continued)

Exception	Cause	Notification
SKPERR	A packet was received in which the length < GMX(0)_RX(0..3)_UDD_SKP[LEN] bytes. This means that there was not enough packet data in order to get past the Skip field and into the L2 HDR.	GMX(0)_RX(0..3)_FRM_CHK[SKPERR] enables the check. If enabled, GMX(0)_RX(0..3)_INT_REG[SKPERR] is set to 1 and work-queue-entry WORD2[OPCODE] is set to 0xC.
PCTERR	A packet with a bad preamble was received. Refer to Section 24.3.3.1. CN71XX drops the packet in this case. This error can also occur when a bad code group is received when not in the middle of a packet/frame: • non-/S/, non-idle control code group in lane 0 • any non-idle control code group in lanes 1-3 • any data code group in any lane PCTERR will never occur due to valid fault (XGMII sequence). Please note that the PCS layer may generate faults as the result of bad code groups.	GMX(0)_RX(0..3)_FRM_CTL[PRE_CHK] enables the preamble checker. If enabled, GMX(0)_RX(0..3)_INT_REG[PCTERR] is set to 1 for a bad preamble. The bad code group check is always enabled. GMX(0)_RX(0..3)_INT_REG[PCTERR] is always set for a bad code group. Whenever it sets GMX(0)_RX(0..3)_INT_REG[PCTERR], CN71XX loads GMX(0)_RX_XAUI_BAD_COL with useful debug information. The packet is dropped.
RSVERR	GMX reconciliation logic received a reserved control code group. If this error occurs in the middle of a packet, it will cause a BAD_TERM error (see below).	This check is always enabled and sets GMX(0)_RX(0..3)_INT_REG[RSVERR] to 1.
PAUSE_DRP	A pause packet was dropped due to full GMX RX FIFO.	This check is always enabled and sets GMX(0)_RX(0..3)_INT_REG[PAUSE_DRP] to 1. It also loads GMX(0)_RX(0..3)_PAUSE_DROP_TIME[STATUS] with pause_time of the pause packet.
LOC_FAULT	PCSX is out of reset and the receiver is not code group aligned. Per 802.3 specifications, when this happens, CN71XX transmits remote fault-ordered sets. (If GMX(0)_TX_XAUI_CTL[UNI_EN] = 1, packets are also transferred.) This happens on a SYNLOS or BITLCKLOS error (see below). If this error occurs in the middle of a packet, it will cause a BAD_TERM error (see below).	This check is always enabled and sets GMX(0)_RX(0..3)_INT_REG[LOC_FAULT] to one. PCSX(0)_INT_REG[ALGNLOS] is also set on a transition from aligned to unaligned (i.e. is edge-triggered).
REM_FAULT	A remote fault ordered set was received, indicating that the remote transmitter detected a local fault. Per 802.3 specifications, when CN71XX is receiving remote faults and GMX(0)_TX_XAUI_CTL[UNI_EN] = 0, it transmits idles rather than packet data. If this error occurs in the middle of a packet, it will cause a BAD_TERM error (see below).	This check is always enabled and sets GMX(0)_RX(0..3)_INT_REG[REM_FAULT] to 1.
BAD_SEQ	A reserved sequence was received. If this error occurs in the middle of a packet, it will cause a BAD_TERM error (see below).	This check is always enabled and sets GMX(0)_RX(0..3)_INT_REG[BAD_SEQ] to 1.
BAD_TERM	The GMX reconciliation layer logic received a frame that was not terminated by a /T/. (The frame was terminated by a control code other than /T/ or /E/.)	This check is always enabled and sets GMX(0)_RX(0..3)_INT_REG[BAD_TERM] to 1 and work-queue-entry WORD2[OPCODE] to 0x9.

Cavium Confidential For Bowen Yang ALU 02/17/2014

Table 24–2 Receive Errors/Exceptions (Continued)

Exception	Cause	Notification
SKPERR	A packet was received in which the length < GMX(0)_RX(0..3)_UDD_SKP[LEN] bytes. This means that there was not enough packet data in order to get past the Skip field and into the L2 HDR.	GMX(0)_RX(0..3)_FRM_CHK[SKPERR] enables the check. If enabled, GMX(0)_RX(0..3)_INT_REG[SKPERR] is set to 1 and work-queue-entry WORD2[OPCODE] is set to 0xC.
PCTERR	A packet with a bad preamble was received. Refer to Section 24.3.3.1. CN71XX drops the packet in this case. This error can also occur when a bad code group is received when not in the middle of a packet/frame: • non-/S/, non-idle control code group in lane 0 • any non-idle control code group in lanes 1-3 • any data code group in any lane PCTERR will never occur due to valid fault (XGMII sequence). Please note that the PCS layer may generate faults as the result of bad code groups.	GMX(0)_RX(0..3)_FRM_CTL[PRE_CHK] enables the preamble checker. If enabled, GMX(0)_RX(0..3)_INT_REG[PCTERR] is set to 1 for a bad preamble. The bad code group check is always enabled. GMX(0)_RX(0..3)_INT_REG[PCTERR] is always set for a bad code group. Whenever it sets GMX(0)_RX(0..3)_INT_REG[PCTERR], CN71XX loads GMX(0)_RX_XAUI_BAD_COL with useful debug information. The packet is dropped.
RSVERR	GMX reconciliation logic received a reserved control code group. If this error occurs in the middle of a packet, it will cause a BAD_TERM error (see below).	This check is always enabled and sets GMX(0)_RX(0..3)_INT_REG[RSVERR] to 1.
PAUSE_DRP	A pause packet was dropped due to full GMX RX FIFO.	This check is always enabled and sets GMX(0)_RX(0..3)_INT_REG[PAUSE_DRP] to 1. It also loads GMX(0)_RX(0..3)_PAUSE_DROP_TIME[STATUS] with pause_time of the pause packet.
LOC_FAULT	PCSX is out of reset and the receiver is not code group aligned. Per 802.3 specifications, when this happens, CN71XX transmits remote fault-ordered sets. (If GMX(0)_TX_XAUI_CTL[UNI_EN] = 1, packets are also transferred.) This happens on a SYNLOS or BITLCKLOS error (see below). If this error occurs in the middle of a packet, it will cause a BAD_TERM error (see below).	This check is always enabled and sets GMX(0)_RX(0..3)_INT_REG[LOC_FAULT] to one. PCSX(0)_INT_REG[ALGNLOS] is also set on a transition from aligned to unaligned (i.e. is edge-triggered).
REM_FAULT	A remote fault ordered set was received, indicating that the remote transmitter detected a local fault. Per 802.3 specifications, when CN71XX is receiving remote faults and GMX(0)_TX_XAUI_CTL[UNI_EN] = 0, it transmits idles rather than packet data. If this error occurs in the middle of a packet, it will cause a BAD_TERM error (see below).	This check is always enabled and sets GMX(0)_RX(0..3)_INT_REG[REM_FAULT] to 1.
BAD_SEQ	A reserved sequence was received. If this error occurs in the middle of a packet, it will cause a BAD_TERM error (see below).	This check is always enabled and sets GMX(0)_RX(0..3)_INT_REG[BAD_SEQ] to 1.
BAD_TERM	The GMX reconciliation layer logic received a frame that was not terminated by a /T/. (The frame was terminated by a control code other than /T/ or /E/.)	This check is always enabled and sets GMX(0)_RX(0..3)_INT_REG[BAD_TERM] to 1 and work-queue-entry WORD2[OPCODE] to 0x9.

Cavium Confidential For Bowen Yang ALU 02/17/2014

Table 24–2 Receive Errors/Exceptions (Continued)

Exception	Cause	Notification
SYNLOS	Code group synchronization is lost by PCSX. This happens on a BITLCKLOS error (see below).	This check is always enabled and PCSX(0)_INT_REG[SYNLOS] on a transition from synchronized to unsynchronized (i.e. it is edge-triggered).
BITLCKLOS	Bit lock is lost on one or more lanes by PCSX.	This check is always enabled and sets PCSX(0)_INT_REG[BITLCKLOS] on a transition from all locked to not all locked (i.e. it is edge-triggered).
UNSOP UNEOP UNDAT	GMX has several internal error checks. These errors should not occur during normal operation.	These checks are always enabled and sets GMX(0)_RX(0..3)_INT_REG[UNSOP,UNEOP,UNDAT] to 1.
RXSYNBAD RXBAD	PCSX has several internal error checks. These errors should not occur during normal operation.	These checks are always enabled and set PCSX(0)_INT_REG[RXSYNBAD,RXBAD] to 1.
HG2CC	CN71XX received a HiGig2 message that had a bad CRC8 or an unexpected control character when GMX(0)_HG2_CONTROL[HG2RX_EN]=1.	These checks are always enabled when GMX(0)_HG2_CONTROL[HG2RX_EN] = 1. The message is discarded and GMX(0)_RX(0..3)_INT_REG[HG2CC] is set to one.
HG2FLD	CN71XX received a HiGig2 message that was neither a physical-link nor logical-link message.	These checks are always enabled when GMX(0)_HG2_CONTROL[HG2RX_EN] = 1. The message is discarded and GMX(0)_RX(0..3)_INT_REG[HG2FLD] is set to 1.

24.5.2 Transmit Errors/Exception Checks

Some of the transmit errors are logged in GMX(0)_TX_INT_REG.

Additionally, GMX(0)_BAD_REG contains a number of error conditions that should not happen during normal operation.

Table 24–3 lists the exceptions, their causes, and how they are handled.

Table 24–3 Transmit Errors/Exceptions

Exception	Cause	Notification
PKO_NXA	The RXAUI transmitter received a request to send data out a port other than zero for the interface.	CN71XX asserts GMX(0)_TX_INT_REG[PKO_NXA] and logs the offending port in GMX(0)_NXA_ADR.
PKO_NXP	The RXAUI transmitter received a request to send data with an assigned pipe value that is either 0 or out-of-range as assigned by GMX(0..4)_TX(0..3)_PIPE.	CN71XX asserts GMX(0..4)_TX_INT_REG[PKO_NXA] and logs the offending pipe in GMX(0..4)_NXA_ADR[PIPE].
UNDFLW	In the unlikely event that PKO cannot keep the GMX TX FIFO full, the RXAUI packet transfer will underflow. This should be detected by the receiving device as an FCS error. Internally, the packet is drained and lost.	CN71XX asserts GMX(0)_TX_INT_REG[UNDFLW<0>] and drains the packet.
OUT_OVR LOSTSTAT STATOVR INB_NXA	The GMX transmitter has several internal error checks. These checks never assert in a correctly configured system. Any assertion indicates that an internal problem has occurred.	CN71XX sets corresponding bit in GMX(0)_BAD_REG.

Cavium Confidential For Bowen Yang ALU 02/21/2014

24.5.3 Transmit Error Propagation

If PKO detects an error when reading packet data, the error is passed to the RXAUI transmitter. If GMX(0)_TX_CORRUPT[CORRUPT<0>] is set, CN71XX sends four error propagation control characters (/E/) in place of the FCS in order to notify the receiving device that the packet is corrupted.

Software can detect that PKO detected an error through the PKO_REG_ERROR[PARITY] bit.

24.6 Reset, Initialization, and Link Status Changes

24.6.1 GMX/PCSX Reset and Enables

In addition to chip reset, the PCSX state machines are in reset for a brief amount of time after software writes PCSX(0)_CONTROL1_REG[RESET] = 1. (Software should read PCSX(0)_CONTROL1_REG[RESET] until it is 0 to determine when the reset sequence is complete.)

GMX cannot receive any new input packet data on the interface nor transfer packet data out a port when PCSX is in reset. (The DLM will be transmitting data codes of zero when PCSX is in reset.) If GMX is in the middle of receiving an inbound packet when PCS enters reset, then GMX will force a BAD_TERM to close the packet. Section 24.5.1 describes the BAD_TERM error.

The GMX state machines are disabled when GMX(0)_PRT*n*_CFG[EN] = 0. When a GMX port is disabled by clearing GMX(0)_PRT*n*_CFG[EN] to 0, CN71XX stops sending new packets out the port. Subsequent outbound packets will remain enqueued in the transmit FIFO for the port when a GMX port is disabled this way. (These enqueued packets will be sent out later if GMX is later re-enabled.) If CN71XX is in the middle of sending out a packet when GMX(0)_PRT*n*_CFG[EN] is cleared to 0, that packet is completed in entirety.

When GMX is disabled by clearing GMX(0)_PRT0_CFG[EN] = 0, the GMX receive path stops putting packets into the receive FIFO. If CN71XX is in the middle of receiving a packet when GMX(0)_PRT0_CFG[EN] is cleared to 0, the packet is received in entirety (assuming it is not dropped). New packets are always ignored when GMX(0)_PRT0_CFG[EN] = 0.

When PCSX(0)_MISC_CTL_REG[GMXENO] = 1, the effect is the same on the receive path as when GMX(0)_PRT0_CFG[EN] = 0: the GMX receive path also stops putting received packets into the receive FIFO. If CN71XX is in the middle of receiving a packet when PCSX(0)_MISC_CTL_REG[GMXENO] is set to 1, the packet is received in entirety (assuming it is not dropped). New packets are always ignored after PCSX(0)_MISC_CTL_REG[GMXENO] = 1. Note that PCSX(0)_MISC_CTL_REG[GMXENO] has no effect on transmitted packets. Cavium recommends that PCSX(0)_MISC_CTL_REG[GMXENO] be set to 1 as PCS logic exits reset, up until the point when the link status appears up and stable and GMX is initialized. Section 24.6.2 describes more about link bring-up.

Table 24–4 describes the effect of various enables on CN71XX behavior:

Cavium Confidential For Bowen Fang ARU 10/27/2014

Table 24–4 Description of Different Enable Modes

GMX*_PRT0_CFG[EN]	PCSX*_MISC_CTL_REG[GMXENO]	Description
0	X	CN71XX will not put new packets into the receive FIFO and will not transmit any outbound packets.
1	0	Normal operating mode. CN71XX puts received packets into the receive FIFO and transmits outbound packets that PKO puts in the transmit FIFO or are autogenerated.
1	1	CN71XX will not put new packets into the receive FIFO, but will transmit outbound packets that PKO puts in the transmit FIFO or are autogenerated. As much as possible, this will drain the GMX FIFOs.

24.6.2 Link Bring-Up

This sequence should be used to bring up a CN71XX RXAUI SerDes interface:

1. Enable the interface. Set GMX(0)_INF_MODE[EN] = 1 if not already set.
2. Disable GMX.
   - **2a.** Clear GMX(0)_PRT*_CFG[EN] to 0.
   - **2b.** Wait for GMX(0)_PRT0_CFG[RX_IDLE,TX_IDLE] to both become set, indicating that GMX is idle.

   At this point, GMX is neither sending nor receiving packets.
3. At this point, it may be appropriate to disable all GMX and PCSX interrupts, as a number of them will occur during bring-up of the Link.
   - zero GMX(0)_RX(0..3)_INT_EN
   - zero GMX(0)_TX_INT_EN
4. Next, bring PCSX and the GMX reconciliation layer logic up:
   - **4a.** Take PCSX through a reset sequence. Write PCSX(0)_CONTROL1_REG[RESET] = 1. Read PCSX(0)_CONTROL_REG[RESET] until it changes value to 0. Set PCSX(0)_MISC_CTL_REG[GMXENO] to 1 to disable reception.
   - **4b.** Set the polarity and lane swapping of the DLM SerDes. Refer to Section 24.3.1, PCSX(0)_MISC_CTL_REG[TX_SWAP,RX_SWAP] and PCSX(0)_TX_RX_POLARITY_REG[TXPLRT,RXPLRT].
   - **4c.** Write PCSX(0)_CONTROL1_REG[LO_PWR] = 0.
   - **4d.** Select Deficit Idle Count mode and unidirectional enable/disable via GMX(0)_TX_XAUI_CTL[DIC_EN,UNI_EN].

   While the PCSX is in reset, data code groups of 0 are sent. Immediately once PCSX comes out of reset, a LOC_FAULT (refer to Section 24.5.1) occurs. This normally causes the Remote Fault Sequence to be driven onto the TX bus (only during idle cycles if GMX(0)_TX_XAUI_CTL[UNI_EN] = 1) by the reconciliation layer transmit logic until the link achieves alignment. Note that any receive-polarity and/or receive-lane-swap register changes (that occur in (4b) above) will likely cause PCSX to restart bringup of the RXAUI channel.
5. Check to make sure that the link appears up and stable. The link appears fully up and stable when PCSX(0)*_10GBX_STATUS_REG[ALIGND] = 1 and GMX(0)_RX_XAUI_CTL[STATUS] = 0. Section 24.6.4 discusses more about link status.

Cavium Confidential For Bowen Yang 02/27/2014

6. Configure GMX.

   6a. Program all relevant GMX configuration while GMX(0)_PRT0_CFG[EN] = 0. This includes all things described in this chapter.

   6b. Set GMX(0)_PRT*n*_CFG[EN] to 1 to enable GMX. At this point, GMX can transmit packets, but cannot receive packets since PCSX(0)_MISC_CTL_REG[GMXENO] = 1.

7. At this point, it is appropriate to clear out all the error state that accumulated during link bring-up and re-enable the relevant interrupts.
   - clear out GMX(0)_RX(0..3)_INT_REG
   - clear out GMX(0)_TX_INT_REG
   - clear out PCSX(0)_INT_REG
   - read PCSX(0)_STATUS1_REG[RCV_LNK] until it is set
   - read PCSX(0)_STATUS2_REG[XMTFLT,RCVFLT] until they are clear
   - set GMX(0)_RX(0..3)_INT_EN appropriately
   - set GMX(0)_TX_INT_EN appropriately
   - set PCSX(0)_INT_EN_REG appropriately

8. Enable packet receive by clearing PCSX(0)_MISC_CTL_REG[GMXENO] = 0. This step should only be executed after PIP/IPD have been configured, as packets should not arrive at PIP/IPD until they are fully configured. Refer to Chapter 9.

24.6.3 Link Bring-Down

When bringing the link down, software should immediately set PCSX(0)_MISC_CTL_REG[GMXENO] to 1 to prevent extraneous packets from arriving.

Clear GMX(0)_PRT0_CFG[EN] to 0 if GMX should not be drained. Section 24.6.1 describes the affect of these enables on behavior.

If GMX is being drained (i.e. if GMX(0)_PRT0_CFG[EN] remains asserted), it may also be appropriate to forcibly disable pause frame generation by CN71XX by setting GMX(0)_TX_OVR_BP[EN<0>] to 1, and clearing GMX(0)_TX_OVR_BP[BP<0>] to 0.

24.6.4 Link Status and Changes

The link configuration/status includes the state of the CN71XX PCSX logic and the state of the CN71XX GMX reconciliation layer logic. Note that the state of the reconciliation layer logic is influenced by local faults (i.e. LOC_FAULT's, refer to Section 24.5.1) generated by the CN71XX PCSX logic and by remote faults (i.e. REM_FAULTs, refer to Section 24.5.1).

The state of the CN71XX PCSX logic is indicated by:

- Whether each SerDes lane achieved bit lock (PCSX(0)_BIT_LOCK_STATUS_REG[BITLCK*])
- Whether each SerDes lane achieved code group synchronization (PCSX(0)_10GBX_STATUS_REG[L*SYNC])
- Whether the interface achieved lane alignment (PCSX(0)_10GBX_STATUS_REG[ALIGND])

Lane alignment and code group synchronization are defined in Clause 48 of the IEEE 802.3-2005 specifications. Bit-lock is specific to CN71XX and indicates that the DLM believes it is locked to the received commas.

Cavium Confidential For Bowen Yang ALU 02/27/2014

The state of the CN71XX GMX reconciliation layer receive logic is the link_fault value indicated by GMX(0)_RX_XAUI_CTL[STATUS], and defined in Clauses 46 and 66.3 of the IEEE 802.3-2005 specifications. This state is normally used as the link_fault state by the reconciliation layer transmit logic, but the link status can alternatively be forced using GMX(0)_TX_XAUI_CTL[LS_BYP,LS]. Note that unidirectional mode (i.e GMX(0)_TX_XAUI_CTL[UNI_EN] = 1) also has a substantial effect on the CN71XX reconciliation layer transmit logic.

After the link is up and stable, link status changes will always be indicated by LOC_FAULT's and REM_FAULTs (Refer to Section 24.5.1 above). LOC_FAULTs occur when the CN71XX PCSX logic is out of reset and have not achieved lane alignment, as defined by Clause 48 of the IEEE 802.3-2005 specifications. REM_FAULTs indicate that the link partner's receiver has not achieved lane alignment, as defined by Clauses 46 and 66.3 of the IEEE 802.3-2005 specifications. We recommend that the application receive LOC_FAULT/REM_FAULT interrupts and quickly adjust to link status changes.

24.7 Statistics

CN71XX supports a rich statistics gathering, described in Table 24–5.

Table 24–5 CN71XX RXAUI Statistics

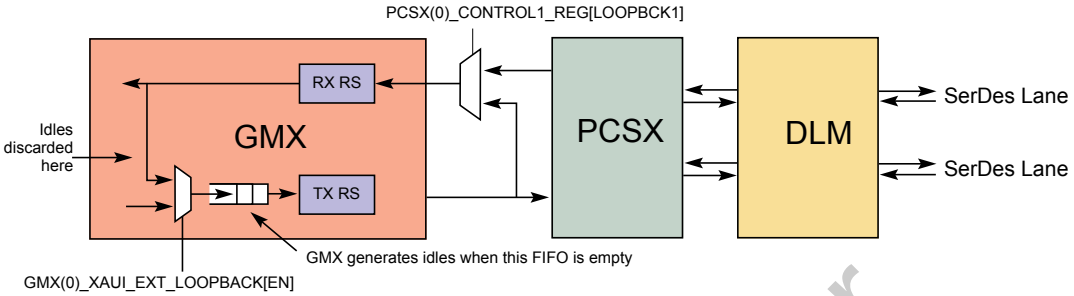
CSR Field	Statistic
RX	
GMX(0)_RX(0..3)_STATS_PKTS[CNT]	32-bit count of all good packets
GMX(0)_RX(0..3)_STATS_OCTS[CNT]	48-bit count of all bytes from good packets
GMX(0)_RX(0..3)_STATS_PKTS_CTL[CNT]	32-bit count of all control/pause packets
GMX(0)_RX(0..3)_STATS_OCTS_CTL[CNT]	48-bit count of all bytes from control/pause packets
GMX(0)_RX(0..3)_STATS_PKTS_DMAC[CNT]	32-bit count of all DMAC filtered packets
GMX(0)_RX(0..3)_STATS_OCTS_DMAC[CNT]	48-bit count of all bytes from DMAC filtered packets
GMX(0)_RX(0..3)_STATS_PKTS_DRP[CNT]	32-bit count of all packets dropped due to RX FIFO full
GMX(0)_RX(0..3)_STATS_OCTS_DRP[CNT]	48-bit count of all bytes from packets dropped due to RX FIFO full
GMX(0)_RX(0..3)_STATS_PKTS_BAD[CNT]	32-bit count of all bad packets
TX	
GMX(0)_TX(0..3)_STAT2[OCTS]	48-bit count of all bytes sent
GMX(0)_TX(0..3)_STAT3[PKTS]	32-bit count of all packets sent
GMX(0)_TX(0..3)_STAT4[HIST0]	32-bit count of packets sent with an octet count < 64
GMX(0)_TX(0..3)_STAT4[HIST1]	32-bit count of packets sent with an octet count = 64
GMX(0)_TX(0..3)_STAT5[HIST2]	32-bit count of packets sent with an octet count of 65–127
GMX(0)_TX(0..3)_STAT5[HIST3]	32-bit count of packets sent with an octet count of 128–255
GMX(0)_TX(0..3)_STAT6[HIST4]	32-bit count of packets sent with an octet count of 256–511
GMX(0)_TX(0..3)_STAT6[HIST5]	32-bit count of packets sent with an octet count of 512–1023
GMX(0)_TX(0..3)_STAT7[HIST6]	32-bit count of packets sent with an octet count of 1024–1518
GMX(0)_TX(0..3)_STAT7[HIST7]	32-bit count of packets sent with an octet count of > 1518
GMX(0)_TX(0..3)_STAT8[BCST]	32-bit count of packets sent to a multicast DMAC
GMX(0)_TX(0..3)_STAT8[MCST]	32-bit count of packets sent to the broadcast DMAC
GMX(0)_TX(0..3)_STAT9[CTL]	32-bit count of control/pause packets sent
GMX(0)_TX(0..3)_STAT9[UNDFLW]	32-bit count of packets sent that experienced a transmit underflow and were truncated

Cavium Confidential For Bowen Yang ALU 04/7/2014

24.8 Loopback

CN71XX RXAUI logic supports both internal and external loopback modes. Figure 24–11 depicts the loopback architecture.

Figure 24–11 PCS Loopback



When PCSX is not in reset, PCSX(0)_CONTROL1_REG[LOOPBCK1] = 1, and GMX is enabled (i.e. GMX(0)_PRT0_CFG[EN] = 1), the CN71XX RXAUI logic implements internal loopback. In this mode, packets that are sent out the GMX transmit FIFO arrive at the GMX receive FIFO. As Figure 24–11 indicates, the internally looped packet data traverses all GMX transmit and receive state machines, including the reconciliation layer logic that is in GMX. The packet data is received at the same rate as transmitted. Note that this internal loopback is not functional when the DLM is not being clocked, as may be the case if the DLM is not used. A PCSX reset sequence should always be used when either or exiting this internal loopback mode. Section 24.6.1 describes how to reset PCSX.

When GMX(0)_XAUI_EXT_LOOPBACK[EN] = 1 (and PCSX is not in reset), the GMX logic loops all packets received on the RXAUI SerDes lanes back out the interface. As Figure 24–11 indicates, this external loopback occurs after all the PCSX receive state machines as well as the GMX receive path reconciliation layer logic. By that point, packet data is captured, but all idles that were present in the input SerDes are discarded. When external loopback is enabled, GMX injects the loopback packets into the transmit path before most of the transmit reconciliation layer logic. (IPG generation is the only transmit reconciliation layer function not retained.) GMX (and, later, PCSX) regenerates all needed idles that separate packets when the depicted FIFO is empty.

Cavium Confidential For Beta Only 02/27/2014

24.9 Enumerations

GMX Interface Mode
GMX_INF_MODE_E

Enumerates the values for GMX(0)_INF_MODE[MODE].

Value	Mnemonic	Description
0x0	DISABLED	The interface is disabled. When GMX0 is configured for RXAUI mode, GMX1 must be disabled.
0x1	SGMII	The interface is in SGMII mode.
0x2	QSGMII	The interface is in QSGMII mode.
0x3	RXAUI	The interface is in RXAUI mode. When GMX0 is configured for RXAUI mode, GMX1 must be disabled. This mode is not a valid operating mode for GMX1.

Cavium Confidential For Bowen Yang ALU 02/27/2014

24.10 GMX Registers

The GMX registers used by the RXAUI interface are listed in Table 24–6.

Table 24–6 GMX Registers

Register	Address[1]	CSR Type[2]	Detailed Description
GMX(0)_RX(0..3)_INT_REG	0x1180008000000 + α*0x8000000 + β*0x800	RSL	See page 1131
GMX(0)_RX(0..3)_INT_EN	0x1180008000008 + α*0x8000000 + β*0x800	RSL	See page 1133
GMX(0)_PRT(0..3)_CFG	0x1180008000010 + α*0x8000000 + β*0x800	RSL	See page 1134
GMX(0)_RX(0..3)_FRM_CTL	0x1180008000018 + α*0x8000000 + β*0x800	RSL	See page 1135
GMX(0)_RX(0..3)_FRM_CHK	0x1180008000020 + α*0x8000000 + β*0x800	RSL	See page 1136
GMX(0)_RX(0..3)_JABBER	0x1180008000038 + α*0x8000000 + β*0x800	RSL	See page 1137
GMX(0)_RX(0..3)_DECISION	0x1180008000040 + α*0x8000000 + β*0x800	RSL	See page 1137
GMX(0)_RX(0..3)_UDD_SKP	0x1180008000048 + α*0x8000000 + β*0x800	RSL	See page 1138
GMX(0)_RX(0..3)_STATS_CTL	0x1180008000050 + α*0x8000000 + β*0x800	RSL	See page 1138
GMX(0)_RX(0..3)_IFG	0x1180008000058 + α*0x8000000 + β*0x800	RSL	See page 1139
GMX(0)_RX(0..3)_PAUSE_DROP_TIME	0x1180008000068 + α*0x8000000 + β*0x800	RSL	See page 1139
GMX(0)_RX(0..3)_STATS_PKTS	0x1180008000080 + α*0x8000000 + β*0x800	RSL	See page 1139
GMX(0)_RX(0..3)_STATS_OCTS	0x1180008000088 + α*0x8000000 + β*0x800	RSL	See page 1140
GMX(0)_RX(0..3)_STATS_PKTS_CTL	0x1180008000090 + α*0x8000000 + β*0x800	RSL	See page 1140
GMX(0)_RX(0..3)_STATS_OCTS_CTL	0x1180008000098 + α*0x8000000 + β*0x800	RSL	See page 1140
GMX(0)_RX(0..3)_STATS_PKTS_DMAC	0x11800080000A0 + α*0x8000000 + β*0x800	RSL	See page 1141
GMX(0)_RX(0..3)_STATS_OCTS_DMAC	0x11800080000A8 + α*0x8000000 + β*0x800	RSL	See page 1141
GMX(0)_RX(0..3)_STATS_PKTS_DRP	0x11800080000B0 + α*0x8000000 + β*0x800	RSL	See page 1141
GMX(0)_RX(0..3)_STATS_OCTS_DRP	0x11800080000B8 + α*0x8000000 + β*0x800	RSL	See page 1142
GMX(0)_RX(0..3)_STATS_PKTS_BAD	0x11800080000C0 + α*0x8000000 + β*0x800	RSL	See page 1142
GMX(0)_RX(0..3)_ADR_CTL	0x1180008000100 + α*0x8000000 + β*0x800	RSL	See page 1143
GMX(0)_RX(0..3)_ADR_CAM_EN	0x1180008000108 + α*0x8000000 + β*0x800	RSL	See page 1144
GMX(0)_RX(0..3)_ADR_CAM_ALL_EN	0x1180008000110 + α*0x8000000 + β*0x800	RSL	See page 1144
GMX(0)_RX(0..3)_ADR_CAM(0..5)	0x1180008000180 + α*0x8000000 + β*0x800 + χ*0x08	RSL	See page 1145
GMX(0)_TX(0..3)_THRESH	0x1180008000210 + α*0x8000000 + β*0x800	RSL	See page 1145
GMX(0)_TX(0..3)_APPEND	0x1180008000218 + α*0x8000000 + β*0x800	RSL	See page 1145
GMX(0)_TX(0..3)_SLOT	0x1180008000220 + α*0x8000000 + β*0x800	RSL	See page 1146
GMX(0)_TX(0..3)_BURST	0x1180008000228 + α*0x8000000 + β*0x800	RSL	See page 1146
GMX(0)_SMAC(0..3)	0x1180008000230 + α*0x8000000 + β*0x800	RSL	See page 1146
GMX(0)_TX(0..3)_PAUSE_PKT_TIME	0x1180008000238 + a*0x8000000 + b*0x800	RSL	See page 1147
GMX(0)_TX(0..3)_MIN_PKT	0x1180008000240 + a*0x8000000 + b*0x800	RSL	See page 1147
GMX(0)_TX(0..3)_PAUSE_PKT_INTERVAL	0x1180008000248 + a*0x8000000 + b*0x800	RSL	See page 1148
GMX(0)_TX(0..3)_SOFT_PAUSE	0x1180008000250 + a*0x8000000 + b*0x800	RSL	See page 1148
GMX(0)_TX(0..3)_PAUSE_TOGO	0x1180008000258 + a*0x8000000 + b*0x800	RSL	See page 1149
GMX(0)_TX(0..3)_PAUSE_ZERO	0x1180008000260 + a*0x8000000 + b*0x800	RSL	See page 1149
GMX(0)_TX(0..3)_STATS_CTL	0x1180008000268 + a*0x8000000 + b*0x800	RSL	See page 1149
GMX(0)_TX(0..3)_CTL	0x1180008000270 + a*0x8000000 + b*0x800	RSL	See page 1149
GMX(0)_TX(0..3)_STAT0	0x1180008000280 + a*0x8000000 + b*0x800	RSL	See page 1150
GMX(0)_TX(0..3)_STAT1	0x1180008000288 + a*0x8000000 + b*0x800	RSL	See page 1150
GMX(0)_TX(0..3)_STAT2	0x1180008000290 + a*0x8000000 + b*0x800	RSL	See page 1150
GMX(0)_TX(0..3)_STAT3	0x1180008000298 + a*0x8000000 + b*0x800	RSL	See page 1151

Cavium Confidential For Bowen Yang @2/7/2014

Table 24–6 GMX Registers

Register	Address[1]	CSR Type[2]	Detailed Description
GMX(0)_TX(0..3)_STAT4	0x11800080002A0 + a*0x8000000 + b*0x800	RSL	See page 1151
GMX(0)_TX(0..3)_STAT5	0x11800080002A8 + a*0x8000000 + b*0x800	RSL	See page 1151
GMX(0)_TX(0..3)_STAT6	0x11800080002B0 + a*0x8000000 + b*0x800	RSL	See page 1152
GMX(0)_TX(0..3)_STAT7	0x11800080002B8 + a*0x8000000 + b*0x800	RSL	See page 1152
GMX(0)_TX(0..3)_STAT8	0x11800080002C0 + a*0x8000000 + b*0x800	RSL	See page 1152
GMX(0)_TX(0..3)_STAT9	0x11800080002C8 + a*0x8000000 + b*0x800	RSL	See page 1153
GMX(0)_BIST	0x1180008000400 + a*0x8000000	RSL	See page 1153
GMX(0)_RX_PRTS	0x1180008000410 + a*0x8000000	RSL	See page 1153
GMX(0)_RX_BP_DROP(0..3)	0x1180008000420 + a*0x8000000 + b*0x8	RSL	See page 1154
GMX(0)_RX_BP_ON(0..3)	0x1180008000440 + α*0x8000000 + β*0x8	RSL	See page 1154
GMX(0)_RX_BP_OFF(0..3)	0x1180008000460 + α*0x8000000 + β*0x8	RSL	See page 1155
GMX(0)_TX_PRTS	0x1180008000480 + α*0x8000000	RSL	See page 1155
GMX(0)_TX_IFG	0x1180008000488 + α*0x8000000	RSL	See page 1155
GMX(0)_TX_JAM	0x1180008000490 + α*0x8000000	RSL	See page 1156
GMX(0)_TX_COL_ATTEMPT	0x1180008000498 + α*0x8000000	RSL	See page 1156
GMX(0)_TX_PAUSE_PKT_DMAC	0x11800080004A0 + α*0x8000000	RSL	See page 1156
GMX(0)_TX_PAUSE_PKT_TYPE	0x11800080004A8 + α*0x8000000	RSL	See page 1156
GMX(0)_TX_OVR_BP	0x11800080004C8 + α*0x8000000	RSL	See page 1157
GMX(0)_TX_BP	0x11800080004D0 + α*0x8000000	RSL	See page 1157
GMX(0)_TX_CORRUPT	0x11800080004D8 + α*0x8000000	RSL	See page 1158
GMX(0)_RX_PRT_INFO	0x11800080004E8 + α*0x8000000	RSL	See page 1158
GMX(0)_TX_LFSR	0x11800080004F8 + α*0x8000000	RSL	See page 1158
GMX(0)_TX_INT_REG	0x1180008000500 + α*0x8000000	RSL	See page 1159
GMX(0)_TX_INT_EN	0x1180008000508 + α*0x8000000	RSL	See page 1159
GMX(0)_NXA_ADR	0x1180008000510 + α*0x8000000	RSL	See page 1160
GMX(0)_BAD_REG	0x1180008000518 + α*0x8000000	RSL	See page 1160
GMX(0)_TX_XAUI_CTL	0x1180008000528 + α*0x8000000	RSL	See page 1161
GMX(0)_RX_XAUI_CTL	0x1180008000530 + α*0x8000000	RSL	See page 1161
GMX(0)_RX_XAUI_BAD_COL	0x1180008000538 + α*0x8000000	RSL	See page 1162
GMX(0)_XAUI_EXT_LOOPBACK	0x1180008000540 + α*0x8000000	RSL	See page 1162
GMX(0)_RX_HG2_STATUS	0x1180008000548 + α*0x8000000	RSL	See page 1162
GMX(0)_HG2_CONTROL	0x1180008000550 + α*0x8000000	RSL	See page 1163
GMX(0)_TX_HG2_REG1	0x1180008000558 + α*0x8000000	RSL	See page 1163
GMX(0)_TX_HG2_REG2	0x1180008000560 + α*0x8000000	RSL	See page 1164
GMX(0)_PRT0_CBFC_CTL	0x1180008000580 + α*0x8000000 + β*0x8	RSL	See page 1164
GMX(0)_TX0_CBFC_XOFF	0x11800080005A0 + α*0x8000000 + β*0x8	RSL	See page 1165
GMX(0)_TX0_CBFC_XON	0x11800080005C0 + α*0x8000000 + β*0x8	RSL	See page 1165
GMX(0)_RXAUI_CTL	0x1180008000740 + α*0x8000000	RSL	See page 1165
GMX(0)_WOL_CTL	0x1180008000780 + α*0x8000000	RSL	See page 1165
GMX(0)_INF_MODE	0x11800080007F8 + α*0x8000000	RSL	See page 1166

1. For registers with multiple versions, α represents the values within the specified range, β the second range and χ the third range.
2. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang 02/21/2014

GMX Interrupt Registers

GMX(0)_RX(0..3)_INT_REG

This register allows interrupts to be sent to the control processor. If an error is noted in this register, and the corresponding bit in GMX(0)_RX(0..3)_INT_EN is set, an exception is raised. In RXAUI mode, port 0 is used for interrupt logging.

- Exception conditions <10:0> can also set the rcv/opcode in the received packet's workQ entry. GMX(0)_RX(0..3)_FRM_CHK provides a bit mask for configuring which conditions set the error.
- In half duplex operation, the expectation is that collisions will appear as either MINERR or CAREXT errors.

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:30>	—	RAZ	—	—	Reserved
<29>	WOL	R/W1C/H	0	0	A wake-on-LAN event has occurred.
<28>	HG2CC	R/W1C/H	0	0	HiGig2 received message CRC or control character error. Set when either a CRC8 error is detected or when a control character is found in the message bytes after the K.SOM. **NOTE:** HG2CC has higher priority than HG2FLD, i.e. a HiGig2 message that results in HG2CC getting set never sets HG2FLD.
<27>	HG2FLD	R/W1C/H	0	0	HiGig2 received message field error, as follows: **1.** MSG_TYPE field is not 0x00, i.e. it is not a FLOW CONTROL message, which is the only defined type for HiGig2. **2.** FWD_TYPE field is not 0x0, i.e. Link Level message, which is the only defined type for HiGig2. **3.** FC_OBJECT field is neither 0x0 for Physical Link nor 0x2 for Logical Link, which are the only two defined types in HiGig2.
<26>	UNDAT	R/W1C/H	0	0	Unexpected data. (RXAUI mode only.)
<25>	UNEOP	R/W1C/H	0	0	Unexpected EOP. (RXAUI mode only.)
<24>	UNSOP	R/W1C/H	0	0	Unexpected SOP. (RXAUI mode only.)
<23>	BAD_TERM	R/W1C /H	0	0	Frame is terminated by a control character other than /T/. The error propagation control character /E/ is included as part of the frame and does not cause a frame termination. (RXAUI mode only.)
<22>	BAD_SEQ	R/W1C/H	0	0	Reserved sequence detected. (RXAUI mode only.)
<21>	REM_FAULT	R/W1C/H	0	0	Remote fault sequence detected. (RXAUI mode only.)
<20>	LOC_FAULT	R/W1C/H	0	0	Local fault sequence detected. (RXAUI mode only.)
<19>	PAUSE_DRP	R/W1C/H	0	0	Pause packet was dropped due to full GMX RX FIFO.
<18:16>	—	RAZ	—	—	Reserved
<15>	IFGERR	R/W1C/H	0	0	Interframe gap violation. Does not necessarily indicate a failure. (SGMII/QSGMII/1000BASE-X only.)
<14>	COLDET	R/W1C/H	0	0	Collision detection. Collisions can only occur in half-duplex mode. A collision is assumed by the receiver when the slottime (GMX(0)_PRT(0..3)_CFG[SLOTTIME]) is not satisfied. In 10/100 mode, this will result in a frame < SLOTTIME. In 1000 mode, it could result either in frame < SLOTTIME or a carrier extend error with the SLOTTIME. These conditions are visible by 1) transfer ended before slottime – COLDET or 2) carrier extend error – CAREXT. **NOTE:** RXAUI interfaces are inherently full-duplex so collisions cannot occur.
<13>	FALERR	R/W1C/H	0	0	False-carrier error, or carrier-extend error after slottime is satisfied. (SGMII/QSGMII/1000BASE-X only.)

Cavium Confidential For Bowei Yan ALU 04/22/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<12>	RSVERR	R/W1C/H	0	0	Reserved opcode error.
<11>	PCTERR	R/W1C/H	0	0	Bad preamble / protocol error. Checks that the frame begins with a valid PREAMBLE sequence. Does not check the number of PREAMBLE cycles. In RXAUI mode, the column of data that was bad is logged in GMX(0)_RX_XAUI_BAD_COL.
<10>	OVRERR	R/W1C/H	0	0	Internal data aggregation overflow error. This interrupt should never assert. (SGMII/QSGMII/1000BASE-X only.)
<9>	—	RAZ	—	—	Reserved
<8>	SKPERR	R/W1C/H	0	0	Skipper error.
<7>	RCVERR	R/W1C/H	0	0	Data-reception error. Frame was received with data-reception error
<6:5>	—	RAZ	—	—	Reserved
<4>	FCSERR	R/W1C/H	0	0	FCS/CRC error. Frame was received with FCS/CRC error
<3>	JABBER	R/W1C/H	0	0	System-length error: frame was received with length > sys_length. An RX Jabber error indicates that a packet was received which is longer than the maximum allowed packet as defined by the system. GMX truncates the packet at the JABBER count. Failure to do so could lead to system instability.
<2>	—	RAZ	—	—	Reserved
<1>	CAREXT	R/W1C/H	0	0	Carrier-extend error. (SGMII/QSGMII/1000BASE-X only.)
<0>	MINERR	R/W1C	0	—	Pause frame was received with length < minFrameSize. Frame length checks are typically handled in PIP (PIP_INT_REG[MINERR]), but pause frames are normally discarded before being inspected by PIP. Total frame DA+SA+TL+DATA+PAD+FCS < 64.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GMX Interrupt-Enable Registers

GMX(0)_RX(0..3)_INT_EN

This register provides the interrupt-enable bits for the corresponding error in GMX(0)_RX(0..3)_INT_REG. In RXAUI mode, port 0 is used for checking. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:30>	—	RAZ	—	—	Reserved
<29>	WOL	R/W/H	0	0	A wake-on-LAN event has occurred.
<28>	HG2CC	R/W	0	0	HiGig2 CRC8 or control-charactererror interrupt enable.
<27>	HG2FLD	R/W	0	0	HiGig2 badfielderror interrupt enable.
<26>	UNDAT	R/W	0	0	Unexpected data. (RXAUI mode only.)
<25>	UNEOP	R/W	0	0	Unexpected EOP. (RXAUI mode only.)
<24>	UNSOP	R/W	0	0	Unexpected SOP. (RXAUI mode only.)
<23>	BAD_TERM	R/W	0	0	Frame is terminated by a control character other than /T/. The error propagation control character /E/ is included as part of the frame and does not cause a frame termination. (RXAUI mode only.)
<22>	BAD_SEQ	R/W	0	0	Reserved sequence detected. (RXAUI mode only.)
<21>	REM_FAULT	R/W	0	0	Remote fault sequence detected. (RXAUI mode only.)
<20>	LOC_FAULT	R/W	0	0	Local fault sequence detected. (RXAUI mode only.)
<19>	PAUSE_DRP	R/W	0	0	Pause packet was dropped due to full GMX RX FIFO.
<18:16>	—	RAZ	—	—	Reserved
<15>	IFGERR	R/W	0x0	0	Interframe-gap violation. (SGMII/QSGMII/1000BASE-X only.)
<14>	COLDET	R/W	0x0	0	Collision detection. (SGMII/QSGMII/1000BASE-X only.)
<13>	FALERR	R/W	0x0	0	False-carrier error, or carrier-extend error after slottime is satisfied. (SGMII/QSGMII/1000BASE-X only.)
<12>	RSVERR	R/W	0x0	0	Reserved opcodes.
<11>	PCTERR	R/W	0x0	0	Bad preamble / protocol.
<10>	OVRERR	R/W	0x0	0	Internal data aggregation overflow error. (SGMII/QSGMII/1000BASE-X only.)
<9>	—	RAZ	—	—	Reserved
<8>	SKPERR	R/W	0x0	0	Skipper error.
<7>	RCVERR	R/W	0x0	0	Frame was received with data reception error.
<6:5>	—	RAZ	—	—	Reserved
<4>	FCSERR	R/W	0x0	0	Frame was received with FCS/CRC error.
<3>	JABBER	R/W	0x0	0	Frame was received with length > sys_length.
<2>	—	RAZ	—	—	Reserved
<1>	CAREXT	R/W	0x0	0	Carrier extend error. (SGMII/QSGMII/1000BASE-X only.)
<0>	MINERR	R/W	0	0	Pause frame was received with length < minFrameSize.

Cavium Confidential For Bowen Fang ALU 02/27/2014

GMX Port Configuration Registers
GMX(0)_PRT(0..3)_CFG

This register controls the configuration of the port. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved
<13>	TX_IDLE	RO/H	1	—	TX machine is idle.
<12>	RX_IDLE	RO/H	1	—	RX machine is idle.
<11:9>	—	RAZ	—	—	Reserved
<8>	SPEED_MSB	R/W	0	—	Link speed MSB [SPEED_MSB:SPEED] 10 = 10 Mbs operation 00 = 100 Mbs operation 01 = 1000 Mbs operation 11 = Reserved (SGMII/QSGMII/1000BASE-X only.)
<7:4>	—	RAZ	—	—	Reserved
<3>	SLOTTIME	R/W	1	—	Slot time for half-duplex operation 0 = 512 bit times (10/100 Mbs operation) 1 = 4096 bit times (1000 Mbs operation) (SGMII/QSGMII/1000BASE-X only.)
<2>	DUPLEX	R/W	1	—	Duplex mode: 0 = half-duplex (collisions/extensions/bursts), 1 = full-duplex (SGMII/QSGMII/1000BASE-X only.)
<1>	SPEED	R/W	1	—	Link Speed LSB [SPEED_MSB:SPEED] 10 = 10 Mbs operation 00 = 100 Mbs operation 01 = 1000 Mbs operation 11 = Reserved (SGMII/QSGMII/1000BASE-X only.)
<0>	EN	R/W	0	—	Link enable. When EN is clear, packets are not received or transmitted (including PAUSE and JAM packets). If EN is cleared while a packet is currently being received or transmitted, the packet is allowed to complete before the bus is idled. On the RX side, subsequent packets in a burst are ignored.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Frame Control Registers
GMX(0)_RX(0..3)_FRM_CTL

This register controls the handling of the frames.

The CTL_BCK/CTL_DRP bits control how the hardware handles incoming PAUSE packets. The most common modes of operation:

- CTL_BCK = 1, CTL_DRP = 1: hardware handles everything
- CTL_BCK = 0, CTL_DRP = 0: software sees all pause frames
- CTL_BCK = 0, CTL_DRP = 1: all pause frames are completely ignored

These control bits should be set to CTL_BCK = 0,CTL_DRP = 0 in half-duplex mode. Since PAUSE packets only apply to full duplex operation, any PAUSE packet would constitute an exception which should be handled by the processing cores. PAUSE packets should not be forwarded.

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	—	—	Reserved
<12>	PTP_MODE	R/W	0	0	Timestamp mode. When PTP_MODE is set, a 64-bit timestamp is prepended to every incoming packet. See Section 14.4. The timestamp bytes are added to the packet in such a way as to not modify the packet's receive byte count. This implies that the GMX(0)_RX(0..3)_JABBER, MINERR, GMX(0)_RX(0..3)_DECISION, GMX(0)_RX(0..3)_UDD_SKP, and the GMX_RX_STATS_* do not require any adjustment as they operate on the received packet size. When the packet reaches PIP, its size will reflect the additional bytes and is subject to the restrictions below. If PTP_MODE = 1 and PRE_CHK = 1, PRE_STRP must be 1. If PTP_MODE = 1, PIP_PRT_CFG(0/1..3,16..19,32..3943)[SKIP] should be increased by 8, PIP_PRT_CFG(0/1..3,16..19,32..3943)[HIGIG_EN] should be 0, PIP_FRM_LEN_CHK0[MAXLEN] should be increased by 8, PIP_FRM_LEN_CHK0[MINLEN] should be increased by 8, PIP_TAG_INC(0..63)[EN] should be adjusted, and PIP_PRT_CFGB(0/1..3,16..19,32..39)[ALT_SKP_EN] should be 0.
<11>	—	RAZ	—	—	Reserved
<10>	NULL_DIS	R/W	0	0	When set, do not modify the MOD bits on NULL ticks due to partial packets.
<9>	PRE_ALIGN	R/W	0	0	When set, PREAMBLE parser aligns the SFD byte regardless of the number of previous PREAMBLE nibbles. In this mode, [PRE_STRP] should be set to account for the variable nature of the PREAMBLE. [PRE_CHK] must be set to enable this and all PREAMBLE features. (SGMII at 10/100Mbs only)
<8:7>	—	RAZ	—	—	Reserved
<6>	PRE_FREE	RO	1	1	When set, PREAMBLE checking is less strict. GMX will begin the frame at the first SFD. [PRE_CHK] must be set to enable this and all PREAMBLE features. (SGMII/QSGMII/1000BASE-X only.)
<5>	CTL_SMAC	R/W	0	0	Control pause frames can match station SMAC.
<4>	CTL_MCST	R/W	1	1	Control pause frames can match globally assign multicast address.
<3>	CTL_BCK	R/W	1	1	Forward pause information to TX block.
<2>	CTL_DRP	R/W	1	1	Drop control-pause frames.

Cavium Confidential For Power Card ALD 12/7/2012

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<1>	PRE_STRP	R/W	1	1	Strip off the preamble (when present). 0 = PREAMBLE + SFD is sent to core as part of frame 1 = PREAMBLE + SFD is dropped [PRE_CHK] must be set to enable this and all PREAMBLE features. If PTP_MODE=1 and PRE_CHK=1, PRE_STRP must be 1. When PRE_CHK is set (indicating that the PREAMBLE will be sent), PRE_STRP determines if the PREAMBLE+SFD bytes are thrown away or sent to the core as part of the packet. In either mode, the PREAMBLE+SFD bytes are not counted toward the packet size when checking against the MIN and MAX bounds. Furthermore, the bytes are skipped when locating the start of the L2 header for DMAC and Control frame recognition.
<0>	PRE_CHK	R/W	1	1	Check the preamble for correctness. This port is configured to send a valid 802.3 PREAMBLE to begin every frame. GMX checks that a valid PREAMBLE is received (based on PRE_FREE). When a problem does occur within the PREAMBLE sequence, the frame is marked as bad and not sent into the core. The GMX(0)_RX(0..3)_INT_REG[PCTERR] interrupt is also raised. When GMX0_TX_XAUI_CTL[HG_EN] is set, PRE_CHK must be zero. If PTP_MODE=1 and PRE_CHK=1, PRE_STRP must be 1.

Frame Check Registers
GMX(0)_RX(0..3)_FRM_CHK

This register indicates which frame errors will set the ERR bit of the frame. In RXAUI mode, port 0 is used for checking. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8>	SKPERR	R/W	1	1	Skipper error.
<7>	RCVERR	R/W	1	1	Frame was received with data-reception error.
<6:5>	—	RAZ	—	—	Reserved
<4>	FCSERR	R/W	1	1	Frame was received with FCS/CRC error.
<3>	JABBER	R/W	1	1	Frame was received with length > sys_length.
<2>	—	RAZ	—	—	Reserved
<1>	CAREXT	R/W	1	1	Carrier extend error. (SGMII/QSGMII/1000BASE-X only.)
<0>	MINERR	R/W	1	1	Pause frame was received with length < minFrameSize.

Cavium Confidential For Bowen Yang ALU 02/17/2011

GMX Maximum Packet-Size Registers
GMX(0)_RX(0..3)_JABBER

This register specifies the maximum size for packets, beyond which the GMX truncates. In RXAUI mode, port 0 is used for checking. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	CNT	R/W	0x2800	0x2800	Byte count for jabber check. Failing packets set the JABBER interrupt and are optionally sent with opcode = JABBER. GMX truncates the packet to CNT bytes. CNT must be 8-byte aligned such that CNT[2:0] = 000.

Note:

The packet that is sent to the packet-input logic will have an additional eight bytes if GMX(0)_RX(0..3)_FRM_CTL[PRE_CHK] = 1 and GMX(0)_RX(0..3)_FRM_CTL[PRE_STRP] = 0. The maximum packet that will be sent is defined as:

```
max_sized_packet =
GMX(0)_RXn_JABBER[CNT]+
((GMX(0)_RXn_FRM_CTL[PRE_CHK] &
!GMX(0)_RXn_FRM_CTL[PRE_STRP]) × 8)
```

GMX Packet Decision Registers
GMX(0)_RX(0..3)_DECISION

This register specifies the byte count used to determine when to accept or to filter a packet. As each byte in a packet is received by GMX, the L2 byte count (i.e. the number of bytes from the beginning of the L2 header (DMAC)) is compared against CNT. In normal operation, the L2 header begins after the PREAMBLE + SFD (GMX(0)_RX(0..3)_FRM_CTL[PRE_CHK] = 1) and any optional UDD skip data (GMX(0)_RX(0..3)_UDD_SKP[LEN]). See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved
<4:0>	CNT	R/W	0x18	0x18	The byte count used to decide when to accept or filter a packet. Refer to Table 24–7.

Notes:

When GMX(0)_RX(0..3)_FRM_CTL[PRE_CHK] = 0, PREAMBLE + SFD are prepended to the packet and require UDD skip length to account for them.

Table 24–7 GMX Decisions

Port Mode	L2 Size[1]	
	<GMX_RX_DECISION bytes (default = 0x24)	≥GMX_RX_DECISION bytes (default = 0x24)
Full Duplex	Accept packet No filtering is applied	Apply filters Accept packet based on DMAC and PAUSE packet filters
Half Duplex	Drop packet Packet is unconditionally dropped	Apply filters Accept packet based on DMAC

1. where L2_size = MAX(0, total_packet_size – GMX(0)_RX(0..3)_UDD_SKP[LEN] – ((GMX(0)_RX(0..3)_FRM_CTL[PRE_CHK] = 1) × 8)

Cavium Confidential For Power King ALU 02/27/2014

GMX User-Defined Data Skip Registers
GMX(0)_RX(0..3)_UDD_SKP

This register specifies the amount of user-defined data (UDD) added before the start of the L2C data. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8>	FCSSEL	R/W	0	0	Include the skip bytes in the FCS calculation 0 = all skip bytes are included in FCS 1 = the skip bytes are not included in FCS When GMX0_TX_XAUI_CTL[HG_EN] is set, this field must be zero. The skip bytes are part of the packet and are sent through the IOI packet interface and are handled by PIP. The system can determine if the UDD bytes are included in the FCS check by using the FCSSEL field, if the FCS check is enabled.
<7>	—	RAZ	—	—	Reserved
<6:0>	LEN	R/W	0x0	0x0	Amount of user-defined data before the start of the L2C data, in bytes. Setting to 0 means L2C comes first; maximum value is 64. LEN must be 0x0 in half-duplex operation. When GMX0_TX_XAUI_CTL[HG_EN] is set, this field must be set to 12 or 16 (depending on HiGig header size) to account for the HiGig header. LEN = 12 selects HiGig/HiGig+; LEN = 16 selects HiGig2.

Notes:

Assume that the PREAMBLE/SFD is always at the start of the frame, even before UDD bytes. In most cases, there will be no preamble in these cases, since it will be packet interface in direct communication with another packet interface (MAC to MAC), without a PHY involved.

In all cases, the UDD bytes are sent through the packet interface as part of the packet. The UDD bytes are never stripped from the actual packet.

If LEN ≠ 0 (i.e. there is some UDD), then length checking should be disabled (i.e. GMX(0)_RX(0..3)_FRM_CHK[LENERR] should be cleared and GMX(0)_RX(0..3)_INT_REG[LENERR] set to 0.

Address filtering and control-packet filtering can be performed as needed.

GMX RX Statistics Control Registers
GMX(0)_RX(0..3)_STATS_CTL

This register controls the receive statistics registers. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	RD_CLR	R/W	0	0	Clear on read. When this bit is set, any receive statistics registers are cleared when read.

Cavium Confidential For Bowman Ang 02/27/2014

GMX Minimum Interframe-Gap Cycles Registers
GMX(0)_RX(0..3)_IFG

This register specifies the minimum number of interframe-gap (IFG) cycles between packets. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	IFG	R/W	0x8	0x8	Min IFG (in IFG×8 bits) between packets used to determine IFGERR. Normally IFG is 96 bits. **NOTE:** Note in some operating modes, IFG cycles can be inserted or removed in order to achieve clock rate adaptation. For these reasons, the default value is slightly conservative and does not check up to the full 96 bits of IFG. (SGMII/QSGMII/1000BASE-X only.)

GMX RX Pause-Packet Drop Time Registers
GMX(0)_RX(0..3)_PAUSE_DROP_TIME

This register extracts the [TIME] field in a PAUSE packet that was dropped due to a GMX RX FIFO full condition. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	STATUS	R/W1C/H	0x0	—	Time extracted from the dropped PAUSE packet.

GMX RX Good Packets Registers
GMX(0)_RX(0..3)_STATS_PKTS

This register provides a count of good received packets. The following are examples of "not good" packets:

- packets recognized as PAUSE packets
- packets dropped due to the DMAC filter
- packets dropped due to FIFO full status
- packets that have opcodes ≠ 0 (FCS, Length, etc.).

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of received good packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/21/2014

GMX RX Good Packets Octet Registers
GMX(0)_RX(0..3)_STATS_OCTS

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	CNT	RC/W/H	0x0	—	Octet count of received good packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

GMX RX Pause Packets Registers
GMX(0)_RX(0..3)_STATS_PKTS_CTL

This register provides a count of all packets received that were recognized as flow-control or pause packets. Note that pause packets with any kind of error are also counted in GMX(0)_RX(0..3)_STATS_PKTS_BAD.

- Pause packets can be optionally dropped or forwarded based on GMX(0)_RX(0..3)_FRM_CTL[CTL_DRP]. This count increments regardless of whether the packet is dropped.
- Pause packets are never counted in GMX(0)_RX(0..3)_STATS_PKTS_BAD.
- Packets dropped due to the DMAC filter are counted in GMX(0)_RX(0..3)_STATS_PKTS_DMAC and not here.

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of received flow-control or pause packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

GMX RX Pause Packets Octet Registers
GMX(0)_RX(0..3)_STATS_OCTS_CTL

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	CNT	RC/W/H	0x0	—	Octet count of received flow-control or pause packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GMX RX DMAC Packets Registers
GMX(0)_RX(0..3)_STATS_PKTS_DMAC

This register provides a count of all packets received that were dropped by the DMAC filter. Packets that match the DMAC are dropped and counted here regardless of whether they were bad packets. These packets are never counted in GMX(0)_RX(0..3)_STATS_PKTS. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of filtered DMAC packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

GMX RX DMAC Packets Octet Registers
GMX(0)_RX(0..3)_STATS_OCTS_DMAC

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	CNT	RC/W/H	0x0	—	Octet count of filtered DMAC packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

GMX RX Overflow Packets Registers
GMX(0)_RX(0..3)_STATS_PKTS_DRP

This register provides a count of all packets received that were dropped due to a full receive FIFO. It counts both partial packets in which there was enough space in the RX FIFO to begin to buffer, and the packet and total drops in which no packet was sent to PIP/IPD. It counts all packets dropped by the FIFO, both good and bad packets received. It does not count packets dropped by the DMAC or PAUSE-packet filters.

Some packets that were not able to satisfy the DECISION_CNT may not actually be dropped by CN71XX, but they will be counted here as if they were dropped.

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of dropped packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang 02/27/2014

GMX RX Overflow Packets Octet Registers GMX(0)_RX(0..3)_STATS_OCTS_DRP

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	CNT	RC/W/H	0x0	—	Octet count of dropped packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

GMX RX Bad Packets Registers GMX(0)_RX(0..3)_STATS_PKTS_BAD

This register provides a count of all packets received with some error that were not dropped either due to the DMAC filter or lack of room in the receive FIFO. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of received packets with opcode ≠ 0

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_RX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Address-Filtering Control Registers
GMX(0)_RX(0..3)_ADR_CTL

See Table 24–6 for the address. In RXAUI mode, only GMX_RX0_ADR_CTL is used. GMX_RX(1..3)_ADR_CTL should not be used.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	CAM_MODE	R/W	0	—	Allow or deny DMAC address filter 0 = reject the packet on DMAC address match 1 = accept the packet on DMAC address match
<2:1>	MCST	R/W	0x0	—	Multicast mode: 0x0 = Use the address-filter CAM 0x1 = Force reject all multicast packets 0x2 = Force accept all multicast packets 0x3 = Reserved
<0>	BCST	R/W	1	—	Accept all broadcast packets.

Algorithm:

Here is some pseudo-code that represents the address-filter behavior.

```
bool dmac_addr_filter(uint8 prt, uint48 dmac) {
  ASSERT(prt >= 0 && prt <= 3);
  if (is_bcst(dmac))                              // broadcast accept
    return (GMX_RX{prt}_ADR_CTL[BCST] ? ACCEPT : REJECT);
  if (is_mcst(dmac) & GMX_RX{prt}_ADR_CTL[MCST] == 1)   // multicast reject
    return REJECT;
  if (is_mcst(dmac) & GMX_RX{prt}_ADR_CTL[MCST] == 2)   // multicast accept
    return ACCEPT;

  cam_hit = 0;

  for (i=0; i<32; i++) {
    if (GMX_RX{prt}_ADR_CAM_ALL_EN[EN<i>] == 0)
      continue;
    uint48 unswizzled_mac_adr = 0x0;
    for (j=5; j>=0; j--) {
       unswizzled_mac_adr = (unswizzled_mac_adr << 8) |
                            GMX_RX{i>>3}_ADR_CAM{j}[ADR<(i&7)*8+7:(i&7)*8>]
                           ;
    }
    if (unswizzled_mac_adr == dmac) {
      cam_hit = 1;
      break;
    }
  }

  if (cam_hit)
    return (GMX_RX{prt}_ADR_CTL[CAM_MODE] ? ACCEPT : REJECT);
  else
    return (GMX_RX{prt}_ADR_CTL[CAM_MODE] ? REJECT : ACCEPT);
}
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

Address-Filtering Control Enable Registers GMX(0)_RX(0..3)_ADR_CAM_EN

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	EN	R/W	0x0	—	CAM-entry enable bits. GMX has 32 DMAC entries that can be accessed with the GMX(0)_RX(0..3)_ADR_CAM(0..5) CSRs. These 32 DMAC entries can be used by any of the two SGMII/QSGMII MACs or the RXAUI MAC. Each port has independent control of which of the 32 DMAC entries to include in the CAM lookup. Legacy GMX implementations used eight DMAC entries while current implementations use 32 common entries. This register is intended for legacy applications that only require eight DMAC CAM entries per MAC. New applications may choose to ignore GMX(0)_RX*n*_ADR_CAM_EN using GMX(0)_RX*n*_ADR_CAM_ALL_EN instead. EN controls the enable bits for the eight legacy CAM entries as follows: port0, EN = GMX_RX0_ADR_CAM_ALL_EN[EN<7:0>] port1, EN = GMX_RX1_ADR_CAM_ALL_EN[EN<15:8>] port2, EN = GMX_RX2_ADR_CAM_ALL_EN[EN<23:16>] port3, EN = GMX_RX3_ADR_CAM_ALL_EN[EN<31:24>] The full 32 independent per-MAC enable bits are in GMX(0)_RX*n*_ADR_CAM_ALL_EN. Therefore, write operations to GMX(0)_RX*n*_ADR_CAM_ALL_EN[EN] are reflected in EN and writes to EN are reflected in GMX(0)_RXX_ADR_CAM_ALL_EN[EN]. In RXAUI mode, only GMX(0)_RX0_ADR_CAM_EN is used; GMX_RX(1..3)_ADR_CAM_EN should not be used.

Address-Filtering Control Enable Registers GMX(0)_RX(0..3)_ADR_CAM_ALL_EN

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	EN	R/W	0x0	—	CAM-entry enable bits. GMX has 32 DMAC entries that can be accessed with the GMX(0)_RX(0..3)_ADR_CAM(0..5) CSRs. These 32 DMAC entries can be used by any of the two SGMII/QSGMII MACs or the RXAUI MAC. Each port has independent control of which of the 32 DMAC entries to include in the CAM lookup. GMX(0)_RX*n*_ADR_CAM_ALL_EN was not present in legacy GMX implementations which had only eight DMAC CAM entries. New applications may choose to ignore GMX(0)_RX*n*_ADR_CAM_EN using GMX(0)_RX*n*_ADR_CAM_ALL_EN instead. EN represents the full 32 independent per-MAC enable bits. Writes to EN are reflected in GMX(0)_RX*n*_ADR_CAM_EN[EN] and writes to GMX(0)_RX*n*_ADR_CAM_EN[EN] are reflected in EN. Refer to GMX(0)_RX*n*_ADR_CAM_EN for the CSR mapping. In RXAUI mode, only GMX(0)_RX0_ADR_CAM_ALL_EN is used; GMX(0)_RX(1..3)_ADR_CAM_ALL_EN should not be used.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Address-Filtering CAM Control Registers GMX(0)_RX(0..3)_ADR_CAM(0..5)

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	ADR	R/W	0x0	—	The DMAC address to match on. Each entry contributes eight bits to one of eight matchers. The CAM matches against unicast or multicast DMAC addresses. All GMX(0)_RX(0..3)_ADR_CAM(0..5) registers can be used in either SGMII/QSGMII or RXAUI mode such that any GMX MAC can use any of the 32 common DMAC entries. GMX(0)_RX(1..3)_ADR_CAM(0..5) are the only non-port0 registers used in RXAUI mode.

Packet TX Threshold Registers GMX(0)_TX(0..3)_THRESH

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8:0>	CNT	R/W	0x20	0x20	Number of 16-byte ticks to accumulate in the TX FIFO before sending on the packet interface. This field should be large enough to prevent underflow on the packet interface and must never be set to 0x0. This register cannot exceed the TX FIFO depth as follows. GMX_TX_PRTS = 0,1: CNT maximum = 0x100 GMX_TX_PRTS = 2: CNT maximum = 0x080 GMX_TX_PRTS = 3,4: CNT maximum = 0x040 **NOTE:** In RXAUI mode, port 0 is used for checking. Since RXAUI mode uses a single TX FIFO and is a higher data rate, Cavium recommends a value of 0x80.

Packet TX Append Control Registers GMX(0)_TX(0..3)_APPEND

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	FORCE_FCS	R/W	1	1	Append the Ethernet FCS on each pause packet when FCS = 0. Pause packets are normally padded to 60 bytes. If GMX(0)_TX(0..3)_MIN_PKT[MIN_SIZE] exceeds 59, then FORCE_FCS is not used.
<2>	FCS	R/W	1	1	Append the Ethernet FCS on each packet.
<1>	PAD	R/W	1	1	Append PAD bytes such that a minimum-sized packet is transmitted.
<0>	PREAMBLE	R/W	1	1	Prepend the Ethernet preamble on each transfer. When GMX0_TX_XAUI_CTL[HG_EN] is set, PREAMBLE must be zero.

Cavium Confidential For Bowen Yang ALO 02/27/2014

Packet TX Slottime Counter Registers
GMX(0)_TX(0..3)_SLOT

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved
<9:0>	SLOT	R/W	0x200	0x200	Slottime (refer to 802.3 to set correctly) (SGMII/SGMII/1000BASE-X only.) 10/100Mbs: set SLOT to 0x40 (corresponds to 512 bit times) 1000Mbs: set SLOT to 0x200 (corresponds to 4096 bitimes)

Packet TX Burst-Counter Registers
GMX(0)_TX(0..3)_BURST

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	BURST	R/W	0x2000	0x2000	Burst (refer to 802.3 to set correctly). (SGMII/QSGMII/1000BASE-X only.) Only valid for 1000Mb/s half-duplex operation as follows: half duplex/1000Mb/s: 0x2000 all other modes: 0x0

Packet SMAC Registers
GMX(0)_SMAC(0..3)

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	SMAC	R/W	0x0	—	The SMAC field is used for generating and accepting control pause packets.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Packet TX Pause Packet Pause-Time Registers GMX(0)_TX(0..3)_PAUSE_PKT_TIME

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TIME	R/W	0x60	—	Provides the pause_time field placed in outbound 802.3 PAUSE packets, HiGig2 messages, or PFC/CBFC pause packets in 512 bit-times. Normally, TIME > GMX(0)_TX(0..3)_PAUSE_PKT_INTERVAL[INTERVAL].

Notes:

Choosing proper values of GMX(0)_TX(0..3)_PAUSE_PKT_TIME[TIME] and GMX(0)_TX(0..3)_PAUSE_PKT_INTERVAL[INTERVAL] can be challenging. Cavium suggests that TIME be much greater than INTERVAL and GMX(0)_TX(0..3)_PAUSE_ZERO[SEND] be set to 1. This allows a periodic refresh of the PAUSE count and then when the backpressure condition is lifted, a PAUSE packet with TIME = 0 is sent indicating that CN71XX is ready for additional data.

If the system chooses to not set GMX(0)_TX(0..3)_PAUSE_ZERO[SEND], then Cavium suggests that TIME and INTERVAL are programmed such that they satisfy the following rule:

INTERVAL ≤ TIME – (largest_pkt_size + IFG + pause_pkt_size)

where:

largest_pkt_size = the largest packet that the system can send (normally 1518 bytes)

IFG = the interframe gap

pause_pkt_size = the size of the PAUSE packet (normally 64 bytes).

Packet TX Minimum-Size-Packet Registers GMX(0)_TX(0..3)_MIN_PKT

This register specifies the minimum size a packet should be. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	MIN_SIZE	R/W	0x3B	0x3B	Minimum frame size in bytes before the FCS is applied. Padding is only appended when GMX(0)_TX(0..3)_APPEND[PAD] for the corresponding port is set. In SGMII mode, packets are padded to MIN_SIZE+1. The reset value pads to 60 bytes. In RXAUI mode, packets are padded to MIN(252,(MIN_SIZE+1 & ~0x3)). When GMX0_TX_XAUI_CTL[HG_EN] is set, the HiGig header (12B or 16B) is normally added to the packet, so MIN_SIZE should be 59 + 12 = 71B for HiGig or 59 + 16 = 75B for HiGig2.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Packet TX Pause-Packet Transmission-Interval Registers GMX(0)_TX(0..3)_PAUSE_PKT_INTERVAL

This register specifies how often Pause packets are sent. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	INTERVAL	R/W	0x10	—	Arbitrate for a 802.3 Pause packet, HiGig2 message, or PFC/CBFC pause packet every (INTERVAL × 512) bit-times. Normally, 0 < INTERVAL < GMX(0)_TX(0..3)_PAUSE_PKT_TIME[TIME]. INTERVAL = 0 only send a single PAUSE packet for each backpressure event.

Notes:

Choosing proper values of GMX(0)_TX(0..3)_PAUSE_PKT_TIME[TIME] and GMX(0)_TX(0..3)_PAUSE_PKT_INTERVAL[INTERVAL] can be challenging. Cavium suggests that TIME be much greater than INTERVAL and GMX(0)_TX(0..3)_PAUSE_ZERO[SEND] be set to 1. This allows a periodic refresh of the PAUSE count and then when the backpressure condition is lifted, a PAUSE packet with TIME = 0 is sent indicating that CN71XX is ready for additional data.

If the system chooses to not set GMX(0)_TX(0..3)_PAUSE_ZERO[SEND], then Cavium suggests that TIME and INTERVAL are programmed such that they satisfy the following rule:

INTERVAL <= TIME – (largest_pkt_size + IFG + pause_pkt_size)

where:

largest_pkt_size = the largest packet that the system can send (normally 1518 bytes)

IFG = the interframe gap

pause_pkt_size = the size of the PAUSE packet (normally 64 bytes).

Packet TX Software-Pause Registers GMX(0)_TX(0..3)_SOFT_PAUSE

This register specifies the pause time. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TIME	R/W	0x0	—	Back off the TX bus for (TIME × 512) bit-times

Cavium Confidential For Bowen Yang ALU 8/27/2014

Packet TX Time-to-Backpressure Registers
GMX(0)_TX(0..3)_PAUSE_TOGO

This register is the pause-deferral counter. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:16>	MSG_TIME	RO/H	—	—	Amount of time remaining to backpressure, from the HiGig2 physical message pause timer (only valid on port0).
<15:0>	TIME	RO/H	—	—	Amount of time remaining to backpressure, from the standard 802.3 pause timer.

Packet TX Pause-Zero-Enable Registers
GMX(0)_TX(0..3)_PAUSE_ZERO

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	SEND	R/W	1	1	Send pause-zero enable.When this bit is set, and the backpressure condition is clear, it allows sending a pause packet with pause_time of 0 to enable the channel.

GMX TX Statistics Control Registers
GMX(0)_TX(0..3)_STATS_CTL

This register controls the transmit statistics registers. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	RD_CLR	R/W	0	0	Clear on read. When this bit is set, any transmit statistics registers are cleared when read.

GMX Transmit Control Registers
GMX(0)_TX(0..3)_CTL

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	XSDEF_EN	R/W	1	1	Enables the excessive-deferral check for statistics and interrupts. (SGMII/QSGMII/1000BASE-X half-duplex only.)
<0>	XSCOL_EN	R/W	1	1	Enables the excessive-collision check for statistics and interrupts. (SGMII/QSGMII/1000BASE-X half-duplex only.)

Cavium Confidential For Bowen Yang ALU 02/27/2014

Transmit Statistics Registers 0
GMX(0)_TX(0..3)_STAT0

This register maintains statistics on excessive deferrals and excessive collisions. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	XSDEF	RC/W/H	0x0	—	Number of packets dropped (never successfully sent) due to excessive deferrals. (SGMII/QSGMII/1000BASE-X half-duplex only.)
<31:0>	XSCOL	RC/W/H	0x0	—	Number of packets dropped (never successfully sent) due to excessive collision. Defined by GMX(0)_TX_COL_ATTEMPT[LIMIT]. (SGMII/QSGMII/1000BASE-X half-duplex only.)

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Registers 1
GMX(0)_TX(0..3)_STAT1

This register maintains statistics on single collisions and multiple collisions. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	SCOL	RC/W/H	0x0	—	Number of packets sent with a single collision. (SGMII/QSGMII/1000BASE-X half-duplex only.)
<31:0>	MCOL	RC/W/H	0x0	—	Number of packets sent with multiple collisions but less than GMX(0)_TX_COL_ATTEMPT[LIMIT]. (SGMII/QSGMII/1000BASE-X half-duplex only.)

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Registers 2
GMX(0)_TX(0..3)_STAT2

This register maintains statistics on the number of octets sent on the interface. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	OCTS	RC/W/H	0x0	—	Number of total octets sent on the interface. Does not count octets from frames that were truncated due to collisions in half-duplex mode.

NOTE:
- Octet counts are the sum of all data transmitted on the wire, including packet data, pad bytes, FCS bytes, Pause bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when GMX(0)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Transmit Statistics Registers 3
GMX(0)_TX(0..3)_STAT3

Used for GMX_TX_STATS_PKTS.

This register maintains statistics on the number of frames sent on the interface. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	PKTS	RC/W/H	0x0	—	Number of total frames sent on the interface. Does not count frames that were truncated due to collisions in half-duplex mode.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Registers 4
GMX(0)_TX(0..3)_STAT4

This register maintains statistics on the number of packets sent on the interface with octet counts of 64 and less than 64. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	HIST1	RC/W/H	0x0	—	Number of packets sent with an octet count of 64.
<31:0>	HIST0	RC/W/H	0x0	—	Number of packets sent with an octet count < 64.

NOTE:
- Packet length is the sum of all data transmitted on the wire for the given packet, including packet data, pad bytes, FCS bytes, PAUSE bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when GMX(0)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Registers 5
GMX(0)_TX(0..3)_STAT5

This register maintains statistics on the number of packets sent on the interface with octet counts between 65 and 127 and between 128 and 255. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	HIST3	RC/W/H	0x0	—	Number of packets sent with an octet count of 128 – 255.
<31:0>	HIST2	RC/W/H	0x0	—	Number of packets sent with an octet count of 65 – 127.

NOTE:
- Packet length is the sum of all data transmitted on the wire for the given packet, including packet data, pad bytes, FCS bytes, PAUSE bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when GMX(0)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/17/2014

Transmit Statistics Registers 6
GMX(0)_TX(0..3)_STAT6

This register maintains statistics on the number of packets sent on the interface with octet counts between 256 and 511 and between 512 and 1023. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	HIST5	RC/W/H	0x0	—	Number of packets sent with an octet count of 512 – 1023.
<31:0>	HIST4	RC/W/H	0x0	—	Number of packets sent with an octet count of 256 – 511.

NOTE:
- Packet length is the sum of all data transmitted on the wire for the given packet, including packet data, pad bytes, FCS bytes, PAUSE bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when GMX(0)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Registers 7
GMX(0)_TX(0..3)_STAT7

This register maintains statistics on the number of packets sent on the interface with octet counts between 1024 and 1518 and greater than 1518. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	HIST7	RC/W/H	0x0	—	Number of packets sent with an octet count of > 1518.
<31:0>	HIST6	RC/W/H	0x0	—	Number of packets sent with an octet count of 1024 – 1518.

NOTE:
- Packet length is the sum of all data transmitted on the wire for the given packet, including packet data, pad bytes, FCS bytes, PAUSE bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when GMX(0)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Registers 8
GMX(0)_TX(0..3)_STAT8

This register maintains statistics on the number of packets sent to broadcast DMAC and to multicast DMAC. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	MCST	RC/W/H	0x0	—	Number of packets sent to multicast DMAC. Does not include BCST packets.
<31:0>	BCST	RC/W/H	0x0	—	Number of packets sent to broadcast DMAC. Does not include MCST packets.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.
- Note that the GMX determines if the packet is MCST or BCST from the DMAC of the packet. GMX assumes that the DMAC lies in the first six bytes of the packet as per the 802.3 frame definition. If the system requires additional data before the L2 header, then the MCST and BCST counters may not reflect reality and should be ignored by software.

Cavium Confidential For Bowen Yang ALU 04/27/2014

Transmit Statistics Registers 9
GMX(0)_TX(0..3)_STAT9

This register maintains statistics on the number of underflow packets and control packets. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	UNDFLW	RC/W/H	0x0	—	Number of underflow packets.
<31:0>	CTL	RC/W/H	0x0	—	Number of Control packets (PAUSE flow control) generated by GMX. It does not include control packets forwarded or generated by the cores. CTL counts the number of generated PFC frames; CTL does not track the number of generated HG2 messages.

NOTE:
- Cleared by either a write (of any value) or a read when GMX(0)_TX(0..3)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Table 24–6GMX BIST Results Register
GMX(0)_BIST

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved.
<24:0>	STATUS	RO	0x0	0x0	BIST results. 0 = pass, 1 = fail 0: gmx#.inb.fif_bnk0 1: gmx#.inb.fif_bnk1 2: gmx#.inb.fif_bnk2 3: gmx#.inb.fif_bnk3 4: gmx#.inb.fif_bnk_ext0 5: gmx#.inb.fif_bnk_ext1 6: gmx#.inb.fif_bnk_ext2 7: gmx#.inb.fif_bnk_ext3 8: gmx#.outb.fif.fif_bnk0 9: gmx#.outb.fif.fif_bnk1 10: gmx#.outb.fif.fif_bnk2 11: gmx#.outb.fif.fif_bnk3 12: gmx#.outb.fif.fif_bnk_ext0 13: gmx#.outb.fif.fif_bnk_ext1 14: gmx#.outb.fif.fif_bnk_ext2 15: gmx#.outb.fif.fif_bnk_ext3 16: gmx#.csr.gmi0.srf8x64m1_bist 17: gmx#.csr.gmi1.srf8x64m1_bist 18: gmx#.csr.gmi2.srf8x64m1_bist 19: gmx#.csr.gmi3.srf8x64m1_bist 20: gmx#.csr.drf20x32m2_bist 21: gmx#.csr.drf20x48m2_bist 22: gmx#.outb.stat.drf16x27m1_bist 23: gmx#.outb.stat.drf40x64m1_bist 24: xgmii.tx.drf16x38m1_async_bist

RX Ports Register
GMX(0)_RX_PRTS

This register specifies the number of ports, which indicates the number of FIFOs that make up the RX buffer. See Table 24–6 for the address.

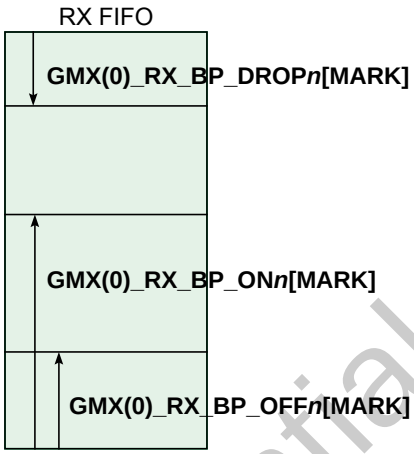
Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved
<2:0>	PRTS	R/W	0x4	0x4	In SGMII/QSGMII/1000BASE-X mode, the RX buffer can be carved into several logical buffers depending on the number or implemented ports. 0 or 1 port = 512 ticks / 4096 bytes 2 ports = 256 ticks / 2048 bytes 3 or 4 ports = 128 ticks / 1024 bytes This field must be set to 0x1 in RXAUI mode.

Cavium Confidential For Boulder Yang 02/27/2015

RX FIFO Packet-Drop Registers
GMX(0)_RX_BP_DROP(0..3)

This register provides the mark for an RX FIFO beyond which packets are dropped and not buffered. The actual watermark is dynamic with respect to the GMX(0)_RX_PRTS register. GMX(0)_RX_PRTS controls the depth of the port's FIFO, so as ports are added or removed, the drop point may change. The relationship of the values in GMX(0)_RX_BP_DROP(0..3), GMX(0)_RX_BP_ON(0..3), and GMX(0)_RX_BP_OFF(0..3) is shown Figure 24–12.

Figure 24–12 RX FIFO Values



In RXAUI mode, port 0 is used for checking. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	MARK	R/W	0x2	0x2	Number of eight-byte ticks to reserve in the RX FIFO. When the FIFO exceeds this count, packets are dropped and not buffered. MARK should typically be programmed to the number of ports + 1. Failure to program correctly can lead to system instability.

RX Backpressure On Registers
GMX(0)_RX_BP_ON(0..3)

This register provides the high watermark for port/interface backpressure, the FIFO-full level at which backpressure is asserted. In RXAUI mode, port 0 is used for checking. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8:0>	MARK	R/W	0x40	0x40	High watermark (in eight-byte ticks) for asserting backpressure. Each register is for an individual port. In RXAUI mode, port 0 is used for the unified RX FIFO. GMX_RX_BP_ON must satisfy: BP_OFF[MARK] ≤ BP_ON[MARK] < (FIFO_SIZE – BP_DROP[MARK]) A value of 0x0 immediately asserts backpressure.

Cavium Confidential For Bowen Yang ALL 6/27/2014

RX Backpressure Off Registers GMX(0)_RX_BP_OFF(0..3)

This register provides the low watermark for port/interface backpressure, the FIFO-full level at which backpressure can be deasserted. In RXAUI mode, port 0 is used for checking. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	MARK	R/W	0x10	0x10	Low-water mark (in eight-byte ticks) to deassert backpressure.

TX Ports Register GMX(0)_TX_PRTS

This register specifies the number of ports allowed on the interface, which is one higher than highest-numbered port (e.g. if port 3 is the highest port number, this register specifies 4). See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved
<4:0>	PRTS	R/W	0x4	—	Number of ports allowed on the interface. (SGMII/QSGMII/1000BASE-X only.)

Packet TX Interframe Gap Register GMX(0)_TX_IFG

Consider the following when programming IFG1 and IFG2:

- For 10/100/1000 Mb/s half-duplex systems that require IEEE 802.3 compatibility, IFG1 must be in the range of 1–8, IFG2 must be in the range of 4–12, and the IFG1 + IFG2 sum must be 12.
- For 10/100/1000 Mb/s full-duplex systems that require IEEE 802.3 compatibility, IFG1 must be in the range of 1–11, IFG2 must be in the range of 1–11, and the IFG1 + IFG2 sum must be 12.
- For RXAUI/10Gb/s systems that require IEEE 802.3 compatibility, the IFG1 + IFG2 sum must be 12. IFG1[1:0] and IFG2[1:0] must be zero.
- For all other systems, IFG1 and IFG2 can be any value in the range of 1–15, allowing for a total possible IFG sum of 2–30.

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:4>	IFG2	R/W	0x4	—	Remainder of interFrameGap timing, equal to interFrameGap – IFG1 (in IFG2 × 8 bits). If CRS is detected during IFG2, the interFrameSpacing timer is not reset and a frame is transmitted once the timer expires.
<3:0>	IFG1	R/W	0x8	—	First portion of interFrameGap timing, in the range of 0 to 2/3 (in IFG2 × 8 bits). If CRS is detected during IFG1, the interFrameSpacing timer is reset and a frame is not transmitted.

Cavium Confidential For Bowen Yang ALU 6/23/2014

Packet TX JAM Pattern Register GMX(0)_TX_JAM

This register provides the pattern used in JAM bytes. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	JAM	R/W	0xEE	—	JAM pattern. (SGMII/1000Base-X half-duplex only.)

Packet TX Collision Attempts Before Dropping Frame Register GMX(0)_TX_COL_ATTEMPT

This register provides the number of collision attempts allowed before dropping the frame. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved
<4:0>	LIMIT	R/W	0x10	0x10	Number of collision attempts allowed. (SGMII/1000Base-X half-duplex only.)

Packet TX Pause-Packet DMAC-Field Register GMX(0)_TX_PAUSE_PKT_DMAC

This register provides the DMAC value that is placed in outbound pause packets. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	DMAC	R/W	0x0180C2000001	0x0180C2000001	The DMAC field, which is placed is outbound pause packets.

Packet TX Pause Packet TYPE Field Register GMX(0)_TX_PAUSE_PKT_TYPE

This register provides the Type field that is placed in outbound pause packets. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TYPE	R/W	0x8808	0x8808	The Type field placed is outbound pause packets.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Packet Interface TX Override Backpressure Register GMX(0)_TX_OVR_BP

This register provides the ability to assert per-port backpressure, as well as the ability to override the assertion of backpressure for SGMII / 1000BASE-X and RXAUI ports. In RXAUI mode, only the least significant bit (corresponding to port 0) of EN, BP, and IGN_FULL are used.

GMX(0)_TX_OVR_BP[EN<0>] must be set to one and GMX(0)_TX_OVR_BP[BP<0>] must be cleared to zero to forcibly disable hardware-automatic 802.3 pause packet generation with the HiGig2 Protocol when GMX(0)_HG2_CONTROL[HG2TX_EN] = 0. (The HiGig2 protocol is indicated by GMX(0)_TX_XAUI_CTL[HG_EN] = 1 and GMX(0)_RX0_UDD_SKP[LEN]=16.) Hardware can only auto-generate backpressure through HiGig2 messages (optionally, when GMX(0)_HG2_CONTROL[HG2TX_EN] = 1) with the HiGig2 protocol. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:32>	TX_PRT_BP	R/W	0x0	0x0	Per-port backpressure sent to PKO: 0 = port is available. 1 = port should be backpressured. TX_PRT_BP should not be set until GMX_INF_MODE[EN] has been enabled
<31:12>	—	RAZ	—	—	Reserved
<11:8>	EN	R/W	0x0	0x0	Per-port backpressure enable/override (<8> for port 0, ..., <11> for port 3). Bits in this field must be set to allow backpressure to be asserted for the corresponding port. Ports that don't have the corresponding bit set are not able to assert backpressure.
<7:4>	BP	R/W	0x0	0x0	Per-port backpressure status to use: (<4> for port 0, ..., <7> for port 3) 0 = port is available 1 = port should be backpressured
<3:0>	IGN_FULL	R/W	0x0	0x0	Ignore the RX FIFO full when computing backpressure (<0> for port 0, ..., <3> for port 3).

Packet Interface TX Backpressure Status Register GMX(0)_TX_BP

This register shows the status of backpressure for SGMII/QSGMII/1000BASE-X and RXAUI ports. In RXAUI mode, only the least significant bit (corresponding to port 0) of EN, BP, and IGN_FULL are used. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	BP	RO	0x0	0x0	Per-port backpressure status: (<0> for port 0, ..., <3> for port 3) 0 = port is available, 1 = port should be backpressured

Cavium Confidential For Bowen Yang ALU 8/27/2014

TX Corrupt Packets Register GMX(0)_TX_CORRUPT

This register enables the STX to corrupt a packet under certain conditions. Packets sent from PKO with the ERR wire asserted are corrupted by the transmitter if CORRUPT[port#] is set (RXAUI uses port# 0).

Corruption means that GMX sends a bad FCS value. If GMX(0)_TX(0..3)_APPEND[FCS] is clear, then no FCS is sent and the GMX cannot corrupt it. The corrupt FCS value is 0xEEEEEEEE for SGMII/QSGMII/1000BASE-X and four bytes of the error propagation code in RXAUI mode. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	CORRUPT	R/W	0xF	0xF	Per-port error propagation. 0 = Never corrupt packets 1 = Corrupt packets with ERR

RX Port State Information Register GMX(0)_RX_PRT_INFO

This register reports the RX status for the port. In RXAUI mode, only the least significant bit (corresponding to port 0) of DROP and COMMIT are used. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:20>	—	RAZ	—	—	Reserved
<19:16>	DROP	RO/H	0x0	0x0	Per-port indication that data was dropped.
<15:4>	—	RAZ	—	—	Reserved
<3:0>	COMMIT	RO/H	0x0	0x0	Per-port indication that SOP was accepted.

TX LFSR Register GMX(0)_TX_LFSR

This register shows the contents of the linear feedback shift register (LFSR), which is used to implement truncated binary exponential backoff. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	LFSR	R/W	0xFFFF	—	Contains the current state of the LFSR, which is used to feed random numbers to compute truncated binary exponential backoff. (SGMII/ 1000Base-X half-duplex only.)

Cavium Confidential For Bowen Yang ALU 02/27/2014

TX Interrupt Register GMX(0)_TX_INT_REG

This register allows per-port interrupts to be sent to the control processor. If an error is noted in this register, and the corresponding bit in GMX(0)_TX_INT_EN is set, an exception is raised. In RXAUI mode, only the least significant bit (corresponding to port 0) of UNDFLW is used. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved
<24>	XCHANGE	R/W1C/H	0	0	XAUI link status changed – this denotes a change to GMX_RX_XAUI_CTL[STATUS] (XAUI mode only)
<23:20>	PTP_LOST	R/W1C/H	0x0	0x0	A packet with a PTP request could not be sent due to XSCOL.
<19:16>	LATE_COL	R/W1C/H	0x0	0x0	TX late collision. (SGMII/1000Base-X half-duplex only.)
<15:12>	XSDEF	R/W1C/H	0x0	0x0	TX excessive deferrals. (SGMII/1000Base-X half-duplex mode only.)
<11:8>	XSCOL	R/W1C/H	0x0	0x0	TX excessive collisions. (SGMII/1000Base-X half-duplex mode only.)
<7:6>	SPARE	R/W1C/H	0x0	0x0	Spare bits
<5:2>	UNDFLW	R/W1C/H	0x0	0x0	TX underflow
<1>	—	RAZ	—	—	Reserved
<0>	PKO_NXA	R/W1C/H	0	0	Port address out-of-range from PKO interface.

TX Interrupt-Enable Register GMX(0)_TX_INT_EN

This register provides the interrupt-enable bits for the corresponding error in GMX(0)_TX_INT_REG. In RXAUI mode, only the least significant bit (corresponding to port 0) of UNDFLW is used. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved
<24>	XCHANGE	R/W/H	0	0	RXAUI link status changed. This denotes a change to GMX_RX_XAUI_CTL[STATUS] (RXAUI mode only).
<23:20>	PTP_LOST	R/W	0x0	0x0	A packet with a PTP request could not be sent due to XSCOL.
<19:16>	LATE_COL	R/W	0x0	0x0	TX late collision. (SGMII/QSGMII/1000BASE-X only.)
<15:12>	XSDEF	R/W	0x0	0x0	TX excessive deferrals. (SGMII/QSGMII/1000BASE-X half-duplex only.)
<11:8>	XSCOL	R/W	0x0	0x0	TX excessive collisions. (SGMII/QSGMII/1000BASE-X half-duplex only.)
<7:6>	SPARE	R/W	0x0	0x0	Spare bits.
<5:2>	UNDFLW	R/W	0x0	0x0	TX Underflow
<1>	—	RAZ	—	—	Reserved
<0>	PKO_NXA	R/W	0	0	Port address out-of-range from PKO Interface

Cavium Confidential For Boyuan Yang 02/12/2021

Address-out-of-Range Error Register GMX(0)_NXA_ADR

This register shows the logged address that caused an address-out-of-range error. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	PRT	RO/H	0x0	—	Logged address for address-out-of-range exceptions. The logged address is from the first exception that caused the problem. Ports 3:0 are valid.

GMX Miscellaneous Error Register GMX(0)_BAD_REG

This register contains a variety of error conditions that were not covered in other error registers. In RXAUI mode, only the least significant bit (corresponding to port 0) of INB_NXA, LOSTSTAT, OUT_OVR, are used. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:31>	—	RAZ	—	—	Reserved
<30:27>	INB_NXA	R/W1C/H	0x0	0x0	Inbound port > GMX(0)_RX_PRTS
<26>	STATOVR	R/W1C/H	0	0	TX statistics overflow. The common FIFO to SGMII/QSGMII and RXAUI had an overflow. TX statistics are corrupted.
<25:22>	LOSTSTAT	R/W1C/H	0x0	0x0	TX statistics data was over-written. In SGMII mode, one bit per port. In RXAUI mode, only port 0 is used. TX statistics are corrupted.
<21:6>	—	RAZ	0x0	0x0	Reserved
<5:2>	OUT_OVR	R/W1C/H	0x0	0x0	Outbound data FIFO overflow (per port).
<1:0>	—	RAZ	—	—	Reserved

GMX Backpressure Statistics Register GMX(0..1)_STAT_BP

This register indicates the number of cycles that the TX/Stats block has held up operation. See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:17>	—	RAZ	—	—	Reserved
<16>	BP	RO/H	0	0	When the TX stats machine cannot update the stats registers quickly enough, the machine has the ability to backpressure TX datapath. This is a rare event and will not occur in normal operation. 0 = No backpressure is applied. 1 = Backpressure is applied to TX datapath to allow stat update operations to complete.
<15:0>	CNT	R/W1C/H	0x0	0x0	Number of cycles that backpressure has been asserted, saturating counter.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GMX TX RXAUI Control Register
GMX(0)_TX_XAUI_CTL

See Table 24–6 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:11>	—	RAZ	—	—	Reserved
<10:9>	HG_PAUSE_HGI	R/W	0x2	0x2	HGI field for hardware generated HiGig pause packets. (RXAUI mode only.)
<8>	HG_EN	R/W	0	0	Enable HiGig mode. (RXAUI mode only.) When this field is set and GMX_RX_UDD_SKP[SKIP] = 12, the interface is in HiGig/HiGig+ mode and the following must be set: GMX_RX_FRM_CTL[PRE_CHK] == 0 GMX_RX_UDD_SKP[FCSSEL] == 0 GMX_RX_UDD_SKP[SKIP] == 12 GMX_TX_APPEND[PREAMBLE] == 0 (depending on the HiGig header size) When this field is set and GMX_RX_UDD_SKP[SKIP] = 16, the interface is in HiGig2 mode and the following must be set: GMX_RX_FRM_CTL[PRE_CHK] == 0 GMX_RX_UDD_SKP[FCSSEL] == 0 GMX_RX_UDD_SKP[SKIP] == 16 GMX_TX_APPEND[PREAMBLE] == 0 GMX_PRT0_CBFC_CTL[RX_EN] == 0 GMX_PRT0_CBFC_CTL[TX_EN] == 0
<7>	—	RAZ	—	—	Reserved
<6>	LS_BYP	R/W	0	0	Bypass the link status, as determined by the XGMII receiver, and set the link status of the transmitter to LS. (RXAUI mode only.)
<5:4>	LS	R/W	0	0	Link status. 0 = Link OK; link runs normally. RS passes MAC data to PCS 1 = Local Fault. RS layer sends continuous remote fault sequences. 2 = Remote Fault RS layer sends continuous idle sequences. 3 = Link Drain RS layer drops full packets to allow GMX and PKO to drain their FIFOs (RXAUI mode only.)
<3:2>	—	RAZ	—	—	Reserved
<1>	UNI_EN	R/W	0	0	Enable Unidirectional mode (IEEE Clause 66). (RXAUI mode only.)
<0>	DIC_EN	R/W	0	1	Enable the deficit idle counter for IFG averaging (RXAUI mode only).

GMX RX RXAUI Control Register
GMX(0)_RX_XAUI_CTL

See Table 24–6 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1:0>	STATUS	RO/H	0x0	0x0	Link status: 0x0=Link OK, 0x1=Local fault, 0x2=Remote fault, 0x3=Reserved. (RXAUI mode only.)

Cavium Confidential
Bowen Yang
ALU
02/27/2014

GMX RX RXAUI Bad Column Register GMX(0)_RX_XAUI_BAD_COL

See Table 24–6 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:40>	—	RAZ	—	—	Reserved
<39>	VAL	R/W1C/H	0	—	Set when GMX_RX_INT_REG[PCTERR] is set. (RXAUI mode only.)
<38:36>	STATE	RO/H	0	—	When GMX_RX_INT_REG[PCTERR] is set, this field contains the receive state at the time of the error. (RXAUI mode only.)
<35:32>	LANE_RXC	RO/H	0	—	When GMX_RX_INT_REG[PCTERR] is set, this field contains the RXAUI column at the time of the error. (RXAUI mode only.)
<31:0>	LANE_RXD	RO/H	0	—	When GMX_RX_INT_REG[PCTERR] is set, this field contains the RXAUI column at the time of the error. (RXAUI mode only.)

GMX RXAUI External Loopback Enable Register GMX(0)_XAUI_EXT_LOOPBACK

See Table 24–6 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved
<4>	EN	R/W	0	0	Loopback enable. Puts the packet interface in external loopback mode on the RXAUI bus in which the RX lines are reflected on the TX lines. (RXAUI mode only)
<3:0>	THRESH	R/W	0x6	0x6	Threshold on the TX FIFO. Software must only write the typical value. Any other value will cause loopback mode not to function correctly. (RXAUI mode only)

GMX RX HiGig2 Status Register GMX(0)_RX_HG2_STATUS

See Table 24–6 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:32>	PHTIM2GO	RO/H	0x0	—	Physical time to go for removal of physical-link pause. Initial value from received HiGig2 message packet. Nonzero only when physical backpressure active.
<31:16>	XOF	RO/H	0x0	0x0	XOF backpressure vector from HiGig2 message packet or from PFC/CBFC packets. Nonzero only when logical backpressure is active. All bits are 0 when [LGTIM2GO] = 0.
<15:0>	LGTIM2GO	RO/H	0x0	—	Logical packet flow backpressure time remaining. Initial value set from XOF time field of HiGig2 message packet received or a function of the enabled and current timers for PFC/CBFC packets. Nonzero only when logical back pressure is active.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GMX HiGig2 Control Register
GMX(0)_HG2_CONTROL

HiGig2 TX- and RX-enable are normally set together for HiGig2 messaging. Setting just the TX or RX bit results in only the HG2 message transmit or receive capability.

Setting [PHYS_EN] and [LOGL_EN] to 1 allows link pause or backpressure to PKO as per the received HiGig2 message. Setting these fields to 0 disables link pause and backpressure to PKO in response to received messages.

GMX0_TX_XAUI_CTL[HG_EN] must be set to 1 (to enable HiGig) whenever either [HG2TX_EN] or [HG2RX_EN] are set. GMX0_RX(0..3)_UDD_SKP[LEN] must be set to 16 (to select HiGig2) whenever either [HG2TX_EN] or [HG2RX_EN] are set.

GMX0_TX_OVR_BP[EN<0>] must be set to 1 and GMX0_TX_OVR_BP[BP<0>] must be cleared to 0 (to forcibly disable hardware-automatic 802.3 pause packet generation) with the HiGig2 Protocol when GMX0_HG2_CONTROL[HG2TX_EN] = 0. (The HiGig2 protocol is indicated by GMX0_TX_XAUI_CTL[HG_EN] = 1 and GMX0_RX(0..3)_UDD_SKP[LEN]=16.) Hardware can only autogenerate backpressure via HiGig2 messages (optionally, when HG2TX_EN = 1) with the HiGig2 protocol.

See Table 24–6 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:19>	—	RAZ	—	—	Reserved
<18>	HG2TX_EN	R/W	0	0	Enable Transmission of HG2 physical and logical messages. When set, also disables hardware autogenerated (802.3 and PFC/CBFC) pause frames. (CN71XX cannot generate proper 802.3 or PFC/CBFC pause frames in HiGig2 mode.)
<17>	HG2RX_EN	R/W	0	0	Enable extraction and processing of HG2 message packet from RX flow. Physical logical pause information is used to pause physical-link, backpressure PKO. This field must be set when HiGig2 messages are present in the receive stream.
<16>	PHYS_EN	R/W	1	—	Physical-link pause enable for received HiGig2 physical pause message.
<15:0>	LOGL_EN	R/W	0xFFFF	—	XOF enables for received HiGig2 messages or PFC/CBFC packets.

GMX TX HiGig2 Register 1
GMX(0)_TX_HG2_REG1

See Table 24–6 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TX_XOF	R/W1S	0x0	—	TX HiGig2 message for logical-link pause when any bit value changes. This field and GMX0_TX_HG2_REG2[TX_XON] map to the same sixteen physical flops. When written with the address of GMX0_TX_HG2_REG1, it exhibits write-1-to-set behavior; when written with the address of GMX0_TX_HG2_REG2, it exhibits write-1-to-clear behavior. For read operations, either address returns the $GMX0_TX_HG2_REG1 values. Only write in HiGig2 mode, i.e. when GMX_TX_XAUI_CTL[HG_EN] = 1 and GMX_RX_UDD_SKP[SKIP] = 16.

Cavium Confidential For Bowbli Yang AU 02/27/2014

GMX TX HiGig2 Register 2
GMX(0)_TX_HG2_REG2

See Table 24–6 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TX_XON	R/W1C	0x0	—	TX HiGig2 message for logical-link pause when any bit value changes. GMX0_TX_HG2_REG1[TX_XOF] and this field map to the same sixteen physical flops. When written with the address of GMX0_TX_HG2_REG1, it exhibits write-1-to-set behavior; when written with the address of GMX0_TX_HG2_REG2, it exhibits write-1-to-clear behavior. For read operations, either address returns the $GMX0_TX_HG2_REG1 values. Only write in HiGig2 mode, i.e. when GMX_TX_XAUI_CTL[HG_EN] = 1 and GMX_RX_UDD_SKP[SKIP] = 16.

GMX PFC Control Register
GMX(0)_PRT0_CBFC_CTL

See Table 24–6 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:48>	PHYS_EN	R/W	0xFF	0xFF	Physical backpressure enable. Determines which ports will have physical backpressure pause packets. The value placed in the Class Enable Vector field of the PFC/CBFC pause packet is PHYS_EN | LOGL_EN.
<47:32>	LOGL_EN	R/W	0xFF	0xFF	Logical backpressure enable. Determines which ports will have logical backpressure pause packets. The value placed in the Class Enable Vector field of the PFC/CBFC pause packet is PHYS_EN | LOGL_EN.
<31:16>	PHYS_BP	R/W	0xFFFF	—	Physical backpressure. When RX_EN is set and the hardware is backpressuring any ports (from either PFC/CBFC pause packets or the GMX_TX_OVR_BP[TX_PRT_BP] register) and the ports specified by PHYS_BP are backpressured, simulate physical backpressure by deferring all packets on the transmitter.
<15:4>	—	RAZ	—	—	Reserved
<3>	BCK_EN	R/W	0	0	Backpressure enable. When set to 1, forward the PFC/CBFC pause information to the backpressure block.
<2>	DRP_EN	R/W	0	0	Drop-control enable. When set to 1, drop PFC/CBFC pause frames.
<1>	TX_EN	R/W	0	0	Transmit enable. When set to 1, allow for PFC/CBFC pause packets. Must be clear in HiGig2 mode i.e. when GMX_TX_XAUI_CTL[HG_EN] = 1 and GMX_RX_UDD_SKP[SKIP]=16.
<0>	RX_EN	R/W	0	0	Receive enable. When set to 1, allow for PFC/CBFC pause packets. Must be clear in HiGig2 mode i.e. when GMX_TX_XAUI_CTL[HG_EN] = 1 and GMX_RX_UDD_SKP[SKIP]=16.

Cavium Confidential For Beverly Lu

GMX PFC Transmit XOFF Register GMX(0)_TX0_CBFC_XOFF

See Table 24–6 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	XOFF	R/W1S/H	0x0	0x0	Transmit off. Write a 1 to allow backpressure to the port. Do not write in HiGig2 mode i.e. when GMX_TX_XAUI_CTL[HG_EN] = 1 and GMX_RX_UDD_SKP[SKIP]=16.

GMX PFC Transmit XON Register GMX(0)_TX0_CBFC_XON

See Table 24–6 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	XON	R/W1C/H	0x0	0x0	Transmit on. Write a 1 to turn off backpressure to the port. Do not write in HiGig2 mode i.e. when GMX_TX_XAUI_CTL[HG_EN] = 1 and GMX_RX_UDD_SKP[SKIP]=16.

GMX RXAUI Control Register GMX(0)_RXAUI_CTL

See Table 24–6 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	DISPARITY	R/W	0	—	Selects which disparity calculation to use when combining or splitting the RXAUI lanes. (RXAUI mode only.) 0 = Interleave lanes before PCS layer as described in the Dune Networks/Broadcom RXAUI v2.1 specification (obeys 6.25GHz SerDes disparity). 1 = Interleave lanes after PCS layer as described in the Marvell RXAUI Interface specification (does not obey 6.25GHz SerDes disparity).

Wake-On-LAN Control Register GMX(0)_WOL_CTL

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:36>	—	RAZ	—	—	Reserved.
<35:32>	MAGIC_EN	R/W	0x0	—	Port enable for magic-packet WOL event detection.
<31:20>	—	RAZ	—	—	Reserved.
<19:16>	DIRECT_EN	R/W	0x0	—	Port enable for directed-packet WOL event detection.
<15:1>	—	RAZ	—	—	Reserved.
<0>	EN	R/W	0	—	Enter WOL mode.

Cavium Confidential For Bowen Yang 02/17/2011

Interface Mode Register
GMX(0)_INF_MODE

See Table 24–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:4>	MODE	R/W	0x0	—	Interface operating mode. Enumerated by GMX_INF_MODE_E. MODE should be written before any of the DLM registers are written and the EN is asserted.
<3:2>	—	RAZ	—	—	Reserved
<1>	EN	R/W	0	—	Interface enable. This field must be set to enable the packet interface. It should be enabled before any other requests to GMX, including enabling port backpressure with IPD_CTL_STATUS[PBP_EN].
<0>	—	RAZ	—	—	Reserved

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

24.11 PCSX Registers

The PCSX registers are listed in Table 24–8.

Table 24–8 PCSX Registers

Register	Address	CSR Type[1]	Detailed Description
PCSX(0)_CONTROL1_REG	0x00011800B0000800	RSL	See page 1167
PCSX(0)_STATUS1_REG	0x00011800B0000808	RSL	See page 1168
PCSX(0)_SPD_ABIL_REG	0x00011800B0000810	RSL	See page 1168
PCSX(0)_CONTROL2_REG	0x00011800B0000818	RSL	See page 1168
PCSX(0)_STATUS2_REG	0x00011800B0000820	RSL	See page 1169
PCSX(0)_10GBX_STATUS_REG	0x00011800B0000828	RSL	See page 1169
PCSX(0)_TX_RX_STATES_REG	0x00011800B0000830	RSL	See page 1170
PCSX(0)_RX_SYNC_STATES_REG	0x00011800B0000838	RSL	See page 1170
PCSX(0)_TX_RX_POLARITY_REG	0x00011800B0000840	RSL	See page 1170
PCSX(0)_MISC_CTL_REG	0x00011800B0000848	RSL	See page 1171
PCSX(0)_BIT_LOCK_STATUS_REG	0x00011800B0000850	RSL	See page 1171
PCSX(0)_INT_REG	0x00011800B0000858	RSL	See page 1171
PCSX(0)_INT_EN_REG	0x00011800B0000860	RSL	See page 1172
PCSX(0)_LOG_ANL_REG	0x00011800B0000868	RSL	See page 1172
PCSX(0)_BIST_STATUS_REG	0x00011800B0000870	RSL	See page 1172

1. RSL-type registers are accessed indirectly across the IOI.

PCSX Control Register 1
PCSX(0)_CONTROL1_REG

PCSX control register #1, which conforms to IEEE Std 802.3-2005, Section 45.2.3. See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15>	RESET	R/W/H	1	0	Set to reset. 1 = software PCSX reset, 0 = normal operation. The bit returns to 0 after PCS has been reset. Takes 32 coprocessor clock cycles to reset PCS.
<14>	LOOPBCK1	R/W	0	0	Enable loopback. 0=normal operation, 1 = internal loopback mode—XGMII TX data received from the GMX TX port is returned back to the GMX, XGMII RX port.
<13>	SPDSEL1	RO	1	1	This bit is always 1 (write has no effect.) When SPDSEL1 and SPDSEL0 are both set to 1, SPD selects the speed.
<12>	—	RAZ	—	—	Reserved
<11>	LO_PWR	R/W	1	0	Low-power consumption state. 1 = power down (hardware reset), 0 = normal operation
<10:7>	—	RAZ	—	—	Reserved
<6>	SPDSEL0	RO	1	1	This bit is always 1 (write has no effect.) When SPDSEL1 and SPDSEL0 are both set to 1, SPD selects the speed.
<5:2>	SPD	RO	0x0	0x0	Speed. Always select 10Gb/s (write operations have no effect).
<1:0>	—	RAZ	—	—	Reserved

Cavium Confidential FOI
Boyen Yang
AU
02/7/2014

PCSX Status Register 1
PCSX(0)_STATUS1_REG

PCSX status register #1, which conforms to IEEE Std 802.3-2005, Section 45.2.3. See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7>	FLT	RO/H	0	0	Fault condition: 1 = fault condition detected, 0 = no fault condition. This bit is a logical OR of PCSX(0)_STATUS2[XMTFLT,RCVFLT].
<6:3>	—	RAZ	—	—	Reserved
<2>	RCV_LNK	RO/H	0	1	Receive link: 1 = receive link up, 0 = receive link down. Latching low version of PCSX(0)_10GBX_STATUS_REG[ALIGND]. Link down status continues until software read.
<1>	LPABLE	RO	1	1	Always set to 1 for low power ability indication.
<0>	—	RAZ	—	—	Reserved

PCSX Speed Ability Register
PCSX(0)_SPD_ABIL_REG

PCSX speed ability register, which conforms to IEEE Std 802.3-2005, Section 45.2.3. See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	TENPASST	RO	0	0	Always set to 0; no 10PASS-TS/2BASE-TL capability support.
<0>	TENGB	RO	1	1	Always set to 1; 10Gb/s supported.

PCSX Control Register 2
PCSX(0)_CONTROL2_REG

PCSX control register #2, which conforms to IEEE Std 802.3-2005, Section 45.2.3. See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1:0>	TYPE	RO	0x1	0x1	Always set to 0x1; 10GBASE-X only supported.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PCSX Status Register 2
PCSX(0)_STATUS2_REG

PCSX status register #2, which conforms to IEEE Std 802.3-2005, Section 45.2.3. See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:14>	DEV	RO	0x2	0x2	Device present. Always set to 0x2, which means a device is present at the address.
<13:12>	—	RAZ	—	—	Reserved
<11>	XMTFLT	RC/H	0	0	Transmit fault. 0 = No transmit fault, 1 = transmit fault. Implements latching high function until software read.
<10>	RCVFLT	RC/H	0	0	Receive fault. 0 = No receive fault, 1 = receive fault. Implements latching high function until software read.
<9:3>	—	RAZ	—	—	Reserved
<2>	TENGB_W	RO	0	0	Always 0; no 10GBASE-W capability.
<1>	TENGB_X	RO	1	1	Always 1; 10GBASE-X capable.
<0>	TENGB_R	RO	0	0	Always 0; no 10GBASE-R capability.

PCSX 10 GBX Status Register
PCSX(0)_10GBX_STATUS_REG

PCSX 10 GBX status register, which conforms to IEEE Std 802.3-2005, Section 45.2.3. See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	—	—	Reserved
<12>	ALIGND	RO/H	0	1	Lane alignment. 1 = Lane alignment achieved, 0 = Lanes not aligned.
<11>	PATTST	RO	0	0	Pattern testing. Always set to 0; no pattern testing capability.
<10:4>	—	RAZ	—	—	Reserved
<3>	L3SYNC	RO/H	0	1	1 = Receive lane 3 code group synchronized, 0 = not synchronized.
<2>	L2SYNC	RO/H	0	1	1 = Receive lane 2 code group synchronized, 0 = not synchronized.
<1>	L1SYNC	RO/H	0	1	1 = Receive lane 1 code group synchronized, 0 = not synchronized.
<0>	L0SYNC	RO/H	0	1	1 = Receive lane 0 code group synchronized, 0 = not synchronized.

Cavium Confidential For Bowen Yang ASU 02/27/2014

PCSX Transmit/Receive States Register PCSX(0)_TX_RX_STATES_REG

PCSX transmit/receive states register, which conforms to IEEE Std 802.3-2005, Section 45.2.3. See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved
<13>	TERM_ERR	R/W1C/H	0	0	Terminate error. When set to 1, check end function detected error in packet terminate ||T|| column or the one after it.
<12>	SYN3BAD	RO/H	0	0	1 = lane 3 code group synchronized state machine in bad state.
<11>	SYN2BAD	RO/H	0	0	1 = lane 2 code group synchronized state machine in bad state.
<10>	SYN1BAD	RO/H	0	0	1 = lane 1 code group synchronized state machine in bad state.
<9>	SYN0BAD	RO/H	0	0	1 = lane 0 code group synchronized state machine in bad state.
<8>	RXBAD	RO/H	0	0	1 = Receive state machine in a bad state; hardware malfunction.
<7:5>	ALGN_ST	RO/H	0x0	—	Lane alignment state machine state.
<4:3>	RX_ST	RO/H	0x0	—	Receive state machine state.
<2:0>	TX_ST	RO/H	0x0	—	Transmit state machine state.

PCSX Receive Synchronized States Register PCSX(0)_RX_SYNC_STATES_REG

PCSX receive synchronized states register, which conforms to IEEE Std 802.3-2005, Section 45.2.3. See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:12>	SYNC3ST	RO/H	0x0	—	Receive lane 3 code group synchronized state machine state.
<11:8>	SYNC2ST	RO/H	0x0	—	Receive lane 2 code group synchronized state machine state.
<7:4>	SYNC1ST	RO/H	0x0	—	Receive lane 1 code group synchronized state machine state.
<3:0>	SYNC0ST	RO/H	0x0	—	Receive lane 0 code group synchronized state machine state.

PCSX TX/RX Polarity Register PCSX(0)_TX_RX_POLARITY_REG

PCSX TX/RX polarity register, which conforms to IEEE Std 802.3-2005, Section 45.2.3. See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved
<9:6>	XOR_RXPLRT	R/W	0x0	0x0	Per-lane RX polarity control.
<5:2>	XOR_TXPLRT	R/W	0x0	0x0	Per-lane TX polarity control.
<1>	RXPLRT	R/W	0	0	Receive polarity. 1 = inverted polarity, 0 = normal polarity. Must be 0.
<0>	TXPLRT	R/W	0	0	Transmit polarity. 1 = inverted polarity, 0 = normal polarity. Must be 0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PCSX Miscellaneous Control Register
PCSX(0)_MISC_CTL_REG

PCSX miscellaneous control register, which conforms to IEEE Std 802.3-2005, Section 45.2.3. The lane-swap fields for RXAUI are provided to simplify interconnection layout between devices. See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	TX_SWAP	R/W	0	—	TX RXAUI lane swap. 0 = do not swap RXAUI lanes going out to DLMs. 1 = swap lane 3 with lane 0, and lane 2 with lane 1.
<2>	RX_SWAP	R/W	0	—	RX RXAUI lane swap. 0 = do not swap RXAUI lanes coming in from DLMs. 1 = swap lane 3 with lane 0, and lane 2 with lane 1.
<1>	XAUI	RO/H	—	—	1 = RXAUI mode selected, 0 = not RXAUI mode selected.
<0>	GMXENO	R/W	0	0	GMX port enable override. GMX enable/disable status is held during data packet reception.

PCSX Bit Lock Status Register
PCSX(0)_BIT_LOCK_STATUS_REG

PCSX bit lock status register, which conforms to IEEE Std 802.3-2005, Section 45.2.3. See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	BITLCK3	RO/H	0	—	Receive lane 3 bit-lock status.
<2>	BITLCK2	RO/H	0	—	Receive lane 2 bit-lock status.
<1>	BITLCK1	RO/H	0	—	Receive lane 1 bit-lock status.
<0>	BITLCK0	RO/H	0	—	Receive lane 0 bit-lock status.

PCSX Interrupt Register
PCSX(0)_INT_REG

PCSX interrupt register, which conforms to IEEE Std 802.3-2005, Section 45.2.3. See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved
<6>	DBG_SYNC	R/W1C/H	0	0	Code group sync failure debug help. DBG_SYNC is an edge-triggered interrupt. When set, it indicates PCS Synchronization state machine (in Figure 48–7, state diagram in IEEE Std 802.3-2005) changes state SYNC_ACQUIRED_1 to SYNC_ACQUIRED_2 indicating an invalid code group was received on one of the four receive lanes. This interrupt should be always disabled and used only for link problem debugging help.
<5>	ALGNLOS	R/W1C/H	0	0	Set when RXAUI lanes lose alignment.
<4>	SYNLOS	R/W1C/H	0	0	Set when code group sync lost on one or more lanes.
<3>	BITLCKLS	R/W1C/H	0	0	Set when bit lock lost on one or more RXAUI lanes.
<2>	RXSYNBAD	R/W1C/H	0	0	Set when RX code group synchronize state machine is in bad state in one of the two RXAUI lanes.
<1>	RXBAD	R/W1C/H	0	0	Set when RX state machine is in bad state.
<0>	TXFLT	R/W1C/H	0	0	None defined at this time; always 0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PCSX Interrupt Enable Register
PCSX(0)_INT_EN_REG

PCSX interrupt enable register, which conforms to IEEE Std 802.3-2005, Section 45.2.3. See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved
<6>	DBG_SYNC_EN	R/W/H	0	0	Enable code group sync failure debug help.
<5>	ALGNLOS_EN	R/W	0	1	Enable ALGNLOS interrupt.
<4>	SYNLOS_EN	R/W	0	1	Enable SYNLOS interrupt.
<3>	BITLCKLS_EN	R/W	0	1	Enable BITLCKLS interrupt.
<2>	RXSYNBAD_EN	R/W	0	1	Enable RXSYNBAD interrupt.
<1>	RXBAD_EN	R/W	0	1	Enable RXBAD interrupt.
<0>	TXFLT_EN	R/W	0	1	Enable TXFLT interrupt.

PCSX Logic Analyzer Register
PCSX(0)_LOG_ANL_REG

Logic Analyzer is enabled with PCSX(0)_LOG_ANL_REG[LA_EN] for RXAUI only. PCSX(0..4)_LOG_ANL_REG[PKT_SZ] is effective only when LA_EN = 1. For normal operation (RXAUI), this bit must be 0. The dropped lane is used to send RXC[1:0]. See also PCSX(0)_LOG_ANL_REG.

See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved
<6>	ENC_MODE	R/W	0	0	Encoded mode: 1 = send RXAUI encoded data, 0 = send RXAUI raw data to GMX.
<5:4>	DROP_LN	R/W	0x0	0x0	RXAUI lane# from which to drop logic analyzer packets. Bit <5>, Bit <4> Drop lane # 0 0 Drop lane 0 data 0 1 Drop lane 1 data 1 0 reserved 1 1 reserved
<3>	LAFIFOVFL	R/W1C/H	0	0	Logic analyzer FIFO overflow. 1=logic analyses FIFO overflowed during packetization. Write 1 to clear this bit.
<2>	LA_EN	R/W	0	0	Logic analyzer enable. 1 = logic analyzer enabled, 0 = logic analyzer disabled.
<1:0>	PKT_SZ	R/W	0x0	0x0	Logic analyzer packet size. Bit <1>, Bit <0> 0 0 Packet size 1K bytes 0 1 Packet size 4K bytes 1 0 Packet size 8K bytes 1 1 Packet size 16K bytes.

PCSX BIST Status Register
PCSX(0)_BIST_STATUS_REG

See Table 24–8 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	BIST_STATUS	RO	0	0	0 = pass (or BIST in progress/never run), 1 = fail.

Cavium Confidential For Bowen Yang 02-27-2014 Asic

Chapter 25

Reduced Gigabit Media-Independent Interface (RGMII)

This chapter contains the following subjects:

- Overview
- Introduction
- GMX Packet-Interface Buffering
- AGL Features
- Wake-on-LAN (WOL)
- AGL Errors/Exceptions
- AGL Interrupts
- Configuration, Reset, and Link Status Changes
- Statistics
- AGL Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

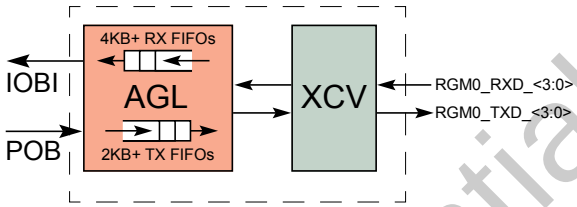
CN71XX implements a single RGMII interface through a single port/lane packet interface. The full-duplex port/lane consists of eight external signals in total: four receive signals and four transmit signals.

CN71XX couples logic that implements the RGMII protocols with a 10/100/1000 802.3 MAC that is fully compliant with the IEEE 802.3 and HP RGMII 1.3 specifications.

25.1 Introduction

Figure 25–1 is a block diagram showing the CN71XX packet interface. **AGL** implements the 10/100/1000 802.3 MAC and includes 4KB receive buffers and 2KB transmit buffers. AGL sends packet-input data to PIP/IPD (refer to Chapter 9) and receives packet-output data from PKO (refer to Chapter 10).

Figure 25–1 CN71XX Packet Interfaces Block Diagram



25.2 GMX Packet-Interface Buffering

The CN71XX packet interface contains a 4KB receive-staging buffer and a 2KB transmit-staging buffer.

In the output case, PKO (described in Chapter 10) has much buffering to add to this packet-interface buffering. More importantly, the CN71XX hardware buffers packet data in L2/DRAM, and this buffering can be many times larger than the packet interface and PKO buffering combined.

In the input case, it is possible for PIP/IPD (described in Chapter 9) to exhaust all of the available L2/DRAM buffering, leaving the packet interface as the only buffering option for input packets. But most CN71XX applications use packet drop and/or per-port backpressure to prevent buffer exhaustion (refer to Sections 9.6 and 9.7). CN71XX can backpressure (via pause packets in full-duplex mode or forced collisions in half-duplex mode) to avoid overflow of the input buffer when its usage exceeds programmable on/off thresholds (refer to Section 25.3.2).

Cavium Confidential For Boren Yang 6/27/2014

25.3 AGL Features

This section discusses some notable features provided by AGL, some of which being outside the 802.3 specifications.

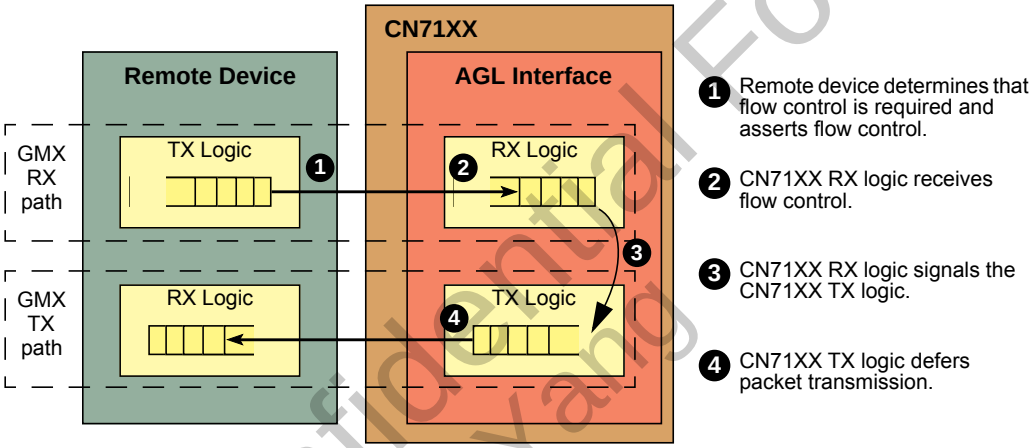
25.3.1 Flow Control

The following subsections discuss how the CN71XX AGL handles flow control for the receive and transmit paths. A remote device's ability to control data flow to the CN71XX is described in Section 25.3.1.1, and CN71XX's ability to control data flow to the a remote device is described in Section 25.3.1.2.

25.3.1.1 Remote-Device Flow Control

Remote-device flow control is shown in Figure 25–2.

Figure 25–2 Remote-Device Flow Control



CN71XX supports flow control for both half- and full-duplex modes of operation. When the receive interface in full-duplex mode, the CN71XX can be configured to receive pause control packets as defined in Annexes 31A and 31B of the 802.3 specification based on the AGL_GMX_RX(0)_FRM_CTL[CTL_BCK/CTL_DRP] registers. Common modes of operation are:

CTL_BCK = 1, CTL_DRP = 1	CN71XX inspects pause packets, sets the transmit defer counter to the TIME parameter in the packets, and then drops the pause packets. CN71XX then defers the transmission of new packets until the timer expires.
CTL_BCK = 0, CTL_DRP = 0	CN71XX processes pause packets like normal packets, so software will see them. Software can optionally implement a backpressure scheme. Software can also set/reset the defer counter by writing to AGL_GMX_TX(0)_SOFT_PAUSE[TIME]. Like pause frames, software must periodically write to AGL_GMX_TX(0)_SOFT_PAUSE[TIME] if the flow condition extends beyond a TIME interval. CN71XX asserts AGL_GMX_RX(0)_INT_REG[PAUSE_DRP] whenever it drops a pause packet in this mode.
CTL_BCK = 0, CTL_DRP = 1	All pause frames are dropped and completely ignored by both hardware and software.

Cavium Confidential For Bowen Yang ADI 02/27/2014

CTL_BCK = 1, CTL_DRP = 0	CN71XX sends pause packets to software like normal packets, and sets the transmit-defer counter to the TIME parameter in the packet. CN71XX then defers the transmission of new outbound packets until the timer expires. As software can see all pause frames, it can optionally implement a backpressure scheme. Software can also set/reset the defer counter by writing to AGL_GMX_TX(0)_SOFT_PAUSE[TIME]. Like pause frames, software must periodically write to AGL_GMX_TX(0)_SOFT_PAUSE[TIME] if the flow condition extends beyond a TIME interval. CN71XX asserts AGL_GMX_RX(0)_INT_REG[PAUSE_DRP] whenever it drops a pause packet in this mode.

Note that the transmit defer counter is not set if either of the following is true:

- minimum packet-length checking is enabled in GMX (i.e. AGL_GMX_RX(0)_FRM_CHK[MINERR] = 1) and the packet size is less than minFrameSize (minFrameSize = 60 + 4 × AGL_GMX_RX(0)_FRM_CHK[FCSERR])
- FCS checking is enabled (AGL_GMX_RX(0)_FRM_CHK[FCSERR] = 1) and the pause packet failed the FCS/CRC check.

Both CTL_BCK and CTL_DRP should be set to 0s in half-duplex mode (pause packets should not be present in half-duplex mode).

If PREAMBLE/SFD is present (i.e. AGL_GMX_RX(0)_FRM_CTL[PRE_CHK] = 1) and/or if user-defined data is present (i.e. AGL_GMX_RX(0)_UDD_SKP[LEN] ≠ 0) in a received packet, CN71XX discards them before it determines whether a packet is a pause frame. CN71XX recognizes the remaining packet bytes as a pause frame if all these conditions are true:

- Either CTL_BCK or CTL_DRP are set
- The first 6 bytes of the packet (i.e. the L2 destination address) match one of these two enabled comparisons:
  - AGL_GMX_RX(0)_FRM_CTL[CTL_MCST] = 1 and the first 6 bytes equal the standard multicast address 0x01-80-C2-00-00-01
  - AGL_GMX_RX(0)_FRM_CTL[CTL_SMAC] = 1 and the first 6 bytes equal AGL_GMX_SMAC(0)[SMAC]
- The 13th and 14th bytes (i.e. the L2 type field) equal 88-08 (MAC control), and
- The 15th and 16th bytes (i.e. the MAC control opcode) equal 00-01 (PAUSE)

Note that PREAMBLE/SFD should normally be present in received pause frames (and AGL_GMX_RX(0)_FRM_CTL[PRE_CHK] should be set) whenever either CTL_BCK or CTL_DRP are set.

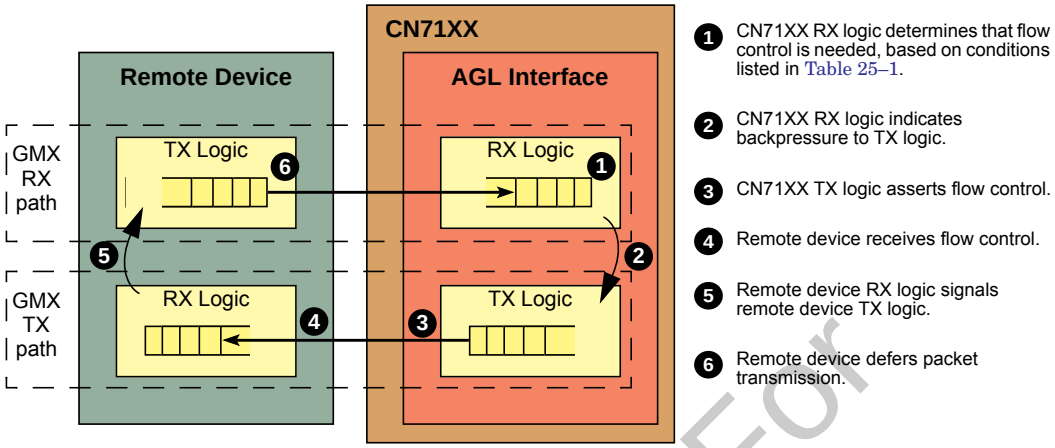
In half-duplex mode, the remote device creates collisions to flow control. GMX views these collisions as normal collisions.

Cavium Confidential For Bowen Yang AEU 02/27/2014

25.3.1.2 CN71XX Flow Control

CN71XX flow control is shown in Figure 25–3.

Figure 25–3 CN71XX Flow Control

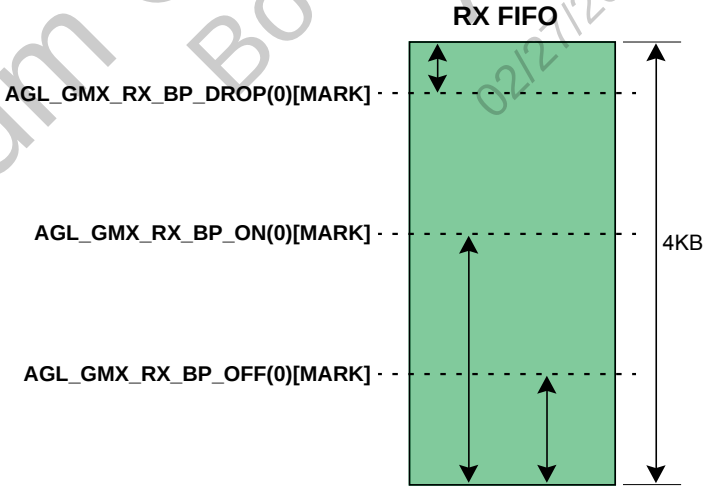


CN71XX controls the flow of packets from the transmitting device under the conditions specified in Table 25–1.

Table 25–1 Flow Control Conditions

Condition	Description
Receive FIFO Flow Control	The receive FIFO can control the flow as it fills up. Each port/lane has a dedicated receive FIFO with dedicated controls. Figure 25–4 shows the relative FIFO depths for each programmable parameter. CN71XX begins to flow control to port when the FIFO accumulates AGL_GMX_RX_BP_ON(0) data. Flow control is removed once the FIFO drops below AGL_GMX_RX_BP_OFF(0) data.
Per-Port Flow Control	PIP/IPD can also communicate per-port flow control. This flow control is exhibited the same way as receive FIFO flow control. Refer to Sections 9.6 and 9.7 for more detailed information.
Software Flow Control	Software can use AGL_GMX_TX_OVR_BP[EN/BP] to force or ignore the flow-control state as desired. If AGL_GMX_TX_OVR_BP[EN*n*] = 1 for a port, the flow-control state will be AGL_GMX_TX_OVR_BP[BP*n*]. This feature is intended for use only in system testing.

Figure 25–4 Receive FIFO Parameters



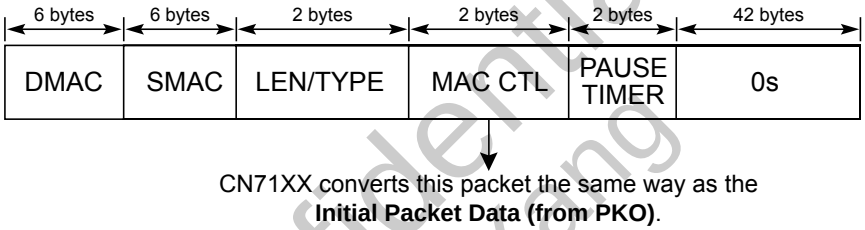
Cavium Confidential For Bowen Wang ALU 02/27/2014

In half-duplex mode when flow control is active for the port, CN71XX forces collisions by sending JAM packets.

In full-duplex mode, CN71XX can automatically generate pause packets. Figure 25–5 shows CN71XX's autogenerated pause packets. Much of the packet is programmable, and can differ for different ports. CN71XX sends these 60-byte packets out the SerDes interface similar to normal packets. The only processing difference is that FCS can be added to autogenerated pause packets in cases when it isn't added to normal packets generated by PKO. When flow control is active, CN71XX autogenerates pause packets at a frequency indicated by AGL_GMX_TX(0)_PAUSE_PKT_INTERVAL[INTERVAL]. If AGL_GMX_TX(0)_PAUSE_ZERO[SEND] = 1, CN71XX autogenerates a pause packet with a pause time of 0 when flow-control deasserts, which can allow the remote transmitter to restart more quickly.

Figure 25–5 Generated Pause Packet Format

DMAC = AGL_GMX_TX_PAUSE_PKT_DMAC[SMAC]
SMAC = AGL_GMX_SMAC(0)[SMAC]
LEN/TYPE = AGL_GMX_TX_PAUSE_PKT_TYPE[TYPE]
MAC CTL = 00–01
PAUSE TIMER = AGL_GMX_TX(0)_PAUSE_PKT_TIME[TIME]



CN71XX converts this packet the same way as the **Initial Packet Data (from PKO)**.

Note that CN71XX should likely only autogenerate pause packets (in full-duplex mode) when it automatically inserts PREAMBLE/SFD (i.e. when AGL_GMX_TX(0)_APPEND[PREAMBLE] = 1), and when there is no UDD present in out-going packets. The CN71XX GMX transmit logic is not aware of any UDD in transmitted packets. In cases where PREAMBLE/SFD is not automatically generated or when there is UDD present in out-going packets, software should likely generate outgoing pause packets.

25.3.2 MAC Receive Preamble Processing

AGL can be configured to check or ignore PREAMBLE/SFD data that marks the beginning of each frame based on AGL_GMX_RX(0)_FRM_CTL[PRE_CHK]. This may be useful for system configuration using multiple CN71XXs or other devices that choose to communicate over directly over RGMII and do not require a PHY device. This may be useful for non-PHY connections. If PREAMBLE data is present, it can be stripped or forwarded as part of the packet based on AGL_GMX_RX(0)_FRM_CTL[PRE_STRP].

25.3.3 Received Packet FCS Processing

AGL optionally checks for FCS errors on the packet (AGL_GMX_RX(0)_FRM_CTL[FCSERR]). Note that the FCS can optionally include UDD data bytes at the beginning of a packet (AGL_GMX_RX(0)_UDD_SKP[FCSSEL]). Refer to Section 25.3.5.2 below.

Cavium Confidential For Bowen Yang 02/27/2014

25.3.4 MAC Receive Packet Dropping

There are several mechanisms within AGL that will drop packets:

- preamble errors
- receive-FIFO packet dropping
- pause packet drops
- DMAC filters
- Receive collisions

These mechanisms are described in the following subsections.

25.3.4.1 PREAMBLE Errors

If preamble checking is enabled (AGL_GMX_RX(0)_FRM_CTL[PRE_CHK] = 1) and the packet does not send a valid PREAMBLE followed by SFD, the packet is dropped. (In this case, a valid PREAMBLE is defined to be data bytes followed by the SFD byte, as per the 802.3 standard.)

25.3.4.2 Receive-FIFO Packet Dropping

When the receive FIFO exceeds AGL_GMX_RX_BP_DROP(0) (as shown in Figure 25–4), the packet cannot be buffered and must be dropped. If there is room in the FIFO for a partial portion of the packet, the beginning of the packet is buffered but the remainder is dropped, and the packet finishes with a PARTIAL completion code. All other packets are completely dropped.

25.3.4.3 Pause Packet Drops

AGL drops any incoming pause packet when AGL_GMX_RX(0)_FRM_CTL[CTL_DRP] = 1. See Section 25.3.2 for additional details. In addition, pause packets can be dropped if AGL_GMX_RX(0)_FRM_CTL[CTL_DRP] = 0 and the receive-FIFO is full. See Section 25.3.4.2 for additional details. Hardware sets AGL_GMX_RX(0)_INT_REG[PAUSE_DRP] = 1 and AGL_GMX_RX(0)_PAUSE_DROP_TIME[STATUS] = pause_time extracted from the last pause packet that is dropped when a pause packet cannot be buffered in the Receive FIFO. The hardware can optionally raise an interrupt if AGL_GMX_RX(0)_INT_EN[PAUSE_DRP] = 1.

25.3.4.4 DMAC Filter

AGL can store up to eight full MAC addresses to match against (AGL_GMX_RX(0)_ADR_CAM_EN and AGL_GMX_RX(0)_ADR_CAM(0..5)). AGL can be configured to accept or drop packets based on this address match (AGL_GMX_RX(0)_ADR_CTL[CAM_MODE]). In addition, AGL can be configured to accept or reject multicast or broadcast packets (AGL_GMX_RX(0)_ADR_CTL[MCST/BCST]).

Cavium Confidential For Bowen Wang 02/7/2014

The full algorithm is shown below:

```
bool agl_dmac_addr_filter(uint48 dmac) {
  if (is_bcst(dmac))                                        //broadcast accept
    return (AGL_GMX_RX0/1_ADR_CTL[BCST] ? ACCEPT : REJECT);
  if (is_mcst(dmac) & AGL_GMX_RX0/1_ADR_CTL[MCST] == 1)    //multicast reject
    return REJECT;
  if (is_mcst(dmac) & AGL_GMX_RX0/1_ADR_CTL[MCST] == 2)    //multicast accept
    return ACCEPT;

  cam_en = 0;
  cam_hit = 0;

  for (i=0; i<8; i++) {
    if (AGL_GMX_RX0/1_ADR_CAM_EN[EN<i>] == 0)
       continue;
    cam_en++;
    uint48 unswizzled_mac_adr = 0x0;
    for (j=5; j>=0; j--) {
       cam_byte = AGL_GMX_RX0/1_ADR_CAM{j}[ADR<i*8+7:i*8>];
       unswizzled_mac_adr = (unswizzled_mac_adr << 8) | cam_byte;
    }
    if (unswizzled_mac_adr == dmac) {
       cam_hit = 1;
       break;
    }
  }
  if (cam_hit)
    return (AGL_GMX_RX0/1_ADR_CTL[CAM_MODE] ? ACCEPT : REJECT);
  else
    return (AGL_GMX_RX0/1_ADR_CTL[CAM_MODE] ? REJECT : ACCEPT);
}
```

25.3.4.5 Receive Collisions

When operating in half-duplex mode (AGL_GMX_PRT(0)_CFG[DUPLEX] = 0), the receiver drops any packets in which the slottime was not satisfied. Note that late collisions are not normally dropped since they satisfy the slottime. Late collisions can be dropped due to any of the other reasons above.

Cavium Confidential For Bowen Yang ALU 02/27/2014

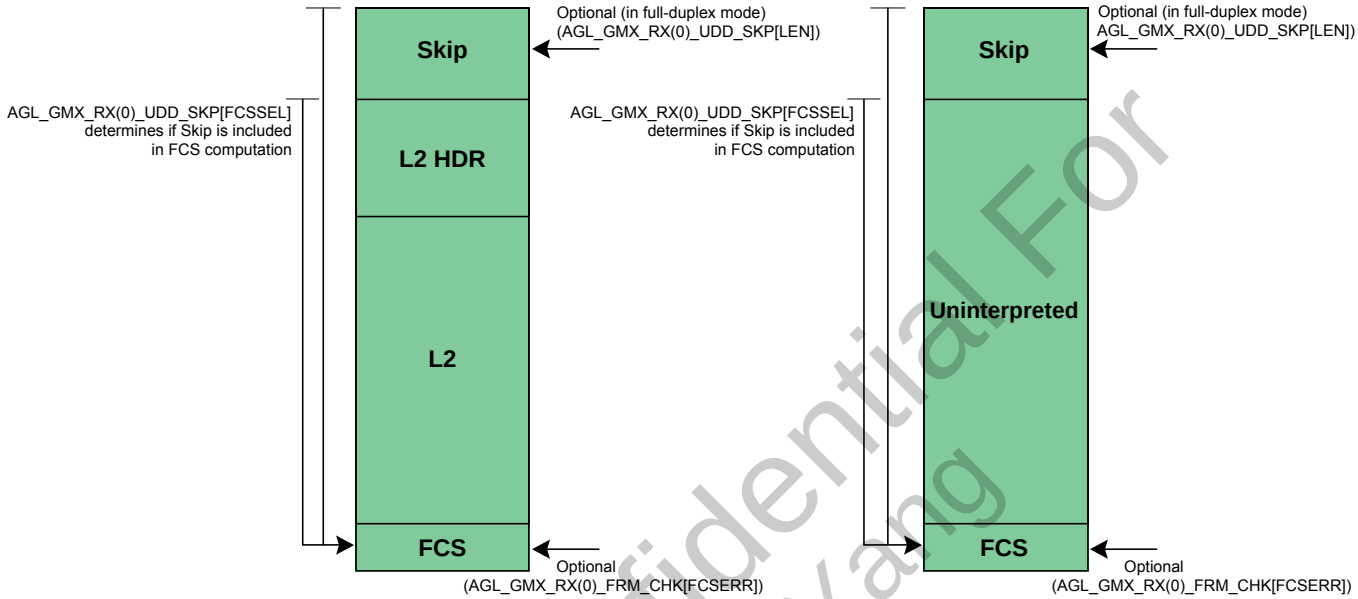
25.3.5 Receive-Packet Inspection

The following subsections describe the receive packets.

25.3.5.1 Receive-Packet Formats

Figure 25–6 shows the receive-packet formats supported by AGL, which differ from those described in Section 9.2. The packet interface supports all six of the L2 header types as defined in Figure 9–8.

Figure 25–6 Packet Formats



AGL inspects the packet to extract the L2 parameters that are required for various tasks and checks. The packet parser must be able to determine where the L2 header is in the packet to extract correct data.

25.3.5.2 Receive User-Defined Data

Normally, the L2 header is either at the start of the packet or after the optional preamble/SFD bytes. In full-duplex mode only, AGL supports user-defined data between the optional preamble/SFD and the start of the L2 header. AGL can support 0–64 bytes of UDD data based on AGL_GMX_RX(0)_UDD_SKP[LEN]. Note that this is a static configuration that cannot change on a per-packet basis. AGL can also be configured to compute the FCS over the UDD data or just the L2 header and L2 payload based on AGL_GMX_RX(0)_UDD_SKP[FCSSEL].

Systems have two choices when skip bytes are sent over AGL channels (followed by an ordinary Ethernet L2 header):

1. If the skip amount is constant for all packets, set AGL_GMX_RX(0)_UDD_SKP[LEN] to the total skip.
2. If the amount of total skip data before the L2 HDR is variable, the L2 HDR inspection must be disabled. (See Section 25.3.5.4 for more details.)

Cavium Confidential For Bowen Yang ALU 01/27/2014

25.3.5.3 Receive Packets without L2 Headers

Some systems may not use the standard Ethernet L2 HDR header or may develop custom headers. Normal L2 HDR inspection can be disabled. If the L2 HDR is not present in the packet, the L2 HDR inspection must be disabled. See Section 25.3.5.4 for more details.

25.3.5.4 Disabling L2 Header Inspection

If the L2 header is not present or is not at a fixed offset in the packet, L2 header inspection should be disabled as follows:

- **L2 length checks** – the L2 length field cannot be extracted from the packet. The length check can be disabled by setting AGL_GMX_RX(0)_FRM_CHK[LENERR] = 0.
- **DMAC filtering** – the L2 DMAC field cannot be extracted from the packet. The DMAC filter can be put into promiscuous mode by setting the following parameters:

  AGL_GMX_RX(0)_ADR_CTL[CAM_MODE] = 0

  AGL_GMX_RX(0)_ADR_CTL[MCST] = 0

  AGL_GMX_RX(0)_ADR_CTL[BCST] = 1

  AGL_GMX_RX(0)_ADR_CAM_EN[EN] = 0.
- **Pause-packet recognition** – the L2 DMAC and type fields cannot be extracted from the packet. Pause-packet recognition can be disabled by setting the following parameters:

  AGL_GMX_RX(0)_FRM_CTL[CTL_BCK] = 0

  AGL_GMX_RX(0)_FRM_CTL[CTL_DRP] = 0

25.3.6 Receive Link Status

The RGMII protocol allows the RGMII link status to be on the data lines during IDLE or NOP RGMII cycles. The PHY can communicate the link status, speed, and duplex value. The current status can be read by software through the AGL_GMX_RX(0)_RX_INBND register and the hardware can be configured to interrupt upon changes.

25.3.7 MAC Packet Transmission

The transmit FIFO is 1KB. AGL begins packet transmission when the FIFO has an entire packet or the FIFO length exceeds a programmed number of bytes (AGL_GMX_TX(0)_THRESH). The threshold value should be large enough to avoid underflow and small enough to avoid excessive latencies. The recommended value for this register is the reset value of the register that will work across the widest number of configurations. In half-duplex mode, if the packet suffers a collision, it is automatically retried AGL_GMX_TX_COL_ATTEMPT times. If the packet still collides, AGL discards the packet, sets AGL_GMX_TX_INT_REG[XSCOL<0/1> / XSDEF<0/1>], and an interrupt will occur if AGL_GMX_TX_INT_EN[XSCOL<0/1> / XSDEF<0/1>] is set.

Cavium Confidential For Bowen Yang A.U 02/27/2014

25.3.8 MAC Transmit-Packet Options

The transmit machine can prepend/append various parts of a standard Ethernet frame, as shown in Table 25–2:

Table 25–2 Transmit-Packet Options

Field	Control Register[Field]	Comment
PREAMBLE/SFD	AGL_GMX_TX(0)_APPEND[PREAMBLE]	Prepend the standard PREAMBLE and SFD at the start of each packet.
PAD	AGL_GMX_TX(0)_APPEND[PAD]/ AGL_GMX_TX(0)_MIN_PKT[MIN_SIZE]	Append 0s in order to pad out a packet to the minimum packet size.
FCS	AGL_GMX_TX(0)_APPEND[FCS]	Compute and append the standard FCS at the end of the packet.

25.3.9 MAC Collision Processing

AGL fully supports half-duplex CDMA/CS operation in 10/100/1000Mb/s modes. In the event of a detected collision, AGL sets AGL_GMX_RX(0)_INT_REG[COLDET] and takes the following steps:

- the AGL transmit engine bursts the JAM pattern, backs off the bus, and retransmits the packet later.
- CN71XX carrier extends to meet the higher slottime of 1000Mb/s operation.
- CN71XX attempts retransmission AGL_GMX_TX_COL_ATTEMPT[LIMIT] times before giving up. If the retransmission limit is reached, it is considered an excessive collision (refer to XSCOL in Table 25–4). The packet is drained and AGL resets the collision count for the next packet.

Late collisions (i.e. collisions that occur after the slottime is satisfied) are considered an error and the packet is not retransmitted.

25.3.10 Bursts

CN71XX supports packet bursting on half-duplex links in 1000Mb/s mode. The RGMII transmit continues sending packets (if packets are available to send) up until the burst limit (AGL_GMX_PRT(0)_CFG[BURST]). The IFG cycles are filled with carrier extends.

25.3.11 Connectivity

The AGL interface, as defined by the 802.3 IEEE specification, is intended to connect a MAC device (CN71XX) to a PHY device. While this is the intended usage model, it is possible to use the AGL interface to connect the CN71XX to a MAC device (i.e. another CN71XX) instead of a PHY device (MAC and PHY devices are not interchangeable). The RGMII specification is not completely symmetrical with respect to the TX and RX buses.

Connecting CN71XX-to-CN71XX in a MAC-to-MAC system requires board/system design work to ensure that the AGL interface timing is met. (See Section 34.4 for timing information.) While the CN71XX can support MAC-to-MAC connections, not all MAC devices may support this. Care should be taken when choosing and connecting the CN71XX to a non-OCTEON MAC device. The MAC device must be able to support a MAC link partner instead of a PHY partner.

Cavium Confidential For Bowen Yang ADI 12/27/2014

25.4 Wake-on-LAN (WOL)

CN71XX supports power-savings features in which the network can wake the core from a power-savings state. This allows the core to be placed in a low-power state, but still allow remote and networked administration of the device. The CN71XX core can be awakened either by directed packets or by magic packets.

25.4.1 Packet Detection

AGX supports WOL via four types of packets:

- directed exact match
- directed multicast match
- directed broadcast match
- magic packet match

When operating in wake-on-LAN mode, the configured MAC port unconditionally drops all received packets. The MAC will raise an interrupt to indicate that either a directed or magic packet has been detected and the system is requested to exit the low-power state.

25.4.1.1 Directed-Packet Match

When a MAC port is configured for direct-packet-match mode, all received packets are applied to the DMAC address filter as defined by dmac_addr_filter function as described in Section 25.3.4.4.

- When the function returns ACCEPT, an interrupt is asserted.
- No action is taken upon REJECT.

Care must be taken when AGL_GMX_RX(0)_ADR_CTL[CAM_MODE] = 0, as the DMAC filter is operating in inversion mode. AGX has eight DMAC registers to use for exact address matches for either directed or directed multicast DMAC.

25.4.1.2 Magic-Packet Match

When a MAC port is configured for magic-packet-match mode, all received packets are parsed for the magic-data pattern. The pattern consists of six synchronization bytes of 0xFF followed by the AGL_GMX_SMAC(0)[SMAC<47:0>] repeated 16 times.

The pattern can be located anywhere in the packet once the DMAC, SMAC, and TYPE are parsed. The pattern must not be interrupted for magic-packet detection to occur. An example magic packet might look like the following:

```
UDD_SKIP
DMAC
SMAC
TYPE
[optional bytes]
FF FF FF FF FF FF   // Synchronization
SMAC0
SMAC1
SMAC2
```

Cavium Confidential For Bowen Yang BGA LD 02/20/2014

:
:
SMAC15
[optional bytes]
FCS

Magic packets are not required to have a good FCS to have a match.

25.4.2 Operation

When software determines it wants to enter WOL mode, it sets AGL_GMX_WOL_CTL[EN] = 1.

This places the MAC in WOL mode, which causes the MAC to drop incoming packets. The drop mechanism uses the RX FIFO full path. Packets in that have been partially buffered in the RX FIFO when WOL mode is entered are terminated with the PARTIAL opcode error. All subsequent incoming packets will be completely dropped. Packets will continue to be dropped for the duration of the WOL event (i.e. while AGL_GMX_WOL_CTL[EN] = 1).

Once AGL_GMX_WOL_CTL[EN] is set, software can reconfigure the DMAC filtering, UDD_SKIP, and MAGIC packet control to achieve the desired packet matching. Systems can choose to use either directed-packet match, magic-packet match, or both. Mulitple MACs can be enabled for WOL matching simultaneously and have distinct matching modes and criteria.

Software can then begin WOL-event detection on a individual port by setting AGL_GMX_WOL_CTL[DIRECT_EN<prt>/MAGIC_EN<prt>] = 1.

When a WOL event occurs, it asserts the interrupt bit, AGL_GMX_RX(0)_INT_REG[WOL] and notifies CIU if the corresponding enable bit is set (AGL_GMX_RX(0)_INT_EN[WOL].

The MAC will continue to drop packets. The interrupt handler can put the MACs back into normal operation and deassert AGL_GMX_WOL_CTL[EN]; packets will then begin flowing in.

The CAMs can be disabled in directed-packet mode by configuring the DCAM CAM properly.

AGL_GMX_RX(0)_ADR_CTL[CAM_MODE] = 1
AGL_GMX_RX(0)_ADR_CTL[MCST] = 1
AGL_GMX_RX(0)_ADR_CTL[BCST] = 0
AGL_GMX_RX(0)_ADR_CAM_EN[EN] = 0

25.5 AGL Errors/Exceptions

AGL implements several error and exception checks. The following subsections describe the exception checkers and how the exception is communicated to software.

Cavium Confidential For Bowen Yang ALU 02/27/2014

25.5.1 AGL MAC Receive Error/Exception Checks

Exceptions can be enabled by setting the appropriate bit in AGL_GMX_RX(0)_FRM_CHK. If the bit is set, the exception is logged in AGL_GMX_RX(0)_INT_REG and can optionally raise an interrupt based on the corresponding bits in AGL_GMX_RX(0)_INT_EN. In addition, packets arrive with an input ring completion code that indicates the error.

The exceptions, their causes, and how they are handled are shown in Table 25–4.

Table 25–3 Receive Errors/Exceptions

Exception/ Completion Code	Cause	Notification
PARTIAL	Any part of a packet was dropped due to lack of space in the receive FIFO. This may occur commonly during normal operation.	This check is always on and sets WORD2[RE] in the work-queue entry to 0x1. The partial packet is buffered and received with a completion code of 0x1.
JABBER_ERR	A packet was received with length > AGL_GMX_RX(0)_JABBER bytes.	The packet is unconditionally truncated at AGL_GMX_RX(0)_JABBER bytes. AGL_GMX_RX(0)_FRM_CHK[JABBER] enables the check. If enabled, AGL_GMX_RX(0)_INT_REG[JABBER] is set. The truncated packet is buffered and received with a completion code of 0x2.
OVER_ERR OVER_FCS	A packet was received with length > AGL_GMX_RX(0)_FRM_MAX bytes. For tagged frames, MAX increases by four bytes for each VLAN found up to a maximum of two VLANs, or MAX + 8 bytes.	AGL_GMX_RX(0)_FRM_CHK[MAXERR] enables the check. If enabled, AGL_GMX_RX(0)_INT_REG[MAXERR] is set to 1. The packet is buffered and received with a completion code of 0x3 or 0x4.
ALIGN_ERR	A packet was received that fails the FCS check (if enabled) and was not an integer number of bytes. If the FCS check is disabled, all packets that are not an integer number of bytes are considered ALIGN_ERRs.	AGL_GMX_RX(0)_FRM_CHK[ALNERR] enables the alignment check and AGL_GMX_RX(0)_FRM_CHK[FCSERR] enables the FCS check. If enabled, AGL_GMX_RX(0)_INT_REG[ALNERR] is set. The packet is buffered and received with a completion code of 0x5.
UNDER_ERR UNDER_FCS	A packet was received with length < AGL_GMX_RX(0)_FRM_MIN bytes.	AGL_GMX_RX(0)_FRM_CHK[MINERR] enables the check. If enabled, AGL_GMX_RX(0)_INT_REG[MINERR] is set to 1. The packet is buffered and received with a completion code of 0x6 or 0x8.
FCS_ERR	A packet was received that failed the FCS/ CRC check.	AGL_GMX_RX(0)_FRM_CHK[FCSERR] enables the check. If enabled, AGL_GMX_RX(0)_INT_REG[FCSERR] is set to 1. The packet is buffered and received with a completion code of 0x7.
CAREXT_ERR	A packet was received with one or more carrier extend errors.	AGL_GMX_RX(0)_FRM_CHK[CAREXT] enables the check. If enabled, AGL_GMX_RX(0)_INT_REG[CAREXT] is set to 1. The packet is buffered and received with a completion code of 0x9.
LENGTH_MIS	A packet's received length does not match the length field within the packet. The check is only valid for packets with a valid length field (64–1500 bytes). For untagged frames, the lengths must exactly match. For tagged frames, the check can either be optionally disabled or the checker will account for the inserted VLAN bytes.	AGL_GMX_RX(0)_FRM_CHK[LENERR] enables the check for all packets. When AGL_GMX_RX(0)_FRM_CTL[VLAN_LEN] = 1, the check is suppressed for VLAN packets. If enabled, AGL_GMX_RX(0)_INT_REG[LENERR] is set to 1. The packet is buffered and received with a completion code of 0xA.

Cavium Confidential For Bowen Yang @ALI 07/2014

Table 25–3 Receive Errors/Exceptions

Exception/ Completion Code	Cause	Notification
DAT_ERR	A packet was received with one more cycles of data-reception error (RXERR).	AGL_GMX_RX(0)_FRM_CHK[RCVERR] enables the check. If enabled, AGL_GMX_RX(0)_INT_REG[RCVERR] is set to 1. The packet is buffered and received with a completion code of 0xB.
SKIP_ERR	A packet was received in which the length < AGL_GMX_RX(0)_UDD_SKP[LEN] bytes. This means that there was not enough packet data in order to get passed the Skip field and into the L2 HDR.	AGL_GMX_RX(0)_FRM_CHK[SKPERR] enables the check. If enabled, AGL_GMX_RX(0)_INT_REG[SKPERR] is set to 1. The packet is buffered and received with a completion code of 0xC.
NIB_ERR	A packet received had a stutter error (data not repeated on both RGMII clock edges. 10/100Mbs only).	AGL_GMX_RX(0)_FRM_CHK[NIBERR] enables the check. If enabled, AGL_GMX_RX(0)_INT_REG[NIBERR] is set to 1. The packet is buffered and received with a completion code of 0xD.
Bad preamble	A packet with a bad preamble was received. The packet is discarded.	AGL_GMX_RX(0)_FRM_CTL[PRE_CHK/PRE_FREE] enables the preamble checker. If enabled, AGL_GMX_RX(0)_INT_REG[PCTERR] is set to 1.
Reserved Opcode	AGL detected a reserved opcode on the AGL interface.	This check is always on and sets AGL_GMX_RX(0)_INT_REG[RSVERR] to 1.
False carrier	AGL detected a false carrier on the AGL interface.	This check is always on and sets AGL_GMX_RX(0)_INT_REG[FALERR] to 1.
Collision	In half-duplex mode, AGL detected a collision. This can and will occur under normal operation in half-duplex mode. The packet is normally retransmitted.	This check is always on and sets AGL_GMX_RX(0)_INT_REG[COLDET] to 1.
Inter-Frame Gap Error	AGL detected an interframe gap violation. This exception does not necessarily indicate a system failure.	This check is always on and sets AGL_GMX_RX(0)_INT_REG[IFGERR] to 1.
Overflow	Internal FIFO overflow occurred.	This check is always on and sets AGL_GMX_BAD_REG[OVRFLW] to 1.
Pause packet dropped	A pause packet was dropped due to lack of space in the receive FIFO. This may occur commonly during normal operation, depending on configuration. Refer to Section 25.3.4.3.	This check is always on and sets AGL_GMX_RX(0)_INT_REG[PAUSE_DRP] to 1. It also loads AGL_GMX_RX(0)_PAUSE_DROP_TIME[STATUS] with pause_time of the pause packet.

Cavium Confidential
Bowen Tang
ALD
02/27/2014

25.5.2 AGL MAC Transmit Error/Exception Checks

Transmit exceptions are logged in two registers.

- AGL_GMX_TX_INT_REG
- AGL_GMX_BAD_REG

AGL_GMX_TX_INT_REG contains the most common exceptions and can optionally raise an interrupt based on the corresponding bits in AGL_GMX_TX_INT_EN. AGL_GMX_BAD_REG contains internal checks that should never assert in normal operation.

The exceptions, their causes, and how they are handled are shown in Table 25–4.

Table 25–4 Transmit Errors/Exceptions

Exception	Cause	Notification
Underflow	In the unlikely event that AGL cannot keep the outbound FIFO full, the packet transfer will underflow. This should be detected by the receiving device as an FCS error. Internally, the packet is drained and lost.	This check is always on and AGL sets AGL_GMX_TX_INT_REG[UNDFLW<0/1>].
Excessive Collision	An excessive collision occurs in half-duplex mode if AGL detects a collision during each of AGL_GMX_TX_COL_ATTEMPT[LIMIT] attempts to send a packet. Internally, the packet is drained and lost.	This check is on when AGL_GMX_TX_COL_ATTEMPT[LIMIT] ≠ 0 AGL sets AGL_GMX_TX_INT_REG[XSCOL<0/1>] if AGL_GMX_TX(0)_CTL[XSCOL_EN] = 1.
Excessive Deferral	An excessive collision occurs in half-duplex mode if AGL detects that it cannot send a packet on the transmit interface for maxDeferTime (as defined by IEEE 802.3 specification) time.	This check is always on and AGL sets AGL_GMX_TX_INT_REG[XSDEF<0/1>] if AGL_GMX_TX(0)_CTL[XSDEF_EN] = 1.
Late Collision	A late collision occurs in half-duplex mode if AGL detects a collision after the slot time for the packet. AGL discards the packet and receivers are likely to encounter an FCS error.	This check is always on and AGL sets AGL_GMX_TX_INT_REG[LATE_COL<0/1>].
OUT_OVR LOSTSTAT STATOVR OVRFLW TXPOP TXPSH	The transmitter has several internal error checks. These checks should never assert in a correctly configured system. Any assertion indicates an internal problem has occurred.	AGL sets AGL_GMX_BAD_REG[OUT_OVR, LOSTSTAT, STATOVR, OVRFLW, TXPOP, TXPSH]

25.6 AGL Interrupts

There are two sources of interrupts from AGL:

- AGL_GMX_RX(0)_INT_REG AND AGL_GMX_RX(0)_INT_EN
- AGL_GMX_TX_INT_REG AND AGL_GMX_TX_INT_EN

The AGL_GMX_RX(0)_INT_REG and AGL_GMX_TX_INT_REG interrupts tend to be error interrupts. Section 25.5 describes these interrupts. These interrupts can cause CIU_BLOCK_INT[AGL] to set, which can cause core interrupts. Refer to Chapter 13.

Cavium Confidential For Bowen Yang ALU 02/27/2014

25.7 Configuration, Reset, and Link Status Changes

25.7.1 AGL Bring-up Sequence

This section describes a sequence of steps that initialize AGL for packet receipt and transmit.

1. Initialize other AGL configuration (other than AGL_GMX_PRT(0)_CFG and AGL_PRT(0)_CTL). For typical situations where CN71XX is directly connected via AGL to a 10/100/1000 PHY, none is needed. The reset values are appropriate for most of these situations, with the possible exception of the DMAC filter. (Section 25.3.4.4 describes the DMAC filter.)

2. Initialize the physical layer.
   a. RGMII Initialization
      – Write AGL_PRT(0)_CTL[REFCLK_SEL] to select the desired RGMII DLL reference clock. Normally REFCLK_SEL will be set to 0 to select the RGM_REF_CLK input; but other options are available. See REFCLK_SEL description in Section 25.9.
      – Take the DLL out of reset: set AGL_PRT(0)_CTL[DLLRST] to 0.
      – Take the clock tree out of reset: set AGL_PRT(0)_CTL[CLKRST] = 0.
      – Once the 125MHz reference clock is stable, allow 256 cycles for the DLL to lock.
      – Optionally, bypass the DLL setting with AGL_PRT(0)_CTL[CLKTX_BYP, CLKRX_BYP, CLKTX_SET, CLKRX_SET].

NOTE: The RGMII timing specification requires that devices transmit clock and data synchronously. The specification requires external sources (namely the PC board trace routes) to introduce the appropriate 1.5 to 2.0 ns of delay.

To eliminate the need for the PC board delays, the AGL RGMII interface has optional onboard DLLs for both transmit and receive. For correct operation, at most one of the transmitter, board, or receiver involved in an RGMII link should introduce delay. By default/reset, the AGL RGMII receivers delay the received clock, and the AGL RGMII transmitters do not delay the transmitted clock. Whether this default works as-is with a given link partner depends on the behavior of the link partner and the PC board.
These are the possible modes of AGL RGMII receive operation:

- AGL_PRT(0)_CTL[CLKRX_BYP] = 0 (reset value) – The CN71XX AGL RGMII receive interface introduces clock delay using its internal DLL. This mode is appropriate if neither the remote transmitter nor the PC board delays the clock.
- AGL_PRT(0)_CTL[CLKRX_BYP] = 1, [CLKRX_SET] = 0x0 – The CN71XX AGL RGMII receive interface introduces no clock delay. This mode is appropriate if either the remote transmitter or the PC board delays the clock.

These are the possible modes of AGL RGMII transmit operation:

- AGL_PRT(0)_CTL[CLKTX_BYP] = 1, [CLKTX_SET] = 0x0 (reset value) – The CN71XX AGL RGMII transmit interface introduces no clock delay. This mode is appropriate is either the remote receiver or the PC board delays the clock.
- AGL_PRT(0)_CTL[CLKTX_BYP] = 0 – The CN71XX AGL RGMII transmit interface introduces clock delay using its internal DLL. This mode is appropriate if neither the remote receiver nor the PC board delays the clock.

Cavium Confidential For Bowen Yang 02.27.2015

- Bypass the compensation controller setting with AGL_PRT(0)_CTL[DRV_BYP, DRV_NCTL, DRV_PCTL].
- Force the previous write to be visible to hardware by reading back AGL_PRT(0)_CTL.
- Enable the compensation controller by setting AGL_PRT(0)_CTL[COMP] = 1.
- Force the previous write to be visible to hardware by reading back AGL_PRT(0)_CTL.
- Wait 1040 reference clock cycles for the compensation state machine to lock.
- If the compensation controller is used, release the bypass AGL_PRT(0)_CTL[DRV_BYP].

3. Set AGL_PRT(0)_CTL[CLKRST] to 1.
4. Write AGL_GMX_PRT(0)_CFG, setting [TX_EN], [RX_EN], and [EN] to 1. DUPLEX should be appropriate to the desired link status. This requires the link status to be determined. The link status is typically determined via the SMI/MDIO interface. Refer to Chapter 28.
5. Once RGMII RXC clock is stable, enable the interface, set AGL_PRT(0)_CTL[ENABLE] = 1.
6. Initialize interrupt enables so that desired interrupts can be received. This includes AGL_GMX_RX(0)_INT_EN and AGL_GMX_TX_INT_EN.

   Initialize other AGL configuration (other than AGL_GMX_PRT(0)_CFG). For typical situations where CN71XX is directly connected via MIX to a 10/100/1000 PHY, none is needed.

25.7.2 AGL Interface link status changes

Status changes while the AGL link is operational are typically determined via the SMI/MDIO interface (see Chapter 28). If the PHY supports optional in-band status, then software can use AGL_GMX_RX(0)_RX_INBND[STATUS,SPEED,DUPLEX].

An AGL Block Reset (refer to Section 25.7.2) should be used to reprogram AGL for the new link status.

Cavium Confidential For Bowen Yang ALU 02/27/2014

25.8 Statistics

AGL supports a rich statistics gathering, described in Table 25–5, in order to support RX RMON and TX RMON. Refer to the CSR descriptions in Section 25.9 for a complete list of statistics.

Table 25–5 AGL Statistics Gathering

CSR Field	Statistic
RX	
AGL_GMX_RX0/1_STATS_PKTS[CNT]	32-bit count of all good packets
AGL_GMX_RX0/1_STATS_OCTS[CNT]	48-bit count of all bytes from good packets
AGL_GMX_RX0/1_STATS_PKTS_CTL[CNT]	32-bit count of all control/pause packets
AGL_GMX_RX0/1_STATS_OCTS_CTL[CNT]	48-bit count of all bytes from control/pause packets
AGL_GMX_RX0/1_STATS_PKTS_DMAC[CNT]	32-bit count of all DMAC filtered packets
AGL_GMX_RX0/1_STATS_OCTS_DMAC[CNT]	48-bit count of all bytes from DMAC filtered packets
AGL_GMX_RX0/1_STATS_PKTS_DRP[CNT]	32-bit count of all packets dropped due to RX FIFO full
AGL_GMX_RX0/1_STATS_OCTS_DRP[CNT]	48-bit count of all bytes from packets dropped due to RX FIFO full
AGL_GMX_RX0/1_STATS_PKTS_BAD[CNT]	32-bit count of all bad packets
TX	
AGL_GMX_TX0/1_STAT0[XSCOL]	32-bit count of packets dropped due to excessive collisions
AGL_GMX_TX0/1_STAT0[XSDEF]	32-bit count of packets dropped due to excessive deferral
AGL_GMX_TX0/1_STAT1[MCOL]	32-bit count of packets sent that experienced multiple collisions before successful transmission
AGL_GMX_TX0/1_STAT1[SCOL]	32-bit count of packets sent that experienced a single collision before successful transmission
AGL_GMX_TX0/1_STAT2[OCTS]	48-bit count of all bytes sent
AGL_GMX_TX0/1_STAT3[PKTS]	32-bit count of all packets sent
AGL_GMX_TX0/1_STAT4[HIST0]	32-bit count of packets sent with an octet count < 64
AGL_GMX_TX0/1_STAT4[HIST1]	32-bit count of packets sent with an octet count == 64
AGL_GMX_TX0/1_STAT5[HIST2]	32-bit count of packets sent with an octet count of 65–127
AGL_GMX_TX0/1_STAT5[HIST3]	32-bit count of packets sent with an octet count of 128–255
AGL_GMX_TX0/1_STAT6[HIST4]	32-bit count of packets sent with an octet count of 256–511
AGL_GMX_TX0/1_STAT6[HIST5]	32-bit count of packets sent with an octet count of 512–1023
AGL_GMX_TX0/1_STAT7[HIST6]	32-bit count of packets sent with an octet count of 1024–1518
AGL_GMX_TX0/1_STAT7[HIST7]	32-bit count of packets sent with an octet count of > 1518
AGL_GMX_TX0/1_STAT8[BCST]	32-bit count of packets sent to a multicast DMAC
AGL_GMX_TX0/1_STAT8[MCST]	32-bit count of packets sent to the broadcast DMAC
AGL_GMX_TX0/1_STAT9[CTL]	32-bit count of control/pause packets sent
AGL_GMX_TX0/1_STAT9[UNDFLW]	32-bit count of packets sent that experienced a transmit underflow and were truncated

Cavium Confidential For Bowen Yang 01/27/14

25.9 AGL Registers

The AGL registers are listed in Table 25–6.

Table 25–6 AGL Registers

Register	Address	CSR Type[1]	Detailed Description
AGL_GMX_RX(0)_INT_REG	0x11800E0000000 + α*0x800	RSL	See page 1194
AGL_GMX_RX(0)_INT_EN	0x11800E0000008 + α*0x800	RSL	See page 1195
AGL_GMX_PRT(0)_CFG	0x11800E0000010 + α*0x800	RSL	See page 1196
AGL_GMX_RX(0)_FRM_CTL	0x11800E0000018 + α*0x800	RSL	See page 1197
AGL_GMX_RX(0)_FRM_CHK	0x11800E0000020 + α*0x800	RSL	See page 1198
AGL_GMX_RX(0)_FRM_MIN	0x11800E0000028 + α*0x800	RSL	See page 1198
AGL_GMX_RX(0)_FRM_MAX	0x11800E0000030 + α*0x800	RSL	See page 1199
AGL_GMX_RX(0)_JABBER	0x11800E0000038 + α*0x800	RSL	See page 1199
AGL_GMX_RX(0)_DECISION	0x11800E0000040 + α*0x800	RSL	See page 1200
AGL_GMX_RX(0)_UDD_SKP	0x11800E0000048 + α*0x800	RSL	See page 1200
AGL_GMX_RX(0)_STATS_CTL	0x11800E0000050 + α*0x800	RSL	See page 1201
AGL_GMX_RX(0)_IFG	0x11800E0000058 + α*0x800	RSL	See page 1201
AGL_GMX_RX(0)_RX_INBND	0x11800E0000060 + α*0x800	RSL	See page 1201
AGL_GMX_RX(0)_PAUSE_DROP_TIME	0x11800E0000068 + α*0x800	RSL	See page 1202
AGL_GMX_RX(0)_STATS_PKTS	0x11800E0000080 + α*0x800	RSL	See page 1202
AGL_GMX_RX(0)_STATS_OCTS	0x11800E0000088 + α*0x800	RSL	See page 1202
AGL_GMX_RX(0)_STATS_PKTS_CTL	0x11800E0000090 + α*0x800	RSL	See page 1203
AGL_GMX_RX(0)_STATS_OCTS_CTL	0x11800E0000098 + α*0x800	RSL	See page 1203
AGL_GMX_RX(0)_STATS_PKTS_DMAC	0x11800E00000A0 + α*0x800	RSL	See page 1203
AGL_GMX_RX(0)_STATS_OCTS_DMAC	0x11800E00000A8 + α*0x800	RSL	See page 1204
AGL_GMX_RX(0)_STATS_PKTS_DRP	0x11800E00000B0 + α*0x800	RSL	See page 1204
AGL_GMX_RX(0)_STATS_OCTS_DRP	0x11800E00000B8 + α*0x800	RSL	See page 1204
AGL_GMX_RX(0)_STATS_PKTS_BAD	0x11800E00000C0 + α*0x800	RSL	See page 1204
AGL_GMX_RX(0)_ADR_CTL	0x11800E0000100 + α*0x800	RSL	See page 1205
AGL_GMX_RX(0)_ADR_CAM_EN	0x11800E0000108 + α*0x800	RSL	See page 1205
AGL_GMX_RX(0)_ADR_CAM(0..5)	0x11800E0000180 + α*0x800	RSL	See page 1206
AGL_GMX_TX(0)_CLK	0x11800E0000208 + α*0x800	RSL	See page 1206
AGL_GMX_TX(0)_THRESH	0x11800E0000210 + α*0x800	RSL	See page 1206
AGL_GMX_TX(0)_APPEND	0x11800E0000218 + α*0x800	RSL	See page 1207
AGL_GMX_SMAC(0)	0x11800E0000230 + α*0x800	RSL	See page 1207
AGL_GMX_TX(0)_PAUSE_PKT_TIME	0x11800E0000238 + α*0x800	RSL	See page 1208
AGL_GMX_TX(0)_MIN_PKT	0x11800E0000240 + α*0x800	RSL	See page 1208
AGL_GMX_TX(0)_PAUSE_PKT_INTERVAL	0x11800E0000248 + α*0x800	RSL	See page 1209
AGL_GMX_TX(0)_SOFT_PAUSE	0x11800E0000250 + α*0x800	RSL	See page 1209
AGL_GMX_TX(0)_PAUSE_TOGO	0x11800E0000258 + α*0x800	RSL	See page 1209
AGL_GMX_TX(0)_PAUSE_ZERO	0x11800E0000260 + α*0x800	RSL	See page 1210
AGL_GMX_TX(0)_STATS_CTL	0x11800E0000268 + α*0x800	RSL	See page 1210
AGL_GMX_TX(0)_CTL	0x11800E0000270 + α*0x800	RSL	See page 1210
AGL_GMX_TX(0)_STAT0	0x11800E0000280 + α*0x800	RSL	See page 1210

Cavium Confidential For Beverly Yang ALU 2/27/2014

Table 25–6 AGL Registers

Register	Address	CSR Type[1]	Detailed Description
AGL_GMX_TX(0)_STAT1	0x11800E0000288 + α*0x800	RSL	See page 1211
AGL_GMX_TX(0)_STAT2	0x11800E0000290 + α*0x800	RSL	See page 1211
AGL_GMX_TX(0)_STAT3	0x11800E0000298 + α*0x800	RSL	See page 1211
AGL_GMX_TX(0)_STAT4	0x11800E00002A0 + α*0x800	RSL	See page 1212
AGL_GMX_TX(0)_STAT5	0x11800E00002A8 + α*0x800	RSL	See page 1212
AGL_GMX_TX(0)_STAT6	0x11800E00002B0 + α*0x800	RSL	See page 1212
AGL_GMX_TX(0)_STAT7	0x11800E00002B8 + α*0x800	RSL	See page 1213
AGL_GMX_TX(0)_STAT8	0x11800E00002C0 + α*0x800	RSL	See page 1213
AGL_GMX_TX(0)_STAT9	0x11800E00002C8 + α*0x800	RSL	See page 1213
AGL_GMX_BIST	0x11800E0000400	RSL	See page 1214
AGL_GMX_RX_BP_DROP(0)	0x11800E0000420 + α*0x8	RSL	See page 1214
AGL_GMX_RX_BP_ON(0)	0x11800E0000440 + α*0x8	RSL	See page 1214
AGL_GMX_RX_BP_OFF(0)	0x11800E0000460 + α*0x8	RSL	See page 1214
AGL_GMX_TX_IFG	0x11800E0000488	RSL	See page 1215
AGL_GMX_TX_JAM	0x11800E0000490	RSL	See page 1215
AGL_GMX_TX_COL_ATTEMPT	0x11800E0000498	RSL	See page 1215
AGL_GMX_TX_PAUSE_PKT_DMAC	0x11800E00004A0	RSL	See page 1216
AGL_GMX_TX_PAUSE_PKT_TYPE	0x11800E00004A8	RSL	See page 1216
AGL_GMX_TX_OVR_BP	0x11800E00004C8	RSL	See page 1216
AGL_GMX_TX_BP	0x11800E00004D0	RSL	See page 1216
AGL_GMX_RX_PRT_INFO	0x11800E00004E8	RSL	See page 1217
AGL_GMX_TX_LFSR	0x11800E00004F8	RSL	See page 1217
AGL_GMX_TX_INT_REG	0x11800E0000500	RSL	See page 1217
AGL_GMX_TX_INT_EN	0x11800E0000508	RSL	See page 1218
AGL_GMX_BAD_REG	0x11800E0000518	RSL	See page 1218
AGL_GMX_STAT_BP	0x11800E0000520	RSL	See page 1219
AGL_GMX_WOL_CTL	0x11800E0000780	RSL	See page 1219
AGL_GMX_RX_TX_STATUS	0x11800E00007E8	RSL	See page 1219
AGL_PRT(0)_CTL	0x11800E0002000 + α*0x8	RSL	See page 1220

1. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

AGL Interrupt Register
AGL_GMX_RX(0)_INT_REG

This register allows interrupts to be sent to the control processor. If an error is noted in this register, and the corresponding bit in AGL_GMX_RX(0)_INT_EN is set, an exception is raised. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Type Description
<63:30>	—	RAZ	—	—	Reserved
<29>	WOL	R/W1C/H	0	0	A Wake-on-LAN event has occurred.
<28:20>	—	RAZ	—	—	Reserved
<19>	PAUSE_DRP	R/W1C/H	0	0	Pause packet was dropped due to full GMX RX FIFO.
<18>	PHY_DUPX	R/W1C/H	0	0	Change in the RGMII inbound LinkDuplex.
<17>	PHY_SPD	R/W1C/H	0	0	Change in the RGMII inbound LinkSpeed.
<16>	PHY_LINK	R/W1C/H	0	0	Change in the RGMII inbound LinkStatus.
<15>	IFGERR	R/W1C/H	0	0	Interframe gap violation, which does not necessarily indicate a failure.
<14>	COLDET	R/W1C/H	0	0	Collision detection. Collisions can only occur in half-duplex mode. A collision is assumed by the receiver when the received frame is less than AGL_GMX_RX(0)_FRM_MIN — this is normally a MINERR.
<13>	FALERR	R/W1C/H	0	0	False-carrier error, or carrier-extend error after slottime is satisfied.
<12>	RSVERR	R/W1C/H	0	0	Packet reserved opcode error.
<11>	PCTERR	R/W1C/H	0	0	Bad preamble/protocol error. Checks that the frame begins with a valid PREAMBLE sequence. Does not check the number of PREAMBLE cycles.
<10>	OVRERR[1]	R/W1C/H	0	0	Internal data aggregation overflow error. This interrupt should never assert.
<9>	NIBERR	R/W1C/H	0	0	Nibble error (hi_nibble ≠ lo_nibble)
<8>	SKPERR[1]	R/W1C/H	0	0	Skipper error
<7>	RCVERR[1]	R/W1C/H	0	0	Data-reception error. Frame was received with packet data-reception error.
<6>	LENERR[1]	R/W1C/H	0	0	Length error: frame was received with a length error. Length errors occur when the received packet does not match the length field. LENERR is only checked for packets between 64 and 1500 bytes. For untagged frames, the length must be an exact match. For tagged frames, the checker takes into account the additional tagged bytes - up to 2 tags. The check can be completely disabled for all tagged frames using AGL_GMX_RX(0)_FRM_CTL[VLAN_LEN].
<5>	ALNERR[1]	R/W1C/H	0	0	Alignment error: frame was received with an alignment error. Indicates that the packet received was not an integer number of bytes. If FCS checking is enabled, ALNERR only asserts if the FCS is bad. If FCS checking is disabled, ALNERR asserts in all non-integer frame cases
<4>	FCSERR[1]	R/W1C/H	0	0	FCS/CRC error. Frame was received with FCS/CRC error
<3>	JABBER[1]	R/W1C/H	0	0	System-length error: frame was received with length > sys_length. An RX JABBER error indicates that a packet was received that is longer than the maximum allowed packet as defined by the system. GMX truncates the packet at the JABBER count. Failure to do so could lead to system instability.

Cavium Confidential For Bowen Yang 03/26/14

Bit Pos	Field Name	Access	Reset Value	Typical Value	Type Description
<2>	MAXERR[1]	R/W1C/H	0	0	Maximum-length error: frame was received with length > max_length. For untagged frames, the total frame DA+SA+TL+DATA+PAD+FCS > AGL_GMX_RX(0)_FRM_MAX. For tagged frames, DA+SA+VLAN+TL+DATA+PAD+FCS > AGL_GMX_RX(0)_FRM_MAX + (4 × VLAN_VAL) + (4 × VLAN_STACKED).
<1>	CAREXT	R/W1C/H	0	0	Carrier extend error
<0>	MINERR[1]	R/W1C/H	0	0	Minimum-length error. Frame was received with length < min_length. Total frame DA+SA+TL+DATA+PAD+FCS < AGL_GMX_RX(0)_FRM_MIN.

1. Exception conditions <10:0> can also set the **rcv/opcode** in the received packet's work-queue entry. AGL_GMX_RX(0)_FRM_CHK provides a bit mask for configuring which conditions set the error.

Interrupt Enable Register
AGL_GMX_RX(0)_INT_EN

This register provides the interrupt-enable bits for the corresponding error in AGL_GMX_RX(0)_INT_REG. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:30>	—	RAZ	—	—	Reserved
<29>	WOL	R/W/H	0	0	A Wake-on-LAN event has occurred.
<28:20>	—	RAZ	—	—	Reserved
<19>	PAUSE_DRP	R/W/H	0	0	Pause packet was dropped due to full GMX RX FIFO.
<18>	PHY_DUPX	R/W	0	0	Change in the RMGII inbound LinkDuplex
<17>	PHY_SPD	R/W	0	0	Change in the RMGII inbound LinkSpeed
<16>	PHY_LINK	R/W	0	0	Change in the RMGII inbound LinkStatus
<15>	IFGERR	R/W	0	0	Interframe-gap violation.
<14>	COLDET	R/W	0	0	Collision detection.
<13>	FALERR	R/W	0	0	False-carrier error, or carrier-extend error after slottime is satisfied.
<12>	RSVERR	R/W	0	0	Packet reserved opcodes.
<11>	PCTERR	R/W	0	0	Bad preamble / protocol.
<10>	OVRERR	R/W	0	0	Internal data aggregation overflow error.
<9>	NIBERR	RAZ	0	0	Nibble error (hi_nibble ≠ lo_nibble)
<8>	SKPERR	R/W	0	0	Skipper error.
<7>	RCVERR	R/W	0	0	Frame was received with AGL data reception error.
<6>	LENERR	R/W	0	0	Frame was received with length error.
<5>	ALNERR	R/W	0	0	Frame was received with an alignment error.
<4>	FCSERR	R/W	0	0	Frame was received with FCS/CRC error.
<3>	JABBER	R/W	0	0	Frame was received with length > sys_length.
<2>	MAXERR	R/W	0	0	Frame was received with length > max_length.
<1>	CAREXT	RAZ	0	0	Carrier extend error
<0>	MINERR	R/W	0	0	Frame was received with length < min_length.

Cavium Confidential For Board Use Only 9/2/2014

Port Configuration Register
AGL_GMX_PRT(0)_CFG

This register controls the configuration of the port. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved
<13>	TX_IDLE	RO/H	1	—	TX machine is idle.
<12>	RX_IDLE	RO/H	1	—	RX machine is idle.
<11:9>	—	RAZ	—	—	Reserved
<8>	SPEED_MSB	R/W	0	—	Link speed most-significant bit. SPEED_MSB SPEED 0 0 = 100Mb/s operation 0 1 = 1000Mb/s operation 1 0 = 10Mb/s operation 1 1 = reserved
<7>	—	RAZ	—	—	Reserved
<6>	BURST	R/W	0	—	Half-duplex burst enable. Only valid for 1000Mb/s half-duplex operation. 0 = burst length of 0x2000 (halfdup / 1000Mb/s) 1 = burst length of 0x0 (all other modes)
<5>	TX_EN	R/W	0	—	Port enable. Must be set for CN71XX to send RMGII traffic. When this bit is clear on a given port, all packet cycles appear as inter-frame cycles.
<4>	RX_EN	R/W	0	—	Port enable. Must be set for CN71XX to receive RMGII traffic. When this bit is clear on a given port, all packet cycles appear as inter-frame cycles.
<3>	SLOTTIME	R/W	1	—	Slot time for half-duplex operation 0 = 512 bit times (10/100 Mb/s operation) 1 = 4096 bit times (1000Mb/s operation)
<2>	DUPLEX	R/W	1	—	Duplex mode: 0 = half-duplex (collisions/extensions/bursts), 1 = full-duplex
<1>	SPEED	R/W	1	—	Link speed least-significant bit SPEED_MSB SPEED 0 0 = 100Mb/s operation 0 1 = 1000Mb/s operation 1 0 = 10Mb/s operation 1 1 = reserved
<0>	EN	R/W	0	—	Link enable. When EN is clear, packets are not received or transmitted (including PAUSE and JAM packets). If EN is cleared while a packet is currently being received or transmitted, the packet is allowed to complete before the bus is idled. On the RX side, subsequent packets in a burst are ignored.

Cavium Confidential For Power Yang 2012-4-11

Frame Control Register
AGL_GMX_RX(0)_FRM_CTL

This register controls the handling of the frames. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	—	—	Reserved
<12>	PTP_MODE	R/W	0	0	Timestamp mode. When PTP_MODE is set, a 64-bit timestamp is prepended to every incoming packet. If PTP_MODE = 1 and PRE_CHK = 1, PRE_STRP must be 1. The timestamp bytes are added to the packet in such a way as to not modify the packet's receive byte count. This implies that AGL_GMX_RX(0)_JABBER, AGL_GMX_RX(0)_FRM_MIN, AGL_GMX_RX(0)_FRM_MAX, AGL_GMX_RX(0)_DECISION, AGL_GMX_RX(0)_UDD_SKP, and AGL_GMX_RX(0)_STATS_* do not require any adjustment as they operate on the received packet size.
<11>	—	RAZ	—	—	Reserved
<10>	NULL_DIS	R/W	0	0	When set, do not modify the MOD bits on NULL ticks due to PARITAL packets.
<9>	PRE_ALIGN	R/W	1	1	When set, PREAMBLE parser aligns the SFD byte regardless of the number of previous PREAMBLE nibbles. In this mode, PRE_STRP should be set to account for the variable nature of the PREAMBLE. PRE_CHK must be set to enable this and all PREAMBLE features.
<8>	PAD_LEN	R/W	0	0	Pad length-check disable. When this bit is set, it disables the length check for non-minimum-sized packets with padding in the client data.
<7>	VLAN_LEN	R/W	0	0	Length-check disable. When this bit is set, it disables the length check for VLAN packets.
<6>	PRE_FREE	RO	1	1	When set, PREAMBLE checking is less strict. AGL will begin the frame at the first SFD. This field must be set if [PRE_ALIGN] is set. [PRE_CHK] must be set to enable this and all PREAMBLE features.
<5>	CTL_SMAC	R/W	0	0	Control pause frames can match station SMAC.
<4>	CTL_MCST	R/W	1	1	Control pause frames can match globally assign multicast address.
<3>	CTL_BCK	R/W	1	1	Forward pause information to TX block. Refer to **Note:** below table.
<2>	CTL_DRP	R/W	1	1	Drop control-pause frames. Refer to **Note:** below table.
<1>	PRE_STRP	R/W	1	1	Strip off the preamble (when present). 0 = PREAMBLE + SFD is sent to core as part of frame 1 = PREAMBLE + SFD is dropped When [PRE_CHK] = 1 (indicating that the PREAMBLE will be sent), [PRE_STRP] determines if the PREAMBLE + SFD bytes are thrown away or sent to the core as part of the packet. In either mode, the PREAMBLE + SFD bytes are not counted toward the packet size when checking against the MIN and MAX bounds. Furthermore, the bytes are skipped when locating the start of the L2 header for DMAC and Control frame recognition. This field must be set if [PRE_ALIGN] is set. [PRE_CHK] must be set to enable this and all PREAMBLE features.
<0>	PRE_CHK	R/W	1	1	This port is configured to send a valid 802.3 PREAMBLE to begin every frame. AGL checks that a valid PREAMBLE is received (based on PRE_FREE). When a problem does occur within the PREAMBLE sequence, the frame is marked as bad and not sent into the core. The AGL_GMX_RX(0)_INT_REG[PCTERR] interrupt is also raised.

Cavium Confidential For Review Only 2012-12-21

Note:

- CTL_BCK/CTL_DRP

  These bits control how the hardware handles incoming PAUSE packets. The most common modes of operation:

  - CTL_BCK = 1, CTL_DRP = 1: hardware handles everything
  - CTL_BCK = 0, CTL_DRP = 0: software sees all pause frames
  - CTL_BCK = 0, CTL_DRP = 1: all pause frames are completely ignored

    These control bits should be set to CTL_BCK = 0,CTL_DRP = 0 in half-duplex mode. Since PAUSE packets only apply to full duplex operation, any PAUSE packet would constitute an exception which should be handled by the processing cores. PAUSE packets should not be forwarded.

Frame Check Register AGL_GMX_RX(0)_FRM_CHK

This register indicates which frame errors will set the ERR bit of the frame. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved
<9>	NIBERR	R/W	1	1	Nibble error
<8>	SKPERR	R/W	1	1	Skipper error.
<7>	RCVERR	R/W	1	1	Frame was received with packet data reception error.
<6>	LENERR	R/W	0	0	Frame was received with length error. If AGL_GMX_RX(0)_UDD_SKP[LEN] ≠ 0, then LENERR is forced to zero in hardware. Most systems should use the checking in PKI/PIP for checking the L2 length field.
<5>	ALNERR	R/W	1	1	Frame was received with an alignment error.
<4>	FCSERR	R/W	1	1	Frame was received with FCS/CRC error.
<3>	JABBER	R/W	1	1	Frame was received with length > sys_length.
<2>	MAXERR	R/W	1	1	Frame was received with length > max_length.
<1>	CAREXT	R/W	1	1	Carrier extend error
<0>	MINERR	R/W	1	1	Frame was received with length < min_length.

Frame Minimum-Length Register AGL_GMX_RX(0)_FRM_MIN

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	LEN	R/W	0x40	0x40	Byte count for Min-sized frame check AGL_GMX_RXn_FRM_CHK[MINERR] enables the check for port *n*. If enabled, failing packets set the MINERR interrupt and the WQE opcode is set to UNDER_FCS (0x6, if packet has bad FCS) or UNDER_ERR (0x8, if packet has good FCS).

Cavium Confidential For Bowen Yang ALI 02/27/2014

Frame Maximum-Length Register AGL_GMX_RX(0)_FRM_MAX

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	LEN	R/W	0x600	0x600	Byte count for max-sized frame check AGL_GMX_RXn_FRM_CHK[MAXERR] enables the check for port *n*. If enabled, failing packets set the MAXERR interrupt and the WQE opcode is set to OVER_FCS (0x3, if packet has bad FCS) or OVER_ERR (0x4, if packet has good FCS). LEN <= AGL_GMX_RX_JABBER[CNT]. When changing the LEN field, be sure that LEN does not exceed AGL_GMX_RX(0)_JABBER[CNT]. Failure to meet this constraint will cause packets that are within the maximum length parameter to be rejected because they exceed the AGL_GMX_RX(0)_JABBER[CNT] limit.

Maximum Packet-Size Register AGL_GMX_RX(0)_JABBER

This register specifies the maximum size for packets, beyond which the GMX truncates. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	CNT	R/W	0x2800	0x2800	Byte count for jabber check. Failing packets set the JABBER interrupt and are optionally sent with opcode = JABBER. GMX truncates the packet to CNT bytes. CNT must be 8-byte aligned such that CNT[2:0] = 000. Be sure the CNT field value is at least as large as the AGL_GMX_RX(0)_FRM_MAX[LEN] value. Failure to meet this constraint will cause packets that are within the AGL_GMX_RX(0)_FRM_MAX[LEN] length to be rejected because they exceed the CNT limit.

Note:

The packet that is sent to the packet-input logic will have an additional eight bytes if AGL_GMX_RX(0)_FRM_CTL[PRE_CHK] = 1 and AGL_GMX_RX(0)_FRM_CTL[PRE_STRP] = 0. The maximum packet that will be sent is defined as:

```
max_sized_packet =
AGL_GMX_RX_JABBER[CNT]+
((AGL_GMX_RX_FRM_CTL[PRE_CHK] & !AGL_GMX_RX_FRM_CTL[PRE_STRP]) × 8)
```

Cavium Confidential For ALC Howard Yang 02/27/2014

Packet Decision Register AGL_GMX_RX(0)_DECISION

This register specifies the byte count used to determine when to accept or to filter a packet. As each byte in a packet is received by GMX, the L2 byte count (i.e. the number of bytes from the beginning of the L2 header (DMAC)) is compared against CNT. In normal operation, the L2 header begins after the PREAMBLE + SFD (AGL_GMX_RX(0)_FRM_CTL[PRE_CHK] = 1) and any optional UDD skip data (AGL_GMX_RX(0)_UDD_SKP[LEN]). See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved
<4:0>	CNT	R/W	0x18	0x18	The byte count used to decide when to accept or filter a packet. Refer to Table 25-7.

Note:

When AGL_GMX_RX(0)_FRM_CTL[PRE_CHK] = 0, PREAMBLE + SFD are prepended to the packet and require UDD skip length to account for them.

Table 25-7 GMX Decisions

Port Mode	L2 Size[1]	
	≤AGL_GMX_RX_DECISION bytes (default = 0x18)	>AGL_GMX_RX_DECISION bytes (default = 0x18)
Full Duplex	Accept packet No filtering is applied	Apply filters Accept packet based on DMAC and PAUSE packet filters
Half Duplex	Drop packet Packet is unconditionally dropped	Apply filters Accept packet based on DMAC

1. where L2_size = MAX(0, total_packet_size – AGL_GMX_RX(0)_UDD_SKP[LEN] – ((AGL_GMX_RX(0)_FRM_CTL[PRE_CHK] = 1) × 8)

User-Defined Data Skip Register AGL_GMX_RX(0)_UDD_SKP

This register specifies the amount of user-defined data (UDD) added before the start of the L2C data. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8>	FCSSEL	R/W	0	0	Include the skip bytes in the FCS calculation 0 = all skip bytes are included in FCS 1 = the skip bytes are not included in FCS The skip bytes are part of the packet and are sent through the IOI packet interface and are handled by PKI. The system can determine if the UDD bytes are included in the FCS check by using the FCSSEL field, if the FCS check is enabled.
<7>	—	RAZ	—	—	Reserved
<6:0>	LEN	R/W	0x0	0x0	Amount of user-defined data before the start of the L2C data, in bytes. 0x0 means L2C comes first; maximum value is 0x40. LEN must be 0x0 in half-duplex operation.

Notes:

Assume that the PREAMBLE/SFD is always at the start of the frame, even before UDD bytes. In most cases, there will be no preamble in these cases, since it will be MAC-to-MAC communication without a PHY involved.

Cavium Confidential For Bowen Yang ALU 9/27/2014

In all cases, the UDD bytes are sent through the packet interface as part of the packet. The UDD bytes are never stripped from the actual packet.

If LEN ≠ 0 (i.e. there is some UDD), then length checking should be disabled (i.e. AGL_GMX_RX(0)_FRM_CHK[LENERR] should be cleared and AGL_GMX_RX(0)_INT_REG[LENERR] set to 0.

Address filtering and control-packet filtering can be performed as needed.

RX Statistics Control Register AGL_GMX_RX(0)_STATS_CTL

This register controls the receive statistics registers. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	RD_CLR	R/W	0	0	Clear on read. When this bit is set, any receive statistics registers are cleared when read.

RX Minimum Interframe-Gap Cycles Register AGL_GMX_RX(0)_IFG

This register specifies the minimum number of interframe-gap (IFG) cycles between packets. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3:0>	IFG	R/W	0x8	0x8	Min IFG (in IFG × 8 bits) between packets used to determine IFGERR. Normally IFG is 96 bits. Note in some operating modes, IFG cycles can be inserted or removed in order to achieve clock rate adaptation. For these reasons, the default value is slightly conservative and does not check up to the full 96 bits of IFG.

RGMII InBand Link Status Register AGL_GMX_RX(0)_RX_INBND

These fields are only valid if the attached PHY supports the optional in-band status (see Section 3.4.1 of the RGMII specification, version 2.1 for more information). See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	DUPLEX	RO	0	—	RGMII inbound LinkDuplex. 0 = half-duplex, 1 = full-duplex.
<2:1>	SPEED	RO	0x0	—	RGMII inbound LinkSpeed. 00 = 2.5 MHz 01 = 25 MHz 10 = 125 MHz 11 = Reserved
<0>	STATUS	RO	0	—	RGMII inbound LinkStatus. 0 = down, 1 = up.

Cavium Confidential For Bowen Yang ALU 02/27/2014

RX Pause-Packet Drop Time Registers
AGL_GMX_RX(0)_PAUSE_DROP_TIME

This register extracts the [TIME] field in a PAUSE packet that was dropped due to an AGL GMX RX FIFO full condition. This register is reset when MIX(0)_CTL[RESET] is set to 1. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	STATUS	R/W1C/H	0x0	—	Time extracted from the dropped PAUSE packet.

RX Good Packets Register
AGL_GMX_RX(0)_STATS_PKTS

This register provides a count of good received packets. The following are examples of "not good" packets:

- packets recognized as PAUSE packets
- packets dropped due to the DMAC filter
- packets dropped due to FIFO full status
- packets that have opcodes ≠ 0 (FCS, Length, etc.).

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of received good packets.

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_RX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

RX Good Packets Octet Register
AGL_GMX_RX(0)_STATS_OCTS

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	CNT	RC/W/H	0x0	—	Octet count of received good packets.

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_RX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/27/2014

RX Pause Packets Register
AGL_GMX_RX(0)_STATS_PKTS_CTL

This register provides a count of all packets received that were recognized as flow-control or PAUSE packets. Note that PAUSE packets with any kind of error are also counted in AGL_GMX_RX(0)_STATS_PKTS_BAD.

- Pause packets can be optionally dropped or forwarded based on AGL_GMX_RX(0)_FRM_CTL[CTL_DRP]. This count increments regardless of whether the packet is dropped.
- Pause packets are never counted in AGL_GMX_RX(0)_STATS_PKTS.
- Packets dropped due to the DMAC filter are counted in AGL_GMX_RX(0)_STATS_PKTS_DMAC and not here.

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of received flow-control or PAUSE packets.

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_RX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

RX Pause Packets Octet Register
AGL_GMX_RX(0)_STATS_OCTS_CTL

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	CNT	RC/W/H	0x0	—	Octet count of received flow-control or PAUSE packets.

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_RX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

RX DMAC Packets Register
AGL_GMX_RX(0)_STATS_PKTS_DMAC

This register provides a count of all packets received that were dropped by the DMAC filter. Packets that match the DMAC are dropped and counted here regardless of whether they were bad packets. These packets are never counted in AGL_GMX_RX(0)_STATS_PKTS.

Some packets that are not able to satisfy the DECISION_CNT may not actually be dropped by CN71XX, but they will be counted here as if they were dropped. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of filtered DMAC packets.

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_RX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 12/27/2014

RX DMAC Packets Octet Register
AGL_GMX_RX(0)_STATS_OCTS_DMAC

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	CNT	RC/W/H	0x0	—	Octet count of filtered DMAC packets.

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_RX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

RX Overflow Packets Register
AGL_GMX_RX(0)_STATS_PKTS_DRP

This register provides a count of all packets received that were dropped due to a full receive FIFO. It counts all packets dropped by the FIFO, both good and bad packets received. It does not count packets dropped by the DMAC or PAUSE-packet filters. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of dropped packets.

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_RX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

RX Overflow Packets Octet Register
AGL_GMX_RX(0)_STATS_OCTS_DRP

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	CNT	RC/W/H	0x0	—	Octet count of dropped packets.

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_RX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

RX Bad Packets Register
AGL_GMX_RX(0)_STATS_PKTS_BAD

This register provides a count of all packets received with some error that were not dropped either due to the DMAC filter or lack of room in the receive FIFO. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	CNT	RC/W/H	0x0	—	Count of received packets with opcode ≠ 0

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_RX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Address-Filtering Control Register AGL_GMX_RX(0)_ADR_CTL

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	CAM_MODE	R/W	0x0	—	Allow or deny DMAC address filter 0 = reject the packet on DMAC address match 1 = accept the packet on DMAC address match
<2:1>	MCST	R/W	0x0	—	Multicast Mode 0 = Use the address-filter CAM 1 = Force reject all multicast packets 2 = Force accept all multicast packets 3 = Reserved
<0>	BCST	R/W	1	—	Accept all broadcast packets

Algorithm:

Following is some pseudo-code that represents the address-filter behavior.

```
bool dmac_addr_filter(uint8 prt, uint48 dmac) {
  ASSERT(prt >= 0 && prt <= 3);
  if (is_bcst(dmac))                                  // broadcast accept
    return (AGL_GMX_RX{prt}_ADR_CTL[BCST] ? ACCEPT : REJECT);
  if (is_mcst(dmac) & AGL_GMX_RX{prt}_ADR_CTL[MCST] == 1)   // multicast reject
    return REJECT;
  if (is_mcst(dmac) & AGL_GMX_RX{prt}_ADR_CTL[MCST] == 2)   // multicast accept
    return ACCEPT;

  cam_hit = 0;

  for (i=0; i<8; i++) {
    if (AGL_GMX_RX{prt}_ADR_CAM_EN[EN<i>] == 0)
      continue;
    uint48 unswizzled_mac_adr = 0x0;
    for (j=5; j>=0; j--) {
       unswizzled_mac_adr = (unswizzled_mac_adr << 8) |
                                    AGL_GMX_RX{prt}_ADR_CAM{j}[ADR<i*8+7:i*8>];
    }
    if (unswizzled_mac_adr == dmac) {
      cam_hit = 1;
      break;
    }
  }

  if (cam_hit)
    return (AGL_GMX_RX{prt}_ADR_CTL[CAM_MODE] ? ACCEPT : REJECT);
  else
    return (AGL_GMX_RX{prt}_ADR_CTL[CAM_MODE] ? REJECT : ACCEPT);
}
```

Address-Filtering Control Enable Register AGL_GMX_RX(0)_ADR_CAM_EN

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	EN	R/W	0x0	—	CAM-entry enable bits.

Cavium Confidential For Bowen Yang ALU 8/27/2014

Address-Filtering CAM Control Registers
AGL_GMX_RX(0)_ADR_CAM(0..5)

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	ADR	R/W	0x0	—	The DMAC address to match on. Each entry contributes eight bits to one of eight matchers.The CAM matches against unicast or multicast DMAC addresses.

RGMII TX-Clock-Generation Register
AGL_GMX_TX(0)_CLK

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	CLK_CNT	R/W	0x1	0x1	Controls the RGMII TXC frequency. TXC(period) = rgm_ref_clk(period) × CLK_CNT. Normal Programming Values: 1. RGMII, 1000 Mb/s (AGL_GMX_PRT_CFG[SPEED] is 1), CLK_CNT is 1. 2. RGMII, 10/100 Mb/s (AGL_GMX_PRT_CFG[SPEED] is 0), CLK_CNT is 50/5/ RGMII Example: Given a 125MHz PLL reference clock: CLK_CNT == 1 → 125.0 MHz TXC clock period (8 ns × 1) CLK_CNT == 5 → 25.0MHz TXC clock period (8 ns × 5) CLK_CNT == 50 → 2.5MHz TXC clock period (8 ns × 50)

Packet TX Threshold Register
AGL_GMX_TX(0)_THRESH

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	CNT	R/W	0x20	0x20	Number of 16-byte ticks to accumulate in the TX FIFO before sending on the packet interface. This register should be large enough to prevent underflow on the packet interface and must never be set below 4. This register cannot exceed the the TX FIFO depth which is 128 eight-byte entries (1KB).

Cavium Confidential For Bowen Yang ALU 02/27/2014

Packet TX Append Control Register AGL_GMX_TX(0)_APPEND

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved
<3>	FORCE_FCS	R/W	1	1	Append the Ethernet FCS on each PAUSE packet when FCS = 0. Pause packets are normally padded to 60 bytes. If AGL_GMX_TX(0)_MIN_PKT[MIN_SIZE] exceeds 59, then FORCE_FCS is not used.
<2>	FCS	R/W	1	1	Append the Ethernet FCS on each packet.
<1>	PAD	R/W	1	1	Append PAD bytes such that a minimum-sized packet is transmitted.
<0>	PREAMBLE	R/W	1	1	Prepend the Ethernet preamble on each transfer.

Packet SMAC Register AGL_GMX_SMAC(0)

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	SMAC	R/W	0x0	—	The SMAC field is used for generating and accepting control-pause packets.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Packet TX Pause Packet Pause-Time Register AGL_GMX_TX(0)_PAUSE_PKT_TIME

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TIME	R/W	0x60	—	Provides the pause_time field placed is outbound PAUSE packets, in 512 bit-times. Normally, TIME > AGL_GMX_TX(0)_PAUSE_PKT_INTERVAL[INTERVAL].

Notes:

Choosing proper values of AGL_GMX_TX(0)_PAUSE_PKT_TIME[TIME] and AGL_GMX_TX(0)_PAUSE_PKT_INTERVAL[INTERVAL] can be challenging. Cavium suggests that TIME be much greater than INTERVAL and AGL_GMX_TX(0)_PAUSE_ZERO[SEND] be set to 1. This allows a periodic refresh of the PAUSE count and then when the backpressure condition is lifted, a PAUSE packet with TIME = 0 is sent indicating that CN71XX is ready for additional data.

If the system chooses to not set AGL_GMX_TX(0)_PAUSE_ZERO[SEND], then Cavium suggests that TIME and INTERVAL are programmed such that they satisfy the following rule:

INTERVAL ≤ TIME – (largest_pkt_size + IFG + pause_pkt_size)

where:

largest_pkt_size = the largest packet that the system can send (normally 1518 bytes)

IFG = the interframe gap

pause_pkt_size = the size of the PAUSE packet (normally 64 bytes).

Packet TX Minimum-Size-Packet Register AGL_GMX_TX(0)_MIN_PKT

This register specifies the minimum size a packet should be. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	MIN_SIZE	R/W	0x3B	0x3B	Minimum frame size in bytes before the FCS is applied. Padding is only appended when AGL_GMX_TX(0)_APPEND[PAD] for the corresponding packet port is set. Packets are padded to MIN_SIZE+1. The reset value pads to 60 bytes.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Packet TX Pause Packet Transmission Interval Register AGL_GMX_TX(0)_PAUSE_PKT_INTERVAL

This register specifies how often PAUSE packets are sent. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	INTERVAL	R/W	0x10	—	Arbitrate for a PAUSE packet every (INTERVAL × 512) bit-times. Normally, 0 < INTERVAL < AGL_GMX_TX(0)_PAUSE_PKT_TIME[TIME]. INTERVAL = 0 only send a single PAUSE packet for each backpressure event.

Notes:

Choosing proper values of AGL_GMX_TX(0)_PAUSE_PKT_TIME[TIME] and AGL_GMX_TX(0)_PAUSE_PKT_INTERVAL[INTERVAL] can be challenging. Cavium suggests that TIME be much greater than INTERVAL and AGL_GMX_TX(0)_PAUSE_ZERO[SEND] be set to 1. This allows a periodic refresh of the PAUSE count and then when the backpressure condition is lifted, a PAUSE packet with TIME = 0 is sent indicating that CN71XX is ready for additional data.

If the system chooses to not set AGL_GMX_TX(0)_PAUSE_ZERO[SEND], then Cavium suggests that TIME and INTERVAL are programmed such that they satisfy the following rule:

INTERVAL ≤ TIME – (largest_pkt_size + IFG + pause_pkt_size)

where:

largest_pkt_size = the largest packet that the system can send (normally 1518 bytes)

IFG = the interframe gap

pause_pkt_size = the size of the PAUSE packet (normally 64 bytes).

Packet TX Software Pause Register AGL_GMX_TX(0)_SOFT_PAUSE

This register specifies the software pause time. This register is reset when MIX(0)_CTL[RESET] is set to 1. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TIME	R/W	0x0	—	Back off the TX bus for (TIME × 512) bit-times. For full-duplex operation only.

Packet TX Time-to-Backpressure Registers AGL_GMX_TX(0)_PAUSE_TOGO

This register is the pause-deferral counter. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TIME	RO/H	—	—	Amount of time remaining to backpressure.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Packet TX Pause-Zero-Enable Registers
AGL_GMX_TX(0)_PAUSE_ZERO

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	SEND	R/W	1	1	Send pause-zero enable. When this bit is set, and the backpressure condition is clear, it allows sending a pause packet with pause_time of 0 to enable the channel.

TX Statistics Control Register
AGL_GMX_TX(0)_STATS_CTL

This register controls the transmit statistics registers. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	RD_CLR	R/W	0	0	Clear on read. When this bit is set, any transmit statistics registers are cleared when read.

Transmit Control Register
AGL_GMX_TX(0)_CTL

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	XSDEF_EN	R/W	1	1	Enables the excessive-deferral check for statistics and interrupts.
<0>	XSCOL_EN	R/W	1	1	Enables the excessive-collision check for statistics and interrupts.

Transmit Statistics Registers 0
AGL_GMX_TX(0)_STAT0

This register maintains statistics on excessive deferrals and excessive collisions. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	XSDEF	RC/W/H	0x0	—	Number of packets dropped (never successfully sent) due to excessive deferrals.
<31:0>	XSCOL	RC/W/H	0x0	—	Number of packets dropped (never successfully sent) due to excessive collision. Defined by AGL_GMX_TX_COL_ATTEMPT[LIMIT].

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_TX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Transmit Statistics Register 1
AGL_GMX_TX(0)_STAT1

This register maintains statistics on single collisions and multiple collisions. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	SCOL	RC/W/H	0x0	—	Number of packets sent with a single collision.
<31:0>	MCOL	RC/W/H	0x0	—	Number of packets sent with multiple collisions but less than AGL_GMX_TX_COL_ATTEMPT[LIMIT].

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_TX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Register 2
AGL_GMX_TX(0)_STAT2

This register maintains statistics on the number of octets sent on the interface. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	OCTS	RC/W/H	0x0	—	Number of total octets sent on the interface. Does not count octets from frames that were truncated due to collisions in half-duplex mode.

NOTE:
- Octet counts are the sum of all data transmitted on the wire, including packet data, pad bytes, FCS bytes, PAUSE bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when AGL_GMX_TX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Register 3
AGL_GMX_TX(0)_STAT3

Used for AGL_GMX_RX(0)_STATS_PKTS.

This register maintains statistics on the number of frames sent on the interface. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	PKTS	RC/W/H	0x0	—	Number of total frames sent on the interface. Does not count frames that were truncated due to collisions in half-duplex mode.

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_TX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Transmit Statistics Register 4
AGL_GMX_TX(0)_STAT4

This register maintains statistics on the number of packets sent on the interface with octet counts of 64 and less than 64. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	HIST1	RC/W/H	0x0	—	Number of packets sent with an octet count of 64.
<31:0>	HIST0	RC/W/H	0x0	—	Number of packets sent with an octet count < 64.

NOTE:
- Packet length is the sum of all data transmitted on the wire for the given packet, including packet data, pad bytes, FCS bytes, PAUSE bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when AGL_GMX_TX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Register 5
AGL_GMX_TX(0)_STAT5

This register maintains statistics on the number of packets sent on the interface with octet counts between 65 and 127 and between 128 and 255. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	HIST3	RC/W/H	0x0	—	Number of packets sent with an octet count of 128–255.
<31:0>	HIST2	RC/W/H	0x0	—	Number of packets sent with an octet count of 65–127.

NOTE:
- Packet length is the sum of all data transmitted on the wire for the given packet, including packet data, pad bytes, FCS bytes, PAUSE bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when AGL_GMX_TX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Register 6
AGL_GMX_TX(0)_STAT6

This register maintains statistics on the number of packets sent on the interface with octet counts between 256 and 511 and between 512 and 1023. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	HIST5	RC/W/H	0x0	—	Number of packets sent with an octet count of 512–1023.
<31:0>	HIST4	RC/W/H	0x0	—	Number of packets sent with an octet count of 256–511.

NOTE:
- Packet length is the sum of all data transmitted on the wire for the given packet, including packet data, pad bytes, FCS bytes, PAUSE bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when AGL_GMX_TX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Transmit Statistics Register 7
AGL_GMX_TX(0)_STAT7

This register maintains statistics on the number of packets sent on the interface with octet counts between 1024 and 1518 and greater than 1518. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	HIST7	RC/W/H	0x0	—	Number of packets sent with an octet count of > 1518.
<31:0>	HIST6	RC/W/H	0x0	—	Number of packets sent with an octet count of 1024–1518.

NOTE:
- Packet length is the sum of all data transmitted on the wire for the given packet, including packet data, pad bytes, FCS bytes, PAUSE bytes, and JAM bytes. The octet counts do not include the preamble byte or extend cycles.
- Cleared by either a write (of any value) or a read when AGL_GMX_TX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Transmit Statistics Register 8
AGL_GMX_TX(0)_STAT8

This register maintains statistics on the number of packets sent to broadcast DMAC and to multicast DMAC. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	MCST	RC/W/H	0x0	—	Number of packets sent to multicast DMAC. Does not include BCST packets.
<31:0>	BCST	RC/W/H	0x0	—	Number of packets sent to broadcast DMAC. Does not include MCST packets.

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_TX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.
- Note that the GMX determines if the packet is MCST or BCST from the DMAC of the packet. GMX assumes that the DMAC lies in the first six bytes of the packet as per the 802.3 frame definition. If the system requires additional data before the L2 header, then the MCST and BCST counters may not reflect reality and should be ignored by software.

Transmit Statistics Registers 9
AGL_GMX_TX(0)_STAT9

This register maintains statistics on the number of underflow packets and control packets. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	UNDFLW	RC/W/H	0x0	—	Number of underflow packets.
<31:0>	CTL	RC/W/H	0x0	—	Number of Control packets (PAUSE flow control) generated by GMX. It does not include control packets forwarded or generated by the cores.

NOTE:
- Cleared by either a write (of any value) or a read when AGL_GMX_TX(0)_STATS_CTL[RD_CLR] is set.
- Counters will wrap.

Cavium Confidential For Bowen Yang ALU 02/27/2014

AGL GMX BIST Results Register
AGL_GMX_BIST

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved
<24:0>	STATUS	RO	0x0	0x0	BIST results. 0 = pass, 1 = fail <0> = gmx#.inb.fif_bnk0 <1> = gmx#.inb.fif_bnk1 <2> = gmx#.inb.fif_bnk2 <3> = gmx#.inb.fif_bnk3 <4> = gmx#.inb.fif_bnk_ext0 <5> = gmx#.inb.fif_bnk_ext1 <6> = gmx#.inb.fif_bnk_ext2 <7> = gmx#.inb.fif_bnk_ext3 <8 >= gmx#.outb.fif.fif_bnk0 <9> = gmx#.outb.fif.fif_bnk1 <10, 11> = RAZ <12> = gmx#.outb.fif.fif_bnk_ext0 <13> = gmx#.outb.fif.fif_bnk_ext1 <14–19> = RAZ <20> = gmx#.csr.drf20x32m2_bist <21> = gmx#.csr.drf20x48m2_bist <22> = gmx#.outb.stat.drf16x27m1_bist <23> = gmx#.outb.stat.drf40x64m1_bist <24> = RAZ

RX FIFO Packet-Drop Register
AGL_GMX_RX_BP_DROP(0)

This register provides the mark for an RX FIFO beyond which packets are dropped and not buffered. The relationship of the values in AGL_GMX_RX_BP_DROP(0), AGL_GMX_RX_BP_ON(0), and AGL_GMX_RX_BP_OFF(0) is shown Figure 25–4. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	MARK	R/W	0x2	0x2	Number of eight-byte ticks to reserve in the RX FIFO. When the FIFO exceeds this count, packets are be dropped and not buffered. This field should typically be programmed to 2. Failure to program correctly can lead to system instability.

RX Backpressure On Register
AGL_GMX_RX_BP_ON(0)

This register provides the high watermark for port/interface backpressure, the FIFO-full level at which backpressure is asserted. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8:0>	MARK	R/W	0x40	0x40	High watermark (in eight-byte ticks) for asserting backpressure.

RX Backpressure Off Register
AGL_GMX_RX_BP_OFF(0)

This register provides the low watermark for port/interface backpressure, the FIFO-full level at which backpressure can be deasserted. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5:0>	MARK	R/W	0x10	0x10	Low watermark (in eight-byte ticks) to deassert backpressure.

Cavium Confidential For Bowen Yang ALU 02/11/2014

Packet TX Interframe Gap Register
AGL_GMX_TX_IFG

Consider the following when programming IFG1 and IFG2:

- For half-duplex systems that require IEEE 802.3 compatibility, IFG1 must be in the range of 1–8, IFG2 must be in the range of 4–12, and the IFG1 + IFG2 sum must be 12.
- For full-duplex systems that require IEEE 802.3 compatibility, IFG1 must be in the range of 1–11, IFG2 must be in the range of 1–11, and the IFG1 + IFG2 sum must be 12.
- For all other systems, IFG1 and IFG2 can be any value in the range of 1–15, allowing for a total possible IFG sum of 2–30.

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:4>	IFG2	R/W	0x4	—	1/3 of the interFrameGap timing. If CRS is detected during IFG2, the interFrameSpacing timer is not reset and a frame is transmitted once the timer expires. 0x4 = 1000 Mb/s 0xA = 10/100 Mb/s
<3:0>	IFG1	R/W	0x8	—	2/3 of the interFrameGap timing. If CRS is detected during IFG1, the interFrameSpacing timer is reset and a frame is not transmitted. 0x8 = 1000 Mb/s 0xE = 10/100 Mb/s

Packet TX JAM Pattern Register
AGL_GMX_TX_JAM

This register provides the pattern used in JAM bytes. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	JAM	R/W	0xEE	—	JAM pattern.

Packet TX Collision Attempts Before Dropping Frame Register
AGL_GMX_TX_COL_ATTEMPT

This register provides the number of collision attempts allowed before dropping the frame. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved
<4:0>	LIMIT	R/W	0x10	0x10	Number of collision attempts allowed.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Packet TX Pause Packet DMAC Field Register AGL_GMX_TX_PAUSE_PKT_DMAC

This register provides the DMAC value that is placed in outbound PAUSE packets. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved
<47:0>	DMAC	R/W	0x0180C2000001	0x0180C2000001	The DMAC field, which is placed is outbound PAUSE packets.

Packet TX Pause Packet TYPE Field Register AGL_GMX_TX_PAUSE_PKT_TYPE

This register provides the Type field that is placed in outbound PAUSE packets. TSee Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TYPE	R/W	0x8808	0x8808	The Type field placed is outbound pause packets.

Packet TX Override Backpressure Register AGL_GMX_TX_OVR_BP

This register provides the ability to assert per-port backpressure, as well as the ability to override the assertion of backpressure. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:9>	—	RAZ	—	—	Reserved
<8>	EN	R/W	0	0	Per-port backpressure enable/override
<7:5>	—	RAZ	—	—	Reserved
<4>	BP	R/W	0	0	Per-port backpressure status to use: 0 = port is available 1 = port should be backpressured
<3:1>	—	RAZ	—	—	Reserved
<0>	IGN_FULL	R/W	0	0	Ignore the RX FIFO full when computing backpressure.

Packet TX Backpressure Status Register AGL_GMX_TX_BP

This register shows the status of backpressure. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	BP	RO	0	0	Port backpressure status: 0 = port is available, 1 = port should be backpressured

Cavium Confidential For Bowen Yang ALU 8/27/2014

RX Port State Information Register
AGL_GMX_RX_PRT_INFO

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:17>	—	RAZ	—	—	Reserved
<16>	DROP	RO/H	0	0	Port indication that data was dropped.
<15:1>	—	RAZ	—	—	Reserved
<0>	COMMIT	RO/H	0	0	Port indication that SOP was accepted.

TX LFSR Register
AGL_GMX_TX_LFSR

This register shows the contents of the linear feedback shift register (LFSR), which is used to implement truncated binary exponential backoff. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	LFSR	R/W	0xFFFF	—	Contains the current state of the LFSR, which is used to feed random numbers to compute truncated binary exponential backoff.

TX Interrupt Register
AGL_GMX_TX_INT_REG

This register allows per-port interrupts to be sent to the control processor. If an error is noted in this register, and the corresponding bit in AGL_GMX_TX_INT_EN is set, an exception is raised. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:21>	—	RAZ	—	—	Reserved
<20>	PTP_LOST	R/W1C/H	0	0	A packet with a PTP request could not be sent due to XSCOL.
<19:17>	—	RAZ	—	—	Reserved
<16>	LATE_COL	R/W1C/H	0	0	TX late collision.
<15:13>	—	RAZ	—	—	Reserved
<12>	XSDEF	R/W1C/H	0	0	TX excessive deferrals (half-duplex mode only)
<11:9>	—	RAZ	—	—	Reserved
<8>	XSCOL	R/W1C/H	0	0	TX excessive collisions (half-duplex mode only)
<7:3>	—	RAZ	—	—	Reserved
<2>	UNDFLW	R/W1C/H	0	0	TX underflow
<1>	—	RAZ	—	—	Reserved
<0>	PKO_NXA	R/W1C/H	0	0	Port address out-of-range from PKO interface.

Cavium Confidential For Bowen Yang ALU 02/27/2014

TX Interrupt-Enable Register
AGL_GMX_TX_INT_EN

This register provides the interrupt-enable bits for the corresponding error in AGL_GMX_TX_INT_REG. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:21>	—	RAZ	—	—	Reserved
<20>	PTP_LOST	R/W	0	0	A packet with a PTP request was not able to be sent due to XSCOL.
<19:17>	—	RAZ	—	—	Reserved
<16>	LATE_COL	R/W	0	0	TX late collision.
<15:13>	—	RAZ	—	—	Reserved
<12>	XSDEF	R/W	0	0	TX excessive deferrals (half-duplex mode only)
<11:9>	—	RAZ	—	—	Reserved
<8>	XSCOL	R/W	0	0	TX excessive collisions (half-duplex mode only)
<7:3>	—	RAZ	—	—	Reserved
<2>	UNDFLW	R/W	0	0	TX Underflow
<1>	—	RAZ	—	—	Reserved
<0>	PKO_NXA	R/W	0	0	Port address out-of-range from PKO Interface

Miscellaneous Error Register
AGL_GMX_BAD_REG

This register contains a variety of error conditions that were not covered in other error registers. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:35>	—	RAZ	—	—	Reserved
<34>	TXPSH	R/W1C	0	0	TX FIFO overflow.
<33>	TXPOP	R/W1C	0	0	TX FIFO underflow.
<32>	OVRFLW	R/W1C	0	0	RX FIFO overflow.
<31:27>	—	RAZ	—	—	Reserved
<26>	STATOVR	R/W1C/H	0	0	TX statistics overflow.
<25:23>	—	RAZ	—	—	Reserved
<22>	LOSTSTAT	R/W1C/H	0	0	TX statistics data was overwritten. TX statistics registers are corrupted.
<21:3>	—	RAZ	0x0	0x0	Reserved
<2>	OUT_OVR	R/W1C/H	0	0	Outbound data FIFO overflow.
<1:0>	—	RAZ	0x0	0x0	Reserved

Cavium Confidential For Bowen Yang ALU 02/27/2014

Backpressure Statistics Register
AGL_GMX_STAT_BP

This register indicates the number of cycles that the TX/Stats block has held up operation. See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:17>	—	RAZ	—	—	Reserved
<16>	BP	RO/H	0	0	Current TX stats backpressure state. When the TX stats machine cannot update the stats registers quickly enough, the machine has the ability to backpressure the TX datapath. This is a rare event and will not occur in normal operation. 0 = no backpressure is applied. 1 = backpressure is applied to TX datapath to allow stat update operations to complete.
<15:0>	CNT	R/W1C/H	0x0	0x0	Saturating counter. Number of cycles that BP has been asserted.

Wake-on-LAN Control Register
AGL_GMX_WOL_CTL

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:33>	—	RAZ	—	—	Reserved.
<32>	MAGIC_EN	R/W	0	—	Port enable for magic packet WOL event detection.
<31:17>	—	RAZ	—	—	Reserved.
<16>	DIRECT_EN	R/W	0	—	Port enable for directed packet WOL event detection.
<15:1>	—	RAZ	—	—	Reserved.
<0>	EN	R/W	0	—	Enter WOL mode.

GMX RX/TX Status Register
AGL_GMX_RX_TX_STATUS

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved
<4>	TX	RC/H	0	0	Transmit data since last read.
<3:1>	—	RAZ	—	—	Reserved
<0>	RX	RC/H	0	0	Receive data since last read.

Cavium Confidential For Bowen Yang ALU 02/27/2014

AGL Port Control Register
AGL_PRT(0)_CTL

The RGMII timing specification requires that devices transmit clock and data synchronously. The specification requires external sources (namely the PC board trace routes) to introduce the appropriate 1.5 to 2.0 ns of delay.

To eliminate the need for the PC board delays, the RGMII interface has optional onboard DLLs for both transmit and receive. For correct operation, at most one of the transmitter, board, or receiver involved in an RGMII link should introduce delay. By default/reset, the RGMII receivers delay the received clock, and the RGMII transmitters do not delay the transmitted clock. Whether or not this default works as-is with a given link partner depends on the behavior of the link partner and the PC board.

These are the possible modes of RGMII receive operation:

- AGL_PRT(0)_CTL[CLKRX_BYP] = 0 (reset value) – The CN71XX RGMII receive interface introduces clock delay using its internal DLL. This mode is appropriate if neither the remote transmitter nor the PC board delays the clock.
- AGL_PRT(0)_CTL[CLKRX_BYP] = 1, [CLKRX_SET] = 0x0 – The CN71XX RGMII receive interface introduces no clock delay. This mode is appropriate if either the remote transmitter or the PC board delays the clock.

These are the possible modes of RGMII transmit operation:

- AGL_PRT(0)_CTL[CLKTX_BYP] = 1, [CLKTX_SET] = 0x0 (reset value) – The CN71XX RGMII transmit interface introduces no clock delay. This mode is appropriate if either the remote receiver or the PC board delays the clock.
- AGL_PRT(0)_CTL[CLKTX_BYP] = 0 – The CN71XX RGMII transmit interface introduces clock delay using its internal DLL. This mode is appropriate if neither the remote receiver nor the PC board delays the clock.

See Table 25–6 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	DRV_BYP	R/W	0	0	When set, bypass the compensation controller and use DRV_NCTL and DRV_PCTL.
<62:61>	—	RAZ	—	—	Reserved.
<60:56>	CMP_PCTL	RO/H	—	—	PCTL drive strength from the hardware compensation controller.
<55:53>	—	RAZ	—	—	Reserved
<52:48>	CMP_NCTL	RO/H	—	—	NCTL drive strength from the hardware compensation controller.
<47:45>	—	RAZ	—	—	Reserved
<44:40>	DRV_PCTL	R/W	0xD	—	PCTL drive strength to use in bypass mode. Reset value of 0xD is for 50 Ω termination.
<39:37>	—	RAZ	—	—	Reserved
<36:32>	DRV_NCTL	R/W	0xC	—	NCTL drive strength to use in bypass mode. Reset value of 0xC is for 50 Ω termination.
<31>	—	RAZ	—	—	Reserved
<30:24>	CLK_SET	RO/H	—	—	The clock delay as determined by the on board hardware DLL.
<23>	CLKRX_BYP	R/W	0	—	Bypass the RX clock delay setting. Skews RXC from RXD, RXCTL. By default, hardware internally shifts the RXC clock to sample RXD, RXCTL assuming clock and data and sourced synchronously from the link partner.
<22:16>	CLKRX_SET	R/W	0x0	—	RX clock delay setting to use in bypass mode. Skews RXC from RXD.

Cavium Confidential For Bowen Yang ALU 02/17/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<15>	CLKTX_BYP	R/W	1	—	Bypass the TX clock delay setting. Skews TXC from TXD, TXCTL. By default, clock and data and sourced synchronously.
<14:8>	CLKTX_SET	R/W	0x0	—	TX clock delay setting to use in bypass mode. Skews TXC from TXD.
<7:6>	REFCLK_SEL	R/W	0x0	—	Select the reference clock to use. Normal RGMII specification requires a 125MHz oscillator. In order to reduce system cost, a 500MHz coprocessor clock can be divided down and remove the requirements for the external oscillator. Additionally, in some well defined systems, the link partner may be able to source the RXC. The RGMII would operate correctly in 1000Mbs mode only. 0x0 = RGMII REFCLK 0x1 = RGMII RXC (1000Mbs only) 0x2 = divided coprocessor clock 0x3 = reserved
<5>	—	RAZ	—	—	Reserved.
<4>	DLLRST	R/W	1	0	DLL reset.
<3>	COMP	R/W	0	1	Compensation enable.
<2>	ENABLE	R/W	0	1	Port enable.
<1>	CLKRST	R/W	0	1	Clock tree reset.
<0>	MODE	RO	0	0	Port mode. 0 = RGMII.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 26

Serial AT Attachment Interface (SATA)

This chapter contains the following subjects:

- Overview
- Architecture
- Initialization
- Clocks and Reset
- Endian-Swapping
- Interrupt Handler
- Host-Mode Programming Model
- Enumerations
- Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The CN71XX SATA subsystem contains an AHCI-compliant host-bus adapter with two physical SATA ports. Each port has the following features:

- Compliant with the following specifications:
  - Serial ATA 3.1
  - AHCI Revision 1.31
- SATA 1.5 Gb/s, 3.0 Gb/s, and 6.0 Gb/s speeds
  - Supports speed negotiation
- eSATA (up to Gen2m)
- Native support for standard Linux and Windows drivers
- Port-multiplier support with FIS-based switching for up to 15 devices per port
- Power management includes direct partial-to-slumber mode
- Digital support of device hot-plugging
  - Cold presence detect
  - Mechanical presence switch
- Activity LED support

Each port has the following unsupported features and limitations:

- Enclosure management
- Multiple-message MSI
- Non-zero buffer offsets
- NVMHCI and BIOS/OS handoff AHCI features
- Prefetch implemented in a proprietary method, not through the command header P-bit as defined in the AHCI specification
- Protocol-based port selector presence detection
- SATA legacy mode

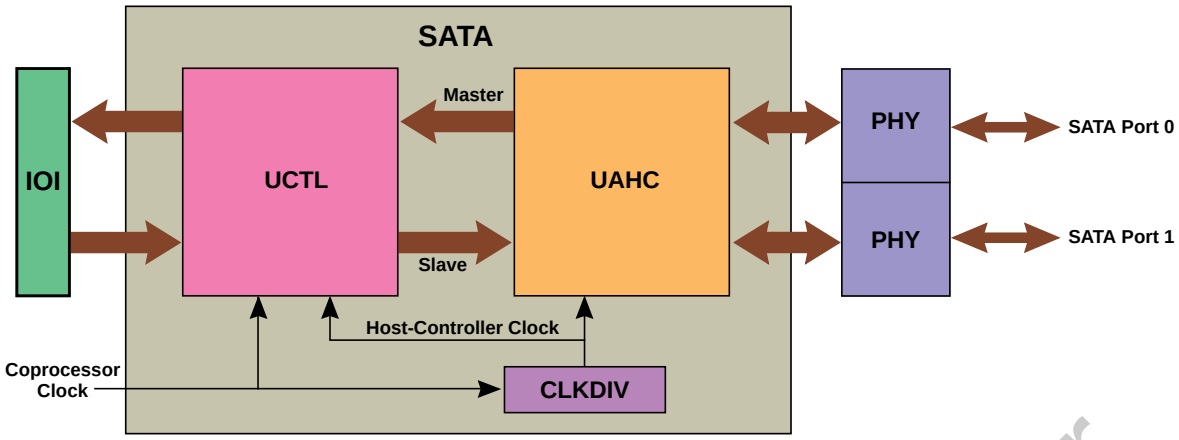
26.1 Architecture

This section describes the general SATA host-controller architecture and its major components, shown in Figure 26–1. There are two major blocks:

- UCTL (SATA controller shim to IOI)
- UAHC (SATA AHCI host controller)

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 26–1 SATA Subsystem Block Diagram



26.1.1 UCTL

UCTL is the bridge unit between the I/O interconnect (IOI) and the SATA AHCI host controller (UAHC). It performs the following functions:

- provides interfaces for the applications to access the UAHC AHCI registers on the CN71XX I/O space.
- provides a bridge for UAHC to fetch AHCI command table entries and data buffers from L2C through I/O inbound bus (IOBI) and outbound bus (IOBO).
- posts interrupts to the CIU.
- contains registers that:
  - control the behavior of the UAHC
  - control the clock/reset generation to UAHC
  - control endian swapping for all UAHC registers and DMA accesses

26.1.2 UAHC

The UAHC implements the link and transport layers of the SATA specification and is the interface to the application layer. It contains two AHCI-compatible host controllers. It supports the following functions:

- port DMA
- 32-entry hardware-assisted native command queuing
- FIS-based port multiplier switching
- OOB signal detection and generation
- 8b/10b encoding/decoding
- command completion coalescing
- mechanical presence switch, cold presence detect, and activity LED support
- port power management

Cavium Confidential For Bowen Yang ALU 02/27/2014

26.2 Initialization

Software-controlled resets in the SATA subsystem are the following:

- SATA_UCTL_CTL[SATA_UCTL_RST] resets the UCTL except for the UCTL CSR offsets [0x0 – 0x2F].
- SATA_UCTL_CTL[SATA_UAHC_RST] resets UAHC.

If the SATA subsystem is not being used, Cavium recommends that software turn off the host-controller clock and write SATA_UCTL_CTL[CSCLK_EN] to 0 to place the SATA logic in the lowest power-state.

26.3 Clocks and Reset

The SATA subsystem includes the following clocks(refer to Figure 26–1):

- CN71XX coprocessor clock (SCLK)

  Most of the coprocessor-clock logic is disabled by clearing SATA_UCTL_CTL[CSCLK_EN] = 0 after reset. For applications that use SATA interfaces, SATA_UCTL_CTL[CSCLK_EN] should be set to 1 to enable SCLK in the SATA subsystem (see step 6 of Section 26.3.1).

- SATA AXI host-controller clock (ACLK)

  This clock is divided down from the coprocessor clock. The divider is controlled by SATA_UCTL_CTL[A_CLK_EN, A_CLKDIV_RST, A_CLKDIV_SEL, A_CLK_BYP_SEL]. The host-controller clock runs up to 400MHz.

Table 26–1 shows the maximum AXI host-controller clock frequencies and the appropriate divisor.

Table 26–1 Maximum ACLK Frequency

Port Speed		Coprocessor-Clock (SCLK) Frequency (MHz)								
Port 0	**Port 1**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**
Gen3	Gen3	300 (div 1)	350 (div 1)	400 (div 1)	225 (div 2)	250 (div 2)	275 (div 2)	300 (div 2)	325 (div 2)	350 (div 2)
Gen3	Gen2	300 (div 1)	175 (div 2)	200 (div 2)	225 (div 2)	167 (div 3)	183 (div 3)	200 (div 3)	162 (div 4)	175 (div 4)
Gen3	Gen1	150 (div 2)	175 (div 2)	200 (div 2)	150 (div 3)	167 (div 3)	137 (div 4)	150 (div 4)	162 (div 4)	175 (div 4)
Gen2	Gen3	300 (div 1)	175 (div 2)	200 (div 2)	225 (div 2)	167 (div 3)	183 (div 3)	200 (div 3)	162 (div 4)	175 (div 4)
Gen2	Gen2	150 (div 2)	117 (div 3)	133 (div 3)	150 (div 3)	125 (div 4)	137 (div 4)	150 (div 4)	108 (div 6)	116 (div 6)
Gen2	Gen1	100 (div 3)	87 (div 4)	100 (div 4)	112 (div 4)	83 (div 6)	91 (div 6)	100 (div 6)	81 (div 8)	87 (div 8)
Gen1	Gen3	150 (div 2)	175 (div 2)	200 (div 2)	150 (div 3)	167 (div 3)	137 (div 4)	150 (div 4)	162 (div 4)	175 (div 4)
Gen1	Gen2	100 (div 3)	87 (div 4)	100 (div 4)	112 (div 4)	83 (div 6)	91 (div 6)	100 (div 6)	81 (div 8)	87 (div 8)
Gen1	Gen1	75 (div 4)	58 (div 6)	66 (div 6)	56 (div 8)	62 (div 8)	68 (div 8)	75 (div 8)	81 (div 8)	87 (div 8)

Cavium Confidential For Bowen Yang 03/27/2015

26.3.1 Cold Reset

1. Ensure that the SerDes reference clock is up and stable.
2. Optionally program the GPIO CSRs for SATA features.
   a. For cold-presence detect:
      i. Select a GPIO for the input and program GPIO_SATA_CTL[SEL] for port0/port1
      ii. select a GPIO for the output and program GPIO_BIT_CFG*[OUTPUT_-SEL] for port0/port1
   b. For mechanical-presence detect, select a GPIO for the input and program GPIO_SATA_CTL[SEL] for port0/port1
   c. For LED activity, select a GPIO for the output and program GPI-O_BIT_CFG*[OUTPUT_SEL] for port0/port1
3. Assert all resets:
   a. UAHC reset: SATA_UCTL_CTL[UAHC_RST] = 1
   b. UCTL reset: SATA_UCTL_CTL[UCTL_RST] = 1
4. Configure the ACLK:
   a. Reset the clock dividers: SATA_UCTL_CTL[A_CLKDIV_RST] = 1.
   b. Select the ACLK frequency (refer to maximum values in Table 26–1).
      i. SATA_UCTL_CTL[A_CLKDIV] = desired value,
      ii. SATA_UCTL_CTL[A_CLKDIV_EN] = 1 to enable the ACLK.
   c. Deassert the ACLK clock divider reset: SATA_UCTL_CTL[A_CLKDIV_RST] = 0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

5. Deassert UCTL and UAHC resets:
   a. SATA_UCTL_CTL[UCTL_RST] = 0
   b. SATA_UCTL_CTL[UAHC_RST] = 0
   c. Wait 10 ACLK cycles before accessing any ACLK-only registers.
6. Enable conditional SCLK of UCTL by writing SATA_UCTL_CTL[CSCLK_EN] = 1.
7. Configure PHY for SATA. Refer to Section 21.1.2.
8. Poll DLM2_MPLL_STATUS for MPLL lock
9. Initialize UAHC as described in the AHCI specification (UAHC_* registers).

If the UAHC registers are accessed over IOI before step 6, the UAHC register interface might hang the IOI.

If the UCTL registers offsets 0x30 – 0xF8 are accessed over RSL before step 6, the UCTL's RSL interface might hang.

26.4 Endian-Swapping

The CN71XX architecture defaults to big-endian. The UAHC defaults to little-endian CSR accesses and DMA accesses.

The UCTL provides a configurable byte-lane swapper on the DMA data path.

The UCTL provides a configurable byte-lane swapper on the IOI data path to the UAHC CSRs (note that the RSL path to the UAHC CSRs always presents the CSR data on the RSL bus as big-endian).

The two swapper configuration registers are SATA_UCTL_SHIM_CFG[CSR_ENDIAN_MODE, DMA_ENDIAN_MODE]. Both fields are implemented to use the same enumeration mapping, and both default to big-endian mode.

26.5 Interrupt Handler

The CIU in CN71XX contains one interrupt group for SATA.

This enumeration contains both the interrupts that are defined by the AHCI specification and the interrupts in the UCTL shim. For more information about interrupts, see Chapter 13.

Cavium Confidential For Bowen Yang ALD 02/27/2014

26.5.1 AHCI Interrupts

There are two interrupts in this group:

- The main AHCI runtime interrupts (SATA_UCTL_UAHC_INTRQ_IP)
- The power-management-event interrupt (SATA_UCTL_UAHC_PME_REQ_IP).

See the AHCI specification for more information.

26.5.2 UCTL Interrupts

These interrupts correspond to the UCTL_INTSTAT vector. The SATA_UCTL interrupt is mainly for fatal errors (protocol violations). For more information, see the interrupt-bit descriptions in Section 26.8.1.

26.6 Host-Mode Programming Model

After hardware initialization (see Section 26.2), SATA is fully compliant with the AHCI specification. Refer to the specification for host-mode programming information.

Cavium Confidential For Bowen Yang ALU 02/27/2014

26.7 Enumerations

SATA UCTL DMA Read Command Enumeration
SATA_UCTL_DMA_READ_CMD_E

Enumerates IOI inbound command selections for DMA read operations.

Value	Mnemonic	Description
0	LDI	Use LDI (allocate).
1	LDT	Use LDT (no allocate).

SATA_UCTL Endian-Mode Enumeration
SATA_UCTL_ENDIAN_MODE_E

Enumerate the endian mode selections.

Value	Mnemonic	Description
0x0	LITTLE	Core is in little-endian mode. A-B-C-D-E-F-G-H becomes A-B-C-D-E-F-G-H. This is the mode to use with open-source AHCI drivers when the core is in little-endian mode.
0x1	BIG	Core is in big-endian mode. A-B-C-D-E-F-G-H becomes H-G-F-E-D-C-B-A (swap bytes within the 32-bit word, then swap 32-bit words within 64-bit doubleword). This is the mode to use with open-source xHCI drivers when the core is in big-endian mode.
0x2	RSVD2	Reserved. A-B-C-D-E-F-G-H becomes D-C-B-A-H-G-F-E.
0x3	RSVD3	Reserved. A-B-C-D-E-F-G-H becomes E-F-G-H-A-B-C-D.

SATA UCTL XM Bad DMA Type Enumeration
SATA_UCTL_XM_BAD_DMA_TYPE_E

Enumerates the type of DMA error seen.

Value	Mnemonic	Description
0x0	NONE	No error logged.
0x1	ADDR_OOB	AxAddr<63:40> ≠ 0x0.
0x2	MULTIBEAT_BYTE	AxSize = 0x0 and AxLen ≠ 0x0.
0x3	MULTIBEAT_HALFWORD	AxSize = 0x1 and AxLen ≠ 0x0.
0x4	MULTIBEAT_WORD	AxSize = 0x2 and AxLen ≠ 0x0.

26.8 Registers

The CN71XX SATA controller implements two kinds of registers:

- UCTL RSL-type registers.
- UAHC NCB-type registers.

RSL-type registers are accessed indirectly across the IOI. NCB-type registers are accessed directly across the IOI.

Section 26.8.1 shows the UCTL registers, and Section 26.8.2 shows the UAHC registers.

Cavium Confidential For Bowen Yang ANU 02/17/2014

26.8.1 SATA UCTL Registers

UCTL registers are only accessible via RSL. They generally contain basic clock controls, resets, and static configuration bits for the entire SATA subsystem. Note that UCTL registers other than SATA_UCTL_CTL and SATA_UCTL_BIST_STATUS can be accessed only after the host-controller clock is enabled and SATA_UCTL_CTL[SATA_UCTL_RST] is deasserted.

Register addresses not listed in Table 26–2 will return 0xFFFFFFFF_FFFFFFFF on read operations; write operations will have no side-effects. No addresses should alias. The SATA UCTL registers are big-endian and are listed in Table 26–2.

Table 26–2 SATA Registers (UCTL)

Register	Address	CSR Type[1]	Detailed Description
SATA_UCTL_CTL	0x118006C000000	RSL	See page 1232
SATA_UCTL_BIST_STATUS	0x118006C000008	RSL	See page 1234
SATA_UCTL_INTSTAT	0x118006C000030	RSL	See page 1235
SATA_UCTL_SHIM_CFG	0x118006C0000E8	RSL	See page 1236

1. RSL-type registers are accessed indirectly across the I/O Interconnect.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SATA UCTL Control Register
SATA_UCTL_CTL

This register controls clocks, resets, power, and BIST for the SATA. See Table 26–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	CLEAR_BIST	R/W	0	0	BIST fast-clear mode select. There are two major modes of BIST: FULL and CLEAR. 0 = FULL BIST is run by the BIST state machine. 1 = CLEAR BIST is run by the BIST state machine. A clear-BIST run clears all entries in SATA RAMs to 0x0. To avoid race conditions, software must first perform a CSR write operation that puts CLEAR_BIST into the correct state and then perform another CSR write operation to set START_BIST (keeping CLEAR_BIST constant). CLEAR BIST completion is indicated by SATA_UCTL_BIST_STATUS[NDONE]. A BIST clear operation takes almost 2,000 host-controller clock cycles for the largest RAM.
<62>	START_BIST	R/W	0	0	Start BIST. The rising edge starts BIST on the memories in SATA. To run BIST, the host-controller clock must be both configured and enabled, and should be configured to the maximum available frequency given the available coprocessor clock and dividers. Refer to Section 26.3.1 for clock initialization procedures. BIST defect status can be checked after FULL BIST completion, both of which are indicated in SATA_UCTL_BIST_STATUS. The FULL BIST run takes almost 80,000 host-controller clock cycles for the largest RAM.
<61:31>	—	RAZ	—	—	Reserved.
<30>	A_CLK_EN	R/W	0	1	Host-controller clock enable. When set to 1, the host-controller clock is generated. This also enables access to UCTL registers 0x30–0xF8.
<29>	A_CLK_BYP_SEL	R/W	0	0	Select the bypass input to the host-controller clock divider. 0 = use the divided coprocessor clock from the A_CLKDIV divider 1 = use the bypass clock from the GPIO pins (generally bypass is only used for scan purposes). This signal is a multiplexer-select signal; it does not enable the host-controller clock. You must set A_CLK_EN separately. A_CLK_BYP_SEL select should not be changed unless A_CLK_EN is disabled. The bypass clock can be selected and running even if the host-controller clock dividers are not running.
<28>	A_CLKDIV_RST	R/W	1	0	Host-controller-clock divider reset. Divided clocks are not generated while the divider is being reset. This also resets the suspend-clock divider.
<27>	—	RAZ	—	—	Reserved.

Cavium Confidential
Bowen Yang ALU
02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<26:24>	A_CLKDIV_SEL	R/W	0x3	0x3	The host-controller clock frequency is the coprocessor-clock frequency divided by A_CLKDIV_SEL. The host-controller clock frequency must be at or below the requirements listed in Table 26–1, *Maximum ACLK Frequency*. This field can be changed only when A_CLKDIV_RST = 1. The divider values are the following: 0x0 = divide by 1 0x4 = divide by 6 0x1 = divide by 2 0x5 = divide by 8 0x2 = divide by 3 0x6 = divide by 16 0x3 = divide by 4 0x7 = divide by 24
<23:5>	—	RAZ	—	—	Reserved.
<4>	CSCLK_EN	R/W	0	1	Turns on the USB UCTL interface clock (coprocessor clock). This enables access to UAHC registers via the IOI, as well as UCTL registers starting from 0x30 via the RSL bus.
<3:2>	—	RAZ	—	—	Reserved.
<1>	SATA_UAHC_RST	R/W	0	0	Software reset; resets UAHC; active-high.
<0>	SATA_UCTL_RST	R/W	0	0	Software reset; resets UCTL; active-high. Resets UAHC DMA and register shims and the UCTL RSL registers 0x30-0xF8. It does not reset UCTL RSL registers 0x0-0x28. The UCTL RSL registers starting from 0x30 can be accessed only after the host-controller clock is active and UCTL_RST is deasserted.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

SATA UCTL BIST Status Register
SATA_UCTL_BIST_STATUS

This register provides the results from BIST runs of SATA's memories. Bits<41:40,35:32> indicate whether the BIST is complete or not (1 = incomplete, 0 = completed). At completion, bits<9:8,3:0> indicate the status (1 = BIST error, 0 = BIST pass). This registers is reset by IOI reset. See Table 26–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:42>	—	RAZ	—	—	Reserved.
<41>	UCTL_XM_R_BIST_NDONE	RO/H	1	0	BIST is not complete for the UCTL AxiMaster read-data FIFO.
<40>	UCTL_XM_W_BIST_NDONE	RO/H	1	0	BIST is not complete for the UCTL AxiMaster write-data FIFO.
<39:36>	—	RAZ	—	—	Reserved.
<35>	UAHC_P0_RXRAM_BIST_NDONE	RO/H	1	0	BIST is not complete for the UAHC Port 0 RxFIFO RAM.
<34>	UAHC_P1_RXRAM_BIST_NDONE	RO/H	1	0	BIST is not complete for the UAHC Port 1 RxFIFO RAM.
<33>	UAHC_P0_TXRAM_BIST_NDONE	RO/H	1	0	BIST is not complete for the UAHC Port 0 TxFIFO RAM.
<32>	UAHC_P1_TXRAM_BIST_NDONE	RO/H	1	0	BIST is not complete for the UAHC Port 1 TxFIFO RAM.
<31:10>	—	RAZ	—	—	Reserved.
<9>	UCTL_XM_R_BIST_STATUS	RO/H	0	0	BIST status of the UCTL AxiMaster read-data FIFO.
<8>	UCTL_XM_W_BIST_STATUS	RO/H	0	0	BIST status of the UCTL AxiMaster write-data FIFO.
<7:4>	—	RAZ	—	—	Reserved.
<3>	UAHC_P0_RXRAM_BIST_STATUS	RO/H	0	0	BIST status of the UAHC Port0 RxFIFO RAM.
<2>	UAHC_P1_RXRAM_BIST_STATUS	RO/H	0	0	BIST status of the UAHC Port1 RxFIFO RAM.
<1>	UAHC_P0_TXRAM_BIST_STATUS	RO/H	0	0	BIST status of the UAHC Port0 TxFIFO RAM.
<0>	UAHC_P1_TXRAM_BIST_STATUS	RO/H	0	0	BIST status of the UAHC Port1 TxFIFO RAM.

Cavium Confidential For Bowen Yang ALU 02/21/2014

SATA UCTL Interrupt Status Register
SATA_UCTL_INTSTAT

This register provides a summary of different bits of RSL interrupts. See Table 26–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved.
<2>	XM_BAD_DMA	R/W1C/H	0	0	Detected bad DMA access from UAHC to IOI. The error information is logged in SATA_UCTL_SHIM_CFG[XM_BAD_DMA_*]. Received a DMA request from UAHC that violates the assumptions made by the AXI-to-IOI shim. Such scenarios include: illegal length/size combinations and address out-of-bounds. For more information on exact failures, see description in SATA_UCTL_SHIM_CFG[XM_BAD_DMA_TYPE]. The hardware does not translate the request correctly and results may violate IOI protocols.
<1>	XS_NCB_OOB	R/W1C/H	0	0	Detected out-of-bound register access to UAHC over IOI. The UAHC defines 1MB of register space, starting at offset 0x0. Any accesses outside of this register space cause this bit to be set to 1. The error information is logged in SATA_UCTL_SHIM_CFG[XS_NCB_OOB_*].
<0>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SATA UCTL Shim Configuration Register
SATA_UCTL_SHIM_CFG

See Table 26–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	XS_NCB_OOB_WRN	RO/H	—	—	Read/write error log for out-of-bound UAHC register access. 0 = read, 1 = write.
<62:57>	—	RAZ	—	—	Reserved.
<56:48>	XS_NCB_OOB_OSRC	RO/H	—	—	SRCID error log for out-of-bound UAHC register access. The IOI outbound SRCID for the OOB error.
<47>	XM_BAD_DMA_WRN	RO/H	—	—	Read/write error log for bad DMA access from UAHC. 0 = read error log, 1 = write error log.
<46:44>	—	RAZ	—	—	Reserved.
<43:40>	XM_BAD_DMA_TYPE	RO/H	—	—	ErrType error log for bad DMA access from UAHC. Encodes the type of error encountered (error largest encoded value has priority). See SATA_UCTL_XM_BAD_DMA_TYPE_E.
<39:13>	—	RAZ	—	—	Reserved.
<12>	DMA_READ_CMD	R/W	1	1	Selects the IOI read command used by DMA accesses. See SATA_UCTL_DMA_READ_CMD_E.
<11:10>	—	RAZ	—	—	Reserved.
<9:8>	DMA_ENDIAN_MODE	R/W	0x1	0x1	Selects the endian format for DMA accesses to the L2C. See SATA_UCTL_ENDIAN_MODE_E.
<7:2>	—	RAZ	—	—	Reserved.
<1:0>	CSR_ENDIAN_MODE	R/W	0x1	0x1	Selects the endian format for IOI CSR accesses to the UAHC. Note that when UAHC CSRs are accessed via RSL, they are returned as big-endian. See SATA_UCTL_ENDIAN_MODE_E.

Cavium Confidential For Bowen Yang ALU 02/27/2014

26.8.2 SATA UAHC Registers

UAHC registers are implemented by the AHCI host controller; primary access is provided to these registers over NCB and debug access is provided over RSL.

Access to the UAHC registers that the host-controller clock is enabled and SATA_UCTL_CTL[SATA_UCTL_RST] = 0. Additionally, SATA_UCTL_CTL[SATA_UAHC_RST] must be deasserted.

The UAHC registers occupy 4KB of address space. The interior mapping is shown in Table 26–3.

Table 26–3 UAHC Register Address Map

Offset	Register Type
0x000 – 0x0FF	Generic Host Registers, see AHCI spec for details
0x100 – 0x1FF	Port Registers, see AHCI spec for details
0x100 – 0x17F	Port0 Registers
0x180 – 0x1FF	Port1 Registers

For debug purposes, UAHC registers can also be accessed by presenting an NCB-type address over RSL. Registers accessed this way are statically mapped from UAHC's little-endian mode to the RSL's big-endian mode.

The SATA UAHC registers are listed in Table 26–4.

Table 26–4 SATA Registers (UAHC)

Register	Address[1]	CSR Type[2]	Detailed Description
SATA_UAHC_GBL_CAP	0x16C0000000000	NCB	See page 1238
SATA_UAHC_GBL_GHC	0x16C0000000004	NCB	See page 1239
SATA_UAHC_GBL_IS	0x16C0000000008	NCB	See page 1239
SATA_UAHC_GBL_PI	0x16C000000000C	NCB	See page 1239
SATA_UAHC_GBL_VS	0x16C0000000010	NCB	See page 1239
SATA_UAHC_GBL_CCC_CTL	0x16C0000000014	NCB	See page 1240
SATA_UAHC_GBL_CCC_PORTS	0x16C0000000018	NCB	See page 1240
SATA_UAHC_GBL_CAP2	0x16C0000000024	NCB	See page 1240
SATA_UAHC_GBL_BISTAFR	0x16C00000000A0	NCB	See page 1241
SATA_UAHC_GBL_BISTCR	0x16C00000000A4	NCB	See page 1242
SATA_UAHC_GBL_BISTFCTR	0x16C00000000A8	NCB	See page 1242
SATA_UAHC_GBL_BISTSR	0x16C00000000AC	NCB	See page 1243
SATA_UAHC_GBL_BISTDECR	0x16C00000000B0	NCB	See page 1243
SATA_UAHC_GBL_OOBR	0x16C00000000BC	NCB	See page 1243
SATA_UAHC_GBL_TIMER1MS	0x16C00000000E0	NCB	See page 1243
SATA_UAHC_GBL_GPARAM1R	0x16C00000000E8	NCB	See page 1244
SATA_UAHC_GBL_GPARAM2R	0x16C00000000EC	NCB	See page 1244
SATA_UAHC_GBL_PPARAMR	0x16C00000000F0	NCB	See page 1245
SATA_UAHC_GBL_TESTR	0x16C00000000F4	NCB	See page 1245
SATA_UAHC_GBL_VERSIONR	0x16C00000000F8	NCB	See page 1245
SATA_UAHC_GBL_IDR	0x16C00000000FC	NCB	See page 1245
SATA_UAHC_P(0..1)_CLB	0x16C0000000100 + α*0x80	NCB	See page 1246
SATA_UAHC_P(0..1)_FB	0x16C0000000108 + α*0x80	NCB	See page 1246
SATA_UAHC_P(0..1)_IS	0x16C0000000110 + α*0x80	NCB	See page 1246
SATA_UAHC_P(0..1)_IE	0x16C0000000114 + α*0x80	NCB	See page 1247

Cavium Confidential For Bowen Yang ALU 02/2014

Table 26–4 SATA Registers (UAHC)

Register	Address[1]	CSR Type[2]	Detailed Description
SATA_UAHC_P(0..1)_CMD	0x16C0000000118 + α*0x80	NCB	See page 1247
SATA_UAHC_P(0..1)_TFD	0x16C0000000120 + α*0x80	NCB	See page 1248
SATA_UAHC_P(0..1)_SIG	0x16C0000000124 + α*0x80	NCB	See page 1248
SATA_UAHC_P(0..1)_SSTS	0x16C0000000128 + α*0x80	NCB	See page 1248
SATA_UAHC_P(0..1)_SCTL	0x16C000000012C + α*0x80	NCB	See page 1249
SATA_UAHC_P(0..1)_SERR	0x16C0000000130 + α*0x80	NCB	See page 1249
SATA_UAHC_P(0..1)_SACT	0x16C0000000134 + α*0x80	NCB	See page 1250
SATA_UAHC_P(0..1)_CI	0x16C0000000138 + α*0x80	NCB	See page 1250
SATA_UAHC_P(0..1)_SNTF	0x16C000000013C + α*0x80	NCB	See page 1250
SATA_UAHC_P(0..1)_FBS	0x16C0000000140 + α*0x80	NCB	See page 1250
SATA_UAHC_P(0..1)_DMACR	0x16C0000000170 + α*0x80	NCB	See page 1251
SATA_UAHC_P(0..1)_PHYCR	0x16C0000000178 + α*0x80	NCB	See page 1251
SATA_UAHC_P(0..1)_PHYSR	0x16C000000017C + α*0x80	NCB	See page 1251

1. For registers with multiple versions, α represents the values within the specified range.
2. NCB-type registers are accessed directly across the I/O Interconnect.

SATA AHCI HBA Capabilities Register SATA_UAHC_GBL_CAP

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	S64A	RO	1	1	Supports 64-bit addressing.
<30>	SNCQ	RO	1	1	Supports native command queuing.
<29>	SSNTF	RO	1	1	Supports SNotification register.
<28>	SMPS	R/W	—	—	Supports mechanical presence switch.
<27>	SSS	R/W	—	—	Supports staggered spin-up.
<26>	SALP	RO	1	1	Supports agressive link power management.
<25>	SAL	RO	1	1	Supports activity LED.
<24>	SCLO	RO	1	1	Supports command list override.
<23:20>	ISS	RO	0x3	0x3	Interface speed support.
<19>	SNZO	RO	0	0	Supports non-zero DMA offsets.
<18>	SAM	RO	1	1	Supports AHCI mode only.
<17>	SPM	RO	1	1	Supports port multiplier.
<16>	FBSS	RO	1	1	Supports FIS-based switching.
<15>	PMD	RO	1	1	PIO multiple DRQ block.
<14>	SSC	RO	1	1	Slumber state capable.
<13>	PSC	RO	1	1	Partial state capable.
<12:8>	NCS	RO	0x1F	0x1F	Number of command slots.
<7>	CCCS	RO	1	1	Command completion coalescing support.
<6>	EMS	RO	0	0	Enclosure management support.
<5>	SXS	RO	—	—	Supports external SATA.
<4:0>	NP	RO	0x1	0x1	Number of ports. 0x1 = two ports, all other values reserved.

Cavium Confidential For Bowen Yang AL U 9/27/2014

SATA AHCI Global HBA Control Register
SATA_UAHC_GBL_GHC

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	AE	RO	1	1	AHCI enable.
<30:2>	—	RAZ	—	—	Reserved.
<1>	IE	R/W	0	—	Interrupt enable.
<0>	HR	R/W1/H	0	—	HBA reset. Writing a 1 resets the UAHC. Hardware clears this bit once reset is complete.

SATA AHCI Interrupt Status Register
SATA_UAHC_GBL_IS

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:3>	—	RAZ	—	—	Reserved.
<2:0>	IPS	R/W1C/H	0x0	—	Interrupt pending status.

SATA UAHC AHCI Ports Implemented Register
SATA_UAHC_GBL_PI

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:2>	—	RAZ	—	—	Reserved.
<1:0>	PI	R/W	0x0	—	Number of ports implemented. This field is one-time writable, then becomes read-only.

SATA AHCI Version Register
SATA_UAHC_GBL_VS

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	MJR	RO	0x1	0x1	Major version number.
<15:0>	MNR	RO	0x300	0x300	Minor version number. No DEVSLP support.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SATA AHCI CCC Control Register
SATA_UAHC_GBL_CCC_CTL

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	TV	R/W	0x1	0x0	Time-out value. Writable only when EN = 0.
<15:8>	CC	R/W	0x1	—	Command completions. Writable only when EN = 0.
<7:3>	INTR	RO	0x2	0x2	Specifies the port interrupt used by the CCC feature.
<2:1>	—	RAZ	—	—	Reserved.
<0>	EN	R/W	0	—	CCC enable.

SATA AHCI CCC Ports Register
SATA_UAHC_GBL_CCC_PORTS

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:2>	—	RAZ	—	—	Reserved.
<1:0>	PRT	R/W	0x0	0x3	Per port CCC enable.

SATA AHCI HBA Capabilities Extended Register
SATA_UAHC_GBL_CAP2

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:6>	—	RAZ	—	—	Reserved.
<5>	DESO	RO	0	0	Device sleep entrance from slumber only.
<4>	SADM	RO	0	0	Supports aggressive device sleep management.
<3>	SDS	RO	0	0	Supports device sleep.
<2>	APST	RO	1	1	Automatic partial to slumber transitions.
<1>	NVMP	RO	0	0	NVMHCI present.
<0>	BOH	RO	0	0	Supports BIOS/OS handoff.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SATA UAHC BIST Activate FIS Register SATA_UAHC_GBL_BISTAFR

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	—	RAZ	—	—	Reserved.
<15:8>	NCP	RO	0x0	—	Bits 7:0 of the second DWORD of BIST activate FIS. 0xF1 = low transition density pattern (LTDP) 0xB5 = high transition density pattern (HTDP) 0xAB = low frequency spectral component pattern (LFSCP) 0x7F = simultaneous switching outputs pattern (SSOP) 0x78 = mid frequency test pattern (MFTP) 0x4A = high frequency test pattern (HFTP) 0x7E = low frequency test pattern (LFTP) All other values = lone bit pattern (LBP)
<7:0>	PD	RO	0x0	—	Bits 23:16 of the first DWORD of the BIST activate FIS. Only the following values are supported: 0x10 = Far-end retimed 0xC0 = Far-end transmit only 0xE0 = Far-end transmit only with scrambler bypassed

Cavium Confidential For Bowen Yang ALU 02/27/2014

SATA UAHC BIST Control Register
SATA_UAHC_GBL_BISTCR

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:26>	—	RAZ	—	—	Reserved.
<25>	OLD_PHY_READY	RO	0	0	Old phy_ready.
<24>	LATE_PHY_READY	RO	0	0	Late phy_ready.
<23:21>	—	RAZ	—	—	Reserved.
<20>	FERLIB	WO	0	—	Far-end retimed loopback.
<19>	—	RAZ	—	—	Reserved.
<18>	TXO	WO	0	—	Transmit only.
<17>	CNTCLR	WO	0	—	Counter clear.
<16>	NEALB	WO	0	—	Near-end analog loopback.
<15>	LLB	R/W	0	0	Lab loopback mode.
<14>	—	RAZ	—	—	Reserved.
<13>	ERRLOSSEN	R/W	1	1	Error loss detect enable.
<12>	SDFE	R/W	1	1	Signal detect feature enable.
<11>	—	RAZ	—	—	Reserved.
<10:8>	LLC	R/W	0x7	0x0	Link layer control. <10> = RPD - repeat primitive drop enable <9> = DESCRAM - descrambler enable <8> = SCRAM - scrambler enable
<7>	—	RAZ	—	—	Reserved.
<6>	ERREN	R/W	0	—	Error enable.
<5>	FLIP	R/W	0	0	Flip disparity.
<4>	PV	R/W	0	—	Pattern version.
<3:0>	PATTERN	RO	0x0	—	SATA compliant pattern selection.

SATA UAHC BIST FIS Count Register
SATA_UAHC_GBL_BISTFCTR

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	COUNT	RO	0x0	—	Received BIST FIS count.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SATA UAHC BIST Status Register
SATA_UAHC_GBL_BISTSR

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	—	RAZ	—	—	Reserved.
<23:16>	BRSTERR	RO	—	—	Burst error.
<15:0>	FRAMERR	RO	—	—	Frame error.

SATA UAHC BIST DWORD Error Count Register
SATA_UAHC_GBL_BISTDECR

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DWERR	RO	0x0	0x0	DWORD error count.

SATA UAHC OOB Register
SATA_UAHC_GBL_OOBR

This register xxx. See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	WE	R/W/H	0	0	Write enable.
<30:24>	CWMIN	R/W/H	0x7	—	COMWAKE minimum value. Writable only if WE is set.
<23:16>	CWMAX	R/W/H	0xE	—	COMWAKE maximum value. Writable only if WE is set.
<15:8>	CIMIN	R/W/H	0x18	—	COMINIT minimum value. Writable only if WE is set.
<7:0>	CIMAX	R/W/H	0x2B	—	COMINIT maximum value. Writable only if WE is set.

SATA UAHC Timer 1ms Register
SATA_UAHC_GBL_TIMER1MS

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	—	RAZ	—	—	Reserved.
<19:0>	TIMV	R/W	—	—	1ms timer value. Writable only when SATA_UAHC_GBL_CCC_CTL[EN] = 0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SATA UAHC Global Parameter Register 1
SATA_UAHC_GBL_GPARAM1R

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	ALIGN_M	RO	0	0	RX data alignment mode (ALIGN_MODE).
<30>	RX_BUFFER	RO	1	1	RX data buffer mode (RX_BUFFER_MODE).
<29:28>	PHY_DATA	RO	0x2	0x2	PHY data width (PHY_DATA_WIDTH).
<27>	PHY_RST	RO	1	1	PHY reset mode (PHY_RST_MODE).
<26:21>	PHY_CTRL	RO	0x20	0x20	PHY control width (PHY_CTRL_W).
<20:15>	PHY_STAT	RO	0x20	0x20	PHY status width (PHY_STAT_W).
<14>	LATCH_M	RO	0	0	Latch mode (LATCH_MODE).
<13:11>	PHY_TYPE	RO	0x2	0x2	PHY interface type (PHY_INTERFACE_TYPE).
<10>	RETURN_ERR	RO	1	1	AMBA error response (RETURN_ERR_RESP).
<9:8>	AHB_ENDIAN	RO	0x0	0x0	AHB bus endianness (AHB_ENDIANNESS).
<7>	S_HADDR	RO	0	0	AMBA slave address bus width (S_HADDR_WIDTH).
<6>	M_HADDR	RO	1	1	AMBA master address bus width (M_HADDR_WIDTH).
<5:3>	S_HDATA	RO	0x0	0x0	AMBA slave data width (S_HDATA_WIDTH).
<2:0>	M_HDATA	RO	0x1	0x1	AMBA master data width (M_HDATA_WIDTH).

SATA UAHC Global Parameter Register 2
SATA_UAHC_GBL_GPARAM2R

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	—	RAZ	—	—	Reserved.
<19>	BIST_M	RO	0	0	BIST loopback checking depth (BIST_MODE).
<18>	FBS_MEM_S	RO	0	0	Context RAM memory location (FBS_MEM_S).
<17:16>	FBS_PMPN	RO	0x3	0x3	Maximum number of port multiplier ports (FBS_PMPN_MAX).
<15>	FBS_SUPPORT	RO	1	1	FIS-based switching support (FBS_SUPPORT).
<14>	DEV_CP	RO	1	1	Cold presence detect (DEV_CP_DET).
<13>	DEV_MP	RO	1	1	Mechanical presence switch (DEV_MP_SWITCH).
<12>	ENCODE_M	RO	1	1	8b/10b encoding/decoding (ENCODE_MODE).
<11>	RXOOB_CLK_M	RO	1	1	RX OOB clock mode (RXOOB_CLK_MODE).
<10>	RX_OOB_M	RO	1	1	RX OOB mode (RX_OOB_MODE).
<9>	TX_OOB_M	RO	1	1	TX OOB mode (TX_OOB_MODE).
<8:0>	RXOOB_CLK	RO	0x64	0x64	RX OOB clock frequency (RXOOB_CLK).

Cavium Confidential For Boren Yang ALU 02/12/2014

SATA UAHC Port Parameter Register
SATA_UAHC_GBL_PPARAMR

The port is selected by the SATA_UAHC_GBL_TESTR[PSEL]. See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:12>	—	RAZ	—	—	Reserved.
<11>	TX_MEM_M	RO	1	1	TX FIFO memory read port type (P*n*_TX_MEM_MODE).
<10>	TX_MEM_S	RO	0	0	TX FIFO memory type (P*n*_TX_MEM_SELECT).
<9>	RX_MEM_M	RO	1	1	RX FIFO memory read port type (P*n*_RX_MEM_MODE).
<8>	RX_MEM_S	RO	0	0	RX FIFO memory type (P*n*_RX_MEM_SELECT).
<7:4>	TXFIFO_DEPTH	RO	0x7	0x7	TX FIFO depth in FIFO words.
<3:0>	RXFIFO_DEPTH	RO	0x7	0x7	RX FIFO depth in FIFO words.

SATA UAHC Test Register
SATA_UAHC_GBL_TESTR

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:19>	—	RAZ	—	—	Reserved.
<18:16>	PSEL	R/W	0x0	—	Port select.
<15:1>	—	RAZ	—	—	Reserved.
<0>	TEST_IF	R/W	0	0	Test interface.

SATA UAHC Version Register
SATA_UAHC_GBL_VERSIONR

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	VER	RO	0x343130C0	0x343130C0	SATA IP version number.

SATA UAHC ID Register
SATA_UAHC_GBL_IDR

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	ID	RO	0x0	0x0	Core ID.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SATA UAHC Command-List Base-Address Registers
SATA_UAHC_P(0..1)_CLB

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	CLB	R/W	0x0	—	Command-list base address.
<9:0>	—	RAZ	—	—	Reserved.

SATA UAHC FIS Base-Address Registers
SATA_UAHC_P(0..1)_FB

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	FB	R/W	0x0	—	FIS base address.
<7:0>	—	RAZ	—	—	Reserved.

SATA UAHC Interrupt Status Registers
SATA_UAHC_P(0..1)_IS

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	CPDS	R/W1C/H	0	—	Cold-port detect status.
<30>	TFES	R/W1C/H	0	—	Task-file error status.
<29>	HBFS	R/W1C/H	0	—	Host-bus fatal-error status.
<28>	HBDS	R/W1C/H	0	—	Host-bus data-error status.
<27>	IFS	R/W1C/H	0	—	Interface fatal-error status.
<26>	INFS	R/W1C/H	0	—	Interface non-fatal-error status.
<25>	—	RAZ	—	—	Reserved.
<24>	OFS	R/W1C/H	0	—	Overflow status.
<23>	IMPS	R/W1C/H	0	—	Incorrect port-multiplier status.
<22>	PRCS	RO/H	0	—	PHY-ready change status.
<21:8>	—	RAZ	—	—	Reserved.
<7>	DMPS	R/W1C/H	0	—	Device mechanical-presence status.
<6>	PCS	RO/H	0	—	Port-connect-change status.
<5>	DPS	R/W1C/H	0	—	Descriptor processed.
<4>	UFS	RO	0	—	Unknown FIS interrupt.
<3>	SDBS	R/W1C/H	0	—	Set device bits interrupt.
<2>	DSS	R/W1C/H	0	—	DMA setup FIS interrupt.
<1>	PSS	R/W1C/H	0	—	PIO setup FIS interrupt.
<0>	DHRS	R/W1C/H	0	—	Device-to-host register FIS interrupt.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SATA UAHC Interrupt Enable Registers
SATA_UAHC_P(0..1)_IE

This register xxx. See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	CPDE	R/W	0	—	Cold-port-detect enable.
<30>	TFEE	R/W	0	—	Task-file-error enable.
<29>	HBFE	R/W	0	—	Host-bus fatal-error enable.
<28>	HBDE	R/W	0	—	Host-bus data-error enable.
<27>	IFE	R/W	0	—	Interface fatal-error enable.
<26>	INFE	R/W	0	—	Interface non-fatal-error enable.
<25>	—	RAZ	—	—	Reserved.
<24>	OFE	R/W	0	—	Overflow enable.
<23>	IMPE	R/W	0	—	Incorrect port-multiplier enable.
<22>	PRCE	R/W	0	—	PHY-ready-change enable.
<21:8>	—	RAZ	—	—	Reserved.
<7>	DMPE	R/W	0	—	Device mechanical-presence enable.
<6>	PCE	R/W	0	—	Port-connect-change enable.
<5>	DPE	R/W	0	—	Descriptor-processed enable.
<4>	UFE	R/W	0	—	Unknown-FIS-interrupt enable.
<3>	SDBE	R/W	0	—	Set device-bits-interrupt enable.
<2>	DSE	R/W	0	—	DMA-setup FIS interrupt enable.
<1>	PSE	R/W	0	—	PIO-setup FIS interrupt enable.
<0>	DHRE	R/W	0	—	Device-to-host register FIS interrupt enable.

SATA UAHC Command Registers
SATA_UAHC_P(0..1)_CMD

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:28>	ICC	R/W	0x0	—	Interface communication control.
<27>	ASP	R/W	0	—	Aggressive slumber/partial.
<26>	ALPE	R/W	0	—	Aggressive link-power-management enable.
<25>	DLAE	R/W	0	—	Drive LED on ATAPI enable.
<24>	ATAPI	R/W	0	—	Device is ATAPI.
<23>	APSTE	R/W	0	—	Automatic partial to slumber transitions enable.
<22>	FBSCP	R/W	0	—	FIS-based switching capable port. Write-once.
<21>	ESP	R/W	0	—	External SATA port. Write-once.
<20>	CPD	R/W	0	—	Cold-presence detection. Write-once.
<19>	MPSP	R/W	0	—	Mechanical presence switch attached to port. Write-once.
<18>	HPCP	R/W	—	—	Hot-plug-capable support. Write-once.
<17>	PMA	R/W	0	—	Port multiplier attached.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<16>	CPS	RO	0	—	Cold presence state.
<15>	CR	RO	0	—	Command list running.
<14>	FR	RO/H	0	—	FIS receive running.
<13>	MPSS	RO	—	—	Mechanical presence switch state.
<12:8>	CCS	RO	0	—	Current-command slot.
<7:5>	—	RAZ	—	—	Reserved.
<4>	FRE	R/W	0	—	FIS-receive enable.
<3>	CLO	WO	0	—	Command-list override.
<2>	POD	R/W	0	—	Power-on device. R/W only if CPD = 1, else read only.
<1>	SUD	R/W	0	—	Spin-up device. R/W only if SATA_UAHC_GBL_CAP[SSS] = 1, else read only. Setting this bit triggers a COMRESET initialization sequence.
<0>	ST	R/W	0	—	Start.

SATA UAHC Task File Data Registers
SATA_UAHC_P(0..1)_TFD

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	—	RAZ	—	—	Reserved.
<15:8>	TFERR	RO	0x0	—	Copy of task-file error register.
<7:0>	STS	RO/H	0x7F	—	Copy of task-file status register.

SATA UAHC Signature Registers
SATA_UAHC_P(0..1)_SIG

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	SIG	RO/H	0xFFFFFFFF	—	Signature.

SATA UAHC SATA Status Registers
SATA_UAHC_P(0..1)_SSTS

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:12>	—	RAZ	—	—	Reserved.
<11:8>	IPM	RO/H	0x0	—	Interface power management.
<7:4>	SPD	RO/H	0x0	—	Current interface speed.
<3:0>	DET	RO/H	0x0	—	Device detection.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SATA UAHC SATA Control Registers
SATA_UAHC_P(0..1)_SCTL

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:10>	—	RAZ	—	—	Reserved.
<9:8>	IPM	R/W	0x0	—	Interface power-management transitions allowed.
<7:6>	—	RAZ	—	—	Reserved.
<5:4>	SPD	R/W	0x0	—	Speed allowed.
<3>	—	RAZ	—	—	Reserved.
<2:0>	DET	R/W	0x0	—	Device-detection initialization.

SATA UAHC SATA Error Registers
SATA_UAHC_P(0..1)_SERR

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:27>	—	RAZ	0x0	—	Reserved.
<26>	DIAG_X	R/W1C/H	0	—	Exchanged.
<25>	DIAG_F	R/W1C/H	0	—	Unknown FIS type.
<24>	DIAG_T	R/W1C/H	0	—	Transport state transition error.
<23>	DIAG_S	R/W1C/H	0	—	Link sequence error.
<22>	DIAG_H	R/W1C/H	0	—	Handshake error.
<21>	DIAG_C	R/W1C/H	0	—	CRC error.
<20>	DIAG_D	R/W1C/H	0	—	Disparity error.
<19>	DIAG_B	R/W1C/H	0	—	10B to 8B decode error.
<18>	DIAG_W	R/W1C/H	0	—	COMWAKE detected.
<17>	DIAG_I	R/W1C/H	0	—	PHY internal error.
<16>	DIAG_N	R/W1C/H	0	—	PHY ready change.
<15:12>	—	RAZ	—	—	Reserved.
<11>	ERR_E	R/W1C/H	0	—	Internal error.
<10>	ERR_P	R/W1C/H	0	—	Protocol error.
<9>	ERR_C	R/W1C/H	0	—	Non-recovered persistent communication error.
<8>	ERR_T	R/W1C/H	0	—	Non-recovered transient data integrity error.
<7:2>	—	RAZ	—	—	Reserved.
<1>	ERR_M	R/W1C/H	0	—	Recovered communication error.
<0>	ERR_I	R/W1C/H	0	—	Recovered data integrity.

Cavium Confidential For
Buwen Yang
ALU
02/27/2014

SATA UAHC SATA Active Registers
SATA_UAHC_P(0..1)_SACT

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DS	R/W1S/H	0x0	—	Device status.

SATA UAHC Command Issue Registers
SATA_UAHC_P(0..1)_CI

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	CI	R/W1S/H	0x0	—	Command issued.

SATA UAHC SATA Notification Registers
SATA_UAHC_P(0..1)_SNTF

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	—	RAZ	—	—	Reserved.
<15:0>	PMN	R/W1C/H	0x0	—	PM notify.

SATA UAHC FIS-Based Switching Control Registers
SATA_UAHC_P(0..1)_FBS

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	—	RAZ	—	—	Reserved.
<19:16>	DWE	RO	0x0	0x0	Device with error.
<15:12>	ADO	RO	0xF	0xF	Active device optimization.
<11:8>	DEV	R/W	0x0	—	Device to issue.
<7:3>	—	RAZ	—	—	Reserved.
<2>	SDE	RO	0	0	Single device error.
<1>	DEC	R/W1C/H	0	0	Device error clear.
<0>	EN	R/W	0	—	Enable.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SATA UAHC DMA Control Registers
SATA_UAHC_P(0..1)_DMACR

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:8>	—	RAZ	—	—	Reserved.
<7:4>	RXTS	R/W	—	—	Receive transaction size. This field is R/W when SATA_UAHC_P(0..1)_CMD[ST] = 0 and read only when SATA_UAHC_P(0..1)_CMD[ST] = 1.
<3:0>	TXTS	R/W	—	—	Transmit transaction size. This field is R/W when SATA_UAHC_P(0..1)_CMD[ST] = 0 and read only when SATA_UAHC_P(0..1)_CMD[ST] = 1.

SATA UAHC PHY Control Registers
SATA_UAHC_P(0..1)_PHYCR

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	CTRL	R/W	0x0	0x0	Port PHY control.

SATA UAHC PHY Status Registers
SATA_UAHC_P(0..1)_PHYSR

See Table 26–4 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	STAT	RO	0x0	0x0	Port PHY status.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 27

TDM/PCM Interface

This chapter contains the following subjects:

- Overview
- Signal Usage
- Clocking
- TDM Engines
- Initialization Sequence
- TDM/PCM Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The TDM/PCM unit is a highly flexible implementation capable of connecting to many variations on the TDM serial interface.

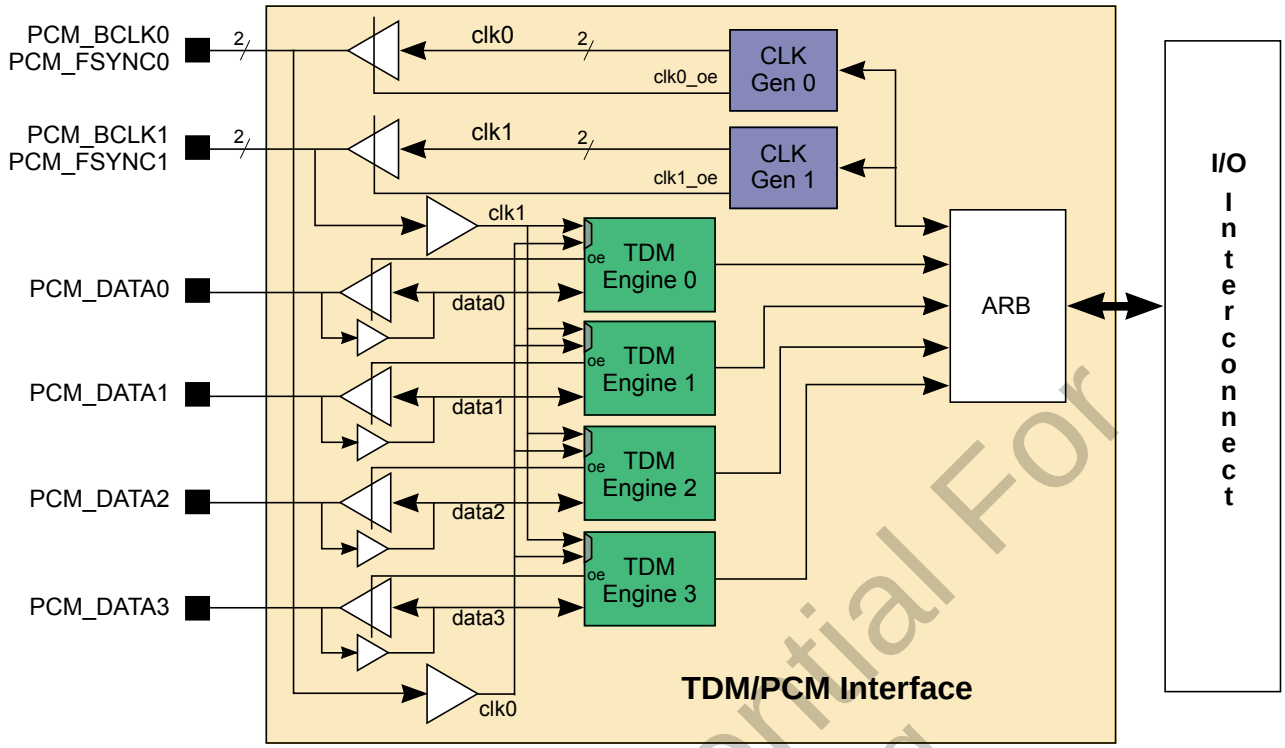
The key features are:

- Two-bit clock (BCLK)/FSYNC pairs.
  - Each is capable of being generated or received
  - Supports all frequencies from 256KHz to 32.768 MHz
  - 32 to 512 (256KHz to 32.768MHz) channels are supported (assuming standard 8-KHz FSYNC signal)
  - FSYNC polarity, location, length, and sample points are highly configurable
  - Support for 1, 2, or 4 clocks per bit
  - Support for T1 frame bit
  - Extra and missed FSYNC detection
- Four data highways
  - Each can be associated with either of the two BCLK/FSYNC pairs
  - Transmit/receive on any combination of channels
  - Support for LSB-first or MSB-first data transmission/reception
  - Highly configurable sample points and drive times
  - Transmit and receive DMA engines per highway (for a total of 8)
    - Transmit engine
      - 16-byte FIFO to buffer output data
      - Programmable fetch threshold between 0 and 15 bytes
      - Programmable fetch size between 1 and 16 bytes
      - Configurable transmit-memory region size
      - Interrupts based on number of frames read and memory-region wrap
    - Receive engine
      - Received data written to main memory with no processor intervention
      - Configurable receive-memory region size
      - Interrupts based on number of frames written and memory-region wrap

Cavium Confidential For Bowen Yang ALU 02/27/2014

A high-level block diagram is shown in Figure 27–1.

Figure 27–1 TDM/PCM Block Diagram



27.1 Signal Usage

All PCM/TDM signals are dual-function signals. When they are not being used for PCM/TDM purposes, they can function as GPIO pins (see Chapter 33 for more details). Table 27–1 describes the PCM/TDM signals and the CSR bits that switch them from standard GPIO pins to their PCM/TDM functionality. In addition, GPIO_XBIT_CFG(16..19)[TX_OE] or GPIO_BIT_CFG(12..15)[TX_OE] must be 0 for any of these pins when using the PCM/TDM functionality associated with the given pin.

Table 27–1 Signal Functionality

Signal Name	Enable	Description
GPIO_19/PCM_BCLK0	PCM_CLK0_GEN[N] ≠ 0x0	Bit clock for clock pair 0
GPIO_18/PCM_FSYNC0	PCM_CLK0_GEN[N] ≠ 0x0	FSYNC for clock pair 0
GPIO_17/PCM_DATA0	PCM0_TDM_CFG[ENABLE] = 1	Data highway for TDM engine 0
GPIO_16/PCM_DATA1	PCM1_TDM_CFG[ENABLE] = 1	Data highway for TDM engine 1
GPIO_15/PCM_BCLK1	PCM_CLK1_GEN[N] ≠ 0x0	Bit clock for clock pair 1
GPIO_14/PCM_FSYNC1	PCM_CLK1_GEN[N] ≠ 0x0	FSYNC for clock pair 1
GPIO_13/PCM_DATA2	PCM2_TDM_CFG[ENABLE] = 1	Data highway for TDM engine 2
GPIO_12/PCM_DATA3	PCM3_TDM_CFG[ENABLE] = 1	Data highway for TDM engine 3

As Table 27–1 shows, most of the pins can be independently used in PCM/TDM mode and GPIO mode. For the remainder of this chapter, the pins will be referred to by their PCM/TDM names.

Cavium Confidential For Bowen Yang A0 02/27/2015

27.2 Clocking

The PCM/TDM unit has completely independent BCLK/FSYNC pairs. The PCM_CLK(0,,1)_CFG and PCM_CLK(0,,1)_GEN registers control the characteristics of each clock pair, as shown in Table 27–2.

Table 27–2 BCLK/FSYNC Generation

CSR Field	Description
PCM_CLK*n*_GEN[N]	Controls the frequency of the bit clock generated. Must be 0 for reception.
PCM_CLK*n*_GEN[NUMSAMP]	Specifies the number of BCLK samples required to detect a change in BCLK. A value of 0 effectively removes filtering.
PCM_CLK*n*_CFG[BCLKPOL]	If set to 1, this field inverts the PCM_BCLKn pin on both transmissions and receptions.
PCM_CLK*n*_CFG[BITLEN]	Specifies the number of BCLK cycles per bit time.
PCM_CLK*n*_CFG[NUMSLOTS]	Specifies the number of 8-bit timeslots per frame (maximum: 512 – EXTRABIT).
PCM_CLK*n*_CFG[EXTRABIT]	Indicates whether an extra frame bit is present.
PCM_CLK*n*_CFG[FSYNCLOC]	Specifies the number of BCLK edges before the first bit of the frame at which FSYNC asserts.
PCM_CLK*n*_CFG[FSYNCLEN]	Specifies the number of BCLK edges for which FSYNC remains asserted.
PCM_CLK*n*_CFG[FSYNCPOL]	Indicates whether FSYNC should assert low or high.
PCM_CLK*n*_CFG[FSYNCSAMP]	Specifies the number of core clock cycles after BCLK edge to sample PCM_FSYNC*n*.

27.2.1 BCLK Generation

The BCLK clock-generator frequency is controlled by the following relation:

$$f(BCLK) = \frac{f(SCLK) \times PCM_CLKn_GEN[N]}{2^{32}}$$

where:
f(BCLK) is the frequency of BCLK
f(SCLK) is the frequency of SCLK (coprocessor clock)

- To stop the BCLK clock generator, PCM_CLK*n*_GEN[N] must be written to 0. This also disables the driving of the PCM_BCLK*n* and PCM_FSYNC*n* pins.
- Any value other than 0 turns the PCM_BCLK*n* and PCM_FSYNC*n* pins into outputs and prevents their use as a GPIO pin. Supported BCLK frequencies range from 256 KHz to 32.768 MHz.

An example of a BCLK frequency is shown in Table 27–3.

Table 27–3 Sample BCLK Frequency

Desired BCLK Frequency	SCLK Frequency	PCM_CLK*n*_GEN[N]
8.192 MHz	350 MHz	0x05FDEAB9
	500 MHz	0x0431BDE8

Cavium Confidential For Boyan Yang ASU 02/27/2017

27.2.2 FSYNC Generation

The FSYNC generator is enabled whenever the corresponding BCLK generator is enabled. The exact shape of the FSYNC waveform is determined by the fields PCM_CLK*n*_CFG[BITLEN, NUMSLOTS, EXTRABIT, FSYNCLOC, FSYNCLEN, FSYNCPOL].

The FSYNC frequency will be whatever is required to satisfy settings for these fields. If a standard 8-KHz FSYNC frequency is desired, [BITLEN, NUMSLOTS, EXTRABIT] must have the correct values for the chosen BCLK frequency.

27.2.3 BCLK Reception

BCLK is received by the TDM/PCM module via sampling the PCM_BCLKn pin into the SCLK (coprocessor clock) domain. This is true whether CN71XX is generating BCLK or not. The PCM_CLK*n*_GEN[NUMSAMP] setting is used to filter noise on a BCLK input with a slow transition time. If the CN71XX is generating BCLK, it can be set to 0.

Based on these settings, an internal BCLK is generated with one-rise edge per bit time. This internal BCLK is used for FSYNC reception and all data sampling and driving in the TDM engines.

27.2.4 FSYNC Reception

The PCM_FSYNC0/1 pin is sampled to detect an assertion. The sample point is defined as a number of SCLK cycles (specified by PCM_CLK*n*_CFG[FSYNCSAMP]) after a BCLK falling or rising edge. If PCM_CLK*n*_CFG[FSYNCLOC] is odd a falling edge is used; if it is even a rising edge is used.

Once an assertion has been detected, it is moved by the number of BCLK edges specified by PCM_CLK*n*_CFG[FSYNCLOC] and the start of the frame (bit 0 of bit time 0) is aligned to this point. If this does not match the expected position of FSYNC assertion, an FSYNC_EXTRA error is logged in the PCM_INT_SUM register associated with any TDM engines using the BCLK/FSYNC pair. In addition, the TDM engine stops transmitting and receiving. An optional interrupt may be sent as well.

A second check on FSYNC reception looks at when the internal counters think an FSYNC assertion should have happened. If no FSYNC assertion is seen, an FSYNC_MISSING error is logged in the PCM_INT_SUM register associated with any TDM engines using the BCLK/FSYNC pair. In addition, the TDM engine stops transmitting and receiving. An optional interrupt may be sent as well.

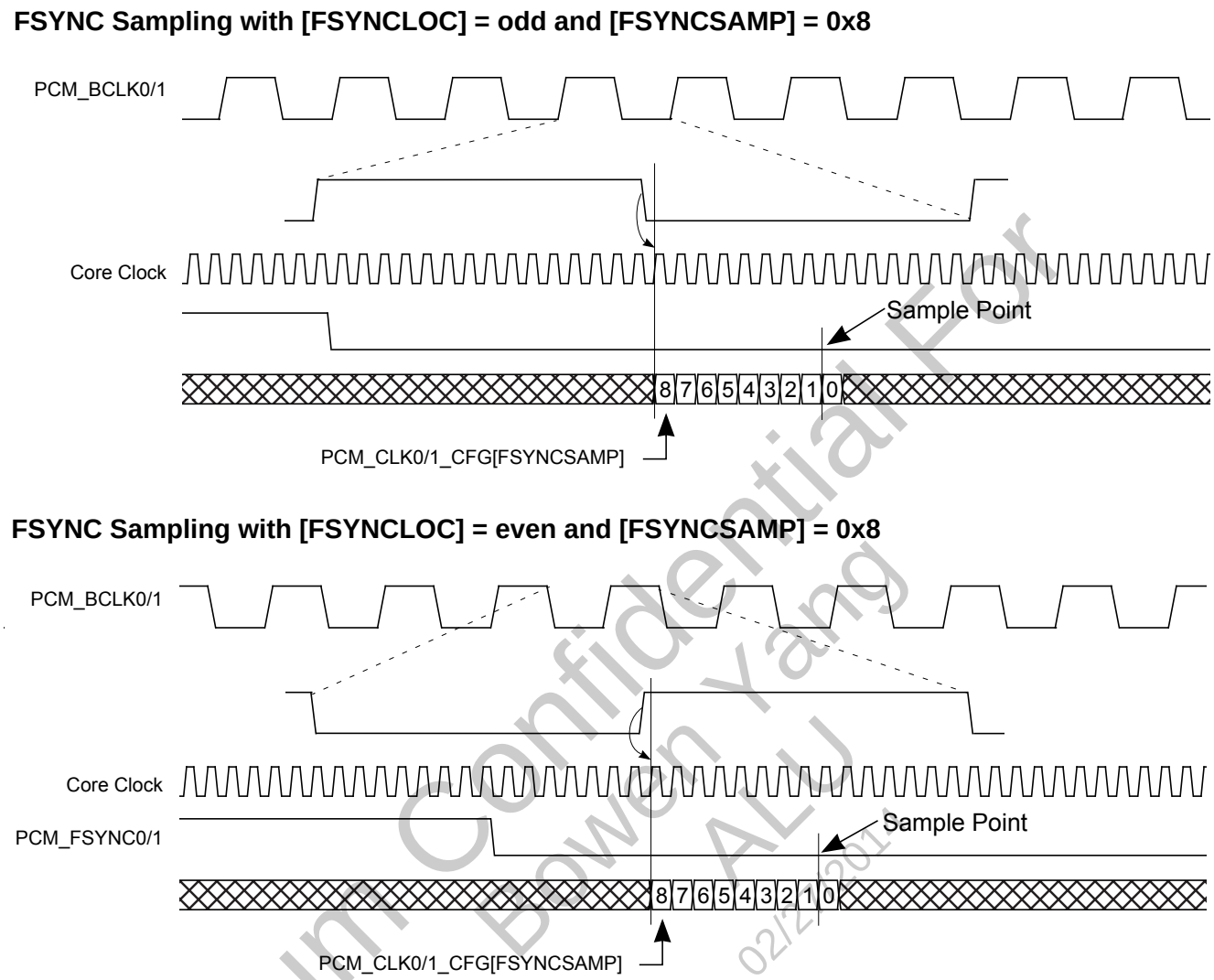
Cavium Confidential For Alan Bowen Kang 6/17/2014

27.2.5 Examples BCLK/FSYNC Waveforms

27.2.5.1 FSYNC Sampling

Examples of FSYNC sampling are shown in Figure 27–2.

Figure 27–2 FSYNC Sampling

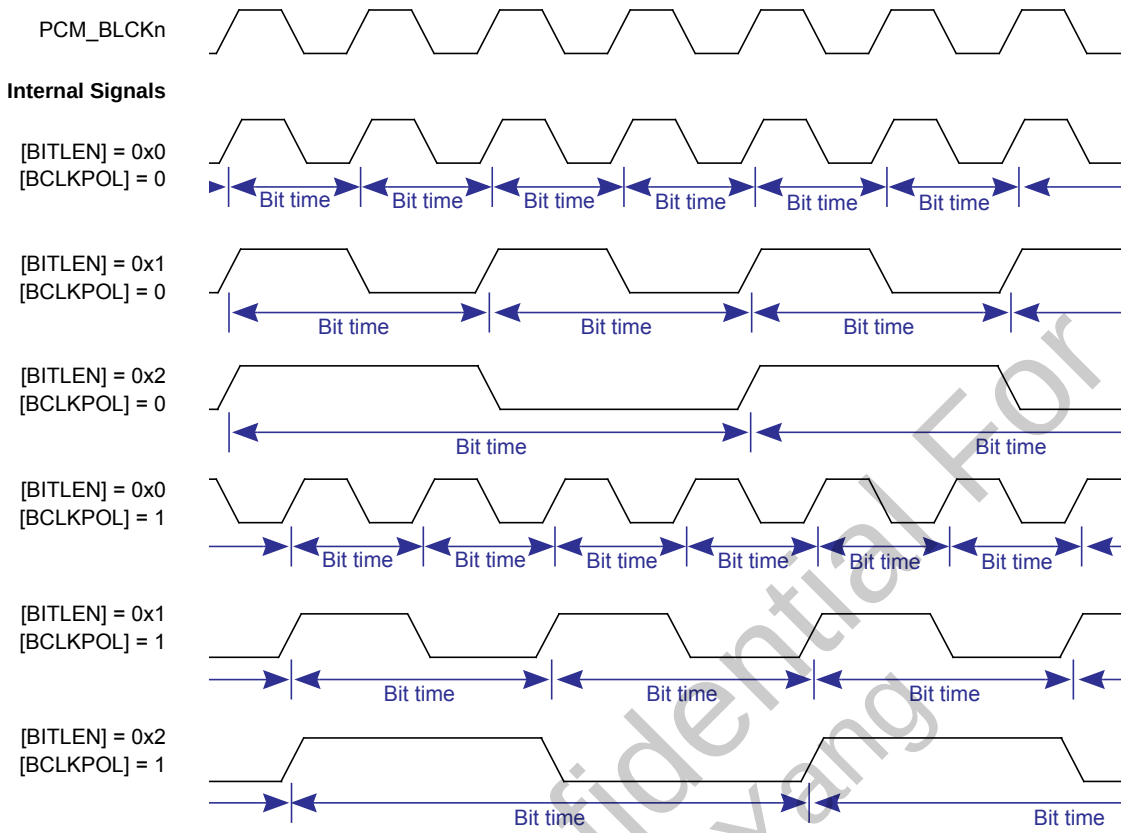


Cavium Confidential For Bowen Yang ALU 02/2/2014

27.2.5.2 Internal BCLK

An example of BCLK waveforms are shown in Figure 27–3.

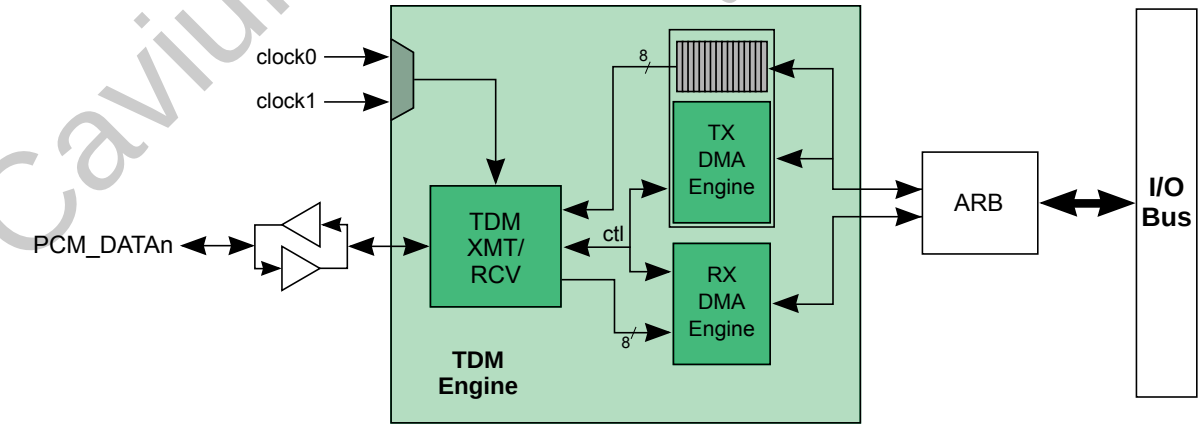
Figure 27–3 BCLK Waveforms



27.3 TDM Engines

The CN71XX has four TDM engines, each attached to its own PCM_DATA*n* pin. Configuration for the engine is controlled via the PCM*n*_TDM_CFG and PCM*n*_DMA_CFG registers. A TDM engine cannot operate without its corresponding DMA engines. The TDM-engine block diagram is shown in Figure 27–4.

Figure 27–4 TDM-Engine Block Diagram



Cavium Confidential For
Bowen Yang
AU
02/27/2014

27.3.1 TDM Engine Configuration

Each TDM engine can use either of the two BCLK/FSYNC pairs. The PCM*n*_TDM_CFG[USECLK1] field selects which one is used. Note that the TDM engine must be disabled before changing the [USECLK1] setting.

The total number of slots in a frame is inherited from the chosen BCLK/FSYNC pair (PCM_CLK(0,,1)_CFG[NUMSLOTS]). The number of transmit and receive slots are specified in each TDM engine via the PCM*n*_TDM_CFG[RXSLOTS, TXSLOTS] settings.

Note that each TDM engine can transmit or receive on any of the timeslots independently. The exact timeslots that are transmitted/received is specified by the PCM*n*_TXMSK(0..7) and PCM*n*_RXMSK(0..7) registers. The sum of all 1s in the PCM*n*_TXMSK(0..7) registers must equal PCM*n*_TDM_CFG[TXSLOTS]. Likewise, the sum of all 1s in the PCM*n*_RXMSK(0..7) registers must equal PCM*n*_TDM_CFG[RXSLOTS].

The PCM*n*_TDM_CFG[LSBFIRST] bit, when set to 1, tells the TDM engine to send the least-significant bit of the data byte first instead of the most-significant bit, otherwise the most-significant bit is sent first.

PCM*n*_TDM_CFG[SAMPPT] controls the sample point for receiving data from the PCM_DATA*n* pin. It specifies the number of SCLK (coprocessor clock) cycles from the start of a bit time to sample the pin.

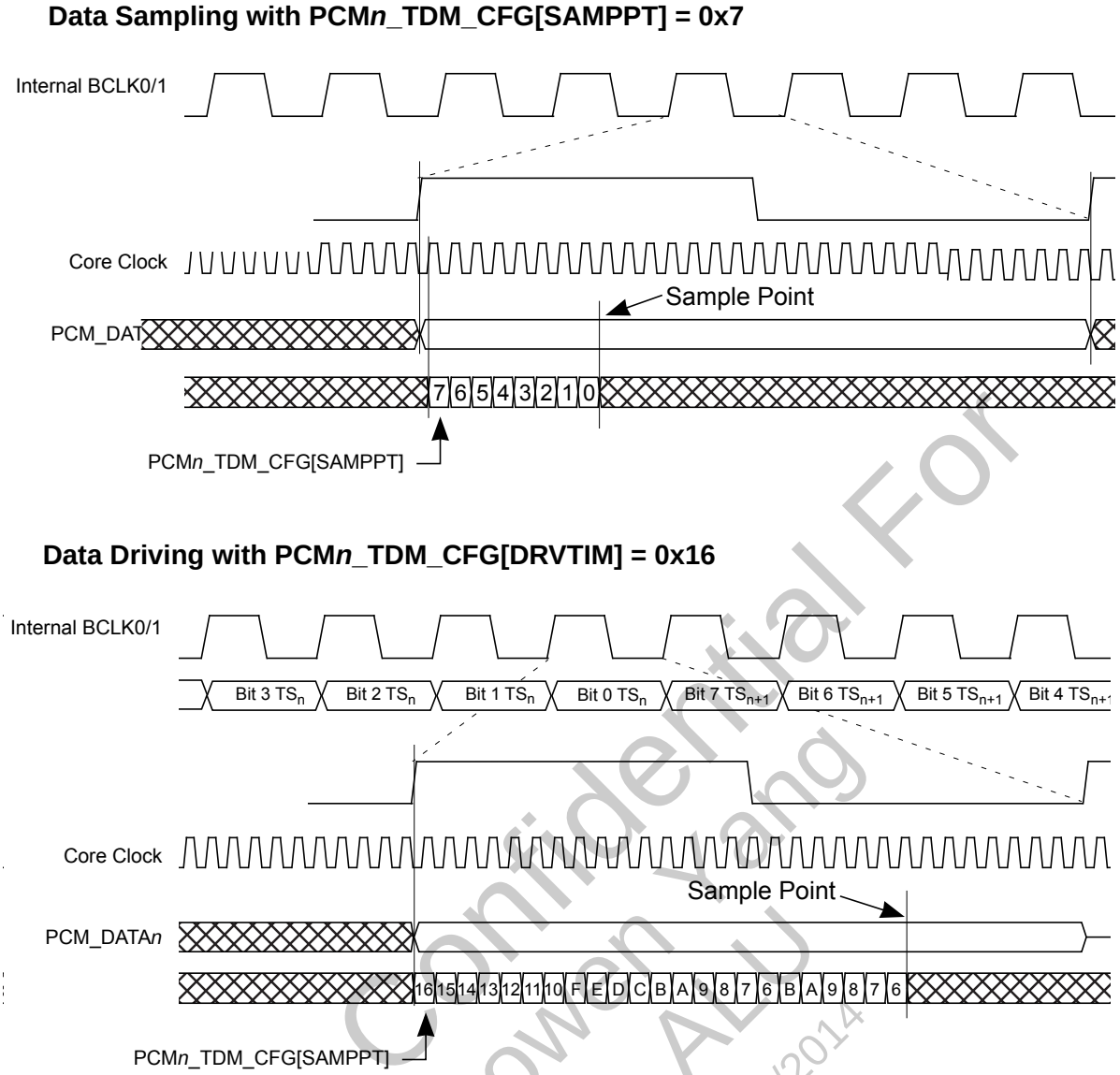
When the TDM engine reaches the end of a transmission timeslot and is not going to be transmitting in the next timeslot, PCM*n*_TDM_CFG[DRVTIM] specifies the number of SCLK cycles from the start of that final bit time that the CN71XX's driver should turn off. This allows the CN71XX to release the PCM_DATA*n* wire so another device can start driving the next timeslot without contention.

The PCM*n*_TDM_CFG[ENABLE] bit switches the PCM_DATA*n* pin from a GPIO to the TDM functionality.

Examples of data sampling and data driving are shown in Figure 27–5.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 27–5 Data Sampling/Data Driving



27.3.2 DMA Engines

There are two DMA engines in each TDM engine: one to receive data and write it to main memory and one to read data from main memory for transmission. Both DMA engines use a common memory layout for the memory regions written to and read from.

27.3.2.1 Transmit/Receive Memory Regions

The format of the memory regions used for transmitting and receiving data is optimized to best use the CN71XX L2 cache. To prevent large portions of the L2 cache from being swept with each frame, eight frames worth of data for a single channel are stored per 64-bit word in memory. A group of 64-bit words comprising all timeslots in a frame contains all data to be transmitted or received for eight complete frames, and is termed a superframe.

Cavium Confidential For Bowen Yang ALU 02/27/2014

The PCM*n*_TXSTART register points to the start of the memory-transmit region, and PCM*n*_TXCNT specifies the number of superframes in the memory-transmit region.

For example, a four-superframe memory region to support transmission of three timeslots (timeslots 5, 9, and 12) would be laid out in memory as shown in Table 27–4, with the following parameters set:

PCMn_TXCNT = 0x4
PCMn_DMA_CFG[TXSLOTS] = 3
PCMn_TXMSK0<5, 9, 12> = 1

Table 27–4 Superframe Memory-Region Example

Address	Slot	Data							
		[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
PCM*n*_TXSTART + 0	TS5	TS5/Fr0	TS5/Fr1	TS5/Fr2	TS5/Fr3	TS5/Fr4	TS5/Fr5	TS5/Fr6	TS5/Fr7
PCM*n*_TXSTART + 8	TS9	TS9/Fr0	TS9/Fr1	TS9/Fr2	TS9/Fr3	TS9/Fr4	TS9/Fr5	TS9/Fr6	TS9/Fr7
PCM*n*_TXSTART + 16	TS12	TS12/Fr0	TS12/Fr1	TS12/Fr2	TS12/Fr3	TS12/Fr4	TS12/Fr5	TS12/Fr6	TS12/Fr7
PCM*n*_TXSTART + 24	TS5	TS5/Fr8	TS5/Fr9	TS5/Fr10	TS5/Fr11	TS5/Fr12	TS5/Fr13	TS5/Fr14	TS5/Fr15
PCM*n*_TXSTART + 32	TS9	TS9/Fr8	TS9/Fr9	TS9/Fr10	TS9/Fr11	TS9/Fr12	TS9/Fr13	TS9/Fr14	TS9/Fr15
PCM*n*_TXSTART + 40	TS12	TS12/Fr8	TS12/Fr9	TS12/Fr10	TS12/Fr11	TS12/Fr12	TS12/Fr13	TS12/Fr14	TS12/Fr15
PCM*n*_TXSTART + 48	TS5	TS5/Fr16	TS5/Fr17	TS5/Fr18	TS5/Fr19	TS5/Fr20	TS5/Fr21	TS5/Fr22	TS5/Fr23
PCM*n*_TXSTART + 56	TS9	TS9/Fr16	TS9/Fr17	TS9/Fr18	TS9/Fr19	TS9/Fr20	TS9/Fr21	TS9/Fr22	TS9/Fr23
PCM*n*_TXSTART + 64	TS12	TS12/Fr16	TS12/Fr17	TS12/Fr18	TS12/Fr19	TS12/Fr20	TS12/Fr21	TS12/Fr22	TS12/Fr23
PCM*n*_TXSTART + 72	TS5	TS5/Fr24	TS5/Fr25	TS5/Fr26	TS5/Fr27	TS5/Fr28	TS5/Fr29	TS5/Fr30	TS5/Fr31
PCM*n*_TXSTART + 80	TS9	TS9/Fr24	TS9/Fr25	TS9/Fr26	TS9/Fr27	TS9/Fr28	TS9/Fr29	TS9/Fr30	TS9/Fr31
PCM*n*_TXSTART + 88	TS12	TS12/Fr24	TS12/Fr25	TS12/Fr26	TS12/Fr27	TS12/Fr28	TS12/Fr29	TS12/Fr30	TS12/Fr31

PCM*n*_TXSTART must point to an eight-byte-aligned physical address. The memory-region size can be very large if required: PCM*n*_TXCNT can range up to 65535, with a maximum of 512 slots per frame; the maximum memory region could be nearly 32MB, holding data for 524,280 frames.

When the BCLK/FSYNC pair specifies that an extra frame bit is in the frame (i.e. PCM_CLK*n*_CFG[EXTRABIT] is set to 1), the value for that bit is stored as the first channel of a frame. The bit resides in either the MSB or the LSB of the data byte, depending on the setting of PCM*n*_TDM_CFG[LSBFIRST] and whether it is a transmission or reception. The bit placement is shown in Table 27–5.

Table 27–5 LSB/MSB Bit Placement

	PCM*n*_TDM_CFG[LSBFIRST]	
	= 0	= 1
Transmission	MSB	LSB
Reception	LSB	MSB

The memory receive region is handled in the same manner using the PCM*n*_RXSTART and PCM*n*_RXCNT CSRs.

Cavium Confidential Bowen Yang 02/7/2014

27.3.2.2 Transmit DMA Engine

The transmit DMA engine is configured via the PCM*n*_DMA_CFG, PCM*n*_TXSTART, and PCM*n*_TXCNT registers.

- The PCM*n*_TXSTART, PCM*n*_TXCNT registers and PCM*n*_DMA_CFG[TXSLOTS] field describe the transmit-memory region (see Section 27.3.2.1 for complete details).

  The PCM*n*_DMA_CFG[TXSLOTS] field tells the DMA engine the number of timeslots that are actually transmitted during each frame. This can range from all to none of the total number of slots in a frame. This number also is the number of 64-bit words in a superframe.

- The DMA engine's data requests are controlled via the PCM*n*_DMA_CFG[THRESH] and PCM*n*_DMA_CFG[FETCHSIZ] fields.

  The transmit DMA engine has a 16-byte FIFO to hold data waiting to be sent. When the number of bytes remaining in the FIFO is less than or equal to PCM*n*_DMA_CFG[THRESH], the DMA engine requests up to PCM*n*_DMA_CFG[FETCHSIZ] + 1 (the real fetch size) 64-bit words and extracts the bytes associated with the needed frame. Because of this behavior it is required that:

  $$\text{(threshold setting)} + \text{(the real fetch size)} \leq 16$$

  Otherwise the 16-byte FIFO will overflow.

The transmit DMA engine does not fetch across a frame boundary. In the case where there are fewer bytes remaining in the frame than the real fetch size, the DMA engine instead fetches all the remaining bytes in the frame and reevaluates the FIFO status upon receiving those bytes. Each transmit DMA engine can have only one request to main memory at a time.

There are two interrupts available to let the core know that the transmit memory region needs attention.

- The first is the TXRD interrupt, which signals an interrupt after PCM*n*_DMA_CFG[TXRD] full frames (see Table 27–4) have been read by the transmit DMA engine.
- The second option is the TXWRAP interrupt. This interrupt signals to the core when the transmit DMA engine has finished reading the last superframe of the transmit-memory region (i.e. it is about to start reading the first superframe again – it has wrapped to the beginning of the transmit-memory region).

One or both of these interrupts can be disabled if desired. If both are disabled the core will need to poll the PCM*n*_INT_SUM register to determine when the transmit memory region needs attention.

In addition, a TXEMPTY interrupt occurs if the transmit FIFO is empty when the TDM engine needs another byte to transmit. Note that this is an error. Operation is undefined after a TXEMPTY condition is encountered.

Cavium Confidential For Bowen Yang ALI 02-7-2014

27.3.2.3 Receive DMA Engine

The receive DMA engine is configured via the PCM*n*_DMA_CFG, PCM*n*_RXSTART, and PCM*n*_RXCNT registers.

- The PCM*n*_RXSTART, PCM*n*_RXCNT registers and PCM*n*_DMA_CFG[RXSLOTS] field describe the transmit-memory region (see Section 27.3.2.1 for complete details).

  The PCM*n*_DMA_CFG[RXSLOTS] field tells the DMA engine the number of timeslots that are actually received during each frame. This can range from all to none of the total number of slots in a frame. This number also is the number of 64-bit words in a superframe.

When a data byte is received by the TDM engine, it is immediately sent to the receive DMA engine, which writes each byte to main memory as soon as is practical. No write merging is done.

There are two interrupts available to let the CPU know that the receive memory region needs attention.

- The first option is the RXST interrupt, which signals an interrupt after PCM*n*_DMA_CFG[RXST] full frames have been written by the receive DMA engine.
- The second option is the RXWRAP interrupt. This interrupt signals to the core when the receive DMA engine has finished writing the last superframe of the receive-memory region (i.e. it is about to start writing the first superframe again – it has wrapped to the beginning of the receive-memory region).

These interrupts are not signaled until confirmation of the write to main memory has been received. No other interlock is required to assure the core that it will be able to see the data.

One or both of these interrupts can be disabled if desired. If both are disabled the CPU needs to poll the PCM*n*_INT_SUM register to determine when the receive-memory region needs attention.

In addition, an RXOVF interrupt occurs if the previous data byte has not been sent to main memory when the TDM engine provides another byte to send. This is an error condition indicating receive data has been lost.

Cavium Confidential For Bowen Yang At U 02/27/2017

27.4 Initialization Sequence

Perform the following steps in order to initialize the PCM/TDM interface. To determine the best values, refer to Sections 27.2 and 27.3.

1. Initialize the clock generation/reception by writing the appropriate values to PCM_CLK*n*_CFG and PCM_CLK*n*_GEN registers.
2. Setup transmission and reception parameters for each TDM engine with the following steps:
   - write the appropriate value to PCM*n*_DMA_CFG.
   - disable the FSYNCEXTRA and FSYNCMISSED interrupts by writing PCM*n*_INT_ENA[FSYNCEXTRA, FSYNCMISSED] = 0.
   - configure memory regions by writing the appropriate values to:
     PCM*n*_TXSTART[ADDR]
     PCM*n*_TXCNT[CNT]
     PCM*n*_TXMSK(0..7)[MASK]
     PCM*n*_RXSTART[ADDR]
     PCM*n*_RXCNT[CNT]
     PCM*n*_RXMSK(0..7)[MASK]
   - write PCM*n*_TDM_CFG[ENABLE] = 0.
3. Wait for clock reception to stabilize (i.e. read PCM_CLK*n*_CFG until [FSYNCGOOD] = 1).
4. Clear the FSYNCEXTRA and FSYNCMISSED interrupts by writing PCM*n*_INT_SUM[FSYNCEXTRA, FSYNCMISSED] = 1.
5. Enable the desired interrupts by writing 1 to the appropriate fields in PCM*n*_INT_ENA.
6. Write PCM*n*_TDM_CFG[ENABLE] = 1 to begin transmission and reception of PCM data.

Cavium Confidential For Bowen Yang ALU 02/27/2014

27.5 TDM/PCM Registers

The TDM/PCM registers are listed in Table 27–6.

Table 27–6 PCM/TDM Registers

Register	Address[1]	CSR Type[2]	Detailed Description
PCM_CLK(0,,1)_CFG	0x0001070000010000 + α*0x4000	NCB	See page 1267.
PCM_CLK(0..3)_GEN	0x0001070000010008 + α*0x4000	NCB	See page 1268.
PCM(0..3)_TDM_CFG	0x0001070000010010 + α*0x4000	NCB	See page 1268.
PCM(0..3)_DMA_CFG	0x0001070000010018 + α*0x4000	NCB	See page 1269.
PCM(0..3)_INT_ENA	0x0001070000010020 + α*0x4000	NCB	See page 1269.
PCM(0..3)_INT_SUM	0x0001070000010028 + α*0x4000	NCB	See page 1270.
PCM(0..3)_TDM_DBG	0x0001070000010030 + α*0x4000	NCB	See page 1270.
PCM_CLK(0..1)_DBG	0x0001070000010038 + α*0x4000	NCB	See page 1270.
PCM(0..3)_TXSTART	0x0001070000010040 + α*0x4000	NCB	See page 1270
PCM(0..3)_TXCNT	0x0001070000010048 + α*0x4000	NCB	See page 1270
PCM(0..3)_TXADDR	0x0001070000010050 + α*0x4000	NCB	See page 1271
PCM(0..3)_RXSTART	0x0001070000010058 + α*0x4000	NCB	See page 1271
PCM(0..3)_RXCNT	0x0001070000010060 + α*0x4000	NCB	See page 1271
PCM(0..3)_RXADDR	0x0001070000010068 + α*0x4000	NCB	See page 1271
PCM(0..3)_TXMSK(0..7)	0x0001070000010080 + α*0x4000 + β*0x8	NCB	See page 1272
PCM(0..3)_RXMSK(0..7)	0x00010700000100C0 + α*0x4000 + β*0x8	NCB	See page 1272

1. For registers with multiple versions, α represents the values within the specified range, β the second range and χ the third range.
2. NCB-type registers are accessed directly across the I/O Interconnect.

Cavium Confidential DO
Bowen Yang
ALU
02/27/2014

PCM Clock Configuration Registers
PCM_CLK(0,,1)_CFG

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63>	FSYNCGOOD	RO/H	0	1	FSYNC status. When set, the last frame had a correctly positioned FSYNC pulse. When clear, either no FSYNC pulse or an extra FSYNC pulse was seen on most recent frame. **NOTE:** This is intended for startup. The FSYNCEXTRA and FSYNCMISSING interrupts are intended for detecting loss of SYNC during normal operation.
<62:48>	—	RAZ	—	—	Reserved
<47:32>	FSYNCSAMP	R/W	0x0	—	Number of SCLK (coprocessor clock) cycles from the appropriate internal BCLK edge to sample FSYNC. **NOTE:** Used initially to synchronize to the start of a frame, later used to check for FSYNC errors.
<31:26>	—	RAZ	—	—	Reserved
<25:21>	FSYNCLEN	R/W	0x0	0x2	Number of ½ BCLK cycles FSYNC is asserted for. **NOTE:** Only used when PCM_CLK(0..3)_GEN[N] ≠ 0
<20:16>	FSYNCLOC	R/W	0x0	0x0	FSYNC location, in ½ BCLK cycles before timeslot 0, bit 0. **NOTE:** Also used to detect framing errors and therefore must have a correct value even if PCM_CLK(0..3)_GEN[N] = 0
<15:6>	NUMSLOTS	R/W	0x0	—	Number of eight-bit slots in a frame. **NOTE:** This, along with EXTRABIT and f(BCLK) determines FSYNC frequency when PCM_CLK(0..3)_GEN[N] ≠ 0 **NOTE:** Also used to detect framing errors and therefore must have a correct value even if PCM_CLK(0..3)_GEN[N]=0
<5>	EXTRABIT	R/W	0	0	Extra frame bit. When set, add one extra bit time for frame bit. When clear, no frame bit. **NOTE:** If PCM_CLK(0..3)_GEN[N] ≠ 0, FSYNC is delayed one extra bit time. **NOTE:** Also used to detect framing errors and therefore must have a correct value even if PCM_CLK(0..3)_GEN[N] = 0 **NOTE:** The extra bit comes from the LSB/MSB of the first byte of the frame in the transmit memory region.
<4:3>	BITLEN	R/W	0x0	0x0	Number of BCLK cycles in a bit time. 0 = 1 BCLK cycle 1 = 2 BCLK cycles 2 = 4 BCLK cycles 3 = undefined
<2>	BCLKPOL	R/W	0	0	BCLK polarity. When set, BCLK falling edge is start of bit time. When clear, BCLK rising edge is start of bit time. **NOTE:** Also used to detect framing errors and therefore must have a correct value even if PCM_CLK(0..3)_GEN[N]=0
<1>	FSYNCPOL	R/W	0	0	FSYNC polarity. When set, FSYNC idles high, asserts low. When clear, FSYNC idles low, asserts high. **NOTE:** Also used to detect framing errors and therefore must have a correct value even if PCM_CLK(0..3)_GEN[N] = 0.
<0>	ENA	R/W	0	0	Enable. When set, clock-receiving logic is looking for SYNC. When clear, clock-receiving logic is doing nothing,

Cavium Confidential For Bo Shi Yang 5/27/2014

PCM Clock Registers
PCM_CLK(0..3)_GEN

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:48>	DELTASAMP	R/W	0x0	0x0	Signed number of SCLK (coprocessor clock) cycles to move sampled BCLK edge. **NOTE:** The complete number of SCLK cycles to move is: NUMSAMP + 2 + 1 + DELTASAMP NUMSAMP: to compensate for sampling delay + 2: to compensate for dual-rank synchronizer + 1: for uncertainty + DELTASAMP: for debugging
<47:32>	NUMSAMP	R/W	0x0	—	Number of SCLK (coprocessor clock) samples to detect BCLK change when receiving clock.
<31:0>	N	R/W	0x0	—	Determines BCLK frequency [f(BCLK)] when generating clock. $f(BCLK) = f(SCLK) \times \frac{N}{2^{32}} \rightarrow N = \frac{f(BCLK)}{f(SCLK)} \times 2^{32}$ **NOTE:** Writing N = 0 stops the clock generator, and causes BCLK and FSYNC to be received.

PCM TDM Configuration Registers
PCM(0..3)_TDM_CFG

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:48>	DRVTIM	R/W	0x0	—	Drive time. The number of SCLK (coprocessor clock) cycles from start of bit time to stop driving last bit of timeslot (if not driving next timeslot).
<47:32>	SAMPPT	R/W	0x0	—	Sample time. The number of SCLK cycles from start of bit time to sample data bit.
<31:3>	—	RAZ	—	—	Reserved
<2>	LSBFIRST	R/W	0	0	LSB first. When this bit is set to 1, shift/receive LSB first. When it is clear, shift/receive MSB first.
<1>	USECLK1	R/W	0	0	Use CLK1. When this bit is set to 1, this TDM engine is based on BCLK/FSYNC1. When it is clear, this TDM engine is based on BCLK/FSYNC0.
<0>	ENABLE	R/W	0	0	TDM engine enable. When this bit is set to 1, TDM engine is enabled, otherwise the pins are GPIO pins. **NOTE:** When TDM is disabled by detection of an FSYNC error, all transmission and reception is halted. In addition, PCMn_TX/RXADDR are updated to point to the position at which the error was detected.

Cavium Confidential For Bowen Yang ALL 10/27/2014

PCM DMA Configuration Registers
PCM(0..3)_DMA_CFG

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63>	RDPEND	RO/H	0	0	Read pending. When set, L2C read responses are outstanding. When clear, no L2C read responses pending. **NOTE:** When restarting after stopping a running TDM engine, software must wait for RDPEND to read 0 before writing PCM*n*_TDM_CFG[ENABLE] to a 1.
<62:54>	—	RAZ	—	—	Reserved
<53:44>	RXSLOTS	R/W	0x0	—	Number of eight-bit slots to receive per frame (i.e. number of slots in a receive superframe).
<43:42>	—	RAZ	—	—	Reserved
<41:32>	TXSLOTS	R/W	0x0	—	Number of eight-bit slots to transmit per frame (i.e. number of slots in a transmit superframe).
<31:30>	—	RAZ	—	—	Reserved
<29:20>	RXST	R/W	0x0	0x1	Number of frame writes for interrupt.
<19>	—	RAZ	—	—	Reserved
<18>	USELDT	R/W	0	0	Use LDT command. When set, use LDT command to read from L2C. When clear, use LDI command to read from L2C.
<17:8>	TXRD	R/W	0x0	0x1	Number of frame reads for interrupt.
<7:4>	FETCHSIZ	R/W	0x0	0x7	FETCHSIZ+1 timeslots are read when threshold is reached.
<3:0>	THRESH	R/W	0x0	0x8	Threshold. If number of bytes remaining in the DMA FIFO is ≤ THRESH, initiate a fetch of timeslot data from the transmit memory region. **NOTE:** There are only 16 bytes of buffer for each engine so the settings for FETCHSIZ and THRESH must be such that the buffer is not overrun: THRESH + min(FETCHSIZ + 1,TXSLOTS) must be ≤ 16.

PCM Interrupt-Enable Registers
PCM(0..3)_INT_ENA

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7>	RXOVF	R/W	0	1	RX overflow. Enable interrupt if RX byte overflows.
<6>	TXEMPTY	R/W	0	1	TX empty. Enable interrupt on TX byte empty.
<5>	TXRD	R/W	0	1	TX read. Enable DMA engine frame-read interrupts.
<4>	TXWRAP	R/W	0	1	TX wrap. Enable TX region-wrap interrupts.
<3>	RXST	R/W	0	1	RX store. Enable DMA engine frame-store interrupts.
<2>	RXWRAP	R/W	0	1	RX wrap. Enable RX region-wrap interrupts.
<1>	FSYNCEXTRA	R/W	0	1	Enable FSYNC extra interrupts. **NOTE:** FSYNCEXTRA errors are defined as an FSYNC found in the wrong spot of a frame given the programming of PCM_CLK(0,,1)_CFG[NUMSLOTS] and PCM_CLK(0,,1)_CFG[EXTRABIT].
<0>	FSYNCMISSED	R/W	0	1	Enable FSYNC missed interrupts. **NOTE:** FSYNCMISSED errors are defined as an FSYNC missing from the correct spot in a frame given the programming of PCM_CLK(0,,1)_CFG[NUMSLOTS] and PCM_CLK(0,,1)_CFG[EXTRABIT].

Cavium Confidential For Bowen Yang ALU 2/27/2014

PCM Interrupt Registers
PCM(0..3)_INT_SUM

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7>	RXOVF	R/W1C/H	0	0	RX byte overflowed.
<6>	TXEMPTY	R/W1C/H	0	0	TX byte was empty when sampled.
<5>	TXRD	R/W1C/H	0	0	DMA engine frame-read interrupt occurred.
<4>	TXWRAP	R/W1C/H	0	0	TX region wrap-interrupt occurred.
<3>	RXST	R/W1C/H	0	0	DMA engine frame-store interrupt occurred.
<2>	RXWRAP	R/W1C/H	0	0	RX region-wrap interrupt occurred.
<1>	FSYNCEXTRA	R/W1C/H	0	0	FSYNC extra interrupt occurred.
<0>	FSYNCMISSED	R/W1C/H	0	0	FSYNC missed interrupt occurred.

PCM TDM Debug Registers
PCM(0..3)_TDM_DBG

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	DEBUGINFO	RO/H	—	—	Miscellaneous debug information.

PCM Clock Debug Registers
PCM_CLK(0..1)_DBG

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	DEBUGINFO	RO/H	—	—	Miscellaneous debug information.

PCM Transmit-Memory Start Registers
PCM(0..3)_TXSTART

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:36>	—	RAZ	—	—	Reserved
<35:3>	ADDR	R/W	—	—	Starting physical address for the transmit memory region.
<2:0>	—	RAZ	—	—	Reserved

PCM Transmit Count Registers
PCM(0..3)_TXCNT

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	CNT	R/W	—	—	Number of superframes in transmit memory region.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PCM Transmit Address Registers
PCM(0..3)_TXADDR

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:36>	—	RAZ	—	—	Reserved
<35:3>	ADDR	R/W/H	—	—	Physical address of the next read from the transmit memory region.
<2:0>	FRAM	R/W/H	—	—	Frame offset. **NOTE:** This is used to extract the correct byte from each 64-bit word read from the transmit memory region.

PCM Receive-Memory Start Registers
PCM(0..3)_RXSTART

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:36>	—	RAZ	—	—	Reserved
<35:3>	ADDR	R/W	—	—	Starting physical address for the receive memory region.
<2:0>	—	RAZ	—	—	Reserved

PCM Receive Count Registers
PCM(0..3)_RXCNT

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	CNT	R/W	—	—	Number of superframes in receive memory region.

PCM Receive Address Registers
PCM(0..3)_RXADDR

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:36>	—	RAZ	—	—	Reserved
<35:0>	ADDR	R/W/H	—	—	Physical address of the next write operation to the receive memory region.

Cavium Confidential For Bowen Yang ALU 02/27/2014

PCM Transmit Mask 0–7 Registers
PCM(0..3)_TXMSK(0..7)

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	MASK	R/W	—	—	Transmit mask bits. (1 = transmit, 0 = don't transmit) TXMSK0 covers slots 63 to 0 TXMSK1 covers slots 127 to 64 TXMSK2 covers slots 191 to 128 TXMSK3 covers slots 255 to 192 TXMSK4 covers slots 319 to 256 TXMSK5 covers slots 383 to 320 TXMSK6 covers slots 447 to 384 TXMSK7 covers slots 511 to 448

PCM Receive Mask 0–7 Registers
PCM(0..3)_RXMSK(0..7)

See Table 27–6 for address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:0>	MASK	R/W	—	—	Receive mask bits. (1 = receive, 0 = don't receive). RXMSK0 covers slots 63 to 0 RXMSK1 covers slots 127 to 64 RXMSK2 covers slots 191 to 128 RXMSK3 covers slots 255 to 192 RXMSK4 covers slots 319 to 256 RXMSK5 covers slots 383 to 320 RXMSK6 covers slots 447 to 384 RXMSK7 covers slots 511 to 448

Cavium Confidential For Bowen Yang ALU 02/27/2014

Chapter 28

Boot Bus

This section contains the following subjects:

- Overview
- Addresses
- Address Matching and Regions
- Reset Configuration and Booting
- Region Timing
- Request Queuing
- Connections
- Operations
- Boot-Bus Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

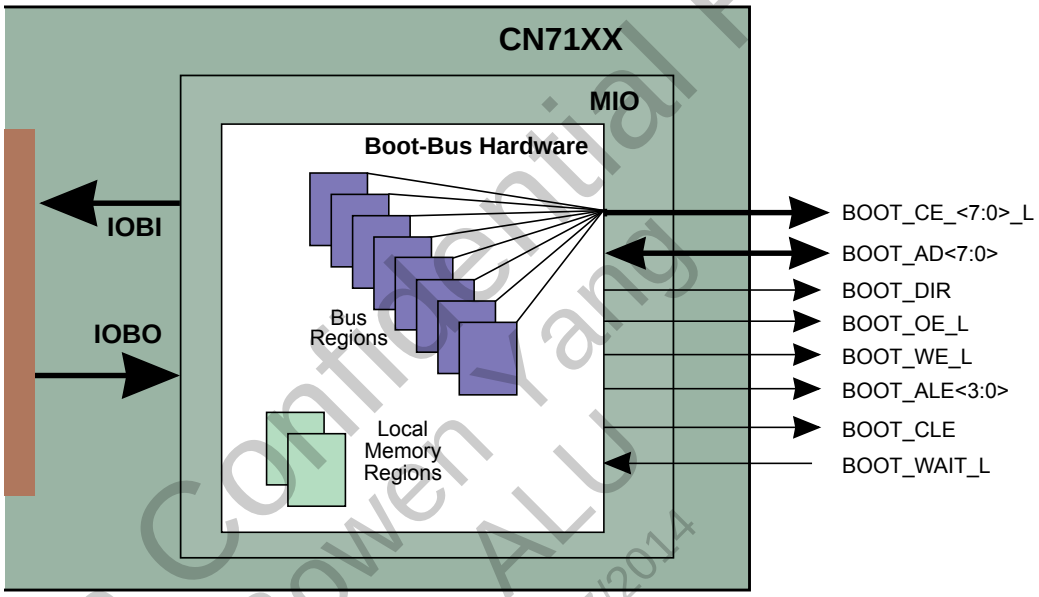
The CN71XX boot-bus hardware provides a general-purpose bus that typically attaches to nonvolatile devices, like flash memories (NOR, NAND, and eMMC), and ROMs, and can service the initial boot address on CN71XX. The bus protocol is flexible, so it can interface to many other types of devices as well. This chapter describes the interface for all devices except directly attached eMMC flash memories. For a description of the interface to the directly attached eMMC and NAND flash memories, see Chapter 29 and Chapter 30.

The big-endian boot bus contains:

- eight chip-selects
- eight address/data lines
- eight-bit data width

Figure 28–1 shows the boot-bus hardware.

Figure 28–1 Boot-Bus Hardware



Cavium Confidential For Bowen Yang ALU 02/27/2014

The boot-bus hardware can configure up to eight regions that correspond to the eight chip-select signals. Each region/chip select has different bus configuration and timing parameters. When CN71XX self-boots (i.e. when the external pin REMOTE_BOOT = 0), the boot bus must provide the initial instructions to boot the chip on chip-select 0.

The boot-bus hardware also contains two local, 128-byte memory regions. These memory regions must be used to service core-boot and debug-exception vector references when there is no device attached to the boot bus that will service them. The local memory regions can also accelerate read operations to commonly accessed areas of the boot bus.

A remote PCIe host can also use the boot bus. The boot-bus hardware services all PCIe expansion-ROM references, as well as all references to the boot-bus physical addresses via the PCIe windowed addressing mechanism. (The remote PCIe host

accesses the SLI_WIN_WR_ADDR, SLI_WIN_RD_ADDR, SLI_WIN_WR_DATA, SLI_WIN_WR_MASK, and SLI_WIN_RD_DATA registers to use the PCI windowed-addressing mechanism.)

28.1 Addresses

The boot bus consumes a 4GB portion of the CN71XX physical-address (PA) space range from 0x1 0000 0000 0000 to 0x1 0000 FFFF FFFF.

NOTE: CN71XX cores also convert all of their physical addresses in the range 0x0 0000 1000 0000 to 0x0 0000 1FFF FFFF to CN71XX boot-bus physical addresses in the range 0x1 0000 1000 0000 to 0x1 0000 1FFF FFFF.

The boot-bus physical addresses shown in Table 28–1 are particularly important, as they are exception vectors for the cores.

Table 28–1 Exception Vectors

Vector Address (PA)	Comment
0x1 0000 1FC0 0000	A core vectors to this physical address after all resets, soft resets, or nonmaskable interrupt (NMI) exceptions.
0x1 0000 1FC0 0480	If the EJTAG TAP register field ECR[ProbEn] is not set, a core vectors to this physical address after all debug exceptions (i.e. whenever it enters debug mode).

Other core exceptions can also vector to boot-bus addresses in certain situations. The other exception vectors can be avoided by configuration if desired (by clearing the core CP0 register bit Status[BEV]), but the vector addresses in Table 28–1 cannot be avoided by core configuration.

If a system does not have any devices attached to the boot bus, the two local memory regions in the boot-bus hardware must service the (instruction fetch) read operations that immediately follow the exceptions listed in Table 28–1. In such a system, CN71XX cannot self-boot (i.e. the external REMOTE_BOOT pin must be set to 1). Each local memory region can hold up to 32 instructions at these vector addresses. If more instructions are needed to service these exceptions in these systems, the exception handler must jump to L2/DRAM within these first 32 instructions.

The boot-bus address range shown in Table 28–2 is also of particular interest regarding PCIe expansion ROM references.

Table 28–2 PCIe Expansion-ROM Address Range

Address (PA) Range	Comment
0x1 0000 1FC1 0000 to 0x1 0000 1FC1 FFFF	This is the 64KB physical-address range used for PCIe expansion ROM references. The PCIe hardware converts expansion-ROM read operations by a remote PCIe host into CN71XX boot-bus accesses in this range.

Cavium Confidential For Bowen Yang 02/27/2014

28.2 Address Matching and Regions

The boot-bus hardware only services boot-bus references by matching their PA with programmed regions. For read operations, the hardware first checks for a match with the local memory regions. If the read PA matches one of the two local memory regions, the hardware services the read with the data in the memory inside the unit. Otherwise (i.e. for any write operations or for a read operation that does not hit in a local memory region), the boot-bus hardware checks for a match of the PA with the bus regions. On a match, the hardware uses that region (and its corresponding chip select and timing parameters) to complete the reference.

Each of the two local memory regions are enabled when the corresponding MIO_BOOT_LOC_CFG*n*[EN] is set. Each local memory region is 128 bytes when enabled, and matches when PA<31:7> of the read PA equals MIO_BOOT_LOC_CFG*n*[BASE].

The storage for each local memory region is held in the boot-bus logic, and must be filled by software before the local memory region is enabled. Software can fill local memory region 0 by performing the following steps:

1. Set MIO_BOOT_LOC_ADR to 0x0 (set to 0x80 for local memory region 1).
2. Write MIO_BOOT_LOC_DAT 16 times with eight bytes each time.

Each of the eight bus regions are enabled when the corresponding MIO_BOOT_REG_CFG*n*[EN] is set. The size in bytes of each enabled bus region is:

$$(\text{MIO_BOOT_REG_CFG}n\text{[SIZE]} + 1) \times 2^{16}$$

A boot-bus read or write operation matches an enabled bus region when both of the following conditions are met:

$$\text{PA<31:16>} \geq \text{MIO_BOOT_REG_CFG}n\text{[BASE]}, \quad \text{AND}$$

$$\text{PA<31:16>} \leq \text{MIO_BOOT_REG_CFG}n\text{[BASE]} + \text{MIO_BOOT_REG_CFG}n\text{[SIZE]}$$

When a boot-bus reference matches an enabled bus region, the chip-select for the region asserts to complete the reference. The boot-bus address used to access the external device is:

$$\text{Addr<27:0>} = \text{PA<31:0>} - (\text{MIO_BOOT_REG_CFG}n\text{[BASE]} \ll 16)$$

Software must ensure the following conditions:

- Software must not allow more than one local memory region to match any PA.
- Software must not allow more than one bus region to match any PA.

It is legal for a PA to match both a local memory region and a bus region, though. When this happens, a read takes the data from the local memory region.

Cavium Confidential For Bowen Yang ALD 02/21/2014

28.3 Reset Configuration and Booting

The reset configuration for the bus regions must be used when CN71XX self-boots (i.e. when REMOTE_BOOT = 0), as the core-reset vector is used immediately out of reset in this case. After reset, bus region 0 is set up this way:

- it is the only valid region
- it is maximum size
- it maps CN71XX PA = 0x1 0000 1FC0 0000 to boot address = 0x0 on the device attached to chip-select 0.
- it operates in multiplexed mode.

This means that when CN71XX self-boots, the device attached to chip-select 0 must contain the boot code for the CN71XX starting at boot address = 0. When the CN71XX self-boots, core 0 executes this code immediately following chip-reset deassertion. The reset timing parameters for region 0 are set to conservative values to accommodate a wide range of devices. If desired, the timing parameters for region 0 can be changed for faster operation while booting.

The reset configuration for the bus regions typically must also be used when PCIe expansion-ROM references occur in the system, since PCIe expansion-ROM references from a remote host occur early in the PCIe-bus boot sequence. The boot-bus regions will typically not have changed from their reset values at the time that the read operations occur, so all PCIe expansion-ROM read operations map to the device attached to chip select 0.

- To fully support a PCIe expansion ROM via CN71XX, a device typically must be attached to chip-select 0, and must contain the expansion-ROM data starting at boot address = 0x10000 in the device.
- If a device is not physically attached to boot-bus chip-select 0, any PCIe expansion-ROM read operations return 0 (as do any other references to the region), and it appears that a PCIe expansion ROM is not present to remote PCIe host software.
- If a device is attached to chip-select 0 and PCIe expansion-ROM read operations to CN71XX occur, the remote PCIe-host configuration software still thinks that an expansion ROM is not present as long as read operations near physical address 0x1 0000 1FC1 0000 (i.e. boot address = ~0x10000 in the device) return 0s.

  When a system has a memory device attached to chip-select 0, but wants it to appear to a remote PCIe host that the PCIe expansion ROM is not present, the system must set the boot addresses = ~0x10000 to 0 on the device attached to chip-select 0.

Cavium Confidential For Bowen Yang ALU 24/27/2014

The pin straps that CN71XX uses for configuration are shown in Table 28–3.

Table 28–3 Boot-Bus Pin Straps

Pin Strap	Comment
BOOT_AD[7:0]	When using eMMC, BOOT_AD[7:0] is EMMC_DAT[7:0].
BOOT_ALE0 BOOT_CLE	External pull-up resistors that change the boot-bus-driver drive strength in the following manner: **ALE0 CLE Drive Strength** 0 0 full strength (default) 0 1 25Ω 1 0 50Ω 1 1 65Ω Refer to MIO_BOOT_PIN_DEFS[TERM].
BOOT_ALE[3:1]	Specifies the boot source: 0x0 = parallel NOR boot (default) 0x4 = SPI NOR boot (16-bit address) 0x1 = reserved 0x5 = SPI NOR boot (24-bit address) 0x2 = eMMC/SD boot 0x6 = SPI NOR boot (32-bit address) 0x3 = reserved 0x7 = reserved
BOOT_DIR	Enable/disable VRM: 1 = VRM disable, 0 = VRM enable

CN71XX samples these pin straps when its internal reset signal first deasserts following a cold reset.

NOTE: 32-bit MWDMA devices that are attached to CN71XX must not be driving any BOOT_AD* pins. These devices must asynchronously disable their drivers whenever BOOT_DIR is asserted high.

28.4 Region Timing

The mode and timing of each boot-bus region can be configured independently (using MIO_BOOT_REG_TIM*n*), so the characteristics of each device on the bus can be completely different. The different chip-selects insulate the different devices on the bus from each other. CN71XX is the default driver of the BOOT_DAT data bus.

Most individual devices only require a single set of modes and timing parameters. The boot bus has an OR mode, enabled by setting MIO_BOOT_REG_CFG*n*[OR], that makes CN71XX support two different modes/timings for a device, though the device need only connect to a single BOOT_CE_*n*_L wire. When MIO_BOOT_REG_CFG*n*[OR] is set, BOOT_CE_*n*_L asserts whenever either region *n* or region *n–1* matches.

- If region *n* matches, the mode and timing for region *n* are used.
- If region *n–1* matches, the mode and timing for region *n–1* are used.

In either case, BOOT_CE_*n*_L asserts. This can be useful for devices like the compact flash. The compact flash has both attribute and common memories, each with very different characteristics. MIO_BOOT_REG_CFG*n*[OR] is not needed if the different memories can be accessed at different times, however (with a MIO_BOOT_REG_TIM*n* reconfiguration between them).

Table 28–4 shows the configuration of the two mode bits that determine bus operation.

Table 28–4 Bus Operation

MIO_BOOT_REG_TIM*n*		
<63>[PAGEM]	<62>[WAITM]	Comment
0	0	Static-timed, page-mode disabled
0	1	Dynamic-timed (via BOOT_WAIT_L)
1	0	Static-timed, page-mode enabled
1	1	Illegal

Cavium Confidential For Bowen Yang ALL 02/27/2014

Table 28–5 describes the time intervals used in the timing diagrams that follow.

Table 28–5 Boot-Bus Timing Parameters

Parameter	Field	Description
T_{ADR}	MIO_BOOT_REG_TIM*n*[ADR]	The time between the deassertion of BOOT_ALE and the assertion of BOOT_CE_*n*_L in multiplexed mode.
T_{CE}	MIO_BOOT_REG_TIM*n*[CE]	The time that BOOT_CE_*n*_L is asserted before BOOT_OE_L (read operation) or BOOT_WE_L (write operation) asserts.
T_{OE}	{MIO_BOOT_REG_CFG*n*[OE_EXT], MIO_BOOT_REG_TIM*n*[OE]}	The time that BOOT_OE_L is asserted for a read operation.
T_{RH}	MIO_BOOT_REG_TIM*n*[RD_HLD]	The time that BOOT_CE_*n*_L remains asserted after BOOT_OE_L deasserts for a read operation.
T_{PAUSE}	MIO_BOOT_REG_TIM*n*[PAUSE]	The time that the boot address retains its value after BOOT_CE_*n*_L deasserts. Also affects the minimum idle time between bus references.
T_{WE}	{MIO_BOOT_REG_CFG*n*[WE_EXT], MIO_BOOT_REG_TIM*n*[WE]}	The time that BOOT_WE_L is asserted for a write operation.
T_{WH}	MIO_BOOT_REG_TIM*n*[WR_HLD]	The time that BOOT_CE_*n*_L remains asserted after BOOT_WE_L deasserts for a write operation.
T_{WAIT}	MIO_BOOT_REG_TIM*n*[WAIT]	The time that BOOT_WAIT_L must be deasserted before BOOT_OE_L (read operation) or BOOT_WE_L (write operation) deasserts.
T_{ALE}	MIO_BOOT_REG_TIM*n*[ALE]	The time that BOOT_ALE is asserted in multiplexed mode.

The timing configuration parameters, with the exception of MIO_BOOT_REG_TIM*n*[WAIT], specify the time interval in internal coprocessor-clock cycles:

time_interval = **(TM × config_parameter + 1)** coprocessor-clock cycles

where TM = MIO_BOOT_REG_CFG*n*[TIM_MULT]

The wait time after the deassertion of BOOT_WAIT_L is specified in internal coprocessor-clock cycles:

time_interval = **MIO_BOOT_REG_TIM*n*[WAIT]** coprocessor-clock cycles

Table 28–6 describes the fixed time intervals used in the timing diagrams.

Table 28–6 Boot-Bus Fixed-Timing Parameters

Parameter	Value	Description
T_{DS}	6 ns – (MIO_BOOT_REG_CFG*n*[RD_DLY] coprocessor-clock cycles)	Data setup time with respect to the rising edge of BOOT_OE_L or the transition of BOOT_AD<2:0> (in page mode).
T_{DH}	(MIO_BOOT_REG_CFG*n*[RD_DLY] coprocessor-clock cycles) –1 ns	Data hold time with respect to the rising edge of BOOT_OE_L or the transition of BOOT_AD<2:0> (in page mode).
T_{WS}	6 ns	Wait setup time with respect to the expiration of T_{OE} or T_{WE}.

Examples of timing sequences are shown in the following sections.

Cavium Confidential For Bowen Yang ALU 02/12/2014

28.4.1 Static-Timed Read Sequences

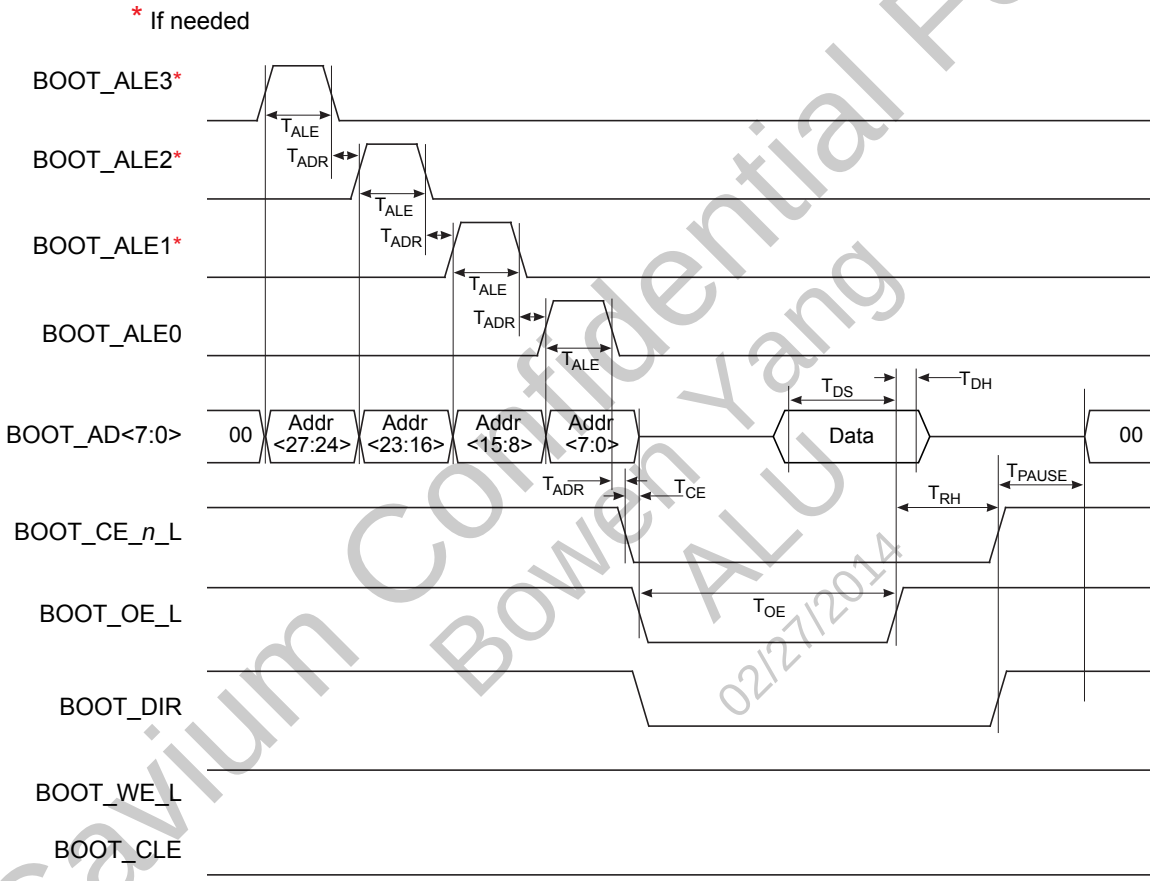
Figure 28–2 shows a boot-bus single-byte read transaction in eight-bit, multiplexed mode. All read operations to the device are single-byte when MIO_BOOT_REG_TIM*n*[PAGEM] = 0 and MIO_BOOT_REG_CFG*n*[WIDTH] = 0.

BOOT_ALE0 asserts and the boot address (BOOT_AD<7:0>) becomes valid first, then BOOT_ALE0 deasserts, then BOOT_CE_*n*_L asserts, then BOOT_OE_L asserts.

CN71XX stops driving the boot-address bits (BOOT_AD<7:0>) at the same time that BOOT_OE_L asserts. CN71XX samples the boot data as it deasserts BOOT_OE_L. After that, BOOT_CE_*n*_L deasserts, then the boot address returns to 0s. CN71XX drives the boot data to 0s after it deasserts BOOT_CE_*n*_L.

BOOT_DIR is asserted high when CN71XX is driving the boot bus. When BOOT_DIR is deasserted low, the device drives the bus.

Figure 28–2 Static-Timed Read Sequence



Cavium Confidential For Bowen Yang ALU 02/27/2014

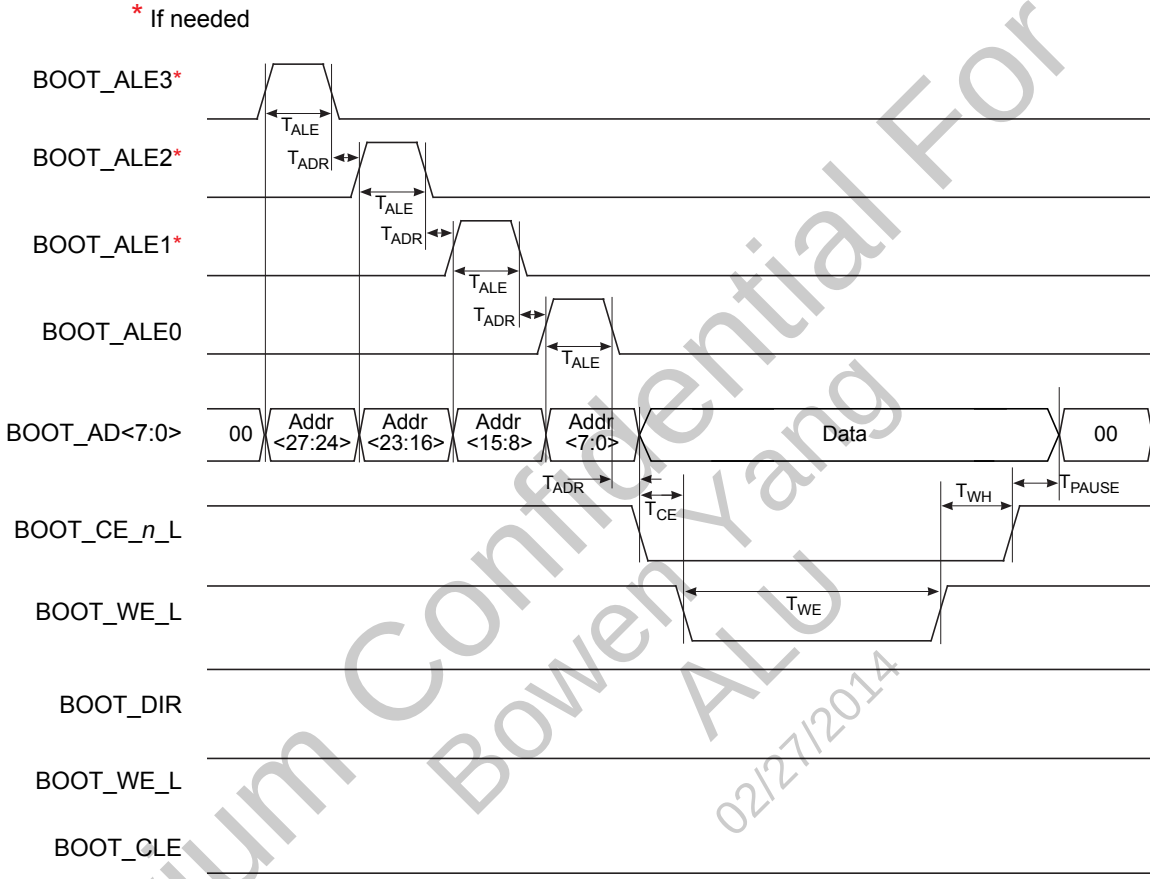
28.4.2 Static-Timed Write Sequences

Figure 28–3 shows a boot-bus write transaction in eight-bit, multiplexed mode. All boot-bus write transactions are single-byte when MIO_BOOT_REG_CFG*n*[WIDTH] = 0.

BOOT_ALE<3:0> asserts and the boot address (BOOT_AD<7:0>) becomes valid. Then BOOT_CE_*n*_L asserts and boot data (BOOT_AD<7:0>) becomes valid, and BOOT_WE_L asserts. After a time delay, BOOT_WE_L deasserts, BOOT_CE_*n*_L deasserts, and finally the boot address and boot data return to 0s.

BOOT_DIR is asserted high when CN71XX is driving the boot bus. For write transactions, CN71XX never stops driving the bus or BOOT_DIR.

Figure 28–3 Static-Timed Write Sequence



Cavium Confidential For BoWen Yang ALU 02/27/2014

28.4.3 Dynamic-Timed Sequences

Dynamic-timed read and write operations are performed in eight-bit, multiplexed mode. The following examples are shown:

- eight-bit, multiplexed read operation
- eight-bit, multiplexed write operation

28.4.3.1 Dynamic-Timed Read Sequence

Figure 28–4 shows a dynamic-timed read transaction (i.e. when MIO_BOOT_REG_TIM*n*[WAITM] = 1). This transaction is almost identical to the single-byte read transaction shown in Figure 28–2, except for the delay in the deassertion of BOOT_OE_L caused by the target device asserting BOOT_WAIT_L. Note that BOOT_WAIT_L must be asserted before T_{OE} would normally expire, otherwise the transaction will not be extended. After T_{OE} has expired and BOOT_WAIT_L has been deasserted for T_{WAIT} cycles, BOOT_OE_L deasserts and the transaction continues as a normal single byte read. Note that if BOOT_WAIT_L is still asserted 2^{15} cycles after T_{OE} would normally expire, the transaction completes as normal (though presumably with bad read data) and MIO_BOOT_ERR[WAIT_ERR] is set.

BOOT_DIR is asserted high when CN71XX is driving the boot bus. When BOOT_DIR is deasserted low, the device drives the bus.

Figure 28–4 Dynamic-Timed Read Sequence

TBD

28.4.3.2 Dynamic-Timed Write Sequence

Figure 28–5 shows a dynamic-timed write transaction (i.e. when MIO_BOOT_REG_TIM*n*[WAITM] = 1). This transaction is almost identical to the byte-write transaction shown in Figure 28–3, except for the delay in the deassertion of BOOT_WE_L caused by the target device asserting BOOT_WAIT_L. Note that BOOT_WAIT_L must be asserted before T_{WE} would normally expire, otherwise the transaction will not be extended. After T_{WE} has expired and BOOT_WAIT_L has been deasserted for T_{WAIT} cycles, BOOT_WE_L deasserts and the transaction continues as a normal byte write. Note that if BOOT_WAIT_L is still asserted 2^{15} cycles after T_{WE} would normally expire, the transaction completes as normal (though presumably the write will not complete in the target) and MIO_BOOT_ERR[WAIT_ERR] is set.

BOOT_DIR is asserted high when CN71XX is driving the boot bus. For write transactions, CN71XX never stops driving the bus or BOOT_DIR.

Figure 28–5 Dynamic-Timed Write Sequence

TBD

Cavium Confidential For Boven Yang 2017/7/4

28.5 Request Queuing

The boot-bus logic has a FIFO that holds load/store operations destined for the boot bus. When

FIFO size ≥ MIO_BOOT_THR[FIF_THR],

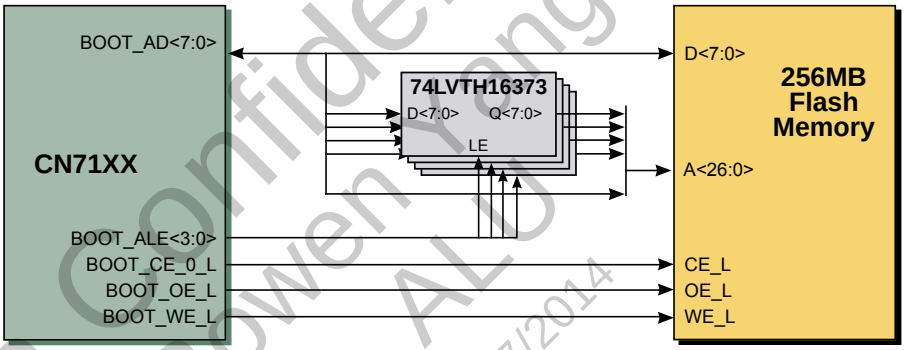
the boot-bus logic applies backpressure, and no more CSR references (destined anywhere in CN71XX) can complete. Each load/store operation takes one FIFO entry. It can take a long time for this queue to drain, depending on the timing of the devices on the bus and whether DMA operations interrupt the load/store operations. Overflow of this FIFO should preferably be avoided when fast response from CN71XX is required. Load operations cannot cause the FIFO to overflow, since each core has at most one load outstanding, but a single core can overflow the FIFO with IOBDMA or store operations.

MIO_BOOT_THR[FIF_CNT] contains the current FIFO size. Software can avoid the CN71XX performance effects of overflowing this FIFO by monitoring the current size of the FIFO after a series of IOBDMAs/store operations.

28.6 Connections

Figure 28–6 shows a sample connection from the boot bus to a 8-bit-wide, 256MB flash memory, using BOOT_CE_0_L.

Figure 28–6 Sample Boot-Bus Connection 1 (ALE, 8W)



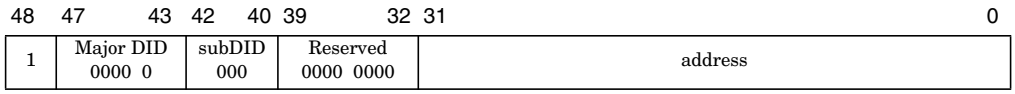
Cavium Confidential For Bowen Yang ALU 02/27/2014

28.7 Operations

The boot bus performs load, IOBDMA, and store operations, which are described in the following subsections.

28.7.1 Load Operations

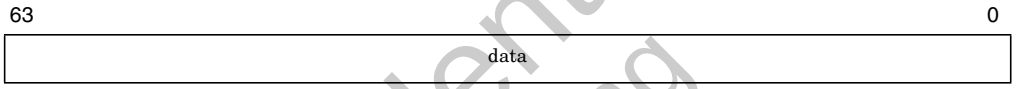
Load Address Field



- **address -** The address to be matched to local memory and bus regions.
  - for 64-bit load operations: address<2:0> must be 000.
  - for 32-bit load operations: address<1:0> must be 00.
  - for 16-bit load operations: address<0> must be 0.

Load Result Field

64-bit operation result

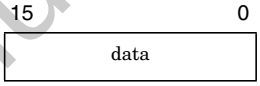


- **data -** data is the result of the boot-bus load, in big-endian format.

32-bit operation result

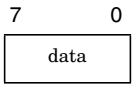
- **data -** data is the result of the boot-bus load operation, in big-endian format.

16-bit operation result



- **data -** data is the result of the boot-bus load operation, in big-endian format.

8-bit operation result

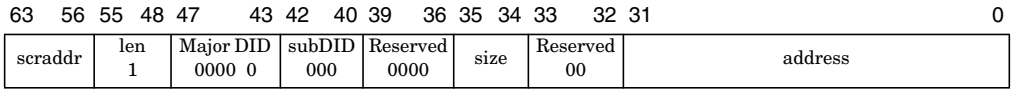


- **data -** data is the result of the boot-bus load operation.

Cavium Confidential For Bowen Yang ALU 02/27/2014

28.7.2 IOBDMA Operations

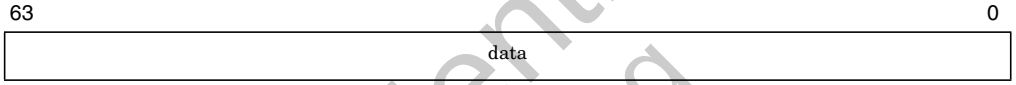
IOBDMA Address Field



- **scraddr -** Defined in Chapter 2.
- **len -** Must be 1. Defined in Chapter 2.
- **size** - Indicates the size of the operation
  00 = 8-byte operation
  01 = 4-byte operation
  10 = 2-byte operation
  11 = 1-byte operation
- **address -** The address to be matched to local memory and bus regions.
  - for 64-bit IOBDMA operations: address<2:0> must be 000.
  - for 32-bit IOBDMA operations: address<1:0> must be 00.
  - for 16-bit IOBDMA operations: address<0> must be 0.

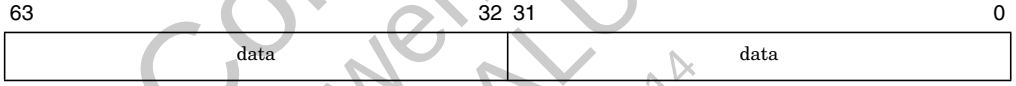
IOBDMA Result Field

8-byte IOBDMA operation result



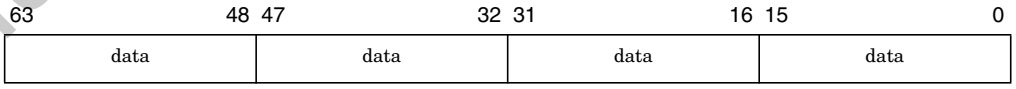
- **data -** data is the result of the boot-bus read operations totalling 8 bytes, in big-endian format.

4-byte IOBDMA operation result



- **data -** data is the result of the boot-bus read operations totalling 4 bytes, in big-endian format, duplicated in <63:32> and <31:0>.

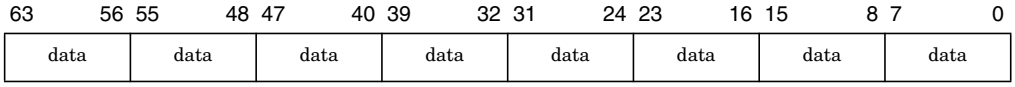
2-byte IOBDMA operation result



- **data -** data is the result of the boot-bus read operations totalling 2 bytes, in big-endian format, duplicated in <63:48>, <47:32>, <31:16>, and <15:0>.

Cavium Confidential For Bowen Yang ALU 02/27/2014

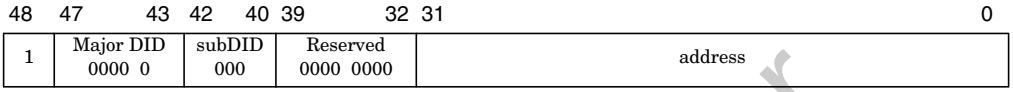
1-byte IOBDMA operation result



- **data -** data is the result of the boot-bus read operations totalling 8 bytes, in big-endian format, duplicated in all bytes.

28.7.3 Store Operations

Store Address Field



- **address -** The address to be matched to local memory and bus regions.
  - for 64-bit store operations: address<2:0> must be 000.
  - for 32-bit store operations: address<1:0> must be 00.
  - for 16-bit store operations: address<0> must be 0.

The boot-bus hardware writes the store data onto the boot bus in big-endian format.

Cavium Confidential For Bowen Yang ALU 02/27/2014

28.8 Boot-Bus Registers

The boot-bus registers are listed in Table 28–7.

Table 28–7 Boot-Bus Registers

Register	Address[1]	CSR Type[2]	Detailed Description
MIO_BOOT_REG_CFG0	0x0001180000000000	RSL	See page 1288
MIO_BOOT_REG_CFG(1..7)	0x0001180000000000+α*0x8	RSL	See page 1289
MIO_BOOT_REG_TIM0	0x0001180000000040	RSL	See page 1290
MIO_BOOT_REG_TIM(1..7)	0x0001180000000040+α*0x8	RSL	See page 1290
MIO_BOOT_LOC_CFG(0..1)	0x0001180000000080+α*0x8	RSL	See page 1291
MIO_BOOT_LOC_ADR	0x0001180000000090	RSL	See page 1291
MIO_BOOT_LOC_DAT	0x0001180000000098	RSL	See page 1291
MIO_BOOT_ERR	0x00011800000000A0	RSL	See page 1292
MIO_BOOT_INT	0x00011800000000A8	RSL	See page 1292
MIO_BOOT_THR	0x00011800000000B0	RSL	See page 1292
MIO_BOOT_COMP	0x00011800000000B8	RSL	See page 1293
MIO_BOOT_PIN_DEFS	0x00011800000000C0	RSL	See page 1293
MIO_GPIO_COMP	0x00011800000000C8	RSL	See page 1294
MIO_BOOT_CTL	0x00011800000000D0	RSL	See page 1295
MIO_BOOT_BIST_STAT	0x00011800000000F8	RSL	See page 1295

1. For registers with multiple versions, α represents the values within the specified range, β the second range, and χ the third range.
2. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO Boot Region 0 Configuration Register MIO_BOOT_REG_CFG0

This register contains region enable, region OR, region ALE/SAM mode, region width, region size, and region base address parameters. See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:44>	—	RAZ	—	—	Reserved
<43:42>	DMACK	R/W	0x0	—	Reserved.
<41:40>	TIM_MULT	R/W	0x0	—	Region 0 timing multiplier. Specifies the timing multiplier for a region. The timing multiplier applies to all timing parameters, except for WAIT and RD_DLY, which simply count coprocessor-clock cycles. TIM_MULT is encoded as follows: 0 = 4×, 1 = 1×, 2 = 2×, 3 = 8×.
<39:37>	RD_DLY	R/W	0x0	—	Region 0 read sample delay. Specifies the read sample delay in coprocessor-clock cycles for a region. For read operations, the data bus is normally sampled on the same coprocessor-clock edge that drives BOOT_OE_L to the inactive state (or the coprocessor-clock edge that toggles the lower address bits in page mode). This parameter can delay that sampling edge by up to seven coprocessor-clock cycles. **NOTE:** The number of coprocessor-clock cycles counted by the PAGE and RD_HLD timing parameters must be greater than or equal to RD_DLY.
<36>	SAM	R/W	0	—	Region 0 strobe AND mode. Internally combines the output-enable and write-enable strobes into a single strobe that is then driven onto both BOOT_OE_L and BOOT_WE_L. **NOTE:** This is useful for parts that use a single strobe along with a read/write bit (the read/write bit can be driven from an address pin).
<35:34>	WE_EXT	R/W	0x0	—	Region 0 write-enable count extension.
<33:32>	OE_EXT	R/W	0x0	—	Region 0 output-enable count extension.
<31>	EN	R/W	1	1	Region 0 enable.
<30>	ORBIT	R/W	0	0	Reserved (since there is no previous region).
<29>	ALE	RO	1	1	Region 0 address-latch-enable mode. ALE mode enables the multiplexed address/data bus mode. This is the only mode supported for CN71XX (forced to 1).
<28>	WIDTH	RO	0	—	Region 0 bus width. Bus width is fixed at 8 bits (forced to 0).
<27:16>	SIZE	R/W	0xFFF	—	Region 0 size. Region size is specified in 64K blocks and in “block–1” notation (i.e. 0 = 1 64K block, 1 = 2 64K blocks, etc.).
<15:0>	BASE	R/W	0x1FC0	0x1FC0	Region 0 base address. Specifies address bits [31:16] of the first 64K block of the region. **NOTE:** When region 0 is defined to be NAND flash, only MIO_BOOT_REG_CFG0[EN], MIO_BOOT_REG_CFG0[SIZE], and MIO_BOOT_REG_CFG0[BASE] remain functional. All other fields in MIO_BOOT_REG_CFG0 have no effect.

Cavium Confidential Rev. Beverly Yang 02/27/2014

MIO Boot Region (1..7) Configuration Registers
MIO_BOOT_REG_CFG(1..7)

The region *n* configuration register (one register for each of seven regions) contains configuration parameters for boot region *n*. See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:44>	—	RAZ	—	—	Reserved
<43:42>	DMACK	R/W	0x0	—	Reserved.
<41:40>	TIM_MULT	R/W	0x0	—	Region 1–7 timing multiplier. Specifies the timing multiplier for a region. The timing multiplier applies to all timing parameters, except for WAIT and RD_DLY, which simply count coprocessor-clock cycles. TIM_MULT is encoded as follows: 0x0 = 4×, 0x1 = 1×, 0x2 = 2×, 0x3 = 8×.
<39:37>	RD_DLY	R/W	0x0	—	Region 1–7 read sample delay. Specifies the read sample delay in coprocessor-clock cycles for a region. For read operations, the data bus is normally sampled on the same coprocessor-clock edge that drives BOOT_OE_L to the inactive state (or the coprocessor-clock edge that toggles the lower address bits in page mode). This parameter can delay that sampling edge by up to seven coprocessor-clock cycles. **NOTE:** The number of coprocessor-clock cycles counted by the PAGE and RD_HLD timing parameters must be greater than or equal to RD_DLY.
<36>	SAM	R/W	0	—	Region 1–7 strobe AND mode. Internally combines the output-enable and write-enable strobes into a single strobe that is then driven onto both BOOT_OE_L and BOOT_WE_L. **NOTE:** This is useful for parts that use a single strobe along with a read/write bit (the read/write bit can be driven from an address pin).
<35:34>	WE_EXT	R/W	0x0	—	Region 1–7 write-enable count extension.
<33:32>	OE_EXT	R/W	0x0	—	Region 1–7 output-enable count extension.
<31>	EN	R/W	0	—	Region 1–7 enable.
<30>	ORBIT	R/W	0	—	Region 1–7 OR bit. Asserts the given region's chip enable when there is an address hit in the previous region. **NOTE:** This is useful for CF cards because it allows the use of 2 separate timing configurations for common memory and attribute memory.
<29>	ALE	RO	1	1	Region 1–7 address-latch-enable mode. ALE mode enables the multiplexed address/data bus mode. This is the only mode supported for CN71XX (forced to 1).
<28>	WIDTH	RO	0	—	Region 1–7 bus width. Bus width is fixed at 8 bits (forced to 0).
<27:16>	SIZE	R/W	0x0	—	Region 1–7 size. Region size is specified in 64K blocks and in "block–1" notation (i.e. 0 = 1 64K block, 1 = 2 64K blocks, etc.).
<15:0>	BASE	R/W	0x0	—	Region 1–7 base address. Specifies address bits [31:16] of the first 64K block of the region.

Cavium Confidential
Bowen Yang
08/7/2014

MIO Boot Timing Register for Region 0
MIO_BOOT_REG_TIM0

The region 0 timing register contains page-mode, wait-mode, and timing parameters for region 0. When region 0 is defined to be NAND flash, all fields in MIO_BOOT_TIM_CFG0 have no effect. See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	PAGEM	RO	0	—	Reserved must be 0.
<62>	WAITM	R/W	0	—	Region 0 wait-mode enable. (See Table 28–5.)
<61:60>	PAGES	R/W	0x0	—	Region 0 page size: 0x0 = 8 bytes, 0x1 = 2 bytes, 0x2 = 4 bytes, 0x3 = 8 bytes
<59:54>	ALE[1]	R/W	0x4	—	Region 0 ALE count.
<53:48>	PAGE[1]	R/W	0x3F	—	Region 0 page count
<47:42>	WAIT	R/W	0x3F	—	Region 0 wait count, must be nonzero when WAITM is set to 1.
<41:36>	PAUSE	R/W	0x11	—	Region 0 pause count
<35:30>	WR_HLD	R/W	0x3F	—	Region 0 write hold count
<29:24>	RD_HLD	R/W	0x5	—	Region 0 read hold count
<23:18>	WE[1]	R/W	0x3F	—	Region 0 write enable count
<17:12>	OE[1]	R/W	0x3F	—	Region 0 output enable count
<11:6>	CE	R/W	0x5	—	Region 0 chip enable count
<5:0>	ADR	R/W	0x8	—	Region 0 address count

1. Value must be nonzero to ensure legal transitions on the corresponding boot bus outputs.

MIO Boot Timing Registers for Regions 1–7
MIO_BOOT_REG_TIM(1..7)

The region *n* timing register (one register for each of seven regions) contains page-mode, wait-mode, and timing parameters for region *n*. See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	PAGEM	RO	0	—	Reserved must be 0.
<62>	WAITM	R/W	0	—	Region 1–7 wait mode
<61:60>	PAGES	R/W	0x0	—	Region 1–7 page size: 0x0 = 8 bytes, 0x1 = 2 bytes, 0x2 = 4 bytes, 0x3 = 8 bytes
<59:54>	ALE[1]	R/W	0x3F	—	Region 1–7 ALE count.
<53:48>	PAGE[1]	R/W	0x3F	—	Region 1–7 page count
<47:42>	WAIT	R/W	0x3F	—	Region 1–7 wait count, must be nonzero when WAITM is set to 1.
<41:36>	PAUSE	R/W	0x3F	—	Region 1–7 pause count
<35:30>	WR_HLD	R/W	0x3F	—	Region 1–7 write-hold count
<29:24>	RD_HLD	R/W	0x3F	—	Region 1–7 read-hold count
<23:18>	WE[1]	R/W	0x3F	—	Region 1–7 write-enable count
<17:12>	OE[1]	R/W	0x3F	—	Region 1–7 output-enable count
<11:6>	CE	R/W	0x3F	—	Region 1–7 chip-enable count
<5:0>	ADR	R/W	0x3F	—	Region 1–7 address count

1. Value must be nonzero to ensure legal transitions on the corresponding boot bus outputs.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO Boot Local Memory Region Configuration Registers MIO_BOOT_LOC_CFG(0..1)

The local memory region configuration register (one for each of two regions) contains local memory region enable and local memory region base-address parameters. Each local memory region is 128 bytes organized as 16 entries × 8 bytes. See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31>	EN	R/W	0x0	—	Local memory region 0/1 enable
<30:28>	—	RAZ	—	—	Reserved
<27:3>	BASE	R/W	0x0	—	Local memory region 0/1 base address, specifying address bits [31:7] of the region.
<2:0>	—	RAZ	—	—	Reserved

MIO Boot Local Memory Region Address Register MIO_BOOT_LOC_ADR

The local memory region address register specifies the address for reading or writing the local memory region. This address post-increments following an access to the MIO boot local memory region data register.

- Local memory region 0 is addresses 0x00–0x78.
- Local memory region 1 is addresses 0x80–0xF8.

See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:3>	ADR	R/W/H	0x0	—	Local memory region address
<2:0>	—	RAZ	—	—	Reserved

MIO Boot Local Memory Region Data Register MIO_BOOT_LOC_DAT

This is a pseudo-register that reads/writes the local memory region at the address specified by the MIO boot local memory region address register (MIO_BOOT_LOC_ADR) when accessed. See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	DATA	R/W/H	—	—	Local memory region data

Cavium Confidential For Bowen Yang ALU 9/7/2014

MIO Boot Error Register
MIO_BOOT_ERR

The boot error register contains the address decode error and wait mode error bits. See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	WAIT_ERR	R/W1C	0	0	Wait mode error. This bit is set when wait mode is enabled and the external wait signal is not deasserted after 32K coprocessor-clock cycles.
<0>	ADR_ERR	R/W1C	0	0	Address decode error. This bit is set when a boot-bus access does not hit in any of the eight remote regions or two local memory regions.

MIO Boot Interrupt Register
MIO_BOOT_INT

The boot interrupt register contains the interrupt-enable bits for address-decode errors and wait-mode errors. See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved
<1>	WAIT_INT	R/W	0	—	Wait-mode error interrupt enable
<0>	ADR_INT	R/W	0	—	Address-decode error interrupt enable

MIO Boot Threshold Register
MIO_BOOT_THR

The boot threshold register contains MIO boot-threshold values. See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:22>	—	RAZ	—	—	Reserved
<21:16>	DMA_THR	R/W	0x0	—	DMA threshold. When non-DMA accesses are pending, perform a DMA access after this value of non-DMA accesses have completed. If set to zero, only perform a DMA access when non-DMA accesses are not pending.
<15:14>	—	RAZ	—	—	Reserved
<13:8>	FIF_CNT	RO/H	0x0	—	FIFO count. Contains the current IOI FIFO count.
<7:6>	—	RAZ	—	—	Reserved
<5:0>	FIF_THR	R/W	0x19	0x19	IOI busy threshold. Should always read 0x19 (the only legal value).

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO Boot Compensation Register
MIO_BOOT_COMP

This register sets the termination of boot-bus output pins. See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:11>	—	RAZ	—	—	Reserved
<10:8>	PCTL	R/W	—	—	Boot bus PCTL. This controls the pull-up drive strength of the boot-bus drivers. • 0x6 = No pull-up resistors (50Ω termination) • 0x7 = Pull-up resistor on BOOT_CLE (40Ω termination) • 0x4 = Pull-up resistor on BOOT_ALE (75Ω termination)
<9:3>	—	RAZ	—	—	Reserved
<2:0>	NCTL	R/W	—	—	Boot bus NCTL. This controls the pull-down drive strength of the boot-bus drivers. • 0x6 = No pull-up resistors (50Ω termination) • 0x7 = Pull-up resistor on BOOT_CLE (40Ω termination) • 0x4 = Pull-up resistor on BOOT_ALE (75Ω termination)

MIO Boot Pin Defaults Register
MIO_BOOT_PIN_DEFS

This register reflects the value of the BOOT_AD<31:0> pins, sampled when the internal reset signal first deasserts following a cold reset. See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:33>	—	RAZ	—	—	Reserved
<32>	DLM_SUPPLY	RO	—	—	DLM power supply setting based on DLMC_VPH_SELECT_18 pin. 1 = 1.8V 0 = 2.5V All others = reserved
<31:30>	RGM_SUPPLY	RO	—	—	RGMII power supply setting based on VDD_RGM_SUPPLY_SELECT pin. 01 = 1.8V 10 = 2.5V All others = reserved.
<29:27>	SMI_SUPPLY	RO	—	—	SMI power supply setting based on VDD_SMI_SUPPLY_SELECT pin. 001 = 1.8V 010 = 2.5V 100 = 3.3V All others = reserved.
<26:24>	IO_SUPPLY	RO	—	—	I/O power supply setting based on VDD_IO_SUPPLY_SELECT pin. 001 = 1.8V 010 = 2.5V 100 = 3.3V All others = reserved.
<23:17>	NS	---	—	—	Reserved.
<16>	REF_SEL	RO	—	—	Reference clock selection based on UART0_RTS_N pin at powerup. 0 = DLM_REF_CLK[1] pins div 2, must be set to 100 MHz 1 = PLL_REF_CLK pin (default), must be set to 50 MHz.

Cavium Confidential for Bowen Yang AU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<15>	ALE	RO	1	1	Set to 1 for backward compatibility
<14>	WIDTH	RO	0	0	Set to 0 for backward compatibility
<13>	VRM_DISABLE	RO	—	—	VRM disable based on BOOT_DIR pin at powerup
<12:10>	DEVICE	RO	—	—	Boot device based on BOOT_ALE<3:1> pins at power up. 0 = Parallel NOR 1 = Parallel NOR via int bootrom (reserved) 2 = EMMC/SD 3 = reserved 4 = SPI boot 16-bit address 5 = SPI boot 24-bit address 6 = SPI boot 32-bit Address (reserved) 7 = SPI NAND boot 32-bit address (reserved)
<9:8>	TERM	RO	—	—	Selects default driver termination latched from BOOT_ALE<0>, BOOT_CLE pins at powerup
<7:0>	USER0	RO	—	—	BOOT_AD<7:0> pins latched at powerup

MIO GPIO Compensation Register MIO_GPIO_COMP

This register sets the termination of GPIO output pins. See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:11>	—	RAZ	—	—	Reserved
<10:8>	PCTL	R/W	0x6	—	GPIO bus PCTL. Suggested values: • 0x6 (50Ω termination) • 0x7 (40Ω termination) • 0x4 (75Ω termination)
<9:3>	—	RAZ	—	—	Reserved
<2:0>	NCTL	R/W	0x6	—	GPIO bus NCTL. Suggested values: • 0x6 (50Ω termination) • 0x7 40Ω termination) • 0x4 (75Ω termination)

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO Boot Controller Selection Register MIO_BOOT_CTL

The hardware allows only one controller to use the boot bus and the MIO_NDF_DMA engine at a time. The SEL field determines the owner. This field should be changed only when the current controller operations are complete. Here are several situations.

1. If eMMC/SD is currently doing a read or write command without a DMA, then the MIO_EMM_INT[CMD_DONE] should occur before SEL is changed.
2. If eMMC/SD is currently doing a DMA operation with the device then the MIO_EMM_INT[DMA_DONE] should occur before SEL is changed. Care must be taken not to reprogram the MIO_NDF_DMA_CFG register until both the previous MIO_NDF_DMA_INT[DONE] has occurred and the new SEL has been set.
3. If NAND is currently doing one or more operations to the external device without using the DMA, then the NDF_INT[EMPTY] should occur to guarantee all commands are executed before changing the SEL value.
4. If NAND is currently doing one or more operations to the external device which includes DMAs then the SEL should not change until both the last MIO_NDF_DMA_INT[DONE] and NDF_INT[EMPTY] have occurred.

See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	SEL	R/W	0	—	Controller selection. 0 = eMMC/SD; 1 = NAND.

MIO Boot BIST Status Register MIO_BOOT_BIST_STAT

The boot BIST status register contains the BIST status for the MIO boot memories: 0 = pass, 1 = fail. See Table 28–7 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved
<13:0>	STAT	RO	0x0	0x0	BIST status.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 29

eMMC Interface Unit

This section contains the following subjects:

- Overview
- Access via CSR
- eMMC Access Via DMA
- eMMC DMA Interrupts
- Booting from an eMMC/SD Device
- Software Sequences
- eMMC Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The CN71XX boot-bus hardware can directly attach to eMMC flash parts. The eMMC interface can service up to four flash memory devices. The interface and the controller are flexible and can therefore attach to many different eMMC, MMC, and SD devices.

This chapter focuses on eMMC interface, and is a companion to Chapter 28, Boot Bus, which describes the other boot-bus features that share the same boot-bus pins. The boot-bus logic arbitrates for access to the boot-bus interface between the eMMC Interface Unit and other accesses.

The CN71XX eMMC interface unit conforms to the JEDEC Embedded MultiMediaCard (e•MMC) specification. (Refer to the JEDEC standard: JESD84-A441, March 2010.)

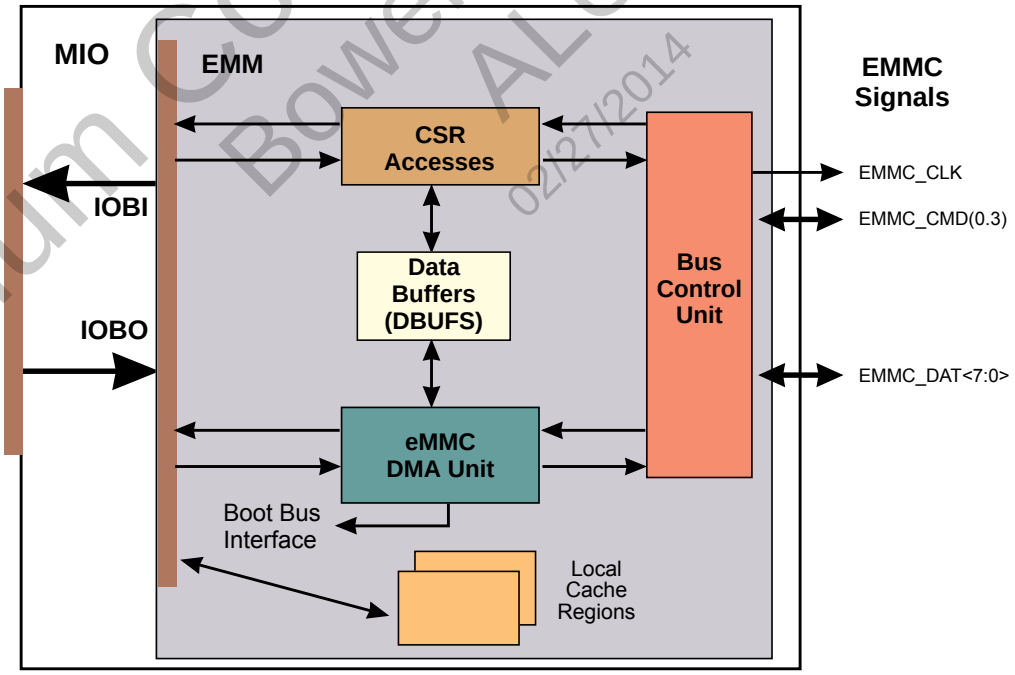
The CN71XX eMMC interface unit also conforms to the SD Specifications, Part 1, Physical Layer Specification, Version 4.00, May 20, 2011 published by the SD Card Association.

CN71XX has the following eMMC interface features:

- Shares data pins with the boot bus
- EMMC_CLK and EMMC_CMD(0..3) use boot-bus chip-select signals
- Support for single-data rate (SDR) and dual-data rate (DDR)
- Support for a 1, 4, and 8-bit data bus
- Transfer rates up to 52 MHz
- Software interface for direct access to command/response bus
- DMA read/write operations always on 512-byte blocks
- Stream mode is not supported; note that multi-block reads/writes are supported
- SPI mode is not supported
- eMMC boot is supported

The block diagram of the eMMC interface unit is shown in Figure 29–1.

Figure 29–1 eMMC Interface Block Diagram



Cavium Confidential For Bowen Yang ALU 02/27/2014

The bus control unit handles the details of the eMMC protocol. It services requests from the DMA unit and software (through CSRs). Software initiates all eMMC operations via CSR accesses. The DMA unit performs 512-byte block reads, 512-byte block writes, 512-byte multiple block reads, and 512-byte block writes.

The EMM contains two 512-byte data buffers (DBUFS). When multiple data buffers are used for an operation by either software or the DMA unit, each iterates through the data buffers starting with the lowest numbered. This allows for a block transfer between L2/DRAM and a data buffer to occur in parallel with a block transfer between a data buffer and the eMMC card.

The eMMC interface and boot bus share signals, which are listed in Table 29–1. The eMMC interface also uses one GPIO interface signal, also listed in Table 29–1. When software enables the eMMC bus through MIO_EMM_CFG[BUS_ENA], the CN71XX uses the eMMC interface signal names.

Table 29–1 eMMC and Boot-bus Shared Signals

Signal Name	eMMC Interface Signal Name
BOOT_AD<7:0>	EMMC_DAT<7:0>
BOOT_CE_5_L	EMMC_CLK
BOOT_CE_4_L	EMMC_CMD3
BOOT_CE_3_L	EMMC_CMD2
BOOT_CE_2_L	EMMC_CMD1
BOOT_CE_1_L	EMMC_CMD0
GPIO_8	EMMC_RST

The CN71XX eMMC controller controls the boot-bus and GPIO signals specified in Table 29–1. Table 29–2 shows how the signals map to eMMC devices.

Table 29–2 Boot-bus Signal Mapping to eMMC Devices

Signal Name	Corresponding eMMC Signal Name
BOOT_AD<7:0>	EMMC_DAT<7:0>
BOOT_CE_5_L	EMMC_CLK
BOOT_CE_<4:1>_L	EMMC_CMD
GPIO_8	EMMC_RST

When multiple eMMC parts are attached to CN71XX, they will share the DAT and CLK pins (i.e. share BOOT_AD<7:0> and BOOT_CE_5_L), but each have separate CMD pins (i.e. each have a unique one of BOOT_CE_<4:1>_L).

The CN71XX eMMC controller never uses the following boot bus pins:

BOOT_CE_<7:6,0>_L	Always deasserted while the eMMC controller uses the boot bus
BOOT_ALE	Always deasserted while the eMMC controller uses the boot bus
BOOT_OE_L	Always deasserted while the eMMC controller uses the boot bus
BOOT_WE_L	Always deasserted while the eMMC controller uses the boot bus
BOOT_WAIT_L	ignored while the eMMC controller uses the boot bus

The design includes a eMMC controller that drives the boot-bus pins appropriately for eMMC devices. Software directs the eMMC controller via CSRs and, optionally, interrupts.

Cavium Confidential For Bowen Yang ALU 04/17/2014

29.1 Access via CSR

The eMMC CSRs allow software to directly perform eMMC operations. This includes operations such as SLEEP_AWAKE, SWITCH, SEND_CSD, SEND_CID, SET_WRITE_PROT, LOCK_UNLOCK, and others as listed in the JEDEC specification. Additionally, CSRs allow software to directly perform operations on blocks smaller than 512 bytes. Note that stream and multiple-block commands are not supported.

Software sends commands using the MIO_EMM_CMD CSR. Prior to sending a command, software must ensure that all operations have completed, indicated by MIO_EMM_DMA[DMA_VAL,SWITCH_VAL, CMD_VAL] = 0. Also, software should not send a command if MIO_EMM_RSP_STS[DMA_PEND] = 1.

To send a command, software writes the various attributes of the operation to MIO_EMM_CMD and sets MIO_EMM_CMD[CMD_VAL] = 1. The command attributes include the eMMC bus, the 6-bit command index, and the 32-bit argument. For *adtc*-type operations, software must also specify a data buffer and offset by writing MIO_EMM_CMD[DBUF,OFFSET]. Note that all block transfers must be a multiple of eight bytes.

The hardware detects the valid flag and executes the operation after arbitrating for the boot bus. Upon completion of the operation, hardware clears MIO_EMM_CMD[CMD_VAL] and sets MIO_EMM_RSP_STS[CMD_DONE]. Additionally, the response (R1, R3, R4, or R5) is available in MIO_EMM_RSP_HI/LO. Software can also examine MIO_EMM_RSP_STS to gather status related to the response.

29.1.1 Command/Response Type Override

The CN71XX hardware automatically maps eMMC commands and responses to a type (i.e. read data, write data, R1, R2, etc), based on the default types defined for each command index in the JEDEC specification.

Commands are classified as one the following (CTYPE):

- No Data
- Read data into Dbuf
- Write data from Dbuf

Responses are classified as one the following (RTYPE):

- No Response
- R1, 48 bits
- R2, 136 bits
- R3, 48 bits
- R4, 48 bits
- R5, 48 bits

Cavium Confidential For Bowen Yang ALU 09/27/2014

On the CN71XX, these command and response types are programmable to accommodate manufacturer-specific and alternate commands. Software can override the default mapping using MIO_EMM_CMD[CTYPE_XOR, RTYPE_XOR]. To maintain the default mapping, software writes these fields to zero. Table 29–3 shows the default RTYPE and CTYPE mapping for each eMMC command.

Table 29–3 Command and Response Types

Command Index	RTYPE	CTYPE
CMD0	R0	0
CMD1	R3	0
CMD2	R2	0
CMD3	R1	0
CMD4	R0	0
CMD5	R1	0
CMD6	R1	0
CMD7	R1	0
CMD8	R1	1
CMD9	R2	0
CMD10	R2	0
CMD11	R1	1
CMD12	R1	0
CMD13	R1	0
CMD14	R1	1
CMD15	R0	0
CMD16	R1	0
CMD17	R1	1
CMD18	R1	1
CMD19	R1	3
CMD20	R1	2
CMD21	R0	0
CMD22	R0	0
CMD23	R1	0
CMD24	R1	2
CMD25	R1	2
CMD26	R1	2
CMD27	R1	2
CMD28	R1	0
CMD29	R1	0
CMD30	R1	1
CMD31	R1	1
CMD32	R0	0
CMD33	R0	0
CMD34	R0	0
CMD35	R1	0
CMD36	R1	0
CMD37	R0	0
CMD38	R1	0
CMD39	R4	0
CMD40	R5	0
CMD41	R0	0
CMD42	R1	2
CMD43 – CMD54	R0	0
CMD55	R1	0
CMD56	R1	1 or 2[1]
CMD57 – CMD63	R0	0

1. Based on command argument.

29.1.2 SD Card Support

The command/response type overrides are necessary for SD card operations. (See MIO_EMM_CMD[CTYPE_XOR, RTYPE_XOR].) Several SD commands differ from the default values selected for eMMC. For example, the eMMC CMD3 (SET_RCA) has an R1 response type and takes the RCA as an argument; while the SD CMD3 (SEND_RCA) has an R6 response (RTYPE_XOR of 110) and no argument. Table 29–4 shows some of the common differences. See the eMMC and SD specifications for additional information.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 29–4 eMMC vs. SD Command/Response Types

CMD	eMMC/SD	Description	Typical ARG	CTYPE_XOR	RTYPE_XOR	Notes
3	EMMC	Set RCA	[31:16] RCA	—	R1	
	SD	Send RCA	—	—	R6 (010)	RCA in response
8	EMMC	Read ECSD	—	C1	R1	ECSD in DBUF
	SD	Send interface condition	[11:8] Voltage [7:1] Check	C0 (01)	R7 (010)	Voltage in response
11	EMMC	Read Stream	[31:0] Addr	C1	R1	Not Supported
	SD	Voltage Switch	0	C0 (01)	R1 (000)	Switch to 1.8V
19	EMMC	Bus Test Write	—	C2	R1	
	SD	Tuning	0	C2	R1	64-bit Tuning Block, Offset = 63
20	EMMC	Stream Write	[31:0] Addr	C2	R1	Not supported
	SD	Speed Class	[31:28] Speed	C0 (10)	R1 (000)	SDXC only
32	SD	Erase Block Start	[31:0] Addr	C0	R1 (001)	
33	SD	Erase Block End	[31:0] Addr	C0	R1 (001)	
58	SD	Read OCR	0	C0	R3 (011)	
59	SD	CRC On/Off	[0:0] CRC On	C0	R1 (001)	Turns on/off CRC checking?
ACMD41	SD	Get Condition	—	C0	R1 (001)	Poll on response[39]

29.2 eMMC Access Via DMA

DMA operations can be performed as a series of 512-byte block writes or as 512-byte multiple block reads/writes. The DMA is between L2/DRAM and the eMMC bus.

Both write and read DMA operations to the eMMC device require a pair of DMA engines to move the data. MIO_NDF_DMA_CFG is used to program the DMA between L2C/DRAM and the eMMC local buffers, and MIO_EMM_DMA transfers data between the local buffers and the eMMC device.

MIO_NDF_DMA_CFG is programmed first to move in the opposite direction to MIO_EMM_DMA.

29.2.1 DMA Write Operations

For DMA write operations, software must program MIO_EMM_DMA with the starting card address, the number of 512-byte blocks to write, the data buffer threshold, and MIO_EMM_DMA[RW] = 1. Additionally, software must specify the addressing mode of the card via MIO_EMM_DMA[SECTOR]. When [SECTOR] = 1, the card address is interpreted as a 512-byte block address. When [SECTOR] = 0, the card address is interpreted as a byte address. Software sets MIO_EMM_DMA[MULTI] = 0 for single-block writes and MIO_EMM_DMA[MULTI] = 1 for multiple-block writes.

The hardware transfers data to the next available DBUF. Once the DBUF has enough data to satisfy the data threshold, the hardware issues a 512-byte block write to the eMMC interface after arbitrating for the boot bus, first delivering the command to the card and then receiving the response from the card. The card response may contain bad card status or bad CRC (or both or neither).

Cavium Confidential For Bowen Yang ALU 02/27/2014

If the response contains good CRC and good card status, the hardware transfers the 512-byte block to the card. The hardware receives and checks the 3-bit CRC token status bits returned by the card after the 512-byte block transfer. If it passes, the hardware advances the card address, and decrements the block count.

For a failing indication, or if the response contains good CRC but bad card status, or if the response contains bad CRC, the hardware pauses the DMA transfer. The hardware simply stops issuing 512-byte block write commands, clears MIO_EMM_RSP_STS[DMA_VAL], sets MIO_EMM_RSP_STS[DMA_PEND], and releases the bus. Software may investigate then resume or terminate the DMA operation.

Software may choose to resume the DMA transfer by writing MIO_EMM_DMA[DMA_VAL] = 1 or terminate the DMA transfer by writing MIO_EMM_DMA[DMA_VAL] = 1 and MIO_EMM_DMA[DAT_NULL] = 1. See Section 29.5.7 and Section 29.5.8.

Note that if the card is returning bad status, the card will not be in the receive state, and eventually, the watchdog timer will expire and cause the bus to be released.

29.2.2 DMA Read Operations

For DMA read operations, software must program MIO_EMM_DMA with the starting card address, the number of 512-byte blocks to read, the data buffer threshold, and MIO_EMM_DMA[RW] = 0. Additionally, software must specify the addressing mode of the card via MIO_EMM_DMA[SECTOR]. When [SECTOR] = 1, the card address is interpreted as a 512-byte block address. When [SECTOR] = 0, the card address is interpreted as a byte address. Software sets MIO_EMM_DMA[MULTI] = 0 for single-block read operations and MIO_EMM_DMA[MULTI] = 1 for multiple-block read operations.

The hardware issues a 512-byte block read to the eMMC interface after arbitrating for the boot bus, indicating the card address and the selected DBUF. The hardware delivers the command to the card and then receives the response from the card. The card response may contain bad card status or bad CRC (or both or neither).

If the response contains good CRC and good card status, the hardware checks the data CRC and provides the data/response CRC status. If both the response and data CRC are good, the hardware advances the card address, decrements the block count, and begins transferring the data to the boot-bus DMA memory. If either the response CRC or the data CRC are bad, the hardware pauses the DMA transfer by clearing MIO_EMM_RSP_STS[DMA_VAL], setting MIO_EMM_RSP_STS[DMA_PEND], and then releases the bus. With multiple-block reads, the hardware also discards the data.

If the response contains good CRC and bad card status, no data is transferred, the hardware pauses the DMA transfer by clearing MIO_EMM_RSP_STS[DMA_VAL], setting MIO_EMM_RSP_STS[DMA_PEND], and then releases the bus.

If the response contains bad CRC for a single-block read operation, the card may or may not return any data. The hardware assumes that the card status was good and attempts to receive the read data normally. If this assumption turns out to be incorrect, the operation ends when the watchdog timer expires. The watchdog timer causes the bus to be released. Software may choose to resume the DMA transfer by writing MIO_EMM_DMA[DMA_VAL] = 1 or terminate the DMA transfer by writing MIO_EMM_DMA[DMA_VAL] = 1 and MIO_EMM_DMA[DAT_NULL = 1].

Cavium Confidential For Beverly Yang 09/30/2014

If the response contains bad CRC for a multiple-block read operation, the card may or may not return any data. The hardware sets MIO_EMM_RSP_STS[RSP_CRC_ERR] and issues a STOP_TRANSMISSION command to the card. Once the hardware receives the STOP_TRANSMISSION response, it clears MIO_EMM_RSP_STS[DMA_VAL] and sets MIO_EMM_RSP_STS[DMA_PEND].

Software may choose to resume the DMA transfer by writing MIO_EMM_DMA[DMA_VAL] = 1 or terminate the DMA transfer by writing MIO_EMM_DMA[DMA_VAL] = 1 and MIO_EMM_DMA[DAT_NULL] = 1. See Section 29.5.7 and Section 29.5.8.

29.3 eMMC DMA Interrupts

The eMMC logic provides interrupts for both the MIO_NDF_DMA engine (MIO_NDF_DMA_INT) and the MIO_EMM_DMA engine (MIO_EMM_INT). Cavium recommends that software monitor the MIO_EMM_INT[DMA_DONE] and MIO_EMM_INT[DMA_ERR] bits to ensure that the eMMC transfer has completed successfully.

If the eMMC device is being read, it is also necessary to ensure that the transfer to the L2C has occurred before software accesses or executes code from these memory locations. This is done by monitoring MIO_NDF_DMA_INT and not proceeding until this bit has been set.

29.4 Booting from an eMMC/SD Device

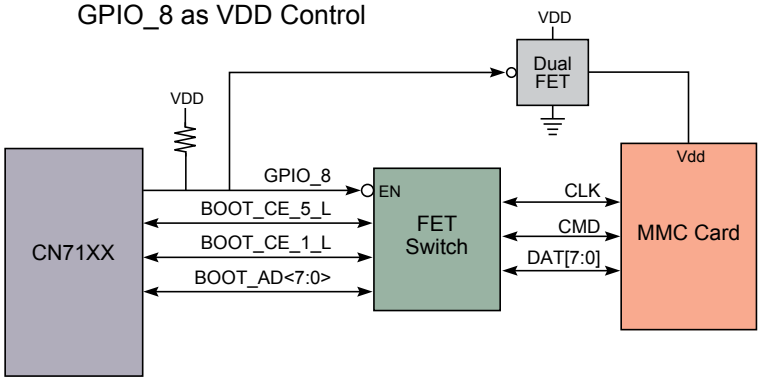
eMMC boot is selected by the BOOT_ALE[3:1] signals at reset. When eMMC boot is selected, there must be an eMMC device whose CMD pin attaches to BOOT_CE_1_L, and that eMMC device must have boot code in its lower-most blocks. eMMC boot should not be selected in REMOTE_BOOT mode.

When this option is selected, CN71XX attempts to dynamically determine the eMMC speed and data bus width to use to read in the blocks that contain the boot code. In CN71XX pass 1.0, the CN71XX first issues eMMC commands CMD0, CMD1, CMD2, CMD3, and CMD7 in sequence to the eMMC part attached to BOOT_CE_1_L. Starting with CN71XX pass 1.1, in order to support both eMMC and SD devices, the sequence is CMD0, CMD1, CMD8, ACMD41/CMD1, CMD2, CMD3, and CMD7. It does this with a one-bit data bus at a speed that does not exceed 400 KHz. Then CN71XX attempts to read in the 16 lower-most blocks (8 KBytes) from the part with an 8-bit data bus for eMMC and a 4-bit data bus for SD, at a speed that does not exceed 20 MHz. If that first read attempt fails, CN71XX then attempts to read the 16 blocks from the part with a 1-bit data bus at a speed that does not exceed 20 MHz. If that second read attempt fails, CN71XX further attempts to read the 16 blocks from the part with a 1-bit data bus at 400 KHz or less. If that third and final read attempt fails, or if earlier non-read commands failed, CN71XX sets the boot bus pins to a noticeable BOOT_FAILURE pattern (see MIO_EMM_CFG[BOOT_FAIL] description), stops, and the chip will not boot.

The CN71XX drives the GPIO_8 signal to the value opposite the pull-up/pull-down resistor connected to the pin. It is presumed that the board designer will use GPIO_8 to control either a FET supplying VDD to the eMMC card or directly to the reset pin on the device. In the case of GPIO connecting to VDD, the pull-up/pull-down should be used to disable VDD and also to disconnect EMMC_DAT<7:0>, EMMC_CLK, and EMMC_CMD on the card via FET switches.

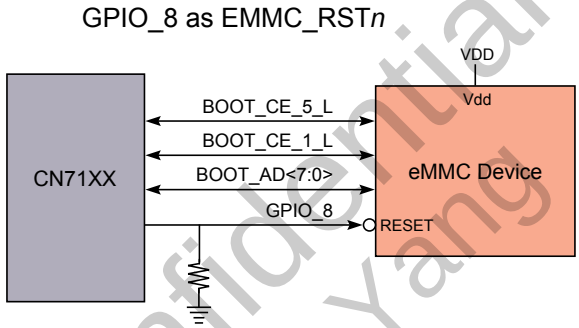
Cavium Confidential For Broadcom Use Only

Figure 29–2 GPIO_8 as VDD Control



In the reset case, a pulldown resistor should be used to maintain reset while GPIO_8 is not driven by CN71XX. If either option is used, care must be taken to guarantee a minimum pulse time and a minimum number of clock cycles before a CMD0 is issued per the eMMC specification.

Figure 29–3 GPIO_8 as EMMC_RSTn



When this option is selected, CN71XX reads the 8 KB boot code from the boot eMMC part into a L2 scratchpad after first creating the L2 scratchpad. The scratchpad is 256 KB and occupies CN71XX L2/DRAM physical addresses 0x1000000 to 0x103FFFF. CN71XX copies the boot code from the eMMC part to CN71XX physical addresses 0x1000000 to 0x1001FFF. (Section 3.2.8 describes the available CN71XX physical addresses.) Then CN71XX core 0 jumps to virtual address 0xFFFFFFFF81000000 (i.e. CN71XX physical address 0x1000000). Note that no DRAM controllers have been initialized at the point that CN71XX starts executing the boot code that was in the boot eMMC.

Cavium Confidential For Bowen Yang BLU 12/2/2014

Some more detailed description of the state of CN71XX at the time that core 0 jumps to the eMMC boot code held in the L2 scratchpad:

- The MIPS core 0 is alive and has been initialized, including TLB initialization
- CvmCtl[IPREF, DEFET] may (or may not) be set
- PageGrain[ELPA] may (or may not) be set
- Other core state (e.g. GPRs and other user state) may not be initialized
- A 256KB scratchpad has been created starting at physical address 0x1000000 in L2 – all L2 ways are disabled for MIPS core 0.
- The boot code has been copied from the boot eMMC into physical addresses 0x1000000 to 0x1001FFF in the scratchpad.
- The boot eMMC (attached to BOOT_CE_1_L) was brought up and is left in the state where all the boot code was successfully read in
- The following CSRs/fields may have been modified from their reset values:
  - MIO_EMM_CFG[BUS_ENA<0>]
  - MIO_EMM_SWITCH
  - MIO_EMM_MODE0
  - MIO_EMM_DMA
  - MIO_EMM_CMD
  - MIO_EMM_RSP_STS
  - MIO_EMM_RSP_LO
  - MIO_EMM_RSP_HI
  - MIO_EMM_INT
  - MIO_EMM_WDOG
  - MIO_EMM_BUF_IDX
  - MIO_EMM_BUF_DAT
  - MIO_EMM_STS_MASK
  - MIO_EMM_RCA
  - MIO_NDF_DMA_CFG
  - MIO_NDF_DMA_INT
  - GPIO_BIT_CFG8
  - GPIO_RX_DAT
  - GPIO_TX_SET
  - GPIO_TX_CLR

29.5 Software Sequences

This section supplies software sequences for various eMMC interface processes.

Cavium Confidential For Bowen Yang ALU 02/27/2014

29.5.1 Power Cycle eMMC Card

1. Clear GPIO_BIT_CFG8[TX_OE] to disable power / enable reset.
2. Wait 20 ms.
3. Read GPIO_RX_DAT[DAT<8>].
4. If GPIO_RX_DAT[DAT<8>] is set, write 0x0000_0100 to GPIO_TX_CLR, else write 0x000_0100 to GPIO_TX_CLR.
5. Set GPIO_BIT_CFG8[TX_OE] to enable power / disable reset.
6. Wait 2 ms.
7. Set corresponding MIO_EMM_CFG[BUS_ENA] to enable EMM_CLK.
8. Set clock rate using MIO_EMM_SWITCH.
9. Wait a minimum of 74 eMMC clock cycles.

29.5.2 Reset Card

1. Write MIO_EMM_CMD[CMD_VAL] = 1, [CMD_IDX] = 0, [ARG] = 0x0.
2. Read MIO_EMM_RSP_STS.
3. If MIO_EMM_RSP_STS[CMD_DONE] = 0, repeat step 2.

29.5.3 Initialize Card – eMMC Only

Determine the card operating conditions (SEND_OP_COND).

1. Write MIO_EMM_CMD[CMD_VAL] = 1, [CMD_IDX] = 1, [ARG] = 0x40FF8000.
2. Read MIO_EMM_RSP_STS.
3. If MIO_EMM_RSP_STS[CMD_DONE] = 0, repeat step 2.
4. If MIO_EMM_RSP_LO[DAT<39>] = 0, repeat steps 1–3.
5. If MIO_EMM_RSP_LO[DAT<38:37> = 0x0, then byte addressing mode, else sector addressing mode.

NOTE: The prior pseudo-code assumes that there is only one eMMC part per chip-select. If there are more, it may be necessary to consult the SEC_COUNT field in the extended CSD of the part to determine byte/sector addressing mode, per eMMC specifications.

Determine the card identification number (ALL_SEND_CID).

1. Write MIO_EMM_CMD[CMD_VAL] = 1, [CMD_IDX] = 2, [ARG] = 0x0.
2. Read MIO_EMM_RSP_STS.
3. If MIO_EMM_RSP_STS[CMD_DONE] = 0, repeat step 2.

Assign the card a nonzero relative address (SET_RELATIVE_ADDR).

1. Write MIO_EMM_CMD[CMD_VAL = 1], [CMD_IDX] = 3, [ARG] = (RCA<<16).
2. Read MIO_EMM_RSP_STS.
3. If MIO_EMM_RSP_STS[CMD_DONE] = 0, repeat step 2.

Select the card (SELECT/DESELECT_CARD).

1. Write MIO_EMM_CMD[CMD_VAL] = 1, [CMD_IDX] = 7, [ARG] = (RCA<<16)
2. Read MIO_EMM_RSP_STS.
3. If MIO_EMM_RSP_STS[CMD_DONE] = 0, repeat step 2.
4. Write MIO_EMM_RCA[RCA] = RCA.

Cavium Confidential For Bowen Yang ADI 02/07/2012

29.5.4 Initialize Card – eMMC and SD

Verify SD memory card interface operating condition (SEND_IF_COND).

1. Write MIO_EMM_CMD[CMD_VAL] = 1, [CMD_IDX] = 8, [ARG] = 0x0, [CTYPE_XOR] = 01.
2. Read MIO_EMM_RSP_STS.
3. If MIO_EMM_RSP_STS[CMD_DONE] = 0, repeat step 2.

Start initialization and determine operating conditions (SD_SEND_OP_COND).

1. Write MIO_EMM_CMD[CMD_VAL] = 1, [CMD_IDX] = 55, [ARG] = 0x0.
2. Read MIO_EMM_RSP_STS.
3. If MIO_EMM_RSP_STS[CMD_DONE] = 0, repeat step 2.
4. Write MIO_EMM_CMD[CMD_VAL] = 1, [CMD_IDX] = 41, [RTYPE_XOR] = 001. If CMD8 above times out, write MIO_EMM_CMD[ARG] = 0x00FF8000; otherwise write MIO_EMM_CMD[ARG] = 0x04FF8000.
5. Read MIO_EMM_RSP_STS.
6. If MIO_EMM_RSP_STS[CMD_DONE] = 0, repeat step 6.
7. If MIO_EMM_RSP_STS[RSP_TIMEOUT] = 1, then initialize eMMC. Go to Section 29.5.3.
8. If MIO_EMM_RSP_LO[DAT<39>] = 0, repeat steps 1–6.
9. If MIO_EMM_RSP_LO[DAT<38:37> = 0x0, then byte addressing mode, else sector addressing mode.

Determine the card identification number (ALL_SEND_CID).

1. Write MIO_EMM_CMD[CMD_VAL] = 1, [CMD_IDX] = 2, [ARG] = 0x0.
2. Read MIO_EMM_RSP_STS.
3. If MIO_EMM_RSP_STS[CMD_DONE] = 0, repeat step 2.

Assign the card a nonzero relative address (SEND_RELATIVE_ADDR).

1. Write MIO_EMM_CMD[CMD_VAL = 1], [CMD_IDX] = 3, [ARG] = 0.
2. Read MIO_EMM_RSP_STS.
3. If MIO_EMM_RSP_STS[CMD_DONE] = 0, repeat step 2.
4. RCA = MIO_EMM_RSP_LO<39:24>.

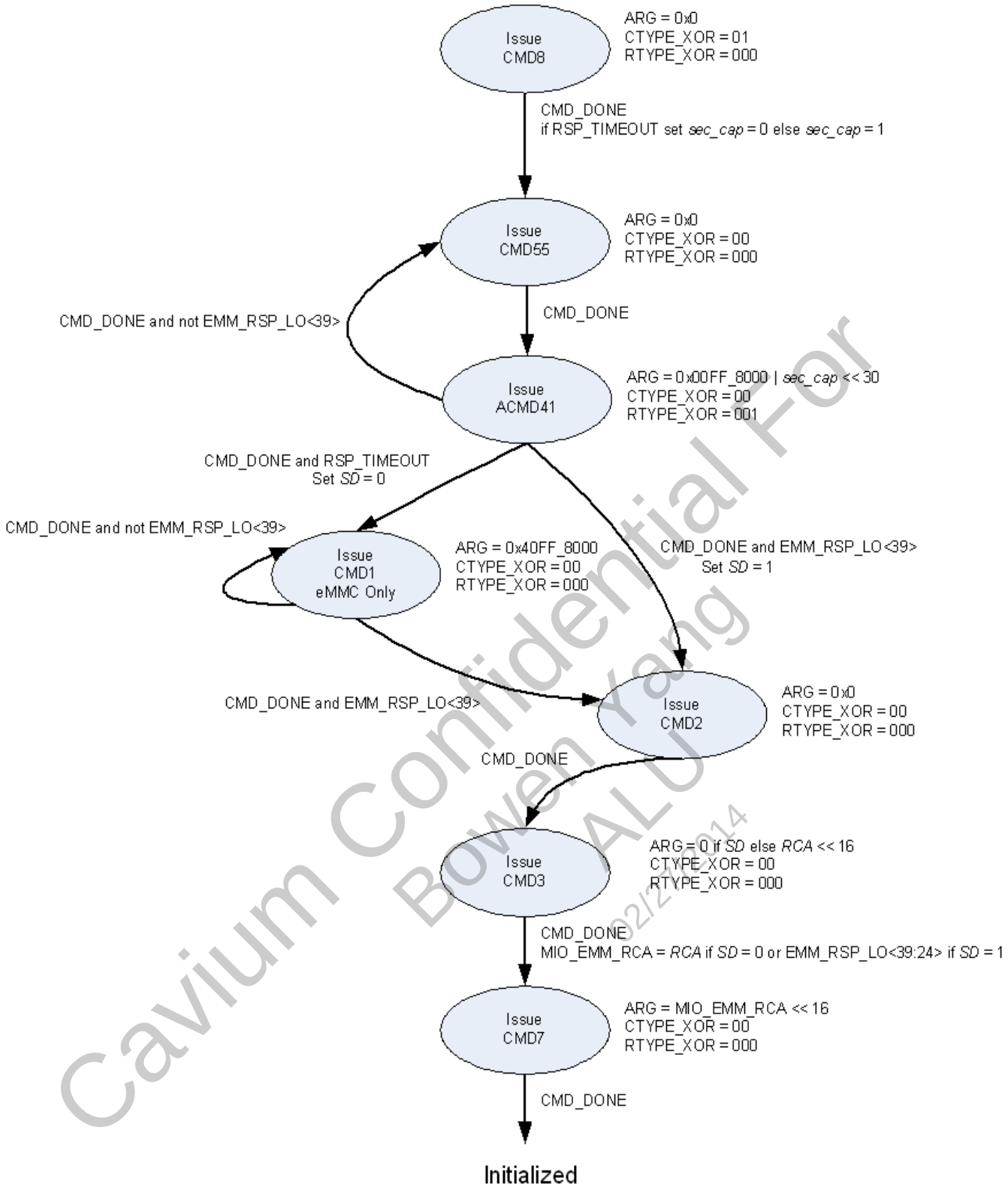
Select the card (SELECT/DESELECT_CARD).

1. Write MIO_EMM_CMD[CMD_VAL] = 1, [CMD_IDX] = 7, [ARG] = (RCA<<16)
2. Read MIO_EMM_RSP_STS.
3. If MIO_EMM_RSP_STS[CMD_DONE] = 0, repeat step 2.
4. Write MIO_EMM_RCA[RCA] = RCA.

Figure 29–4 depicts the eMMC and SD Initialization sequence.

Cavium Confidential For ALL Bowen Yang 02/27/2014

Figure 29–4 eMMC and SD Initialization Sequence



Cavium Confidential For Bowen Yang ALU 02/2/2014

29.5.5 Switch Operating Mode

Select new power class, bus width, timing mode, and CLK frequency. Typically the power class, bus width, timing mode, and clock frequency are eMMC device, customer application, and board-layout dependent. eMMC devices provide information on supported power class, bus width, and timing mode in the extended CSD register (CMD8). The bus width is limited by both the extended CSD register and the number of data bits connected on the board. Power supply limitations and bus capacitance may limit both power class and clock frequency. After the switch command executes, the actual parameters used are available in the MIO_EMM_MODE(0..3) register.

NOTE: Only do these steps after card initialization.

1. Write MIO_EMM_SWITCH with desired modes and [SWITCH_EXE] = 1.
2. Read MIO_EMM_RSP_STS.
3. If MIO_EMM_RSP_STS[SWITCH_VAL] = 1, repeat step 2.
4. Check MIO_EMM_SWITCH[SWITCH_ERR*] for SWITCH errors.

29.5.6 Initiate a DMA Transfer

NOTE: Only do these steps after card initialization.

Direct the NDF DMA engine to move data between L2/DRAM and EMM.

1. Write MIO_NDF_DMA_CFG with L2/DRAM attributes.

Direct the eMMC controller to move data between EMM and the eMMC card.

1. Write MIO_EMM_DMA with eMMC attributes.

Wait for the operation to complete.

1. Read MIO_EMM_RSP_STS.
2. If MIO_EMM_RSP_STS[DMA_VAL] = 1, repeat step 1.
3. If MIO_EMM_RSP_STS[DMA_PEND] = 1, go to resume/terminate[1].

29.5.7 Resume a DMA transfer

NOTE: Only do these steps when MIO_EMM_RSP_STS[DMA_PEND] = 1.

Direct the eMMC controller to resume.

1. Read/Modify/Write the MIO_EMM_DMA register setting the DMA_VAL bit.

Wait for the operation to complete.

1. Read MIO_EMM_RSP_STS[DMA_VAL] = 1.
2. If MIO_EMM_RSP_STS[CMD_DONE] = 0, repeat step 1.
3. If MIO_EMM_RSP_STS[DMA_PEND] = 1, go to resume/terminate[1].

1. Examining MIO_EMM_RSP_LO can help determine why DMA was unable to complete and help decide whether to resume or terminate.

Cavium Confidential For Bowen Yang ALU 02/27/2014

29.5.8 Terminate a DMA transfer

NOTE: Only do these steps when MIO_EMM_RSP_STS[DMA_PEND] = 1.

Direct the eMMC controller to terminate.

1. Read/Modify/Write the MIO_EMM_DMA register setting the DMA_VAL and DMA_NULL bits.

Wait for the operation to complete.

1. Read MIO_EMM_RSP_STS.
2. If MIO_EMM_RSP_STS[DMA_VAL] = 1, repeat step 1.

29.6 eMMC Registers

The eMMC registers are listed in Table 29–5.

Table 29–5 eMMC Registers

Register	Address[1]	CSR Type[2]	Detailed Description
MIO_NDF_DMA_CFG	0x0001180000000168	RSL	See page 1312
MIO_NDF_DMA_INT	0x0001180000000170	RSL	See page 1312
MIO_NDF_DMA_INT_EN	0x0001180000000178	RSL	See page 1312
MIO_EMM_CFG	0x0001180000002000	RSL	See page 1313
MIO_EMM_MODE(0..3)	0x1180000002008+ α*0x8	RSL	See page 1313
MIO_EMM_SWITCH	0x0001180000002048	RSL	See page 1314
MIO_EMM_DMA	0x0001180000002050	RSL	See page 1315
MIO_EMM_CMD	0x0001180000002058	RSL	See page 1316
MIO_EMM_RSP_STS	0x0001180000002060	RSL	See page 1317
MIO_EMM_RSP_LO	0x0001180000002068	RSL	See page 1318
MIO_EMM_RSP_HI	0x0001180000002070	RSL	See page 1319
MIO_EMM_INT	0x0001180000002078	RSL	See page 1319
MIO_EMM_INT_EN	0x0001180000002080	RSL	See page 1319
MIO_EMM_WDOG	0x0001180000002088	RSL	See page 1320
MIO_EMM_SAMPLE	0x0001180000002090	RSL	See page 1320
MIO_EMM_STS_MASK	0x0001180000002098	RSL	See page 1321
MIO_EMM_RCA	0x00011800000020A0	RSL	See page 1321
MIO_EMM_BUF_IDX	0x00011800000020E0	RSL	See page 1321
MIO_EMM_BUF_DAT	0x00011800000020E8	RSL	See page 1321

1. For registers with multiple versions, α represents the values within the specified range.
2. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang 8/27/2014

eMMC DMA Configuration Register
MIO_NDF_DMA_CFG

This register aggregates all state machines used in eMMC/SD flash controller for debug purposes. Sixty-four-bit operations must be used to access this register. See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	EN	R/W	0x0	—	DMA engine enable.
<62>	RW	R/W	0x0	—	DMA engine R/W bit: 0 = read, 1 = write.
<61>	CLR	R/W	0x0	—	DMA engine abort. When set to 1, DMA is aborted and EN is cleared on completion.
<60>	—	RAZ	—	—	Reserved.
<59>	SWAP32	R/W	0x0	—	DMA engine enable 32-bit swap.
<58>	SWAP16	R/W	0x0	—	DMA engine enable 16-bit swap.
<57>	SWAP8	R/W	0x0	—	DMA engine enable 8-bit swap.
<56>	ENDIAN	R/W	0x0	—	DMA engine endian mode: 0 = big-endian, 1 = little-endian.
<55:36>	SIZE	R/W	0x0	—	DMA engine size. Specified in the number of 64-bit transfers (encoded in –1 notation). For example, to transfer 512 bytes, SIZE = 64 – 1 = 63.
<35:0>	ADR	R/W	0x0	—	DMA engine address. Must be 64-bit aligned.

eMMC DMA Interrupt Register
MIO_NDF_DMA_INT

Sixty-four-bit operations must be used to access this register. See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	DONE	R/W1C	0	—	DMA engine request completion interrupt.

eMMC DMA Interrupt-Enable Register
MIO_NDF_DMA_INT_EN

Sixty-four-bit operations must be used to access this register. See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	DONE	R/W	0	—	DMA engine request completion interrupt enable.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO eMMC Configuration Register
MIO_EMM_CFG

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:17>	—	RAZ	0x0	—	Reserved
<16>	BOOT_FAIL	R/W	0	—	Software should set BOOT_FAIL when an unrecoverable error occurs while attempting to boot from eMMC or NOR Flash. When set, the following pattern is output: BOOT_AD<7:0> driven to 0 BOOT_CE_<7:0>_L driven to 1 BOOT_ALE driven to 0 BOOT_OE_L driven to 1 BOOT_WE_L driven to 1
<15:4>	—	RAZ	0x0	—	Reserved
<3:0>	BUS_ENA	R/W	0x0	—	eMMC bus enable mask Setting bit0 of BUS_ENA causes BOOT_CE[1] to become dedicated eMMC bus 0 command (i.e. disabling any NOR use). Setting bit1 of BUS_ENA causes BOOT_CE[2] to become dedicated eMMC bus 1 command (i.e. disabling any NOR use). Setting bit2 of BUS_ENA causes BOOT_CE[3] to become dedicated eMMC bus 2 command (i.e. disabling any NOR use). Setting bit3 of BUS_ENA causes BOOT_CE[4] to become dedicated eMMC bus 3 command (i.e. disabling any NOR use). Setting any bit of BUS_ENA causes BOOT_CE[5] to become the eMMC clock for both bus0 and bus1.

MIO eMMC Operating Mode Register
MIO_EMM_MODE(0..3)

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:49>	—	RAZ	0x0	—	Reserved
<48>	HS_TIMING	RO/H	0	—	Current high-speed timing mode. Required when CLK frequency is higher than 20MHz.
<47:43>	—	RAZ	0x0	—	Reserved
<42:40>	BUS_WIDTH	RO/H	0x0	—	Current card-bus mode. Out of reset, the card is in 1-bit data bus mode. Select bus width. 0x0 = 1-bit data bus (power on) 0x1 = 4-bit data bus 0x2 = 8-bit data bus 0x3 = Reserved 0x4 = Reserved 0x5 = 4-bit data bus (dual data rate) 0x6 = 8-bit data bus (dual data rate) 0x7 = Reserved
<39:36>	—	RAZ	0x0	—	Reserved

Cavium Confidential For Bowen Yang ALU 02/21/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<35:32>	POWER_CLASS	RO/H	0x0	—	Out of reset, the card power class is 0, which is the minimum current consumption class for the card. EXT_CSD bytes [203:200] and [239:238] contain the power class for different BUS_WITDH and CLK frequencies. Software should write this field with the 4-bit field from the EXT_CSD bytes corresponding to the selected operating mode.
<31:16>	CLK_HI	RO/H	0x9C4	—	Current number of coprocessor-clock cycles to hold the eMMC CLK pin high
<15:0>	CLK_LO	RO/H	0x9C4	—	Current number of coprocessor-clock cycles to hold the eMMC CLK pin low

MIO eMMC Operating Mode Switch Register MIO_EMM_SWITCH

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:62>	—	RAZ	0x0	—	Reserved.
<61:60>	BUS_ID	R/W/H	0x0	—	Bus ID. Specifies the eMMC bus.
<59>	SWITCH_EXE	R/W/H	0	—	When SWITCH_EXE is 0, the operating modes are updated directly without performing any SWITCH operations. This allows software to perform the SWITCH operations manually, then update the hardware. Software writes this bit to a 1 to indicate that hardware should perform the necessary SWITCH operations. • First, the POWER_CLASS switch is performed. If it fails, SWITCH_ERR0 is set and the remaining SWITCH operations are not performed. If it succeeds, the POWER_CLASS field is updated and the HS_TIMING switch is performed. • If the HS_TIMING switch fails, SWITCH_ERR1 is set and the remaining SWITCH operations are not performed. If it succeeds, the HS_TIMING field is updated and the BUS_WIDTH switch operation is performed. • If the BUS_WIDTH switch operation fails, SWITCH_ERR2 is set. If it succeeds, the BUS_WITDH is updated. Changes to CLK_HI and CLK_LO are discarded if any SWITCH_ERR occurs.
<58>	SWITCH_ERR0	RO/H	0	—	Error encounter while performing POWER_CLASS switch. See MIO_EMM_RSP_STS.
<57>	SWITCH_ERR1	RO/H	0	—	Error encounter while performing HS_TIMING switch. See MIO_EMM_RSP_STS.
<56>	SWITCH_ERR2	RO/H	0	—	Error encounter while performing BUS_WIDTH switch. See MIO_EMM_RSP_STS.
<55:49>	—	RAZ	0x0	—	Reserved.
<48>	HS_TIMING	R/W	0	—	Requested update to HS_TIMING.
<47:43>	—	RAZ	0x0	—	Reserved.
<42:40>	BUS_WIDTH	R/W	0x0	—	Requested update to BUS_WIDTH.
<39:36>	—	RAZ	0x0	—	Reserved.
<35:32>	POWER_CLASS	R/W	0x0	—	Requested update to POWER_CLASS.
<31:16>	CLK_HI	R/W	0x9C4	—	Requested update to CLK_HI.
<15:0>	CLK_LO	R/W	0x9C4	—	Requested update to CLK_LO.

Cavium Confidential For Bowen Wang AM 10/27/2015

MIO eMMC DMA Configuration Register MIO_EMM_DMA

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:62>	—	RAZ	0x0	—	Reserved.
<61:60>	BUS_ID	R/W	0x0	—	Bus ID. Specifies the eMMC bus.
<59>	DMA_VAL	R/W/H	0	—	Software writes this bit to a 1 to indicate that hardware should perform the DMA transfer. Hardware clears this bit when the DMA operation completes or is terminated.
<58>	SECTOR	R/W/H	0	—	Sector addressing. Specifies that CARD_ADDR and eMMC are using sector addressing (512B).
<57>	DAT_NULL	R/W	0	—	Do not perform any eMMC commands. A DMA read request returns all 0s. A DMA write operation tosses the data. In the case of a failure, this can be used to unwind the DMA engine.
<56:51>	THRES	R/W	0x0	—	Threshold. Specifies the number of eight-byte blocks of data that must exist in the DBUF before starting the 512-byte block transfer. Zero indicates to wait for the entire block.
<50>	REL_WR	R/W	0	—	Sets the reliable write parameter when performing CMD23 (SET_BLOCK_COUNT) for a multiple block.
<49>	RW	R/W	0	—	Read/write bit (0 = read, 1 = write)
<48>	MULTI	R/W	0	—	Perform operation using a multiple-block command instead of a series of single-block commands.
<47:32>	BLOCK_CNT	R/W/H	0x0	—	Block count. Indicates the number of blocks to read/write. Hardware decrements the block count after each successful block transfer.
<31:0>	CARD_ADDR	R/W/H	0x0	—	Data address. The data address for media ≤ 2GB is a 32-bit byte address, and the data address for media > 2GB is a 32-bit sector address (512B). Hardware advances the card address after each successful block transfer by 512 for byte addressing and by 1 for sector addressing.

Cavium Confidential
Bowen Yang
ALU
02/27/2014

MIO eMMC Command Register
MIO_EMM_CMD

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:62>	—	RAZ	0x0	—	Reserved
<61:60>	BUS_ID	R/W	0x0	—	Bus ID. Specifies the eMMC bus.
<59>	CMD_VAL	R/W/H	—	—	Request valid. Software writes this bit to a 1. Hardware clears it when the operation completes.
<58:56>	—	RAZ	0x0	—	Reserved
<55>	DBUF	R/W	0	—	Data buffer. Specifies the data buffer to be used for a block transfer.
<54:49>	OFFSET	R/W	0x0	—	Debug only. Specify the number of 8-byte transfers used in the command. Value is 64-OFFSET. The block transfer still starts at the first byte in the 512-byte data buffer. Software must ensure CMD16 has updated the card block length.
<48:43>	—	RAZ	0x0	—	Reserved
<42:41>	CTYPE_XOR	R/W	0x0	—	Command type override; typically zero. Value is XORed with the default command type. See Table 29–3 for command types per command index. Types are: 0x0 = No data 0x1 = Read data into Dbuf 0x2 = Write data from Dbuf 0x3 = Reserved
<40:38>	RTYPE_XOR	R/W	0x0	—	Response type override; typically zero. Value is XORed with default response type. See Table 29–3 for response types per command index. Types are: 0x0 = No response 0x1 = R1, 48 bits 0x2 = R2, 136 bits 0x3 = R3, 48 bits 0x4 = R4, 48 bits 0x5 = R5, 48 bits 0x6, 0x7 = Reserved
<37:32>	CMD_IDX	R/W/H	0x0	—	eMMC command.
<31:0>	ARG	R/W/H	0x0	—	eMMC command argument.

Cavium Confidential For Bowen Zhang ALU 02/27/2014

MIO eMMC Response Status Register
MIO_EMM_RSP_STS

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:62>	—	RAZ	0x0	—	Reserved
<61:60>	BUS_ID	RO/H	0	—	Bus ID. Specifies the eMMC bus to which the response status corresponds
<59>	CMD_VAL	RO/H	0	—	Read-only copy of MIO_EMM_CMD[CMD_VAL]. CMD_VAL = 1 indicates that a direct operation is in progress.
<58>	SWITCH_VAL	RO/H	0	—	Read-only copy of MIO_EMM_SWITCH[SWITCH_EXE]. SWITCH_VAL = 1 indicates that a switch operation is in progress.
<57>	DMA_VAL	RO/H	0	—	Read-only copy of MIO_EMM_DMA[DMA_VAL]. DMA_VAL = 1 indicates that a DMA operation is in progress.
<56>	DMA_PEND	RO/H	0	—	DMA transfer pending. Indicates that the DMA engine has a pending transfer resulting from an error. Software can resume the transfer by writing MIO_EMM_DMA[DMA_VAL] = 1. Software can terminate the transfer by writing MIO_EMM_DMA[DMA_VAL] = 1 and MIO_EMM_DMA[DAT_NULL] = 1. Hardware then clears DMA_PEND and perform the DMA operation.
<55:29>	—	RAZ	0x0	—	Reserved
<28>	DBUF_ERR	RO/H	0	—	Data-buffer error. For CMD_TYPE = 0x1, indicates that DMA read data arrived from the card without a free DBUF.
<27:24>	—	RAZ	0x0	—	Reserved
<23>	DBUF	RO/H	0	—	Indicates the data-buffer corresponding to the most recently attempted block transfer
<22>	BLK_TIMEOUT	RO/H	0	—	Timeout waiting for read data or three-bit CRC token
<21>	BLK_CRC_ERR	RO/H	0	—	Block CRC error. For CMD_TYPE = 0x1, indicates a card read-data CRC mismatch. DBUF indicates the failing data buffer. For CMD_TYPE = 0x2, indicates card returned three-bit CRC status token indicating that the card encountered a write-data CRC check mismatch. DBUF indicates the failing data buffer.
<20>	RSP_BUSYBIT	RO/H	0	—	Debug only. eMMC protocol utilizes DAT0 as a busy signal during block writes and R1b responses.
<19>	STP_TIMEOUT	RO/H	0	—	Stop transmission response timeout
<18>	STP_CRC_ERR	RO/H	0	—	Stop transmission response had a CRC error
<17>	STP_BAD_STS	RO/H	0	—	Stop transmission response had bad status.
<16>	STP_VAL	RO/H	0	—	Stop transmission response valid
<15>	RSP_TIMEOUT	RO/H	0	—	Response timeout
<14>	RSP_CRC_ERR	RO/H	0	—	Response CRC error
<13>	RSP_BAD_STS	RO/H	0	—	Response bad status
<12>	RSP_VAL	RO/H	0	—	Response ID. See MIO_EMM_RSP_HI/LO.
<11:9>	RSP_TYPE	RO/H	0x0	—	Indicates the response type. See MIO_EMM_RSP_HI/LO.
<8:7>	CMD_TYPE	RO/H	0x0	—	eMMC command type (0x0 = no data, 0x1 = read, 0x2 = write)
<6:1>	CMD_IDX	RO/H	0x0	—	eMMC command index most recently attempted
<0>	CMD_DONE	RO/H	0	—	eMMC command completed. Once the command has completed, the status is final and can be examined by software.

Cavium Confidential
Bowen Yang
04/27/14

MIO eMMC Response Data Low Register
MIO_EMM_RSP_LO

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	DAT	RO	—	—	Command response (as per JEDEC eMMC spec and SD Specifications) RSP_TYPE = **1** DAT<63:46> 0x0 DAT<45:40> Command index DAT<39:8> Card status DAT<7:1> CRC7 DAT<0> End bit RSP_TYPE = **2** DAT<63:1> CID<63:1> or CSD>63:1> including CRC DAT<0> End bit RSP_TYPE = **3** DAT<63:46> 0x0 DAT<45:40> Check bits (0x3F) DAT<39:8] OCR register DAT<7:1] Check bits (0x7F) DAT<0> End bit RSP_TYPE = **4** DAT<63:46> 0x0 DAT<45:40> CMD39 ('10111') DAT<39:24> RCA<31:16> DAT<23> Status DAT<22:16> Register address DAT<15:8> Register contents DAT<7:1> CRC7 DAT<0> End bit RSP_TYPE = **5** DAT<63:46> 0x0 DAT<45:40> CMD40 ('10100') DAT<39:24> RCA<31:16> DAT<23> Status DAT<22:16> Register address DAT<15:8> Not defined. Can be used for IRQ data DAT<7:1> – CRC7 DAT<0> End bit

Cavium Confidential For
Bowen Yang
BOWEN ALU
02/27/2014

MIO eMMC Response Data High Register
MIO_EMM_RSP_HI

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	DAT	RO	—	—	Command response (as per JEDEC eMMC spec and SD specifications) RSP_TYPE = **1** DAT<63:0> 0x0 RSP_TYPE = **2** DAT<63:0> CID<127:64> or CSD<127:64> RSP_TYPE = **3** DAT<63:0> 0x0 RSP_TYPE = **4** DAT<63:0> 0x0 RSP_TYPE = **5** DAT<63:0> 0x0

MIO eMMC Interrupt Register
MIO_EMM_INT

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	0x0	—	Reserved
<6>	SWITCH_ERR	R/W1C/H	—	—	Switch operation encountered an error
<5>	SWITCH_DONE	R/W1C/H	—	—	Switch operation completed successfully
<4>	DMA_ERR	R/W1C/H	—	—	DMA transfer encountered an error. See MIO_EMM_RSP_STS.
<3>	CMD_ERR	R/W1C/H	—	—	Operation specified by MIO_EMM_CMD encountered an error. See MIO_EMM_RSP_STS.
<2>	DMA_DONE	R/W1C/H	—	—	All blocks have been successfully transferred between the card and the boot bus DMA engine. For a DMA read, software must wait for MIO_NDF_DMA_INT to indicate that the read data has been written to L2/DRAM.
<1>	CMD_DONE	R/W1C/H	—	—	Operation specified by MIO_EMM_CMD completed successfully.
<0>	BUF_DONE	R/W1C/H	—	—	The next 512-byte block transfer of a multiblock transfer has completed.

MIO eMMC Interrupt Enable Register
MIO_EMM_INT_EN

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	0x0	—	Reserved
<6>	SWITCH_ERR	R/W	—	—	Enable bit for MIO_EMM_INT[SWITCH_ERR].
<5>	SWITCH_DONE	R/W	—	—	Enable bit for MIO_EMM_INT[SWITCH_DONE].
<4>	DMA_ERR	R/W	—	—	Enable bit for MIO_EMM_INT[DMA_ERR]. See MIO_EMM_RSP_STS.
<3>	CMD_ERR	R/W	—	—	Enable bit for MIO_EMM_INT[CMD_ERR]. See MIO_EMM_RSP_STS.
<2>	DMA_DONE	R/W	—	—	Enable bit for MIO_EMM_INT[DMA_DONE].
<1>	CMD_DONE	R/W	—	—	Enable bit for MIO_EMM_INT[CMD_DONE].
<0>	BUF_DONE	R/W	—	—	Enable bit for MIO_EMM_INT[BUF_DONE].

Cavium Confidential For Bowering ALU 02/27/2014

MIO eMMC Watchdog Register
MIO_EMM_WDOG

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:26>	—	RAZ	0x0	—	Reserved
<25:0>	CLK_CNT	R/W	0x27EA818	—	Maximum number of CLK_CNT cycles to wait for the card to return a response, read data, or the three-bit CRC status token following write data. The following timeouts are detected: 1. Expected response to a command doesn't occur causing MIO_EMM_RSP_STS[RSP_TIMEOUT] 2. On a read command, expected data isn't returned causing MIO_EMM_RSP_STS[BLK_TIMEOUT] 3. On a multi read command, expected data isn't returned causing MIO_EMM_RSP_STS[BLK_TIMEOUT] 4. On a write command, expected token to a write block isn't received causing MIO_EMM_RSP_STS[BLK_TIMEOUT] 5. If a stop command is issued by the hardware and no response is returned causing MIO_EMM_RSP_STS[STP_TIMEOUT] Issues this timeout doesn't cover are stalls induced by the card that are not limited by the specifications. For example, when a write multi command is issued to the card and a block (not the last) is transferred the card can stall the CN71XX by forcing emmc_data<0> low for as long as it wants to free up buffer space. The second case is when the last block of a write or multi write command is being transferred and the card elects to perform some background tasks. The same stall mechanism with emmc_data<0> is used but this can last for an extended time period.

MIO eMMC Sampling Register
MIO_EMM_SAMPLE

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:26>	—	RAZ	0x0	—	Reserved.
<25:16>	CMD_CNT	R/W	0x0	0x0	Number of coprocessor-clock cycles before the eMMC clock rising edge to sample the command pin.
<15:10>	—	RAZ	0x0	—	Reserved.
<9:0>	DAT_CNT	R/W	0x0	0x0	Number of coprocessor-clock cycles before the eMMC clock edge to sample the data pin.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO eMMC Status Mask Register
MIO_EMM_STS_MASK

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	0x0	—	Reserved.
<31:0>	STS_MSK	R/W	0xE4390080	—	Any bit set in STS_MSK causes the corresponding bit in the card status to be considered when computing response bad status.

MIO eMMC Relative Card Address Register
MIO_EMM_RCA

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	0x0	—	Reserved
<15:0>	CARD_RCA	R/W/H	0x1	0x1	Whenever software performs CMD7, hardware updates CARD_RCA with the relative card address from the MIO_EMM_CMD[ARG], unless the operation encounters an error.

MIO eMMC Data Buffer Address Register
MIO_EMM_BUF_IDX

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:17>	—	RAZ	0x0	—	Reserved
<16>	INC	R/W	0	—	Automatically advance BUF_NUM/OFFSET after each access to MIO_EMM_BUF_DAT. Wraps after the last offset of the last data buffer.
<15:7>	—	RAZ	0x0	—	Reserved
<6>	BUF_NUM	R/W	0	—	Specify the data buffer for the next access to MIO_EMM_BUF_DAT.
<5:0>	OFFSET	R/W	0x0	—	Specify the 8B data buffer offset for the next access to MIO_EMM_BUF_DAT.

MIO eMMC Data Buffer Access Register
MIO_EMM_BUF_DAT

See Table 29–5 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	DAT	R/W	—	—	Direct access to the 1KB data buffer memory. Address specified by MIO_EMM_BUF_IDX.

Cavium Confidential For Bowen Yang ALU 08/7/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 30

NAND Flash Interface Unit

This section contains the following subjects:

- Overview
- NAND Flash Access Via the NDF_CMD Queue
- Detailed NDF_CMD Command Descriptions
- NDF_CMD Command Examples
- NAND Flash Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The CN71XX boot-bus hardware can directly attach to NAND flash parts. The NAND flash interface can service up to eight flash memory devices. The interface and the controller are flexible and can therefore attach to many different NAND flash parts.

This chapter focuses on NAND flash boot-bus connections, and is a companion to Chapter 25, Boot Bus, which describes the other boot-bus features that share the same boot-bus pins. In some cases, you may find it advantageous to refer to both chapters simultaneously. The boot-bus logic arbitrates for access to the boot-bus interface between the NAND controller and other accesses.

The boot bus supports the following features with directly attached NAND Flash devices:

- eight chip-select signals (shared with non-NAND boot-bus devices)
- highly software-programmable internal NAND controller
- DMA engine data movement between NAND interface and the CN71XX L2/DRAM
- eight-bit data width

The CN71XX NAND flash controller controls the boot-bus signals specified in Table 30–1. For convenience, this list shows how the signals can map to ONFI 1.0 and ONFI 2.0 devices. Note that this does not imply that CN71XX supports only NAND parts that follow the ONFI standard.

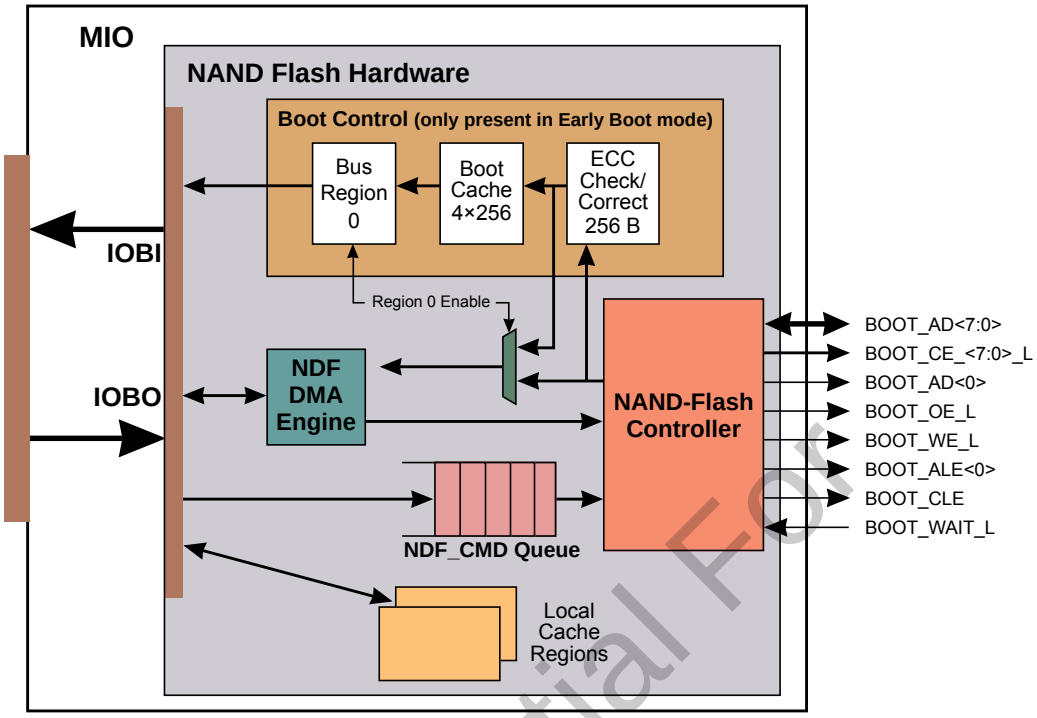
Table 30–1 NAND Flash Signals

CN71XX Signal	Corresponding Signal	
	ONFI 1.0	ONFI 2.0
Bidirectional Signals		
BOOT_AD<7:0>	IO	IO
BOOT_AD<0>	N/A	DQS
Output Signals		
BOOT_CE_<7:0>_L	CE#	CE#
BOOT_ALE<0>	ALE	ALE
BOOT_CLE	CLE	CLE
BOOT_OE_L	RE#	W/R#
BOOT_WE_L	WE#	CLK
BOOT_DIR	N/A	N/A
Input Signal		
BOOT_WAIT_L	R/B#	R/B#

Figure 30–1 shows the conceptual boot-bus hardware most relevant to direct NAND flash boot bus access.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 30–1 NAND Flash Interface



The design includes a NAND flash controller that drives the boot-bus pins appropriately for NAND flash devices. During normal operation, software directs the NAND flash controller via the NDF_CMD queue. During boot, the NAND flash controller also services references to boot-bus region 0.

The NAND flash controller can drive up to eight different chip-select signals. (The eight available chip-select signals may be shared between NAND flash and other boot-bus components.) With the commands in the NDF_CMD queue, software controls the detailed behavior of the NAND controller. With proper software control, the NAND controller can handle a wide variety of NAND flash devices, including many parts that conform to the ONFI specifications, and many others.

Software also directs the NDF DMA engine to move data between L2/DRAM and the NAND flash controller as necessary.

CN71XX does not support booting directly from NAND devices.

Cavium Confidential For Bowen Yang at U 02/27/2017

30.1 NAND Flash Access Via the NDF_CMD Queue

Figure 30–1 shows the NDF_CMD queue, which is a key component of the CN71XX NAND flash boot-bus logic. The NDF_CMD queue directly drives all detailed pin behavior when CN71XX is directly-connected to NAND flash parts. The remainder of this section describes the architecture of the NDF_CMD queue and its interactions with other CN71XX boot bus components. Section 30.2 describes the individual NDF_CMD-queue commands. Section 30.3 describes some example NDF_CMD sequences.

The NDF_CMD queue is enabled whenever either NDF_MISC[BT_DMA] or NDF_MISC[BT_DIS] is set. NDF_MISC[BT_DMA] should be cleared to zero for normal operations. When booting from a NAND flash device, NDF_MISC[BT_DIS] must never be set simultaneously with MIO_BOOT_REG_CFG0[EN]. Otherwise, NDF_MISC[BT_DIS] should be set to enable NDF DMA transactions outside of early boot.

The queue is 1536 bytes. NDF_MISC[BT_DIS] should not be changed when any NDF_CMD commands are present (i.e. NDF_MISC[BT_DIS] should only be changed when NDF_MISC[FR_BYT] is at its maximum possible value, given the current NDF_MISC[BT_DIS]).

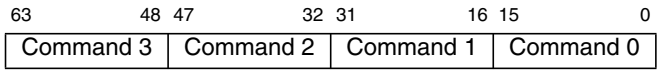
Also, NDF_MISC[RST_FF] should be set, then cleared immediately following the NDF_MISC[BT_DIS] change in value to clear the NDF_CMD queue for subsequent use.

NDF_CMD commands are each a multiple of 16 bits. An individual command may be as small as 16 bits and as large as 128 bits. The start of all commands must be at a naturally-aligned 16-bit boundary. Only the WAIT_STATUS* commands, whose start must be at a naturally-aligned 64-bit boundary, is further constrained.

Software fills the NDF_CMD queue via 64-bit store operations to the NDF_CMD register. Each 64-bit store operation can contain from one to four whole or partial commands. Commands can straddle multiple NDF store operations.

Multiple commands are packed into the 64-bit NDF_CMD queue store operations in little-endian format. Figure 30–2 shows an example with four 16-bit commands packed into one 64-bit NDF store operation.

Figure 30–2 Sixty-Four-bit NDF Store Operation



Command 0 is executed first and command 3 is executed last.

Software writes the NDF_CMD queue with a sequence of commands that implement NAND flash transactions. Each sequence must include BUS_ACQ/BUS_REL commands. NAND flash commands arbitrate with other CN71XX boot-bus users for access to the boot bus pins. The NAND flash controller arbitrates for and releases the boot-bus pins when it executes the BUS_ACQ/BUS_REL commands, respectively.

Cavium Confidential For Bowen Yan At D 02/27/2014

Boot-bus arbitration opens up the possibility of deadlock. For example, suppose that the NAND flash controller has successfully acquired control of the boot-bus pins, but cannot possibly release them because software has not yet written the BUS_REL to the NDF_CMD queue. Now suppose the CN71XX core needs to use the boot bus to service a load request from the core. The core can't make progress because the NAND controller cannot release the boot bus, and the NAND controller can't make progress because the core has not yet added the BUS_REL command.

The CN71XX solution to this deadlock is the NDF_DRBELL. The NAND flash controller does not attempt to acquire the boot bus until it has a doorbell. software should "ring the doorbell" once for each BUS_ACQ/BUS_REL pair after it adds the pair to the NDF_CMD queue. The value written to NDF_DRBELL is the number of doorbells, so software can ring the doorbell multiple times with a single NDF_DRBELL store.

Many NAND flash transactions require data movement. Neither the NDF_CMD queue nor the NAND flash controller can independently move data. The NDF DMA engine, shown in Figure 30–1, is also required. The MIO_NDF_DMA_CFG register controls the NDF DMA engine. When a NAND flash transaction requires data movement (i.e. when the command sequence contains a RD*_CMD or WR_CMD command), software needs to program the NDF DMA engine in conjunction with adding the NDF_CMD commands in order to complete the transaction.

Typically, there is a one-to-one correspondence between the NAND flash transaction and an NDF DMA-engine programming, though it is possible to have multiple NAND flash transactions per NDF DMA programming and to have multiple NDF DMA programmings per NAND flash transaction. The aforementioned deadlock may be a concern, so software may need to delay the NDF_DRBELL until it has both loaded the transaction into the NDF_CMD queue and programmed the NDF DMA engine, and multiple NDF DMA programmings per NAND flash transaction may be impossible.

The NAND flash controller transfers data back and forth between the NDF DMA engine in 8-byte words. If all DATA_BYTES to all RD*_CMD or WR_CMD commands in the bus transaction are a multiple of 8 bytes, then the total NDF DMA-transfer length in 8-byte words is the sum of the DATA_BYTES of the individual RD*_CMD or WR_CMD commands divided by 8. If a single NDF DMA programming is used for the transaction, the transfer length (in 8-byte words) is MIO_NDF_DMA_CFG[SIZE]+1. In this case, the data bytes moved for the transaction on the boot bus is exactly the data bytes moved to/from L2/DRAM.

Cavium Confidential For Boren Yang 02/7/2014

If an individual RD*_CMD or WR_CMD command has a DATA_BYTES value that is not a multiple of 8, the NAND flash controller moves DATA_BYTES on the boot bus, but rounds up the number of bytes moved between it and the NDF DMA engine to be a multiple of 8 bytes. In these cases, the NDF DMA engine moves more bytes than the NAND flash controller moves on the boot bus.

- For a RD*_CMD command, the NAND flash controller inserts extra bytes as necessary, and the NDF DMA engine writes the extra bytes to L2/DRAM. These extra bytes have an unpredictable value.
- For a WR_CMD command, the NDF DMA engine reads extra bytes from L2/DRAM, and the NDF DMA engine discards them.

NOTE: When multiple commands are used, the size round up happens separately for each individual command.

CN71XX clears MIO_NDF_DMA_CFG[EN] and sets MIO_NDF_DMA_INT[DONE] after the NDF DMA engine completes.

Software can poll MIO_NDF_DMA_INT[DONE] to determine completion, or MIO_NDF_DMA_INT_EN[DONE] can be set to enable interrupts. When enabled, these interrupts assert CIU_INT*_SUM*[BOOTDMA] (refer to Chapter 13).

There are other interrupts that software may find useful. CN71XX sets NDF_INT[EMPTY] to 1 each time the NDF_CMD queue becomes empty. CN71XX sets NDF_INT[FULL] each time the NDF_CMD queue becomes full. For debug and errors, CN71XX stops executing a command and sets NDF_INT[WDOG] whenever an exit condition of a command is not satisfied for too long. The only commands that can set this interrupt are WAIT, WAIT_STATUS and WAIT_STATUS_ALE.

When NDF_INT interrupts are enabled (via NDF_INT_EN), they assert CIU_INT*_SUM*[NAND] (refer to Chapter 13).

The NDF DMA engine adds to MIO_NDF_DMA_CFG[ADR] and counts MIO_NDF_DMA_CFG[SIZE] down as it moves data. The NDF DMA engine considers the DMA operation to be complete when MIO_NDF_DMA_CFG[SIZE] counts down to all 1s. If MIO_NDF_DMA_CFG[CLR] is set to 1, the NDF DMA engine also considers the DMA operation complete when the NAND flash controller executes a BUS_REL command (i.e. when the NAND flash controller releases the boot bus). In all cases, the MIO_NDF_DMA_CFG[SIZE] and MIO_NDF_DMA_CFG[ADR] changes indicate the number of 64-bit words transferred.

Software should ensure that the NAND flash controller and NDF DMA engine, and their respective NDF_CMD queue and MIO_NDF_DMA_CFG programmings are in sync.

The following are examples of things that can possibly lead to problems:

- If MIO_NDF_DMA_CFG[CLR] is clear, it is possible for the NDF DMA engine to remain active waiting for data after the NDF_CMD queue clears of all commands. The NDF DMA engine will wait forever for more data movement in this case.
- If MIO_NDF_DMA_CFG[SIZE] is too small, it is possible for the NDF DMA engine to complete before the NDF_CMD queue is drained. The NAND flash controller will wait forever for more data movement in this case.

Cavium Confidential For Bowen Yang @ AMD

30.2 Detailed NDF_CMD Command Descriptions

Each command contains a four-bit major opcode in the least-significant four bits of the command. A given command is always a multiple of 16-bits, and may be as small as 16-bits and as large as 128-bits.

The remainder of this section describes the detailed behavior of the commands as the NAND flash controller executes them from the NDF_CMD queue. Section 30.1 describes global concepts regarding the NDF_CMD queue, while Section 30.3 shows some example usages of the NDF_CMD queue.

The detailed behavior includes timing. Note that the execution of any single command may be delayed by contention for internal resources. Generally speaking, the specified timings are the minimum timings.

The execution of many of the specified commands causes the NAND flash controller to change the values of boot-bus pins. For a list of the boot-bus pins controlled by the NAND flash controller, refer to Table 30–1.

System designs can optionally connect the R/B# pin of NAND flash devices to BOOT_WAIT_L. Note that CN71XX supplies only one BOOT_WAIT_L pin, not one per NAND flash device. R/B# is open-drain, so it is feasible to connect the R/B# of multiple NAND parts to BOOT_WAIT_L. However, most system designs may not connect to BOOT_WAIT_L, which implies that either static delays (via the WAIT command) or status-register-read operations (via the WAIT_STATUS or WAIT_STATUS_ALE commands) must be used to ensure data availability.

Note that the CN71XX NAND flash controller internally applies hysteresis to the BOOT_WAIT_L pin before it interprets it. Whenever the NAND controller samples BOOT_WAIT_L asserted, it immediately internally considers BOOT_WAIT_L to be asserted. But when the NAND controller internally considers BOOT_WAIT_L to be asserted, it will only change to internally consider BOOT_WAIT_L to be deasserted after it samples the actual BOOT_WAIT_L pin as deasserted for NDF_MISC[WAIT_CNT] coprocessor-clock cycles.

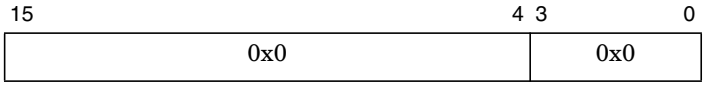
Table 30–2 shows the NAND flash commands, which are described in the remainder of this section.

Table 30–2 NAND Flash Commands

Command	Section
NOP	Section 30.2.1
SET_TM_PAR	Section 30.2.2
WAIT	Section 30.2.3
BUS_ACQ	Section 30.2.4
BUS_REL	Section 30.2.5
CHIP_EN	Section 30.2.6
CHIP_DIS	Section 30.2.7
CLE_CMD	Section 30.2.8
ALE_CMD	Section 30.2.9
RD_CMD	Section 30.2.10
RD_EDO_CMD	Section 30.2.11
WR_CMD	Section 30.2.12
WAIT_STATUS	Section 30.2.13
WAIT_STATUS_ALE	Section 30.2.13

Cavium Confidential For Bowen Yang ALU 02/27/2014

30.2.1 NOP Command

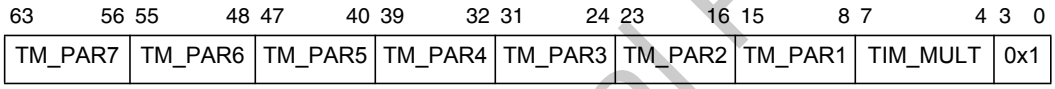


This 16-bit no-operation command does nothing except take one coprocessor-clock cycle to execute. It can be used as a pad between other commands. Pads can be used as desired, but are required in the following situations:

- At the end of a NDF-command-queue sequence to make the length of the command sequence a multiple of 64-bits.
- Prior to WAIT_STATUS* commands for correct alignment of the command.

The NOP command does not modify any boot-bus signals, nor does it respond to any boot-bus signals.

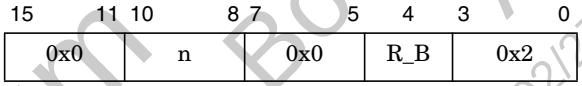
30.2.2 SET_TM_PAR Command



This 64-bit command sets seven different generalized timing parameters and one corresponding timing multiplier. Other commands use these timing parameters. SET_TM_PAR executes in one coprocessor-clock cycle; it does not change any boot-bus signals, nor does it respond to any boot bus signals.

- **TM_PAR*n*<7:0> –** timing parameter *n* (n =1–7), specifying the number of coprocessor-clock cycles. A value of 0x0 means one cycle. CN71XX scales all these parameters by 2^{TIM_MULT} when it uses any of them. There is also a fixed-value TM_PAR0, which always equals the constant 4 (before being scaled by TIM_MULT).
- **TIM_MULT<3:0> –** the timing multiplier for the seven timing parameters.

30.2.3 WAIT Command



This 16-bit command waits for a number of cycles, then can test an exit condition before completing. It does not change any boot-bus signals, nor does it respond to any boot bus signals when R_B = 0. It responds only to BOOT_WAIT_L when R_B = 1.

- **n –** Selects one of the eight available timing parameters. The current TM_PAR*n* and TIM_MULT indicate the number of cycles to wait. After the wait, CN71XX waits for the exit condition to be satisfied.
- **R_B –** indicates the exit condition

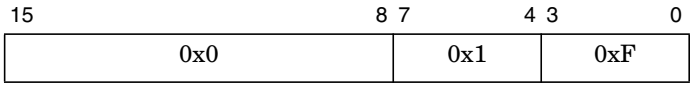
  If R_B = 0, the exit condition is satisfied after one cycle.

  If R_B = 1, the exit condition is satisfied once BOOT_WAIT_L deasserts.

The WAIT command takes at least (TM_PAR*n* << TIM_MULT) + 1 coprocessor-clock cycles to execute. Use the WAIT command to insert timing delays and/or wait until the BOOT_WAIT_L signal deasserts.

Cavium Confidential For Bowen Yang ALU 02/27/2014

30.2.4 BUS_ACQ Command



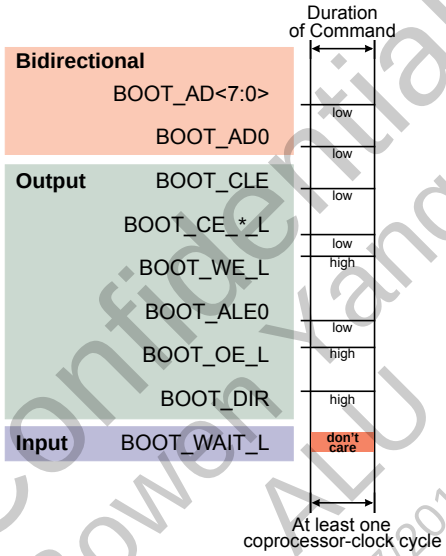
This 16-bit command first waits for a doorbell, then waits to acquire the bus. Once NDF_DRBELL ≠ 0, CN71XX decrements NDF_DRBELL by 1 and proceeds. After that, CN71XX completes the BUS_ACQ command once the NAND flash controller has won arbitration and has control of the boot bus. BUS_ACQ takes at least one coprocessor-clock cycle to execute.

A BUS_ACQ/BUS_REL pair defines a NAND flash transaction.

Use this command before any other command that changes any boot-bus signal. Only the NOP, SET_TM_PAR, and WAIT commands may execute without control of the boot bus. BUS_ACQ is necessary to hold off other CN71XX boot-bus users while the NAND flash controller completes its NAND flash transaction.

BUS_ACQ leaves the boot-bus signals in the state indicated in Figure 30–3.

Figure 30–3 BUS_ACQ Signal Timing



The prior boot-bus user could have been a region read/write operation, a MWDMA read/write operation, or a prior NAND-flash-controller transaction. In the region and MWDMA cases, referring to the timing diagrams in Section 28.4, the boot bus can be acquired immediately following the end of the T_{PAUSE} interval shown in the timing diagrams. In the NAND-flash-controller case, the boot-bus signal timing prior to the BUS_ACQ command depends on the previous NAND flash transaction.

Restrictions:

- A corresponding BUS_REL command must execute with each BUS_ACQ. Otherwise, the boot bus will be locked up.
- The NAND flash controller must not have control of the boot bus when a BUS_ACQ is executed. There must not be another BUS_ACQ command between the BUS_ACQ/BUS_REL pair.
- The inability to release control of the boot bus can cause deadlocks (refer to Section 30.1).

Cavium Confidential For Bowen Yang ALU 04/27/2014

- It may be necessary to insert a WAIT command after the BUS_ACQ for the proper timing margins relative to the actions of the prior boot-bus user.

30.2.5 BUS_REL Command

15 — 8	7 — 4	3 — 0
0x0	0x0	0xF

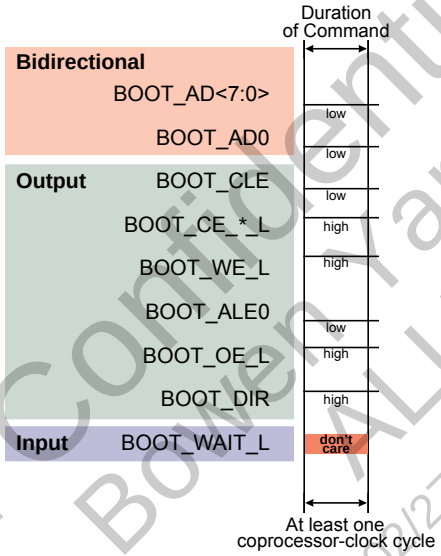
This 16-bit command releases control of the boot bus to the other boot bus users. Refer to the BUS_ACQ description in Section 30.2.4 for related information. The BUS_REL command executes in one coprocessor-clock cycle.

The BUS_REL command terminates the transaction defined by the BUS_ACQ/BUS_REL pair. This termination may cause an active NDF DMA engine to stop the transfer if the transaction includes any of the RD*_CMD or WR_CMD commands. Refer to Section 30.1 and to the descriptions regarding MIO_NDF_DMA_CFG[CLR].

Use this command to release the boot bus for other boot-bus users.

BUS_REL leaves the boot-bus signals in the state indicated in Figure 30–4.

Figure 30–4 BUS_REL Signal Timing



The next boot-bus user could be a region read/write operation, a MWDMA read/write operation, or a prior NAND-flash-controller transaction. In the region and MWDMA cases, referring to the timing diagrams in Sections 28.4 and 28.6, the BUS_REL may be immediately prior to the BOOT_DMACK* asserting edge shown in all the timing diagrams. In the NAND-flash-controller case, the boot-bus signal timing prior to the BUS_ACQ depends on the previous NAND flash transaction.

Restrictions:

- BUS_REL must only execute when the NAND flash controller has control of the bus.
- All chip-enable signals must be deasserted prior to a BUS_REL command. The prior commands in the transaction must also leave all the boot-bus signals in the state indicated in Figure 30–4.

Cavium Confidential For Bowen Yang ALU 02/27/2014

- It may be necessary to insert a WAIT command prior to the BUS_REL for the proper timing margins relative to the actions of the next boot-bus user.

30.2.6 CHIP_EN Command

15 10	9 8	7	6 4	3 0
0x0	DATA_BUS_WIDTH	1	CE	0x3

This command asserts a chip-enable pin. The chip-enable pin remains asserted and the bus-width selection remains in effect until a subsequent CHIP_DIS command. CHIP_EN executes in one coprocessor-clock cycle.

- **CE –** specifies one of the eight chip-select signal to assert.
- **DATA_BUS_WIDTH –** specifies the bus width

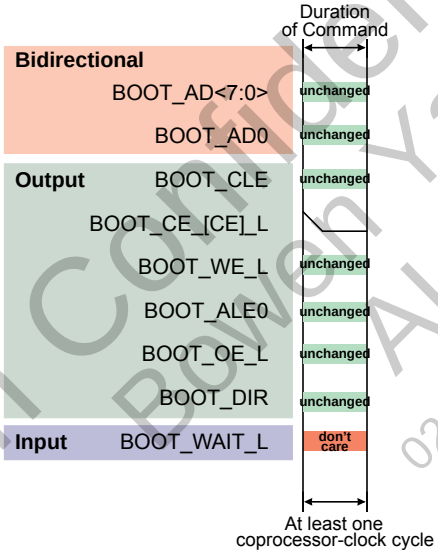
  01 = selects eight-bit IOI width (BOOT_AD<7:0>)

  00, 10, 11 = reserved

Use CHIP_EN to select the NAND flash device to communicate with, before any other commands communicate with the NAND flash device.

Except for the selected BOOT_CE_[CE]_L chip-enable signal, this command does not change any boot-bus signals nor does it respond to any boot-bus signals (refer to Figure 30–5).

Figure 30–5 CHIP_EN Signal Timing

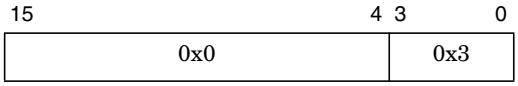


Restrictions:

- The selected BOOT_CE_*n*_L chip-enable signal should be attached to a NAND flash device.
- The NAND flash controller must have control of the boot bus at the time that the CHIP_EN command executes. A BUS_ACQ/BUS_REL command pair should surround this CHIP_EN command.
- At most one chip-enable signal can be asserted at any time. The CHIP_EN command must have a corresponding CHIP_DIS command. There must not be another CHIP_EN command between this CHIP_EN/CHIP_DIS command pair.

Cavium Confidential For Bowen Yang ALU 02/27/2014

30.2.7 CHIP_DIS Command

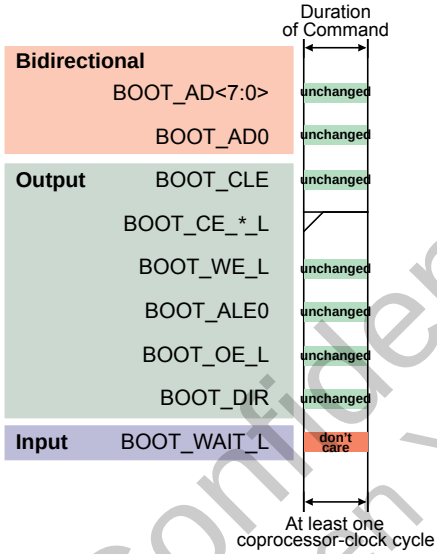


This command deasserts all chip-enable signals, and is generally paired with a preceding CHIP_EN command. Refer to Section 30.2.6 for related information. The CHIP_DIS command executes in one coprocessor-clock cycle.

Use CHIP_DIS to deassert the chip-enable signals.

Except for the one asserted BOOT_CE_*_L chip-enable signal, this command does not change any boot-bus signals nor does it respond to any boot-bus signals (refer to Figure 30–6).

Figure 30–6 CHIP_DIS Signal Timing



Restrictions:

- The NAND flash controller must have control of the boot bus at the time that the CHIP_EN command executes. A BUS_ACQ/BUS_REL command pair should surround this CHIP_DIS command.
- At most one chip enable can be asserted at any time. Each CHIP_EN command must have a corresponding CHIP_DIS command. There must not be another CHIP_EN command between this CHIP_EN/CHIP_DIS command pair.

Cavium Confidential For Bowen Yang ALU 02/27/2014

30.2.8 CLE_CMD Command

31 25	24 22	21 19	18 16	15 8	7 4	3 0
0x0	clen3	clen2	clen1	CMD_DATA	0x0	0x4

This 32-bit command moves a NAND command to the NAND flash device by transitioning the BOOT_CLE and BOOT_WE_L pins and driving the command on the BOOT_AD pins.

- **CMD_DATA –** specifies the command that is sent on the BOOT_AD pins. The CMD_DATA is valid on the BOOT_AD pins when BOOT_CLE is asserted.
- **clen1 –** specifies the time interval between when BOOT_CLE asserts and BOOT_WE_L asserts (TM_PARclen1 << TIM_MULT coprocessor-clock cycles).
- **clen2 –** specifies the time interval that BOOT_WE_L remains asserted (TM_PARclen2 << TIM_MULT coprocessor-clock cycles).
- **clen3 –** specifies the time interval between the deassertion of BOOT_WE_L and BOOT_CLE (TM_PARclen3 << TIM_MULT coprocessor-clock cycles).

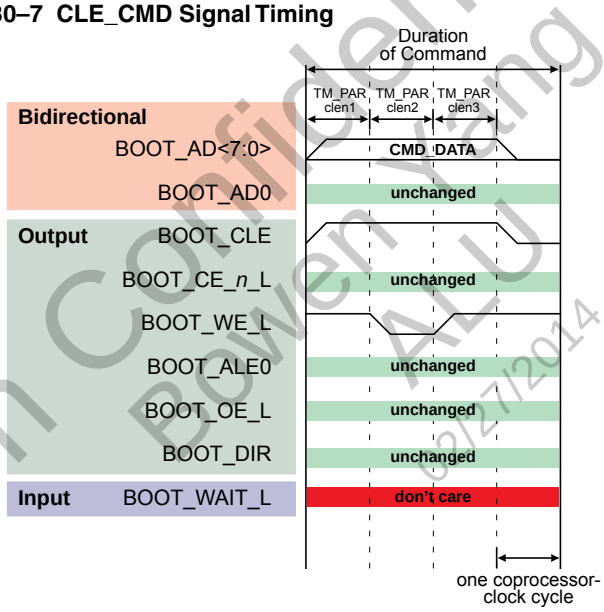
The CLE_CMD command requires

((TM_PARclen1 + TM_PARclen2 + TM_PARclen3) << TIM_MULT) + 1

coprocessor-clock cycles to execute.

Figure 30–7 shows the boot-bus-signal transitions during CLE_CMD execution.

Figure 30–7 CLE_CMD Signal Timing



Restrictions:

- The CLE_CMD command must be surrounded by a CHIP_EN/CHIP_DIS pair.

Cavium Confidential For Bowen Yang ALU 8/27/2014

30.2.9 ALE_CMD Command

(present when ADR_BYTE_NUM = 7, 8)		(present when ADR_BYTE_NUM = 5–8)		(present when ADR_BYTE_NUM = 3–8)		(always present)	
95 88	87 80	79 72	71 64	63 56	55 48	47 40	39 32
ADR_BYT8	ADR_BYT7	ADR_BYT6	ADR_BYT5	ADR_BYT4	ADR_BYT3	ADR_BYT2	ADR_BYT1

31 28	27 25	24 22	21 19	18 16	15 12	11 8	7 4	3 0
0x0	alen4	alen3	alen2	alen1	0x0	ADR_BYTE_NUM	0x0	0x5

This 48-bit to 96-bit command moves a sequence of address bytes to the NAND flash device by transitioning the BOOT_ALE<0> and BOOT_WE_L pins and driving the address bytes on the BOOT_AD pins.

- **ADR_BYTE_NUM –** the number of address bytes sent. Legal values 1-8.
- **ADR_BYTE_[ADR_BYTE_NUM:1] -** The address bytes sent on the BOOT_AD bus. The order is:

  ADR_BYTE_1 (first), ... ADR_BYTE_n (last).

  The ADR_BYTE_* values are valid on the BOOT_AD pins from the asserting edge of BOOT_WE_L to either the next BOOT_WE_L asserting edge or the deasserting edge of BOOT_ALE<0>, whichever is first.

  When NDF_MISC[BT_DMA] = 1, refer to the restriction specified below.

- **alen1 -** specifies the time interval between when BOOT_ALE asserts and BOOT_WE_L first asserts (TM_PAR[alen1] << TIM_MULT coprocessor-clock cycles).
- **alen2 -** specifies the time interval that BOOT_WE_L asserts (TM_PAR[alen2] << TIM_MULT coprocessor-clock cycles).
- **alen3 -** when ADR_BYTE_NUM > 1, specifies the time interval between the prior BOOT_WE_L deassertion and the next BOOT_WE_L assertion (TM_PAR[alen3] << TIM_MULT coprocessor-clock cycles).
- **alen4 -** specifies the time interval between when BOOT_WE_L last deasserts and BOOT_ALE deasserts (TM_PAR[alen4] << TIM_MULT coprocessor-clock cycles).

The ALE_CMD command is 48, 64, 80, or 96 bits for ADR_BYTE_NUM values of 1-2, 3-4, 5-6, or 7-8, respectively. When ADR_BYTE_NUM is odd, CN71XX does not use the last ADR_BYTE_[ADR_BYTE_NUM+1], though it is present in the command.

The ALE_CMD command takes

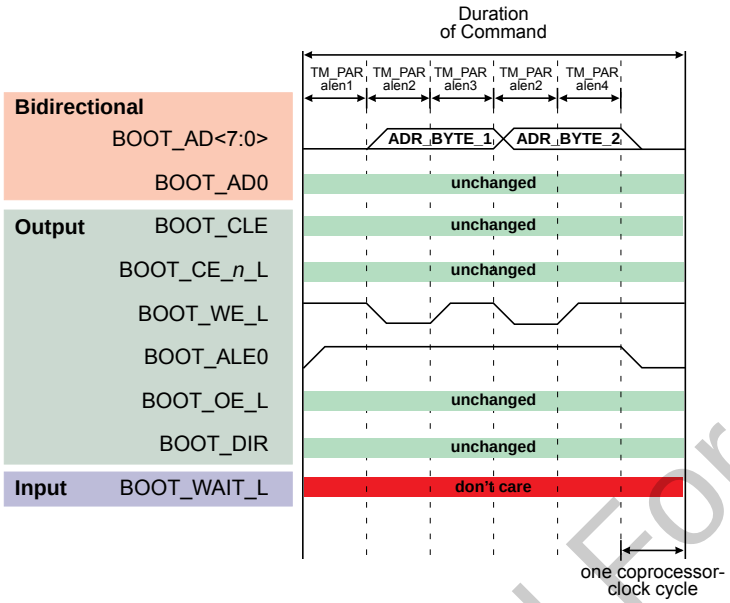
$$((TM_PARalen1 + TM_PARalen2 \times ADR_BYTE_NUM + TM_PARalen3 \times (ADR_BYTE_NUM - 1) + TM_PARalen4) << TIM_MULT) + 1$$

coprocessor-clock cycles to execute.

Figure 30–8 shows the boot-bus-signal transitions during ALE_CMD execution with ADR_BYTE_NUM = 2.

Cavium Confidential For Bowen Yang ALL 6/27/2017

Figure 30–8 ALE_CMD Signal Timing



Restrictions:

- The ALE_CMD command must be surrounded by a CHIP_EN/CHIP_DIS pair.
- When NDF_MISC[BT_DMA] = 1 (during early-boot mode), ALE_CMD commands must position the address at the beginning of a NAND flash cache block (refer to Section 28.1).

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

30.2.10 RD_CMD Command

31 29	28 26	25 23	22 20	19 4	3 0
rdn4	rdn3	rdn2	rdn1	DATA_BYTES	0x9

This 32-bit command reads a sequence of eight-bit data from the NAND flash device by transitioning BOOT_OE_L and reading the bytes on the BOOT_AD pins.

- **DATA_BYTES -** The number of data bytes on the bus. There are DATA_BYTES transfers on the BOOT_AD pins with an eight-bit device.

  DATA_BYTES must be a multiple of the bus width in bytes. When NDF_MISC[BT_DMA] = 1, refer to the restriction below.

- **rdn1 -** specifies the time interval from when BOOT_DIR drops (and CN71XX stops driving the BOOT_AD pins) until the first BOOT_OE_L assertion, as well as the time interval from when BOOT_DIR rises until CN71XX starts driving the BOOT_AD pins again (TM_PARrdn1 << TIM_MULT coprocessor-clock cycles).
- **rdn2 -** specifies the time interval that BOOT_OE_L asserts (TM_PARrdn2 << TIM_MULT coprocessor-clock cycles). CN71XX samples the BOOT_AD pins when it starts deasserting BOOT_OE_L
- **rdn3 -** specifies the minimum time interval from BOOT_OE_L deassertion to BOOT_OE_L reassertion (TM_PARrdn3 << TIM_MULT coprocessor-clock cycles). Also adds to the delay from the last BOOT_OE_L deassertion until CN71XX raises BOOT_DIR.
- **rdn4 -** adds to the delay from the last BOOT_OE_L deassertion until CN71XX raises BOOT_DIR (TM_PARrdn4 << TIM_MULT coprocessor-clock cycles).

The RD_CMD command takes at least

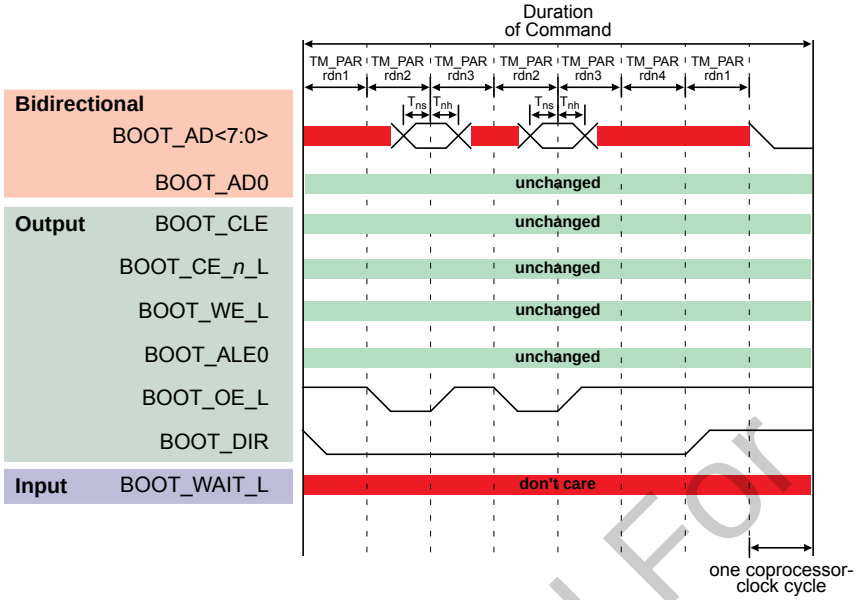
$$\left((\text{TM_PARrdn1} \times 2 + (\text{TM_PARrdn2} + \text{TM_PARrdn3}) \times \text{DATA_BYTES} + \text{TM_PARrdn4}) << \text{TIM_MULT}\right) + 1$$

coprocessor-clock cycles to execute. Note that CN71XX may add extra delay immediately prior to any BOOT_OE_L assertion if the NDF DMA engine cannot accept the data.

Figure 30–9 shows the boot-bus-signal transitions during RD_CMD command execution with DATA_BYTES = 2.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 30–9 RD_CMD Signal Timing



The CN71XX NAND flash controller transfers the bytes to the NDF DMA engine in 64-bit words. The NAND flash controller puts the eight-bit BOOT_AD values into the 64-bit words in big-endian order (most-significant first, least-significant last). (Software can swap the bytes written to L2/DRAM by the NDF DMA engine via MIO_NDF_DMA_CFG[SWAP*].) When the number of bytes transferred on the BOOT_AD pins is not a multiple of 8, the NAND flash controller pads the last 64-bit word with extra (unpredictable) words (refer to Section 30.1). The NDF DMA engine normally receives all the bytes transferred on the BOOT_AD pins. Refer to the restriction when NDF_MISC[BT_DMA] = 1.

CN71XX does not drive the BOOT_AD<7:0> signals at all during a RD_CMD command except for the last coprocessor-clock cycle.

Restrictions:

- The RD_CMD command must be surrounded by a CHIP_EN/CHIP_DIS pair.
- When NDF_MISC[BT_DMA] = 1 (during early boot mode), DATA_BYTES must be a multiple of the NAND flash cache block size (i.e. 264 bytes), and the NDF DMA engine receives 256 bytes for each NAND flash cache block, not 264. Also note that CN71XX performs ECC check/correct in this case (refer to Section 28.1).

Cavium Confidential For Power Plug ALU 9/17/2014

30.2.11 RD_EDO_CMD Command

31 29	28 26	35 23	22 20	194 3	0
rdn4	rdn3	rdn2	rdn1	DATA_BYTES	0xA

This 32-bit command reads a sequence of 8/16-bit data from the NAND flash device by transitioning BOOT_OE_L and reading the bytes on the BOOT_AD pins.

- **DATA_BYTES -** the number of data bytes on the bus. There are DATA_BYTES transfers on the BOOT_AD pins with an eight-bit device. CN71XX samples the data on the BOOT_AD bus TM_PARrdn3 << TIM_MULT coprocessor-clock cycles after each deasserting BOOT_OE_L edge.

  DATA_BYTES must be a multiple of the bus width in bytes. When NDF_MISC[BT_DMA] = 1, refer to the restriction below.

- **rdn1 -** specifies the time interval from when BOOT_DIR drops (and CN71XX stops driving the BOOT_AD pins) until the first BOOT_OE_L assertion, as well as the time interval from when BOOT_DIR rises until CN71XX starts driving the BOOT_AD pins again (TM_PARrdn1 << TIM_MULT coprocessor-clock cycles).
- **rdn2 -** specifies the time interval that BOOT_OE_L asserts (TM_PARrdn2 << TIM_MULT coprocessor-clock cycles).
- **rdn3 -** specifies the minimum time interval from BOOT_OE_L deassertion to BOOT_OE_L reassertion (TM_PARrdn3 << TIM_MULT coprocessor-clock cycles). Also picks the time interval from BOOT_OE_L deassertion to when the BOOT_AD pins are sampled.
- **rdn4 -** specifies the delay from BOOT_AD sample point until CN71XX raises BOOT_DIR (TM_PARrdn4 << TIM_MULT coprocessor-clock cycles).

The RD_EDO_CMD command takes at least

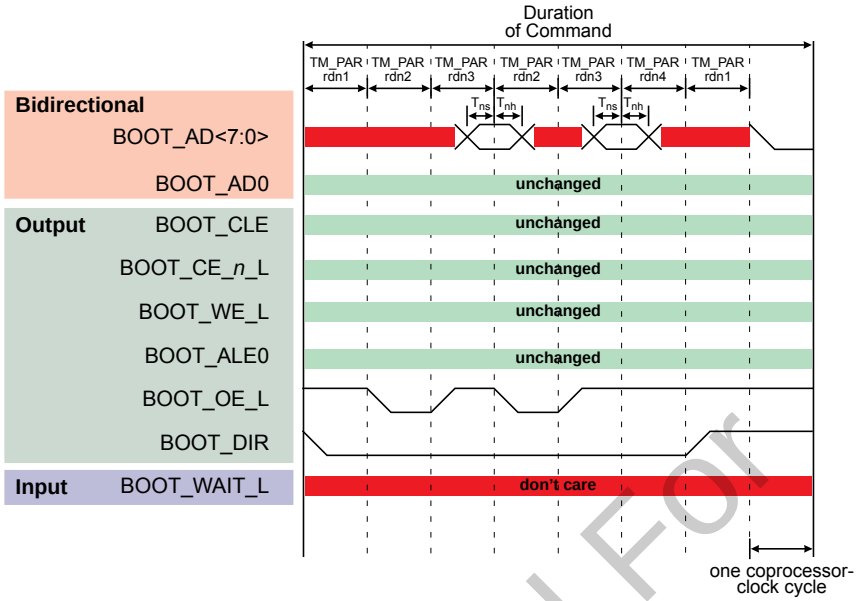
$$\left((\text{TM_PARrdn1} \times 2 + (\text{TM_PARrdn2} + \text{TM_PARrdn3}) \times \text{DATA_BYTES} + \text{TM_PARrdn4}) << \text{TIM_MULT}\right) + 1$$

coprocessor-clock cycles to execute. Note that CN71XX may add extra delay immediately prior to any BOOT_OE_L assertion if the NDF DMA engine cannot accept the data.

Figure 30–10 shows the boot-bus-signal transitions during RD_EDO_CMD execution with DATA_BYTES = 2.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 30–10 RD_EDO_CMD Signal Timing



The CN71XX NAND flash controller transfers the bytes to the NDF DMA engine in 64-bit words. The NAND flash controller puts the eight-bit BOOT_AD values into the 64-bit words in big-endian order (most-significant first, least-significant last). (Software can swap the bytes written to L2/DRAM by the NDF DMA engine via MIO_NDF_DMA_CFG[SWAP*].) When the number of bytes transferred on the BOOT_AD pins is not a multiple of 8, the NAND flash controller pads the last 64-bit word with extra (unpredictable) words (refer to Section 30.1).

The NDF DMA engine normally receives all the bytes transferred on the BOOT_AD pins. Refer to the restriction when NDF_MISC[BT_DMA] = 1.

Restrictions:

- The RD_EDO_CMD command must be surrounded by a CHIP_EN/CHIP_DIS pair.
- When NDF_MISC[BT_DMA] = 1 (during early boot mode), DATA_BYTES must be a multiple of the NAND flash cache block size (i.e. 264 bytes), and the NDF DMA engine receives 256 bytes for each NAND flash cache block, not 264. Also note that CN71XX performs ECC check/correct in this case (refer to Section 28.1).

Cavium Confidential For Bowen Yang ALU 02/27/2014

30.2.12 WR_CMD Command

31 30	29 27	26 24	23 20	19 4	3 0
0x0	wrn2	wrn1	0x0	DATA_BYTES	0x8

- **DATA_BYTES -** the number of data bytes on the bus. There are DATA_BYTES transfers on the BOOT_AD pins with an eight-bit device.
- **wrn1 -** specifies the time interval that BOOT_WE_L asserts (TM_PARwrn1 << TIM_MULT coprocessor-clock cycles).
- **wrn2 -** specifies the time interval from BOOT_OE_L deassertion to either the next BOOT_OE_L assertion, or until CN71XX drives 0s on the BOOT_AD pins again (TM_PARwrn2 << TIM_MULT coprocessor-clock cycles).

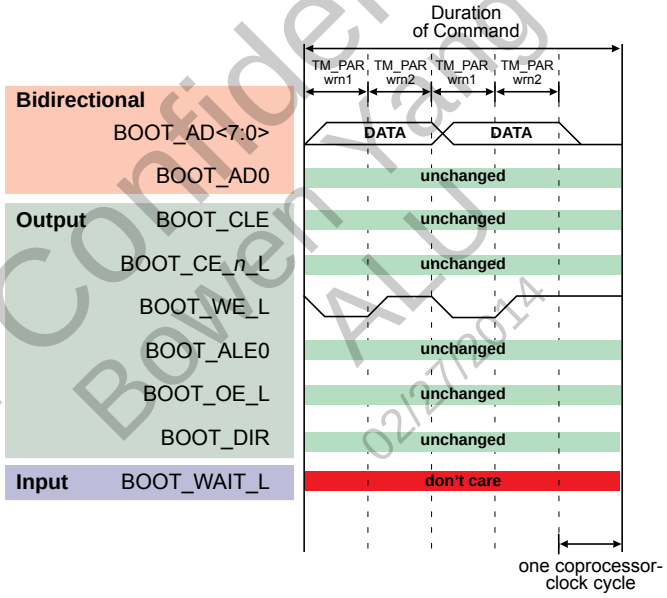
The WR_CMD command takes at least

$$\left(\left((\text{TM_PARwrn1} + \text{TM_PARwrn2}) \times \text{DATA_BYTES}\right) << \text{TIM_MULT}\right) + 1$$

coprocessor-clock cycles to execute. Note that CN71XX may add extra delay immediately prior to any BOOT_WE_L assertion if the NDF DMA engine cannot deliver the data.

Figure 30–11 shows the boot-bus-signal transitions during WR_CMD execution with DATA_BYTES = 2.

Figure 30–11 ALE_CMD Signal Timing



The NAND flash controller receives the bytes from the NDF DMA engine in 64-bit words and extracts the eight-bit BOOT_AD values from the 64-bit words in big-endian order (most-significant first, least-significant last). (Software can swap the bytes read from L2/DRAM by the NDF DMA engine via MIO_NDF_DMA_CFG[SWAP*].) When the number of bytes transferred on the BOOT_AD pins is not a multiple of 8, the NDF DMA engine delivers an entire word, and the NAND flash controller puts the bytes that it needs onto the BOOT_AD pins.

Restrictions:

Cavium Confidential For ALU Bowen Yang 02/27/2014

- The WR_CMD command must be surrounded by a CHIP_EN/CHIP_DIS pair.
- When NDF_MISC[BT_DMA] = 1 (during early boot mode), WR_CMD commands must not be used.

30.2.13 WAIT_STATUS and WAIT_STATUS_ALE Commands

WAIT_STATUS* commands provide a means to stall the command queue until a read status (WAIT_STATUS) or read status enhanced (WAIT_STATUS_ALE) returns a desired value (in some bits).

The encoding is slightly different, but the timing of WAIT_STATUS is largely equivalent to the following command sequence when its comparison succeeds:

CLE_CMD (CMD_DATA, clen1, clen2, clen3)

NOP

NOP

NOP

NOP

RD_CMD (DATA_BYTES = 1, rdn1, rdn2, rdn3, rdn4)

The timing of WAIT_STATUS_ALE is similar to this form when ADR_BYTE_NUM = 3 or 4:

CLE_CMD (CMD_DATA, clen1, clen2, clen3)

ALE_CMD (ADR_BYTE_NUM = 3 or 4, ADR_BYTE_1, ADR_BYTE_2, ADR_BYTE_3, ADR_BYTE_4, alen1, alen2, alen3, alen4)

RD_CMD (DATA_BYTES = 1, rdn1, rdn2, rdn3, rdn4)

and the timing of WAIT_STATUS_ALE is similar to this form when ADR_BYTE_NUM = 1 or 2:

CLE_CMD (CMD_DATA, clen1, clen2, clen3)

ALE_CMD (ADR_BYTE_NUM = 1 or 2, ADR_BYTE_1, ADR_BYTE_2, alen1, alen2, alen3, alen4)

NOP

RD_CMD (DATA_BYTES = 1, rdn1, rdn2, rdn3, rdn4)

The differences between the two WAIT_STATUS* commands and these command sequences are:

- The major opcode of the CLE_CMD subcommand that starts the WAIT_STATUS* instruction is 0xB rather than the 0x4 of an independent CLE_CMD subcommand.
- The DATA_BYTES field of the RD_CMD subcommand is not just 1. Instead, the WAIT_STATUS COMP_BYTE and AND_MASK fields take the position of the DATA_BYTES field.
- The RD_CMD subcommand of the WAIT_STATUS* command only reads the least-significant byte during its one data cycle.

Cavium Confidential For Bowen Yang ALU 08/27/2014

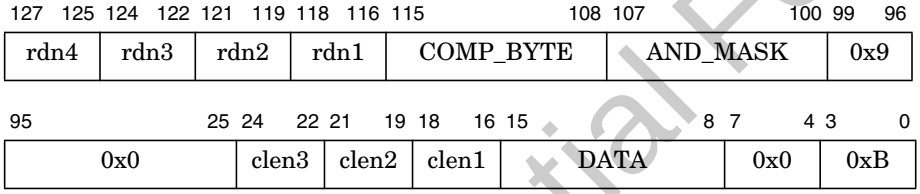
- WAIT_STATUS* does not send the result from its RD_CMD subcommand to the NDF DMA engine. Instead, WAIT_STATUS* calculates the following:

  ANDED_RESULT = AND_MASK & READ_RESULT

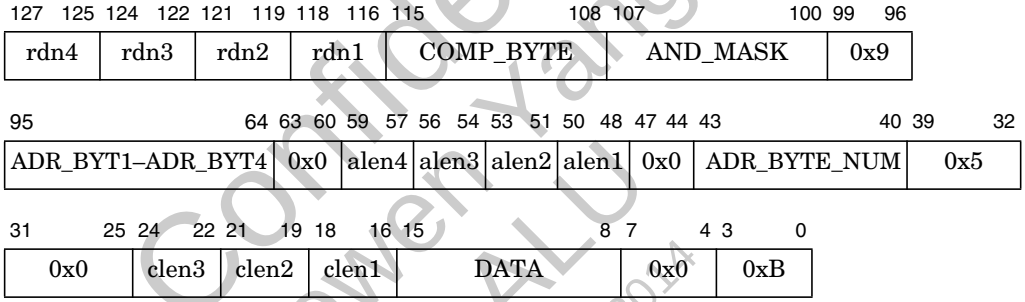
  where READ_RESULT is eight-bit result from the RD_CMD subcommand.

  - If ANDED_RESULT ≠ COMP_BYTE, then CN71XX re-executes all sub-commands of the WAIT_STATUS* instruction.
  - If ANDED_RESULT = COMP_BYTE, WAIT_STATUS* execution is complete.
- The timing has fewer coprocessor-clock cycles between each active subcommand than if the subcommands were independent commands.
- WAIT_STATUS* may be used when NDF_MISC[BT_DMA] = 1 (during early boot), and the DATA_BYTES restrictions associated with the RD_CMD command do not apply.

WAIT_STATUS Command



WAIT_STATUS_ALE Command



A WAIT_STATUS command takes:

$$\left((TM_PARclen1 + TM_PARclen2 + TM_PARclen3 + (TM_PARrdn1 \times 2) + TM_PARrdn2 + TM_PARrdn3 + TM_PARrdn4) << TIM_MULT\right) + 2$$

coprocessor-clock cycles to execute.

A WAIT_STATUS_ALE command takes:

$$\left((TM_PARclen1 + TM_PARclen2 + TM_PARclen3 + TM_PARalen1 + (TM_PARalen2 \times ADR_BYTE_NUM) + (TM_PARalen3 \times (ADR_BYTE_NUM - 1)) + TM_PARalen4 + (TM_PARrdn1 \times 2) + TM_PARrdn2 + TM_PARrdn3 + TM_PARrdn4) << TIM_MULT\right) + 2$$

coprocessor-clock cycles to execute.

Figure 30–12 shows the boot-bus-signal transitions during WAIT_STATUS execution and Figure 30–13 shows the boot-bus-signal transitions during WAIT_STATUS_ALE execution with ADR_BYTE_NUM = 2. In both cases, re-execution is not needed.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 30–12 WAIT_STATUS Signal Timing

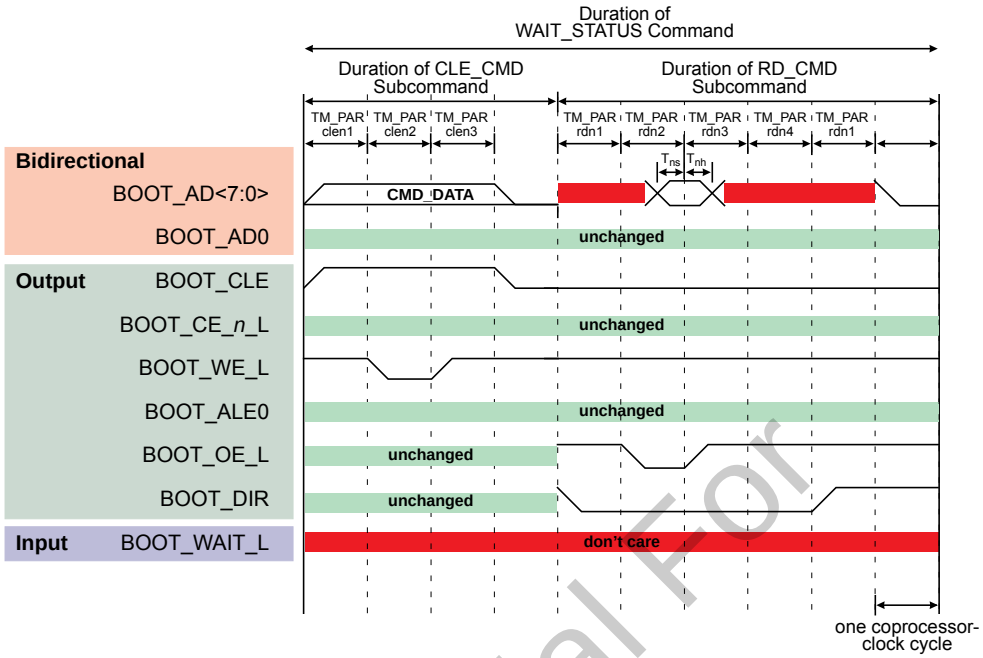
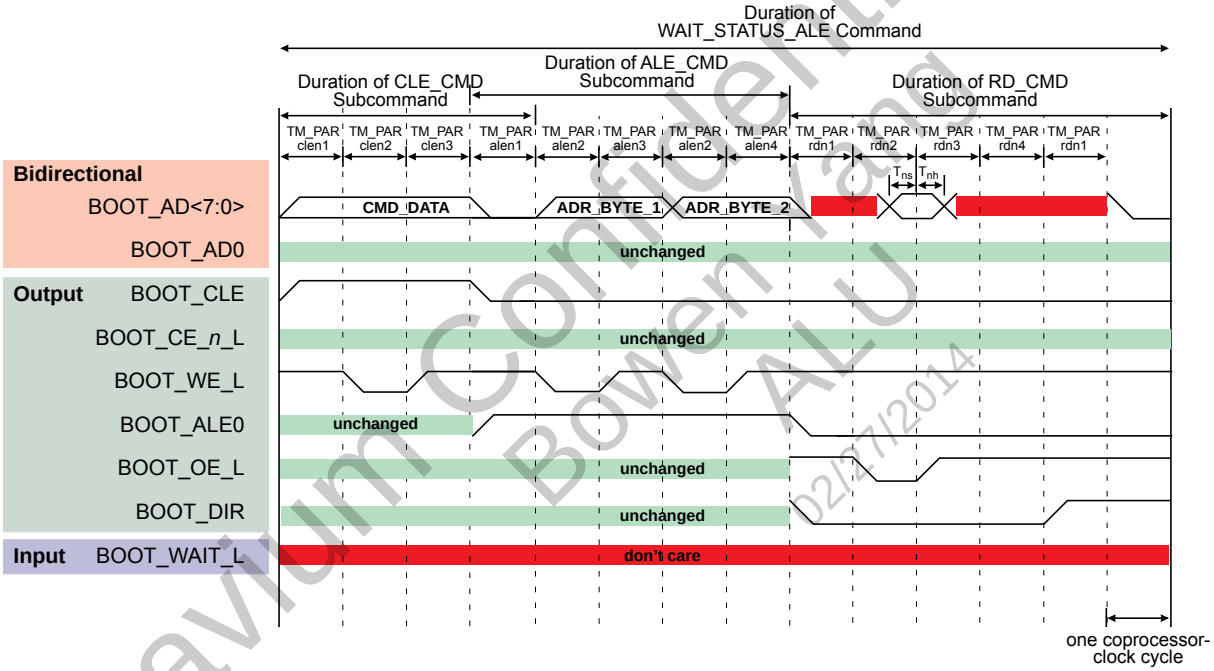


Figure 30–13 WAIT_STATUS_ALE Signal Timing



Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Restrictions:

- WAIT_STATUS* commands must be surrounded by a CHIP_EN/CHIP_DIS pair.
- WAIT_STATUS* must start on a naturally-aligned 64-bit boundary.
- ADR_BYTE_NUM (in the WAIT_STATUS_ALE command) must be 1, 2, 3 or 4.

30.3 NDF_CMD Command Examples

Software has direct control of the CN71XX NAND flash signaling via the NDF_CMD commands described in Section 30.2. This section provides some simple example sequences. Many possible sequences can be generated from the NDF_CMD commands; these examples are only a few of the possibilities.

Cavium Confidential For Bowen Yang ALU 02/27/2014

30.3.1 NAND Flash Cache-Block Read Sequence

The NAND flash device is typically read block-by-block with the accompanying ECC bytes. Here is an example of a 256-byte block with ECC.

The following definitions use the **rowadd** and **coladd** variables defined in Section 30.1. The description below does not correctly specify the eight-bit 64KB page configuration that has three column-address bytes.
(Set ADR_BYTE_3 = coladd<23:16> and ADR_BYTE_4 = rowadd<7:0> in this special case.)

The following sequence implements these read operations:

```
SET_TM_PAR(TIM_MULT = NDF_BT_PG_INFO[T_MULT], TM_PAR1 = 2,
           TM_PAR2 = 2, TM_PAR3 = 2, TM_PAR4 = 2, TM_PAR5 = 2,
           TM_PAR6 = 2, TM_PAR7 = 0xFF)

BUS_ACQ

CHIP_EN(CE = 0, DATA_BUS_WIDTH = 1)

CLE_CMD(CMD_DATA = 0x0, clen1 = TM_PAR4, clen2 = TM_PAR4,
        clen3 = TM_PAR4)

ALE_CMD(ADR_BYTE_NUM = NDF_BT_PG_INFO[ADR_CYC], alen1 =
        TM_PAR6, alen2 = TM_PAR6, alen3 = TM_PAR6, alen4 = TM_PAR5,
        {ADR_BYTE_2,ADR_BYTE_1} = coladd<15:0>,
        {ADR_BYTE_4,ADR_BYTE_3} = rowadd<15:0>,
        ADR_BYTE_[ADR_BYTE_NUM:5] = 0)

NOP      // present when NDF_BT_PG_INFO[ADR_CYC] ≤ 6

NOP      // additionally present when NDF_BT_PG_INFO[ADR_CYC] ≤ 4

CLE_CMD(CMD_DATA = 0x30, clen1 = TM_PAR4, clen2 = TM_PAR4,
        clen3 = TM_PAR4)

WAIT(R_B = 0, n = TM_PAR7)

RD_CMD(DATA_BYTES = 0x104, rdn1 = TM_PAR2, rdn2 = TM_PAR2,
       rdn3 = TM_PAR2, rdn4 = TM_PAR2)

CHIP_DIS

WAIT(R_B = 0, n = TM_PAR2)

BUS_REL

WAIT(R_B = 0, n = TM_PAR2)

NOP

...

NOP      //total of 9 NOPs in this sequence
```

The key parts of this sequence are:

- The two CLE_CMD commands transfer the 0x0 and 0x30 command bytes to the NAND device at the appropriate times.
- The ALE_CMD command transfers the row and column information to the selected NAND flash device.
- The first WAIT command should delay the subsequent RD_CMD command long enough to satisfy the page-read time constraint of the NAND flash device.

Cavium Confidential For Bowen Yang ALU 02/27/2014

- The RD_CMD command transfers 260 (0x104) bytes from the selected NAND flash device.

As with the early boot reset sequence (see TBD), this command sequence uses fixed delays, and does not rely on BOOT_WAIT_L (i.e. R/B#) nor WAIT_STATUS* commands.

With an eight-bit device, when NDF_BT_PG_INFO[T_MULT] is set to its reset value of 0x9, this sequence takes approximately 400,000 coprocessor-clock cycles to execute, with approximately one third of the time spent on the first WAIT command, and most of the remaining two thirds of the time spent in the RD_CMD command. As indicated in Section 30.4, software should write NDF_BT_PG_INFO before the second NAND flash cache-block read operation. This write operation should typically reduce NDF_BT_PG_INFO[T_MULT] below its reset value so that subsequent NAND flash cache-block read operations will complete more quickly. The appropriate NDF_BT_PG_INFO[T_MULT] value depends on both the CN71XX coprocessor-clock speed and the speed of the NAND flash device.

NOTE: DATA_BYTES = 260 (0x104) is not consistent with the restrictions that DMA must follow during early boot. (DATA_BYTES must be a multiple of 264 bytes in the DMA case. Section 28.1 and the RD_CMD* commands descriptions in Section 30.2 describe the DMA restrictions during early boot.)

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

30.3.2 Block-Erase Sequence

This sequence can implement a block erase on an eight-bit NAND flash device on chip-enable 0 with a 2-byte row address:

SET_TM_PAR(TIM_MULT = 3, TM_PAR1 = 3, TM_PAR2 = 6, TM_PAR3 = 0x18,
TM_PAR4 = 3, TM_PAR5 = 2, TM_PAR6 = 0x18, TM_PAR7 = 6)

BUS_ACQ

CHIP_EN(CE = 0, DATA_BUS_WIDTH = 1)

CLE_CMD(CMD_DATA = 0x60, clen1 = TM_PAR1, clen2 = TM_PAR1,
clen3 = TM_PAR1)

ALE_CMD(ADR_BYTE_NUM = 2, alen1 = TM_PAR2, alen2 = TM_PAR1,
alen3 = TM_PAR1, alen4 = TM_PAR1,
{ADR_BYTE_1,ADR_BYTE_0} = rowadd<15:0>,

CLE_CMD(CMD_DATA = 0xD0, clen1 = TM_PAR1, clen2 = TM_PAR1,
clen3 = TM_PAR1)

WAIT(R_B = 0, n = TM_PAR6) //for T_{WB}

WAIT_STATUS(DATA = 0x70, clen1 = TM_PAR1, clen2 = TM_PAR1,
clen3 = TM_PAR1, AND_MASK = 0x40, COMP_BYTE = 0x40,
rdn1 = TM_PAR3, rdn2 = TM_PAR4, rdn3 = TM_PAR5,
rdn4 = TM_PAR7)

CHIP_DIS

WAIT(R_B = 0, n = TM_PAR1)

BUS_REL

WAIT(R_B = 0, n = TM_PAR5)

Depending on the coprocessor-clock frequencies and the speed of the NAND flash device, different timing parameters may be required for the above sequence to function properly.

The key parts of this sequence are:

- The two CLE_CMD commands transfer the 0x60 and 0xD0 command bytes to the NAND device at appropriate times.
- The ALE_CMD command transfers the address of the selected row/block to the device.
- The WAIT_STATUS command waits until T_{BERS} is satisfied for the block erase.

Cavium Confidential For Bowen Yang ALU 02/27/2014

30.3.3 Page-Program Sequence

This sequence can implement a page program (i.e. write operation) on an eight-bit NAND flash device on chip-enable 0 with a 2-byte row address, a 2-byte column address, and a 2112-byte page:

SET_TM_PAR(TIM_MULT = 3, TM_PAR1 = 3, TM_PAR2 = 6, TM_PAR3 = 0x18,
TM_PAR4 = 3, TM_PAR5 = 2, TM_PAR6 = 0x18, TM_PAR7 = 6)

BUS_ACQ

CHIP_EN(CE = 0, DATA_BUS_WIDTH = 1)

CLE_CMD(CMD_DATA = 0x80, clen1 = TM_PAR1, clen2 = TM_PAR1,
clen3 = TM_PAR1)

ALE_CMD(ADR_BYTE_NUM = 4, alen1 = TM_PAR2, alen2 = TM_PAR1,
alen3 = TM_PAR1, alen4 = TM_PAR1,
{ADR_BYTE_2,ADR_BYTE_1} = coladd<15:0>,
{ADR_BYTE_4,ADR_BYTE_3} = rowadd<15:0>)

WAIT(R_B = 0, n = TM_PAR3) // for T_{ADL}

WR_CMD(DATA_BYTES = 0x840, wrn1 = TM_PAR4, wrn2 = TM_PAR5)

CLE_CMD(CMD_DATA = 0x10, clen1 = TM_PAR1, clen2 = TM_PAR1,
clen3 = TM_PAR1)

WAIT(R_B = 0, n = TM_PAR6) // for T_{WB}

WAIT_STATUS(DATA = 0x70, clen1 = TM_PAR1, clen2 = TM_PAR1,
clen3 = TM_PAR1, AND_MASK = 0x40, COMP_BYTE = 0x40,
rdn1 = TM_PAR3, rdn2 = TM_PAR4, rdn3 = TM_PAR5,
rdn4 = TM_PAR7)

CHIP_DIS

WAIT(R_B = 0, n = TM_PAR1)

BUS_REL

WAIT(R_B = 0, n = TM_PAR5)

Depending on the coprocessor-clock frequencies and the speed of the NAND flash device, different timing parameters may be required for the above sequence to function properly.

The key parts of this sequence are:

- The two CLE_CMD commands transfer the 0x80 and 0x10 commands to the NAND part at the appropriate time.
- The ALE_CMD command transfers the row and column information to the selected NAND flash device.
- The WR_CMD command transfers the page data from CN71XX to the NAND flash device.
- The WAIT_STATUS command waits until T_{PROG} is satisfied for the page program.

Cavium Confidential For Bowen Yang ALU 02/2/2014

30.4 NAND Flash Registers

The NAND flash registers are listed in Table 30–3.

Table 30–3 NAND Flash Registers

Register	Address	CSR Type[1]	Detailed Description
NDF_CMD	0x0001070001000000	NCB	See page 1351
NDF_MISC	0x0001070001000008	NCB	See page 1352
NDF_ECC_CNT	0x0001070001000010	NCB	See page 1353
NDF_BT_PG_INFO	0x0001070001000018	NCB	See page 1353
NDF_INT	0x0001070001000020	NCB	See page 1354
NDF_INT_EN	0x0001070001000028	NCB	See page 1354
NDF_DRBELL	0x0001070001000030	NCB	See page 1355
NDF_ST_REG	0x0001070001000038	NCB	See page 1355
MIO_NDF_DMA_CFG	0x0001180000000168	RSL	See page 1356
MIO_NDF_DMA_INT	0x0001180000000170	RSL	See page 1356
MIO_NDF_DMA_INT_EN	0x0001180000000178	RSL	See page 1356

1. NCB-type registers are accessed directly across the IOI; RSL-type registers are accessed indirectly across the IOI.

NAND Flash Memory Command Register NDF_CMD

When software reads this register, NDF_MISC[RD_VAL] is cleared to 0. Software must always write all eight bytes whenever it writes this register. If there are fewer than eight bytes left in the command sequence that software wants the NAND flash controller to execute, it must insert Idle (WAIT) commands to make up eight bytes. Software must also ensure that there is enough space in the NDF_CMD queue to accept these eight bytes by first reading NDF_MISC[FR_BYT].

Sixty-four-bit operations must be used to access this register. See Table 30–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	NF_CMD	R/W	0x0	—	Eight NAND flash memory command bytes.

Cavium Confidential For Bowen Yang ALU 02/27/2014

NAND Flash Memory Miscellaneous Register NDF_MISC

Sixty-four-bit operations must be used to access this register. See Table 30–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:28>	—	RAZ	—	—	Reserved.
<27>	MB_DIS	R/W	0	—	Set to disable multi-bit error hangs. Allows boot loads and boot DMAs to proceed as if no multi-bit errors occurred. Hardware fixes single bit errors as usual.
<26:24>	NBR_HWM	R/W	0x3	0x3	High watermark for NBR FIFO or load/store operations. This field specifies the high watermark for the IOI outbound load/store commands receive FIFO. The FIFO size is 16 entries.
<23:18>	WAIT_CNT	R/W	0x14	—	Wait input filter count. Represents the number of coprocessor-clock cycles for glitch filtering of BOOT_WAIT_L from the NAND flash device.
<17:7>	FR_BYT	RO/H	0x0	—	Unfilled NDF_CMD queue bytes. This field specifies the number of unfilled bytes in the NDF_CMD queue. Bytes become unfilled as commands complete execution and exit. (FIFO is 256 bytes when BT_DIS = 0 and 1536 bytes when BT_DIS = 1.)
<6>	RD_DONE	R/W1C/H	0	0	Read done. This bit is set to 1 by hardware when it reads the last eight bytes out of the NDF_CMD queue in response to RD_CMD being set to 1 by software.
<5>	RD_VAL	RO/H	0	—	This read-only bit is set to 1 by hardware when it reads the next eight bytes from NDF_CMD queue in response to RD_CMD being set to 1. A software read of NDF_CMD clears this bit to 0.
<4>	RD_CMD	R/W	0	0	Read command. When set to 1, the hardware reads the contents of the NDF_CMD queue eight bytes at a time and places the data into NDF_CMD. Software should first read RD_VAL to see if the next eight bytes from the NDF_CMD queue are available in NDF_CMD. All NDF_CMD queue read operations start and end on an eight-byte boundary. A RD_CMD command in the middle of command execution causes the execution to freeze until RD_DONE is set to 1. This bit is cleared on any NDF_CMD software write command.
<3>	BT_DMA	R/W	0	—	Boot-time DMA enable. When set to 1, boot-time DMA is enabled. This indicates to the NAND flash boot-control state machine that boot DMA read operations can begin. Software should set this bit to 1 after loading the NDF_CMD queue. Hardware sets the bit to 0 when boot DMA command execution is complete. If chip-enable 0 is not a NAND flash device, this bit is permanently 0 with software write operations ignored. When BT_DIS = 1, this bit is 0.
<2>	BT_DIS	R/W	0	1	Boot disable. Software must set this field to 1, which causes the boot-control state machines to sleep. This bit indicates to the NAND flash boot-control state machine that boot operation has ended. When this bit changes from 0 → 1, the NDF_CMD queue is emptied as a side effect. This bit must never be set when booting from NAND flash and region zero is enabled.
<1>	EX_DIS	R/W	0	0	Execution disable. When set to 1, this bit stops command execution after completing the execution of all commands currently in the NDF_CMD queue. Once command execution has stopped and then new commands are loaded into the NDF_CMD queue, command execution does not resume while this bit is 1. When this bit is set to 0, execution resumes if the NDF_CMD queue is not empty.

Cavium Confidential For Broadcom NDA 02/7/2011

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<0>	RST_FF	R/W	0	0	Reset FIFO. 1 = reset NDF_CMD queue to empty it; any command in flight is not aborted before resetting. The FIFO comes up empty at the end of power on reset. 0 = normal operation.

NAND Flash Memory ECC Count Register
NDF_ECC_CNT

Sixty-four-bit operations must be used to access this register. See Table 30–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved.
<31:8>	XOR_ECC	RO/H	0x0	—	Result of XOR operation of ECC read bytes and ECC generated bytes. The value pertains to the last single-bit ECC error.
<7:0>	ECC_ERR	RO/H	0x0	—	ECC error count. The number of single-bit errors fixed during boot. This count saturates instead of wrapping around.

NAND Flash Memory Page Information Register
NDF_BT_PG_INFO

This register provides the page size and the number of column-plus-row address-cycle information. Software writes to this register during a boot operation from a NAND flash device.

Additionally, software also writes the multiplier value for timing parameters that is used during the boot process, in the SET_TM_PARAM command. The multiplier value is used only by the boot-load state machine during boot operations. Boot DMA operations do not use this value.

Sixty-four-bit operations must be used to access this register. See Table 30–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:11>	—	RAZ	—	—	Reserved.
<10:7>	T_MULT	R/W	0x9	—	The boot-time TIM_MULT field of the SET_TM_PAR command.
<6:3>	ADR_CYC	R/W	0x8	—	Number of column-address cycles. Legal values are 0x3 – 0x8. Values written to this field smaller than 0x3 are converted to 0x3; values larger than 0x8 are converted to 0x8.
<2:0>	SIZE	R/W	0x0	—	Number of bytes per page in the NAND flash device = $2^{SIZE+1} \times 256$. SIZE Bytes/Page SIZE Bytes/Page 0x0 512 0x4 8K 0x1 1K 0x5 16K 0x2 2K 0x6 32K 0x3 4K 0x7 64K

Cavium Confidential For Bowen Yang ALU 04/2/2014

NAND Flash Memory Interrupt Register NDF_INT

This register contains the bits that can trigger an error interrupt. Sixty-four-bit operations must be used to access this register. See Table 30–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved.
<6>	OVRF	R/W1C/H	0	—	NDF_CMD write when FIFO is full. Generally a fatal error.
<5>	ECC_MULT	R/W1C/H	0	—	Multibit ECC error detected during boot.
<4>	ECC_1BIT	R/W1C/H	0	—	Single-bit ECC error detected and fixed during boot.
<3>	SM_BAD	R/W1C/H	0	—	One of the state machines is in a bad state,
<2>	WDOG	R/W1C/H	0	—	Watchdog timer expired during command execution.
<1>	FULL	R/W1C/H	0	—	NDF_CMD queue is full. FULL status is updated when the NDF_CMD queue becomes full as a result of software writing a new command to it.
<0>	EMPTY	R/W1C/H	0	—	NDF_CMD queue is empty. EMPTY status is updated when the NDF_CMD queue becomes empty as a result of command execution unit fetching the last instruction out of the NDF_CMD queue.

NAND Flash Memory Interrupt Enable Register NDF_INT_EN

This register contains the enable bits that allow an error interrupt. Sixty-four-bit operations must be used to access this register. See Table 30–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved.
<6>	OVRF	R/W/H	0	—	Wrote to a full command FIFO.
<5>	ECC_MULT	R/W/H	0	—	Multibit ECC error detected during boot.
<4>	ECC_1BIT	R/W/H	0	—	Single-bit ECC error detected and fixed during boot.
<3>	SM_BAD	R/W/H	0	—	One of the state machines is in a bad state,
<2>	WDOG	R/W/H	0	—	Watchdog timer expired during command execution.
<1>	FULL	R/W/H	0	—	NDF_CMD queue is full.
<0>	EMPTY	R/W/H	0	—	NDF_CMD queue is empty.

Cavium Confidential For Bowen Yang ALI 02/27/2014

NAND Flash Doorbell Register
NDF_DRBELL

This register is designed to control the execution of the NAND flash commands. The NDF command-execution unit must arbitrate for the flash bus before it can enable a NAND flash device connected to the CN71XX, which it then does by asserting the device's chip-enable signal. Therefore software must first load the NDF_CMD queue, with a full sequence of commands to perform a NAND flash device task.

This command sequence starts with a BUS_ACQ command, and the last command in the sequence must be a BUS_REL command. The execution unit starts execution of the sequence only if the NDF_DRBELL[CNT] is non-zero when it fetches the BUS_ACQ command.

Software can load multiple such sequences, each starting with a CHIP_EN command and ending with a CHIP_DIS command, and then write a data value to this register to increment the CNT field by the number of the command sequences loaded to the NDF_CMD queue.

Software register-write operations increment CNT by the signed eight-bit value being written. Software register-read operations return the current CNT value.

Hardware can also modify the value of CNT. Every time hardware executes a BUS_ACQ command to arbitrate and win the flash bus, it decrements CNT by 1. If CNT is already 0 or negative, the hardware command-execution unit stalls when it fetches the new BUS_ACQ command from the NDF_CMD queue. Only when the software writes to this register with a non-zero data value can the execution unit come out of the stalled condition, and resume execution.

Sixty-four-bit operations must be used to access this register. See Table 30–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved.
<7:0>	CNT	R/W/H	0x0	—	Doorbell count, in 2s-complement format.

NAND State Machine Register
NDF_ST_REG

This register aggregates all state machines used in NAND flash controller for debug purposes. See Table 30–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved.
<15>	EXE_IDLE	RO/H	1	—	Command execution status 1 = idle (execution of command sequence is complete and NDF_CMD queue is empty) 0 = busy
<14:11>	EXE_SM	RO/H	0x0	—	Command-execution state-machine states.
<10:7>	BT_SM	RO/H	0x0	—	Boot-load and boot-DMA state-machine states
<6>	RD_FF_BAD	RO/H	0	—	The NDF_CMD-queue read-back state machine is in a "bad" state
<5:4>	RD_FF	RO/H	0x0	—	NDF_CMD-queue read-back state machine states.
<3>	MAIN_BAD	RO/H	0	—	The main state machine is in a "bad" state.
<2:0>	MAIN_SM	RO/H	0x0	—	Main state-machine states.

Cavium Confidential For Bowen Yang ALU 04/27/2014

NAND Flash DMA Configuration Register
MIO_NDF_DMA_CFG

This register aggregates all state machines used in NAND flash controller for debug purposes.

Sixty-four-bit operations must be used to access this register. See Table 30–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	EN	R/W/H	0	—	DMA engine enable.
<62>	RW	R/W	0	—	DMA engine R/W bit: 0 = read, 1 = write.
<61>	CLR	R/W	0	—	DMA engine clear EN. When set to 1, EN is cleared on a device-terminated burst
<60>	—	RAZ	—	—	Reserved.
<59>	SWAP32	R/W	0	—	DMA engine 32-bit swap.
<58>	SWAP16	R/W	0	—	DMA engine enable 16-bit swap.
<57>	SWAP8	R/W	0	—	DMA engine enable eight-bit swap.
<56>	ENDIAN	R/W	0	—	DMA engine endian mode: 0 = big-endian, 1 = little-endian.
<55:36>	SIZE	R/W/H	0x0	—	DMA engine size. Specified in the number of 64-bit transfers (encoded in –1 notation).
<35:0>	ADR	R/W/H	0x0	—	DMA engine address. Must be 64-bit aligned.

NAND Flash DMA Interrupt Register
MIO_NDF_DMA_INT

Sixty-four-bit operations must be used to access this register. See Table 30–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	DONE	R/W1C/H	0	—	DMA engine request completion interrupt.

NAND Flash DMA Interrupt-Enable Register
MIO_NDF_DMA_INT_EN

Sixty-four-bit operations must be used to access this register. See Table 30–3 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	DONE	R/W	0	—	DMA engine request completion interrupt enable.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Chapter 31

Universal Serial Bus Unit (USB)

This chapter contains the following subjects:

- Overview
- Architecture
- Initialization
- Endian-Swapping
- Interrupt Handler
- Board Implementation
- Enumerations
- UPHY Registers
- UCTL Registers
- UAHC Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The CN71XX USB subsystem contains two instances of the USB subsystem. Each instance contains a dual-role device controller, and one physical USB port. Each instance has the following features:

- One physical USB root port, compliant with *Universal Serial Bus (USB) 3.0 Specification*, Revision 1.0, with errata to June 13, 2011, and compliant with *Universal Serial Bus Specification*, Revision 2.0 with errata to May 28, 2002. It supports the following serial data transmission rates:
  - 5.0-Gb/s SuperSpeed
  - 480-Mb/s high-speed
  - 12-Mb/s full-speed
  - 1.5-Mb/s low-speed
- One dual-role device (DRD) controller, which has two modes: host and device. DRD, when configured for host mode, is compliant with *eXtensible Host Controller Interface (xHCI) for USB*, Revision 1.0, with the following properties:
  - Standard microstate or open-source xHCI and class drivers
  - Concurrent IN and OUT transfers to get the full 8 Gb/s duplex throughput (interpacket delays and protocol overhead included)
  - Concurrent USB 3.0/2.0/1.1 traffic, designed so that USB 2.0 devices do not degrade the overall throughput
  - Dynamic power-savings and HW-LPM support.
- Supports big- and little-endian modes.

The CN71XX USB subsystem does not support the following features:

- On-the-Go (OTG) and Embedded Host (EH)
- Battery Charging Extension (BC)
- xHCI Extended Features
  - Legacy Support
  - Extended Power Management
  - I/O Virtualization
  - Message Interrupt
  - Local Memory
  - Debug
  - Extended Message Interrupt

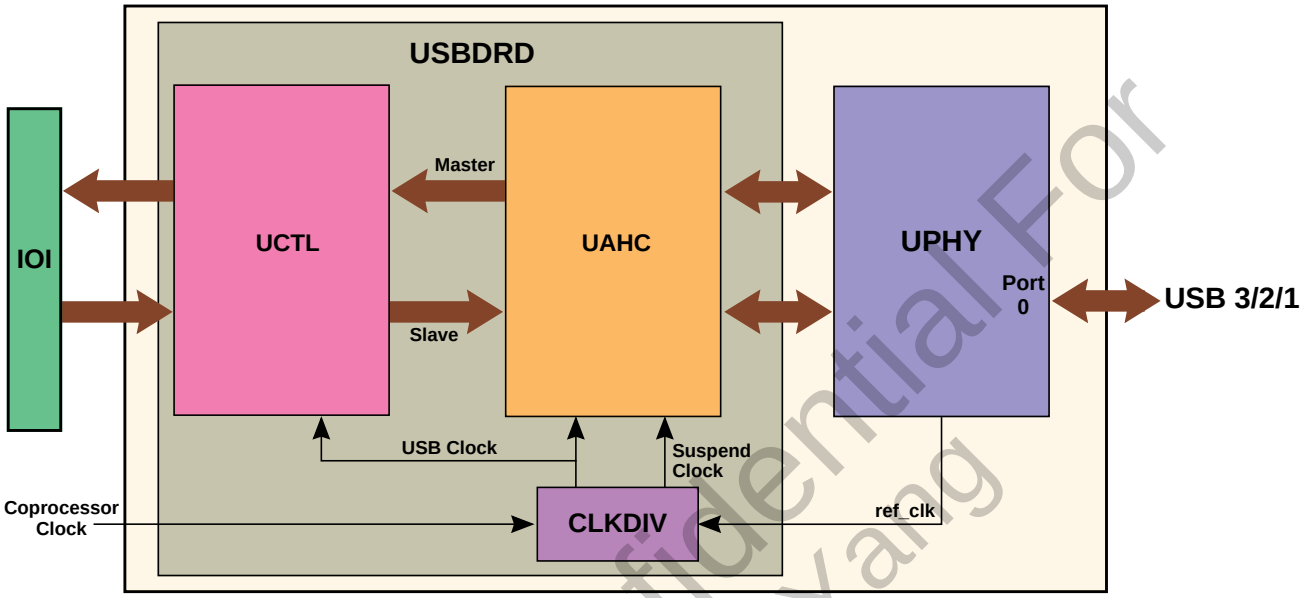
Cavium Confidential For Bowen Yang ALU 02/27/2014

31.1 Architecture

This section describes the general USB subsystem architecture and its major components, shown in Figure 31–1. The subsystem contains two major blocks:

- USBDRD
  - UCTL (shim to IOI)
  - UAHC (dual role device controller)
- UPHY (PCS + SuperSpeed PHY + high-speed PHY)

Figure 31–1 USB Subsystem Diagram



31.1.1 UCTL

UCTL is the bridge unit between the I/O interconnect (IOI) and the USB controller (UAHC). It performs the following functions:

- provides interfaces for the applications to access the UAHC xHCI registers on the CN71XX I/O space.
- provides a bridge for UAHC to fetch xHCI descriptors and data buffers from L2C through I/O inbound bus (IOBI) and I/O outbound bus (IOBO).
- posts interrupts to the CIU.
- contains registers that:
  - control the behavior of the UAHC and UPHY
  - control the clock/reset generation to UAHC, resets to UPHY, and power clamps for UPHY
  - control endian swapping for all UAHC registers and DMA accesses
  - access UPHY registers indirectly

Cavium Confidential For Bowen Yang ALU 02/27/2014

31.1.2 UAHC

The UAHC implements the link, protocol, and upper layers of the USB specification and is the interface to the application layer. It contains one controller configurable to be either an xHCI-compatible host controller or a USB device. It is configured to support the following functions:

- a single USB bus instance and a single USB 3.0 root port
- isochronous transfers
- 64-bit addressing
- clock-gating for dynamic-power-savings
- overcurrent indication
- port power control
- up to 64 slave devices

31.1.3 UPHY

The UPHY implements the physical-layer protocol of the USB specification. The UPHY consists of three major blocks:

- USB2.0 physical layer (PHY) for high-, full-, and low-speed operation
  - usb_dm/usb_dp lines: These lines are USB 2.0 and 1.1 specification-compliant and support high-speed (480 Mb/s) transfers as well as full-speed (12 Mb/s) and low-speed (1.5 Mb/s) transfers.
- USB3.0 physical layer (PHY) for SuperSpeed operation
  - SuperSpeed interface: transmit and receive differential pairs usb_sstx_m/p and usb_ssrx_m/p. These lines are USB 3.0 specification-compliant and support SuperSpeed (5 Gb/s) data transfers.

The UPHY also generates internal controller clock signals from a single input reference clock. For details on the reference clock requirements, see Section 31.5.2. The UPHY also requires a reference resistor for calibration purposes, see Section Section 31.5.2 for details.

The UPHY also provides a few more interfaces for testing purposes.

- CR interface: allows indirect access to the PHY internal registers for debug purposes (see Section 31.7.1).
- JTAG port: automated testing and diagnostic port for analog test and debug of the PHY.
- Boundary scan: implements AC JTAG 1149.6 on high-speed AC-coupled pins.

31.2 Initialization

For more details on the individual register values discussed here, see the individual register descriptions in Section 31.6.

No action is needed from power-on to be in the lowest power state. The hardware must be initialized before control is handed to the xHCI or device driver stack.

The USB subsystem has two input clocks:

- CN71XX coprocessor clock

Cavium Confidential For Bowen Yang ALU 02/27/2014

Most of the coprocessor-clock logic is disabled by USBDRD(0..1)_UCTL_CTL[CSCLK_EN] = 0 after reset. For applications that use USB interfaces, USBDRD(0..1)_UCTL_CTL[CSCLK_EN] should be set to 1 to enable the coprocessor clock in the USB subsystem (see step 9 of Section 31.2.1).

- USB reference clock (USBREF_CLK)

  For details on the reference-clock requirements, see Section 31.5.2. For details on reference-clock configuration, see step 5 of Section 31.2.1.

These two clocks generate four clocks that are only visible within the USB subsystem:

- USB controller clock (HCLK)

  Divided down from the coprocessor clock. The divider is controlled by USBDRD(0..1)_UCTL_CTL[H_CLK_EN, H_CLKDIV_RST, H_CLKDIV_SEL, H_CLK_BYP_SEL].

  The controller clock runs up to 300 MHz and must be at least 150 MHz for full-line-rate SuperSpeed functionality (or 125 MHz for any SuperSpeed functionality), or at least 90 MHz for high-speed/full-speed/low-speed-only full-line-rate functionality (62.5 MHz for any high-speed functionality).

- USB suspend-mode clock (SUSPEND_CLK)

  Used in low-power suspend state. It is a divide-by-16 version of the USB reference clock. It is automatically started when the USB reference clock is available.

- USB2.0 PHY clock (UTMI_CLK)

  60MHz, generated by UPHY from the USB reference clock.

- USB3.0 PHY clock (PIPE3_PCLK)

  125MHz with spread-spectrum modulation, generated by UPHY from the USB reference clock. For details on spread-spectrum configuration, see step 5 of Section 31.2.1.

The following are software-controlled resets in the USB subsystem:

- USBDRD(0..1)_UCTL_CTL[UCTL_RST] resets the UCTL except for the RSL interface and UCTL CSR offsets [0x0..0x2F].
- USBDRD(0..1)_UCTL_CTL[UAHC_RST] resets UAHC
- USBDRD(0..1)_UCTL_CTL[UPHY_RST] resets UPHY

If the USB subsystem is not being used, Cavium recommends that software remove power from the USB PHY and turn off the USB controller clock and CSCLK_EN to place the USB logic in the lowest power-state. See also Section 31.2.4.

Cavium Confidential For Bowen Yang ALU 2/21/2014

31.2.1 Cold Reset

Perform the following steps to initiate a cold reset.

1. Wait for all voltages to reach a stable state. Ensure the reference clock is up and stable.
   a. If 3.3V is up first, 0.85V must be soon after (within tens of ms).
2. If Over Current indication and/or Port Power Control features are desired, program the GPIO CSRs appropriately.
   a. For Over Current Indication, select a GPIO for the input and program GPIO_USBH_CTL[SEL].
   b. For Port Power Control, set one of GPIO_BIT_CFG(0..19)[OUTPUT_SEL] = USBH_VBUS_CTRL.
3. Assert all resets:
   a. UPHY reset: USBDRD(0..1)_UCTL_CTL[UPHY_RST] = 1
   b. UAHC reset: USBDRD(0..1)_UCTL_CTL[UAHC_RST] = 1
   c. UCTL reset: USBDRD(0..1)_UCTL_CTL[UCTL_RST] = 1
4. Configure the controller clock:
   a. Reset the clock dividers: USBDRD(0..1)_UCTL_CTL[H_CLKDIV_RST] = 1.
   b. Select the controller clock frequency
      – USBDRD(0..1)_UCTL_CTL[H_CLKDIV] = desired value.
      – USBDRD(0..1)_UCTL_CTL[H_CLKDIV_EN] = 1 to enable the controller clock.
      – Read USBDRD(0..1)_UCTL_CTL to ensure the values take effect.
   c. Deassert the controller clock divider reset: USBDRD(0..1)_UCTL_CTL[H_CLKDIV_RST] = 0.
5. Configure the strap signals in USBDRD(0..1)_UCTL_CTL.
   a. Reference clock configuration (see Table 31–2): USBDRD(0..1)_UCTL_CTL[REF_CLK_FSEL, MPLL_MULTIPLIER, REF_CLK_SEL, REF_CLK_DIV2].
   b. Configure and enable spread-spectrum for SuperSpeed: USBDRD(0..1)_UCTL_CTL[SSC_RANGE, SSC_EN, SSC_REF_CLK_SEL].
   c. Enable USBDRD(0..1)_UCTL_CTL[REF_SSP_EN].
   d. Configure PHY ports: USBDRD(0..1)_UCTL_CTL[USB*_PORT_PERM_ATTACH, USB*_PORT_DISABLE].
6. The PHY resets in lowest-power mode. Power up the per-port PHY logic by enabling the following:
   a. USBDRD(0..1)_UCTL_CTL [HS_POWER_EN] if high-speed/full-speed/low-speed functionality needed.
   b. USBDRD(0..1)_UCTL_CTL [SS_POWER_EN] if SuperSpeed functionality needed.
7. Wait 10 controller-clock cycles from step 4. for controller clock to start and async FIFO to properly reset.
8. Deassert UCTL and UAHC resets:
   a. USBDRD(0..1)_UCTL_CTL[UCTL_RST] = 0
   b. USBDRD(0..1)_UCTL_CTL[UAHC_RST] = 0

Cavium Confidential For Bowen Yang 02/11/2014

c. You will have to wait 10 controller-clock cycles before accessing any controller-clock-only registers.

9. Enable conditional coprocessor clock of UCTL by writing USBDRD(0..1)_UCTL_CTL[CSCLK_EN] = 1.

10. Set USBDRD(0..1)_UCTL_CTL[DRD_MODE] to 1 for device mode, 0 for host mode.

11. Soft reset the UPHY and UAHC logic via the UAHC controls:

a. USBDRD(0..1)_UAHC_GUSB2PHYCFG(0)[PHYSOFTRST] = 1

b. USBDRD(0..1)_UAHC_GUSB3PIPECTL(0)[PHYSOFTRST] = 1

c. USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1

12. Program USBDRD(0..1)_UAHC_GCTL[PRTCAPDIR] to 0x2 for device mode or 0x1 for host mode.

13. Wait 10 µs after step 12. for the PHY to complete its reset.

14. Deassert UPHY reset: USBDRD(0..1)_UCTL_CTL[UPHY_RST] = 0.

15. Wait for at least 45 µs after step 14. for UPHY to output stable PHYCLOCK.

16. Initialize any other strap signals necessary and make sure they propagate by reading back the last register written.

a. UCTL

- USBDRD(0..1)_UCTL_PORT0_CFG_*[*_TUNE]
- USBDRD(0..1)_UCTL_PORT0_CFG_*[PCS_*]
- USBDRD(0..1)_UCTL_PORT0_CFG_*[LANE0_TX_TERM_OFFSET]
- USBDRD(0..1)_UCTL_PORT0_CFG_*[TX_VBOOST_LVL]
- USBDRD(0..1)_UCTL__PORT0_CFG_*[LOS_BIAS]
- USBDRD(0..1)_UCTL_HOST_CFG
- USBDRD(0..1)_UCTL_SHIM_CFG

b. UAHC: only the following UAHC registers are accessible during CoreSoftReset.

- USBDRD(0..1)_UAHC_GCTL
- USBDRD(0..1)_UAHC_GUCTL
- USBDRD(0..1)_UAHC_GSTS
- USBDRD(0..1)_UAHC_GUID
- USBDRD(0..1)_UAHC_GUSB2PHYCFG(0)
- USBDRD(0..1)_UAHC_GUSB3PIPECTL(0)

17. Release soft reset the UPHY and UAHC logic via the UAHC controls:

a. USBDRD(0..1)_UAHC_GUSB2PHYCFG(0)[PHYSOFTRST] = 0

b. USBDRD(0..1)_UAHC_GUSB3PIPECTL(0)[PHYSOFTRST] = 0

c. USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 0

18. Configure the remaining UAHC_G* registers as needed, including any that were not configured in step 16.-b.

19. Initialize the USB controller:

a. To initialize the UAHC as a USB host controller, the application should perform the steps described in the xHCI specification (UAHC_X* registers). The

Cavium Confidential For Bowen Yang ALU 02/27/2014

xHCI sequence starts with poll for a 0 in USBDRD(0..1)_UAHC_USBSTS[CNR].

b. To initialize the UAHC as a device, the application should TBD. The device initiation sequence starts with a device soft reset by setting USBDRD(0..1)_UAHC_DCTL[CSFTRST] = 1 and wait for a read operation to return 0.

NOTE: If the UAHC registers are accessed over the IOI before step 11. above, the UAHC register interface might hang the IOI.

If the UCTL registers offsets 0x30 – 0xF8 are accessed over the RSL bus before step 9. above, the UCTL's RSL interface might hang.

31.2.2 Hot Reset

1. Quiesce all USB traffic.
2. Follow the initialization sequence from Section 31.2.1 starting with step 3.

31.2.3 Soft Reset

1. Quiesce all USB traffic.
2. Soft reset the PHY and UAHC logic: set USBDRD(0..1)_UAHC_GUSB2PHYCFG(0)[PHYSOFTRST], USBDRD(0..1)_UAHC_GUSB3PIPECTL(0)[PHYSOFTRST], and USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET].
   For device mode, USBDRD(0..1)_UAHC_DCTL[CSFTRST] should also be set.
3. Change any strap values as needed in UCTL registers.
4. Wait 10 µs from step 2. for the PHY to complete its reset.
5. Remove the reset to the PHYs: clear USBDRD(0..1)_UAHC_GUSB2PHYCFG(0)[PHYSOFTRST] and USBDRD(0..1)_UAHC_GUSB3PIPECTL(0)[PHYSOFTRST]
6. Wait at least 45 µs for PHY clocks to stabilize.
7. Remove the soft reset to the UAHC: clear USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET]
   For device mode, wait for a read from USBDRD(0..1)_UAHC_DCTL[CSFTRST] to return to 0.
8. Initialize the controller as described in step 19. of Section 31.2.1.

Note that for host mode there are also sequences defined in the xHCI spec that are even lighter soft resets. See the xHCI specification for details.

Cavium Confidential For Bowen Yang ALU 02/27/2014

31.2.4 Dynamically Powering Down PHYs

No action is needed from power-on to be in the lowest power state.

The UPHY has individual domains for USB3.0 and USB2.0 PHY portions. One domain can be powered off without affecting the functionality of the other domain.

To change the power state, the virtual machine must bring down the USB interface, then call the hypervisor.

1. Assert UPHY and UAHC software resets:
   a. UPHY reset: USBDRD(0..1)_UCTL_CTL[UPHY_RST] = 1
   b. UAHC reset: USBDRD(0..1)_UCTL_CTL[UAHC_RST] = 1
2. Wait one reference-clock cycle.
3. Configure, as needed:
   a. USBDRD(0..1)_UCTL_CTL[HS_POWER_EN] if high-speed/full-speed/low-speed functionality needed.
   b. USBDRD(0..1)_UCTL_CTL[SS_POWER_EN] if SuperSpeed functionality needed.
4. If completely powering down the USB subsystem, jump to step 5. Otherwise:
   a. Wait one reference-clock cycle.
   b. Follow the initialization sequence from Section 31.2.1 starting with step 8. Note that USBDRD(0..1)_UCTL_CTL[UPHY_RST] must be applied to the PHY for at least 10 µs after the PHY has been powered on.
5. Turn off the controller clock: USBDRD(0..1)_UCTL_CTL[H_CLK_EN] = 0.
6. Assert UCTL soft reset: USBDRD(0..1)_UCTL_CTL[UCTL_RST] = 1.
7. Disable conditional coprocessor clock: USBDRD(0..1)_UCTL_CTL[CSCLK_EN] = 0.

31.2.5 Statically Powering Down PHYs

If there is no intent to use the USB interface, tie all UPHY power supply pads to ground (refer to Table 31–1). UPHY signal pads can be left floating. In this configuration, the software should never enable the controller clock.

31.2.6 Suspend/Sleep/LPM

When the USB is in sleep or suspend mode (for USB2.0) or when the link is in the U1, U2, U3, SS.Inactive, SS.Rxdetect, or SS.Disabled state (for USB3.0), the core can be put into low-power mode.

The USBDRD(0..1)_UAHC_GUSB2PHYCFG(0)[ENBLSLPM, SUSPHY], and USBDRD(0..1)_UAHC_GUSB3PIPECTL(0)[SUSPEND_EN] configuration bits must be set to 1 to enable the UAHC to transition to lower-power states. For host mode, see the xHCI specification, sections 4.15 and 4.23 for more details.

For device mode, clock gating can be enabled if USBDRD(0..1)_UAHC_GCTL[DSBLCLKGTNG] is set to 0. When the USB is in a low-power state, the clocks are gated off automatically. When a wake condition is detected either in U1, U2 or changes in the USB/Link state, the core turns on the clocks.

Cavium Confidential For Bowen Yang ALU 02/27/2014

31.3 Endian-Swapping

The CN71XX architecture defaults to big-endian. The UAHC defaults to little-endian CSR accesses and DMA accesses.

The UCTL provides a configurable byte-lane swapper on the DMA datapath.

For CSRs, the open-source Linux software drivers default to viewing xHCI as a PCIe device. PCIe devices require registers to be programmed in little-endian mode. Likewise, CSR writes in device mode also need to be in little-endian. This means the software byte-swaps the register data before issuing the I/O write command to the USB coprocessor. The UCTL provides a configurable byte-lane swapper on the IOI datapath to the UAHC CSRs (note that the RSL path to the UAHC CSRs always presents the CSR data on the RSL bus as big-endian).

The two swapper configuration registers are both in UCTL_SHIM_CFG, fields CSR_ENDIAN_MODE and DMA_ENDIAN_MODE. Both fields are implemented to use the same enumeration mapping, and both default to big-endian mode.

31.4 Interrupt Handler

The central-interrupt unit (CIU) in CN71XX contains the interrupts that are defined by the xHCI specification, the interrupts in the UCTL shim, and device mode interrupts. For more information about interrupts, see Chapter 13 and CIU_CIB_USBDRD(0..1)_RAW(0).

31.4.1 UAHC Interrupts

For host mode, there are two UAHC interrupts: the main xHCI runtime interrupts (UAHC_IMANx_IP) and the host-system-error interrupt (UAHC_USBSTS_HSE). See the xHCI specification for more information.

For device mode, an interrupt is generated when there are more than 0 events in USBDRD(0..1)_UAHC_GEVNTCOUNT(0).

The USBDRD(0..1)_UCTL_INSTAT interrupts are mainly for fatal errors (ECC DBE, protocol violations, etc). For more information, see the interrupt bit descriptions in Section 31.7.

Cavium Confidential For Bowen Yang ALU 02/27/2014

31.5 Board Implementation

If the USB interface is not used, all data-input pads can be left floating; power-supply pads should be tied to ground. Refer to Tables 40–8 and 40–19.

31.5.1 Power Supplies and ESD Protection

The USB requires two supply voltages: a nominal 3.3V source, and a nominal 0.85V source. The connectivity is as shown in Table 31–1.

Table 31–1 Power Supply Requirements

Power Supply	Signal Names	Value (V)		
		Min	Typ	Max
USB high-voltage power supply	USB_VDD33	3.14	3.3	3.46
Low-voltage power supply	USB_DVDD, USB_VP, USB_VPTX	0.82	0.85	0.935

For robust ESD protection, connect the USB_VP, USB_VPTX, and USB_DVDD supplies together through a ferrite bead on-board. These core supplies can also be connected to the ASIC's core supply through a ferrite bead on-board, but they must be isolated on-die and in-package.

Since there are two instances of the USB subsystem, their core supplies USB_VP, USB_VPTX, USB_DVDD, and USB_VDD33 can be shorted together on-board.

31.5.2 Reference Clocks

There are two PLLs in the USB subsystem that need reference clocks: one drives the HighSpeed function, the other drives the SuperSpeed function. The HighSpeed PLL does not support spread-spectrum input. The SuperSpeed PLL must output spread-spectrum, which can be sourced either from the input reference clock or via the SuperSpeed PLL's internal spread-spectrum generation.

The USB's reference clocks can be sourced either from the on-chip 50MHz reference clock or from one of two on-chip differential reference-clock inputs from the GSER (DLMC_REF_CLK0/1). The on-chip 50MHz source is called PLL_REF_CLK. The available configurations for the reference clock are shown in Table 31–2, and the requirements for the off-chip reference clock are shown in Table 38–5. If there is no intent to use off-chip reference clock, tie these inputs to ground. For more information on how to configure the USB reference clocks, see the USBDRD(0..1)_UCTL_CTL register description in Section 31.7.2.

Table 31–2 Reference-Clock Configurations

SerDes Standard	Reference-Clock Frequency	Source	Spread Spectrum	Clock Selection
USB (2.0/ high-speed)	100 MHz	DLMC_REF_CLK*n* (100 MHz only)	Non-SSC	Software control
	50 MHz	PLL_REF_CLK (50 MHz only)		
USB (3.0/ SuperSpeed)	125 MHz 100 MHz 50 MHz	DLMC_REF_CLK*n*	SSC or non-SSC	Software control

Cavium Confidential For Beijen Yang 2022/04/14

31.5.3 Reference Resistor

Attach a 200Ω 1% 100-ppm/C precision resistor-to-ground on the board to the USB_RESREF pad. Each instance of the USB subsystem must have its own reference resistor.

The PHY uses an external resistor to calibrate the termination impedances of the PHY's high-speed inputs and outputs. The resistor is shared between the USB 2.0 high-speed outputs and the SuperSpeed I/O. Each time the PHY is taken out of a reset, a termination calibration is performed. The termination calibration is continuous for USB 2.0 operation except when the USB 2.0 port is suspended. The termination calibration is performed only when coming out of a reset in SuperSpeed operation.

Calibration of the SuperSpeed link can be requested externally by asserting UCTL_PORT*_CFG[RES_TUNE_REQ] = 1. When the calibration is complete, UCTL_PORT*_CFG[RES_TUNE_ACK] → 0. A resistor calibration on the SuperSpeed link cannot be performed while the link is operational.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

31.6 Enumerations

USBDRD UAHC Device Generic Command Enumeration USBDRD_UAHC_DGCMD_CMDTYPE_E

Enumerates commands for USBDRD(0..1)_UAHC_DGCMD[CMDTYPE]. Any command encodings that are not present are considered reserved.

Value	Mnemonic	Description
0x2	SET_PERIODIC_PARAMETERS	Allows the software to program the periodic parameters. USBDRD(0..1)_UAHC_DGCMDPAR[9:0]: Software should set this to the same value programmed by the host through the Set SEL device request, in microseconds. The Set SEL control transfer has six bytes of data and contains four values: **Offset / Name / Meaning** 0 / U1SEL / Time in s for U1 system-exit latency 1 / U1PEL / Time in s for U1 device-to-host exit latency 2 / U2SEL / Time in s for U2 system-exit latency 4 / U2PEL / Time in s for U2 device-to-host exit latency If the device is enabled for U1 and U2, then the U2PEL should be programmed. If the device is enabled only for U1, then U1PEL should be programmed into this parameter. If the value is greater than 125 µs, then the software must program a value of 0x0 into this register.
0x4	SET_SCRATCHPAD_BUFFER_ARRAY_ADDR_L	This command sets bits [31:0] of the external address of the scratchpad buffer array used for save/restore to the value in USBDRD(0..1)_UAHC_DGCMDPAR. If either this command or SET_SCRATCHPAD_BUFFER_ARRAY_ADDR_H is issued while the controller is stopped (USBDRD(0..1)_UAHC_DCTL[RS] = 0), USBDRD(0..1)_UAHC_DGCMD[CMDIOC] must be set to 0. The device scratchpad buffer array has the same format as the xHCI scratchpad buffer array; it contains an array of 64-bit pointers to data buffers that will be used to save the controller's state.
0x5	SET_SCRATCHPAD_BUFFER_ARRAY_ADDR_H	This command sets bits [63:32] of the external address of the scratchpad buffer array used for save/restore to the value in USBDRD(0..1)_UAHC_DGCMDPAR. If either this command or SET_SCRATCHPAD_BUFFER_ARRAY_ADDR_L is issued while the controller is stopped (USBDRD(0..1)_UAHC_DCTL[RS] = 0), USBDRD(0..1)_UAHC_DGCMD[CMDIOC] must be set to 0. The device scratchpad buffer array has the same format as the xHCI scratchpad buffer array; it contains an array of 64-bit pointers to data buffers that will be used to save the controller's state.

Cavium Confidential For Beta Partner Use Only

Value	Mnemonic	Description
0x7	TRANSMIT_DEVICE_NOTIFICATION	This command allows any device notification to be transmitted, using the notification type and notification parameters specified in USBDRD(0..1)_UAHC_DGCMDPAR. USBDRD(0..1)_UAHC_DGCMDPAR[3:0] = Notification Type USBDRD(0..1)_UAHC_DGCMDPAR[31:4] = Notification parameters, depends on the notification type. For example, to transmit a Function Wake, software sets USBDRD(0..1)_UAHC_DGCMDPAR[3:0] to 0x1, and USBDRD(0..1)_UAHC_DGCMDPAR[10:4] to the interface number. This field relates to the Notification Type Specific field in a Device Notification Transaction Packet as described in Section 8.5.6 of the USB3 Specification. The following bits of USBDRD(0..1)_UAHC_DGCMDPAR have been put into the corresponding DWORD described in Section 8.5.6 of the USB3 Specification: USBDRD(0..1)_UAHC_DGCMDPAR[3:0] into DWORD 1[7:4] (Notification Type) USBDRD(0..1)_UAHC_DGCMDPAR[27:4] into DWORD 1[31:8] (Notification Type Specific) USBDRD(0..1)_UAHC_DGCMDPAR[31:28] into DWORD 2[3:0] (Notification Type Specific) There is one exception for the Bus Interval Adjustment Device Notification: USBDRD(0..1)_UAHC_DGCMDPAR[19:4] represents the Bus Interval Adjustment field; however, in the USB3 specification, the Bus Interval Adjustment field is actually at 31:16 of DWORD 1. In the case of Host Role Request, USBDRD(0..1)_UAHC_DGCMDPAR[3:0] = 4, and USBDRD(0..1)_UAHC_DGCMDPAR[5:4] = RSP Phase.
0x9	SELECTED_FIFO_FLUSH	Flush the selected FIFO. USBDRD(0..1)_UAHC_DGCMDPAR[4:0] = FIFO Number. USBDRD(0..1)_UAHC_DGCMDPAR[5] = 1 for TX FIFO and 0 for RX FIFO.
0xA	ALL_FIFO_FLUSH	Flush all FIFOs. This command does not use the USBDRD(0..1)_UAHC_DGCMDPAR field.
0xC	SET_ENDPOINT_NRDY	Issuing this command makes the core think that the given endpoint is in an NRDY state. If there are buffers available in that endpoint, the core will immediately transmit an ERDY. USBDRD(0..1)_UAHC_DGCMDPAR[4:0] = physical endpoint number
0x10	RUN_SOC_BUS_LOOPBACK_TEST	When enabled, executes an SoC Bus loopback test, which allows the data flow from transmit to receive to be tested without any connection to a PHY. 1. Configure EP0 as a non-stream capable Bulk OUT endpoint with the desired MaxPacketSize. 2. Configure EP1 as a non-stream capable Bulk IN endpoint assigned to Tx-FIFO 0 with the desired MaxPacketSize. 3. Issue this command with Parameter[0] set to 1, enabling loopback mode. 4. Issue Start Transfer to EP0. 5. Issue Start Transfer to EP1. 6. The core reads data from the IN buffers and writes it back to the OUT buffers. The IN and OUT must have an equal amount of data buffer. The Tx-FIFO 0 default value of 66 should be changed to 130 for loop-back mode. USBDRD(0..1)_UAHC_DGCMDPAR[0]: When 1, enables loopback mode, when 0, disables loopback mode.

Cavium Confidential For Broden Wang 07/21/21

USBDRD UCTL DMA Read Command Enumeration
USBDRD_UCTL_DMA_READ_CMD_E

Enumerate IOI inbound command selections for DMA read operations.

Value	Mnemonic	Description
0x0	LDI	Use LDI (allocate local).
0x1	LDT	Use LDT (no allocate). Default.

USBDRD UCTL ECC Error Source Enumeration
USBDRD_UCTL_ECC_ERR_SOURCE_E

Enumerate sources of ECC error log information.

Value	Mnemonic	Description
0x0	NONE	No error logged.
0x5	RAM2_SBE	UAHC RAM2 single-bit error.
0x6	RAM1_SBE	UAHC RAM1 single-bit error.
0x7	RAM0_SBE	UAHC RAM0 single-bit error.
0xD	RAM2_DBE	UAHC RAM2 double-bit error.
0xE	RAM1_DBE	UAHC RAM1 double-bit error.
0xF	RAM0_DBE	UAHC RAM0 double-bit error.

USBDRD UCTL Endian-Mode Enumeration
USBDRD_UCTL_ENDIAN_MODE_E

Enumerate endian mode selections.

Value	Mnemonic	Description
0x0	LITTLE	Core is in little-endian mode. A-B-C-D-E-F-G-H becomes A-B-C-D-E-F-G-H. This is the mode to use with Open-Source xHCI drivers when the core is in little-endian mode.
0x1	BIG	Core is in big-endian mode. A-B-C-D-E-F-G-H becomes H-G-F-E-D-C-B-A (swap bytes within the 32-bit word, then swap 32-bit words within 64-bit doubleword). This is the mode to use with Open-Source xHCI drivers when the core is in big-endian mode.
0x2	RSVD2	Reserved. A-B-C-D-E-F-G-H becomes D-C-B-A-H-G-F-E.
0x3	RSVD3	Reserved. A-B-C-D-E-F-G-H becomes E-F-G-H-A-B-C-D.

USBDRD USBDRD_

Cavium Confidential For Boven Yang ALU 08/27/2014

USBDRD UCTL ECC Error Source Enumeration
USBDRD_UCTL_XM_BAD_DMA_TYPE_E

Enumerate type of DMA error seen.

Value	Mnemonic	Description
0x0	NONE	No error logged.
0x1	ADDR_OOB	AxAddr<64:42> ≠ 0x0.
0x2	LEN_GT_16	AxLen > 0xF.
0x3	MULTIBEAT_BYTE	AxSize = 0x0 and AxLen ≠ 0x0.
0x4	MULTIBEAT_HALFWORD	AxSize = 0x1 and AxLen ≠ 0x0.
0x5	MULTIBEAT_WORD	AxSize = 0x2 and AxLen ≠ 0x0.
(else)	—	Reserved.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

31.7 Registers

The CN71XX USB controller implements three kinds of registers: UPHY, UCTL, and UAHC.

- UPHY indirect registers: indirect access via the UCTL_PORT*_CR_DBG_* registers.
- UCTL RSL-type registers: .
  - Instance 0: 2KB starting at address 0x0001_1800_6800_0000
  - Instance 1: 2KB starting at address 0x0001_1800_6900_0000
- UAHC NCB-type registers:
  - Instance 0: 1MB starting at address 0x0001_6800_0000_0000
  - Instance 1: 1MB starting at address 0x0001_6900_0000_0000

RSL-type registers are accessed indirectly across the IOI. NCB-type registers are accessed directly across the IOI.

Section 31.7.1 shows the UPHY registers; Section 31.7.2 shows the UCTL registers; and Section 31.7.3 shows the UAHC registers.

31.7.1 UPHY Registers

The UPHY has control and status registers accessible via an indirect mechanism using the USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG/ USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS registers. The UPHY registers are also accessible via the PHY's JTAG interface.

The UPHY CR port is a simple 16-bit data/address parallel port that is provided for on-chip access to the control registers inside the UPHY. While access to these registers is not required for normal operation, this interface enables access some of the diagnostic features during normal operation or to override some basic control signals. This interface is completely asynchronous—using a handshake between the cr_cap_addr, cr_cap_data, cr_read, and cr_write input commands with cr_ack acknowledgements and cr_data_out outputs.

To access the UPHY registers indirectly through the CR interface, the controller clock must be running, UCTL_RST must be deasserted, and UPHY_RST must be deasserted. Software is responsible for ensuring that only one indirect access is ongoing at a time.

To read a UPHY register via indirect CR interface, perform the following steps:

1. Write USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG with:
   - the address of the register in [DATA_IN]
   - 0x0 in [CAP_ADDR], [CAP_DATA], [READ], and [WRITE]
2. Write USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG with:
   - the address of the register in [DATA_IN]
   - 0x1 in [CAP_ADDR]
   - 0x0 in [CAP_DATA], [READ], and [WRITE]
3. Poll for USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS[ACK] = 0x1.
4. Write all 0s to USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG.
5. Poll for USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS[ACK] = 0x0.

Cavium Confidential For Bowen Yang ALI 02/7/2014

6. Write USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG with:
   - 0x1 in [READ]
   - 0x0 in [DATA_IN], [CAP_ADDR], [CAP_DATA], and [WRITE]
7. Poll for USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS[ACK] = 0x1.
8. Read USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS[DATA_OUT]. This is the read data.
9. Write all 0s to USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG.
10. Poll for USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS[ACK] = 0x0.

To write a UPHY register via indirect CR interface:

1. Write USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG with:
   - the address of the register in [DATA_IN]
   - 0x0 in [CAP_ADDR], [CAP_DATA], [READ], and [WRITE]
2. Write USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG with:
   - the address of the register in [DATA_IN]
   - 0x1 in [CAP_ADDR]
   - 0x0 in [CAP_DATA], [READ], and [WRITE]
3. Poll for USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS[ACK] = 0x1.
4. Write all 0s to USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG.
5. Poll for USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS[ACK] = 0x0.
6. Write USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG with:
   - the write data in [DATA_IN]
   - 0x0 in [READ], [CAP_ADDR], [CAP_DATA], and [WRITE]
7. Write USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG with:
   - the write data in [DATA_IN]
   - 0x1 in [CAP_DATA]
   - 0x0 in [READ], [CAP_ADDR], and [WRITE]
8. Poll for USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS[ACK] 0x1.
9. Write USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG with all 0x0's.
10. Poll for USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS[ACK] 0x0.
11. Write USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG with:
    - 0x1 in [WRITE]
    - 0x0 in [DATA_IN], [READ], and [CAP_ADDR].
12. Poll for USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS[ACK] = 0x1.
13. Write all 0s to USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG.
14. Poll for USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS[ACK] = 0x0.

For partial write operations, a read-modify write operation is required. Note that the CAP_ADDR steps (1-5) do not have to be repeated until the address needs changed.

Cavium Confidential For Bowen Yang ALU 02/27/2014

31.7.2 UCTL Registers

UCTL registers are only accessible via RSL. They generally contain basic clock controls, resets, and static configuration bits for the entire USB subsystem. The UCTL registers should only be accessed using 64-bit requests. Note that UCTL registers starting from 0x30 can be accessed only after the controller clock is enabled and UCTL_RST is deasserted. Register addresses not listed in the table below will return 0xFFFFFFFF_FFFFFFFF on read operations, write operations will have no side-effects. No addresses should alias.

The UCTL registers are listed in Table 31–3.

Table 31–3 USBH-UCTL Registers

Register	Address[1]	CSR Type[2]	Detailed Description
USBDRD(0..1)_UCTL_CTL	0x0001180068000000 + α*0x1000000	RSL	See page 1376
USBDRD(0..1)_UCTL_BIST_STATUS	0x0001180068000008 + α*0x1000000	RSL	See page 1381
USBDRD(0..1)_UCTL_SPARE0	0x0001180068000010 + α*0x1000000	RSL	See page 1381
USBDRD(0..1)_UCTL_INTSTAT	0x0001180068000030 + α*0x1000000	RSL	See page 1382
USBDRD(0..1)_UCTL_PORT0_CFG_HS	0x0001180068000040 + α*0x1000000 + β*0x20	RSL	See page 1383
USBDRD(0..1)_UCTL_PORT0_CFG_SS	0x0001180068000048 + α*0x1000000 + β*0x20	RSL	See page 1385
USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG	0x0001180068000050 + α*0x1000000 + β*0x20	RSL	See page 1388
USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS	0x0001180068000058 + α*0x1000000 + β*0x20	RSL	See page 1388
USBDRD(0..1)_UCTL_HOST_CFG	0x00011800680000E0 + α*0x1000000	RSL	See page 1389
USBDRD(0..1)_UCTL_SHIM_CFG	0x00011800680000E8 + α*0x1000000	RSL	See page 1391
USBDRD(0..1)_UCTL_ECC	0x00011800680000F0 + α*0x1000000	RSL	See page 1392
USBDRD(0..1)_UCTL_SPARE1	0x00011800680000F8 + α*0x1000000	RSL	See page 1392

1. For registers with multiple versions, α represents the values within the specified range and β represents the second range..
2. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

UCTL Control Register
USBDRD(0..1)_UCTL_CTL

This register controls clocks, resets, power, and BIST. This register can be reset by IOI reset. See Table 31–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	CLEAR_BIST	R/W	0	0	BIST fast-clear mode select. A BIST run with this bit set clears all entries in USBDRD RAMs to 0x0. There are two major modes of BIST: full and clear. Full BIST is run by the BIST state machine when CLEAR_BIST is deasserted during BIST. Clear BIST is run if CLEAR_BIST is asserted during BIST. To avoid race conditions, software must first perform a CSR write operation that puts the CLEAR_BIST setting into the correct state and then perform another CSR write operation to set the BIST trigger (keeping the CLEAR_BIST state constant). CLEAR BIST completion is indicated by USBDRD(0..1)_UCTL_BIST_STATUS[NDONE]. A BIST clear operation takes almost 2,000 controller-clock cycles for the largest RAM.
<62>	START_BIST	R/W	0	0	Rising edge starts BIST on the memories in USBDRD. To run BIST, the controller clock must be both configured and enabled, and should be configured to the maximum available frequency given the available coprocessor clock and dividers. Also, the UCTL, UAHC, and UPHY should be held in software-initiated reset (using UPHY_RST, UAHC_RST, UCTL_RST) until BIST is complete. BIST defect status can be checked after FULL BIST completion, both of which are indicated in USBDRD(0..1)_UCTL_BIST_STATUS. The full BIST run takes almost 80,000 controller-clock cycles for the largest RAM.
<61:60>	REF_CLK_SEL	R/W	0x0	0x0	Reference clock select. Choose reference-clock source for the SuperSpeed and high-speed PLL blocks. 0x0 = Reference clock source for both PLLs is DLMC_REF_CLK0. 0x1 = Reference clock source for both PLLs is DLMC_REF_CLK1. 0x2 = Reference clock source for SuperSpeed PLL is DLMC_REF_CLK0, reference clock source for high-speed PLL is PLL_REF_CLK. 0x3 = Reference clock source for SuperSpeed PLL is DLMC_REF_CLK1, reference clock source for high-speed PLL is PLL_REF_CLK. The DLMC_REF_CLK* signals are the shared reference clocks from the SerDes blocks. This value can be changed only during UPHY_RST. **NOTE:** If REF_CLK_SEL = 0x0 or 0x1, then the DLMC_REF_CLK* input chosen cannot be spread-spectrum.

Cavium Confidential For Bowen Yang (ATU) 2021

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<59>	SSC_EN	R/W	1	1	Spread-spectrum clock enable. Enables spread-spectrum clock production in the SuperSpeed function. If the DLMC_REF_CLK* inputs are already spread-spectrum, then do not enable this feature. The clocks sourced to the SuperSpeed function must have spread-spectrum to be compliant with the USB specification. This value may only be changed during UPHY_RST.
<58:56>	SSC_RANGE	R/W	0x0	0x0	Spread-spectrum clock range. Selects the range of spread-spectrum modulation when SSC_EN is asserted and the PHY is spreading the SuperSpeed transmit clocks. Applies a fixed offset to the phase accumulator. 0x0 = –4980 ppm downspread of clock 0x1 = –4492 ppm 0x2 = –4003 ppm 0x3–0x7 = reserved All of these settings are within the USB 3.0 specification. The amount of EMI emission reduction might decrease as the SSC_RANGE increases; therefore, the SSC_RANGE settings can be registered to enable the amount of spreading to be adjusted on a per-application basis. This value can be changed only during UPHY_RST.
<55:47>	SSC_REF_ CLK_SEL	R/W	0x0	0x0	Enables non-standard oscillator frequencies to generate targeted MPLL output rates. Input corresponds to the frequency-synthesis coefficient. [55:53]: modulus – 1, [52:47]: 2's complement push amount A value of 0x0 means this feature is disabled. The legal values are as follows: If REF_CLK_SEL = 0x0 or 0x1, then 0x0 is the only legal value. If REF_CLK_SEL = 0x2 or 0x3: 0x0 = if DLMC_REF_CLK* is another supported frequency (see list in MPLL_MULTIPLIER description). All other values are reserved. This value may only be changed during UPHY_RST. **NOTE:** If REF_CLK_SEL = 0x2 or 0x3, then MPLL_MULTPLIER, REF_CLK_DIV2, and SSC_REF_CLK_SEL must all be programmed to the same frequency setting.

Cavium Confidential For Bowen Yang ALD 10/7/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<46:40>	MPLL_ MULTIPLIER	R/W	0x19	0x19	Multiplies the reference clock to a frequency suitable for intended operating speed. The legal values are: If REF_CLK_SEL = 0x0 or 0x1, then: 0x19 = 100 MHz on DQLMC_REF_CLK* If REF_CLK_SEL = 0x2 or 0x3, then: 0x32 = 50 MHz on DLMC_REF_CLK* 0x19 = 100 MHz on DLMC_REF_CLK* 0x28 = 125 MHz on DLMC_REF_CLK* INTERNAL: 0x19 = 200 MHz on DLMC_REF_CLK* All other values are reserved. This value may only be changed during UPHY_RST. **NOTE:** If REF_CLK_SEL = 0x2 or 0x3, then MPLL_MULTPLIER, REF_CLK_DIV2, and SSC_REF_CLK_SEL must all be programmed to the same frequency setting.
<39>	REF_SSP_EN	R/W	0	1	Enables reference clock to the prescaler for SuperSpeed function. This should always be enabled since this output clock is used to drive the UAHC suspend-mode clock during low-power states. This value can be changed only during UPHY_RST or during low-power states. The reference clock must be running and stable before UPHY_RST is deasserted and before REF_SSP_EN is asserted.
<38>	REF_CLK_DIV2	R/W	0	0	Divides the reference clock by 2 before feeding it into the REF_CLK_FSEL divider. The legal values are: If REF_CLK_SEL = 0x0 or 0x1, then for all DLMC_REF_CLK* frequencies, 0x0 is the only legal value. If REF_CLK_SEL = 0x2 or 0x3: 0x1 = legal if DLMC_REF_CLK* is 125MHz. 0x0 = legal if DLMC_REF_CLK* is another supported frequency (see list in MPLL_MULTIPLIER description). This value can be changed only during UPHY_RST. **NOTE:** If REF_CLK_SEL = 0x2 or 0x3, then MPLL_MULTPLIER, REF_CLK_DIV2, and SSC_REF_CLK_SEL must all be programmed to the same frequency setting.
<37:32>	REF_CLK_FSEL	R/W	0x27	0x27	Selects the reference clock frequency for the SuperSpeed and high-speed PLL blocks. The legal values are as follows: If REF_CLK_SEL = 0x0 or 0x1: 0x27 = 100 MHz on DLMC_REF_CLK* If REF_CLK_SEL = 0x2 or 0x3, then 0x7 is the only legal value. All other values are reserved. This value may only be changed during UPHY_RST. **NOTE:** When REF_CLK_SEL = 0x2 or 0x3, then MPLL_MULTPLIER, REF_CLK_DIV2, and SSC_REF_CLK_SEL must all be programmed to the same frequency setting.
<31>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang ALC 5/12/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<30>	H_CLK_EN	R/W	0	1	Controller-clock enable. When set to 1, the controller clock is generated. This also enables access to UCTL registers 0x30–0xF8.
<29>	H_CLK_BYP_SEL	R/W	0	0	Select the bypass input to the controller-clock divider. 0 = Use the divided coprocessor clock from the H_CLKDIV divider 1 = Use the bypass clock from the GPIO pins This signal is just a multiplexer-select signal; it does not enable the controller clock. You must still set H_CLKDIV_EN separately. H_CLK_BYP_SEL select should not be changed unless H_CLKDIV_EN is disabled. The bypass clock can be selected and running even if the controller-clock dividers are not running.
<28>	H_CLKDIV_RST	R/W	1	0	Controller-clock divider reset. Divided clocks are not generated while the divider is being reset. This also resets the suspend-clock divider.
<27>	—	RAZ	—	—	Reserved.
<26:24>	H_CLKDIV_SEL	R/W	0x3	0x3	Controller clock-frequency-divider select. The controller-clock frequency is the coprocessor-clock frequency divided by H_CLKDIV_SEL and must be at or below 300 MHz. The divider values are the following: 0x0 = divide by 1 0x1 = divide by 2 0x2 = divide by 4 0x3 = divide by 6 0x4 = divide by 8 0x5 = divide by 16 0x6 = divide by 24 0x7 = divide by 32 For USB3: • The controller-clock frequency must be at or above 125 MHz for any USB3 operation. • The controller-clock frequency must be at or above 150 MHz for full-rate USB3 operation. For USB2: • If DRD_MODE = DEVICE, the controller-clock frequency must be at or above 125 MHz for correct USB3 operation. • If DRD_MODE = HOST, the controller-clock frequency must be at or above 62.5 MHz for any USB2 operation. • If DRD_MODE = HOST, the controller-clock frequency must be at or above 90 MHz for full-rate USB2 operation. This field can be changed only when H_CLKDIV_RST = 1.
<23:22>	—	RAZ	—	—	Reserved.
<21>	USB3_PORT_PERM_ATTACH	R/W	0	0	Indicates this port is permanently attached. This is a strap signal; it should be modified only when UPHY_RST is asserted.
<20>	USB2_PORT_PERM_ATTACH	R/W	0	0	Indicates this port is permanently attached. This is a strap signal; it should be modified only when UPHY_RST is asserted.
<19>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang @2017

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<18>	USB3_PORT_DISABLE	R/W	0	0	Disables the USB3 (SuperSpeed) portion of this PHY. When set to 1, this signal stops reporting connect/disconnect events on the port and keeps the port in disabled state. This could be used for security reasons where hardware can disable a port regardless of whether xHCI driver enables a port or not. USBDRD(0..1)_UAHC_HCSPARAMS1[MAXPORTS] is not affected by this signal. This is a strap signal; it should be modified only when UPHY_RST is asserted.
<17>	—	RAZ	—	—	Reserved.
<16>	USB2_PORT_DISABLE	R/W	0	0	Disables USB2 (high-speed/full-speed/low-speed) portion of this PHY. When set to 1, this signal stops reporting connect/disconnect events on the port and keeps the port in disabled state. This could be used for security reasons where hardware can disable a port regardless of whether xHCI driver enables a port or not. USBDRD(0..1)_UAHC_HCSPARAMS1[MAXPORTS] is not affected by this signal. This is a strap signal; it should only be modified when UPHY_RST is asserted. **NOTE:** If Port0 is required to be disabled, ensure that the utmi_clk[0] is running at the normal speed. Also, all the enabled USB2.0 ports should have the same clock frequency as Port0.
<15>	—	RAZ	—	—	Reserved.
<14>	SS_POWER_EN	R/W	0	1	PHY SuperSpeed block power enable. This is a strap signal; it should only be modified when UPHY_RST is asserted.
<13>	—	RAZ	—	—	Reserved.
<12>	HS_POWER_EN	R/W	0	1	PHY high-speed block power enable. This is a strap signal; it should only be modified when UPHY_RST is asserted.
<11:5>	—	RAZ	—	—	Reserved.
<4>	CSCLK_EN	R/W	0	1	Turns on the USB UCTL interface clock (coprocessor clock). This enables access to UAHC registers via the IOI, as well as UCTL registers starting from 0x30 via the RSL bus.
<3>	DRD_MODE	R/W	1	0	DRD mode select. Switches between host and device mode for USBDRD: 1 = device mode, 0 = host mode.
<2>	UPHY_RST	R/W	0	0	PHY reset; resets UPHY; active-high.
<1>	UAHC_RST	R/W	0	0	Software reset; resets UAHC; active-high.
<0>	UCTL_RST	R/W	0	0	Software reset; resets UCTL; active-high. Resets UAHC DMA and register shims. Resets UCTL RSL registers 0x30–0xF8. **NOTE:** Does not reset UCTL RSL registers 0x0–0x28. UCTL RSL registers starting from 0x30 can be accessed only after the controller clock is active and UCTL_RST is deasserted.

Cavium Confidential For Bowen Yang ADI 02/21/2014

UCTL BIST Status Register
USBDRD(0..1)_UCTL_BIST_STATUS

This register indicates the results from the built-in self-test (BIST) runs of USBDRD memories. Wait until [*_NDONE] = 0, then check the corresponding [*_STATUS] field: a 0 indicates pass, a 1 indicates fail. This register can be reset by IOI reset. See Table 31–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:42>	—	RAZ	—	—	Reserved.
<41>	UCTL_XM_R_BIST_NDONE	RO/H	1	0	BIST is not complete for the UCTL AxiMaster read-data FIFO.
<40>	UCTL_XM_W_BIST_NDONE	RO/H	1	0	BIST is not complete for the UCTL AxiMaster write-data FIFO.
<39:35>	—	RAZ	—	—	Reserved.
<34>	UAHC_RAM2_BIST_NDONE	RO/H	1	0	BIST is not complete for the UAHC RxFIFO RAM (RAM2).
<33>	UAHC_RAM1_BIST_NDONE	RO/H	1	0	BIST is not complete for the UAHC TxFIFO RAM (RAM1).
<32>	UAHC_RAM0_BIST_NDONE	RO/H	1	0	BIST is not complete for the UAHC descriptor/ register cache (RAM0).
<31:10>	—	RAZ	—	—	Reserved.
<9>	UCTL_XM_R_BIST_STATUS	RO/H	0	0	BIST status of the UCTL AxiMaster read-data FIFO.
<8>	UCTL_XM_W_BIST_STATUS	RO/H	0	0	BIST status of the UCTL AxiMaster write-data FIFO.
<7:3>	—	RAZ	—	—	Reserved.
<2>	UAHC_RAM2_BIST_STATUS	RO/H	0	0	BIST status of the UAHC RxFIFO RAM (RAM2).
<1>	UAHC_RAM1_BIST_STATUS	RO/H	0	0	BIST status of the UAHC TxFIFO RAM (RAM1).
<0>	UAHC_RAM0_BIST_STATUS	RO/H	0	0	BIST status of the UAHC descriptor/register cache (RAM0).

UCTL Spare Register 0
USBDRD(0..1)_UCTL_SPARE0

This register is a spare register. This register can be reset by IOI reset. See Table 31–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	SPARE	R/W	0x0	—	Spare.

Cavium Confidential For Bowen Yang ALU 02/27/2014

UCTL Interrupt Status Register
USBDRD(0..1)_UCTL_INTSTAT

This register provides a summary of different bits of RSL interrupts. DBEs are detected and SBE are corrected. For debugging output for ECC DBEs/SBEs, see USBDRD(0..1)_UCTL_ECC. This register can be reset by IOI reset. See Table 31–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:22>	—	RAZ	—	—	Reserved.
<21>	RAM2_DBE	R/W1C/H	0	0	Detected double-bit error on the UAHC RxFIFO RAMs (RAM2).
<20>	RAM2_SBE	R/W1C/H	0	0	Detected single-bit error on the UAHC RxFIFO RAMs (RAM2).
<19>	RAM1_DBE	R/W1C/H	0	0	Detected double-bit error on the UAHC TxFIFO RAMs (RAM1).
<18>	RAM1_SBE	R/W1C/H	0	0	Detected single-bit error on the UAHC TxFIFO RAMs (RAM1).
<17>	RAM0_DBE	R/W1C/H	0	0	Detected double-bit error on the UAHC Desc/Reg Cache (RAM0).
<16>	RAM0_SBE	R/W1C/H	0	0	Detected single-bit error on the UAHC Desc/Reg Cache (RAM0).
<15:3>	—	RAZ	—	—	Reserved.
<2>	XM_BAD_DMA	R/W1C/H	0	0	Detected bad DMA access from UAHC to IOI. Error information is logged in USBDRD(0..1)_UCTL_SHIM_CFG[XM_BAD_DMA_*]. Received a DMA request from UAHC that violates the assumptions made by the AXI-to-IOI shim. Such scenarios include: illegal length/size combinations and address out-of-bounds. For more information on exact failures, see the description in USBDRD(0..1)_UCTL_SHIM_CFG[XM_BAD_DMA_TYPE]. The hardware does not translate the request correctly and results may violate IOI protocols. The hardware does not translate the request correctly and results might violate IOI protocols.
<1>	XS_NCB_OOB	R/W1C/H	0	0	Detected out-of-bound register access to UAHC over IOI. The UAHC defines 1MB of register space, starting at offset 0x0. Any accesses outside of this register space cause this bit to be set to 1. Error information is logged in USBDRD(0..1)_UCTL_SHIM_CFG[XS_NCB_OOB_*].
<0>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang ALD 02/27/2014

UCTL Port Configuration High-Speed Register USBDRD(0..1)_UCTL_PORT0_CFG_HS

This register controls configuration and test controls for the high-speed port 0 PHY. This register can be reset by IOI reset or USBDRD(0..1)_UCTL_CTL[UCTL_RST]. See Table 31–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:58>	—	RAZ	—	—	Reserved.
<57:55>	COMP_DIS_ TUNE	R/W	0x4	0x4	Disconnect threshold voltage. Adjusts the voltage level for the threshold used to detect a disconnect event at the host. A positive binary bit setting change results in a +1.5% incremental change in the threshold voltage level, while a negative binary bit setting change results in a –1.5% incremental change in the threshold voltage level.
<54:52>	SQ_RX_TUNE	R/W	0x3	0x3	Squelch threshold adjustment. Adjusts the voltage level for the threshold used to detect valid high-speed data. A positive binary bit setting change results in a –5% incremental change in threshold voltage level, while a negative binary bit setting change results in a +5% incremental change in threshold voltage level.
<51:48>	TX_FSLS_ TUNE	R/W	0x3	0x3	Low-speed/full-speed source impedance adjustment. Adjusts the low- and full-speed single-ended source impedance while driving high. This parameter control is encoded in thermometer code. A positive thermometer code change results in a –2.5% incremental change in source impedance. A negative thermometer code change results in +2.5% incremental change in source impedance. Any non-thermometer code setting (that is, 0x9) is not supported and reserved.
<47:46>	—	RAZ	—	—	Reserved.
<45:44>	TX_HS_XV_ TUNE	R/W	0x3	0x3	Transmitter high-speed crossover adjustment. This bus adjusts the voltage at which the DP0 and DM0 signals cross while transmitting in high-speed mode. 0x3 = default setting 0x2 = +15 mV 0x1 = –15 mV 0x0 = reserved
<43:42>	TX_PREEMP_ AMP_TUNE	R/W	0x1	0x1	High-speed transmitter pre-emphasis current control. Controls the amount of current sourced to DP0 and DM0 after a J-to-K or K-to-J transition. The high-speed transmitter preemphasis current is defined in terms of unit amounts. One unit amount is approximately 600 A and is defined as 1× preemphasis current. 0x3 = High-speed TX preemphasis circuit sources 3× preemphasis current. 0x2 = High-speed TX preemphasis circuit sources 2× preemphasis current. 0x1 = High-speed TX preemphasis circuit sources 1× preemphasis current. 0x0 = High-speed TX preemphasis is disabled. If these signals are not used, set them to 0x0.
<41>	—	RAZ	—	—	Reserved.

Cavium Confidential
Bowen Yang
02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<40>	TX_PREEMP_ PULSE_TUNE	R/W	0	0	High-speed transmitter preemphasis duration control. Controls the duration for which the high-speed preemphasis current is sourced onto DP0 or DM0. The high-speed transmitter preemphasis duration is defined in terms of unit amounts. One unit of preemphasis duration is approximately 580 ps and is defined as 1× preemphasis duration. This signal is valid only if either TX_PREEMP_AMP_TUNE0[1] or TX_PREEMP_AMP_TUNE0[0] is set to 1. 1 = 1×, short preemphasis current duration 0 = 2×, long preemphasis current duration (design default) If this signal is not used, set it to 0.
<39:38>	TX_RES_TUNE	R/W	0x1	0x1	USB source-impedance adjustment. Some applications require additional devices to be added on the USB, such as a series switch, which can add significant series resistance. This bus adjusts the driver source impedance to compensate for added series resistance on the USB. 0x3 = source impedance is decreased by approximately 4 Ω 0x2 = source impedance is decreased by approximately 2 Ω 0x1 = design default. 0x0 = source impedance is decreased by approximately 1.5 Ω Any setting other than the default can result in source-impedance variation across process, voltage, and temperature conditions that does not meet USB 2.0 specification limits. If this bus is not used, leave it at the default setting.
<37:36>	TX_RISE_TUNE	R/W	0x1	0x1	High-speed transmitter rise-/fall-time adjustment. Adjusts the rise/fall times of the high-speed waveform. A positive binary bit setting change results in a –4% incremental change in the high-speed rise/fall time. A negative binary bit setting change results in a +4% incremental change in the high-speed rise/fall time.
<35:32>	TX_VREF_TUNE	R/W	0x3	0x3	High-speed DC voltage-level adjustment. Adjusts the high-speed DC level voltage. A positive binary-bit-setting change results in a +1.25% incremental change in high-speed DC voltage level, while a negative binary-bit-setting change results in a –1.25% incremental change in high-speed DC voltage level.
<31:7>	—	RAZ	—	—	Reserved.
<6:4>	OTGTUNE	R/W	0x4	0x4	VBUS valid threshold adjustment function: this bus adjusts the voltage level for the VBUS<#> valid threshold. To enable tuning at the board level, connect this bus to a register. **NOTE:** A positive binary-bit-setting change results in a +1.5% incremental change in threshold-voltage level, while a negative binary-bit-setting change results in a –1.5% incremental change in threshold-voltage level.
<3:2>	VATEST_ENABLE	R/W	0x0	0x0	Analog test-pin select. Enables analog test voltages to be placed on the ID0 pin. 0x0 = test functionality disabled 0x1 = test functionality enabled 0x2, 0x3 = reserved, invalid settings See also the PHY databook for details on how to select which analog test voltage.
<1>	LOOPBACK_ENA BLE	R/W	0	0	Places the high-speed PHY in loopback mode, which concurrently enables high-speed receive and transmit logic.

Cavium Confidential For Board Bring-Up 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<0>	ATERESET	R/W	0	0	Per-PHY ATE reset. When the USB core is powered up (not in suspend mode), an automatic tester can use this to disable PHYCLOCK and FREECLK, then re-enable them with an aligned phase. 1 = PHYCLOCK and FREECLK outputs are disabled. 0 = PHYCLOCK and FREECLK are available within a specific period after ATERESET is deasserted.

UCTL Port Configuration SuperSpeed Register USBDRD(0..1)_UCTL_PORT0_CFG_SS

This register controls configuration and test controls for the SuperSpeed port 0 PHY.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UCTL_RESET] = 1

See Table 31–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:61>	TX_VBOOST_LVL	R/W	0x4	0x4	TX voltage-boost level. Sets the boosted transmit launch amplitude (mVppd). The default bit setting is intended to set the launch amplitude to approximately 1,008 mVppd. A single, positive binary bit setting change results in a +156 mVppd change in the Tx launch amplitude. A single, negative binary bit setting change results in a –156 mVppd change in the Tx launch amplitude. All settings more than one binary bit change should not be used. 0x3 = 0.844 V launch amplitude 0x4 = 1.008 V launch amplitude 0x5 = 1.156 V launch amplitude All others values are invalid.
<60:58>	LOS_BIAS	R/W	0x5	0x5	Loss-of-signal detector threshold-level control. A positive, binary bit setting change results in a +15 mVp incremental change in the LOS threshold. A negative binary bit setting change results in a –15 mVp incremental change in the LOS threshold. The 0x0 setting is reserved and must not be used. The default 0x5 setting corresponds to approximately 105 mVp. 0x0 = invalid 0x4 = 90 mV 0x1 = 45 mV 0x5 = 105 mV (default) 0x2 = 60 mV 0x6 = 120 mV 0x3 = 75 mV 0x7 = 135 mV
<57>	LANE0_EXT_PCLK_REQ	R/W	0	0	When asserted, this signal enables the pipe0_pclk output regardless of power state (along with the associated increase in power consumption). You can use this input to enable pipe0_pclk in the P3 state without going through a complete boot sequence.
<56>	LANE0_TX2RX_LOOPBK	R/W	0	0	When asserted, data from TX predriver is looped back to RX slicers. LOS is bypassed and based on the tx0_en input so that rx0_los = !tx_data_en.
<55:42>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowei Yang 8/12/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<41:32>	PCS_RX_LOS_ MASK_VAL	R/W	0x3E8	0x3E8	Configurable loss-of-signal mask width. Sets the number of reference clock cycles to mask the incoming LFPS in U3 and U2 states. Masks the incoming LFPS for the number of reference clock cycles equal to the value of pcs_rx_los_mask_val<9:0>. This control filters out short, non-compliant LFPS glitches sent by a noncompliant host. For normal operation, set to a targeted mask interval of 10µs (value = 10µs / Tref_clk). If the USBDRD(0..1)_UCTL_CTL[REF_CLK_DIV2] is used, then (value = 10µs / (2 × Tref_clk)). These equations are based on the SuperSpeed reference clock frequency. The value of PCS_RX_LOS_MASK_VAL should be as follows: **Frequency** **DIV2** **LOS_MASK** 125 MHz 0 0x4E2 100 MHz 0 0x3E8 50 MHz 0 0x1F4 Setting this bus to 0x0 disables masking. The value should be defined when the PHY is in reset. Changing this value during operation might disrupt normal operation of the link.
<31:26>	PCS_TX_ DEEPMH_3P5DB	R/W	0x18	0x18	Fine-tune transmitter driver deemphasis when set to 3.5db. This static value sets the Tx driver deemphasis value when pipeP_tx_deemph[1:0] is set to 0x1 (according to the PIPE3 specification). The values for transmit deemphasis are derived from the following equation: TX de-emphasis (db) = 20 × log_base_10((128 - 2 × pcs_tx_deemph)/128) In general, the parameter controls are static signals to be set prior to taking the PHY out of reset. However, you can dynamically change these values on-the-fly for test purposes. In this case, changes to the transmitter to reflect the current value occur only after the pipeP_tx_deemph[1:0] input changes.
<25:20>	PCS_TX_ DEEMPH_6DB	R/W	0x23	0x23	Fine-tune transmitter driver deemphasis when set to 6db. This static value sets the Tx driver deemphasis value when pipeP_tx_deemph[1:0] is set to 0x2 (according to the PIPE3 specification). This bus is provided for completeness and as a second potential launch amplitude. The values for transmit deemphasis are derived from the following equation: TX de-emphasis (db) = 20 × log_base_10((128 - 2 × pcs_tx_deemph)/128) In general, the parameter controls are static signals to be set prior to taking the PHY out of reset. However, you can dynamically change these values on-the-fly for test purposes. In this case, changes to the transmitter to reflect the current value occur only after the pipeP_tx_deemph[1:0] input changes.

Cavium Confidential For Bowen Ling ALD 08/20/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<19:13>	PCS_TX_ SWING_FULL	R/W	0x7F	0x7F	Launch amplitude of the transmitter. Sets the launch amplitude of the transmitter. The values for transmit amplitude are derived from the following equation: TX amplitude (V) = vptx × ((pcs_tx_swing_full + 1)/128) In general, the parameter controls are static signals to be set prior to taking the PHY out of reset. However, you can dynamically change these values on-the-fly for test purposes. In this case, changes to the transmitter to reflect the current value occur only after the pipeP_tx_deemph[1:0] input changes.
<12:8>	LANE0_TX_ TERM_OFFSET	R/W	0x0	0x0	Transmitter termination offset. Reserved, set to 0x0.
<7:6>	—	RAZ	—	—	Reserved.
<5>	RES_TUNE_ACK	RO/H	—	0	Resistor tune acknowledge. While asserted, indicates a resistor tune is in progress.
<4>	RES_TUNE_REQ	R/W	0	0	Resistor tune request. The rising edge triggers a resistor tune request (if one is not already in progress). When asserted, RES_TUNE_ACK is asserted high until calibration of the termination impedance is complete. **NOTE:** Tuning disrupts the normal flow of data; therefore, assert RES_TUNE_REQ only when the PHY is inactive. The PHY automatically performs a tune when coming out of PRST.
<3:0>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang ALU 02/27/2014

UCTL Port Debug Configuration Register USBDRD(0..1)_UCTL_PORT0_CR_DBG_CFG

This register allows indirect access to the configuration and test controls for the port 0 PHY. For directions to access the PHY registers indirectly through the CR interface, refer to Section 31.7.1.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UCTL_RESET] = 1

See Table 31–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved.
<47:32>	DATA_IN	R/W	0x0	0x0	Address or data to be written to the CR interface.
<31:4>	—	RAZ	—	—	Reserved.
<3>	CAP_ADDR	R/W	0	0	Rising edge triggers the DATA_IN field to be captured as the address.
<2>	CAP_DATA	R/W	0	0	Rising edge triggers the DATA_IN field to be captured as the write data.
<1>	READ	R/W	0	0	Rising edge triggers a register read operation of the captured address.
<0>	WRITE	R/W	0	0	Rising edge triggers a register write operation of the captured address with the captured data.

UCTL Port Debug Status Register USBDRD(0..1)_UCTL_PORT0_CR_DBG_STATUS

This register allows indirect access to the configuration and test controls for the port 0 PHY. For directions for usage, refer to Section 31.7.1.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UCTL_RESET] = 1

See Table 31–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:48>	—	RAZ	—	—	Reserved.
<47:32>	DATA_OUT	RO/H	0x0	0x0	Last data read from the CR interface.
<31:1>	—	RAZ	—	—	Reserved.
<0>	ACK	RO/H	0	0	Acknowledge that the CAP_ADDR, CAP_DATA, READ, WRITE commands have completed.

Cavium Confidential For Bowen Yang ALU 02/27/2014

UCTL Host Controller Configuration Register USBDRD(0..1)_UCTL_HOST_CFG

This register allows configuration of various host controller (UAHC) features. Most of these are strap signals and should be modified only while the controller is not running.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UCTL_RESET] = 1

See Table 31–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:60>	—	RAZ	—	—	Reserved.
<59:48>	HOST_CURRENT_BELT	RO/H	—	0x0	This signal indicates the minimum value of all received BELT values and the BELT that is set by the Set LTV command.
<47:38>	—	RAZ	—	—	Reserved.
<37:32>	FLA	R/W	0x20	0x20	High-speed jitter adjustment. Indicates the correction required to accommodate mac3 clock and utmi clock jitter to measure 125µs duration. With FLA tied to 0x0, the high-speed 125µs micro-frame is counted for 123933ns. The value needs to be programmed in terms of high-speed bit times in a 30 MHz cycle. Default value that needs to be driven is 0x20 (assuming 30 MHz perfect clock). FLA connects to the FLADJ register defined in the xHCI spec in the PCI configuration space. Each count is equal to 16 high-speed bit times. By default when this register is set to 0x20, it gives 125µs interval. Now, based on the clock accuracy, you can decrement the count or increment the count to get the 125 µs uSOF window. This is a strap signal; it should only be modified when UAHC is in reset (soft-reset okay).
<31:29>	—	RAZ	—	—	Reserved.
<28>	BME	R/W	1	1	Bus-master enable. This signal is used to disable the bus-mastering capability of the host. Disabling this capability stalls DMA accesses.
<27:26>	—	RAZ	—	—	Reserved.
<25>	PPC_EN	R/W	0	0	Port-power-control enable. 0 = USBDRD(0..1)_UAHC_HCCPARAMS[PPC] report port-power-control feature is unavailable. 1 = USBDRD(0..1)_UAHC_HCCPARAMS[PPC] reports port-power-control feature is available. PPC output from UAHC is taken to the GPIO signals and sense-converted based on PPC_ACTIVE_HIGH_EN. The MIO GPIO multiplexer must be programmed accordingly. This is a strap signal; it should only be modified when either the UCTL_CTL[UAHC] or UAHC_GCTL[CoreSoftReset] is asserted.

Cavium Confidential For Bowen Yang AU 02/27/2015

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<24>	PPC_ACTIVE_HIGH_EN	R/W	0	0	Port power control sense selection. The active-high port-power-control output to off-chip source is converted to match the off-chip sense. 1 = Port-power control to off-chip source is active-high. 0 = Port-power control to off-chip source is active-low. This is a strap signal; it should only be modified when either the UCTL_CTL[UAHC] or UAHC_GCTL[CoreSoftReset] is asserted.
<23:0>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang ALU 02/27/2014

UCTL Shim Configuration Register USBDRD(0..1)_UCTL_SHIM_CFG

This register allows configuration of various shim (UCTL) features. The fields XS_NCB_OOB_* are captured when there are no outstanding OOB errors indicated in INTSTAT and a new OOB error arrives. The fields XS_BAD_DMA_* are captured when there are no outstanding DMA errors indicated in INTSTAT and a new DMA error arrives.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UCTL_RESET] = 1

See Table 31–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	XS_NCB_OOB_WRN	RO/H	—	—	Read/write error log for out-of-bound UAHC register access. 0 = read, 1 = write
<62:60>	—	RAZ	—	—	Reserved.
<59:48>	XS_NCB_OOB_OSRC	RO/H	—	—	SRCID error log for out-of-bound UAHC register access. The IOI outbound SRCID for the OOB error. **CSR bits / Field bits / Description** [59:58] / [11:10] / chipID [57] / [9] / request source: 0 = core, 1 = IOI-device [56:51] / [8:3] / core/IOI-device number. Note that for IOI devices, [56]/[8] is always 0. [50:48] / [2:0] / SubID
<47>	XM_BAD_DMA_WRN	RO/H	—	—	Read/write error log for bad DMA access from UAHC. 0 = read error log, 1 = write error log
<46:44>	—	RAZ	—	—	Reserved.
<43:40>	XM_BAD_DMA_TYPE	RO/H	—	—	ErrType error log for bad DMA access from UAHC. Encodes the type of error encountered (error largest encoded value has priority). See USBDRD_UCTL_XM_BAD_DMA_TYPE_E.
<39:13>	—	RAZ	—	—	Reserved.
<12>	DMA_READ_CMD	R/W	1	1	Selects the IOI read command used by DMA accesses. See USBDRD_UCTL_DMA_READ_CMD_E.
<11:10>	—	RAZ	—	—	Reserved.
<9:8>	DMA_ENDIAN_MODE	R/W	0x1	0x1	Selects the endian format for DMA accesses to the L2C. See USBDRD_UCTL_ENDIAN_MODE_E.
<7:2>	—	RAZ	—	—	Reserved.
<1:0>	CSR_ENDIAN_MODE	R/W	0x1	0x1	Selects the endian format for IOI CSR accesses to the UAHC. Note that when UAHC CSRs are accessed via RSL, they are returned as big-endian. See USBDRD_UCTL_ENDIAN_MODE_E.

Cavium Confidential For Bowen Yang 12/27/2014

UCTL ECC Control Register
USBDRD(0..1)_UCTL_ECC

This register can be used to disable ECC correction, insert ECC errors, and debug ECC failures.

- The ECC_ERR* fields are captured when there are no outstanding ECC errors indicated in INTSTAT and a new ECC error arrives. Prioritization for multiple events occurring on the same cycle is indicated by the ECC_ERR_SOURCE enumeration: highest encoded value has highest priority.
- The *ECC_*_DIS fields disable ECC correction; SBE and DBE errors are still reported. If *ECC_*_DIS = 0x1, then no data-correction occurs.
- The *ECC_FLIP_SYND fields flip the syndrome<1:0> bits to generate single-bit/ double-bit error for testing.
  - 0x0 = normal operation
  - 0x1 = SBE on bit[0]
  - 0x2 = SBE on bit[1]
  - 0x3 = DBE on bit[1:0]

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UCTL_RESET] = 1

See Table 31–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:60>	—	RAZ	—	—	Reserved.
<59:56>	ECC_ERR_SOURCE	RO/H	0x0	—	Source of ECC error, see USBDRD_UCTL_ECC_ERR_SOURCE_E.
<55:48>	ECC_ERR_SYNDROME	RO/H	0x0	—	Syndrome bits of the ECC error.
<47:32>	ECC_ERR_ADDRESS	RO/H	0x0	—	RAM address of the ECC error.
<31:9>	—	RAZ	—	—	Reserved.
<8:7>	UAHC_RAM2_ECC_FLIP_SYND	R/W	0x0	0x0	Insert ECC error for testing purposes.
<6>	UAHC_RAM2_ECC_COR_DIS	R/W	0	0	Enables ECC correction on UAHC RxFIFO RAMs (RAM2).
<5:4>	UAHC_RAM1_ECC_FLIP_SYND	R/W	0x0	0x0	Insert ECC error for testing purposes.
<3>	UAHC_RAM1_ECC_COR_DIS	R/W	0	0	Enables ECC correction on UAHC TxFIFO RAMs (RAM1).
<2:1>	UAHC_RAM0_ECC_FLIP_SYND	R/W	0x0	0x0	Insert ECC error for testing purposes.
<0>	UAHC_RAM0_ECC_COR_DIS	R/W	0	0	Enables ECC correction on UAHC Desc/Reg cache (RAM0).

UCTL Spare Register 1
USBDRD(0..1)_UCTL_SPARE1

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UCTL_RESET] = 1

See Table 31–3 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	SPARE	R/W	0x0	—	Spare.

Cavium Confidential For Bowen Yang ALU 02/17/2014

31.7.3 UAHC Registers

UAHC registers are implemented by the dual-role device controller. xHCI-related registers are only valid during host mode, and device-mode registers are only valid during device mode. P rimary access is provided to these registers over the IOI. Debug access is provided over RSL.

Access to the UAHC registers follows the same rules as registers 0x30 to 0xF8 of the UCTL (i.e., the controller clock must be enabled and USBDRD(0..1)_UCTL_CTL[UCTL_RST] must be deasserted). Additionally, USBDRD(0..1)_UCTL_CTL[UAHC_RST] must be deasserted.

The UAHC registers occupy 1MB of address space. The interior mapping is shown in Table 31–4.

Table 31–4 UAHC Register-Address Map

Offset	Register Type
0x0_0000 – 0x0_7FFF	xHCI registers (see xHCI spec for details)
0x0000 – 0x001F	xHCI capability registers
0x0020 – 0x005F	operational registers
0x0420 – 0x043F	port registers
0x0440 – 0x0443	runtime registers
0x0460 – 0x047F	interrupt registers
0x0480 – 0x0583	doorbell registers
0x0880 – 0x08AF	xHCI extended capabilities
0x0_C100 – 0x0_C6FF	global registers (G* prefix)
0x0_C700 – 0x0_CBFF	device registers (D* prefix)
0x4_0000 – 0x7_FFFF	internal RAM 0 debug access
0x8_0000 – 0xB_FFFF	internal RAM 1 debug access
0xC_0000 – 0xF_FFFF	internal RAM 2 debug access

UAHC registers support 32 and 64-bit accesses, but it is recommended software use only 64-bit accesses for registers defined as 64-bits wide and 32-bit accesses for registers defined as 32-bits wide. Registers accessed over NCB are endian-swapped according to the USBDRD(0..1)_UCTL_SHIM_CFG[CSR_ENDIAN_MODE] setting.

For debug purposes, UAHC registers can also be accessed by presenting an NCB-type address over RSL. Registers accessed this way are statically mapped from UAHC's little-endian mode to the RSL's big-endian mode.

The UAHC registers are listed in Table 31–5.

Table 31–5 USBH-UAHC Registers

Register	Address[1]	CSR Type[2]	Detailed Description
USBDRD(0..1)_UAHC_CAPLENGTH	0x0001680000000000 + α*0x10000000000	NCB	See page 1396
USBDRD(0..1)_UAHC_HCSPARAMS1	0x0001680000000004 + α*0x10000000000	NCB	See page 1397
USBDRD(0..1)_UAHC_HCSPARAMS2	0x0001680000000008 + α*0x10000000000	NCB	See page 1397
USBDRD(0..1)_UAHC_HCSPARAMS3	0x000168000000000C + α*0x10000000000	NCB	See page 1398
USBDRD(0..1)_UAHC_HCCPARAMS	0x0001680000000010 + α*0x10000000000	NCB	See page 1398
USBDRD(0..1)_UAHC_DBOFF	0x0001680000000014 + α*0x10000000000	NCB	See page 1398
USBDRD(0..1)_UAHC_RTSOFF	0x0001680000000018 + α*0x10000000000	NCB	See page 1399

Cavium Confidential For Power Yang ALL 02/7/2014

Table 31–5 USBH-UAHC Registers (Continued)

Register	Address[1]	CSR Type[2]	Detailed Description
USBDRD(0..1)_UAHC_USBCMD	0x0001680000000020 + α*0x10000000000	NCB	See page 1399
USBDRD(0..1)_UAHC_USBSTS	0x0001680000000024 + α*0x10000000000	NCB	See page 1400
USBDRD(0..1)_UAHC_PAGESIZE	0x0001680000000028 + α*0x10000000000	NCB	See page 1400
USBDRD(0..1)_UAHC_DNCTRL	0x0001680000000034 + α*0x10000000000	NCB	See page 1401
USBDRD(0..1)_UAHC_CRCR	0x0001680000000038 + α*0x10000000000	NCB	See page 1401
USBDRD(0..1)_UAHC_DCBAAP	0x0001680000000050 + α*0x10000000000	NCB	See page 1402
USBDRD(0..1)_UAHC_CONFIG	0x0001680000000058 + α*0x10000000000	NCB	See page 1402
USBDRD(0..1)_UAHC_PORTSC(0..1)	0x0001680000000420 + α*0x10000000000 + β*0x10	NCB	See page 1403
USBDRD(0..1)_UAHC_PORTPMSC_20(0)	0x0001680000000424 + α*0x10000000000 + β*0x10	NCB	See page 1404
USBDRD(0..1)_UAHC_PORTLI_20(0)	0x0001680000000428 + α*0x10000000000 + β*0x10	NCB	See page 1404
USBDRD(0..1)_UAHC_PORTHLPMC_20(0)	0x000168000000042C + α*0x1000000000 + β*0x10	NCB	See page 1404
USBDRD(0..1)_UAHC_PORTPMSC_SS(1)	0x0001680000000424 + α*0x10000000000 + β*0x10	NCB	See page 1405
USBDRD(0..1)_UAHC_PORTLI_SS(1)	0x0001680000000428 + α*0x10000000000 + β*0x10	NCB	See page 1405
USBDRD(0..1)_UAHC_PORTHLPMC_SS(1)	0x000168000000042C + α*0x10000000000 + β*0x10	NCB	See page 1405
USBDRD(0..1)_UAHC_MFINDEX	0x0001680000000440 + α*0x10000000000	NCB	See page 1406
USBDRD(0..1)_UAHC_IMAN(0)	0x0001680000000460 + α*0x10000000000 + β*0x20	NCB	See page 1406
USBDRD(0..1)_UAHC_IMOD(0)	0x0001680000000464 + α*0x10000000000 + β*0x20	NCB	See page 1406
USBDRD(0..1)_UAHC_ERSTSZ(0)	0x0001680000000468 + α*0x10000000000 + β*0x20	NCB	See page 1407
USBDRD(0..1)_UAHC_ERSTBA(0)	0x0001680000000470 + α*0x10000000000 + β*0x20	NCB	See page 1407
USBDRD(0..1)_UAHC_ERDP(0)	0x0001680000000478 + α*0x10000000000 + β*0x20	NCB	See page 1407
USBDRD(0..1)_UAHC_DB(0..64)	0x0001680000000480 + α*0x10000000000 + β*0x4	NCB	See page 1408
USBDRD(0..1)_UAHC_USBLEGSUP	0x0001680000000880 + α*0x10000000000	NCB	See page 1408
USBDRD(0..1)_UAHC_USBLEGCTLSTS	0x0001680000000884 + α*0x10000000000	NCB	See page 1409
USBDRD(0..1)_UAHC_SUPTPRT2_DW0	0x0001680000000890 + α*0x10000000000	NCB	See page 1410
USBDRD(0..1)_UAHC_SUPTPRT2_DW1	0x0001680000000894 + α*0x10000000000	NCB	See page 1410
USBDRD(0..1)_UAHC_SUPTPRT2_DW2	0x0001680000000898 + α*0x10000000000	NCB	See page 1410
USBDRD(0..1)_UAHC_SUPTPRT2_DW3	0x000168000000089C + α*0x10000000000	NCB	See page 1410
USBDRD(0..1)_UAHC_SUPTPRT3_DW0	0x00016800000008A0 + α*0x10000000000	NCB	See page 1411
USBDRD(0..1)_UAHC_SUPTPRT3_DW1	0x00016800000008A4 + α*0x10000000000	NCB	See page 1411
USBDRD(0..1)_UAHC_SUPTPRT3_DW2	0x00016800000008A8 + α*0x10000000000	NCB	See page 1411
USBDRD(0..1)_UAHC_SUPTPRT3_DW3	0x00016800000008AC + α*0x10000000000	NCB	See page 1411
USBDRD(0..1)_UAHC_GSBUSCFG0	0x000168000000C100 + α*0x10000000000	NCB	See page 1412

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 31–5 USBH-UAHC Registers (Continued)

Register	Address[1]	CSR Type[2]	Detailed Description
USBDRD(0..1)_UAHC_GSBUSCFG1	0x000168000000C104 + α*0x10000000000	NCB	See page 1413
USBDRD(0..1)_UAHC_GTXTHRCFG	0x000168000000C108 + α*0x10000000000	NCB	See page 1414
USBDRD(0..1)_UAHC_GRXTHRCFG	0x000168000000C10C + α*0x10000000000	NCB	See page 1416
USBDRD(0..1)_UAHC_GCTL	0x000168000000C110 + α*0x10000000000	NCB	See page 1418
USBDRD(0..1)_UAHC_GPMSTS	0x000168000000C114 + α*0x10000000000	NCB	See page 1421
USBDRD(0..1)_UAHC_GSTS	0x000168000000C118 + α*0x10000000000	NCB	See page 1423
USBDRD(0..1)_UAHC_GUCTL1	0x000168000000C11C + α*0x10000000000	NCB	See page 1423
USBDRD(0..1)_UAHC_GRLSID	0x000168000000C120 + α*0x10000000000	NCB	See page 1424
USBDRD(0..1)_UAHC_GGPIO	0x000168000000C124 + α*0x10000000000	NCB	See page 1424
USBDRD(0..1)_UAHC_GUID	0x000168000000C128 + α*0x10000000000	NCB	See page 1424
USBDRD(0..1)_UAHC_GUCTL	0x000168000000C12C + α*0x10000000000	NCB	See page 1425
USBDRD(0..1)_UAHC_GBUSERRADDR	0x000168000000C130 + α*0x10000000000	NCB	See page 1428
USBDRD(0..1)_UAHC_GPRTBIMAP	0x000168000000C138 + α*0x10000000000	NCB	See page 1428
USBDRD(0..1)_UAHC_GHWPARAMS0	0x000168000000C140 + α*0x10000000000	NCB	See page 1429
USBDRD(0..1)_UAHC_GHWPARAMS1	0x000168000000C144 + α*0x10000000000	NCB	See page 1429
USBDRD(0..1)_UAHC_GHWPARAMS2	0x000168000000C148 + α*0x10000000000	NCB	See page 1430
USBDRD(0..1)_UAHC_GHWPARAMS3	0x000168000000C14C + α*0x10000000000	NCB	See page 1430
USBDRD(0..1)_UAHC_GHWPARAMS4	0x000168000000C150 + α*0x10000000000	NCB	See page 1431
USBDRD(0..1)_UAHC_GHWPARAMS5	0x000168000000C154 + α*0x10000000000	NCB	See page 1431
USBDRD(0..1)_UAHC_GHWPARAMS6	0x000168000000C158 + α*0x10000000000	NCB	See page 1432
USBDRD(0..1)_UAHC_GHWPARAMS7	0x000168000000C15C + α*0x10000000000	NCB	See page 1432
USBDRD(0..1)_UAHC_GDBGFIFOSPACE	0x000168000000C160 + α*0x10000000000	NCB	See page 1433
USBDRD(0..1)_UAHC_GDBGLTSSM	0x000168000000C164 + α*0x10000000000	NCB	See page 1434
USBDRD(0..1)_UAHC_GDBGLNMCC	0x000168000000C168 + α*0x10000000000	NCB	See page 1435
USBDRD(0..1)_UAHC_GDBGBMU	0x000168000000C16C + α*0x10000000000	NCB	See page 1435
USBDRD(0..1)_UAHC_GDBGLSPMUX	0x000168000000C170 + α*0x10000000000	NCB	See page 1436
USBDRD(0..1)_UAHC_GDBGLSP	0x000168000000C174 + α*0x10000000000	NCB	See page 1436
USBDRD(0..1)_UAHC_GDBGEPINFO	0x000168000000C178 + α*0x10000000000	NCB	See page 1437
USBDRD(0..1)_UAHC_GPRTBIMAP_HS	0x000168000000C180 + α*0x10000000000	NCB	See page 1437
USBDRD(0..1)_UAHC_GPRTBIMAP_FS	0x000168000000C188 + α*0x10000000000	NCB	See page 1437
USBDRD(0..1)_UAHC_GUSB2PHYCFG(0)	0x000168000000C200 + α*0x10000000000 + β*0x4	NCB	See page 1438
USBDRD(0..1)_UAHC_GUSB2I2CCTL	0x000168000000C240 + α*0x10000000000 + β*0x4	NCB	See page 1440
USBDRD(0..1)_UAHC_GUSB3PIPECTL(0)	0x000168000000C2C0 + α*0x10000000000 + β*0x4	NCB	See page 1441
USBDRD(0..1)_UAHC_GTXFIFOSIZ(0..3)	0x000168000000C300 + α*0x10000000000 + β*0x4	NCB	See page 1444
USBDRD(0..1)_UAHC_GRXFIFOSIZ(0..2)	0x000168000000C380 + α*0x10000000000 + β*0x4	NCB	See page 1445
USBDRD(0..1)_UAHC_GEVNTADR(0)	0x000168000000C400 + α*0x10000000000 + β*0x10	NCB	See page 1446
USBDRD(0..1)_UAHC_GEVNTSIZ(0)	0x000168000000C408 + α*0x10000000000 + β*0x10	NCB	See page 1446

Cavium Confidential For Bowen Yang 2019/12/11

Table 31–5 USBH-UAHC Registers (Continued)

Register	Address[1]	CSR Type[2]	Detailed Description
USBDRD(0..1)_UAHC_GEVNTCOUNT(0)	0x000168000000C40C + α*0x10000000000 + β*0x10	NCB	See page 1447
USBDRD(0..1)_UAHC_GHWPARAMS8	0x000168000000C600 + α*0x10000000000	NCB	See page 1447
USBDRD(0..1)_UAHC_GTXFIFOPRIDEV	0x000168000000C610 + α*0x10000000000	NCB	See page 1448
USBDRD(0..1)_UAHC_GTXFIFOPRIHST	0x000168000000C618 + α*0x10000000000	NCB	See page 1449
USBDRD(0..1)_UAHC_GRXFIFOPRIHST	0x000168000000C61C + α*0x10000000000	NCB	See page 1450
USBDRD(0..1)_UAHC_GDMAHLRATIO	0x000168000000C624 + α*0x10000000000	NCB	See page 1451
USBDRD(0..1)_UAHC_GFLADJ	0x000168000000C630 + α*0x10000000000	NCB	See page 1452
USBDRD(0..1)_UAHC_DCFG	0x000168000000C700 + α*0x10000000000	NCB	See page 1455
USBDRD(0..1)_UAHC_DCTL	0x000168000000C704 + α*0x10000000000	NCB	See page 1457
USBDRD(0..1)_UAHC_DEVTEN	0x000168000000C708 + α*0x10000000000	NCB	See page 1461
USBDRD(0..1)_UAHC_DSTS	0x000168000000C70C + α*0x10000000000	NCB	See page 1462
USBDRD(0..1)_UAHC_DGCMDPAR	0x000168000000C710 + α*0x10000000000	NCB	See page 1464
USBDRD(0..1)_UAHC_DGCMD	0x000168000000C714 + α*0x10000000000	NCB	See page 1465
USBDRD(0..1)_UAHC_DALEPENA	0x000168000000C720 + α*0x10000000000	NCB	See page 1466
USBDRD(0..1)_UAHC_DEPCMDPAR2_(0..15)	0x000168000000C800 + α*0x10000000000 + β*0x10	NCB	See page 1467
USBDRD(0..1)_UAHC_DEPCMDPAR1_(0..15)	0x000168000000C804 + α*0x10000000000 + β*0x10	NCB	See page 1467
USBDRD(0..1)_UAHC_DEPCMDPAR0_(0..15)	0x000168000000C808 + α*0x10000000000 + β*0x10	NCB	See page 1468
USBDRD(0..1)_UAHC_DEPCMD(0..15)	0x000168000000C80C + α*0x10000000000 + β*0x10	NCB	See page 1468

1. For registers with multiple versions, α represents the values within the specified range and β represents the second range.
2. NCB-type registers are accessed directly across the IOI.

XHCI Capability Length Register USBDRD(0..1)_UAHC_CAPLENGTH

For information on this register, refer to the xHCI Specification, v1.0, section 5.3.1. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	HCIVERSION	RO	0x100	0x100	Host controller interface version number.
<15:8>	—	RAZ	—	—	Reserved.
<7:0>	CAPLENGTH	RO	0x20	0x20	Capability registers length.

Cavium Confidential For Bowen Yang ALU 02/27/2014

XHCI Controller Structural Parameters Register 1 USBDRD(0..1)_UAHC_HCSPARAMS1

For information on this register, refer to the xHCI Specification, v1.0, section 5.3.3. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	MAXPORTS	RO	0x2	0x2	Maximum number of ports.
<23:19>	—	RAZ	—	—	Reserved.
<18:8>	MAXINTRS	RO	0x1	0x1	Maximum number of interrupters.
<7:0>	MAXSLOTS	RO	0x40	0x40	Maximum number of device slots.

XHCI Controller Structural Parameters Register 2 USBDRD(0..1)_UAHC_HCSPARAMS2

For information on this register, refer to the xHCI Specification, v1.0, section 5.3.4. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:27>	MAXSCRATCHPADBUFS_L	RO	0x1	0x1	Maximum number of scratchpad buffers[4:0].
<26>	SPR	RO	1	1	Scratchpad restore.
<25:21>	MAXSCRATCHPADBUFS_H	RO	0x0	0x0	Maximum number of scratchpad buffers[9:5].
<20:8>	—	RAZ	—	—	Reserved.
<7:4>	ERST_MAX	RO	0xF	0xF	Event ring segment table maximum.
<3:0>	IST	RO	0x1	0x1	Isochronous scheduling threshold.

Cavium Confidential For Bowen Yang ALU 02/27/2014

XHCI Controller Structural Parameters Register 3 USBDRD(0..1)_UAHC_HCSPARAMS3

For information on this register, refer to the xHCI Specification, v1.0, section 5.3.5. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	U2_DEVICE_EXIT_LATENCY	RO	0x7FF	0x7FF	U2 device exit latency.
<15:8>	—	RAZ	—	—	Reserved.
<7:0>	U1_DEVICE_EXIT_LATENCY	RO	0xA	0xA	U1 device exit latency.

XHCI Controller Capability Parameters Register USBDRD(0..1)_UAHC_HCCPARAMS

For information on this register, refer to the xHCI Specification, v1.0, section 5.3.6. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	XECP	RO	0x220	0x220	xHCI extended capabilities pointer.
<15:12>	MAXPSASIZE	RO	0xF	0xF	Maximum primary-stream-array size.
<11:9>	—	RAZ	—	—	Reserved.
<8>	PAE	RO	0	0	Parse all event data.
<7>	NSS	RO	0	0	No secondary SID support.
<6>	LTC	RO	1	1	Latency tolerance messaging capability.
<5>	LHRC	RO	1	1	Light HC reset capability.
<4>	PIND	RO	0	0	Port indicators.
<3>	PPC	RO	0	0	Port power control. Value is based on USBDRD(0..1)_UCTL_HOST_CFG[PPC_EN].
<2>	CSZ	RO	1	1	Context size.
<1>	BNC	RO	0	0	BW negotiation capability.
<0>	AC64	RO	1	1	64-bit addressing capability.

XHCI Doorbell Array Offset Register USBDRD(0..1)_UAHC_DBOFF

For information on this register, refer to the xHCI Specification, v1.0, section 5.3.7. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:2>	DBOFF	RO	0x120	0x120	Doorbell array offset.
<1:0>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang ALU 02/27/2014

XHCI Runtime Register-Space Offset Register USBDRD(0..1)_UAHC_RTSOFF

For information on this register, refer to the xHCI Specification, v1.0, section 5.3.8. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:5>	RTSOFF	RO	0x22	0x22	Runtime register-space offset.
<4:0>	—	RAZ	—	—	Reserved.

XHCI Command Register USBDRD(0..1)_UAHC_USBCMD

For information on this register, refer to the xHCI Specification, v1.0, section 5.4.1. This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:12>	—	RAZ	—	—	Reserved.
<11>	EU3S	R/W	0	—	Enable U3 MFINDEX stop.
<10>	EWE	R/W	0	—	Enable wrap event.
<9>	CRS	WO	0	—	Controller restore state.
<8>	CSS	WO	0	—	Controller save state.
<7>	LHCRST	R/W1S/H	0	—	Light-host-controller reset.
<6:4>	—	RAZ	—	—	Reserved.
<3>	HSEE	R/W	0	—	Host-system-error enable.
<2>	INTE	R/W	0	—	Interrupter enable.
<1>	HCRST	R/W1S/H	0	—	Host-controller reset.
<0>	R_S	R/W	0	—	Run/stop.

Cavium Confidential For Bowen Yang ALU 02/27/2014

XHCI Status Register
USBDRD(0..1)_UAHC_USBSTS

For information on this register, refer to the xHCI Specification, v1.0, section 5.4.2. This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:13>	—	RAZ	—	—	Reserved.
<12>	HCE	RO/H	0	—	Host-controller error.
<11>	CNR	RO/H	0	—	Controller not ready.
<10>	SRE	R/W1C/H	0	—	Save/restore error.
<9>	RSS	RO/H	0	—	Restore state status.
<8>	SSS	RO/H	0	—	Save state status.
<7:5>	—	RAZ	—	—	Reserved.
<4>	PCD	R/W1C/H	0	—	Port-change detect.
<3>	EINT	R/W1C/H	0	—	Event interrupt.
<2>	HSE	R/W1C/H	0	—	Host-system error. The typical software response to an HSE is to reset the core.
<1>	—	RAZ	—	—	Reserved.
<0>	HCH	RO/H	1	—	HC halted.

XHCI Page-Size Register
USBDRD(0..1)_UAHC_PAGESIZE

For information on this register, refer to the xHCI Specification, v1.0, section 5.4.3. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	—	RAZ	—	—	Reserved.
<15:0>	PAGESIZE	RO	0x1	0x1	Page size.

Cavium Confidential For Bowen Yang ALU 02/27/2014

XHCI Device Notification Control Register USBDRD(0..1)_UAHC_DNCTRL

For information on this register, refer to the xHCI Specification, v1.0, section 5.4.4.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	—	RAZ	—	—	Reserved.
<15:0>	N	R/W	0x0	—	Notification enable.

XHCI Command Ring Control Register USBDRD(0..1)_UAHC_CRCR

For information on this register, refer to the xHCI Specification, v1.0, section 5.4.5.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	CMD_RING_PTR	WO	0x0	—	Command ring pointer.
<5:4>	—	RAZ	—	—	Reserved.
<3>	CRR	RO/H	0	—	Command ring running.
<2>	CA	WO	0	0	Command abort.
<1>	CS	WO	0	0	Command stop.
<0>	RCS	WO	0	—	Ring cycle state.

Cavium Confidential For Bowen Yang AU 02/27/2014

XHCI Device Context Base-Address-Array Pointer Register USBDRD(0..1)_UAHC_DCBAAP

For information on this register, refer to the xHCI Specification, v1.0, section 5.4.6.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	DCBAAP	R/W	0x0	—	Device context base address array pointer.
<5:0>	—	RAZ	—	—	Reserved.

XHCI Configuration Register USBDRD(0..1)_UAHC_CONFIG

For information on this register, refer to the xHCI Specification, v1.0, section 5.4.7.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:8>	—	RAZ	—	—	Reserved.
<7:0>	MAXSLOTSEN	R/W	0x0	—	Maximum device slots enabled.

Cavium Confidential For Bowen Yang ALU 02/27/2014

XHCI Port Status and Control Registers USBDRD(0..1)_UAHC_PORTSC(0..1)

For information on this register, refer to the xHCI Specification, v1.0, section 5.4.8. Port 1 is USB3.0 SuperSpeed link, Port 0 is USB2.0 high-speed/full-speed/low-speed link. This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	WPR	WO	0	—	Warm port reset.
<30>	DR	RO/H	—	0	Device removable.
<29:28>	—	RAZ	—	—	Reserved.
<27>	WOE	R/W	0	0	Wake-on-overcurrent enable.
<26>	WDE	R/W	0	—	Wake-on-disconnect enable.
<25>	WCE	R/W	0	—	Wake-on-connect enable.
<24>	CAS	RO/H	0	—	Cold-attach status.
<23>	CEC	R/W1C/H	0	—	Port-configuration-error change.
<22>	PLC	R/W1C/H	0	—	Port-link-state change.
<21>	PRC	R/W1C/H	0	—	Port-reset change.
<20>	OCC	R/W1C/H	0	—	Overcurrent change.
<19>	WRC	R/W1C/H	0	—	Warm-port-reset change.
<18>	PEC	R/W1C/H	0	—	Port enabled/disabled change.
<17>	CSC	R/W1C/H	0	—	Connect-status change.
<16>	LWS	WO	0	—	Port-link-state write strobe.
<15:14>	PIC	R/W/H	0x0	0x0	Port indicator control.
<13:10>	PORTSPEED	RO/H	0x0	0x0	Port speed.
<9>	PP	R/W/H	1	1	Port power.
<8:5>	PLS	R/W/H	0x5	—	Port-link state.
<4>	PR	R/W1S/H	0	—	Port reset.
<3>	OCA	RO/H	0	0	Overcurrent active.
<2>	—	RAZ	—	—	Reserved.
<1>	PED	R/W1C/H	0	—	Port enabled/disabled.
<0>	CCS	RO/H	0	—	Current connect status.

Cavium Confidential For Bowen Yang ALU 02/27/2014

XHCI Port Power Management Status/Control Register (High-Speed) USBDRD(0..1)_UAHC_PORTPMSC_20(0)

For information on this register, refer to the xHCI Specification, v1.0, section 5.4.9. This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:28>	PORT_TEST_CONTROL	R/W	0x0	0x0	Port test control.
<27:17>	—	RAZ	—	—	Reserved.
<16>	HLE	R/W	0	—	Hardware LPM enable.
<15:8>	L1_DEVICE_SLOT	R/W	0x0	—	L1 device slot.
<7:4>	HIRD	R/W	0x0	—	Host-initiated resume duration.
<3>	RWE	R/W	0	—	Remove wake enable.
<2:0>	L1S	RO/H	0x0	—	L1 status.

XHCI Port Link Register (High-Speed) USBDRD(0..1)_UAHC_PORTLI_20(0)

For information on this register, refer to the xHCI Specification, v1.0, section 5.4.10. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	—	RAZ	—	—	Reserved.

XHCI Port Hardware LPM Control Register (High-Speed) USBDRD(0..1)_UAHC_PORTHLPMC_20(0)

For information on this register, refer to the xHCI Specification, vTBD, section TBD. This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:14>	—	RAZ	—	—	Reserved.
<13:10>	HIRDD	R/W	0x0	—	TBD
<9:2>	L1_TIMEOUT	R/W	0x0	—	TBD
<1:0>	HIRDM	R/W	0x0	—	Host-initiated resume-duration mode.

Cavium Confidential For Bowen Yang ALU 02/7/2014

XHCI Port Power Management Status/Control Register (SuperSpeed) USBDRD(0..1)_UAHC_PORTPMSC_SS(1)

For information on this register, refer to the xHCI Specification, vTBD, section TBD. This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:17>	—	RAZ	—	—	Reserved.
<16>	FLA	R/W	0	—	Force link PM accept.
<15:8>	U2_TIMEOUT	R/W	0x0	—	U2 timeout.
<7:0>	U1_TIMEOUT	R/W	0x0	—	U1 timeout.

XHCI Port Link Register (SuperSpeed) USBDRD(0..1)_UAHC_PORTLI_SS(1)

For information on this register, refer to the xHCI Specification, v1.0, section 5.4.10. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	—	RAZ	—	—	Reserved.
<15:0>	LINKERRORCOUNT	RO/H	0x0	—	Link error count.

XHCI Port Hardware LPM Control Register (SuperSpeed) USBDRD(0..1)_UAHC_PORTHLPMC_SS(1)

For information on this register, refer to the xHCI Specification, v1.0, section TBD. This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang ALU 08/27/2014

XHCI Microframe Index Register
USBDRD(0..1)_UAHC_MFINDEX

For information on this register, refer to the xHCI Specification, v1.0, section 5.5.1. This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:14>	—	RAZ	—	—	Reserved.
<13:0>	MFINDEX	RO/H	0x0	—	Microframe index.

XHCI Interrupt Management Register
USBDRD(0..1)_UAHC_IMAN(0)

For information on this register, refer to the xHCI Specification, v1.0, section 5.5.2.1. This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:2>	—	RAZ	—	—	Reserved.
<1>	IE	R/W	0	1	Interrupt enable.
<0>	IP	R/W1C/H	0	—	Interrupt pending.

XHCI Interrupt Moderation Register
USBDRD(0..1)_UAHC_IMOD(0)

For information on this register, refer to the xHCI Specification, v1.0, section 5.5.2.2. This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	IMODC	R/W	—	—	Interrupt moderation counter.
<15:0>	IMODI	R/W	0xFA0	0xFA0	Interrupt moderation interval.

Cavium Confidential For Bowen Yang ALU 02/27/2014

XHCI Event-Ring Segment-Table Size Register USBDRD(0..1)_UAHC_ERSTSZ(0)

For information on this register, refer to the xHCI Specification, v1.0, section 5.5.2.3.1. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	—	RAZ	—	—	Reserved.
<15:0>	ERSTSZ	R/W	0x0	—	Event-ring segment-table size.

XHCI Event-Ring Segment-Table Base-Address Register USBDRD(0..1)_UAHC_ERSTBA(0)

For information on this register, refer to the xHCI Specification, v1.0, section 5.5.2.3.2.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	ERSTBA	R/W	0x0	—	Event-ring segment-table base-address bits<63:6>.
<5:0>	—	RAZ	—	—	Reserved.

XHCI Event Ring Dequeue Register USBDRD(0..1)_UAHC_ERDP(0)

For information on this register, refer to the xHCI Specification, v1.0, section 5.5.2.3.3.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	ERDP	R/W	0x0	—	Event ring dequeue pointer bits <63:4>.
<3>	EHB	R/W1C/H	0	—	Event handler busy
<2:0>	DESI	R/W	0x0	—	Dequeue ERST segment index.

Cavium Confidential For Bowen Yang ALU 12/27/2014

XHCI Doorbell Registers
USBDRD(0..1)_UAHC_DB(0..64)

For information on this register, refer to the xHCI Specification, v1.0, section 5.6.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	DBSTREAMID	WO	0x0	—	Doorbell stream ID.
<15:8>	—	RAZ	—	—	Reserved.
<7:0>	DBTARGET	WO	0x0	—	Doorbell target.

XHCI Legacy Support Capability Register
USBDRD(0..1)_UAHC_USBLEGSUP

For information on this register, refer to the xHCI Specification, v1.0, section 7.1.1.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:25>	—	RAZ	—	—	Reserved.
<24>	HC_OS_OWNED_SEMAPHORES	R/W	0	—	HC OS-owned semaphore.
<23:17>	—	RAZ	—	—	Reserved.
<16>	HC_BIOS_OWNED_SEMAPHORES	R/W	0	—	HC BIOS-owned semaphore.
<15:8>	NEXTCAPPTR	RO	0x4	0x4	Next xHCI extended-capability pointer.
<7:0>	CAPID	RO	0x1	0x1	Capability ID = USB legacy support.

Cavium Confidential For Bowen Yang ALU 02.12.2014

XHCI Legacy Support Control/Status Register USBDRD(0..1)_UAHC_USBLEGCTLSTS

For information on this register, refer to the xHCI Specification, v1.0, section 7.1.2. Note that the SMI interrupts are not connected to anything in a CN71XX configuration. This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	SMI_ON_BAR	R/W1C/H	0	—	System management interrupt on BAR. Never generated.
<30>	SMI_ON_PCI_ COMMAND	R/W1C/H	0	—	System management interrupt on PCI command. Never generated.
<29>	SMI_ON_OS_ OWNERSHIP	R/W1C/H	0	—	System management interrupt on OS ownership change. This bit is set to 1 whenever UAHC_USBLEGSUP[HC_OS_OWNED_SEMAPHORES] transitions.
<28:21>	—	RAZ	—	—	Reserved.
<20>	SMI_ON_ HOSTSYSTEMERR	RO/H	0	—	System-management interrupt on host-system error. Shadow bit of UAHC_USBSTS[HSE]. Refer to xHCI Section 5.4.2 for definition and effects of the events associated with this bit being set to 1. To clear this bit to a 0, system software must write a 1 to UAHC_USBSTS[HSE].
<19:17>	—	RAZ	—	—	Reserved.
<16>	SMI_ON_EVENT_ INTERRUPT	RO/H	0	—	System-management interrupt on event interrupt. Shadow bit of UAHC_USBSTS[EINT]. Refer to xHCI Section 5.4.2 for definition. This bit automatically clears when [EINT] clears and sets when [EINT] sets.
<15>	SMI_ON_BAR_EN	R/W	0	—	System-management interrupt on BAR enable.
<14>	SMI_ON_PCI_ COMMAND_EN	R/W	0	—	System-management interrupt on PCI command enable.
<13>	SMI_ON_OS_ OWNERSHIP_EN	R/W	0	—	System-management interrupt on OS ownership enable.
<12:5>	—	RAZ	—	—	Reserved.
<4>	SMI_ON_ HOSTSYSTEMERR_EN	R/W	0	—	System-management interrupt on host-system error enable
<3:1>	—	RAZ	—	—	Reserved.
<0>	USB_SMI_EN	R/W	0	—	USB system-management interrupt enable.

Cavium Confidential For Bowen Ning 6/2/2014

XHCI Supported-Protocol-Capability (USB 2.0) Register 0 USBDRD(0..1)_UAHC_SUPTPRT2_DW0

For information on this register, refer to the xHCI Specification, v1.0, section 7.2. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	MAJORREV	RO	0x2	0x2	Major revision.
<23:16>	MINORREV	RO	0x0	0x0	Minor revision.
<15:8>	NEXTCAPPTR	RO	0x4	0x4	Next capability pointer.
<7:0>	CAPID	RO	0x2	0x2	Capability ID = supported protocol.

XHCI Supported-Protocol-Capability (USB 2.0) Register 1 USBDRD(0..1)_UAHC_SUPTPRT2_DW1

For information on this register, refer to the xHCI Specification, v1.0, section 7.2. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	NAME	RO	0x20425355	0x20425355	Name string: “USB”.

XHCI Supported-Protocol-Capability (USB 2.0) Register 2 USBDRD(0..1)_UAHC_SUPTPRT2_DW2

For information on this register, refer to the xHCI Specification, v1.0, section 7.2. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:28>	PSIC	RO	0x0	—	Protocol speed ID count.
<27:21>	—	RAZ	—	—	Reserved.
<20>	BLC	RO	0	—	BESL LPM capability.
<19>	HLC	RO	1	—	Hardware LMP capability.
<18>	IHI	RO	0	—	Integrated hub implemented.
<17>	HSO	RO	0	—	High-speed only.
<16>	—	RAZ	—	—	Reserved.
<15:8>	COMPATPRTCNT	RO	0x1	0x1	Compatible port count.
<7:0>	COMPATPRTOFF	RO	0x1	0x1	Compatible port offset.

XHCI Supported-Protocol-Capability (USB 2.0) Register 3 USBDRD(0..1)_UAHC_SUPTPRT2_DW3

For information on this register, refer to the xHCI Specification, vTBD, section 7.2. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:5>	—	RAZ	—	—	Reserved.
<4:0>	PROTSLOTTYPE	RO	0x0	0x0	Protocol slot type.

Cavium Confidential For Bowen Yang ALU 02/27/2014

XHCI Supported-Protocol-Capability (USB 3.0) Register 0 USBDRD(0..1)_UAHC_SUPTPRT3_DW0

For information on this register, refer to the xHCI Specification, v1.0, section 7.2. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	MAJORREV	RO	0x3	0x3	Major revision.
<23:16>	MINORREV	RO	0x0	0x0	Minor revision.
<15:8>	NEXTCAPPTR	RO/H	0x0	0x0	Next capability pointer. Value depends on USBDRD(0..1)_UAHC_GUCTL[EXTCAPSUPTEN]. If EXTCAPSUPTEN = 0, value is 0x0. If EXTCAPSUPTEN = 1, value is 0x4.
<7:0>	CAPID	RO	0x2	0x2	Capability ID = supported protocol.

XHCI Supported-Protocol-Capability (USB 3.0) Register 1 USBDRD(0..1)_UAHC_SUPTPRT3_DW1

For information on this register, refer to the xHCI Specification, v1.0, section 7.2. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	NAME	RO	0x20425355	0x20425355	Name string: “USB”.

XHCI Supported-Protocol-Capability (USB 3.0) Register 2 USBDRD(0..1)_UAHC_SUPTPRT3_DW2

For information on this register, refer to the xHCI Specification, v1.0, section 7.2. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:28>	PSIC	RO	0x0	—	Protocol speed ID count.
<27:16>	—	RAZ	—	—	Reserved.
<15:8>	COMPATPRTCNT	RO	0x1	0x1	Compatible port count.
<7:0>	COMPATPRTOFF	RO	0x2	0x2	Compatible port offset.

XHCI Supported-Protocol-Capability (USB 3.0) Register 3 USBDRD(0..1)_UAHC_SUPTPRT3_DW3

See XHCI specification vTBD, section 7.2. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:5>	—	RAZ	—	—	Reserved.
<4:0>	PROTSLOTTYPE	RO	0x0	0x0	Protocol slot type.

Cavium Confidential For Bowen Yang ALU 02/27/2014

UAHC Bus Configuration Register 0
USBDRD(0..1)_UAHC_GSBUSCFG0

This register can be used to configure the core after power-on or a change in mode of operation. This register mainly contains AXI system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on AXI. When INCRBRSTENA is enabled, it has the highest priority over other burst lengths. The core always performs the largest burst when enabled.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:28>	DATRDREQINFO	R/W	0x0	0x0	AXI-cache for data-read operations. Always set to 0x0.
<27:24>	DESRDREQINFO	R/W	0x0	0x0	AXI-cache for descriptor-read operations. Always set to 0x0.
<23:20>	DATWRREQINFO	R/W	0x0	0x0	AXI-cache for data-write operations. Always set to 0x0.
<19:16>	DESWRREQINFO	R/W	0x0	0x0	AXI-cache for descriptor-write operations. Always set to 0x0.
<15:12>	—	R/W	0x0	0x0	Reserved.
<11>	DATBIGEND	R/W	0	0	Data access is big-endian. Keep this set to 0 (little-endian) and use the UCTL(0)SHIM_CFG[DMA_ENDIAN_MODE] setting instead.
<10>	DESCBIGEND	R/W	0	0	Descriptor access is big-endian. Keep this set to 0 (little-endian) and use the UCTL(0)SHIM_CFG[DMA_ENDIAN_MODE] setting instead.
<9:8>	RESERVED_0	RAZ	0x0	0x0	Reserved. Read must ignore.
<7>	INCR256BRSTENA	R/W	0	0	INCR256 burst-type enable. Always set to 0.
<6>	INCR128BRSTENA	R/W	0	0	INCR128 burst-type enable. Always set to 0.
<5>	INCR64BRSTENA	R/W	0	0	INCR64 burst-type enable. Always set to 0.
<4>	INCR32BRSTENA	R/W	0	0	INCR32 burst-type enable. Always set to 0.
<3>	INCR16BRSTENA	R/W	1	1	INCR16 burst-type enable. Allows the AXI master to generate INCR 16-beat bursts.
<2>	INCR8BRSTENA	R/W	1	1	INCR8 burst-type enable. Allows the AXI master to generate INCR eight-beat bursts.
<1>	INCR4BRSTENA	R/W	1	1	INCR4 burst-type enable. Allows the AXI master to generate INCR four-beat bursts.
<0>	INCRBRSTENA	R/W	1	1	Undefined-length INCR burst-type enable. This bit determines the set of burst lengths to be utilized by the master interface. It works in conjunction with the GSBUSCFG0[7:1] enables (INCR*BRSTENA). If disabled, the AXI master will use only the following burst lengths: 1, 4, 8, 16 (assuming the INCR*BRSTENA are set to their reset values) If enabled, the AXI master uses any length less than or equal to the largest-enabled burst length based on the INCR*BRSTENA fields.

Cavium Confidential For Bowen Yang 02/27/2014

UAHC Bus Configuration Register 1
USBDRD(0..1)_UAHC_GSBUSCFG1

This register can be used to configure the core after power-on or a change in mode of operation. This register mainly contains AXI system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on AXI.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:13>	—	RAZ	—	—	Reserved.
<12>	EN1KPAGE	R/W	0	0	1K page-boundary enable. 0 = AXI breaks transfers at the 4K page boundary (default) 1 = AXI breaks transfers at the 1K page boundary
<11:8>	PIPETRANSLIMIT	R/W	0xF	0xF	AXI pipelined transfers burst-request limit. Controls the number of outstanding pipelined transfers requests the AXI master will push to the AXI slave. Once the AXI master reaches this limit, it does not make more requests on the AXI ARADDR and AWADDR buses until the associated data phases complete. This field is encoded as follows: 0x0 = 1 request 0x1 = 2 requests 0x2 = 3 requests 0x3 = 4 requests 0x4 = 5 requests 0x5 = 6 requests 0x6 = 7 requests 0x7 = 8 requests 0x8 = 9 requests 0x9 = 10 requests 0xA = 11 requests 0xB = 12 requests 0xC = 13 requests 0xD = 14 requests 0xE = 15 requests 0xF = 16 requests
<7:0>	RESERVED_1	R/W	0x0	0x0	Reserved. Read must ignore.

Cavium Confidential For Bowen Yang ALU 02/27/2014

UAHC TX Threshold Control Register USBDRD(0..1)_UAHC_GTXTHRCFG

In a normal case, a TX burst starts as soon as one packet is prefetched. This works well as long as the system bus is faster than the USB3.0 bus (a 1024-bytes packet takes ~2.2 µs on the USB bus in SuperSpeed mode). If the system bus latency is larger than 2.2 µs to access a 1024-byte packet, then starting a burst on 1-packet condition leads to an early abort of the burst causing unnecessary performance reduction. This register allows the configuration of threshold and burst size control. This feature is enabled by [USBTXPKTCNTSEL].

Transmit Path:

- The TX Threshold is controlled by [USBTXPKTCNT], and the TX burst size is controlled by [USBMAXTXBURSTSIZE].
- Selecting optimal TX FIFO size, TX Threshold, and Tx burst size avoids TX burst aborts due to an underrun if the system bus is slower than USB. Once in a while an underrun is OK, and there is no functional issue.
- A larger threshold affects the performance, since the scheduler is idle during this time.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:30>	RESERVED_0	RAZ	0x0	0x0	Reserved. Read must ignore.
<29>	USBTXPKTCNTSEL	R/W	0	0	USB transmit packet-count enable. Enables/disables the USB transmission multipacket thresholding: 0 = USB transmission multipacket thresholding is disabled, the core can only start transmission on the USB after the entire packet has been fetched into the corresponding TXFIFO. 1 = USB transmission multipacket thresholding is enabled. The core can only start transmission on the USB after USBTXPKTCNT amount of packets for the USB transaction (burst) are already in the corresponding TXFIFO. This mode is only used for SuperSpeed.
<28>	—	RAZ	—	—	Reserved.
<27:24>	USBTXPKTCNT	R/W	0x0	0x0	USB transmit-packet count. Specifies the number of packets that must be in the TXFIFO before the core can start transmission for the corresponding USB transaction (burst). This field is only valid when USBTXPKTCNTSEL = 1. Valid values are from 0x1 to 0xF. **NOTE:** This field must be ≤ USBMAXTXBURSTSIZE.

Cavium Confidential FOI Bowen Yang 01/20/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<23:16>	USBMAXTXBURSTSIZE	R/W	0x0	0x0	USB maximum TX burst size. When USBTXPKTCNTSEL = 1, this field specifies the maximum bulk OUT burst the core should do. When the system bus is slower than the USB, TX FIFO can underrun during a long burst. Program a smaller value to this field to limit the TX burst size that the core can do. It only applies to SuperSpeed Bulk, Isochronous, and Interrupt OUT endpoints in the host mode. Valid values are from 0x1 to 0x10.
<15:14>	RESERVED_2	R/W	0x0	0x0	Reserved. Read must ignore.
<13:11>	—	RAZ	—	—	Reserved.
<10:0>	RESERVED_4	R/W	0x0	0x0	Reserved.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

UAHC RX Threshold Control Register USBDRD(0..1)_UAHC_GRXTHRCFG

In a normal case, an RX burst starts as soon as 1-packet space is available. This works well as long as the system bus is faster than the USB3.0 bus (a 1024-bytes packet takes ~2.2 µs on the USB bus in SuperSpeed mode). If the system bus latency is larger than 2.2 µs to access a 1024-byte packet, then starting a burst on 1-packet condition leads to an early abort of the burst causing unnecessary performance reduction. This register allows the configuration of threshold and burst size control. This feature is enabled by USBRXPKTCNTSEL.

Receive Path:

- The RX threshold is controlled by USBRXPKTCNT and the RX burst size is controlled by USBMAXRXBURSTSIZE.
- Selecting optimal RX FIFO size, RX Threshold, and RX burst size avoids RX burst aborts due to overrun if the system bus is slower than USB. Once in a while overrun is OK, and there is no functional issue.
- Some devices do not support terminating ACK retry. With these devices, host cannot set ACK=0 and Retry=0 and do retry later and you have to retry immediately. For such devices, minimize retry due to underrun. Setting threshold and burst size guarantees this.
- A larger RX threshold affects the performance since the scheduler is idle during this time.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:30>	—	RAZ	—	—	Reserved.
<29>	USBRXPKTCNTSEL	R/W	0	0	USB receive-packet-count enable. Enables/disables the USB reception multipacket thresholding: 0 = the core can only start reception on the USB when the RX FIFO has space for at least one packet. 1 = the core can only start reception on the USB when the RX FIFO has space for at least USBRXPKTCNT amount of packets. This mode is only used for SuperSpeed.
<28>	—	RAZ	—	—	Reserved.
<27:24>	USBRXPKTCNT	R/W	0x0	0x0	USB receive-packet count. In host-mode, specifies space (in number of packets) that must be available in the RX FIFO before the core can start the corresponding USB RX transaction (burst). In device mode, specifies the space (in number of packets) that must be available in the RX FIFO before the core can send ERDY for a flow-controlled enpoint. This field is only valid when USBRXPKTCNTSEL = 1. The valid values are from 0x1 to 0xF. **NOTE:** This field must be ≤ USBMAXRXBURSTSIZE.

Cavium Confidential For Bowen Yang A-D Internal

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<23:19>	USBMAXRXBURSTSIZE	R/W	0x0	0x0	USB maximum receive-burst size. In host-mode, specifies the maximum bulk IN burst the core should do. When the system bus is slower than the USB, RX FIFO can overrun during a long burst. Program a smaller value to this field to limit the RX burst size that the core can do. It only applies to SuperSpeed Bulk, Isochronous, and Interrupt IN endpoints in the host mode. In device mode, specified the NUMP value that will be sent in ERDy for an OUT endpoint. This field is only valid when USBRXPKTCNTSEL = 1. The valid values are from 0x1 to 0x10.
<18:16>	—	RAZ	—	—	Reserved.
<15>	—	R/W	0	0	Reserved.
<14:11>	—	RAZ	—	—	Reserved.
<10:0>	—	R/W	0x0	0x0	Reserved. Read must ignore.

Cavium Confidential For Bowen Yang ALU 02/27/2014

UAHC Control Register
USBDRD(0..1)_UAHC_GCTL

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	FIELD NAME	Access	Reset Value	Typical Value	Field Description
<31:19>	PWRDNSCALE	R/W	0x618	0x618	Power down scale. The USB3 suspend-clock input replaces pipe3_rx_pclk as a clock source to a small part of the USB3 core that operates when the SuperSpeed PHY is in its lowest power (P3) state, and therefore does not provide a clock. This field specifies how many suspend-clock periods fit into a 16 kHz clock period. When performing the division, round up the remainder. For example, when using an 32-bit PHY and 25-MHz suspend clock, PWRDNSCALE = 25000 kHz/16 kHz = 1563 (rounded up). The minimum suspend-clock frequency is 32 KHz, and maximum suspend-clock frequency is 125 MHz. The LTSSM uses Suspend clock for 12-ms and 100-ms timers during suspend mode. According to the USB 3.0 specification, the accuracy on these timers is 0% to +50%. 12 ms + 0~+50% accuracy = 18 ms (Range is 12 ms – 18 ms) 100 ms + 0~+50% accuracy = 150 ms (Range is 100 ms – 150 ms). The suspend clock accuracy requirement is: (12,000/62.5) × (GCTL[31:19]) × actual suspend_clk_period should be between 12,000 and 18,000 (100,000/62.5) × (GCTL[31:19]) × actual suspend_clk_period should be between 100,000 and 150,000 For example, if your suspend_clk frequency varies from 7.5 MHz to 10.5MHz, then the value needs to programmed is: Power Down Scale = 10500/16 = 657 (rounded up; and fastest frequency used).
<18>	MASTERFILTBYPASS	R/W	0	0	Master filter bypass. Not relevant for Cavium's configuration.
<17>	BYPSSETADDR	R/W	0	0	Bypass SetAddress in device mode. Always set to 0.
<16>	U2RSTECN	R/W	0	0	If the super speed connection fails during POLL or LMP exchange, the device connects at non-SuperSpeed mode. If this bit is set, then device attempts three more times to connect at SuperSpeed, even if it previously failed to operate in SuperSpeed mode. This bit is only applicable in device mode.

Cavium Confidential For Bowery Yang ALL 02/26/2014

Bit Pos	FIELD NAME	Access	Reset Value	Typical Value	Field Description
<15:14>	FRMSCLDWN	R/W	0x0	0x0	Frame scale down. Scales down device view of a SOF/ USOF/ITP duration. For SuperSpeed/high-speed mode: 0x3 = interval is 15.625 µs 0x2 = interval is 31.25 µs 0x1 = interval is 62.5 µs 0x0 = interval is 125 µs For full-speed mode, the scale-down value is multiplied by 8.
<13:12>	PRTCAPDIR	R/W	0x2	0x2	Port capability direction. 0x1 for host configurations 0x2 for device configurations
<11>	CORESOFTRESET	R/W	0	0	Core soft reset: 1 = soft reset to core, 0 = no soft reset. Clears the interrupts and all the USBDRD(0..1)_UAHC_* CSRs except the following registers: USBDRD(0..1)_UAHC_GCTL USBDRD(0..1)_UAHC_GUCTL USBDRD(0..1)_UAHC_GSTS USBDRD(0..1)_UAHC_GRLSID USBDRD(0..1)_UAHC_GGPIO USBDRD(0..1)_UAHC_GUID USBDRD(0..1)_UAHC_GUSB2PHYCFG[*] USBDRD(0..1)_UAHC_GUSB3PIPECTL[*]. When you reset PHYs (using USBDRD(0..1)_UAHC_GUBS3PHYCFG or USBDRD(0..1)_UAHC_GUSB3PIPECTL), you must keep the core in reset state until PHY clocks are stable. This controls the bus, RAM, and MAC domain resets. **NOTE:** Under soft reset, accesses to USBDRD(0..1)_UAHC_* CSRs other than USBDRD(0..1)_UAHC_GCTL may fail (timeout). **NOTE:** This bit is for debug purposes only. Use USBDRD(0..1)_UAHC_USBCMD[HCRST] or USBDRD(0..1)_UAHC_DCTL[CSFTRST] in device mode for soft reset.

Cavium Confidential
Bowen Yang
For
Review
02/27/2019

Bit Pos	FIELD NAME	Access	Reset Value	Typical Value	Field Description
<10>	SOFITPSYNC	R/W	0	0	Synchronize ITP to reference clock. This bit is only used in host-mode. In host mode, if this bit is set to: 0 The core keeps the UTMI/ULPI PHY on the first port in non-suspended state whenever there is a SuperSpeed port that is not in Rx.Detect, SS.Disable, and U3 state. 1 The core keeps the UTMI/ULPI PHY on the first port in non-suspended state whenever the other non-SuperSpeed ports are not in suspended state. This feature is useful because it saves power by suspending UTMI/ULPI when SuperSpeed only is active and it helps resolve when the PHY does not transmit a host resume unless it is placed in suspend state. This bit must be programmed as a part of initialization at power-on reset, and must not be dynamically changed afterwards. **NOTE:** USBDRD(0..1)_UAHC_USB2PHYCFG[*][SUSPHY] eventually decides to put the UTMI/ULPI PHY in to suspend state. In addition, when this bit is set to 1, the core generates ITP off of the REF_CLK-based counter. Otherwise, ITP and SOF are generated off of UTMI/ULPI_CLK[0] based counter. To program the reference clock period inside the core, refer to USBDRD(0..1)_UAHC_GUCTL[REFCLKPER]. If you plan to enable hardware-based LPM (USBDRD(0..1)_UAHC_PORTPMSC[HLE] = 1), this feature cannot be used. Turn off this feature by setting this bit to 0 and use the USBDRD(0..1)_UAHC_GFLADJ [GFLADJ_REFCLK_LPM_SEL] feature. If you set this bit to 1, the USBDRD(0..1)_UAHC_GUSB2PHYCFG [U2_FREECLK_EXISTS] bit must be set to 0.
<9>	U1U2TIMERSCALE	R/W	0	0	Disable U1/U2 timer scaledown. If set to 1, along with SCALEDOWN = 0x1, disables the scale down of U1/U2 inactive timer values. This is for simulation mode only.
<8>	DEBUGATTACH	R/W	0	0	Debug attach. When this bit is set: • SuperSpeed link proceeds directly to the polling-link state (USBDRD(0..1)_UAHC_DCTL[RS] = 1) without checking remote termination. • Link LFPS polling timeout is infinite • Polling timeout during TS1 is infinite (in case link is waiting for TXEQ to finish).
<7:6>	RAMCLKSEL	R/W	0x0	0x0	RAM clock select. Always keep set to 0x0.

Cavium Confidential For Bowen Yang, ADI 9/7/2014

Bit Pos	FIELD NAME	Access	Reset Value	Typical Value	Field Description
<5:4>	SCALEDOWN	R/W	0x0	0x0	Scale-down mode. When scale-down mode is enabled for simulation, the core uses scaled-down timing values, resulting in faster simulations. When scale-down mode is disabled, actual timing values are used. This is required for hardware operation. High-speed/full-speed/low-speed modes: 0x0 = Disables all scale-downs. Actual timing values are used. 0x1 = Enables scale-down of all timing values. These include: • Speed enumeration • HNP/SRP • Suspend and resume 0x2 = N/A 0x3 = Enables bits <0> and <1> scale-down timing values. SuperSpeed mode: 0x0 = Disables all scale-downs. Actual timing values are used. 0x1 = Enables scaled down SuperSpeed timing and repeat values including: • Number of TxEq training sequences reduce to eight • LFPS polling burst time reduce to 100 ns • LFPS warm reset receive reduce to 30 µs. 0x2 = No TxEq training sequences are sent. Overrides bit<4>. 0x3 = Enables bits<0> and <1> scale-down timing values.
<3>	DISSCRAMBLE	R/W	0	0	Disable scrambling. Transmit request to link partner on next transition to recovery or polling.
<2>	U2EXIT_LFPS	R/W	1	1	LFPS U2 exit. 0 = The link treats 248ns LFPS as a valid U2 exit. 1 = The link waits for 8µs of LFPS before it detects a valid U2 exit. This bit is added to improve interoperability with a third party host controller. This host controller in U2 state while performing receiver detection generates an LFPS glitch of about 4s duration. This causes the device to exit from U2 state because the LFPS filter value is 248ns. With the new functionality enabled, the device can stay in U2 while ignoring this glitch from the host controller.
<1>	—	RAZ	—	—	Reserved.
<0>	DSBLCLKGTNG	R/W	0	0	Disable clock gating. When set to 1 and the core is in low-power mode, internal clock gating is disabled, which means the clocks are always running. This bit can be set to 1 after power-up reset.

Cavium Confidential For Bowen Yang AMD 02-27-2014

UAHC Global Power Management Status Register USBDRD(0..1)_UAHC_GPMSTS

This debug register gives information on which event caused the hibernation exit. These registers are for debug purposes. They provide debug information on the internal status and state machines. Global debug registers have design-specific

information, and are used by for debugging purposes. These registers are not intended to be used by the customer. If any debug assistance is needed for the silicon, contact customer support with a dump of these registers.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:28>	PORTSEL	WO	—	—	This field selects the port number. Always 0x0.
<27:17>	—	RAZ	—	—	Reserved.
<16:12>	U3WAKEUP	RO/H	0x0	—	This field gives the USB 3.0 port wakeup conditions. bit<12> = overcurrent detected bit<13> = resume detected bit<14> = connect detected bit<15> = disconnect detected bit<16> = last connection state
<11:10>	—	RAZ	—	—	Reserved.
<9:0>	U2WAKEUP	RO/H	0x0	—	This field indicates the USB 2.0 port wakeup conditions. bit<0> = overcurrent detected bit<1> = resume detected bit<2> = connect detected bit<3> = disconnect detected bit<4> = last connection state bit<5> = ID change detected bit<6> = SRP request detected bit<7> = ULPI interrupt detected bit<8> = USB reset detected bit<9> = resume detected changed

Cavium Confidential For Bowen Yang ALU 02/27/2014

UAHC Core Status Register
USBDRD(0..1)_UAHC_GSTS

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:20>	CBELT	RO/H	0x3E8	0x3E8	Current BELT value. In host mode, indicates the minimum value of all received device BELT values and the BELT value that is set by the set latency tolerance value command.
<19:8>	—	RAZ	—	—	Reserved.
<7>	HOST_IP	RO/H	0	—	Host interrupt pending. Indicates that there is a pending interrupt pertaining to xHC in the host-event queue.
<6>	—	RAZ	—	—	Reserved.
<5>	CSRTIMEOUT	R/W1C/H	0	0	CSR timeout. When set to 1, indicates that software performed a write or read operation to a core register that could not be completed within 0xFFFF controller-clock cycles.
<4>	BUSERRADDRVLD	R/W1C/H	0	0	Bus-error address valid. Indicates that USBDRD(0..1)_UAHC(0)_GBUSERRADDR_* is valid and reports the first bus address that encounters a bus error.
<3:2>	—	RAZ	—	—	Reserved.
<1:0>	CURMOD	RO	0x0	0x0	Current mode of operation: 0x0 for device, 0x1 for host.

UAHC Global User Control Register 1
USBDRD(0..1)_UAHC_GUCTL1

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:2>	—	R/W	0x0	0x0	Reserved.
<1>	OVRLD_L1_SUSP_COM	R/W	0	0	Always set to 0.
<0>	LOA_FILTER_EN	R/W	0	0	If this bit is set, the USB 2.0 port babble is checked at least three consecutive times before the port is disabled. This prevents false triggering of the babble condition when using low quality cables. This bit is only valid in host mode.

Cavium Confidential For Bowen Yang ALU 9/27/2014

UAHC Release ID Register
USBDRD(0..1)_UAHC_GRLSID

This is a read-only register that contains the release number of the core. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	RELEASEID	RO	0x5533250A	0x5533250A	Software can use this register to configure release-specific features in the driver.

UAHC Core General-Purpose I/O Register
USBDRD(0..1)_UAHC_GGPIO

The application can use this register for general purpose input and output ports or for debugging.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	GPO	R/W	0x0	0x0	General purpose output. These outputs are not connected to anything. Can be used as scratch.
<15:0>	GPI	RO	0x0	0x0	General purpose input. These inputs are tied 0x0.

UAHC Core User ID Register
USBDRD(0..1)_UAHC_GUID

This is a read/write register containing the User ID. The power-on value for this register is specified as the User Identification Register. This register can be used in the following ways:

- To store the version or revision of your system
- To store hardware configurations that are outside of the core
- As a scratch register

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	USERID	R/W	0xDECAFBAD	—	User ID. Application-programmable ID field.

Cavium Confidential For Power Yang ALU 03/27/2014

UAHC Core User-Control Register USBDRD(0..1)_UAHC_GUCTL

This register provides a few options for the software to control the core behavior in the host mode. Most of the options are used to improve host inter-operability with different devices.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:22>	REFCLKPER	R/W	0x8	0x8	Reference-clock period. Indicates (in terms of ns) the period of REF_CLK. The default value is set to 0x8 (8 ns/125 MHz). This field must be updated during power on initialization if USBDRD(0..1)_UAHC_GCTL[SOFITPSYNC] = 1 or USBDRD(0..1)_UAHC_GFLADJ [GFLADJ_REFCLK_LPM_SEL] = 1. The programmable maximum value 62 ns, and the minimum value is 8 ns. You use a reference clock with a period that is a integer multiple, so that ITP can meet the jitter margin of 32 ns. The allowable REF_CLK frequencies whose period is not integer multiples are 16/17/19.2/24/39.7 MHz. This field should not be set to 0x0 at any time. If you do not plan to use this feature, then you need to set this field to 0x8, the default value.
<21>	NOEXTRDL	R/W	0	0	No extra delay between SOF and the first packet. Some high-speed devices misbehave when the host sends a packet immediately after an SOF. However, adding an extra delay between an SOF and the first packet can reduce the USB data rate and performance. This bit is used to control whether the host should wait for 2 µs before it sends the first packet after a SOF, or not. You can set this bit to 1 to improve the performance if those problematic devices are not a concern in your host environment. 0 = host waits for 2 µs after an SOF before it sends the first USB packet. 1 = host does not wait after an SOF before it sends the first USB packet.
<20:18>	PSQEXTRRESSP	R/W	0x0	0x0	PSQ extra reserved space. This is a debug feature, and is not intended for normal usage. This parameter specifies how much additional space in the PSQ (protocol-status queue) must be reserved before the U3PTL initiates a new USB transaction and burst beats.
<17>	SPRSCTRLTRANSEN	R/W	0	0	Sparse control transaction enable. Some devices are slow in responding to control transfers. Scheduling multiple transactions in one microframe/frame can cause these devices to misbehave. If this bit is set to 1, the host controller schedules transactions for a control transfer in different microframes/frames.

Cavium Confidential
Bowen Yang
02/21/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<16>	RESBWHSEPS	R/W	0	0	Reserving 85% bandwidth for high-speed periodic EPs. By default, host controller reserves 80% of the bandwidth for periodic EPs. If this bit is set, the bandwidth is relaxed to 85% to accommodate two high-speed, high-bandwidth ISOC EPs. USB 2.0 required 80% bandwidth allocated for ISOC traffic. If two high bandwidth ISOC devices (HD webcams) are connected, and if each requires 1024-bytes × 3 packets per microframe, then the bandwidth required is around 82%. If this bit is set to 1, it is possible to connect two webcams of 1024 bytes × 3 payload per microframe each. Otherwise, you may have to reduce the resolution of the webcams.
<15>	CMDEVADDR	R/W	1	1	Compliance mode for device address. When set to 1, slot ID can have different value than device address if max_slot_enabled < 128. 1 = increment device address on each address device command. 0 = device address is equal to slot ID. The xHCI compliance requires this bit to be set to 1. The 0 mode is for debug purpose only. This allows you to easily identify a device connected to a port in the Lecroy or Eliisys trace during hardware debug. This bit is used in host mode only.
<14>	USBHSTINAUTORETRYEN	R/W	0	0	Host IN auto-retry enable. When set, this field enables the auto-retry feature. For IN transfers (non-isochronous) that encounter data packets with CRC errors or internal overrun scenarios, the auto-retry feature causes the host core to reply to the device with a non-terminating retry ACK (i.e. an ACK transaction packet with Retry = 1 and NumP ≠ 0). If the auto-retry feature is disabled (default), the core responds with a terminating retry ACK (i.e. an ACK transaction packet with Retry = 1 and NumP = 0).
<13>	ENOVERLAPCHK	R/W	0	0	Enable check for LFPS overlap during remote Ux Exit. If this bit is set to: 1 = The SuperSpeed link, when exiting U1/U2/U3, waits for either the remote link LFPS or TS1/TS2 training symbols before it confirms that the LFPS handshake is complete. This is done to handle the case where the LFPS glitch causes the link to start exiting from the low power state. Looking for the LFPS overlap makes sure that the link partner also sees the LFPS. 0 = When the link exists U1/U2/U3 because of a remote exit, it does not look for an LFPS overlap.

Cavium Confidential For Bowen Yang 023.rpt

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<12>	EXTCAPSUPTEN	R/W	0	0	External extended capability support enable. If disabled, a read USBDRD(0..1)_UAHC_SUPTPRT3_DW0 [NEXTCAPPTR] returns 0 in the next capability pointer field. This indicates there are no more capabilities. If enabled, a read to USBDRD(0..1)_UAHC_SUPTPRT3_DW0[NEXTCAPPTR] returns 4 in the next capability pointer field. Always set to 0x0.
<11>	INSRTEXTRFSBODI	R/W	0	0	Insert extra delay between full-speed bulk OUT transactions. Some full-speed devices are slow to receive bulk OUT data and can get stuck when there are consecutive bulk OUT transactions with short inter-transaction delays. This bit is used to control whether the host inserts extra delay between consecutive bulk OUT transactions to a full-speed endpoint. 1 = Host inserts about 12 µs extra delay between consecutive bulk OUT transactions to an full-speed endpoint to work around the device issue. 0 = Host does not insert extra delay. **NOTE:** Setting this bit to 1 reduces the bulk OUT transfer performance for most of the full-speed devices.
<10:9>	DTCT	R/W	0x0	0x0	Device timeout coarse tuning. This field determines how long the host waits for a response from device before considering a timeout. The core first checks the DTCT value. If it is 0, then the timeout value is defined by the DTFT. If it is non-zero, then it uses the following timeout values: 0x0 = 0 µs; use DTFT value instead 0x1 = 500 µs 0x2 = 1.5 ms 0x3 = 6.5 ms
<8:0>	DTFT	R/W	0x10	0x10	Device timeout fine tuning. This field determines how long the host waits for a response from a device before considering a timeout. For DTFT to take effect, DTCT must be set to 0x0. The DTFT value specifies the number of 125MHz clock cycles × 256 to count before considering a device timeout. For the 125 MHz clock cycles (8 ns period), this is calculated as follows: [DTFT value] × 256 × 8 (ns) **DTFT Value** / **Calculation** / **Timeout** 0x2 / 2 × 256 × 8 / 4 µs 0x5 / 5 × 256 × 8 / 10 µs 0xA / 10 × 256 × 8 / 20 µs 0x10 / 16 × 256 × 8 / 32 µs 0x19 / 25 × 256 × 8 / 51 µs 0x31 / 49 × 256 × 8 / 100 µs 0x62 / 98 × 256 × 8 / 200 µs

Cavium Confidential For Bowen Yang ALU 6/27/2014

UAHC Bus-Error-Address Register
USBDRD(0..1)_UAHC_GBUSERRADDR

When the AXI Master Bus returns *Error* response, *the SoC Bus Error* is generated. In the host mode, the host_system_err port indicates this condition. In addition, it is also indicated in USBDRD(0..1)_UAHC_USBSTS[HSE]. Due to the nature of AXI, it is possible that multiple AXI transactions are active at a time. The host controller does not keep track of the start address of all outstanding transactions. Instead, it keeps track of the start address of the DMA transfer associated with all active transactions. It is this address that is reported in the GBUSERRADDR when a bus error occurs. For example, if the host controller initiates a DMA transfer to write 1k of packet data starting at buffer address 0xABCD0000, and this DMA is broken up into multiple 256B bursts on the AXI, then if a bus error occurs on any of these associated AXI transfers, the GBUSERRADDR reflects the DMA start address of 0xABCD0000 regardless of which AXI transaction received the error.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	FIELD NAME	Access	Reset Value	Typical Value	Field Description
<63:0>	BUSADDR	RO/H	0x0	0x0	Bus address. Contains the first bus address that encountered an SoC bus error. It is valid when the USBDRD(0..1)_UAHC_GSTS[BUSERRADDRVLD] = 1. It can only be cleared by resetting the core.

UAHC SuperSpeed Port-to-Bus Instance Mapping Register
USBDRD(0..1)_UAHC_GPRTBIMAP

This register specifies the SuperSpeed USB instance number to which each USB 3.0 port is connected. By default, USB 3.0 ports are evenly distributed among all SuperSpeed USB instances. Software can program this register to specify how USB 3.0 ports are connected to SuperSpeed USB instances. The UAHC only implements one SuperSpeed bus-instance, so this register should always be 0.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved.
<3:0>	BINUM1	R/W	0x0	0x0	SuperSpeed USB instance number for port 1.

Cavium Confidential For Bowen Yang ABU 08/27/2014

UAHC Hardware Parameters Register 0
USBDRD(0..1)_UAHC_GHWPARAMS0

This register contains the hardware configuration options selected at compile-time.

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	AWIDTH	RO	0x40	0x40	USB core bus-address width.
<23:16>	SDWIDTH	RO	0x20	0x20	USB core bus slave-data width.
<15:8>	MDWIDTH	RO	0x40	0x40	USB core bus master-data width.
<7:6>	SBUS_TYPE	RO	0x1	0x1	USB core bus slave type: AXI.
<5:3>	MBUS_TYPE	RO	0x1	0x1	USB core bus master type: AXI.
<2:0>	MODE	RO	0x2	0x2	Operation mode: 0x2: dual-role device.

UAHC Hardware Parameters Register 1
USBDRD(0..1)_UAHC_GHWPARAMS1

This register contains the hardware configuration options selected at compile-time.

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	EN_DBC	RO	—	—	Enable debug capability.
<30>	RM_OPT_FEATURES	RO	0	0	Remove optional features.
<29>	SYNC_RST	RO	0	0	Synchronous reset coding.
<28>	RAM_BUS_CLKS_SYNC	RO	0	0	RAM_CLK and BUS_CLK are synchronous.
<27>	MAC_RAM_CLKS_SYNC	RO	0	0	MAC3_CLK and RAM_CLK are synchronous.
<26>	MAC_PHY_CLKS_SYNC	RO	0	0	MAC3_CLK and PHY_CLK are synchronous.
<25:24>	EN_PWROPT	RO	0x1	0x1	Power optimization mode: bit<0> = Clock-gating feature available bit<1> = Hibernation feature available
<23>	SPRAM_TYP	RO	1	1	SRAM type: one-port RAMs.
<22:21>	NUM_RAMS	RO	0x3	0x3	Number of RAMs.
<20:15>	DEVICE_NUM_INT	RO	0x1	0x1	Number of event buffers (and interrupts) in device-mode.
<14:12>	ASPACEWIDTH	RO	0x4	0x4	Native interface address-space port width.
<11:9>	REQINFOWIDTH	RO	0x4	0x4	Native interface request/response-info port width.
<8:6>	DATAINFOWIDTH	RO	0x4	0x4	Native interface data-info port width.
<5:3>	BURSTWIDTH_M1	RO	0x7	0x7	[Width – 1] of AXI length field.
<2:0>	IDWIDTH_M1	RO	0x3	0x3	[Width – 1] of AXI ID field.

Cavium Confidential For Bowen Yang ALU 02/27/2014

UAHC Core GHW Parameters Register 2
USBDRD(0..1)_UAHC_GHWPARAMS2

This register contains the hardware configuration options selected at compile-time.

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	USERID	RO	0xDECAFBAD	0xDECAFBAD	User ID.

UAHC GHW Parameters Register 3
USBDRD(0..1)_UAHC_GHWPARAMS3

This register contains the hardware configuration options selected at compile-time.

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	—	RAZ	—	—	Reserved.
<30:23>	CACHE_TOTAL_ XFER_RESOURCES	RO	0x8	0x8	Maximum number of transfer resources in the core.
<22:18>	NUM_IN_EPS	RO	0x4	0x4	Maximum number of device-mode IN endpoints active.
<17:12>	NUM_EPS	RO	0x8	0x8	Number of device-mode single-directional endpoints.
<11>	ULPI_CARKIT	RO	0	0	ULPI Carkit is not supported.
<10>	VENDOR_CTL_ INTERFACE	RO	0	0	UTMI+ PHY vendor control interface enabled.
<9:8>	—	RAZ	—	—	Reserved.
<7:6>	HSPHY_DWIDTH	RO	0x2	0x2	Data width of the UTMI+ PHY interface: 0x2 = 8-or-16 bits.
<5:4>	FSPHY_INTERFACE	RO	0x0	0x0	USB 1.1 full-speed serial transceiver interface.
<3:2>	HSPHY_INTERFACE	RO	0x1	0x1	High-speed PHY interface: 0x1 = UTMI+.
<1:0>	SSPHY_INTERFACE	RO	0x1	0x1	SuperSpeed PHY interface: 0x1 = PIPE3.

Cavium Confidential For Bowen Yang ALU 02/27/2014

UAHC GHW Parameters Register 4
USBDRD(0..1)_UAHC_GHWPARAMS4

This register contains the hardware configuration options selected at compile-time.

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:28>	BMU_LSP_DEPTH	RO	0x4	0x4	Depth of the BMU-LSP status buffer.
<27:24>	BMU_PTL_DEPTH_M1	RO	0x7	0x7	Depth of the BMU-PTL source/sink buffers minus 1.
<23>	EN_ISOC_SUPT	RO	1	1	Isochronous support enabled.
<22>	—	RAZ	—	—	Reserved.
<21>	EXT_BUFF_CONTROL	RO	0	0	Enables device external buffer control sideband controls.
<20:17>	NUM_SS_USB_ INSTANCES	RO	0x1	0x1	Number of SuperSpeed bus instances.
<16:13>	HIBER_SCRATCHBUFS	RO	0x1	0x1	Number of hibernation scratchpad buffers.
<12:6>	—	RAZ	—	—	Reserved.
<5:0>	CACHE_TRBS_ PER_TRANSFER	RO	0x4	0x4	Number of TRBs per transfer that can be cached.

UAHC GHW Parameters Register 5
USBDRD(0..1)_UAHC_GHWPARAMS5

This register contains the hardware configuration options selected at compile-time.

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:28>	—	RAZ	—	—	Reserved.
<27:22>	DFQ_FIFO_DEPTH	RO	0x10	0x10	Size of the BMU descriptor fetch-request queue.
<21:16>	DWQ_FIFO_DEPTH	RO	0x20	0x20	Size of the BMU descriptor write queue.
<15:10>	TXQ_FIFO_DEPTH	RO	0x8	0x8	Size of the BMU Tx request queue.
<9:4>	RXQ_FIFO_DEPTH	RO	0x8	0x8	Size of the BMU Rx request queue.
<3:0>	BMU_BUSGM_DEPTH	RO	0x8	0x8	Depth of the BMU-BUSGM source/sink buffers.

Cavium Confidential For Bowen Yang ARD 02/27/2014

UAHC GHW Parameters Register 6
USBDRD(0..1)_UAHC_GHWPARAMS6

This register contains the hardware configuration options selected at compile-time. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	RAM0_DEPTH	RO	0x740	0x740	RAM0 Depth.
<15>	EN_BUS_FILTERS	RO	1	1	Enable VBus filters support.
<14>	EN_BC	RO	0	0	Enable battery-charging support.
<13>	EN_OTG_SS	RO	0	0	Enable OTG SuperSpeed support.
<12>	EN_ADP	RO	0	0	Enable ADP support.
<11>	HNP_SUPPORT	RO	0	0	HNP support.
<10>	SRP_SUPPORT	RO	0	0	SRP support.
<9:8>	—	RAZ	—	—	Reserved.
<7>	EN_FPGA	RO	0	0	Enable FPGA implementation.
<6>	EN_DBG_PORTS	RO	0	0	Enable Debug ports for FGPA.
<5:0>	PSQ_FIFO_DEPTH	RO	0x20	0x20	Size of the BMU-protocol status queue.

UAHC GHW Parameters Register 7
USBDRD(0..1)_UAHC_GHWPARAMS7

This register contains the hardware configuration options selected at compile-time. See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	RAM2_DEPTH	RO	0x308	0x308	RAM2 depth.
<15:0>	RAM1_DEPTH	RO	0x4CE	0x4CE	RAM1 depth.

Cavium Confidential For Bowen Yang ALU 02/27/2014

UAHC Debug FIFO Space Available Register USBDRD(0..1)_UAHC_GDBGFIFOSPACE

This register is for debug purposes. It provides debug information on the internal status and state machines. Global debug registers have design-specific information, and are used for debugging purposes. These registers are not intended to be used by the customer. If any debug assistance is needed for the silicon, contact customer support with a dump of these registers.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	FIELD NAME	Access	Reset Value	Typical Value	Field Description
<31:16>	SPACEAVAILABLE	RO/H	0x82	—	Space available in the selected FIFO.
<15:8>	—	RAZ	—	—	Reserved.
<7:0>	SELECT	R/W	0x0	0x0	FIFO/queue select/port-select. **FIFO/queue select**: <7:5> indicates the FIFO/queue type; <4:0> indicates the FIFO/queue number. For example, 0x21 refers to RxFIFO_1, and 0x5E refers to TxReqQ_30. 0x1F–0x0: TxFIFO_31 to TxFIFO_0 0x3F–0x20: RxFIFO_31 to RxFIFO_0 0x5F–0x40: TxReqQ_31 to TxReqQ_0 0x7F–0x60: RxReqQ_31 to RxReqQ_0 0x9F–0x80: RxInfoQ_31 to RxInfoQ_0 0xA0: DescFetchQ 0xA1: EventQ 0xA2: ProtocolStatusQ **Port-select**: <3:0> selects the port-number when accessing USBDRD(0..1)_UAHC_GDBGLTSSM.

Cavium Confidential For Bowen Yang ALU 02/27/2014

UAHC LTSSM Debug Register
USBDRD(0..1)_UAHC_GDBGLTSSM

In multiport host configuration, the port number is defined by USBDRD(0..1)_UAHC_GDBGFIFOSPACE[SELECT]<3:0>. Value of this register may change immediately after reset. See description in USBDRD(0..1)_UAHC_GDBGFIFOSPACE.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	FIELD NAME	Access	Reset Value	Typical Value	Field Description
<31:27>	—	RAZ	—	—	Reserved.
<26>	LTDBTIMEOUT	RO/H	0x0	—	LTDB timeout.
<25:22>	LTDBLINKSTATE	RO/H	0x4	—	LTDB link state.
<21:18>	LTDBSUBSTATE	RO/H	0x0	—	LTDB substate.
<17:0>	DEBUGPIPESTATUS	RO/H	0x10440	—	Debug PIPE status. <17> Elastic buffer mode <16> TX elec idle <15> Rx polarity <14> TX Detect Rx/loopback <13:11> LTSSM PHY command state 0x0 = PHY_IDLE (PHY command state is in IDLE. No PHY request is pending.) 0x1 = PHY_DET (Request to start receiver detection) 0x2 = PHY_DET_3 (Wait for Phy_Status (receiver detection)) 0x3 = PHY_PWR_DLY (delay Pipe3_PowerDown P0 → P1/P2/P3 request) 0x4 = PHY_PWR_A (delay for internal logic) 0x5 = PHY_PWR_B (wait for Phy_Status(Power-state change request)) <10:9> Power down <8> RxEq train <7:6> TX de-emphasis <5:3> LTSSM clock state 0x0 = CLK_NORM (PHY is in non-P3 state and PCLK is running) 0x1 = CLK_TO_P3 (P3 entry request to PHY) 0x2 = CLK_WAIT1 (wait for Phy_Status (P3 request)) 0x3 = CLK_P3 (PHY is in P3 and PCLK is not running) 0x4 = CLK_TO_P0 (P3 exit request to PHY) 0x5 = CLK_WAIT2 (Wait for Phy_Status (P3 exit request)) <2> TX swing <1> Rx termination <0> TX 1s/0s

Cavium Confidential For Bowen Wang Aug 6 2014

UAHC LNMCC Debug Register
USBDRD(0..1)_UAHC_GDBGLNMCC

See description in USBDRD(0..1)_UAHC_GDBGFIFOSPACE.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:9>	—	RAZ	—	—	Reserved.
<8:0>	LNMCC_BERC	RO/H	—	—	This field indicates the bit-error-rate information for the port selected in USBDRD(0..1)_UAHC_GDBGFIFOSPACE[SELECT] (port-select). This field is for debug purposes only.

UAHC BMU Debug Register
USBDRD(0..1)_UAHC_GDBGBMU

See description in USBDRD(0..1)_UAHC_GDBGFIFOSPACE.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:8>	BMU_BCU_DBG	RO/H	0x0	—	BMU_BCU debug information.
<7:4>	BMU_DCU_DBG	RO/H	0x0	—	BMU_DCU debug information.
<3:0>	BMU_CCU_DBG	RO/H	0x0	—	BMU_CCU debug information.

Cavium Confidential For Bowen Yang ALD 02/27/2014

UAHC LSP Multiplexer Debug Register USBDRD(0..1)_UAHC_GDBGLSPMUX

See description in USBDRD(0..1)_UAHC_GDBGFIFOSPACE.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	FIELD NAME	Access	Reset Value	Typical Value	Field Description
<31:24>	RESERVED_0	R/W	0x0	0x0	Reserved. Read must ignore.
<23:16>	LATRACEPORTMUXSELECT	R/W	0x0	0x0	logic_analyzer_trace port multiplexer select. Only bits<21:16> are used. For details on how the mux controls the debug traces, refer to the Verilog file. A value of 0x3F drives 0s on the logic_analyzer_trace signal. If you plan to OR (instead using a mux) this signal with other trace signals in your system to generate a common trace signal, you can use this feature.
<15>	ENDBC	R/W	0	0	Enable debugging of the debug capability LSP. Use HOSTSELECT to select the DbC LSP debug information presented in the GDBGLSP register.
<14>	RESERVED_1	R/W	0	0	Reserved. Read must ignore.
<13:0>	HOSTSELECT	R/W	0x0	0x0	Host select. Selects the LSP debug information presented in USBDRD(0..1)_UAHC(0)_GDBGLSP.

UAHC LSP Debug Register USBDRD(0..1)_UAHC_GDBGLSP

See description in USBDRD(0..1)_UAHC_GDBGFIFOSPACE.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	LSP_DBG	RO/H	0x0	—	LSP debug information.

Cavium Confidential
Bowen Yang
ALU
02/27/2014

UAHC Endpoint Information Debug Register USBDRD(0..1)_UAHC_GDBGEPINFO

See description in USBDRD(0..1)_UAHC_GDBGFIFOSPACE.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	ENDPT_DBG	RO/H	0x0	—	Endpoint debug information.

UAHC High-Speed Port-to-Bus Instance Mapping Register USBDRD(0..1)_UAHC_GPRTBIMAP_HS

This register specifies the high-speed USB instance number to which each USB 2.0 port is connected. By default, USB 2.0 ports are evenly distributed among all high-speed USB instances. Software can program this register to specify how USB 2.0 ports are connected to high-speed USB instances. The UAHC only implements one high-speed bus-instance, so this register should always be 0.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved.
<3:0>	BINUM1	R/W	0x0	0x0	High-speed USB instance number for port 1.

UAHC Full/LowSpeed Port-to-Bus Instance Mapping Register USBDRD(0..1)_UAHC_GPRTBIMAP_FS

This register specifies the full-speed/low-speed USB instance number to which each USB 1.1 port is connected. By default, USB 1.1 ports are evenly distributed among all full-speed/low-speed USB instances. Software can program this register to specify how USB 1.1 ports are connected to full-speed/low-speed USB instances. The UAHC only implements one full-speed/low-speed bus-instance, so this register should always be 0x0.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:4>	—	RAZ	—	—	Reserved.
<3:0>	BINUM1	R/W	0x0	0x0	Full-speed USB instance number for port 1.

Cavium Confidential For Bowen Yang ALU 6/3/2014

UAHC USB2 PHY-Configuration Register USBDRD(0..1)_UAHC_GUSB2PHYCFG(0)

This register is used to configure the core after power-on. It contains USB 2.0 and USB 2.0 PHY-related configuration parameters. The application must program this register before starting any transactions on either the SoC bus or the USB. Per-port registers are implemented.

NOTE: Do not change this register after the initial programming.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	PHYSOFTRST	R/W	0	0	PHY soft reset. Causes the usb2phy_reset signal to be asserted to reset a UTMI PHY.
<30>	U2_FREECLK_EXISTS	R/W	1	1	Specifies whether your USB 2.0 PHY provides a free-running PHY clock, which is active when the clock control input is active. If your USB 2.0 PHY provides a free-running PHY clock, it must be connected to the utmi_clk[0] input. The remaining utmi_clk[n] must be connected to the respective port clocks. The core uses the Port-0 clock for generating the internal mac2 clock. 0 = USB 2.0 free clock does not exist 1 = USB 2.0 free clock exists Note: This field must be set to zero if you enable ITP generation based on the REF_CLK counter, USBDRD(0..1)_UAHC_GCTL[SOFITPSYNC] = 1, or USBDRD(0..1)_UAHC_GFLADJ [GFLADJ_REFCLK_LPM_SEL] = 1.
<29>	ULPI_LPM_WITH_ OPMODE_CHK	R/W	0	0	Support the LPM over ULPI without NOPID token to the ULPI PHY. Always 0x0.
<28:25>	—	RAZ	—	—	Reserved.
<24:19>	RESERVED_1	R/W	0x0	0x0	Reserved. Read must ignore.
<18>	ULPIEXTVBUSINDICATOR	R/W	0	0	Reserved (unused in this configuration).
<17>	ULPIEXTVBUSDRV	R/W	0	0	Reserved (unused in this configuration).
<16>	ULPICLKSUSM	R/W	0	0	Reserved (unused in this configuration).
<15>	ULPIAUTORES	R/W	0	0	Reserved (unused in this configuration).
<14>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang 01/27/2016

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<13:10>	USBTRDTIM	R/W	0x9	0x9	USB 2.0 turnaround time. Sets the turnaround time in PHY clock cycles. Specifies the response time for a MAC request to the packet FIFO controller (PFC) to fetch data from the DFIFO (SPRAM). The following are the required values for the minimum SoC bus frequency of 60 MHz. USB turnaround time is a critical certification criteria when using long cables and five hub levels. When the MAC interface is 8-bit UTMI+/ULPI, the required values for this field is 0x9. If SoC bus clock is less than 60 MHz, and USB turnaround time is not critical, this field can be set to a larger value.
<9>	RESERVED_0	R/W	0	0	Reserved. Read must ignore.
<8>	ENBLSLPM	R/W	1	1	Enable utmi_sleep_n and utmi_l1_suspend_n. The application uses this field to control utmi_sleep_n and utmi_l1_suspend_n assertion to the PHY in the L1 state. 1 = utmi_sleep_n and utmi_l1_suspend_n assertion from the core is transferred to the external PHY. 0 = utmi_sleep_n and utmi_l1_suspend_n assertion from the core is not transferred to the external PHY. **NOTE:** When hardware LPM is enabled, this bit should be set high for Port0.
<7>	PHYSEL	WO	0	0	USB 2.0 high-speed PHY or USB 1.1 full-speed serial transceiver select.
<6>	SUSPHY	R/W	1	1	Suspend USB2.0 high-speed/full-speed/low-speed PHY. When set, USB2.0 PHY enters suspend mode if suspend conditions are valid. Application needs to set this bit to 1 after the core initialization is completed.
<5>	FSINTF	RO	0	0	Full-speed serial-interface select. Always reads as 0x0.
<4>	ULPI_UTMI_SEL	RO	0	0	ULPI or UTMI+ select. Always reads as 0x0, indicating UTMI+.
<3>	PHYIF	R/W	0	0	PHY interface width: 1 = 16-bit, 0 = 8-bit. All the enabled 2.0 ports should have the same clock frequency as Port0 clock frequency (utmi_clk[0]). The UTMI 8-bit and 16-bit modes cannot be used together for different ports at the same time (i.e., all the ports should be in 8-bit mode, or all of them should be in 16-bit mode).

Cavium Confidential For Bowen Yang 02/17/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<2:0>	TOUTCAL	R/W	0x0	0x0	High-speed/full-speed timeout calibration. The number of PHY clock cycles, as indicated by the application in this field, is multiplied by a bit-time factor; this factor is added to the high-speed/full-speed interpacket timeout duration in the core to account for additional delays introduced by the PHY. This might be required, since the delay introduced by the PHY in generating the linestate condition can vary among PHYs. The USB standard timeout value for high-speed operation is 736 to 816 (inclusive) bit times. The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of connection. The number of bit times added per PHY clock are: • High-speed operation: – one 30-MHz PHY clock = 16 bit times – one 60-MHz PHY clock = 8 bit times • Full-speed operation: – one 30-MHz PHY clock = 0.4 bit times – one 60-MHz PHY clock = 0.2 bit times – one 48-MHz PHY clock = 0.25 bit times

UAHC USB2 I2C Control Register USBDRD(0..1)_UAHC_GUSB2I2CCTL

This register is reserved for future use.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	—	RAZ	—	—	Reserved.

Cavium Confidential For Bowen Yang ALU 02/27/2014

UAHC USB3 Pipe-Control Register USBDRD(0..1)_UAHC_GUSB3PIPECTL(0)

This register is used to configure the core after power-on. It contains USB 3.0 and USB 3.0 PHY-related configuration parameters. The application must program this register before starting any transactions on either the SoC bus or the USB. Per-port registers are implemented.

NOTE: Do not change this register after the initial programming.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	FIELD NAME	Access	Reset Value	Typical Value	Field Description
<31>	PHYSOFTRST	R/W	0	0	USB3 PHY soft reset (PHYSoftRst). After setting this bit to 1, the software needs to clear this bit.
<30>	HSTPRTCMPL	R/W	0	0	Host port compliance. Setting this bit to 1 enables placing the SuperSpeed port link into a compliance state, which allows testing of the PIPE PHY compliance patterns without having to have a test fixture on the USB 3.0 cable. By default, this bit should be set to 0. In compliance-lab testing, the SuperSpeed port link enters compliance after failing the first polling sequence after power on. Set this bit to 0 when you run compliance tests. The sequence for using this functionality is as follows: **1.** Disconnect any plugged-in devices. **2.** Set USBDRD(0..1)_UAHC_USBCMD[HCRST] = 1 or power-on-chip reset. **3.** Set USBDRD(0..1)_UAHC_PORTSC(0..1)[PP] = 0. **4.** Set HSTPRTCMPL = 1. This places the link into compliance state. To advance the compliance pattern, follow this sequence (toggle HSTPRTCMPL): **1.** Set HSTPRTCMPL = 0. **2.** Set HSTPRTCMPL = 1. This advances the link to the next compliance pattern. To exit from the compliance state, set USBDRD(0..1)_UAHC_USBCMD[HCRST] = 1 or power-on-chip reset.
<29>	U2SSINACTP3OK	R/W	0	0	P3 OK for U2/SS.Inactive: 1 = during link state U2/SS.Inactive, put PHY in P3 0 = during link state U2/SS.Inactive, put PHY in P2 (default)
<28>	DISRXDETP3	R/W	0	0	Disables receiver detection in P3. If PHY is in P3 and the core needs to perform receiver detection: 1 = core changes the PHY power state to P2 and then performs receiver detection. After receiver detection, core changes PHY power state to P3. 0 = core performs receiver detection in P3 (default)

Cavium Confidential For Bowen Yang 08/11/2014

Bit Pos	FIELD NAME	Access	Reset Value	Typical Value	Field Description
<27>	UX_EXIT_IN_PX	R/W	0	0	UX exit in Px: 1 = the core does U1/U2/U3 exit in PHY power state P1/P2/P3 respectively 0 = the core does U1/U2/U3 exit in PHY power state P0 (default behavior) This bit is added for SuperSpeed PHY workaround where SuperSpeed PHY injects a glitch on pipe3_RxElecIdle while receiving Ux exit LFPS, and pipe3_PowerDown change is in progress.
<26>	PING_ENCHANCE_EN	R/W	0	0	Ping enhancement enable. When set to 1, the downstream-port U1-ping-receive timeout becomes 500 ms instead of 300 ms. Minimum Ping.LFPS receive duration is 8 ns (one mac3_clk cycle). This field is valid for the downstream port only.
<25>	U1U2EXITFAIL_ TO_RECOV	R/W	0	0	U1U2exit fail to recovery. When set to 1, and U1/U2 LFPS handshake fails, the LTSSM transitions from U1/U2 to recovery instead of SS.inactive. If recovery fails, then the LTSSM can enter SS.Inactive. This is an enhancement only. It prevents interoperability issue if the remote link does not do the proper handshake.
<24>	REQUEST_P1P2P3	R/W	1	1	Always request P1/P2/P3 for U1/U2/U3. 1 = the core always requests PHY power change from P0 to P1/P2/P3 during U0 to U1/U2/U3 transition. 0 = if immediate Ux exit (remotely initiated, or locally initiated) happens, the core does not request P1/P2/P3 power state change.
<23>	STARTRXDETU3 RXDET	WO	0	0	Start receiver detection in U3/Rx.Detect. If DISRXDETU3RXDET is set to 1 during reset, and the link is in U3 or Rx.Detect state, the core starts receiver detection on rising edge of this bit. This bit is valid for downstream ports only, and this feature must not be enabled for normal operation.
<22>	DISRXDETU3RXDET	R/W	0	0	Disable receiver detection in U3/Rx.Detect. When set to 1, the core does not do receiver detection in U3 or Rx.Detect state. If STARTRXDETU3RXDET is set to 1 during reset, receiver detection starts manually. This bit is valid for downstream ports only, and this feature must not be enabled for normal operation.
<21:19>	DELAYPX	R/W	0x1	0x1	Delay P1P2P3. Delay P0 to P1/P2/P3 request when entering U1/U2/U3 until (DELAYPX × 8) 8B10B error occurs, or Pipe3_RxValid drops to 0. DELAYPXTRANSENTERUX must reset to 1 to enable this functionality.
<18>	DELAYPXTRANS ENTERUX	R/W	1	1	Delay PHY power change from P0 to P1/P2/P3 when link state changing from U0 to U1/U2/U3 respectively. 1 = when entering U1/U2/U3, delay the transition to P1/P2/P3 until the pipe3 signals, Pipe3_RxElecIlde is 1 and pipe3_RxValid is 0 0 = when entering U1/U2/U3, transition to P1/P2/P3 without checking for Pipe3_RxElecIlde and pipe3_RxValid.

Cavium Confidential Draft
Bowen Zang (2/7/2014)

Bit Pos	FIELD NAME	Access	Reset Value	Typical Value	Field Description
<17>	SUSPEND_EN	R/W	1	1	Suspend USB3.0 SuperSpeed PHY (Suspend_en). When set to 1, and if suspend conditions are valid, the USB 3.0 PHY enters suspend mode. Application needs to set this bit to 1 after the core initialization is completed.
<16:15>	DATWIDTH	RO	0x0	0x0	PIPE data width. 0x0 = 32 bits 0x1 = 16 bits 0x2 = 8 bits 0x3 = reserved One clock cycle after reset, these bits receive the value seen on the pipe3_DataBusWidth. This will always be 0x0.
<14>	ABORTRXDETINU2	R/W	0	0	Abort RX Detect in U2. When set to 1, and the link state is U2, the core aborts receiver detection if it receives U2 exit LFPS from the remote link partner. This bit is for downstream port only.
<13>	SKIPRXDET	R/W	0	0	Skip RX detect. When set to 1, the core skips RX detection if pipe3_RxElecIdle is low. Skip is defined as waiting for the appropriate timeout, then repeating the operation.
<12>	LFPSP0ALGN	R/W	0	0	LFPS P0 align. When set to 1: • the core deasserts LFPS transmission on the clock edge that it requests Phy power state 0 when exiting U1, U2, or U3 low power states. Otherwise, LFPS transmission is asserted one clock earlier. • the core requests symbol transmission two pipe3_rx_pclks periods after the PHY asserts PhyStatus as a result of the PHY switching from P1 or P2 state to P0 state. For USB 3.0 host, this is not required.
<11>	P3P2TRANOK	R/W	0	0	P3 P2 transitions OK. 1 = the core transitions directly from Phy power state P2 to P3 or from state P3 to P2. 0 = P0 is always entered as an intermediate state during transitions between P2 and P3, as defined in the PIPE3 specification. According to PIPE3 specification, any direct transition between P3 and P2 is illegal.
<10>	P3EXSIGP2	R/W	0	0	P3 exit signal in P2. When set to 1, the core always changes the PHY power state to P2, before attempting a U3 exit handshake.
<9>	LFPSFILT	R/W	0	0	LFPS filter. When set to 1, filter LFPS reception with pipe3_RxValid in PHY power state P0, ignore LFPS reception from the PHY unless both pipe3_Rxelecidle and pipe3_RxValid are deasserted.
<8>	RXDET2POLLLFPSCTRL	R/W	0	0	RX_DETECT to Polling. 0 = Enables a 400 µs delay to start polling LFPS after RX_DETECT. This allows VCM offset to settle to a proper level. 1 = Disables the 400 µs delay to start polling LFPS after RX_DETECT.
<7>	RESERVED_0	R/W	0	0	Reserved. Read must ignore.

Cavium Confidential For Bowen Wang AU 5/12/2014

Bit Pos	FIELD NAME	Access	Reset Value	Typical Value	Field Description
<6>	TXSWING	R/W	0	0	TX swing. Refer to the PIPE3 specification.
<5:3>	TXMARGIN	R/W	0x0	0x0	TX margin. Refer to the PIPE3 specification, table 5-3.
<2:1>	TXDEEMPHASIS	R/W	0x1	0x1	TX de-emphasis. The value driven to the PHY is controlled by the LTSSM during USB3 compliance mode. Refer to the PIPE3 specification, table 5-3.
<0>	ELASTICBUFFER MODE	R/W	0	0	Elastic buffer mode. Refer to the PIPE3 specification, table 5-3.

UAHC TX FIFO Size Registers USBDRD(0..1)_UAHC_GTXFIFOSIZ(0..3)

This register holds the internal RAM start address/depth of each TxFIFO implemented. Unless packet size/buffer size for each endpoint is different and application-specific, it is recommended that the software use the default value. One register per FIFO. One register per FIFO.

Reset values = 0:{0x0000_0082} 1:{0x0082_0103} 2:{0x0185_0205}

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	TXFSTADDR	R/W	—	—	Transmit FIFO*n* RAM start address. Contains the memory start address for TxFIFO*n*. Reset values: Device mode: TXFIFO0: 0x0 TXFIFO1: 0x42 TXFIFO2: 0x1C6 TXFIFO3: 0x34A Host mode: TXFIFO0: 0x0 TXFIFO1: 0x82 TXFIFO2: 0x185 TXFIFO3: 0x38A
<15:0>	TXFDEP	R/W	—	—	TxFIFO*n* depth. This value is in terms of TX RAM data width. minimum value = 0x20, maximum value = 0x8000. Reset values: Device mode: TXFIFO0: 0x42 TXFIFO1: 0x184 TXFIFO2: 0x184 TXFIFO3: 0x184 Host mode: TXFIFO0 (FSLS): 0x82 TXFIFO1 (HS): 0x103 TXFIFO2 (SS): 0x205

Cavium Confidential For Bowen Yang ALU 02/27/2014

UAHC RX FIFO Size Register USBDRD(0..1)_UAHC_GRXFIFOSIZ(0..2)

The application can program the internal RAM start address/depth of the each RxFIFO as shown below. It is recommended that software use the default value. In Host mode, per-port registers are implemented. One register per FIFO.

Reset values = 0:{0x0000_0084} 1:{0x0084_0104} 2:{0x0188_0180}

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	RXFSTADDR	R/W	—	—	RxFIFO*n* RAM start address. This field contains the memory start address for RxFIFO*n*. Reset values: Device mode: RXFIFO0: 0x0 RXFIFO1: 0x185 RXFIFO2: 0x185 Host mode: RXFIFO0: 0x0 RXFIFO1: 0x84 RXFIFO2: 0x188
<15:0>	RXFDEP	R/W	—	—	RxFIFO*n* depth. This value is in terms of RX RAM Data width. Minimum value = 0x20, maximum value = 0x4000. Reset values: Device mode: RXFIFO0: 0x185 RXFIFO1: 0x0 RXFIFO2: 0x0 Host mode: RXFIFO0: 0x84 RXFIFO1: 0x104 RXFIFO2: 0x180

Cavium Confidential For Bowen Yang ALU 02/27/2014

Global Event-Buffer Address Register USBDRD(0..1)_UAHC_GEVNTADR(0)

This register holds the event-buffer DMA address pointer. Software must initialize this address once during power-on initialization. Software must not change the value of this register after it is initialized. Software must only use the GEVNTCOUNT(0) register for event processing. The lower n bits of the address must be USBDRD(0..1)_UAHC_GEVNTSIZ_(0)[EVNTSIZ]-aligned.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1
- USBDRD(0..1)_UAHC_DCTL[CSFTRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	EVNTADR	R/W/H	0x0	0x0	Holds the start address of the external memory for the event buffer. During operation, hardware does not update this address.

Global Event-Buffer Size Register USBDRD(0..1)_UAHC_GEVNTSIZ(0)

This register holds the event-buffer size and the event-interrupt mask bit. During power-on initialization, software must initialize the size with the number of bytes allocated for the event buffer. The event-interrupt mask masks the interrupt, but events are still queued. After configuration, software must preserve the event-buffer size value when changing the event-interrupt mask.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1
- USBDRD(0..1)_UAHC_DCTL[CSFTRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	EVNTINTMASK	R/W	0	0	Event interrupt mask. When set to 1, this prevents the interrupt from being generated. However, even when the mask is set, the events are queued.
<30:16>	—	RAZ	—	—	Reserved
<15:0>	EVNTSIZ	R/W	0x0	0x0	Event-buffer size. Indicates the size of the event buffer in bytes; must be a multiple of four. This is programmed by software once during initialization. The minimum size of the event buffer is 32 bytes.

Cavium Confidential For Bowen Yang ACU 02/12/2014

Global Event-Buffer Count Register USBDRD(0..1)_UAHC_GEVNTCOUNT(0)

This register holds the number of valid bytes in the event buffer. During initialization, software must initialize the count by writing 0 to the EVNTCOUNT field. Each time the hardware writes a new event to the event buffer, it increments this count. Most events are four bytes, but some events may span over multiple four-byte entries. Whenever the count is greater than 0x0, the hardware raises the corresponding interrupt line (depending on the USBDRD(0..1)_UAHC_GEVNTSIZ(0)[EVNTINTMASK]). On an interrupt, software processes one or more events out of the event buffer. Afterwards, software must write EVNTCOUNT with the number of bytes it processed.

Clock-crossing delays might result in the interrupt's continual assertion after software acknowledges the last event. Therefore, when the interrupt line is asserted, software must read this register and only process events if EVNTCOUNT is greater than 0x0.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1
- USBDRD(0..1)_UAHC_DCTL[CSFTRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	—	RAZ	—	—	Reserved.
<15:0>	EVNTCOUNT	R/W/H	0x0	0x0	When read, returns the number of valid events in the Event Buffer (in bytes). When written, hardware decrements the count by the value written. The interrupt line remains high when count is not 0x0.

UAHC GHW Parameters Register 8 USBDRD(0..1)_UAHC_GHWPARAMS8

This register contains the hardware configuration options selected at compile-time.

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	DCACHE_DEPTH_INFO	RO	0x708	0x—	Dcache depth.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Global Device TX FIFOs Priority Register USBDRD(0..1)_UAHC_GTXFIFOPRIDEV

This register specifies the relative DMA priority level among the device TXFIFOs (one per IN endpoint). Each register bit[*n*] controls the priority (1 = high, 0 = low) of each TXFIFO[*n*]. When multiple TXFIFOs compete for DMA service at a given time (i.e., multiple TXQs contain TX DMA requests and their corresponding TXFIFOs have space available), the TX DMA arbiter grants access on a packet-basis in the following manner:

1. High-priority TXFIFOs are granted access using round-robin arbitration.
2. Low-priority TXFIFOs are granted access using round-robin arbitration only after the high-priority TXFIFOs have no further processing to do (i.e., either the TXQs are empty or the corresponding TXFIFOs are full).

For scatter-gather packets, the arbiter grants successive DMA requests to the same FIFO until the entire packet is completed. When configuring periodic IN endpoints, software must set register bit[*n*] = 1, where *n* is the TXFIFO assignment. This ensures that the DMA for isochronous or interrupt IN endpoints are prioritized over bulk or control IN endpoints.

This register is present only when the core is configured to operate in the device mode (includes DRD and OTG modes). The register size corresponds to the number of device IN endpoints.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:4>	—	RAZ	—	—	Reserved.
<3:0>	TX_PRIORITY	R/W	0x0	0x0	Each register bit[*n*] controls the priority (1 = high, 0 = low) of TXFIFO[*n*] within a speed group.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Global Host TX FIFOs DMA Priority Register USBDRD(0..1)_UAHC_GTXFIFOPRIHST

This register specifies the relative DMA priority level among the host TXFIFOs (one per USB bus instance) within the associated speed group (SuperSpeed or high-speed/full-speed/low-speed). When multiple TXFIFOs compete for DMA service at a given time, the TXDMA arbiter grants access on a packet-basis in the following manner:

1. Among the FIFOs in the same speed group (SuperSpeed or high-speed/full-speed/low-speed):
   a. High-priority TXFIFOs are granted access using round-robin arbitration.
   b. Low-priority TXFIFOs are granted access using round-robin arbitration only after high-priority TXFIFOs have no further processing to do (i.e., either the TXQs are empty or the corresponding TXFIFOs do not have the required data).
2. The TX DMA arbiter prioritizes the SuperSpeed group or high-speed/full-speed/low-speed group according to the ratio programmed in USBDRD(0..1)_UAHC_GDMAHLRATIO.

For scatter-gather packets, the arbiter grants successive DMA requests to the same FIFO until the entire packet is completed. The register size corresponds to the number of configured USB bus instances; for example, in the default configuration, there are 3 USB bus instances (1 SuperSpeed, 1 high-speed, and 1 full-speed/low-speed).

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:3>	—	RAZ	—	—	Reserved.
<2:0>	TX_PRIORITY	R/W	0x0	0x0	Each register bit *n* controls the priority (1: high, 0: low) of TX FIFO<*n*> within a speed group.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Global Host RX FIFOs DMA Priority Register USBDRD(0..1)_UAHC_GRXFIFOPRIHST

This register specifies the relative DMA priority level among the host RXFIFOs (one per USB bus instance) within the associated speed group (SuperSpeed or high-speed/full-speed/low-speed). When multiple RXFIFOs compete for DMA service at a given time, the RXDMA arbiter grants access on a packet-basis in the following manner:

1. Among the FIFOs in the same speed group (SuperSpeed or high-speed/full-speed/low-speed):
   a. High-priority RXFIFOs are granted access using round-robin arbitration.
   b. Low-priority RXFIFOs are granted access using round-robin arbitration only after high-priority RXFIFOs have no further processing to do (i.e., either the RXQs are empty or the corresponding RXFIFOs do not have the required data).
2. The RX DMA arbiter prioritizes the SuperSpeed group or high-speed/full-speed/low-speed group according to the ratio programmed in USBDRD(0..1)_UAHC_GDMAHLRATIO.

For scatter-gather packets, the arbiter grants successive DMA requests to the same FIFO until the entire packet is completed. The register size corresponds to the number of configured USB bus instances; for example, in the default configuration, there are 3 USB bus instances (1 SuperSpeed, 1 high-speed, and 1 full-speed/low-speed).

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:3>	—	RAZ	—	—	Reserved.
<2:0>	RX_PRIORITY	R/W	0x0	0x0	Each register bit[*n*] controls the priority (1 = high, 0 = low) of RXFIFO[*n*] within a speed group.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Global Host DMA High/Low Ratio Register USBDRD(0..1)_UAHC_GDMAHLRATIO

This register specifies the relative priority of the SuperSpeed FIFOs with respect to the high-speed/full-speed/low-speed FIFOs. The DMA arbiter prioritizes the high-speed/full-speed/low-speed round-robin arbiter group every DMA high-low priority ratio grants as indicated in the register separately for TX and RX.

To illustrate, consider that all FIFOs are requesting access simultaneously, and the ratio is 4. SuperSpeed gets priority for four packets, high-speed/full-speed/low-speed gets priority for one packet, SuperSpeed gets priority for four packets, high-speed/full-speed/low-speed gets priority for one packet, and so on.

If FIFOs from both speed groups are not requesting access simultaneously then:

- If SuperSpeed got grants four out of the last four times, then high-speed/full-speed/low-speed get the priority on any future request.
- If high-speed/full-speed/low-speed got the grant last time, SuperSpeed gets the priority on the next request.
- If there is a valid request on either SuperSpeed or high-speed/full-speed/low-speed, a grant is always awarded; there is no idle.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:13>	—	RAZ	—	—	Reserved.
<12:8>	RX_RATIO	R/W	0x8	0x8	Speed ratio for RX arbitration.
<7:5>	—	RAZ	—	—	Reserved.
<4:0>	TX_RATIO	R/W	0x8	0x8	Speed ratio for TX arbitration.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Global Frame Length Adjustment Register USBDRD(0..1)_UAHC_GFLADJ

This register provides options for the software to control the core behavior with respect to SOF (start of frame) and ITP (isochronous timestamp packet) timers and frame timer functionality. It provides the option to override the sideband signal fladj_30mhz_reg. In addition, it enables running SOF or ITP frame timer counters completely off of the REF_CLK. This facilitates hardware LPM in host mode with the SOF or ITP counters being run off of the REF_CLK signal.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	GFLADJ_REFCLK_ 240MHZDECR_PLS1	R/W	0	0	This field indicates that the decrement value that the controller applies for each REF_CLK must be GFLADJ_REFCLK_240MHZ_DECR and GFLADJ_REFCLK_240MHZ_DECR +1 alternatively on each REF_CLK. Set this bit to 1 only if GFLADJ_REFCLK_LPM_SEL is set to 1 and the fractional component of 240/ref_frequency is greater than or equal to 0.5. **Example**: If the REF_CLK is 19.2 MHz then, • USBDRD(0..1)_UAHC_GUCTL[REF_CLK_PERIOD] = 52 • GFLADJ_REFCLK_240MHZ_DECR = (240/19.2) = 12.5 • GFLADJ_REFCLK_240MHZDECR_PLS1 = 1 If the REF_CLK is 24 MHz then, • USBDRD(0..1)_UAHC_GUCTL[REF_CLK_PERIOD] = 41 • GFLADJ_REFCLK_240MHZ_DECR = (240/24) = 10 • GFLADJ_REFCLK_240MHZDECR_PLS1 = 0

Cavium Confidential
Bowei Yang
ALU
02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<30:24>	GFLADJ_REFCLK_240MHZ_DECR	R/W	0x0	0x0	This field indicates the decrement value that the controller applies for each REF_CLK in order to derive a frame timer in terms of a 240-MHz clock. This field must be programmed to a non-zero value only if GFLADJ_REFCLK_LPM_SEL is set to 1. The value is derived as follows: GFLADJ_REFCLK_240MHZ_DECR = 240/ ref_clk_frequency **Examples**: If the REF_CLK is 24 MHz then, • USBDRD(0..1)_UAHC_GUCTL[REF_CLK_PERIOD] = 41 • GFLADJ_REFCLK_240MHZ_DECR = 240/24 = 10 If the REF_CLK is 48 MHz then, • USBDRD(0..1)_UAHC_GUCTL[REF_CLK_PERIOD] = 20 • GFLADJ_REFCLK_240MHZ_DECR = 240/48 = 5 If the REF_CLK is 17 MHz then, • USBDRD(0..1)_UAHC_GUCTL[REF_CLK_PERIOD] = 58 • GFLADJ_REFCLK_240MHZ_DECR = 240/17 = 14
<23>	GFLADJ_REFCLK_LPM_SEL	R/W	0	0	This bit enables the functionality of running SOF/ITP counters on the REF_CLK. This bit must not be set to 1 if USBDRD(0..1)_UAHC_GCTL[SOFITPSYNC] = 1. Similarly, if GFLADJ_REFCLK_LPM_SEL = 1, USBDRD(0..1)_UAHC_GCTL[SOFITPSYNC] must not be set to 1. When GFLADJ_REFCLK_LPM_SEL = 1 the overloading of the suspend control of the USB 2.0 first port PHY (UTMI) with USB 3.0 port states is removed. Note that the REF_CLK frequencies supported in this mode are 16/17/19.2/ 20/24/39.7/40 MHz. **NOTE:** If you set this bit to 1, USBDRD(0..1)_UAHC_GUSB2PHYCFG(0)[U2_FREECLK_EXISTS] must be set to 0.
<22>	—	R/W	0	0	Reserved.

Cavium Confidential For Bowman Yan 02/27/2019

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<21:8>	GFLADJ_REFCLK_FLADJ	R/W	0x0	0x0	This field indicates the frame length adjustment to be applied when SOF/ITP counter is running off of the REF_CLK. This register value is used to adjust: • ITP interval when USBDRD(0..1)_UAHC_GCTL[SOFITPSYNC] = 1 • both SOF and ITP interval when GFLADJ_REFCLK_LPM_SEL = 1. This field must be programmed to a non-zero value only if GFLADJ_REFCLK_LPM_SEL = 1 or USBDRD(0..1)_UAHC_GCTL[SOFITPSYNC] = 1. The value is derived as below: FLADJ_REF_CLK_FLADJ = ((125000/ ref_clk_period_integer) – (125000/ref_clk_period)) × ref_clk_period where, • the ref_clk_period_integer is the integer value of the REF_CLK period got by truncating the decimal (fractional) value that is programmed in USBDRD(0..1)_UAHC_GUCTL[REF_CLK_PERIOD] • the ref_clk_period is the REF_CLK period including the fractional value. **Examples**: If the REF_CLK is 24 MHz then, • USBDRD(0..1)_UAHC_GUCTL[REF_CLK_PERIOD] = 41 • GLADJ_REFCLK_FLADJ = ((125000/41) – (125000/41.6666)) × 41.6666 = 2032 (ignoring the fractional value) If the REF_CLK is 48 MHz then, • USBDRD(0..1)_UAHC_GUCTL[REF_CLK_PERIOD] = 20 • GLADJ_REFCLK_FLADJ = ((125000/20) – (125000/20.8333)) × 20.8333 = 5208 (ignoring the fractional value)
<7>	GFLADJ_30MHZ_REG_SEL	R/W	0	0	This field selects whether to use the input signal fladj_30mhz_reg or the GFLADJ_30MHZ to adjust the frame length for the SOF/ITP. When this bit is set to, 1, the controller uses GFLADJ_30MHZ value 0x0, the controller uses the input signal fladj_30mhz_reg value.
<6>	—	R/W	0	0	Reserved.
<5:0>	GFLADJ_30MHZ	R/W	0x0	0x0	This field indicates the value that is used for frame length adjustment instead of considering from the sideband input signal fladj_30mhz_reg. This enables post-silicon frame length adjustment in case the input signal fladj_30mhz_reg is connected to a wrong value or is not valid. The controller uses this value if GFLADJ_30MHZ_REG_SEL = 1 and the SOF/ITP counters are running off of UTMI(ULPI) clock (GFLADJ_REFCLK_LPM_SEL = 0 and USBDRD(0..1)_UAHC_GCTL[SOFITPSYNC] is 1 or 0). For details on how to set this value, refer to section 5.2.4 Frame Length Adjustment Register (FLADJ) of the xHCI Specification.

Cavium Confidential For Bowen Yang ALI 6/2/2014

Device Configuration Register USBDRD(0..1)_UAHC_DCFG

This register configures the core in device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:24>	—	RAZ	—	—	Reserved.
<23>	IGNORE STREAMPP	R/W	0	0	This bit only affects stream-capable bulk endpoints. When set to 0 and the controller receives a data packet with the Packet Pending (PP) bit set to 0 for OUT endpoints, or it receives an ACK with the NumP field set to 0 and PP set to 0 for IN endpoints, the core attempts to search for another stream (CStream) to initiate to the host. However, there are two situations where this behavior is not optimal: • When the host is setting PP = 0 even though it has not finished the stream, or • When the endpoint on the device is configured with one transfer resource and therefore does not have any other streams to initiate to the host. When set to 1, the core ignores the PP bit for the purposes of stream selection and does not search for another stream when it receives DP(PP = 0) or ACK(NumP = 0, PP = 0). This can enhance the performance when the device system-bus bandwidth is low
<22>	LPMCAP	R/W	0	0	LPM capable. The application uses this bit to control the controller's core LPM capabilities. If the core operates as a non-LPM-capable device, it cannot respond to LPM transactions. 0 = LPM capability is not enabled. 1 = LPM capability is enabled.
<21:17>	NUMP	R/W	0x4	0x4	Number of receive buffers. Indicates the number of receive buffers to be reported in the ACK TP. The DWC_usb3 controller uses this field if USBDRD(0..1)_UAHC_GRXTHRCFG[USBRXPKTCNTSEL] = 0. The application can program this value based on RxFIFO size, buffer sizes programmed in descriptors, and system latency. For an OUT endpoint, this field controls the number of receive buffers reported in the NumP field of the ACK TP transmitted by the core.
<16:12>	INTRNUM	R/W	0x0	0x0	Interrupt number. Indicates interrupt/EventQ number on which non-endpoint-specific device-related interrupts (see DEVT) are generated.
<11:10>	—	RAZ	—	—	Reserved. Read must ignore.

Cavium Confidential Prior Release Only 8/7/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<9:3>	DEVADDR	R/W	0x0	0x0	Device address. The application must perform the following: • Program this field after every SetAddress request. • Reset this field to 0x0 after USB reset.
<2:0>	DEVSPD	R/W	0x4	0x4	Device speed. Indicates the speed at which the application requires the core to connect, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected. 0x4 = SuperSpeed (USB 3.0 PHY clock is 125 MHz or 250 MHz) 0x0 = High-speed (USB 2.0 PHY clock is 30 MHz or 60 MHz) 0x1 = Full-speed (USB 2.0 PHY clock is 30 MHz or 60 MHz)

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Device Control Register
USBDRD(0..1)_UAHC_DCTL

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31>	RS	R/W	0	0	Run/stop. The software writes 1 to this bit to start the device controller operation. To stop the device controller operation, the software must remove any active transfers and write 0 to this bit. When the controller is stopped, it sets USBDRD(0..1)_UAHC_DSTS[DEVCTRLHLT] when the core is idle and the lower layer finishes the disconnect process. The Run/Stop bit must be used in following cases as specified: **1.** After power-on reset and CSR initialization, the software must write 1 to this bit to start the device controller. The controller does not signal connect to the host until this bit is set. **2.** The software uses this bit to control the device controller to perform a soft disconnect. When the software writes 0 to this bit, the host does not see that the device is connected. The device controller stays in the disconnected state until the software writes 1 to this bit. The minimum duration of keeping this bit cleared is 30ms in SuperSpeed and 10ms in high-speed/full-speed/low-speed. **3.** If the software attempts a connect after the soft disconnect or detects a disconnect event, it must set ULSTCHNGREQ to 0x5 (Rx.Detect) before reasserting this bit.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<30>	CSFTRST	R/W1S/H	0	0	Core soft reset. Resets the all clock domains as follows: • Clears the interrupts and all the CSRs except the following registers: USBDRD(0..1)_UAHC_GCTL, USBDRD(0..1)_UAHC_GUCTL, USBDRD(0..1)_UAHC_GSTS, USBDRD(0..1)_UAHC_GRLSID, USBDRD(0..1)_UAHC_GGPIO, USBDRD(0..1)_UAHC_GUID, USBDRD(0..1)_UAHC_GUSB2PHYCFG(0), USBDRD(0..1)_UAHC_GUSB3PIPECTL(0), USBDRD(0..1)_UAHC_DCFG, USBDRD(0..1)_UAHC_DCTL, USBDRD(0..1)_UAHC_DEVTEN, USBDRD(0..1)_UAHC_DSTS • All module state machines (except the SoC bus slave unit) are reset to the IDLE state, and all the TxFIFOs and the RxFIFO are flushed. • Any transactions on the SoC bus master are terminated as soon as possible, after gracefully completing the last data phase of a SoC bus transfer. Any transactions on the USB are terminated immediately. The application can write this bit at any time to reset the core. This is a self-clearing bit; the core clears this bit after all necessary logic is reset in the core, which may take several clock cycles depending on the corefs current state. Once this bit is cleared, the software must wait at least 3 PHY clock cycles before accessing the PHY domain (synchronization delay). Typically, software reset is used during software development and also when you dynamically change the PHY selection bits in the USB configuration registers listed above. When you change the PHY, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain must be reset for proper operation.
<29>	—	RAZ	—	—	Reserved.
<28:24>	HIRD_THRES	R/W	0x0	0x0	HIRD threshold. The core asserts output signals utmi_l1_suspend_n and utmi_sleep_n on the basis of this signal: • The core asserts utmi_l1_suspend_n to put the PHY into deep low-power mode in L1 when both of the following are true: – HIRD value ≥ the value in HIRD_THRES[3:0] (i.e. bits<27:24>) – HIRD_THRES[4] (i.e. bit<28>) is set to 1. • The core asserts utmi_sleep_n on L1 when one of the following is true: – HIRD value < the value in HIRD_THRES[3:0] (i.e. bits<27:24>) – HIRD_THRES[4] (i.e. bit<28>) is set to 0.

Cavium Confidential For Broadcom Internal Use 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<23>	APPL1RES	R/W	0	0	LPM Response Programmed by application. Handshake response to LPM token specified by device application. The response depends on USBDRD(0..1)_UAHC_DCFG[LPMCAP]. If LPMCAP = 0, the core always responds with timeout (that is, no response). If LPMCAP = 1, the core response is based on the value of this bit: 0 = the core responds with an ACK upon a successful LPM transaction, which requires all of the following conditions are satisfied – There are no PID/CRC5 errors in both the EXT token and the LPM token (if not true, inactivity results in a timeout ERROR) – A valid bLinkState = 0001B (L1) is received in the LPM transaction (else STALL) – No data is pending in the Transmit FIFO and OUT endpoints not in flow controlled state (else NYET) 1 = the core responds with an ACK upon a successful LPM, independent of transmit FIFO status and OUT endpoint flow control state. The LPM transaction is successful if all of the following conditions are satisfied. – There are no PID/CRC5 errors in both the EXT token and the LPM token (else ERROR) – A valid bLinkState = 0001B (L1) is received in the LPM transaction (else STALL)
<22:20>	—	RAZ	—	—	Reserved.
<19>	KEEPCONNECT	WO	0	0	Always write 0.
<18>	L1HIBERNATIONEN	WO	0	0	Always write 0.
<17>	CRS	WO	0	0	Controller restore state. This command is similar to the USBDRD(0..1)_UAHC_USBCMD[CRS] bit in host mode and initiates the restore process. When software sets this bit to 1, the controller immediately sets USBDRD(0..1)_UAHC_DSTS[RSS] to 1. When the controller has finished the restore process, it sets USBDRD(0..1)_UAHC_DSTS[RSS] to 0. **NOTE:** When read, this field always returns 0.
<16>	CSS	WO	0	0	Controller save state. This command acts similar to USBDRD(0..1)_UAHC_USBCMD[CSS] in host mode and initiates the restore process. When software sets this bit to 1, the controller immediately sets USBDRD(0..1)_UAHC_DSTS[SSS] = 1. When the controller has finished the save process, it sets USBDRD(0..1)_UAHC_DSTS[SSS] to 0. **NOTE:** When read, this field always returns 0.
<15:13>	—	RAZ	—	—	Reserved.
<12>	INITU2ENA	R/W	0	0	Initiate U2 enable: 0 = do not initiate (default), 1 = initiate On USB reset, hardware clears this bit to 0. Software sets this bit after receiving SetFeature(U2_ENABLE), and clears this bit when ClearFeature(U2_ENABLE) is received. If ACCEPTU2ENA = 0, the link immediately exits U2 state.

Cavium Confidential For Bowen Yang 2017/12/21

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<11>	ACCEPTU2 ENA	R/W	0	0	Accept U2 enable. 0 = Reject U2 except when Force_LinkPM_Accept bit is set (default) 1 = Core accepts transition to U2 state if nothing is pending on the application side. On USB reset, hardware clears this bit to 0. Software sets this bit to 1 after receiving a SetConfiguration command.
<10>	INITU1ENA	R/W	0	0	Initiate U1 enable: 0 = can not initiate (default), 1 = can initiate. On USB reset, hardware clears this bit to 0. Software sets this bit to 1 after receiving SetFeature(U1_ENABLE), and clears this bit when ClearFeature(U1_ENABLE) is received. If ACCEPTU1ENA = 0, the link immediately exits U1 state.
<9>	ACCEPTU1 ENA	R/W	0	0	Accept U1 enable. 0 = Reject U1 except when Force_LinkPM_Accept bit is set (default) 1 = Core accepts transition to U1 state if nothing is pending on the application side. On USB reset, hardware clears this bit to 0. Software sets this bit to 1 after receiving a SetConfiguration command.
<8:5>	ULSTCHNG REQ	WO	0x0	0x0	USB/Link-state change request. Software writes this field to issue a USB/Link state change request. A change in this field indicates a new request to the core. If software wants to issue the same request back-to-back, it must write a 0 to this field between the two requests. The result of the state change request is reflected in USBDRD(0..1)_UAHC_DSTS[USBLNKST]. These bits are self-cleared on the MAC layer exiting suspended state. If software is updating other fields of USBDRD(0..1)_UAHC_DCTL and not intending to force any link state change, then it must write a 0 to this field. SuperSpeed compliance mode is normally entered and controlled by the remote link partner. Refer to the USB3 specification. Alternatively, you can force the local link directly into compliance mode, by resetting the SuperSpeed link with RS set to 0. If you then write 0xA to ULSTCHNGREQ and 1 to RS, the Link goes to compliance. Once you are in compliance, you can alternately write 0x0 and 0xA to this field to advance the compliance pattern. In SuperSpeed mode: **Value — Requested Link State Transition/Action** 0x0 — No Action 0x4 — SS.Disabled 0x5 — Rx.Detect 0x6 — SS.Inactive 0x8 — Recovery 0xA — Compliance All other values are reserved. In high-speed/full-speed/low-speed mode: **Value — Requested USB State Transition** 0x8 — Remote wakeup request All other values are reserved. The Remote wakeup request should be issued 2µs after the device goes into suspend state (i.e. USBDRD(0..1)_UAHC_DSTS[USBLNKST] = 0x3).

Cavium Confidential For Bowen Yang 9/12/2018

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<4:1>	TSTCTL	R/W	0x0	0x0	Test control. 0x0 = Test mode disabled 0x1 = Test_J mode 0x2 = Test_K mode 0x3 = Test_SE0_NAK mode 0x4 = Test_Packet mode 0x5 = Test_Force_Enable All other values are reserved.
<0>	—	RAZ	—	—	Reserved.

Device Event-Enable Register USBDRD(0..1)_UAHC_DEVTEN

This register controls the generation of device-specific events. If an enable bit is set to 0, the event is not generated.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:13>	—	RAZ	—	—	Reserved.
<12>	VNDRDEVTSTRCVEDEN	R/W	0	0	Vendor device test LMP received event.
<11:10>	—	RAZ	—	—	Reserved.
<9>	ERRTICERREN	R/W	0	0	Erratic error event enable.
<8>	—	RAZ	—	—	Reserved.
<7>	SOFEN	R/W	0	0	Start of (micro) frame enable. For debug purposes only; normally software must disable this event.
<6>	U3L2L1SUSPEN	R/W	0	0	U3/L2-L1 suspend event enable.
<5>	HIBERNATIONREQEVTEN	R/W	0	0	This bit enables/disables the generation of the hibernation-request event.
<4>	WKUPEVTEN	R/W	0	0	Resume/remote-wakeup-detected event enable.
<3>	ULSTCNGEN	R/W	0	0	USB/Link state change event enable.
<2>	CONNECTDONEEN	R/W	0	0	Connection done enable.
<1>	USBRSTEN	R/W	0	0	USB reset enable.
<0>	DISCONNEVTEN	R/W	0	0	Disconnect detected event enable.

Cavium Confidential For Bowen Yang AEE 02/27/2014

Device Status Register
USBDRD(0..1)_UAHC_DSTS

This register indicates the status of the device controller with respect to USB-related events.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:30>	—	RAZ	—	—	Reserved.
<29>	DCNRD	RO/H	0	0	Device controller not ready. Will always read-as-zero.
<28>	SRE	R/W1C/H	0	0	Save/restore error. This bit is currently not supported.
<27:26>	—	RAZ	—	—	Reserved.
<25>	RSS	RO	0	0	Restore state status. This bit is similar to the USBDRD(0..1)_UAHC_USBSTS[RSS] in host mode. When the controller has finished the restore process, it completes the command by setting RSS to 0. Will always read as zero.
<24>	SSS	RO	0	0	Save state status. This bit is similar to the USBDRD(0..1)_UAHC_USBSTS[SSS] in host mode. When the controller has finished the save process, it completes the command by setting SSS to 0. Will always read-as-zero.
<23>	COREIDLE	RO/H	—	—	Core idle. Indicates that the core finished transferring all RxFIFO data to system memory, writing out all completed descriptors, and all event counts are 0x0. **NOTE:** While testing for reset values, mask out the read value. This bit represents the changing state of the core and does not hold a static value.
<22>	DEVCTRLHLT	RO/H	0	0	Device controller halted. When set to 1, the core does not generate device events. This bit is set to 0 when USBDRD(0..1)_UAHC_DCTL[RS] = 1. The core sets this bit to 1 when, after software sets USBDRD(0..1)_UAHC_DCTL[RS] to 0, the core is idle and the lower layer finishes the disconnect process.
<21:18>	USBLNKST	RO/H	0x4	0x4	USB/link state. In SuperSpeed mode, uses LTSSM state: 0x0 = U0 0x1 = U1 0x2 = U2 0x3 = U3 0x4 = SS_DIS 0x5 = RX_DET 0x6 = SS_INACT 0x7 = POLL 0x8 = RECOV 0x9 = HRESET 0xA = CMPLY 0xB = LPBK 0xF = resume/reset all other values = reserved In high-speed/full-speed/low-speed mode: 0x0 = on state 0x2 = sleep (L1) state 0x3 = suspend (L2) state 0x4 = disconnected state (default) 0x5 = early suspend state all other values = reserved The link state resume/reset indicates that the core received a resume or USB reset request from the host while the link was in hibernation. Software must write 0x8 (Recovery) to USBDRD(0..1)_UAHC_DCTL[ULSTCHNGREQ] to acknowledge the resume/reset request.

Cavium Confidential
Bowen Yang
ALU
02/27/20

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<17>	RXFIFOEMPTY	RO/H	1	1	RxFIFO empty indication.
<16:3>	SOFFN	RO/H	0x0	0x0	Frame/microframe number of the received SOF. When the core is operating at high-speed, ● <16:6> indicates the frame number ● <5:3> indicates the microframe number When the core is operating at full-speed, ● <16:14> is not used, software can ignore these bits. ● <13:3> indicates the frame number
<2:0>	CONNECTSPD	RO/H	0x4	0x4	Connected speed. Indicates the speed at which the controller core has come up after speed detection through a chirp sequence. 0x4 = SuperSpeed (PHY clock is running at 125 or 250 MHz) 0x0 = High-speed (PHY clock is running at 60 MHz) 0x1 = Full-speed (PHY clock is running at 60 MHz) 0x2 = Low-speed (not supported) 0x3 = Full-speed (PHY clock is running at 48 MHz)

Cavium Confidential For Bowen Yang ALU 02/27/2014

Device Generic Command Parameter Register USBDRD(0..1)_UAHC_DGCMDPAR

This register indicates the device command parameter. It must be programmed before or along with USBDRD(0..1)_UAHC_DGCMD.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1
- USBDRD(0..1)_UAHC_DCTL[CSFTRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	PARAM	R/W	0x0	0x0	Device generic command parameter. The usage depends on which USBDRD(0..1)_UAHC_DGCMD[CMDTYPE] is used. See usage notes in USBDRD_UAHC_DGCMD_CMDTYPE_E descriptions.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Device Generic Command Register USBDRD(0..1)_UAHC_DGCMD

This register enables software to program the core using a single generic-command interface to send link-management packets and notifications. This register contains command, control, and status fields relevant to the current generic command, while USBDRD(0..1)_UAHC_DGCMDPAR provides the command parameter.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1
- USBDRD(0..1)_UAHC_DCTL[CSFTRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	—	RAZ	—	—	Reserved.
<15>	CMDSTATUS	RO	0	0	Command status. 1 = CmdErr, indicates that the device controller encountered an error while processing the command. 0 = indicates command success
<14:11>	—	RAZ	—	—	Reserved.
<10>	CMDACT	R/W1S/H	0	0	Command active. The software sets this bit to 1 to enable the device controller to execute the generic command. The device controller sets this bit to 0 after executing the command.
<9>	—	RAZ	—	—	Reserved.
<8>	CMDIOC	WO	0	0	Command interrupt on complete. When set, the device controller issues a generic-command-completion event after executing the command. Note that this interrupt is mapped to USBDRD(0..1)_UAHC_DCFG[INTRNUM]. **NOTE:** This field must not set to 1 if the USBDRD(0..1)_UAHC_DCTL[RS] = 0.
<7:0>	CMDTYP	WO	0x0	0x0	Specifies the type of command the software driver is requesting the core to perform. See USBDRD_UAHC_DGCMD_CMDTYPE_E for encodings and usage.

Cavium Confidential For Bowen Yang 02/27/2014

Device Active USB Endpoint Enable Register USBDRD(0..1)_UAHC_DALEPENA

This register indicates whether a USB endpoint is active in a given configuration or interface.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1
- USBDRD(0..1)_UAHC_DCTL[CSFTRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	USBACTEP	R/W	0x0	0x0	This field indicates if a USB endpoint is active in the current configuration and interface. It applies to USB IN endpoints 0–15 and OUT endpoints 0–15, with one bit for each of the 32 possible endpoints. Even numbers are for USB OUT endpoints, and odd numbers are for USB IN endpoints, as follows: Bit<0> = USB EP0-OUT Bit<1> = USB EP0-IN Bit<2> = USB EP1-OUT Bit<3> = USB EP1-IN The entity programming this register must set bits<0> and <1> because they enable control endpoints that map to physical endpoints (resources) after USBReset. Application software clears these bits for all endpoints (other than EP0-OUT and EP0-IN) after detecting a USB reset. After receiving SetConfiguration and SetInterface requests, the application must program endpoint registers accordingly and set these bits.

Cavium Confidential For Bowen Feng ALU 02/27/2014

Device Physical Endpoint *n* Command Parameter Register 2 USBDRD(0..1)_UAHC_DEPCMDPAR2_(0..15)

This register indicates the physical endpoint command parameter 2. It must be programmed before issuing the command.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1
- USBDRD(0..1)_UAHC_DCTL[CSFTRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	PARAM2	R/W	—	—	Physical endpoint command parameter 2.

Device Physical Endpoint *n* Command Parameter Register 1 USBDRD(0..1)_UAHC_DEPCMDPAR1_(0..15)

This register indicates the physical endpoint command parameter 1. It must be programmed before issuing the command.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1
- USBDRD(0..1)_UAHC_DCTL[CSFTRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	PARAM1	R/W	—	—	Physical endpoint command parameter 1.

Cavium Confidential For Bowen Zang ALU 02/27/2014

Device Physical Endpoint *n* Command Parameter Register 0 USBDRD(0..1)_UAHC_DEPCMDPAR0_(0..15)

This register indicates the physical endpoint command parameter 0. It must be programmed before issuing the command.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1
- USBDRD(0..1)_UAHC_DCTL[CSFTRST] = 1

See Table 31–5 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:0>	PARAM0	R/W	—	—	Physical endpoint command parameter 0.

Device Physical Endpoint *n* Command Register USBDRD(0..1)_UAHC_DEPCMD(0..15)

This register enables software to issue physical endpoint-specific commands. It contains command, control, and status fields relevant to the current generic command, while the USBDRD(0..1)_UAHC_DEPCMDPAR* registers provide command parameters and return status information.

Several fields (including CMDTYPE) are write-only, so their read values are undefined. After power-up, prior to issuing the first endpoint command, the read value of this register is undefined. In particular, the CMDACT bit can be set after power-on. In this case, it is safe to issue an endpoint command.

This register can be reset in the following ways:

- IOI reset
- USBDRD(0..1)_UCTL_CTL[UAHC_RESET] = 1
- USBDRD(0..1)_UAHC_GCTL[CORESOFTRESET] = 1
- USBDRD(0..1)_UAHC_USBCMD[HCRST or LHCRST] = 1
- USBDRD(0..1)_UAHC_DCTL[CSFTRST] = 1

See Table 31–5 for the address.

Cavium Confidential For Boren Yang Eric U 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<31:16>	COMMAND PARAM	R/W	—	—	Command or event parameters. **When this register is written**: Command parameters: • For Start Transfer command: – <31:16> = StreamID. Specifies the USB StreamID assigned to this transfer. • For Start Transfer command applied to an isochronous endpoint: – <31:16> = StartMicroFramNum. Indicates the (micro)frame number to which the first TRB applies. • For Update Transfer, End Transfer, and Start New Configuration commands: – <22:16> = Transfer Resource Index (XferRscIdx). The hardware-assigned transfer resource index for the transfer, which was returned in response to the Start Transfer command. The application software-assigned transfer resource index for a Start New Configuration command. **When this register is read**: Event Parameters: • For XferNotReady, XferComplete, and Stream events on bulk endpoints: – <31:16> = StreamID. Applies only to bulk endpoints that support streams. This indicates the StreamID of the transfer for which the event is generated. • For XferInProgress: – <31:16> = Isochronous microframe number (IsocMicroFrameNum): Indicates the microframe number of the beginning of the interval that generated the XferInProgress event (debug purposes only). • For XferNotReady events on Isochronous Endpoints: – <31:16> = Isochronous microframe number (IsocMicroFrameNum). Indicates the microframe number during which the endpoint was not ready. **NOTE:** The controller core represents USB bus time as a 14-bit value on the bus and also in the DSTS register (USBDRD(0..1)_UAHC_DSTS[SOFFN]), but as a 16-bit value in the XferNotReady event. Use the 16-bit value to interact with Isochronous endpoints via the StartXfer command. The extra two bits that the controller core produces are necessary for handling wrap-around conditions in the interaction between software and hardware. • For all EPCmdCmplt events: – <27:24> = Command type. The command type that completed (valid only in a DEPEVT event; undefined when read from COMMANDPARAM). • For EPCmdCmplt event in response to Start Transfer command: – <22:16> = Transfer resource index (XferRscIdx). The internal hardware transfer resource index assigned to this transfer. This index must be used in all Update Transfer and End Transfer commands.

Cavium Confidential For Bowen Yang (xxxx)

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<15:12>	CMDSTATUS	R/W	—	—	Command completion status. Additional information about the completion of this command is available in this field. **Within an XferNotReady event**: <15> = indicates the reason why the XferNotReady event is generated: – 0 = XferNotActive: host initiated a transfer, but the requested transfer is not present in the hardware. – 1 = XferActive: host initiated a transfer, the transfer is present, but no valid TRBs are available. <14> = Not used. <13:12> = For control endpoints, indicates what stage was requested when the transfer was not ready: – 0x1 = Control data request. – 0x2 = Control status request. **Within an XferComplete or XferInProgress event**: <15> = LST bit of the completed TRB (XferComplete only). <15> = MissedIsoc: indicates the interval did not complete successfully (XferInProgress only). <14> = IOC bit of the TRB that completed. <13> = Indicates the TRB completed with a short packet reception or the last packet of an isochronous interval. <12> = Reserved. If the host aborts the data stage of a control transfer, software might receive a XferComplete event with the EventStatus field equal to 0. This is a valid event that must be processed as a part of the control-transfer programming model. **Within a Stream Event**: <15:12>: ● 0x2 = Stream not found; this stream event is issued when the stream-capable endpoint performed a search in its transfer resource cache, but could not find an active and ready stream. ● 0x1 = Stream found; this stream event is issued when the stream-capable endpoint found an active and ready stream in its transfer resource cache, and initiated traffic for that stream to the host. The ID of the selected Stream is in the EventParam field. **In response to a Start Transfer command**: <15:12>: ● 0x2 = Indicates expiry of the bus time reflected in the Start Transfer command. ● 0x1 = Indicates there is no transfer resource available on the endpoint. **In response to a Set Transfer Resource (DEPXFERCFG) command**: <15:12>: ● 0x1 = Indicates an error has occurred because software is requesting more transfer resources to be assigned than have been configured in the hardware. **In response to a End Transfer command**: <15:12>: ● 0x1 = Indicates an invalid transfer resource was specified.
<11>	HIPRI_ FORCERM	R/W	—	—	High priority: Only valid for Start Transfer command. ForceRM: Only valid for End Transfer command.

Cavium Confidential For Bowen Yang @2/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<10>	CMDACT	R/W	—	—	Software sets this bit to 1 to enable the device endpoint controller to execute the generic command. The device controller sets this bit to 0 when the CMDSTATUS field is valid and the endpoint is ready to accept another command. This does not imply that all the effects of the previously-issued command have taken place.
<9>	—	RAZ	—	—	Reserved. Uninitialized, read must ignore.
<8>	CMDIOC	R/W	—	—	Command interrupt on complete. When set, the device controller issues a generic Endpoint Command Complete event after executing the command. Note that this interrupt is mapped to USBDRD(0..1)_UAHC_DCFG[INTRNUM]. When the DEPCFG command is executed, the command interrupt on completion goes to the interrupt pointed by the USBDRD(0..1)_UAHC_DCFG[INTRNUM] in the current command. **NOTE:** This field must not set to 1 if the USBDRD(0..1)_UAHC_DCTL[RS] = 0.
<7:4>	—	RAZ	—	—	Reserved. Uninitialized, read must ignore.
<3:0>	CMDTYP	R/W	—	—	Command type. Specifies the type of command the software driver is requesting the core to perform. 0x0 = reserved 0x1 = Set Endpoint Configuration (64- or 96-bit parameter) 0x2 = Set Endpoint Transfer Resource Configuration (32-bit parameter) 0x3 = Get Endpoint State (no parameter needed) 0x4 = Set Stall (no parameter needed) 0x5 = Clear Stall (see Set Stall, no parameter needed) 0x6 = Start Transfer (64-bit parameter) 0x7 = Update Transfer (no parameter needed) 0x8 = End Transfer (no parameter needed) 0x9 = Start New Configuration (no parameter needed)

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 32

System Management Interface (SMI)

This chapter contains the following subjects:

- Overview
- SMI/MDIO Interface
- SMI Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

CN71XX supports up to two SMI interfaces. Each SMI is a two-wire, programmable-frequency, serial interface connecting CN71XX to its accompanying ethernet PHYs. It conforms to the IEEE802.3-2005 standard (refer to IEEE Std 802.3-2005, Sections 22.2.4, 22.3.4 and Section 45 cover SMI/MDIO).

The four signal names are:

- SMI0/1_MDC (management data clock – output clock signals)
- SMI0/1_MDIO (management data input/output – bidirectional signals exchanging control and status information between the SMI and the PHY.)

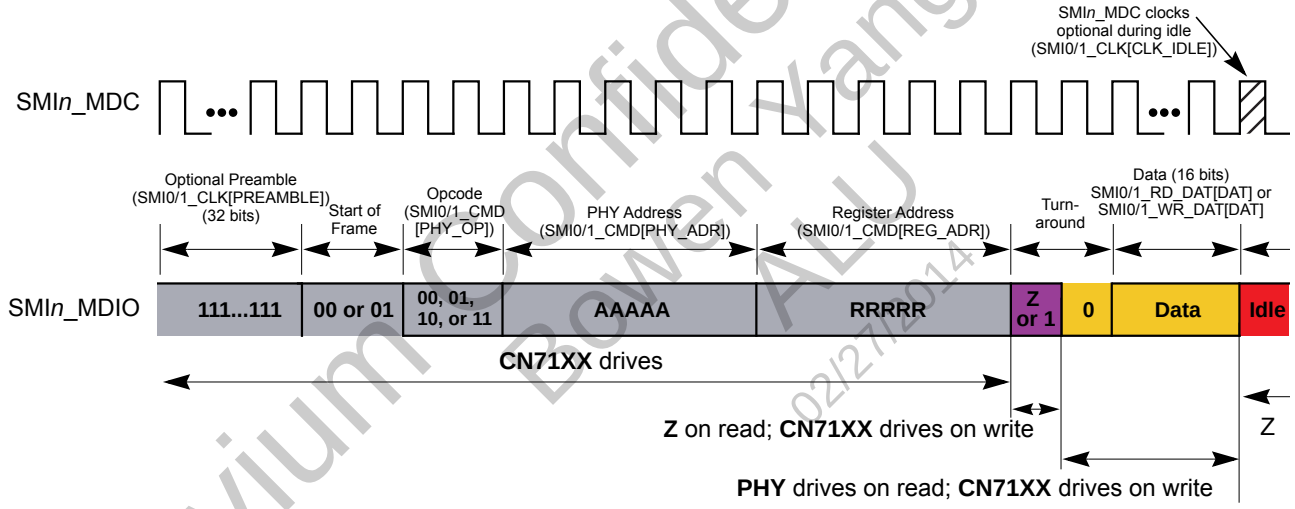
32.1 SMI/MDIO Interface

The SMI(0..1)_EN[EN] bit must be set to 1 to enable any SMI/MDIO transactions and/or to enable any SMI*n*_MDC clock transitions. During the idle time before and after transactions, CN71XX tri-states the SMI*n*_MDIO data wire, and optionally transitions the SMI*n*_MDC clock wire (if SMI(0..1)_CLK[CLK_IDLE] = 0).

If SMI0/1_CLK[MODE] = 1 (i.e. Clause 45 enabled), the Start of Frame field takes on the value 00 and the opcode field can take on any of the four possible values: 00 (address), 01 (write), 11 (read), or 10 post-read-increment address); if SMI0/1_CLK[MODE]=0 (i.e. Clause 22 enabled), the Start of Frame field takes on the value 01 and the opcode field is either 01 (write) or 10 (read).

Figure 32–1 shows an SMI/MDIO transaction mastered by CN71XX.

Figure 32–1 SMI/MDIO Transaction as Master



Software initiates a transaction by writing the SMI(0..1)_CMD register, specifying the following:

- in Clause 22 mode (SMI0/1_CLK[MODE] = 0), read or write operation (SMI0/1_CMD[PHY_OP]); in Clause 45 mode (SMI0/1_CLK[MODE] = 1), address, write, read, or post-read-increment-address operation
- PHY address (SMI0/1_CMD[PHY_ADR])
- register address (SMI0/1_CMD[REG_ADR]).

Cavium Confidential ETH
Bowen Yang
ALU
02/27/2014

CN71XX optionally drives the 32-bit preamble (SMI(0..1)_CLK[PREAMBLE]), then start of frame, operation code, PHY address, register address.

This is followed by the turnaround phase, and only the 16 data cycles remain.

- On a read or a post-read-increment address operation (e.g. SMI0/1_CMD[PHY_OP] = 11 or 10 in Clause 45 mode), the PHY drives (and CN71XX samples) the response for the remaining 16 data cycles.

  According to the 802.3 specification, on read operations, the STA (CN71XX) transitions SMI*n*_MDC and the PHY drives SMI*n*_MDIO with some delay relative to that edge. This is **Edge1** (refer to Figure 32–2).

  The STA then samples SMI*n*_MDIO on the next rising edge of SMI*n*_MDC. This is **Edge2** (refer to Figure 32–2).

  CN71XX can sample the read data relative to either edge using the SMI0/1_CLK[SAMPLE_MODE] register.

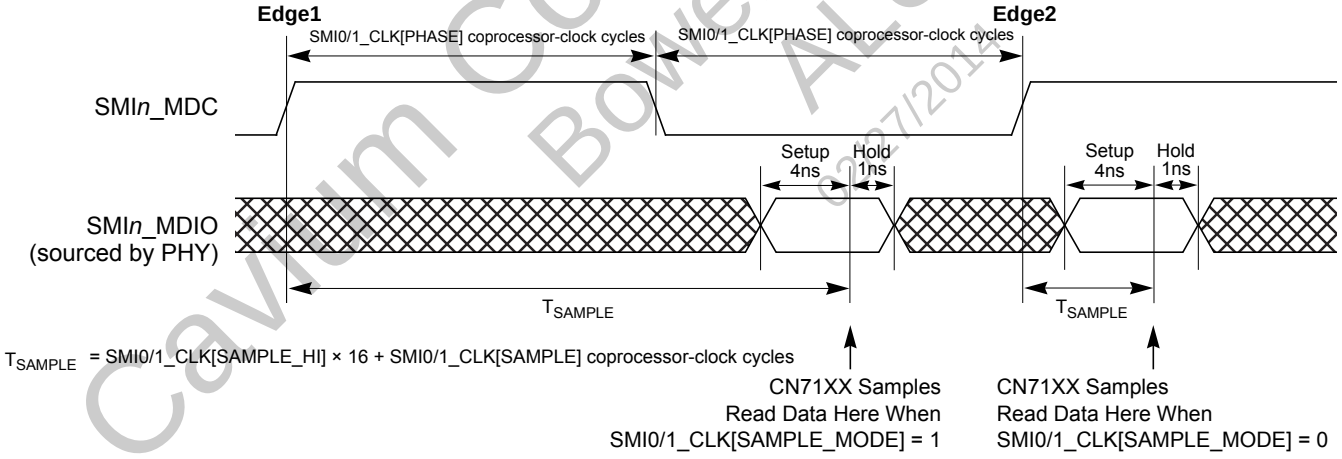
- On a write or address operation (e.g. SMI0/1_CMD[PHY_OP] = 01 or 00 in Clause 45 mode), CN71XX drives the write data for the remaining 16 cycles.

After software submits a read or a post-read-increment address operation via a write to SMI0/1_CMD (e.g. with SMI0/1_CMD[PHY_OP] = 11 or 10 in Clause 45 mode), it should poll SMI0/1_RD_DAT for the result. SMI0/1_RD_DAT[VAL] is set and SMI0/1_RD_DAT[DAT] contains the 16-bits of read data when the transaction is complete. CN71XX clears SMI0/1_RD_DAT[VAL] to 0 on a read operation, and SMI0/1_RD_DAT[DAT] contains valid data only when SMI0/1_RD_DAT[VAL] is set to 1. CN71XX asserts SMI0/1_RD_DAT[PENDING] while the transaction is pending.

Software must write the 16-bits of write data to SMI0/1_WR_DAT[DAT] before it initiates a write or address operation via a write to SMI0/1_CMD (e.g. with SMI0/1_CMD[PHY_OP] = 01 or 00 in Clause 45 mode). Software should then poll SMI0/1_WR_DAT[VAL] to tell when the transaction is complete. CN71XX clears SMI0/1_WR_DAT[VAL] to 0 once software has seen it set to 1. CN71XX asserts SMI0/1_WR_DAT[PENDING] while the transaction is pending.

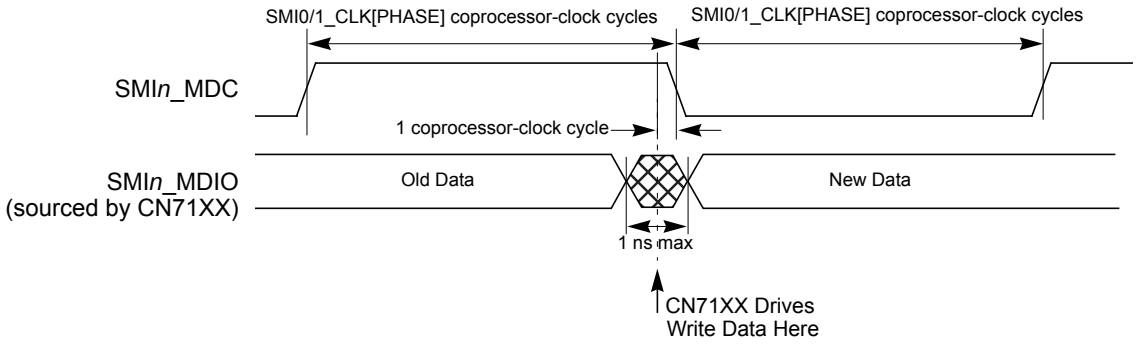
Figures 32–2 and 32–3 indicate timing characteristics of the interface.

Figure 32–2 SMI/MDIO Interface Read Timing Characteristics



Cavium Confidential For Bowen Yang ALU 2/27/2014

Figure 32–3 SMI/MDIO Interface Write Timing Characteristics



The SMI*n*_MDC frequency is defined as follows:

$$\text{SMI}n\text{_MDC frequency} = \frac{(\text{coprocessor frequency})}{(\text{SMI0/1_CLK[PHASE]} \times 2)}$$

CN71XX always drives write data one coprocessor-clock cycle before the falling edge of SMI*n*_MDC, and the sample point for read data is programmable relative to the rising edge of SMI*n*_MDC, based on SMI0/1_CLK[SAMPLE_HI,SAMPLE].

32.2 SMI Registers

The SMI registers are shown in Table 32–1.

Table 32–1 SMI Registers

Register	Address	CSR Type[1]	Detailed Description
SMI(0..1)_CMD	0x0001180000001800 + α * 0x100	RSL	See page 1477
SMI(0..1)_WR_DAT	0x0001180000001808 + α * 0x100	RSL	See page 1477
SMI(0..1)_RD_DAT	0x0001180000001810 + α * 0x100	RSL	See page 1478
SMI(0..1)_CLK	0x0001180000001818 + α * 0x100	RSL	See page 1478
SMI(0..1)_EN	0x0001180000001820 + α * 0x100	RSL	See page 1479
SMI_DRV_CTL	0x0001180000001828	RSL	See page 1479

1. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SMI Command Control Register
SMI(0..1)_CMD

This register forces a read or write command to the PHY. Write operations to this register create SMI transactions. Software will poll (depending on the transaction type). See Table 32–1 for the register address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:18>	—	RAZ	—	—	Reserved
<17:16>	PHY_OP	R/W	0	—	PHY Opcode depending on SMI_CLK[MODE] SMI_CLK[MODE] = 0 (<=1Gbs / Clause 22) 0x0 = write 0x1 = read SMI_CLK[MODE] = 1 (>1Gbs / Clause 45) 0x0 = address 0x1 = write 0x3 = read 0x2 = post-read-increment-address
<15:13>	—	RAZ	—	—	Reserved
<12:8>	PHY_ADR	R/W	0x0	—	PHY address
<7:5>	—	RAZ	—	—	Reserved
<4:0>	REG_ADR	R/W	0x0	—	PHY register offset

SMI Write Data Register
SMI(0..1)_WR_DAT

This register provides the data for a write operation. See Table 32–1 for the register address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:18>	—	RAZ	—	—	Reserved
<17>	PENDING	RO/H	0	—	Write transaction pending. Indicates that an SMI write transaction is in flight.
<16>	VAL	RO/H	0	—	Write data valid. Asserts when the write transaction completes. A read transaction to this register clears VAL.
<15:0>	DAT	R/W?H	0x0	—	Write data

Cavium Confidential For Bowen Yang ALD 02/27/2014

SMI Read Data Register
SMI(0..1)_RD_DAT

This register contains the data in a read operation. See Table 32–1 for the register address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:18>	—	RAZ	—	—	Reserved
<17>	PENDING	RO/H	0	—	Read transaction pending. Indicates that an SMI read transaction is in flight.
<16>	VAL	RO/H	0	—	Read data valid. Asserts when the read transaction completes. A read transaction to this register clears VAL.
<15:0>	DAT	RO/H	0x0	—	Read data

SMI Clock Control Register
SMI(0..1)_CLK

This register determines the SMI timing characteristics. See Table 32–1 for the register address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved
<24>	MODE	R/W	0	0	IEEE operating mode 0x0 = Clause 22 compliant 0x1 = Clause 45 compliant.
<23:21>	—	RAZ	—	—	Reserved
<20:16>	SAMPLE_HI	R/W	0x0	0x0	When to sample read data (extended bits)
<15>	SAMPLE_MODE	R/W	0	0	Read data sampling mode. According to the 802.3 specification, on read operations, the STA transitions SMI*n*_MDC and the PHY drives SMI*n*_MDIO with some delay relative to that edge. This is Edge1. The STA then samples SMI*n*_MDIO on the next rising edge of SMI*n*_MDC. This is Edge2. CN71XX can sample the read data relative to either edge. 0x0 = {SAMPLE_HI,SAMPLE} sample time is relative to Edge2 0x1 = {SAMPLE_HI,SAMPLE} sample time is relative to Edge1
<14>	—	RAZ	—	—	Reserved
<13>	CLK_IDLE	R/W	0	0	SMI*n*_MDC toggle. When set, this bit causes SMI*n*_MDC not to toggle on idle cycles.
<12>	PREAMBLE	R/W	1	1	Send preamble on SMI transaction. This field must be set to 1 when MODE = 1 in order for the receiving PHY to correctly frame the transaction.
<11:8>	SAMPLE	R/W	0x2	0x2	Sample read data. Specifies the number of coprocessor clock cycles after the rising edge of SMI*n*_MDC to wait before sampling read data. (SAMPLE_HI,SAMPLE) > 1 (SAMPLE_HI,SAMPLE) + 3 ≤ 2 × PHASE
<7:0>	PHASE	R/W	0x64	0x64	MDC clock phase. Specifies the number of coprocessor clock cycles that make up an SMI*n*_MDC phase. PHASE > 2

Cavium Confidential For Buwei Yang 2016-01-12

SMI Enable Register
SMI(0..1)_EN

Enables the SMI interface. See Table 32–1 for the register address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	EN	R/W	0	1	SMI/MDIO interface enable: 0x1 = enable interface 0x0 = disable interface: no transactions, no SMI*n*_MDC transitions

SMI Drive Strength Control Register
SMI_DRV_CTL

Enables the SMI interface. See Table 32–1 for the register address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:14>	—	RAZ	—	—	Reserved
<13:8>	PCTL	R/W	0x6	0x6	PCTL drive strength control bits. Suggested values: 0x4 = 75 Ω. 0x6 = 50 Ω. 0x7 = 40 Ω.
<7:6>	—	RAZ	—	—	Reserved
<5:0>	NCTL	R/W	0x6	0x6	NCTL drive strength control bits. Suggested values: 0x4 = 75 Ω. 0x6 = 50 Ω. 0x7 = 40 Ω.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 33

GPIO Unit

This section contains the following subjects:

- Overview
- GPIO Operations
- Glitch Filters
- GPIO Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The GPIO bus is a collection of general-purpose pins that can be configured for various tasks, and is a low-speed bus. Each pin can be configured as needed as one of the following:

- a data-input pin
- a data-output pin
- a clock-like output pin
- a level-sensitive interrupt pin (GPIO_0 – GPIO_15 only)
- an edge-triggered interrupt pin (GPIO_0 – GPIO_15 only)

The GPIO bus is controlled through the following registers.

- GPIO_BIT_CFG(0..15)
- GPIO_XBIT_CFG(16..19)
- GPIO_RX_DAT
- GPIO_TX_SET
- GPIO_TX_CLR
- GPIO_INT_CLR
- GPIO_CLK_GEN(0..3)
- GPIO_PIN_ENA
- GPIO_MULTI_CAST
- GPIO_CLK_SYNCE(0..1)

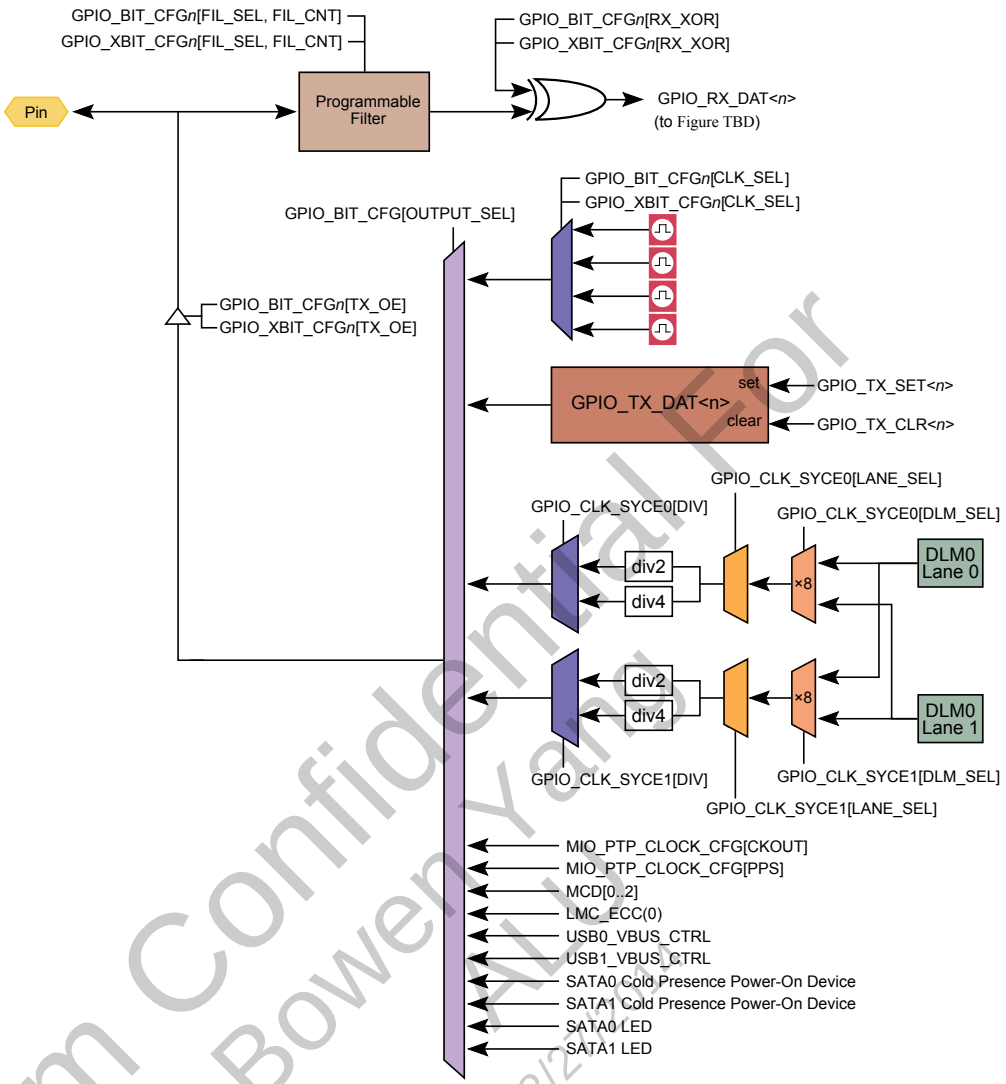
GPIO_BIT_CFG*n* controls the pin mode, including a glitch-filter.

Cavium Confidential For Bowen Yang ALU 02/27/2014

33.1 GPIO Operations

Figure 33–1 shows the GPIO cell.

Figure 33–1 GPIO Cell



Cavium Confidential For Bowen Yang 02/21/2014

33.1.1 Reading the GPIO Bus

To read the GPIO bus, software simply reads the GPIO_RX_DAT register, which contains the state of the GPIO pins.

33.1.2 Writing the GPIO Bus

To write the GPIO bus, software first picks which bits will be outputs by setting the appropriate GPIO_BIT_CFG*n*/GPIO_XBIT_CFG*n*[TX_OE] bits to turn on the output drivers for the desired pins. Then software can change the data (as indicated by the GPIO_TX_DAT element in Figure 33–1) by using the GPIO_TX_SET and GPIO_TX_CLR registers. Both these registers use a bit mask to change data.

Reading GPIO_TX_SET and GPIO_TX_CLR returns GPIO_TX_DAT.

33.1.3 GPIO Interrupts

Finally, GPIO sends interrupt information to the CIU based on the GPIO_BIT_CFG*n*[INT_EN] and GPIO_BIT_CFG*n*[INT_TYPE] (GPIO_0 – GPIO_15 only). If the pin is in edge-triggered mode, then software must clear the edge detector by writing to the GPIO_INT_CLR register or clear the edge detector through CIU_INT*_SUM0/4[GPIO] when multicast mode enabled.

When GPIO_MULTI_CAST[EN] is set and GPIO is edge-triggered (GPIO_BIT_CFG*n*[INT_TYPE] = 1), CN71XX can multicast four GPIO edge-triggered interrupts (GPIO<7:4>). When an edge is detected, multiple FLOPs (one per core) get set; each core can independently receive and store interrupts for each individual per-core CIU_INT*_SUM0/4[GPIO<7:4>] that can be cleared independently.

Writing one to any bit of CIU_INT*_SUM0/4[GPIO<7:4>] only clears the corresponding per-core interrupt. Writing one to any bit of CIU_INT*_SUM0/4[GPIO<15:8>,GPIO<3:0>] clears the common interrupt FLOP for bits other than CIU_INT*_SUM0/4[GPIO<7:4>].

GPIO_INT_CLR works for both multicast and non-multicast modes; writing one to any bit of GPIO_INT_CLR[TYPE] clears the CIU_INT*_SUM0/4 bits associated with the corresponding GPIO port.

33.1.4 GPIO Clock Generator

The GPIO unit contains four clock generators. Each GPIO pin can be configured to output a clock-like signal from one of the clock generators instead of just a static data pattern.

To use the GPIO clock feature for a given pin, software must perform the following steps:

1. Configure one of the GPIO clock generators to the desired frequency using GPIO_CLK_GEN*n*[N].
2. Select the clock generator by setting GPIO_BIT_CFG*n*[CLK_SEL] to the desired generator.
3. Enable the clock output by setting GPIO_BIT_CFG*n*[CLK_GEN] = 1 and GPIO_BIT_CFG*n*[TX_OE] = 1.

Cavium Confidential For Bowen Yang AU 02/27/2014

33.2 Glitch Filters

For each GPIO bit, data comes in through the appropriate pin and is synchronized into the coprocessor-clock domain through a DRSYNC cell. Then enters the glitch filter.

The global counter counts every coprocessor-clock cycle. The glitch filter taps into the global counter to control the sample rate. The bit select chooses one of the 16 bits to watch for a positive edge and to sample the pin.

For example (assume a 1.25 ns coprocessor clock, 800 MHz):

- A value of 0x0 would select counter<0> and would sample every two coprocessor-clock cycles or 2.5 ns.
- A value of 0x4 would select counter<4> which would make a sample every 32 coprocessor-clock cycles or 40 ns.
- Once a sample is made, if the sampled value ≠ current state, the local counter increments until the local counter = CSR count value.
- If sampled value = current state, then the counter resets.
- If the CSR count value = 0x0, then the filter is essentially disabled and just adds a pipe stage.

To get the two data points of 60ns and 1ms, the programming would be as follows (assuming a coprocessor-clock rate of 800 MHz):

60ns: bit_select = 1, count = 12

$$12 \times [2^{(1+1)} \times 1.25]\text{ns per sample} = 60\text{ ns}$$

1ms: bit_select = 15, count = 12

$$12 \times [2^{(15+1)} \times 1.25]\text{ns per sample} = 0.983040\text{ ms}$$

Cavium Confidential For Bowen Yang ALU 02/27/2014

33.3 GPIO Registers

The GPIO registers are listed in Table 33–1.

Table 33–1 GPIO Registers

Register	Address[1]	CSR Type[2]	Detailed Description
GPIO_BIT_CFG(0..15)	0x0001070000000800 + α*0x8	NCB	See page 1487
GPIO_RX_DAT	0x0001070000000880	NCB	See page 1488
GPIO_TX_SET	0x0001070000000888	NCB	See page 1488
GPIO_TX_CLR	0x0001070000000890	NCB	See page 1488
GPIO_INT_CLR	0x0001070000000898	NCB	See page 1488
GPIO_USBH_CTL	0x00010700000008A0	NCB	See page 1489
GPIO_SATA_CTL	0x00010700000008A8	NCB	See page 1489
GPIO_MULTI_CAST	0x00010700000008B0	NCB	See page 1490
GPIO_OCLA_EXTEN_TRIG	0x00010700000008B8	NCB	See page 1491
GPIO_CLK_GEN(0..3)	0x00010700000008C0 + α*0x8	NCB	See page 1492
GPIO_CLK_SYNCE(0..1)	0x00010700000008E0 + α*0x8	NCB	See page 1492
GPIO_XBIT_CFG(16..19)	0x0001070000000880 + α*0x8	NCB	See page 1494

1. For registers with multiple versions, α represents the values within the specified range.
2. NCB-type registers are accessed directly across the IOI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GPIO Bit Configuration Registers
GPIO_BIT_CFG(0..15)

Each register provides configuration information for the corresponding GPIO pin. See Table 33–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:21>	—	RAZ	—	—	Reserved
<20:16>	OUTPUT_SEL	R/W	0x0	0x0	Selects GPIO output. When TX_OE = 1, selects what output data is driven. 0x0 = TX Normal GPIO output, controlled by GPIO_TX. 0x1 = PTP_CKOUT, PTP CKOUT; see MIO_PTP_CLOCK_CFG. 0x2 = PTP_PPS, PTP PPS; see MIO_PTP_CLOCK_CFG. 0x3 = CLK_SYNCE0, see GPIO_CLK_SYNCE. 0x4 = CLK_SYNCE1, see GPIO_CLK_SYNCE. 0x5 = MCD0 multicore debug output; see TBD 0x6 = MCD1 multicore debug output; see TBD 0x7 = MCD2 multicore debug output; see TBD 0x8 = LMC_ECC(0) LMC ECC error detected. 0x9–0xF = Reserved. 0x10 = GPIO_CLK_GEN0, GPIO clock generator; see GPIO_CLK_GEN. 0x11 = GPIO_CLK_GEN1, GPIO clock generator; see GPIO_CLK_GEN. 0x12 = GPIO_CLK_GEN2, GPIO clock generator; see GPIO_CLK_GEN. 0x13 = GPIO_CLK_GEN3, GPIO clock generator; see GPIO_CLK_GEN. 0x14 = USB0_VBUS_CTRL, see USB0 Vbus Control. 0x15 = SATA port0 cold presence power-on device. (p0_cp_pod) 0x16 = SATA port1 cold presence power-on device. (p1_cp_pod) 0x17 = SATA port0 LED (p0_act_led) 0x18 = SATA port1 LED (p1_act_led) 0x19 = USB1_VBUS_CTRL, see USB1 Vbus control. 0x1A–0x1F = Reserved. **NOTE:** GPIO<19:10> controls are ignored if PCM is enabled. See PCM(0..3)_TDM_CFG
<15:12>	—	RAZ	—	—	Reserved
<11:8>	FIL_SEL	R/W	0x0	0x0	Filter select. Global counter bit-select (controls sample rate).
<7:4>	FIL_CNT	R/W	0x0	0x0	Filter count. Specifies the number of consecutive samples to change state.
<3>	INT_TYPE	R/W	0	0	Interrupt type: 0 = level-sensitive (default), 1 = rising-edge triggered.
<2>	INT_EN	R/W	0	0	Interrupt enable. Bit mask to indicate which bits to raise interrupt
<1>	RX_XOR	R/W	0	0	Receive inversion. When set to 1, inverts the received GPIO signal.
<0>	TX_OE	R/W	0	0	Transmit output enable. When set to 1, the GPIO pin is driven as an output pin.

Cavium Confidential
Bowen Yang
ALU
02/27/2014

GPIO Receive Data Register
GPIO_RX_DAT

This register contains the state of the GPIO pins. See Table 33–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:20>	—	RAZ	—	—	Reserved
<19:0>	DAT	RO/H	0x0	0x0	GPIO read data.

GPIO Transmit Data Set Register
GPIO_TX_SET

See Table 33–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:20>	—	RAZ	—	—	Reserved
<19:0>	SET	R/W1S/H	0x0	0x0	Set mask. Bit mask to indicate which GPIO_TX_DAT bits to set to 1. When read, SET returns the GPIO_TX_DAT storage.

GPIO Transmit Data Clear Register
GPIO_TX_CLR

See Table 33–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:20>	—	RAZ	—	—	Reserved
<19:0>	CLR	R/W1C/H	0x0	0x0	Clear mask. Bit mask to indicate which GPIO_TX_DAT bits to set to 0. When read, CLR returns the GPIO_TX_DAT storage.

GPIO Interrupt Clear Register
GPIO_INT_CLR

This register allows the software to clear any edge-triggered interrupts. Interrupts are limited to GPIO bits <15:0>. This register applies to GPIO_0–GPIO_15 only. See Table 33–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:16>	—	RAZ	—	—	Reserved
<15:0>	TYPE	R/W1C/H	0x0	0x0	Clear the interrupt rising edge detector.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GPIO USBH Control Register
GPIO_USBH_CTL

This register selects GPIO<19:0> received data (GPIO_RX_DAT[DAT]) routing to USB. The signals sent to USB from GPIO are filtered and optionally inverted. See Table 33–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	—	—	Reserved.
<12:8>	SEL	R/W/H	0x1F	0x0	Selects the GPIO<19:0> received data (GPIO_RX_DAT[DAT]) for USB0 over-current control. With SEL values 0x14–0x1F, signal is always 0. CSR read out for bits<12:8> have SEL1<4:0> value.
<7:5>	—	RAZ	—	—	Reserved.
<4:0>	SEL1	R/W/H	0x1F	0x0	Selects the GPIO<19:0> received data (GPIO_RX_DAT[DAT]) for USB1 over-current control. With SEL1 values 0x14–0x1F, signal is always 0. CSR read out for bits<4:0> have SEL<12:8> value.

GPIO SATA Control Register
GPIO_SATA_CTL

This register selects GPIO<19:0> received data (GPIO_RX_DAT[DAT]) routing to SATA. The signals sent to SATA from GPIO are filtered and optionally inverted. See Table 33–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:25>	—	RAZ	—	—	Reserved.
<24:20>	SEL4	R/W	0x1F	0x0	Selects the GPIO<19:0> received data (GPIO_RX_DAT[DAT]) for port1 cold presence detect (p1_cp_det). With SEL4 values 0x14–0x1F, signal is always 0.
<19:15>	SEL3	R/W	0x1F	0x0	Selects the GPIO<19:0> received data (GPIO_RX_DAT[DAT]) for port0 cold presence detect (p0_cp_det). With SEL3 values 0x14–0x1F, signal is always 0.
<14:10>	SEL2	R/W	0x1F	0x0	Selects the GPIO<19:0> received data (GPIO_RX_DAT[DAT]) for port1 mechanical presence detect (p1_mp_switch). With SEL2 values 0x14–0x1F, signal is always 0.
<9:5>	SEL1	R/W	0x1F	0x0	Selects the GPIO<19:0> received data (GPIO_RX_DAT[DAT]) for SATA port0 mechanical presence detect (p0_mp_switch). With SEL1 values 0x14–0x1F, signal is always 0.
<4:0>	SEL0	R/W	0x1F	0x0	Selects the GPIO<19:0> received data (GPIO_RX_DAT[DAT]) for SATA compliance lab loopback testing (lab_lb_pin). With SEL0 values 0x14–0x1F, signal is always 0.

Cavium Confidential For Beven Yang 09/7/2021

GPIO Multicast Control Register
GPIO_MULTI_CAST

This register enables multicast GPIO interrupts. See Table 33–1 for address.

GPIO_<7:4> have the option of operating in GPIO interrupt multicast mode. In this mode, the PP GPIO interrupts (CIU_INT<0–7>_SUM0/CIU_INT<0–3>_SUM4[GPIO<7:4>] values are stored per cnMIPS core.

For GPIO_<7:4> (*x*=4–7):

When GPIO_MULTI_CAST[EN] = 1:

When GPIO_BIT_CFG*x*[INT_EN]==1 & GPIO_BIT_CFG*x*[INT_TYPE]==1 (edge detection and interrupt enabled):

- Reads to CIU_INT<0..7>_SUM0/<0..3>_SUM4[GPIO<x>] will return a unique interrupt state per cnMIPS core.
- Reads to CIU_INT32/33_SUM0/4[GPIO<x>] will return the common GPIO<x> interrupt state.
- Write of '1' to CIU_INT<0..7>_SUM0/<0..3>_SUM4[GPIO<x>] will clear the individual interrupt associated with the cnMIPS core.
- Write of '1' to CIU_INT32/33_SUM0/4[GPIO<x>] will clear the common GPIO<x> interrupt state.
- Write of '1' to GPIO_INT_CLR[TYPE<x>] will clear all CIU_INT*_SUM0/4[GPIO<x>] state across all cnMIPS cores and common GPIO<x> interrupt states.

When GPIO_BIT_CFGx[INT_EN]==0 or GPIO_BIT_CFGx[INT_TYPE]==0, either leveled interrupt or interrupt not enabled, write of '1' to CIU_INT*_SUM0/4[GPIO<x>] will have no effects.

When GPIO_MULTI_CAST[EN] = 0: Write of '1' to CIU_INT_SUM0/4[GPIO<x>] will have no effects, as this field is RO, backward compatible with o63.

When GPIO_BIT_CFGx[INT_EN]==1 & GPIO_BIT_CFGx[INT_TYPE]==1 (edge detection and interrupt enabled)

- Reads to CIU_INT*_SUM0/4[GPIO<x>] will return the common GPIO<X> interrupt state.
- Write of '1' to GPIO_INT_CLR[TYPE<x>] will clear all CIU_INT*_SUM0/4[GPIO<x>] state across all cnMIPS cores and common GPIO<x> interrupt states.
- When GPIO_BIT_CFGx[INT_EN]==0 or GPIO_BIT_CFGx[INT_TYPE]==0, either leveled interrupt or interrupt not enabled, write of '1' to CIU_INT*_SUM0/4[GPIO<x>] will have no effects.

GPIO<19..8> and GPIO<3..0> will never be in multicast mode as those don't have per cnMIPS capabilities.

For GPIO<y> (y=0-3,8-19):

When GPIO_BIT_CFGx[INT_EN]==1 & GPIO_BIT_CFGx[INT_TYPE]==1 (edge detection and interrupt enabled):

Cavium Confidential For Bowen Yang ALU 02/27/2014

Reads to CIU_INT*_SUM0/4[GPIO<y>] will return the common GPIO<y> interrupt state.

Write of '1' to GPIO_INT_CLR[TYPE<y>] will clear all CIU_INT*_SUM0/4[GPIO<y>] common GPIO<y> interrupt states.

When GPIO_MULTI_CAST[EN] = 1:

Write of '1' to CIU_INT*_SUM0/4[GPIO<y>] will clear the common GPIO<y> interrupt state.

When GPIO_MULTI_CAST[EN] = 0:

Write of '1' to CIU_INT*_SUM0/4[GPIO<y>] has no effect, as this field is RO, backward compatible to o63.

When GPIO_BIT_CFGx[INT_EN]==0 or GPIO_BIT_CFGx[INT_TYPE]==0, either leveled interrupt or interrupt not enabled, write of '1' to CIU_INT*_SUM0/4[GPIO<y>] will have no effects.

Whenever there is mode change, (GPIO_BIT_CFGx[INT_EN] or GPIO_BIT_CFGx[INT_TYPE] or GPIO_MULTI_CAST[EN]) software needs to write to $GPIO_INT_CLR to clear up all pending/stale interrupts.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved
<0>	EN	R/W	0	0	Enable GPIO interrupt multicast mode. When EN is set, GPIO_<7:4> functions in multicast mode allowing these four GPIOs to interrupt multiple cores. Multicast functionality allows the GPIO to exist as per-cnMIPS interrupts as opposed to a global interrupt.

GPIO OCLA External Trigger Register GPIO_OCLA_EXTEN_TRIG

See Table 33–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:6>	—	RAZ	—	—	Reserved
<5>	M_TRIG	R/W	0	—	Manual trigger. Used only when SEL = 0x1F.
<4:0>	SEL	R/W	0x1F	0x0	Selects the GPIO(0..19) input pin or manual tTrigger to use in the OCLA coprocessor for triggering.

Cavium Confidential For Bowen Yang ALU 02/27/2014

GPIO Clock Generator Registers
GPIO_CLK_GEN(0..3)

See Table 33–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:32>	—	RAZ	—	—	Reserved
<31:0>	N	R/W	0x0	—	Determines the frequency of the GPIO clock generator. N should be less than or equal to 2^{31}–1. **NOTE:** The frequency of the GPIO clock generator equals the coprocessor-clock frequency times N divided by 2^{32}. $N = \frac{F_{GPIO\text{-}CLOCK} \times 2^{32}}{F_{COPROCESSOR\text{-}CLOCK}}$ **NOTE:** Writing N = 0x0 stops the clock generator.

GPIO Clock SYNCE Registers
GPIO_CLK_SYNCE(0..1)

A DLM can be configured as a clock source. The GPIO block can support up to two unique clocks to send out any GPIO pin as configured by GPIO_XBIT_CFG(16..19)[OUTPUT_SEL]. The clock can be divided by 20 or 40 of the selected RX lane clock. The following table shows the clock speed output for different modes.

Table 33–2 Clock Speed Output

SerDes speed (Gbd)	Speed with DIV = 0 (MHz)	Speed with DIV = 1 (MHz)
1.25	62.5	31.25
2.5	125	62.5
3.125	—	78.125
5.0	—	125

See Table 33–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved
<9:8>	DLM_SEL	R/W	0x1	0x1	Selects which DLM to select from: 0x0 = disabled 0x1 = select DLM0 as clock source 0x2 = select DLM1 as clock source 0x3 = select DLM2 as clock source
<7:3>	—	RAZ	—	—	Reserved
<2>	DIV	R/W	0	0	GPIO internal clock divider setting relative to QLM SERDES CLOCK_SYNCE. The maximum supported GPIO output frequency is 125 MHz. 0x0 = Divide by 20. 0x1 = Divide by 40. Speed [GHz] / DIV20 [MHz] / DIV40 [MHz] 1.2500 / 62.50 / 31.25 2.5000 / 125.00 / 62.50 3.1250 / --- / 78.13 5.0000 / --- / 125.00

Cavium Confidential For Bowen Yang ALU 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<1>	—	R/W	0	0	Reserved.
<0>	LANE_SEL	R/W	0	0	Selects which RX lane clock from DLM*x* to use as the GPIO internal DLM*x* clock. The GPIO block can support up to two unique clocks to send out any GPIO pin as configured by GPIO_BIT_CFG(0..15)[OUTPUT_SEL]. The clock can either be divided by 2 or by 4 of the selected RX lane clock.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

GPIO Bit Configuration Registers
GPIO_XBIT_CFG(16..19)

Each register provides configuration information for the corresponding GPIO pin. Only the first 16 GPIO pins can introduce interrupts. The [INT_EN, INT_TYPE] fields are not used and always read as 0. See Table 33–1 for address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:21>	—	RAZ	—	—	Reserved
<20:16>	OUTPUT_SEL	R/W	0x0	0x0	Selects GPIO output. When TX_OE = 1, selects what output data is driven. 0x0 = TX Normal GPIO output, controlled by GPIO_TX. 0x1 = PTP_CKOUT, PTP CKOUT; see MIO_PTP_CLOCK_CFG. 0x2 = PTP_PPS, PTP PPS; see MIO_PTP_CLOCK_CFG. 0x3 = CLK_SYNCE0, see GPIO_CLK_SYNCE. 0x4 = CLK_SYNCE1, see GPIO_CLK_SYNCE. 0x5 = MCD0 multicore debug output; see TBD 0x6 = MCD1 multicore debug output; see TBD 0x7 = MCD2 multicore debug output; see TBD 0x8 = LMC_ECC(0) LMC ECC error detected. 0x9–0xF = Reserved. 0x10 = GPIO_CLK_GEN0, GPIO clock generator; see GPIO_CLK_GEN. 0x11 = GPIO_CLK_GEN1, GPIO clock generator; see GPIO_CLK_GEN. 0x12 = GPIO_CLK_GEN2, GPIO clock generator; see GPIO_CLK_GEN. 0x13 = GPIO_CLK_GEN3, GPIO clock generator; see GPIO_CLK_GEN. 0x14 = USB0_VBUS_CTRL, see USB0 Vbus Control. 0x15 = SATA port0 cold presence power-on device. (p0_cp_pod) 0x16 = SATA port1 cold presence power-on device. (p1_cp_pod) 0x17 = SATA port0 LED (p0_act_led) 0x18 = SATA port1 LED (p1_act_led) 0x19 = USB1_VBUS_CTRL, see USB1 Vbus control. 0x1A–0x1F = Reserved. **NOTE:** GPIO<19:10> controls are ignored if PCM is enabled. See PCM(0..3)_TDM_CFG
<15:12>	—	RAZ	—	—	Reserved
<11:8>	FIL_SEL	R/W	0x0	0x0	Filter select. Global counter bit-select (controls sample rate).
<7:4>	FIL_CNT	R/W	0x0	0x0	Filter count. Specifies the number of consecutive samples to change state.
<3>	INT_TYPE	R/W	0	0	Interrupt type: 0 = level-sensitive (default), 1 = rising-edge triggered.
<2>	INT_EN	R/W	0	0	Interrupt enable. Bit mask to indicate which bits to raise interrupt
<1>	RX_XOR	R/W	0	0	Receive inversion. When set to 1, inverts the received GPIO signal.
<0>	TX_OE	R/W	0	0	Transmit output enable. When set to 1, the GPIO pin is driven as an output pin.

Cavium Confidential For Review Only 02/27/2014

Chapter 34

UART Interface

This chapter contains the following subjects:

- Overview
- Pin Usage
- UART (RS232) Serial Protocol
- Interrupts
- AutoFlow Control
- UART Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

CN71XX has two UART interfaces. Both interfaces contain the essential SIN, SOUT, CTS, and RTS signals. Interface 0 also contains the DTR and DCD signals. The UARTs are typically used for serial communication with a peripheral, modem (data carrier equipment, DCE), or data set. Either a core or a remote host can use the UARTs. However, for the remainder of this chapter, we assume use is only by the cores.

The CN71XX cores transfer bytes to and receive characters from the UART core via 64-bit CSR accesses. The UART core transfers and receives the characters serially. Either polling or interrupts can be used to transfer the bytes.

The UARTs have the following characteristics:

- modeled after the industry-standard 16550
  - programmable data bits per character, optional odd/even parity generation/ checking, and number of stop bits
  - line break generation and detection
- programmable baud rate = coprocessor-clock speed / (divisor × 16)
  - up to 10 Mbaud
- 64-byte transmit and receive FIFOs
- 16750-compatible auto flow control
- interrupts
  - several prioritized interrupt types
  - separate enables for each type
  - programmable transmit holding register empty interrupt
- shadow registers to read/write important fields individually
- software programmable reset
- testing/diagnostic capabilities
  - loopback for RTS/CTS testing
  - FIFO status and testing
  - transmit halt mode

34.1 Pin Usage

Some of the UART pins function as either UART pins or MPI/SPI pins. Table 34–1 describes the pins and the CSR fields that enable the UART pins and disable the MPI/SPI functionality.

Table 34–1 Shared UART/MPI/SPI Signals

Ball Name	Description	Enable UART	UART Signals
UART0_DTR_L/SPI_DO	UART0 data terminal ready	MPI_CFG[ENABLE] = 0	UART0_DTR_L
UART0_DCD_L/SPI_DI	UART0 data carrier detect	MPI_CFG[ENABLE] = 0	UART0_DCD_L

NOTE: The UART0_RTS_L signal is used as a bootstrap signal at power up. This signal has an internal pull-up resistor that causes the standard PLL_REF_CLK to be used as the PLL source. An external pull-down resistor can be attached, which causes DLMC_REF_CLK1 to be the PLL source. DLMC_REF_CLK1 must be 100 MHz, with no spread-spectrum clocking.

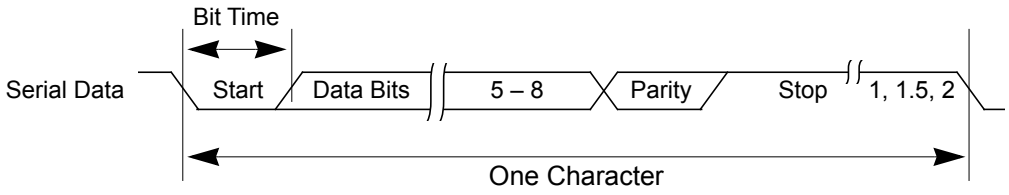
After reset, UART0_RTS_L performs it normal functions.

Cavium Confidential For Bowen Yang ALU 02/27/2014

34.2 UART (RS232) Serial Protocol

Because the serial communication is asynchronous, additional bits (start and stop) are added to the serial data to indicate the beginning and end. Using these bits allows two devices to be synchronized. This structure of serial data accompanied by start and stop bits is referred to as a character, as shown in Figure 34–1.

Figure 34–1 Serial Data Format

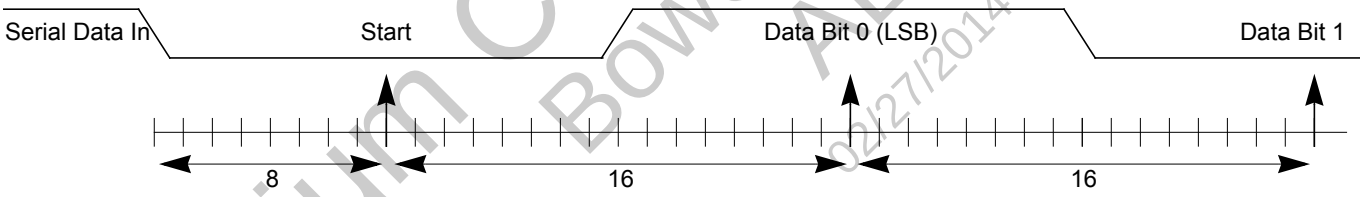


An additional parity bit may be added to the serial character. This bit appears after the last data bit and before the stop bit or bits in the character structure for simple error checking on the received data.

The line control register (MIO_UART_*n*_LCR) is used to control the serial-character characteristics. The individual bits of the data word are sent after the start bit, starting with the least-significant bit (LSB). These are followed by the optional parity bit, followed by the stop bits, which can be 1, 1.5 or 2.

All the bits in the transmission (with exception to the half stop bit when 1.5 stop bits are used) are transmitted for exactly the same time duration, which is referred to as a bit period or bit time. One bit time equals 16 baud-clock cycles. To ensure stability on the line, the receiver samples the serial input data at approximately the mid point of the bit time once the start bit has been detected. As the exact number of baud-clock cycles that each bit was transmitted for is known, calculating the mid point for sampling is not difficult, i.e. every 16 baud-clock cycles after the mid point sample of the start bit. Figure 34–2 shows the sampling points of the first couple of bits in a serial character.

Figure 34–2 Receiver Serial Data Sample Points



The baud clock is a programmable number of coprocessor-clock cycles selected by the divisor latch register (MIO_UART0/1_DLH and MIO_UART0/1_DLL). With a divisor of one, the baud clock equals the CN71XX coprocessor clock, and the baud rate is 1/16 the coprocessor clock.

Cavium Confidential For Bowen Wang ALU 02/27/2014

34.3 Interrupts

For the following section, refer to Figure 13–1 in the CIU chapter. The UART interrupt asserts whenever one of the several prioritized interrupt types are enabled and active. That is, it asserts whenever MIO_UART0/1_IIR[IID] ≠ 1.

The following interrupt types can be enabled with the MIO_UART0/1_IER register:

- receiver error
- receiver data available
- character timeout (in FIFO mode only)
- transmitter holding register empty or at or below threshold (in programmable THRE interrupt mode)
- modem status changed

These interrupt types are covered in more detail in the MIO_UART_IIR description.

34.4 AutoFlow Control

CN71XX implements a 16750-compatible autoRTS and autoCTS serial-data flow-control mode. It is important to note that autoRTS and autoCTS modes can only be enabled when the FIFOs are enabled (i.e. when MIO_UART0/1_FCR[EN] is set).

34.4.1 UART AutoRTS

AutoRTS is active when:

- MIO_UART0/1_MCR[RTS] and MIO_UART0/1_MCR[AFCE] are both set
- The FIFOs are enabled (i.e. MIO_UART0/1_FCR[EN] is set)

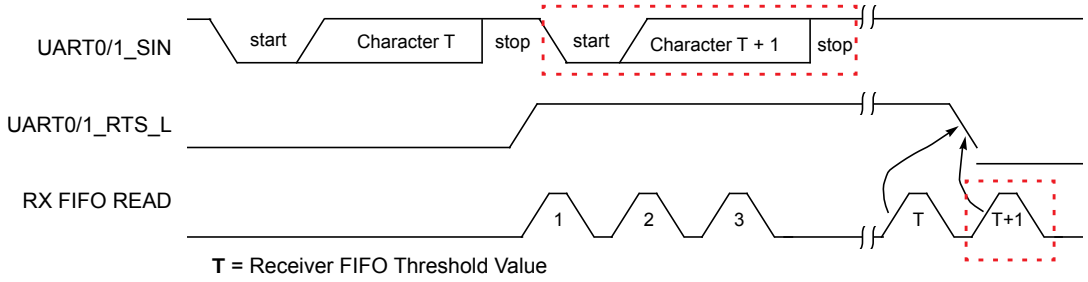
When autoRTS is enabled (active), UART0/1_RTS_L is forced inactive (high) when the receiver FIFO level reaches the threshold set by MIO_UART0/1_FCR[RXTRIG]. When UART0/1_RTS_L is connected to the **cts_l** input of another UART device, the other UART stops sending serial data until the receiver FIFO has available space (until it is completely empty).

The selectable receiver FIFO threshold values are: 1, ¼, ½, and “2 less than full.” Since one additional character may be transmitted to CN71XX after **rts_n** has become inactive (due to data already having entered the transmitter block in the other UART), setting the threshold to “2 less than full” allows maximum use of the FIFO with a safety zone of one character.

Once the cores drain the receiver FIFO completely empty (by reading MIO_UART0/1_RBR), UART0/1_RTS_L again becomes active (low), signalling the other UART to continue sending data. Figure 34–3 shows a timing diagram of autoRTS operation.

Cavium Confidential For Bowen Yang 02/12/2014

Figure 34–3 AutoRTS Timing



T = Receiver FIFO Threshold Value

Character **T+1** is received because UART0/1_RTS_L was not detected before the next character entered the sending UART's transmitter.

When autoRTS is disabled, UART0/1_RTS_L is controlled solely by MIO_UART0/1_MCR[RTS].

34.4.2 UART AutoCTS

AutoCTS is active when:

- MIO_UART0/1_MCR[AFCE] is set to 1;
- The FIFOs are enabled (i.e. MIO_UART0/1_FCR[EN] is set).

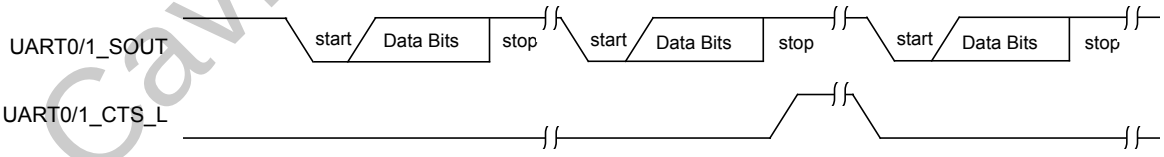
When autoCTS is enabled (active), CN71XX stops transmitting whenever the UART0/1_CTS_L is inactive (high). This prevents overflowing the FIFO of the receiving UART. If UART0/1_CTS_L is not made inactive before the middle of the last stop bit, CN71XX transmits another character before stopping. While CN71XX is not transmitting, the transmitter FIFO can still be written, and even overflowed.

When autoCTS is enabled, the cores can poll any of the following before each write to avoid transmitter FIFO overflow:

- The cores can poll MIO_UART0/1_UART[TFNF] to check if the transmit FIFO is full.
- The cores can poll MIO_UART0/1_TFL[RFL] for the current transmit FIFO level.
- The cores can poll MIO_UART0/1_LSR[THRE] when programmable THRE Interrupt mode is enabled. MIO_UART0/1_LSR[THRE] indicates the transmit FIFO full status in that case. See Section 34.4.3 for more details on programmable THRE interrupt mode.

When UART0/1_CTS_L becomes active (low) again after being inactive (high), transmission resumes. Figure 34–4 is a timing diagram showing autoCTS operation.

Figure 34–4 AutoCTS Timing



When autoCTS is disabled, the transmitter is unaffected by UART0/1_CTS_L.

Cavium Confidential For Bowen Yang @2/27/2015

34.4.3 UART Programmable THRE Interrupt

System performance can be improved with the UART's programmable THRE Interrupt mode. Programmable THRE Interrupt mode is enabled when:

- MIO_UART_IER[PTIME] is set.
- The FIFOs are enabled (i.e. MIO_UART0/1_FCR[EN] is set).

When programmable THRE interrupt mode is enabled, a THRE interrupt is active when the FIFO size is below the threshold selected by MIO_UART0/1_FCR[TXTRIG]. The available empty thresholds are:

- empty
- 2
- ¼
- ½.

When programmable THRE interrupt mode is enabled, the MIO_UART0/1_LSR[THRE] bit also switches its function. Instead of asserting when the transmitter FIFO is empty, it asserts when the FIFO is full, which allows software to fill the FIFO each transmit sequence by polling MIO_UART0/1_LSR[THRE] before writing another character. Proper threshold value selection may both maintain constant character transmission and minimize interruptions.

When programmable THRE interrupt mode is not enabled, THRE interrupts occur when the transmit FIFO is empty, and MIO_UART0/1_LSR[THRE] asserts on an empty transmit FIFO.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

34.5 UART Registers

The UART registers are listed in Table 34–2.

Table 34–2 UART Registers

Register	Address	CSR Type[1]	Detailed Description
MIO_UART(0..1)_RBR	0x0001180000000800 + α*0x400	RSL	See page 1502
MIO_UART(0..1)_IER	0x0001180000000808 + α*0x400	RSL	See page 1503
MIO_UART_IIR	0x0001180000000810 + α*0x400	RSL	See page 1504
MIO_UART(0..1)_LCR	0x0001180000000818 + α*0x400	RSL	See page 1505
MIO_UART(0..1)_MCR	0x0001180000000820 + α*0x400	RSL	See page 1506
MIO_UART(0..1)_LSR	0x0001180000000828 + α*0x400	RSL	See page 1508
MIO_UART(0..1)_MSR	0x0001180000000830 + α*0x400	RSL	See page 1509
MIO_UART(0..1)_SCR	0x0001180000000838 + α*0x400	RSL	See page 1510
MIO_UART(0..1)_THR	0x0001180000000840 + α*0x400	RSL	See page 1510
MIO_UART(0..1)_FCR	0x0001180000000850 + α*0x400	RSL	See page 1511
MIO_UART(0..1)_DLL	0x0001180000000880 + α*0x400	RSL	See page 1512
MIO_UART(0..1)_DLH	0x0001180000000888 + α*0x400	RSL	See page 1512
MIO_UART(0..1)_FAR	0x0001180000000920 + α*0x400	RSL	See page 1513
MIO_UART(0..1)_TFR	0x0001180000000928 + α*0x400	RSL	See page 1513
MIO_UART(0..1)_RFW	0x0001180000000930 + α*0x400	RSL	See page 1513
MIO_UART(0..1)_USR	0x0001180000000938 + α*0x400	RSL	See page 1514
MIO_UART(0..1)_TFL	0x0001180000000A00 + α*0x400	RSL	See page 1514
MIO_UART(0..1)_RFL	0x0001180000000A08 + α*0x400	RSL	See page 1514
MIO_UART(0..1)_SRR	0x0001180000000A10 + α*0x400	RSL	See page 1515
MIO_UART(0..1)_SRTS	0x0001180000000A18 + α*0x400	RSL	See page 1515
MIO_UART(0..1)_SBCR	0x0001180000000A20 + α*0x400	RSL	See page 1515
MIO_UART(0..1)_SFE	0x0001180000000A30 + α*0x400	RSL	See page 1516
MIO_UART(0..1)_SRT	0x0001180000000A38 + α*0x400	RSL	See page 1516
MIO_UART(0..1)_STT	0x0001180000000B00 + α*0x400	RSL	See page 1516
MIO_UART(0..1)_HTX	0x0001180000000B08 + α*0x400	RSL	See page 1516

1. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential
Bowen Yang
ALD
02/27/2014

MIO UART0/1 Receive Buffer Register
MIO_UART(0..1)_RBR

The receive buffer register is a read-only register that contains the data byte received on the serial input port (SIN). The data in this register is valid only if the MIO_UART(0..1)_LSR[DR] bit is set (See the "MIO UART0/1 Line Status Register" on page 1508). See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved.
<7:0>	RBR	RO/H	0x0	—	Receive buffer register. Contains the data byte received on the serial input port

- When the FIFOs are programmed ON, this register accesses the head of the receive FIFO.
- When the FIFOs are programmed OFF, the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an overrun error.
- If the receive FIFO is full (64 characters) and this register is not read before the next data character arrives, then the data already in the FIFO is preserved, but any incoming data is lost. An overrun error also occurs.

NOTE: MIO_UART(0..1)_LCR[DLAB] must be clear to access this register.

The address in Table 34–2 is an alias to simplify these CSR descriptions. The MIO_UART(0..1)_RBR, MIO_UART(0..1)_THR, and MIO_UART(0..1)_DLL registers are one register with different uses at different times.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO UART0/1 Interrupt-Enable Register MIO_UART(0..1)_IER

The interrupt-enable register is a read/write register that contains four bits that enable the generation of interrupts:

- enable received data available interrupt (ERBFI)
- enable transmitter holding register empty interrupt (ETBEI)
- enable receiver line status interrupt (ELSI)
- enable modem status interrupt (EDSSI).

The IER also contains the enable bit for the programmable transmit holding register empty (THRE) interrupt mode (PTIME). See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved.
<7>	PTIME	R/W	0	—	Programmable THRE interrupt-mode enable
<6:4>	—	RAZ	—	—	Reserved.
<3>	EDSSI	R/W	0	—	Enable modem status interrupt
<2>	ELSI	R/W	0	—	Enable receiver line status interrupt
<1>	ETBEI	R/W	0	—	Enable transmitter holding register empty interrupt
<0>	ERBFI	R/W	0	—	Enable received data available interrupt

NOTE: MIO_UART(0..1)_LCR[DLAB] must be clear to access this register.

The address in Table 34–2 is an alias to simplify these CSR descriptions. The MIO_UART(0..1)_IER and MIO_UART(0..1)_DLH registers are one register with different uses at different times.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO UART0/1 Interrupt Identity Register
MIO_UART_IIR

The interrupt identity register is a read-only register that identifies the source of an interrupt. See Table 34–2 for the address.

<table>
<tr><th>Bit Pos</th><th>Field Name</th><th>Access</th><th>Reset Value</th><th>Typical Value</th><th>Field Description</th></tr>
<tr><td><63:8></td><td>—</td><td>RAZ</td><td>—</td><td>—</td><td>Reserved.</td></tr>
<tr><td><7:6></td><td>FEN</td><td>RO</td><td>0x0</td><td>—</td><td>FIFO-enabled.
00 = FIFOs disabled, 01 = reserved, 10 = reserved, 11 = FIFOs enabled</td></tr>
<tr><td><5:4></td><td>—</td><td>RAZ</td><td>0x0</td><td>—</td><td>Reserved.</td></tr>
<tr><td><3:0></td><td>IID</td><td>RO</td><td>0x1</td><td>—</td><td>Interrupt ID. Identifies the highest priority pending interrupt. The interrupt-source decoding, interrupt priority, and interrupt-reset control are shown in the following table:
<table>
<tr><th>IID</th><th>Priority Level</th><th>Interrupt Type</th><th>Interrupt Source</th><th>Interrupt reset by:</th></tr>
<tr><td>0001</td><td>—</td><td>None</td><td>None</td><td>—</td></tr>
<tr><td>0110</td><td>highest</td><td>Receiver line status</td><td>Overrun, parity, or framing errors, or break interrupt</td><td>Reading the line-status register.</td></tr>
<tr><td>0100</td><td>second</td><td>Received data available</td><td>Receiver data available (FIFOs disabled) or RX FIFO trigger level reached (FIFOs enabled)</td><td>Reading the receiver buffer register (FIFOs disabled) or the FIFO drops below the trigger level (FIFOs enabled)</td></tr>
<tr><td>1100</td><td>second</td><td>Character Timeout Indication</td><td>No characters in or out of the RX FIFO during the last four character times and there is at least one character in it during this time.</td><td>Reading the receiver buffer register.</td></tr>
<tr><td>0010</td><td>third</td><td>Transmitter holding register empty</td><td>Transmitter holding register empty (programmable THRE mode disabled) or TX FIFO at or below threshold (programmable THRE mode enabled)</td><td>Reading the interrupt identity register (if it is the source of the interrupt) or writing into MIO_UART(0..1)_THR (FIFOs or THRE mode disabled) or TX FIFO above threshold (FIFOs and THRE mode enabled)</td></tr>
<tr><td>0000</td><td>fourth</td><td>Modem status changed</td><td>Clear to send (CTS),[1] data set ready (DSR), ring indicator (RI), or data carrier detect (DCD) changed.</td><td>Reading the modem status register.</td></tr>
</table></td></tr>
</table>

1. A change in CTS causes an interrupt only when autoflow-control mode is disabled.

NOTE: The address in Table 34–2 is an alias to simplify these CSR descriptions. The MIO_UART(0..1)_FCR and MIO_UART_IIR registers are one register with different uses at different times.

Cavium Confidential For Bowen Xing AE 10/7/2014

MIO UART0/1 Line Control Register MIO_UART(0..1)_LCR

The line control register controls the format of the data that is transmitted and received by the UART. It is always readable and writable. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description[1]
<63:8>	—	RAZ	—	—	Reserved.
<7>	DLAB	R/W	0	—	Divisor latch address bit. Setting this bit enables reading and writing of the divisor latch register (MIO_UART(0..1)_DLL and MIO_UART(0..1)_DLH) to set the baud rate of the UART. This bit must be cleared after initial baud-rate setup in order to access other registers.
<6>	BRK	R/W	0	—	Break control bit. Setting this bit when not in loopback mode (i.e. MIO_UART(0..1)_MCR[LOOP] = 0) sends a break signal by holding the UART0/1_SOUT line low. When in loopback mode, the break condition is internally looped back to the receiver.
<5>	—	RAZ	—	—	Reserved.
<4>	EPS	R/W	0	—	Even parity select bit. Selects between even and odd parity. 1 = an even number of ones is transmitted or checked 0 = an odd number of ones is transmitted or checked
<3>	PEN	R/W	0	—	Parity enable bit. Enables and disables parity generation and detection in transmitted and received serial character respectively.
<2>	STOP	R/W	0	—	Stop control bit. Controls the number of stop bits transmitted. 0 = one stop bit is transmitted in the serial data. 1 = two stop bits are generated and transmitted in the serial data out, UNLESS CLS = 00, then one and a half stop bits are generated. Note that, regardless of the number of stop bits selected, the receiver only checks the first stop bit.
<1:0>	CLS	R/W	0x0	—	Character length select field. Selects the number of data bits per character that are transmitted and received. 00 = 5 bits (bits 0-4 sent) 01 = 6 bits (bits 0-5 sent) 10 = 7 bits (bits 0-6 sent) 11 = 8 bits (all bits sent)

1. The LCR is writable only when the UART is not busy (i.e. when MIO_UART0/1_USR[BUSY] = 0). The LCR is always readable.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO UART0/1 Modem-Control Register MIO_UART(0..1)_MCR

The lower four bits of the modem-control register directly manipulate the outputs of the UART. The DTR, RTS, OUT1, and OUT2 bits are inverted and then drive the corresponding UART outputs: dtr_n, rts_n (UART(0..1)_RTS_L), out1_n, and out2_n.

See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved.
<7:6>	—	RAZ	—	—	Reserved.
<5>	AFCE	R/W	0	—	Autoflow-control enable (AFCE) bit. When FIFOs are enabled and this bit is set, 16750-compatible autoRTS and autoCTS serial data flow control features are enabled.
<4>	LOOP	R/W	0	—	Loopback bit. When set, data on the UART0/1_SOUT line is held high, while serial data output is looped back to the UART0/1_SIN line, internally. In this mode all the interrupts are fully functional. This feature is used for diagnostic purposes. Also, in loopback mode, the modem-control inputs (dsr_n, cts_n, ri_n, dcd_n) are disconnected and the four modem-control outputs (dtr_n, rts_n, out1_n, out1_n) are looped back to the inputs, internally
<3>	OUT2	R/W	0	—	OUT2 output bit. This bit is inverted and drives the out2_n signal
<2>	OUT1	R/W	0	—	OUT1 output bit. This bit is inverted and drives the out1_n signal
<1>	RTS	R/W	0	—	Request to send output bit. This bit is inverted and drives the rts_n signal.[1]
<0>	DTR	R/W	0	—	Data terminal ready output bit. This bit is inverted and drives the dtr_n signal.

1. When autoRTS is enabled, the rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold). The rts_n output is deasserted whenever RTS = 0.

AutoRTS becomes active when the following occurs:

1. MIO_UART(0..1)_MCR[RTS] is set.
2. FIFOs are enabled by setting MIO_UART(0..1)_FCR[EN].
3. MIO_UART(0..1)_MCR[AFCE] is set (must be set after MIO_UART(0..1)_FCR[EN])

When autoRTS is active, the rts_n output is forced inactive high when the receiver FIFO level reaches the threshold set by MIO_UART(0..1)_FCR[RXTRIG]. When rts_n is connected to the cts_n input of another UART device, the other UART stops sending serial data until the receiver FIFO has available space.

The selectable receiver FIFO threshold values are:

- 1
- ¼
- ½
- 2-less-than-full.

Since one additional character may be transmitted to the UART after rts_n has become inactive (due to data already having entered the transmitter block in the other UART), setting the threshold to 2-less-than-full allows maximum use of the FIFO with a safety zone of one character.

Cavium Confidential For Bowen Yang ALU 5/27/2014

Once the receiver FIFO becomes completely empty by reading the MIO_UART(0..1)_RBR, rts_n again becomes active low, signaling the other UART to continue sending data.

It is important to note that, even if everything else is set to enabled and the correct MIO_UART(0..1)_MCR bits are set: if the FIFOs are disabled through MIO_UART(0..1)_FCR[EN], autoflow control is also disabled. When autoRTS is disabled or inactive, rts_n is controlled solely by MIO_UART(0..1)_MCR[RTS].

AutoCTS becomes active when the following occurs:

1. FIFOs are enabled by setting MIO_UART(0..1)_FCR[EN].
2. MIO_UART(0..1)_MCR[AFCE] is set (must be set after MIO_UART(0..1)_FCR[EN])

When active, the UART transmitter is disabled whenever the cts_n input becomes inactive high. This prevents overflowing the FIFO of the receiving UART.

Note that if the cts_n input is not inactivated before the middle of the last stop bit, another character is transmitted before the transmitter is disabled. While the transmitter is disabled, the transmitter FIFO can still be written to, and even overflowed. Therefore, when using this mode, either the true FIFO depth (64 characters) must be known to software, or the programmable THRE interrupt mode must be enabled to access the FIFO-full status through the MIO_UART(0..1)_LSR. When using the FIFO-full status, software can poll this before each write to the transmitter FIFO.

NOTE: FIFO-full status is also available in the MIO_UART0/1_USR register, or the actual level of the FIFO may be read through the MIO_UART0/1_TFL register.

When the cts_n input becomes active low again, transmission resumes. It is important to note that, even if everything else is set to enabled, autoflow control is also disabled if the FIFOs are disabled through MIO_UART(0..1)_FCR[EN]. When autoCTS is disabled or inactive, the transmitter is unaffected by cts_n.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO UART0/1 Line Status Register MIO_UART(0..1)_LSR

The line status register contains status of the receiver and transmitter data transfers. This status can be read by the user at anytime.

See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved.
<7>	FERR	RC	0	—	Error in receiver FIFO bit. This bit is active only when FIFOs are enabled. It is set when there is at least one parity error, framing error, or break indication in the FIFO. This bit is cleared when MIO_UART0/1_LSR is read and the character with the error is at the top of the receiver FIFO and there are no subsequent errors in the FIFO.
<6>	TEMT	RO/H	1	—	Transmitter empty bit. In FIFO mode, this bit is set whenever MIO_UART(0..1)_TSR and the FIFO are both empty. In nonFIFO mode, this bit is set whenever MIO_UART(0..1)_THR and MIO_UART(0..1)_TSR are both empty. This bit is typically used to make sure it is safe to change control registers. Changing control registers while the transmitter is busy can result in corrupt data being transmitted.
<5>	THRE	RO/H	1	—	Transmitter holding register empty bit. When programmable THRE interrupt mode is disabled, this bit indicates that the UART can accept a new character for transmission. This bit is set whenever data is transferred from MIO_UART(0..1)_THR (or TX FIFO) to the transmitter shift register and no new data has been written to MIO_UART(0..1)_THR (or TX FIFO). This also causes a THRE interrupt to occur, if the THRE interrupt is enabled. When FIFOs and programmable THRE Interrupt mode are enabled, this bit's functionality is switched to indicate the transmitter FIFO is full, and no longer controls THRE Interrupts, which are then controlled by the MIO_UART(0..1)_FCR[TXTRIG] threshold setting.
<4>	BI	RC	0	—	Break interrupt bit. This bit is set whenever the serial input (sin) is held in a 0 state for longer than the sum of start time + data bits + parity + stop bits. A break condition on sin causes one and only one character, consisting of all zeros, to be received by the UART. In the FIFO mode, the character associated with the break condition is carried through the FIFO and is revealed when the character is at the top of the FIFO. Reading the LSR clears the BI bit. In the non-FIFO mode, the BI indication occurs immediately and persists until the LSR is read.
<3>	FE[1]	RC	0	—	Framing error bit. This bit is set whenever there is a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data. In FIFO mode, since the framing error is associated with a character received, it is revealed when the character with the framing error is at the top of the FIFO. The framing error bit will be set if a break interrupt has occurred.
<2>	PE[1]	RC	0	—	Parity error bit. This bit is set whenever there is a parity error in the receiver if the parity-enable bit (MIO_UART(0..1)_LCR[PEN]) is set. In FIFO mode, since the parity error is associated with a character received, it is revealed when the character with the parity error arrives at the top of the FIFO. The parity error bit will be set if a break interrupt has occurred.

Cavium Confidential For Beta Testing Only

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<1>	OE[1]	RC	0	—	Overrun error bit. When set, this bit indicates an overrun error has occurred because a new data character was received before the previous data was read. In nonFIFO mode, this bit is set when a new character arrives in the receiver before the previous character was read from MIO_UART0(0..1)_RBR. When this happens, the data in MIO_UART(0..1)_RBR is overwritten. In FIFO mode, an overrun error occurs when the FIFO is full and a new character arrives at the receiver. The data in the FIFO is retained and the data in the receive shift register is lost.
<0>	DR	RO/H	0	—	Data ready bit. When set, this bit indicates the receiver contains at least one character in the MIO_UART(0..1)_RBR or the receiver FIFO. This bit is cleared when the MIO_UART(0..1)_RBR is read in nonFIFO mode, or when the receiver FIFO is empty in FIFO mode.

1. The OE, PE, and FE bits are reset when a read of the MIO_UART0/1_LSR is performed.

MIO UART0/1 Modem Status Register MIO_UART(0..1)_MSR

The modem status register contains the current status of the modem-control input lines and if they changed. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
63:8>	—	RAZ	—	—	Reserved.
<7>	DCD	RO	0	—	Data-carrier-detect input bit. This bit is the complement of the modem-control line dcd_n. In loopback mode, this bit is the same as MIO_UART(0..1)_MCR[OUT2].
<6>	RI	RO	0	—	Ring indicator input bit. This bit is the complement of the modem-control line ri_n. In loopback mode, this bit is the same as MIO_UART(0..1)_MCR[OUT1].
<5>	DSR	RO	0	—	Data-set-ready input bit. This bit is the complement of the modem-control line dsr_n. In loopback mode, this bit is the same as MIO_UART(0..1)_MCR[DTR].
<4>	CTS	RO	—	—	Clear-to-send input bit. This bit is the complement of the modem-control line cts_n. In loopback mode, this bit is the same as MIO_UART(0..1)_MCR[RTS].
<3>	DDCD	RC	0	—	Delta data-carrier-detect. Records whether the dcd_n line has changed since the last time the user read the MIO_UART(0..1)_MSR. In loopback mode, reflects changes in MIO_UART(0..1)_MCR[OUT2].
<2>	TERI	RC	0	—	Trailing edge of ring indicator. Indicates ri_n has changed from an active-low to an inactive-high state since the last time the user read the MIO_UART(0..1)_MSR. In loopback mode, reflects state changes from high to low in MIO_UART(0..1)_MCR[OUT1].
<1>	DDSR	RC	0	—	Delta data-set-ready. Records whether the dsr_n line has changed since the last time the user read the MIO_UART(0..1)_MSR. In loopback mode, reflects changes in MIO_UART(0..1)_MCR[DTR].
<0>	DCTS	RC	0	—	Delta clear-to-send. Records whether the cts_n line has changed since the last time the user read the MIO_UART(0..1)_MSR. In loopback mode, reflects changes in MIO_UART(0..1)_MCR[RTS].

Cavium Confidential For Boren Yang 07/21/2014

MIO UART0/1 Scratchpad Register MIO_UART(0..1)_SCR

The scratchpad register is an 8-bit read/write register for programmers to use as a temporary storage space. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved.
<7:0>	SCR	R/W	0x0	—	Scratchpad register

MIO UART0/1 Transmit Holding Register MIO_UART(0..1)_THR

The transmit holding register is a write-only register that contains data to be transmitted on the serial output port (UART(0..1)_SOUT). Data can be written to MIO_UART(0..1)_THR any time that MIO_UART(0..1)_LSR[THRE] = 1. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	WO	—	—	Reserved.
<7:0>	THR	WO	0x0	—	Transmit holding register.

- If FIFOs are not enabled and MIO_UART(0..1)_LSR[THRE] = 1, writing a single character to MIO_UART(0..1)_THR clears MIO_UART(0..1)_LSR[THRE]. Any additional write operations to MIO_UART(0..1)_THR before MIO_UART(0..1)_LSR[THRE] is set again causes MIO_UART(0..1)_THR data to be overwritten.
- If FIFOs are enabled and MIO_UART(0..1)_LSR[THRE] = 1, and programmable THRE mode is disabled (i.e. MIO_UART(0..1)_IER[PTIME] = 0), 64 characters of data may be written to MIO_UART(0..1)_THR before the FIFO is full. Any attempt to write data when the FIFO is full results in the write data being lost.

NOTE: MIO_UART(0..1)_LCR[DLAB] must be clear to access this register.

The address in Table 34–2 is an alias to simplify these CSR descriptions. The MIO_UART(0..1)_THR, MIO_UART(0..1)_RBR, and MIO_UART(0..1)_DLL registers are one register with different uses at different times.

Cavium Confidential For Wen Tang 02/27/2014 ALL

MIO UART0/1 FIFO Control Register MIO_UART(0..1)_FCR

The FIFO control register is a write-only register that controls the read- and write-data FIFO operation. When FIFOs and programmable-THRE-interrupt mode are enabled, this register also controls the THRE interrupt-empty threshold level. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved.
<7:6>	RXTRIG	WO	0x0	—	Receive trigger. If FIFOs are enabled (i.e. EN = 1), this field is active and set the trigger level in the receiver FIFO for the enable received-data-available interrupt (ERBFI). In autoflow-control mode, the trigger is used to determine when the rts_n signal will be deasserted. The trigger values are: 00 = 1 character in FIFO 01 = FIFO ¼ full 10 = FIFO ½ full 11 = FIFO is two characters less than full
<5:4>	TXTRIG	WO	0x0	—	Transmit trigger. If the FIFOs and programmable-THRE-interrupt mode are enabled, the values in this field control the empty threshold level at which THRE interrupts are generated when the mode is active. 00 = empty FIFO 01 = 2 characters in FIFO 10 = FIFO ¼ full 11 = FIFO ½ full
<3>	—	RAZ	—	—	Reserved.
<2>	TXFR	WO	0	—	TX FIFO reset. Writing a 1 to this bit resets and flushes data in the transmit FIFO. This bit is self-clearing, so it is not necessary to clear this bit.
<1>	RXFR	WO	0	—	RX FIFO reset. Writing a 1 to this bit resets and flushes data in the receive FIFO. This bit is self-clearing, so it is not necessary to clear this bit.
<0>	EN	WO	0	—	FIFO enable. Writing a 1 to this bit enables the transmit and receive FIFOs. Whenever the value of this bit is changed both the TX and RX FIFOs are reset.

NOTE: The address in Table 34–2 is an alias to simplify these CSR descriptions. The MIO_UART(0..1)_FCR and MIO_UART(0..1)_IIR registers are one register with different uses at different times.

Cavium Confidential For Bowen Yang 02/21/2014

MIO UART0/1 Divisor Latch Low Register MIO_UART(0..1)_DLL

The 8-bit divisor latch high register in conjunction with the 8-bit divisor latch low (MIO_UART(0..1)_DLL) register form a 16-bit, read/write, Divisor Latch register that contains the baud-rate divisor for the UART. It is accessed by first setting MIO_UART(0..1)_LCR[DLAB] (bit 7) (refer to "MIO UART0/1 Line Control Register" on page 1505). The output baud rate is equal to the coprocessor-clock frequency divided by sixteen times the value of the baud-rate divisor, as follows:

baud rate = coprocessor-clock frequency / (16 × divisor).

Note that once both divisor latch registers are set, at least eight coprocessor-clock cycles should be allowed to pass before transmitting or receiving data.

See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved.
<7:0>	DLL	R/W	0x0	—	Divisor latch low register

NOTE: If the Divisor Latch registers (DLL and DLH) set to 0x0, the baud clock is disabled and no serial communications will occur.

The address in Table 34–2 is an alias to simplify these CSR descriptions. The MIO_UART(0..1)_DLL, MIO_UART(0..1)_RBR, and MIO_UART(0..1)_THR registers are one register with different uses at different times.

MIO UART0/1 Divisor Latch High Register MIO_UART(0..1)_DLH

The 8-bit divisor latch high register in conjunction with the 8-bit divisor latch low (MIO_UART0/1_DLL) register form a 16-bit, read/write, Divisor Latch register that contains the baud-rate divisor for the UART. It is accessed by first setting MIO_UART0/1_LCR[DLAB] (bit 7) (refer to "MIO UART0/1 Line Control Register" on page 1505). The output baud rate is equal to the coprocessor-clock frequency divided by sixteen times the value of the baud-rate divisor, as follows:

baud rate = coprocessor-clock frequency / (16 × divisor).

Note that once both divisor latch registers are set, at least eight coprocessor-clock cycles should be allowed to pass before transmitting or receiving data.

See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved.
<7:0>	DLH	R/W	0x0	—	Divisor latch high register

NOTE: If the Divisor Latch registers (DLL and DLH) set to 0x0, the baud clock is disabled and no serial communications will occur.

The address in Table 34–2 is an alias to simplify these CSR descriptions. The MIO_UART(0..1)_IER and MIO_UART(0..1)_DLH registers are one register with different uses at different times.

Cavium Confidential For Bowen Yang ATI 2/27/2014

MIO UART0/1 FIFO Access Register
MIO_UART(0..1)_FAR

The FIFO access register is used to enable a FIFO-access mode for testing, so that the receive FIFO can be written by software and the transmit FIFO can be read by software when the FIFOs are enabled. When FIFOs are not enabled it allows the MIO_UART(0..1)_RBR to be written by software and the MIO_UART(0..1)_THR to be read by software. Note, that when the FIFO-access mode is enabled/disabled, the control portion of the receive FIFO and transmit FIFO is reset and the FIFOs are treated as empty. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	FAR	R/W	0	—	FIFO-access mode enable.

MIO UART0/1 Transmit FIFO Read Register
MIO_UART(0..1)_TFR

The transmit FIFO read register is only valid when FIFO-access mode is enabled (i.e. MIO_UART(0..1)_FAR[FAR] = 1). See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved.
<7:0>	TFR	RO	0x0	—	Transmit FIFO data. When FIFOs are enabled, this field gives the data at the top of the transmit FIFO. Each consecutive read pops the transmit FIFO and gives the next data value that is currently at the top of the FIFO. When FIFOs are not enabled, this field gives the data in MIO_UART(0..1)_THR.

MIO UART0/1 Receive FIFO Write Register
MIO_UART(0..1)_RFW

The receive FIFO write register is only valid when FIFO-access mode is enabled (i.e. MIO_UART(0..1)_FAR[FAR] = 1). See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:10>	—	RAZ	—	—	Reserved.
<9>	RFFE	WO	0	—	Receive FIFO framing error.
<8>	RFPE	WO	0	—	Receive FIFO parity error.
<7:0>	RFWD	WO	0x0	—	Receive FIFO write data. When FIFOs are enabled, this field provides write data to the receive FIFO. Each consecutive write pushes the new data to the next write location in the receive FIFO. When FIFOs are not enabled, this field provides write data to the MIO_UART0/1_RBR.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO UART0/1 UART Status Register
MIO_UART(0..1)_USR

The receive FIFO write register is only valid when FIFO-access mode is enabled (i.e. MIO_UART(0..1)_FAR[FAR] = 1). See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:5>	—	RAZ	—	—	Reserved.
<4>	RFF	RO	0	—	RX FIFO full.
<3>	RFNE	RO	0	—	RX FIFO not empty.
<2>	TFE	RO	1	—	TX FIFO empty.
<1>	TFNF	RO	1	—	TX FIFO not full.
<0>	BUSY	RO	0	—	Busy. This bit is always clear.

MIO UART0/1 Transmit FIFO Level Register
MIO_UART(0..1)_TFL

The transmit FIFO level register indicates the number of data entries in the transmit FIFO. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved.
<6:0>	TFL	RO	0x0	—	Transmit FIFO level. Indicates the number of data entries in the transmit FIFO.

MIO UART0/1 Receive FIFO Level Register
MIO_UART(0..1)_RFL

The receive FIFO level register indicates the number of data entries in the receive FIFO. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:7>	—	RAZ	—	—	Reserved.
<6:0>	RFL	RO	0x0	—	Receive FIFO level. Indicates the number of data entries in the receive FIFO.

Cavium Confidential For Bowen Yang ALU 02/27/2014

MIO UART0/1 Software Reset Register
MIO_UART(0..1)_SRR

The software reset register is a write-only register that resets the UART and/or the receive FIFO and/or the transmit FIFO. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:3>	—	RAZ	—	—	Reserved.
<2>	STFR	WO	0	—	Shadow copy of the TX FIFO reset bit (MIO_UART(0..1)_FCR[TXFR]). This bit can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the transmit FIFO
<1>	SRFR	WO	0	—	Shadow copy of the RX FIFO reset bit (MIO_UART(0..1)_FCR[RXFR]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the receive FIFO.
<0>	USR	WO	0	—	UART soft reset. Setting this bit resets the UART.

MIO UART0/1 Shadow Request-to-Send Register
MIO_UART(0..1)_SRTS

The shadow request to send register is a shadow register for the MIO_UART(0..1)_MCR[RTS] bit that can be used to remove the burden of having to perform a read-modify-write on MIO_UART(0..1)_MCR. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	SRTS	R/W	0	—	Shadow request to send.

MIO UART0/1 Shadow Break-Control Register
MIO_UART(0..1)_SBCR

The shadow break control register is a shadow register for the MIO_UART(0..1)_LCR[BREAK] bit that can be used to remove the burden of having to perform a read-modify-write on MIO_UART(0..1)_LCR. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	SBCR	R/W	0	—	Shadow break control.

Cavium Confidential For Bowen Yang AU 02/27/2014

MIO UART0/1 Shadow FIFO-Enable Register MIO_UART(0..1)_SFE

The shadow FIFO enable register is a shadow register for MIO_UART(0..1)_FCR[EN] that can be used to remove the burden of having to store the previously written value to MIO_UART(0..1)_FCR in memory and having to mask this value so that only the FIFO enable bit gets updated. See Table 34–2 for the address.

Bit Pos	Field Name	Field Type	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	SFE	R/W	0	—	Shadow FIFO enable.

MIO UART0/1 Shadow RX Trigger Register MIO_UART(0..1)_SRT

The shadow RX trigger register is a shadow register for the RX trigger bits (MIO_UART(0..1)_FCR[RXTRIG]) that can be used to remove the burden of having to store the previously written value to MIO_UART(0..1)_FCR in memory and having to mask this value so that only the RX trigger bits get updated. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved.
<1:0>	SRT	R/W	0x0	—	Shadow RX trigger.

MIO UART0/1 Shadow TX Trigger Register MIO_UART(0..1)_STT

The shadow TX trigger register is a shadow register for the TX trigger bits (MIO_UART(0..1)_FCR[TXTRIG]) that can be used to remove the burden of having to store the previously written value to MIO_UART(0..1)_FCR in memory and having to mask this value so that only the TX trigger bits get updated. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:2>	—	RAZ	—	—	Reserved.
<1:0>	STT	R/W	0x0	—	Shadow TX trigger.

MIO UART0/1 Halt TX Register MIO_UART(0..1)_HTX

The halt TX register is used to halt transmissions for testing, so that the transmit FIFO can be filled by software when FIFOs are enabled. If FIFOs are not enabled, setting the HTX register will have no effect. See Table 34–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:1>	—	RAZ	—	—	Reserved.
<0>	HTX	R/W	0	—	Halt TX.

Cavium Confidential For Bowen Yang At 02/27/2014

Chapter 35

TWSI Interface

This chapter contains the following subjects:

- Overview
- CN71XX High-Level Controller as a Master
- CN71XX High-Level Controller as a Slave
- Direct TWSI Core Usage
- TWSI Control Registers
- TWSI Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The CN71XX has two two-wire serial interfaces (TWSIs). Each supports a standard suite of features, including the following:

- Standard two-wire SCL/SDA protocols.
- Standard mode (100 Kbps) and fast mode (400 Kbps) support.
- 7-bit and 10-bit addressing.
- Master and slave support.
  - TWSI masters initiate operations on the TWSI interface
  - TWSI slaves only respond to operations initiated by TWSI masters
- Multi-master support.
- Clock stretching.
- Flexible bus reset support. (Software can drive arbitrary patterns on SCL/SDA to create any required reset sequence.)

Figure 35–1 shows the architecture of CN71XX's TWSI logic. It consists of a TWSI core that implements some low-level details of the protocol, together with a high-level controller (HLC) to implement various multibyte sequences. TWSI transactions may be generated by either directing the HLC or directing the low-level details of the TWSI core. It is possible for either cores or a remote host to communicate with a TWSI device, though we assume only the cores for the remainder of this section. The TWSI HLC is enabled when TWSI_CTL[CE] is set (see Section 35.4.4).

Figure 35–1 TWSI Architecture

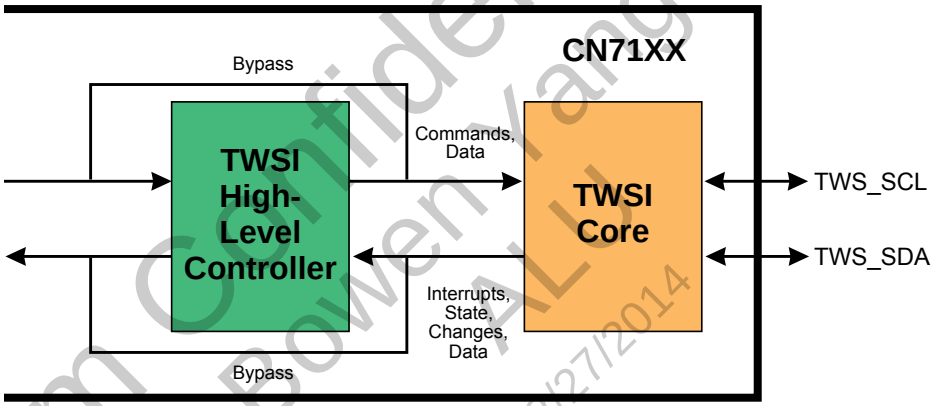
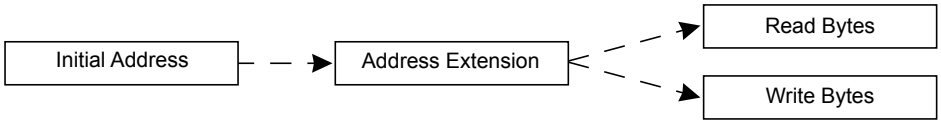


Figure 35–2 displays the steps of the sequences supported by the TWSI HLC. The initial address identifies the device that will be the TWSI slave. CN71XX supports both 7-bit and 10-bit initial addresses. The address extension carries an optional internal address for the identified slave, and also completes the initial address for a read with a 10-bit address. CN71XX optionally supports an internal address of 8 or 16 bits. Finally, the read bytes or write bytes complete the sequence. CN71XX supports between zero and eight byte read/write sequences.

Figure 35–2 Supported TWS HLC Transaction Steps



Cavium Confidential For Bowen Yang ALU 02/27/2014

CN71XX can communicate with another TWSI device as either master or slave. The steps in Figure 35–2 apply in both cases, though the details of each differ.

35.1 CN71XX High-Level Controller as a Master

CN71XX cores become master of HLC TWSI sequences by writing the MIO_TWS(0..1)_SW_TWSI register. CN71XX's HLC is in master mode when MIO_TWS_SW_TWSI[SLONLY] = 0 and both the TWSI_CTL[CE, AAK] bits are set to 1 (see Section 35.4.4).

- If the sequence is a write, the MIO_TWS_SW_TWSI[D] field (and MIO_TWS(0..1)_SW_TWSI_EXT[D] if the write sequence is more than four bytes) contains the write data.
- If the sequence is a read, the MIO_TWS_SW_TWSI[D] field (and MIO_TWS_SW_TWSI_EXT[D] if the read sequence is more than four bytes) contains the read result after the HLC completes the transaction.

The cores either poll MIO_TWS_SW_TWSI or use interrupts to determine when the sequence is complete. CN71XX clears the MIO_TWS_SW_TWSI[V] bit and sets the MIO_TWS(0..1)_INT[ST_INT] bit when the HLC completes the transaction.

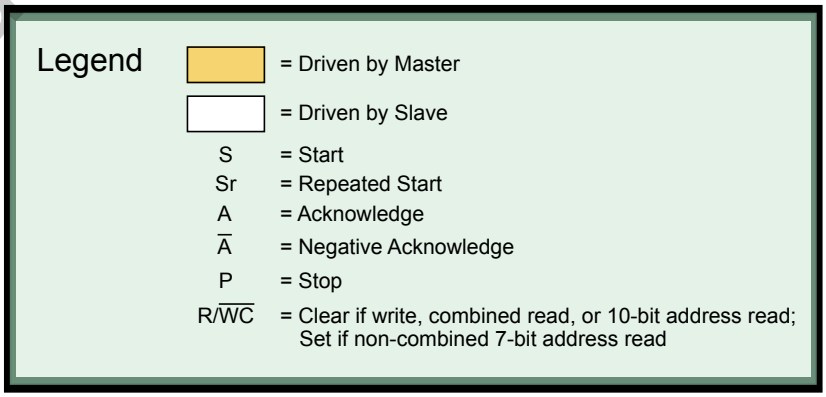
If the mastered high-level sequence has an error, the MIO_TWS_SW_TWSI[R] bit is set to 0 and MIO_TWS_SW_TWSI[D<7:0>] contains an error code. In the case of an error, the TWSI high-level state controller aborts the operation on the TWSI bus and resets its internal state.

Refer to the "TWSI Status Register" on page 1537. The error code is an unexpected TWSI core state found by the HLC.

- An example of an error would be **a NACK where an ACK was expected** by the HLC (0x20, 0x30, 0x48, 0xD8).
- Another example is where the **HLC loses arbitration** (0x38, 0x68, 0xB0, 0x78).
- A final error example is if the **master-mode operation attempts to initiate while the TWSI HLC is servicing a TWSI operation as a slave** (0x80, 0x88, 0xA0, 0xA8, 0xB8, 0xC0, 0xC8).

Figures 35–4 through 35–10 show some of the CN71XX-TWSI sequences (as defined in Figure 35–2). Figure 35–3 provides a legend for the symbols used in these examples.

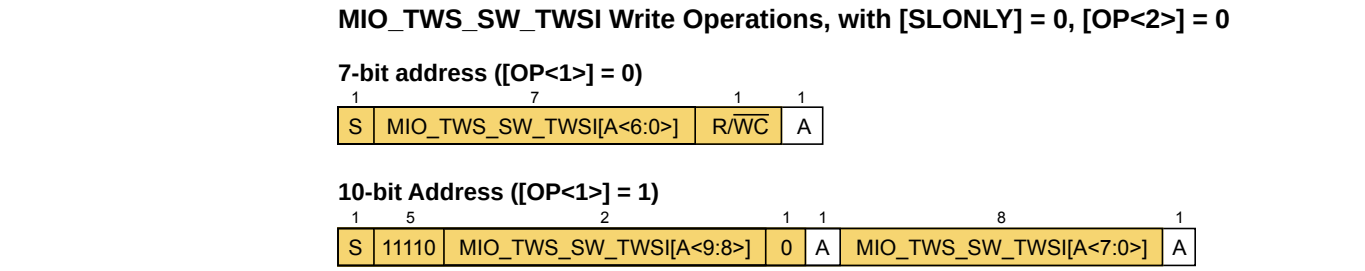
Figure 35–3 Legend for Diagrams



Cavium Confidential For Beiwei Yang ALU 02/27/2014

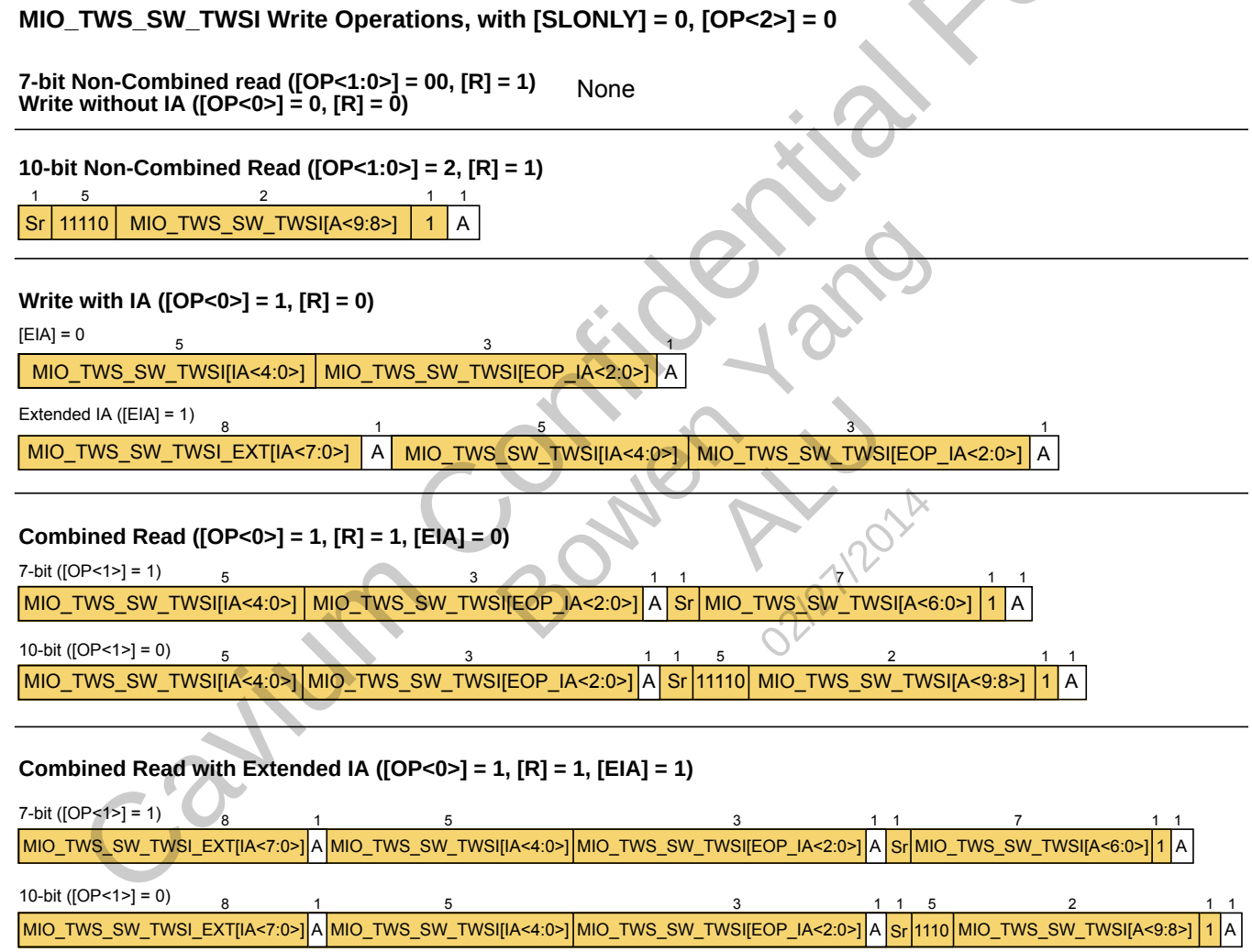
Figure 35–4 shows the possible initial-address steps for sequences mastered by the CN71XX HLC. MIO_TWS_SW_TWSI[OP<1>] selects whether a 7-bit or 10-bit address is used, and MIO_TWS_SW_TWSI[A] contains the address of the remote slave.

Figure 35–4 HLC as Master, Initial-Address Step



All possible address-extension steps for sequences mastered by the HLC are shown in Figure 35–5.

Figure 35–5 HLC as Master, Address-Extension Step

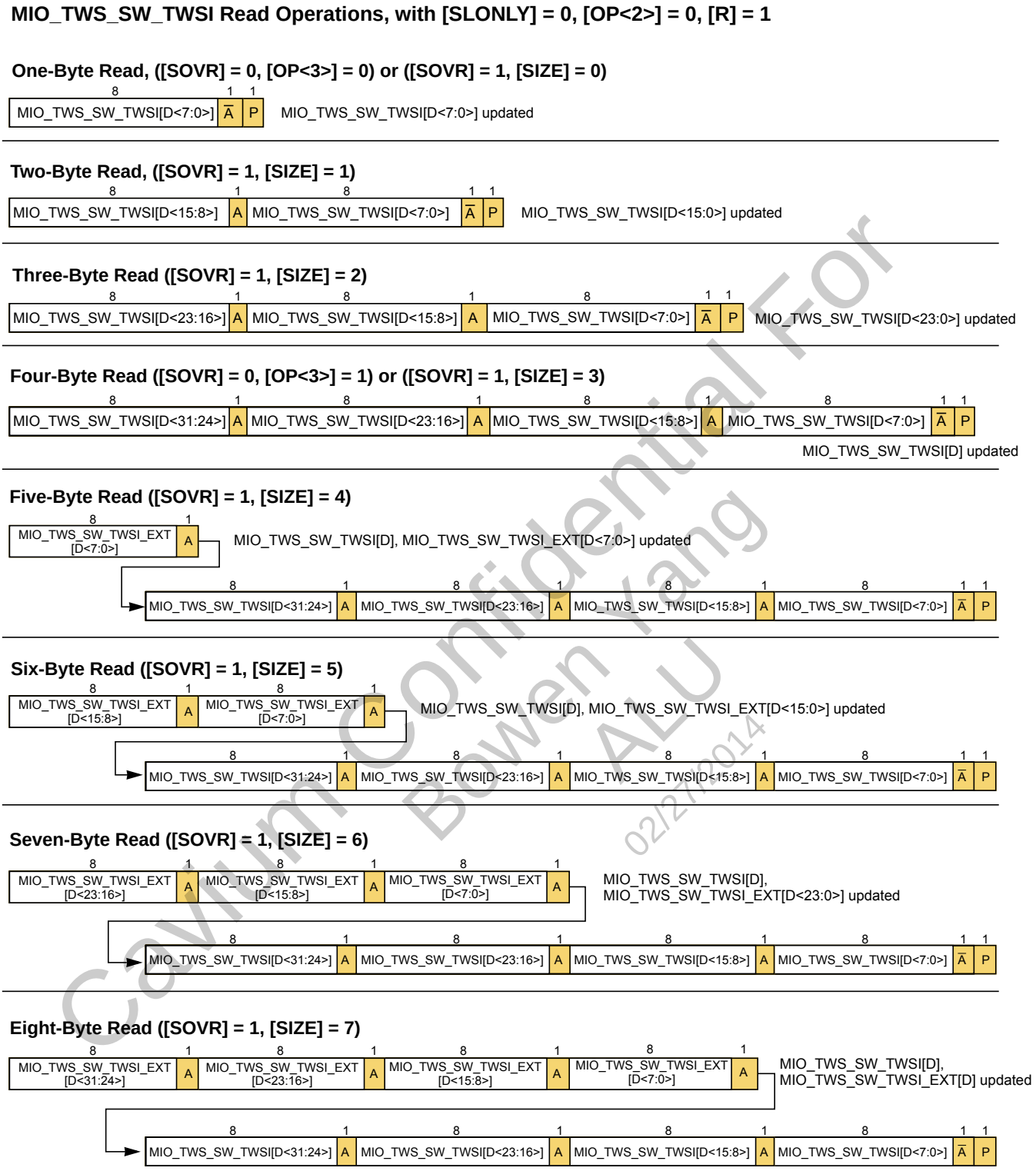


MIO_TWS_SW_TWSI[IA,EOP_IA] and MIO_TWS_SW_TWSI_EXT[IA] contain the internal address to be used by the external slave. The address extension component is empty for 7-bit address non-combined read operations and for write operations without internal addresses.

Cavium Confidential For Bowen Yang ALU 02/17/2014

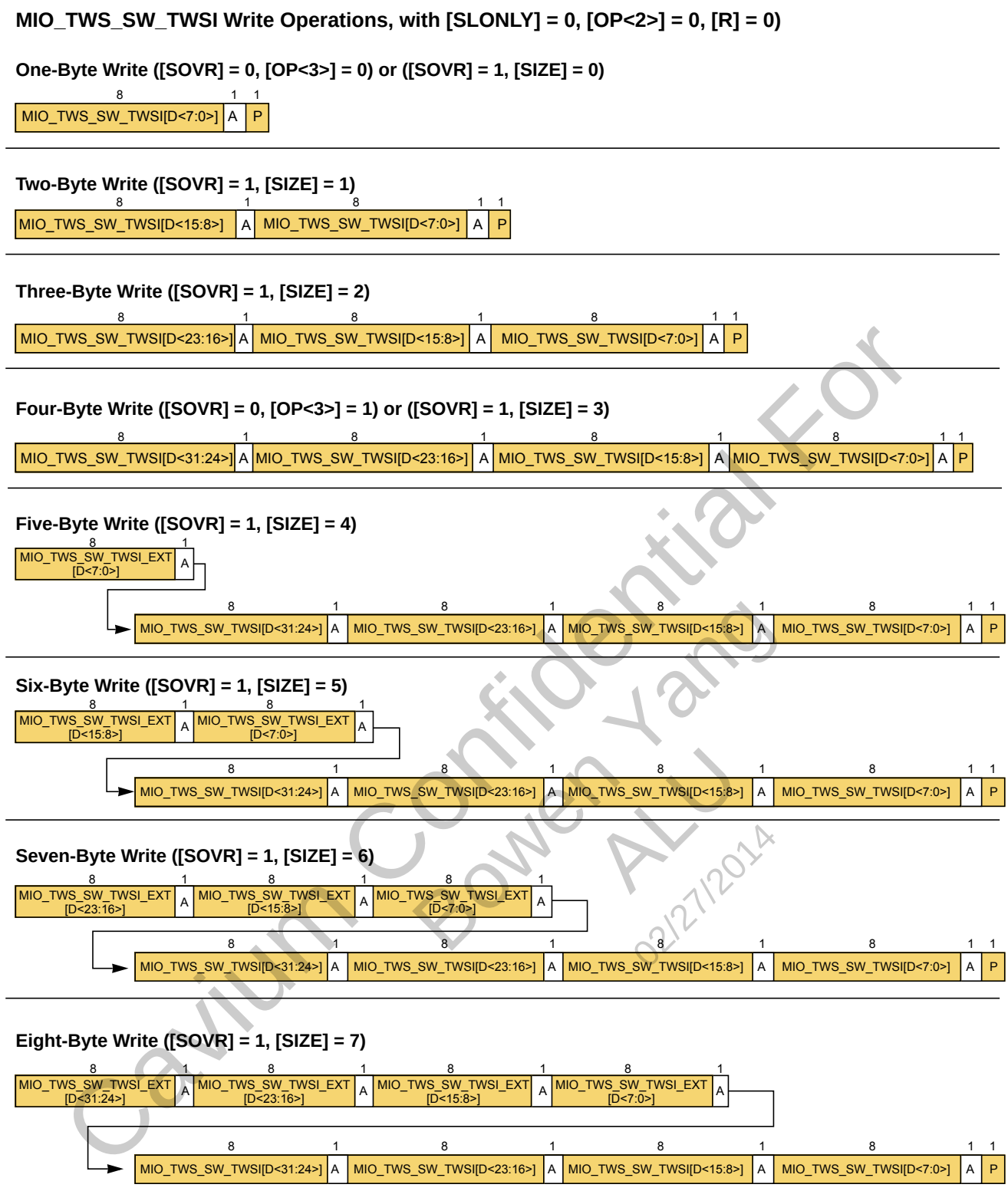
Figure 35–6 shows all possible read-bytes steps for sequences mastered by the CN71XX high-level controller, while Figure 35–7 shows all possible write-bytes steps for sequences mastered by the high-level controller. CN71XX can master any size from one to eight bytes.

Figure 35–6 HLC as Master, Read-Bytes Step



Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 35–7 HLC as Master, Write-Bytes Step



Cavium Confidential For Bowen Yang ALU 02/27/2014

35.2 CN71XX High-Level Controller as a Slave

CN71XX cores can act as a slave to communicate with another TWSI device via the MIO_TWS_SW_TWSI and MIO_TWS_TWSI_SW registers and the TWSI HLC. CN71XX's HLC is in slave mode when MIO_TWS_SW_TWSI[SLONLY] is set to 1 and both TWSI_CTL[CE,AAK] are set to 1.

- To transfer up to four bytes from a core to the TWSI device, with CN71XX's HLC acting as a slave:
  - The core writes the bytes into MIO_TWS_SW_TWSI, setting the MIO_TWS_SW_TWSI[V] bit.
  - The remote TWSI device reads MIO_TWS_SW_TWSI, retrieving the data and its corresponding vbyte. Vbyte<7> is set if MIO_TWS_SW_TWSI[V] is set, and the read clears MIO_TWS_SW_TWSI[V].
  - The core may either poll MIO_TWS_SW_TWSI[V] or wait for an interrupt (MIO_TWS_INT[ST_INT]) to determine when the data has been received, and when more data can be written into MIO_TWS_SW_TWSI.
- To transfer up to four bytes from a TWSI device to a core, with CN71XX's HLC acting as a slave:
  - The remote TWSI device writes MIO_TWS_TWSI_SW, setting MIO_TWS_TWSI_SW[V].
  - The core either receives an interrupt (via MIO_TWS_INT[TS_INT]) or polls MIO_TWS_TWSI_SW[V] to determine when the bytes are present.
  - The core reads MIO_TWS_TWSI_SW, clearing MIO_TWS_TWSI_SW[V].
  - The remote TWSI device polls MIO_TWS_TWSI_SW to determine when it can write it again. Vbyte<7> is clear when MIO_TWS_TWSI_SW[V] is clear.

Figure 35–8 shows the possible initial-address steps (as shown in Figure 35–2) supported for sequences with the HLC as a slave. The TWSI core TWSI slave address register and TWSI slave extended address register contain the initial address that CN71XX will respond to.

Figure 35–8 HLC as Slave, Initial-Address Step

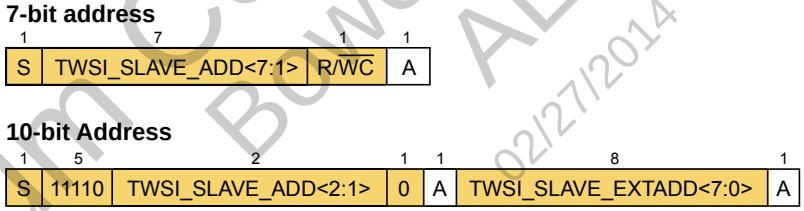


Figure 35–9 shows the possible address-extension steps supported for sequences with the HLC as a slave. Table 35–1 provides the format of CN71XX's slave internal-address (SIA) register. SIA selects between the MIO_TWS_SW_TWSI and MIO_TWS_TWSI_SW registers. The external TWSI device must use the SIA register.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 35–9 HLC as Slave, Address-Extension Step

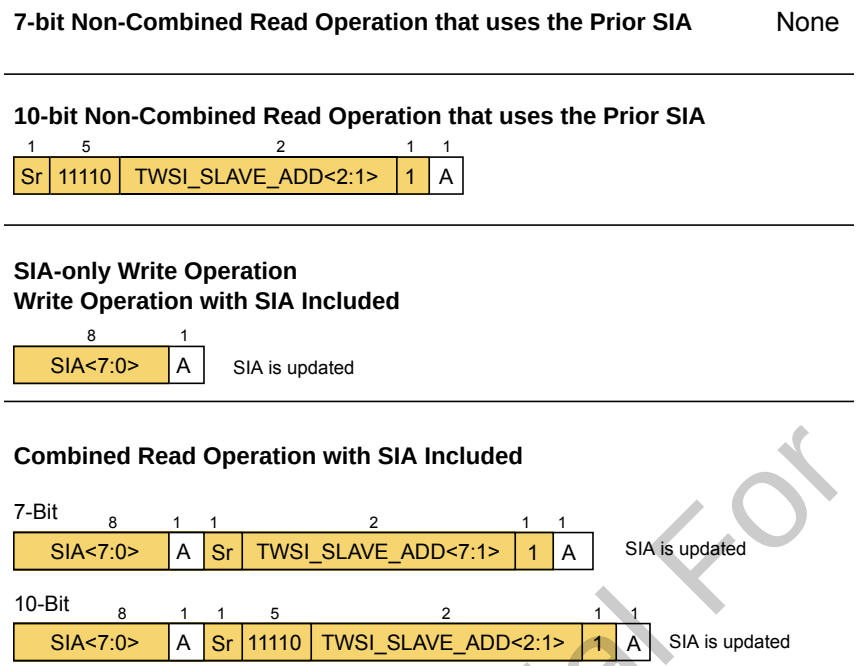


Table 35–1 Slave Internal-Address Register

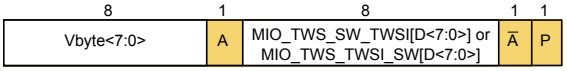
Bit Pos	Field Name	Description
<7:4>	—	Reserved
<3>	SIZE	Size of the data access: 0 = 8-bit access, 1 = 32-bit access. Must be 0 for an MIO_TWS_TWSI_SW read operation.
<2:1>	—	Reserved
<0>	REG	Identifies the register to be accessed: 0 = MIO_TWS_SW_TWSI register (read-only) 1 = MIO_TWS_TWSI_SW register (read/write)

Figure 35–10 shows the possible read and write bytes components supported for sequences with the HLC as a slave. SIA<3> (and the stop condition in the case of a write) allows the external TWSI device to vary the amount of data transferred in the subsequent read or write when N2 acts as a slave. For read operations, it can be one or four bytes, plus one more byte for the Vbyte. For write operations, it can be 0, 1, or four bytes. Note that the 8-bit write to MIO_TWS_TWSI_SW is equivalent to a 4-byte write with 0s for the first 3 bytes.

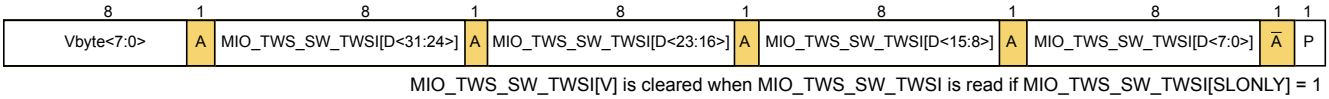
Cavium Confidential For Bowen Yang 12/12/2014

Figure 35–10 HLC as Slave, Read/Write-Bytes Step

One-Byte Read Operation, (SIA<3> = 0), (MIO_TWS_SW_TWSI[SIA<0>] = 0), or (MIO_TWS_TWSI_SW[SIA<0>] = 1)

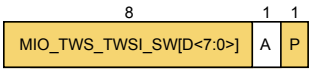


Four-Byte Read, (SIA<3> = 1) or (MIO_TWS_SW_TWSI[SIA<0>] = 0)

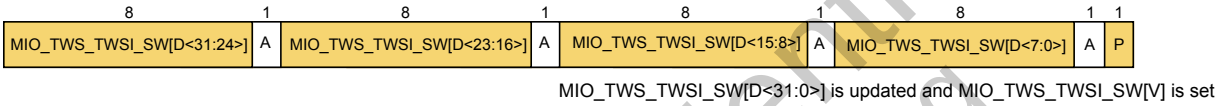


SIA-only Write Operation

One-Byte Write Operation, (SIA<3> = 0) or (MIO_TWS_TWSI_SW[SIA<0>] = 1) with SIA included



Four-Byte Write Operation, (SIA<3> = 1) or (MIO_TWS_TWSI_SW[SIA<0>] = 1) with SIA included



Vbyte provides a validity indication on a read. Table 35–2 describes the Vbyte.

Table 35–2 Validity Indication for Slave Read Operations

Bit Pos	Field Name	Description
<7>	V	If MIO_TWS_SW_TWSI is read, V is: MIO_TWS_SW_TWSI[V] **AND** MIO_TWS_SW_TWSI[SLONLY]. If MIO_TWS_TWSI_SW is read, V is: MIO_TWS_TWSI_SW[V<1>] **AND** MIO_TWS_TWSI_SW[V<0>]
<6:0>	—	Unused, read as 0x0.

35.3 Direct TWSI Core Usage

The CN71XX cores can communicate directly with the TWSI core via CSR accesses through MIO_TWS_SW_TWSI (refer to Section 35.4). Direct TWSI-core accesses should be avoided when the TWSI HLC is enabled (i.e. when TWSI_CTL[CE] is set to 1). TWSI_CTL[CE] is assumed to be clear (thus disabling the HLC) for the rest of Section 35.3. This provides lower-level access to TWSI functions that can be used to create a wider variety of transaction sequences. (Refer to the TWSI Status Register description for more possible status conditions.)

The TWSI interface can be used in:

- master transmit mode
- master receive mode
- slave transmit mode
- slave receive mode

Each of these modes is described in the following subsections.

Cavium Confidential For Bowen Yang 02/27/2015

35.3.1 Master Transmit Mode

In master transmit mode, the TWSI core transmits a number of bytes to a slave receiver.

A master-mode transaction starts when the software sets TWSI_CTL[STA] to 1. The TWSI core then tests the TWSI bus and transmits a START condition when the bus is free. When the START condition has been transmitted, TWSI_CTL[IFLG] is set to 1, and the status code in TWSI_STAT is 0x08 (*START condition transmitted*). Then TWSI_DATA must be loaded with either a 7-bit slave address or the first part of a 10-bit slave address, with the LSB cleared to 0 (i.e. with a write bit) to specify transmit mode. TWSI_CTL[IFLG] should then be cleared to 0 to prompt the transfer to continue.

After the 7-bit slave address (or the first part of a 10-bit address) plus the write bit have been transmitted, TWSI_CTL[IFLG] is set again. There are a number of status codes that might show up in the TWSI_STAT register. They are listed in Table 35–3.

Table 35–3 TWSI_STAT Status Codes (after 7-Bit or First Part of 10-Bit Address)

Status Code	TWSI Core State	Core Response	Next TWSI Core Action
0x18	Addr[1] + W[2] transmitted, ACK received	For a 7-bit address: Write the byte to TWSI_DATA, clear TWSI_CTL[IFLG], **or** Set TWSI_CTL[STA], clear TWSI_CTL[IFLG], **or** Set TWSI_CTL[STP], clear TWSI_CTL[IFLG], **or** Set TWSI_CTL[STA] and TWSI_CTL[STP], clear TWSI_CTL[IFLG] For a 10-bit address: Write extended-address byte to TWSI_DATA, clear TWSI_CTL[IFLG]	 Transmit data byte, receive ACK Transmit repeated START Transmit STOP Transmit STOP then START Transmit extended-address byte.
0x20	Addr[1] + W[2] transmitted, $\overline{\text{ACK}}$ received	Same as for 0x18.	Same as for 0x18.
0x38	Arbitration lost	Clear TWSI_CTL[IFLG], **or** Set TWSI_CTL[STA], clear TWSI_CTL[IFLG]	Return to idle Transmit START when bus is free
0x68	Arbitration lost, SLA[3] + W[2] received, ACK transmitted	Clear TWSI_CTL[IFLG], TWSI_CTL[AAK] = 0 **or** Clear TWSI_CTL[IFLG], TWSI_CTL[AAK] = 1	Receive data byte, transmit $\overline{\text{ACK}}$ Receive data byte, transmit ACK
0x78	Arbitration lost, general call addr received, ACK transmitted	Same as for 0x68.	Same as for 0x68.
0xB0	Arbitration lost, SLA[3] + R[4] received, ACK transmitted	Write the byte to TWSI_DATA, clear TWSI_CTL[IFLG], TWSI_CTL[AAK] = 0 **or** Write the byte to TWSI_DATA, clear TWSI_CTL[IFLG], TWSI_CTL[AAK] = 1	Transmit last byte, receive ACK Transmit data byte, receive ACK

1. Addr represents the contents of TWSI_DATA, either the 7-bit address or the first part of the 10-bit address.
2. W represents the write bit (i.e. LSB of the byte = 0).
3. SLA represents the CN71XX slave address (first part), in the TWSI_SLAVE_ADD register.
4. R represents the read bit (i.e. LSB of the byte = 1).

Cavium Confidential For Owen Yang ALU 02/27/2014

If 10-bit addressing is used, then once the first part of a 10-bit address plus the write bit have been transmitted successfully, the status code is 0x18 or 0x20.

After this interrupt has been serviced, and the second part of the address transmitted, the TWSI_STAT register contains one of the codes shown in Table 35–4.

Table 35–4 TWSI_STAT Status Codes (after Second Part of 10-Bit Address)

Status Code	TWSI Core State	Core Response	Next TWSI Core Action
0x38	Arbitration lost	Clear TWSI_CTL[IFLG], **or** Set TWSI_CTL[STA], clear TWSI_CTL[IFLG]	Return to idle Transmit START when bus is free
0x68	Arbitration lost, SLAX[1] + W[2] received, ACK transmitted	Clear TWSI_CTL[IFLG], TWSI_CTL[AAK] = 0 **or** Clear TWSI_CTL[IFLG], TWSI_CTL[AAK] = 1	Receive data byte, transmit $\overline{ACK}$ Receive data byte, transmit ACK
0xB0	Arbitration lost, SLAX[1] + R[3] received, ACK transmitted	Write the byte to TWSI_DATA, clear TWSI_CTL[IFLG], TWSI_CTL[AAK] = 0 **or** Write the byte to TWSI_DATA, clear TWSI_CTL[IFLG], TWSI_CTL[AAK] = 1	Transmit last byte, receive ACK Transmit data byte, receive ACK
0xD0	Addr[4] + W[2] transmitted, ACK received	Write the byte to TWSI_DATA, clear TWSI_CTL[IFLG], **or** Set TWSI_CTL[STA], clear TWSI_CTL[IFLG], **or** Set TWSI_CTL[STP], clear TWSI_CTL[IFLG], **or** Set TWSI_CTL[STA] and TWSI_CTL[STP], clear TWSI_CTL[IFLG]	Transmit data byte, receive ACK Transmit repeated START Transmit STOP Transmit STOP then START
0xD8	Addr[4] + W[2] transmitted, $\overline{ACK}$ received	Same as for 0xD0.	Same as for 0xD0.

1. SLAX represents the CN71XX slave address (second part), in the TWSI_SLAVE_ADD_EXT register.
2. W represents the write bit (i.e. LSB of the byte = 0).
3. R represents the read bit (i.e. LSB of the byte = 1).
4. Addr represents the contents of TWSI_DATA, the second part of the 10-bit address.

Cavium Confidential For Bowen Yang ALU 02/27/2014

If a repeated START condition has been transmitted, the status code is 0x10 instead of 0x08.

After each data byte has been transmitted, TWSI_CTL[IFLG] is set to 1 and one of the three listed in Table 35–5 is loaded into TWSI_STAT.

Table 35–5 TWSI_STAT Status Codes (after Repeated START Transmission)

Status Code	TWSI Core State	Core Response	Next TWSI Core Action
0x28	Data byte transmitted, ACK received	Write the byte to TWSI_DATA, clear TWSI_CTL[IFLG], **or**	Transmit data byte, receive ACK
		Set TWSI_CTL[STA], clear TWSI_CTL[IFLG], **or**	Transmit repeated START
		Set TWSI_CTL[STP], clear TWSI_CTL[IFLG], **or**	Transmit STOP
		Set TWSI_CTL[STA] and TWSI_CTL[STP], clear TWSI_CTL[IFLG]	Transmit STOP then START
0x30	Data byte transmitted, $\overline{\text{ACK}}$ received	Same as for 0x28.	Same as for 0x28.
0x38	Arbitration lost	Clear TWSI_CTL[IFLG], **or**	Return to idle
		Set TWSI_CTL[STA], clear TWSI_CTL[IFLG]	Transmit START when bus is free

When all bytes have been transmitted, TWSI_CTL[STP] should be set to 1. The TWSI core then does the following:

1. Transmits a STOP condition.
2. Clears TWSI_CTL[STP] to 0.
3. Returns to the idle state.

35.3.2 Master Receive Mode

In master receive mode, the TWSI core receives a number of bytes from a slave transmitter.

Refer to "Master Transmit Mode" on page 1526 and note that this is nearly identical. After the START condition has been transmitted, TWSI_CTL[IFLG] is set to 1 and a status code of 0x08 (*START condition transmitted*) is loaded into TWSI_STAT. Load the TWSI_DATA register with the 7-bit slave address (or the first part of the 10-bit slave address), with the LSB set to 1 to signify a read operation (if 7-bit). Then clear TWSI_CTL[IFLG] to 0 to prompt the transfer to continue.

When the 7-bit slave address (or the first part of the 10-bit slave address) and the read bit have been transmitted, TWSI_CTL[IFLG] is set again to 1. A number of status codes may possibly be in TWSI_STAT. They are shown in Table 35–6.

Cavium Confidential For Cavium Shwen Yang ALU 02/17/2014

Table 35–6 TWSI_STAT Status Codes (Master Receive Mode)

Status Code	TWSI Core State	Core Response	Next TWSI Core Action
0x40	Addr[1] + R[2] transmitted, ACK received	For a 7-bit/10-bit address: Clear TWSI_CTL[IFLG], TWSI_CTL[AAK] = 0 **or** Clear TWSI_CTL[IFLG], TWSI_CTL[AAK] = 1	 Receive data byte, transmit $\overline{\text{ACK}}$ Receive data byte, transmit ACK
0x48	$\overline{\text{Addr}}$[1] + R[2] transmitted, $\overline{\text{ACK}}$ received	For a 7-bit/10-bit address: Set TWSI_CTL[STA], clear TWSI_CTL[IFLG], **or** Set TWSI_CTL[STP], clear TWSI_CTL[IFLG], **or** Set TWSI_CTL[STA] and TWSI_CTL[STP], clear TWSI_CTL[IFLG]	 Transmit repeated START Transmit STOP Transmit STOP then START
0x38	Same as for Master Transmit.	Same as for Master Transmit.	Same as for Master Transmit.
0x68	Same as for Master Transmit.	Same as for Master Transmit.	Same as for Master Transmit.
0x78	Same as for Master Transmit.	Same as for Master Transmit.	Same as for Master Transmit.
0xB0	Same as for Master Transmit.	Same as for Master Transmit.	Same as for Master Transmit.

1. Addr represents the contents of TWSI_DATA, either the 7-bit address or the first part of the 10-bit address.
2. R represents the read bit (i.e. LSB of the byte = 1).

If 10-bit addressing is being used, the slave is first addressed using the full 10-bit address plus the write bit. The master then issues a restart followed by the first part of the 10-bit address again except with the read bit, after which the status code is 0x40 or 0x48. It is the responsibility of the slave to remember that it had been selected prior to the restart.

If a repeated START condition is transmitted, the status code is 0x10 rather than 0x08.

After each data byte has been received, TWSI_CTL[IFLG] is set to 1 and one of the three codes listed in Table 35–7 is in TWSI_STAT.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 35–7 TWSI_STAT Status Codes (Master Receive – Data Received)

Status Code	TWSI Core State	Core Response	Next TWSI Core Action
0x50	Data byte received, ACK transmitted	Read TWSI_DATA, clear TWSI_CTL[IFLG], TWSI_CTL[AAK] = 0 **or** Read TWSI_DATA, clear TWSI_CTL[IFLG], TWSI_CTL[AAK] = 1	Receive data byte, transmit $\overline{ACK}$ Receive data byte, transmit ACK
0x58	Data byte received, $\overline{ACK}$ transmitted	Read TWSI_DATA, Set TWSI_CTL[STA], clear TWSI_CTL[IFLG], **or** Read TWSI_DATA, Set TWSI_CTL[STP], clear TWSI_CTL[IFLG], **or** Read TWSI_DATA, Set TWSI_CTL[STA] and TWSI_CTL[STP], clear TWSI_CTL[IFLG]	Transmit repeated START Transmit STOP Transmit STOP then START
0x38	Arbitration lost in $\overline{ACK}$ bit	Same as for Master Transmit.	Same as for Master Transmit.

When all bytes have been received (an $\overline{ACK}$ should have been transmitted), then set TWSI_CTL[STP]. The TWSI core then does the following:

1. Transmits a STOP condition.
2. Clears TWSI_CTL[STP] to 0.
3. Returns to the idle state.

35.3.3 Slave Transmit Mode

In slave transmit mode, the TWSI core transmits a number of bytes to a master receiver. TWSI_CTL[AAK] must be set to 1 to enable the core to initially respond as a slave.

The TWSI core enters slave transmit mode when it receives its own slave address and a read bit after a START condition. The TWSI core then transmits an acknowledge (ACK) bit and sets TWSI_CTL[IFLG] to 1. The TWSI_STAT register is set to 0xA8.

NOTE: When the TWSI core has an extended slave address (indicated by 11110 in TWSI_SLAVE_ADD[A<7:3>]), it transmits ACK after the first address byte is received after a restart. An interrupt is then generated, TWSI_CTL[IFLG] is set, and TWSI_STAT is set to 0xA8. No second address byte is sent by the master – it is up to the slave to remember that it had been selected prior to the restart.

Slave transmit mode can also be entered directly from a master mode if arbitration is lost in master mode during the transmission of an address and the TWSI core's slave address and read bit are received.

In this situation, TWSI_STAT is set to 0xB0.

The data byte to be transmitted should then be loaded into TWSI_DATA and TWSI_CTL[IFLG] cleared to 0. When the TWSI core has transmitted the byte and received ACK, it sets TWSI_CTL[IFLG] to 1 and sets TWSI_STAT to 0xB8. Once the

Cavium Confidential For Bowen Yang ALU 02/2/2014

last byte to be transmitted has been loaded into TWSI_DATA, both TWSI_CTL[IFLG, AAK] should be cleared to 0. After the last byte has been transmitted, TWSI_CTL[IFLG] is set to 1 and TWSI_STAT is set to 0xC8.

The TWSI core then returns to the idle state and TWSI_CTL[AAK] must be set to 1 before slave mode can be entered again.

If no ACK is received after transmitting a byte, the TWSI core sets TWSI_CTL[IFLG] to 1 and TWSI_STAT to 0xC0, then returns to the idle state.

If the STOP condition is detected after an ACK, the TWSI core returns to idle.

35.3.4 Slave Receive Mode

In slave receive mode, the TWSI core receives a number of bytes from a master transmitter. TWSI_CTL[AAK] must be set to 1 to enable the TWSI core to respond initially as a slave.

The TWSI core enters slave receive mode when it receives its own slave address and a write bit after a START condition. The TWSI core then transmits ACK and sets TWSI_CTL[IFLG] to 1 and TWSI_STAT to 0x60.

It also enters slave receive mode when it receives the general call address 0x00 if TWSI_SLAVE_ADD[GCE] is set to 1. The status code in TWSI_STAT is then set to 0x70.

NOTE: When the TWSI core has an extended slave address (indicated by 11110 in TWSI_SLAVE_ADD[A<7:3>]), it transmits ACK after the first address byte is received after a restart. No interrupt is generated, TWSI_CTL[IFLG] is not set, and TWSI_STAT is not changed.

Only after the second address byte is received does the TWSI core generate an interrupt, set TWSI_CTL[IFLG] to 1, and set the appropriate status code (either 0x60 or 0x70).

Slave transmit mode can also be entered directly from a master mode if arbitration is lost in master mode during the transmission of an address and either the TWSI core's slave address and write bit are received, or the general call address is received when TWSI_SLAVE_ADD[GCE] = 1.

In this situation, TWSI_STAT is set to either 0x68 (slave address) or 0x78 (general call address). TWSI_CTL[IFLG] must be cleared to 0 to allow the data transfer to continue.

If TWSI_CTL[AAK] is set to 1, then after each byte is received, an ACK is transmitted, TWSI_CTL[IFLG] is set to 1, and TWSI_STAT is set to 0x80 (slave address) or 0x90 (general call address). The received-data byte can be read from TWSI_DATA, then TWSI_CTL[IFLG] must be cleared to 0 to allow the transfer to continue.

When the STOP condition or a repeated START condition is detected after ACK, TWSI_CTL[IFLG] is set to 1 and TWSI_STAT is set to 0xA0.

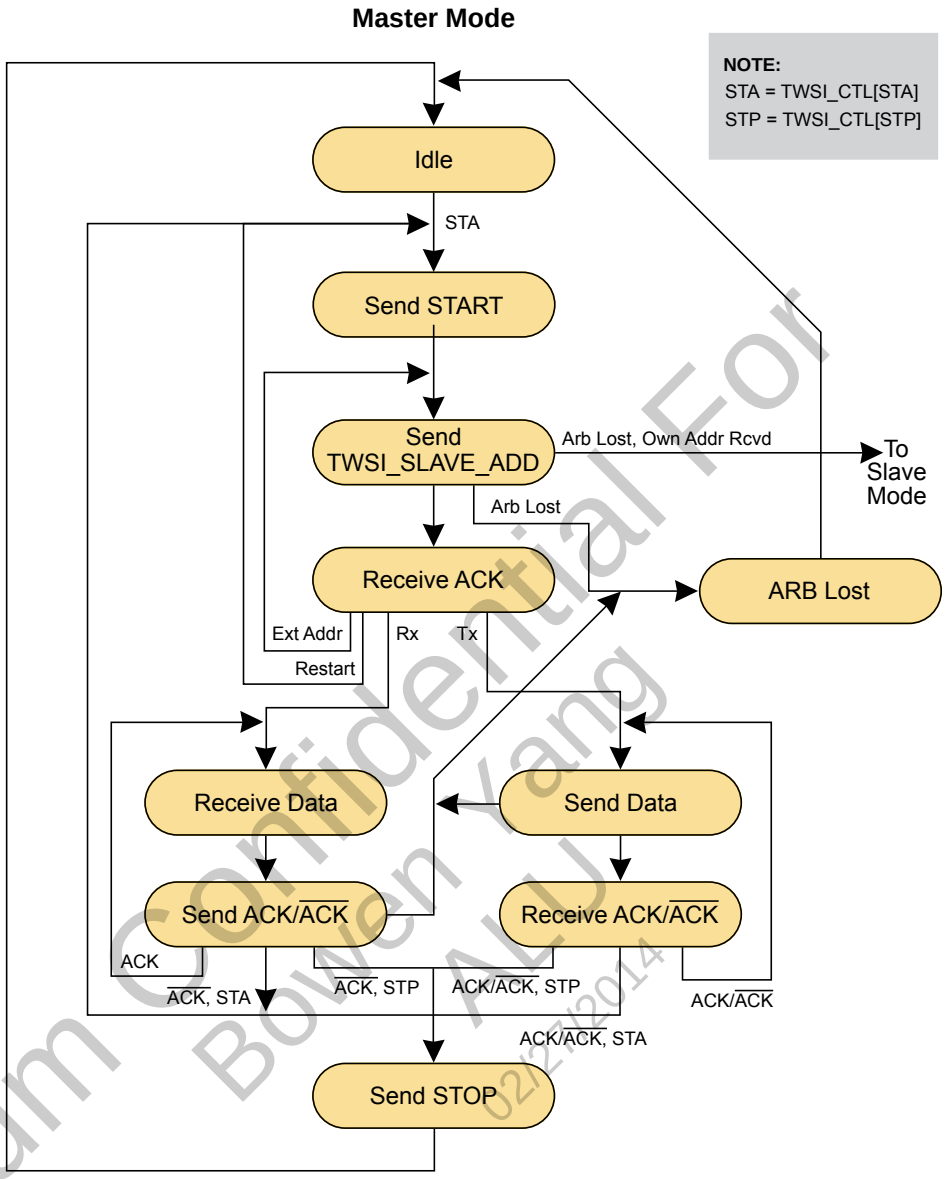
If TWSI_CTL[AAK] = 0 during a transfer, the TWSI core transmits an $\overline{ACK}$ after the next byte is received, sets TWSI_CTL[IFLG] to 1, and sets TWSI_STAT to either 0x88 (slave address) or 0x98 (general call address). When TWSI_CTL[IFLG] is cleared to 0, the TWSI core returns to the idle state.

Cavium Confidential For Boyen Yang Aug 27 2014

35.3.5 TWSI Core Flow Diagrams

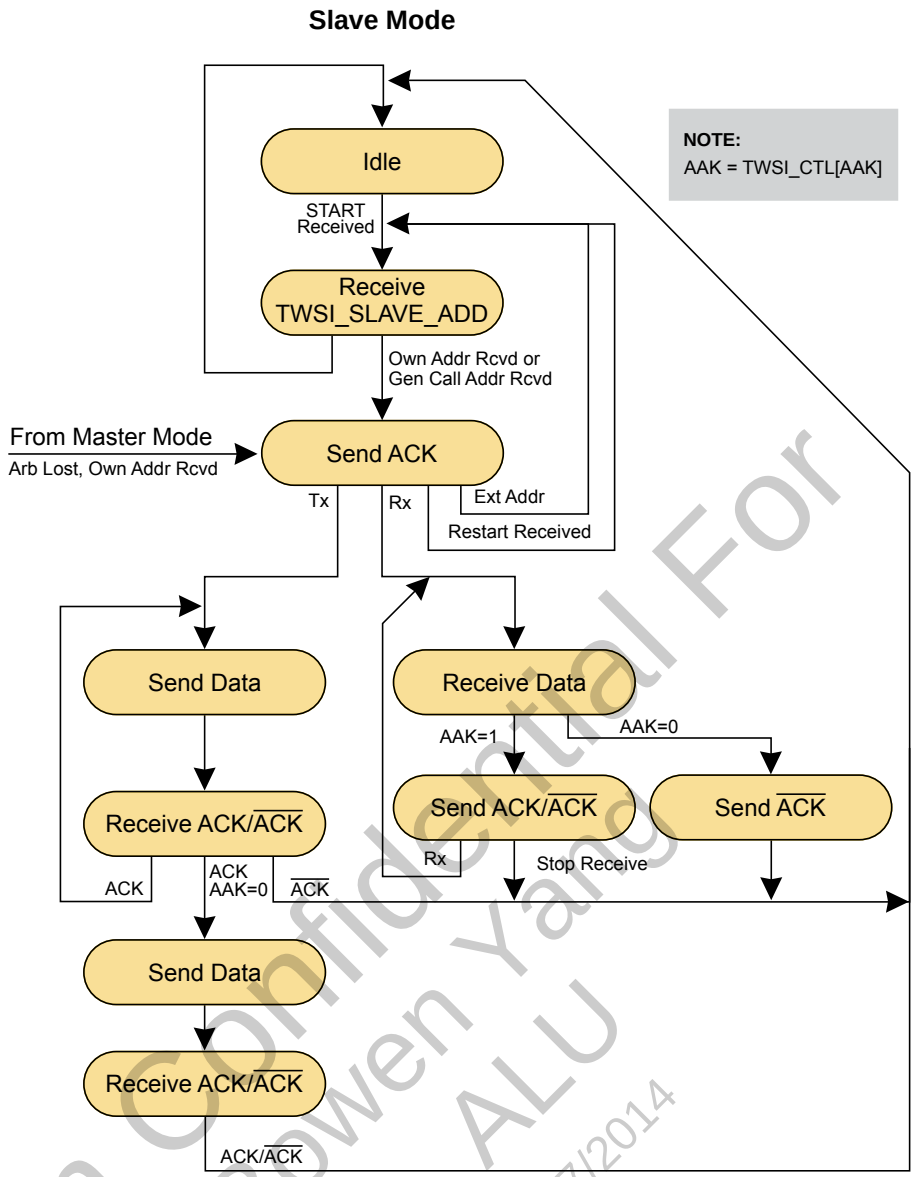
Figure 35–11 shows the master-mode flow, and Figure 35–12 shows the slave-mode flow.

Figure 35–11 Master-Mode Flow Diagram



Cavium Confidential For Bowen Yang ACU 02/23/2014

Figure 35–12 Slave-Mode Flow Diagram



Cavium Confidential For Bowen Yang ALU 02/27/2014

35.4 TWSI Control Registers

This section describes the internal registers of the two TWSI cores. These registers exist for both TWSI controllers and can be read/written by write operations to MIO0_TWS_SW_TWSI and MIO1_TWS_SW_TWSI with [OP<2>] = 1 and [SLONLY] = 0.

The registers described are:

- TWSI slave address register (TWSI_SLAVE_ADD)
- TWSI slave extended-address register (TWSI_SLAVE_EXTADD)
- TWSI data register (TWSI_DATA)
- TWSI control register (TWSI_CTL)
- TWSI status register (TWSI_STAT)
- TWSI master clock register (TWSI_CLK)
- TWSI clock control register (TWSI_CLKCTL)
- TWSI software reset register (TWSI_RST)

35.4.1 TWSI Slave Address Register

The TWSI_SLAVE_ADD register is described in Table 35–8.

Table 35–8 TWSI_SLAVE_ADD Bit Description

Bit Pos	Value with 7-bit Addressing	Value with 10-bit Addressing	Reset Value
<7>	Slave address bit <6>	1	1
<6>	Slave address bit <5>	1	1
<5>	Slave address bit <4>	1	1
<4>	Slave address bit <3>	1	0
<3>	Slave address bit <2>	0	1
<2>	Slave address bit <1>	Slave address bit <9>	1
<1>	Slave address bit <0>	Slave address bit <8>	1
<0>	GCE general call address enable	GCE general call address enable	0

For 7-bit addressing:

Slave address is the 7-bit address of CN71XX when in slave mode. When CN71XX receives this address after a START condition, it sets TWSI_CTL[IFLG] (i.e. the TWSI core-interrupt bit) and the TWSI core enters slave mode, assuming TWSI_CTL[ENAB] is set to 1.

If GCE = 1, CN71XX also recognizes the general call address (0x00).

CN71XX resets to 7-bit addressing with an address of 0x77.

For 10-bit addressing:

When the received address has bits <7:3> = 11110, CN71XX recognizes this as the first part of a 10-bit address, and if the next two bits match TWSI_SLAVE_ADD<2:1> (i.e. slave address <9:8>), it sends an ACK. After the next byte of the address has been received, and if the address matches TWSI_SLAVE_EXTADD, CN71XX sets TWSI_CTL[IFLG] and the TWSI core enters slave mode. (TWSI_SLAVE_ADD<7:3> must be programmed as specified in order for CN71XX to respond as a slave to a 10-bit address.)

Cavium Confidential For Bowen Yang ALU 04/27/2014

35.4.2 TWSI Slave Extended-Address Register

The TWSI_SLAVE_EXTADD register is described in Table 35–9.

Table 35–9 TWSI_SLAVE_EXTADD Bit Description

Bit Pos	Value with 7-bit Addressing	Value with 10-bit Addressing	Reset Value
<7>	—	Slave address bit <7>	0
<6>	—	Slave address bit <6>	0
<5>	—	Slave address bit <5>	0
<4>	—	Slave address bit <4>	0
<3>	—	Slave address bit <3>	0
<2>	—	Slave address bit <2>	0
<1>	—	Slave address bit <1>	0
<0>	—	Slave address bit <0>	0

35.4.3 TWSI Data Register

TWSI_DATA should not be accessed when the TWSI HLC is enabled (i.e. when TWSI_CTL[CE] = 1).

This register contains the data byte/slave address to be transmitted, or the data byte that has just been received.

- In transmit mode, the byte is sent MSB first
- In receive mode, the first bit received is placed in the MSB of the register.

After each byte is transmitted, TWSI_DATA contains the byte that was actually present on the bus, so in the event of lost arbitration, it will contain the received byte.

35.4.4 TWSI Control Register

The TWSI_CTL register is described in Table 35–10.

Table 35–10 TWSI_CTL Bit Description

Bit Pos	Field Name	Reset Value	Description
<7>	CE	1	High-level controller enable. When this bit is set to 1, the HLC interface is enabled and drives the sequence of the TWSI bus commands to transfer or receive data. When this bit is cleared to 0, the HLC interface is disabled. The software can still use the TWSI interface by issuing individual commands using TWSI control, status and data registers
<6>	ENAB	1	Bus enable. Must be set to 1 to use the TWSI interface (either via the HLC interface or via direct core usage).

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 35–10 TWSI_CTL Bit Description (Continued)

Bit Pos	Field Name	Reset Value	Description
<5>	STA	0	Master-mode start. This bit should be clear when the HLC is enabled (i.e. when TWSI_CTL[CE] is set to 1). When this bit is set to 1, the TWSI core enters master mode and transmits a START condition on the bus when the bus is free. • If set to 1 when the controller is already in master mode and one or more bytes have been transmitted, then a repeated START condition is sent. • If this bit is set to 1 when the TWSI core is being accessed by an external device in slave mode, the core completes the data transfer in slave mode then enters master mode when the bus has been released. If both STA and STP bits are set, the TWSI core first transmits the STOP condition (if in master mode) then transmits the START condition. This bit is cleared automatically after a START condition has been sent. Writing a 0 to this bit has no effect.
<4>	STP	0	Master-mode stop. This bit should be clear when the HLC is enabled (i.e. when TWSI_CTL[CE] is set). • If this bit is set to 1 when the TWSI core is in master mode, a STOP condition is transmitted on the TWSI bus. • If this bit is set to 1 when the TWSI core is in slave mode, the HLC behaves as if a STOP condition has been received, but no STOP condition is transmitted on the TWSI bus. If both STA and STP bits are set, the TWSI core first transmits the STOP condition (if in master mode) then transmits the START condition. This bit is cleared automatically. Writing a 0 to this bit has no effect.
<3>	IFLG	0	Interrupt flag. This bit should be ignored when the HLC is enabled (i.e. when TWSI_CTL[CE] is set). This bit is automatically set to 1 when any of 28 (out of the possible 29) TWSI core states is entered. The only state that does not set this bit is state 0xF8 (see Section 35.4.5). When the TWSI core sets this bit, the low period of the TWSI bus clock line (SCL) is stretched and the data transfer is suspended. When 0 is written to this bit, the bit is cleared and the TWSI clock line is released.
<2>	AAK	1	Assert acknowledge. This bit should be set when the HLC is enabled (i.e. when TWSI_CTL[CE] is set). When this bit is set to 1, an ACK is sent during the acknowledge clock pulse on the TWSI bus if any of the following occurs: • the whole of a matching 7-bit slave address or the first or the second byte of a matching 10-bit slave address has been received, • the general call address has been received and it was enabled, • a data byte has been received in master or slave mode. When this bit is cleared to 0, a “not acknowledge” ($\overline{\text{ACK}}$) is sent when a data byte is received in master or slave mode. • If this bit is cleared to 0 in slave transmitter mode, the byte in TWSI_DATA is assumed to be the last byte. After this byte has been transmitted, the TWSI core enters the idle state 0xC8. The TWSI core does not respond as a slave unless this bit is set.
<1:0>	—	00	Reserved

Cavium Confidential For Bowen Yang ALU 02/27/2014

35.4.5 TWSI Status Register

TWSI_STAT is a read-only register containing a state code.

- When this register contains the state 0xF8, no relevant status information is available and no interrupt bit is set.
- All the other 28 states correspond to a non-idle state of the TWSI.

When any of these non-idle states is entered, the corresponding state appears in this register and TWSI_CTL[IFLG] is set to 1. When TWSI_CTL[IFLG] is cleared to 0, the state returns to 0xF8.

If an illegal condition occurs on the TWSI bus, the bus-error state is entered (status code 0x00). To recover from this state, TWSI_CTL[STP] must be set to 1 and TWSI_CTL[IFLG] cleared to 0. The TWSI controller then returns to the idle state, and no STOP condition is transmitted on the TWSI bus.

The TWSI core state codes are listed in Table 35–11.

Table 35–11 TWSI Core-State Codes

Code	TWSI Core State
0x00	Bus error
0x08	START condition transmitted
0x10	Repeated START condition transmitted
0x18	Address + write bit transmitted, ACK received
0x20	Address + write bit transmitted, $\overline{\text{ACK}}$ received
0x28	Data byte transmitted in master mode, ACK received
0x30	Data byte transmitted in master mode, $\overline{\text{ACK}}$ received
0x38	Arbitration lost in address or data byte
0x40	Address + read bit transmitted, ACK received
0x48	Address + read bit transmitted, $\overline{\text{ACK}}$ received
0x50	Data byte received in master mode, ACK transmitted
0x58	Data byte received in master mode, $\overline{\text{ACK}}$ transmitted
0x60	Slave address + write bit received, ACK transmitted
0x68	Arbitration lost in address as master, slave address + write bit received, ACK transmitted
0x70	General call address received, ACK transmitted
0x78	Arbitration lost in address as master, general call address received, ACK transmitted
0x80	Data byte received after slave address received, ACK transmitted
0x88	Data byte received after slave address received, $\overline{\text{ACK}}$ transmitted
0x90	Data byte received after general call address received, ACK transmitted
0x98	Data byte received after general call address received, $\overline{\text{ACK}}$ transmitted
0xA0	STOP or repeated START condition received in slave mode
0xA8	Slave address + read bit received, ACK transmitted
0xB0	Arbitration lost in address as master, slave address + read bit received, ACK transmitted
0xB8	Data byte transmitted in slave mode, ACK received
0xC0	Data byte transmitted in slave mode, $\overline{\text{ACK}}$ received
0xC8	Last byte transmitted in slave mode, ACK received
0xD0	Second address byte + write bit transmitted, ACK received
0xD8	Second address byte + write bit transmitted, $\overline{\text{ACK}}$ received
0xF8	No relevant status information

Cavium Confidential For Bowen Yang ALU 02/27/2014

The TWSI core prevents the software/hardware from causing a TWSI bus error in most of the cases (e.g. it does not let the firmware send a STOP condition immediately after a START condition, as it is illegal). However, the TWSI core does allow the firmware to cause a bus error by sending a STOP condition after the first byte of a 10-bit address. The controller can recover from the error state as described above.

35.4.6 TWSI Master Clock Register

The 8-bit value in TWSI_CLK controls the frequency ratio between CN71XX's coprocessor clock and the TWSI controller's clock (TCLK). The TWSI_CLK register is described in Table 35–12.

Table 35–12 TWSI_CLK Bit Description

Bits	Field Name	Reset Value	Description
<7:0>	THP	0x18	TCLK half period.

The period of TCLK is defined as: 2 × (THP+1) times the coprocessor clock, so the frequency of TCLK is defined as:

$$\text{TCLK frequency} = \frac{\text{coprocessor frequency}}{2 \times (\text{THP} + 1)}$$

TWSI_CLK must never be set to any of the values 0 – 4.

The reset value with 600-MHz coprocessor-clock cycle generates an 12-MHz TCLK cycle.

35.4.7 TWSI Clock Control Register

TWSI_CLKCTL is a write-only register. The bits <6:0> control the frequency at which the TWSI bus is sampled and the frequency of the TWSI clock line when the TWSI controller is in master mode. TWSI_CLKCTL is described in Table 35–13.

Table 35–13 TWSI_CLKCTL Bit Description

Bits	Field Name	Reset Value	Description
<7>	—	—	Reserved. Read as 0.
<6:3>	M	0x2	M divider
<2:0>	N	0x0	N divider

The TWSI clock frequency (TCLK) is first divided by a factor of 2^N, where N is the value defined by <2:0>. The output of this clock divider is FSAMP, which is then divided by a further factor of (M+1)×10, where M is the value defined by bits <6:3>. The output of this clock divider is FOSCL.

The TWSI bus is sampled by the controller at the frequency defined by FSAMP (N is <2:0> of TWSI_CLKCTL):

$$\text{FSAMP} = \text{TCLK} / 2^N$$

The TWSI OSCL output frequency, in master mode, is (N is <2:0> and M is <6:3> of TWSI_CLKCTL):

$$\text{FOSCL} = \frac{\text{FSAMP}}{(M+1)\times 10} = \frac{\text{TCLK}}{2^N \times (M + 1) \times 10}$$

Cavium Confidential For Bowen Yang ALU 02/27/2014

The use of two separately programmable dividers allows the master mode output frequency to be set independently of the frequency at which the TWSI bus is sampled. This is particularly useful in multimaster systems because the frequency at which the TWSI bus is sampled must be at least 10 times the frequency of the fastest master on the bus to ensure that START and STOP conditions are always detected. By using two programmable clock divider stages, a high sampling frequency can be ensured while allowing the master mode output to be set to a lower frequency.

For a coprocessor-clock frequency of 600 MHz, immediately after reset CN71XX samples at the rate of 12 MHz (FSAMP), and CN71XX drives at the frequency of 400 KHz (FOSCL).

35.4.8 TWSI Software Reset Register

A software reset may be applied to the TWSI bus interface by writing any value to TWSI_RST. A software reset sets the TWSI core back to idle (TWSI_STAT = 0xF8) and sets TWSI_CTL[STA, STP, IFLG] to 0s.

NOTE: After this register is written, the TWSI controller is not available for any communication with the TWSI bus for three TCLK cycles (for the default value of the TWSI_CLK, it derives to 315 coprocessor-clock cycles).

35.5 TWSI Registers

The TWSI registers are listed in Table 35–14.

Table 35–14 TWSI Registers

Register	Address	CSR Type[1]	Detailed Description
MIO_TWS(0..1)_SW_TWSI	0x0001180000001000 + α*0x200	RSL	See page 1540
MIO_TWS(0..1)_TWSI_SW	0x0001180000001008 + α*0x200	RSL	See page 1542
MIO_TWS(0..1)_INT	0x0001180000001010 + α*0x200	RSL	See page 1542
MIO_TWS(0..1)_SW_TWSI_EXT	0x0001180000001018 + α*0x200	RSL	See page 1543

1. RSL-type registers are accessed indirectly across the IOI.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Software to TWSI Register MIO_TWS(0..1)_SW_TWSI

This register allows software to:

- Initiate master-mode operations with a write operation, and read the result with a read operation.
- Load four bytes for later retrieval (slave mode) with a write operation and check validity with a read operation.
- Launch a configuration read/write operation with a write operation and read the result with a read operation.

This register should be read or written by software, and read by the TWSI device. The TWSI device can use either two-byte or five-byte read operations to reference this register.

The TWSI device considers this register valid when [V] = 1 and [SLONLY] = 1. See Table 35–14 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63>	V	RC/W/H	0	—	Valid bit: • set on a write operation (should always be written with a 1) • cleared when a TWSI master-mode operation completes • cleared when a TWSI configuration register access completes • cleared when the TWSI device reads the register if SLONLY = 1
<62>	SLONLY	R/W	—	—	Slave-only mode. When this bit is set, no operations are initiated with a write operation. Only the D field is updated in this case When this bit is clear, a write operation initiates either a master-mode operation or a TWSI configuration register access.
<61>	EIA	R/W	0	—	Extended internal address. Sends an additional internal address byte (the MSB of IA is from MIO_TWS_SW_TWSI_EXT[IA]).
<60:57>	OP	R/W	0x0	—	Opcode field. When the register is written with SLONLY = 0, this field initiates one of the following read or write operations: 0x0 = 7-bit byte master-mode operation 0x1 = 7-bit byte combined-read master-mode operation, 7-bit byte write-with-IA master-mode operation 0x2 = 10-bit byte master-mode operation 0x3 = 10-bit byte combined-read master-mode operation, 10-bit byte write-with-IA master-mode operation 0x4 = TWSI master-clock register, TWSI_CLK in Section 35.4.6 0x6 = See EOP_IA field 0x8 = 7-bit 4-byte master-mode operation 0x9 = 7-bit 4-byte combined-read master-mode operation, 7-bit 4-byte write-with-IA master-mode operation 0xA = 10-bit 4-byte master-mode operation 0xB = 10-bit 4-byte combined-read master-mode operation 10-bit 4-byte write-with-IA master-mode operation
<56>	R	R/W/H	0	—	Read bit or result. If this bit is set on a CSR write when SLONLY = 0, the operation is a read operation (if clear, it is a write operation). On a CSR read, this bit returns the result indication for the most recent master-mode operation, 1 = success, 0 = failure.

Cavium Confidential For Bowen Yang 02-13-2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<55>	SOVR	R/W	0	—	Size override. If this bit is set, use the SIZE field to determine the master-mode operation size rather than what OP specifies. For operations greater than four bytes, the additional data is contained in MIO_TWS_SW_TWSI_EXT[D].
<54:52>	SIZE	R/W	0x0	—	Size. Specifies the size in bytes of the master-mode operation if SOVR = 1. Specified in –1 notation (i.e. 0 = 1 byte, 1 = 2 bytes, ... 7 = 8 bytes)
<51:50>	SCR	R/W	0x0	—	Scratch. Unused, but retain state.
<49:40>	ADDR	R/W	0x0	—	Address field. The address of the remote device for a master-mode operation. A<9:7> are only used for 10-bit addressing. Note that when mastering a 7-bit OP, A<6:0> should not take any of the values 0x78, 0x79, 0x7A nor 0x7B. (These 7-bit addresses are reserved to extend to 10-bit addressing).
<39:35>	IA	R/W	0x0	—	Internal Address. Used when launching a combined master-mode operation.
<34:32>	EOP_IA	R/W	0x0	—	Extra opcode, used when OP<3:0> = 0110 and SLONLY = 0. 0x0 = TWSI slave address register (TWSI_SLAVE_ADD) 0x1 = TWSI data register (TWSI_DATA) 0x2 = TWSI control register (TWSI_CTL) 0x3 = (when R = 0) TWSI clock control register (TWSI_CLKCTL) 0x3 = (when R = 1) TWSI status register (TWSI_STAT) 0x4 = TWSI extended slave register (TWSI_SLAVE_ADD_EXT) 0x7 = TWSI soft reset register (TWSI_RST) Also provides the lower 3 bits of internal address when launching a combined master-mode operation.
<31:0>	D	R/W/H	0x0	—	Data field. Used on a write operation when: • Initiating a master-mode write operation (SLONLY = 0) • Writing a TWSI configuration register (SLONLY = 0) • A slave-mode write operation (SLONLY = 1) The read value is updated by: • A write operation to this register • Master-mode completion (contains error code) • TWSI configuration-register read (contains result)

Cavium Confidential For Bowen Yang ALU 02/27/2014

TWSI to Software Register MIO_TWS(0..1)_TWSI_SW

This register allows the TWSI device to transfer data to software and later check that software has received the information.

This register should be read or written by the TWSI device, and read by software. The TWSI device can use one-byte or four-byte payload write operations, and two-byte payload read operations. The TWSI device considers this register valid when V = 1. See Table 35–14 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:62>	V	RC/W	0x0	—	Valid bits. These bits are not directly writable. They are set to 0x1 on any write operation by the TWSI device. They are cleared on any read operation by software.
<61:32>	—	RAZ	—	—	Reserved
<31:0>	D	R/W	—	—	Data field. Updated on a write operation by the TWSI device.

TWSI Interrupt Register MIO_TWS(0..1)_INT

This register contains the TWSI interrupt-enable mask and the interrupt-source bits.

NOTE: The interrupt-source bit for the TWSI core interrupt (CORE_INT) is read-only, the appropriate sequence must be written to the TWSI core to clear this interrupt.

The other interrupt-source bits are write-1-to-clear. TS_INT is set on the update of the MIO_TWS(0..1)_TWSI_SW register (i.e. when it is written by a TWSI device). ST_INT is set whenever the valid bit of the MIO_TWS(0..1)_SW_TWSI is cleared (see above for reasons).

This register also contains a read-only copy of the TWSI bus (SCL and SDA) as well as control bits to override the current state of the TWSI bus (SCL_OVR and SDA_OVR). Setting an override bit high will result in the open drain driver being activated, thus driving the corresponding signal low.

See Table 35–14 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:12>	—	RAZ	—	—	Reserved
<11>	SCL	RO/H	—	—	SCL signal
<10>	SDA	RO/H	—	—	SDA signal
<9>	SCL_OVR	R/W	0	—	SCL override
<8>	SDA_OVR	R/W	0	—	SDA override
<7>	—	RAZ	—	—	Reserved
<6>	CORE_EN	R/W	0	—	TWSI core interrupt-enable. Cleared to 0 when the HLC is enabled.
<5>	TS_EN	R/W	0	—	MIO_TWS_TWSI_SW register-update interrupt-enable. Cleared to 0 when the HLC is disabled.
<4>	ST_EN	R/W	0	—	MIO_TWS_SW_TWSI register-update interrupt-enable. Cleared to 0 when the HLC is disabled.
<3>	—	RAZ	—	—	Reserved
<2>	CORE_INT	RO	0	—	TWSI core interrupt. Ignored when the HLC is enabled.
<1>	TS_INT	R/W1C/H	0	—	MIO_TWS_TWSI_SW register-update interrupt. Ignored when the HLC is disabled.
<0>	ST_INT	R/W1C/H	0	—	MIO_TWS_SW_TWSI register-update interrupt. Ignored when the HLC is disabled.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Software to TWSI Extension Register MIO_TWS(0..1)_SW_TWSI_EXT

This register contains an additional byte of internal address and four additional bytes of data to be used with TWSI master-mode operations.

The IA field is sent as the first byte of internal address when performing master-mode combined-read/write-with-IA operations and MIO_TWS_SW_TWSI[EIA] is set. The D field extends the data field of MIO_TWS_SW_TWSI for a total of 8 bytes (SOVR must be set to perform operations greater than 4 bytes).

See Table 35–14 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:40>	—	RAZ	—	—	Reserved
<39:32>	IA	R/W	0x0	—	Extended internal address. Sent as the first byte of internal address when performing master-mode combined-read/write-with-IA operations and MIO_TWS_SW_TWSI[EIA] is set.
<31:0>	D	R/W/H	0x0	—	Extended data. Extends the data field of MIO_TWS_SW_TWSI for a total of eight bytes (MIO_TWS_SW_TWSI[SOVR] must be set to 1 to perform operations greater than four bytes).

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 36

SPI/MPI Unit

This chapter contains the following subjects:

- Overview
- Pin Usage
- SPI/MPI Configuration
- SPI/MPI Usage
- SPI as a Boot Device
- Examples
- SPI/MPI Registers

Cavium Confidential For Bowen Yang ALU 02/27/2014

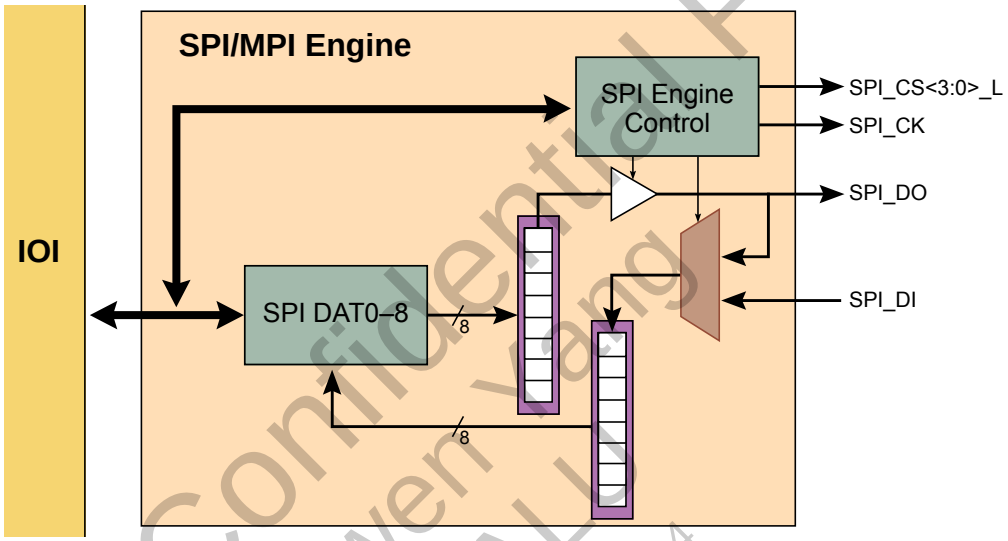
Overview

The serial peripheral interface (SPI)/multiprocessor peripheral interface (MPI) is a highly flexible implementation capable of connecting to many variations on the SPI/MPI. The key features are the following:

- Clock generation to 25 MHz
- Ability to tristate transmit data wire for MPI-style single-wire transmit/receive
- MSB- or LSB-first transmission/reception
- Interrupt- or polling-based transactions
- Continuous clocking mode
- Flexible timing between commands
- Use as a boot device
- Optional integrated chip-select signals

An overview of the interface is shown in Figure 36–1.

Figure 36–1 SPI/MPI Overview



Cavium Confidential For Bowen Yang ALU 02/27/2014

36.1 Pin Usage

Some of the SPI/MPI pins, when not enabled for SPI/MPI use, function as boot bus and UART pins. Table 36–1 describes the SPI/MPI pins and the CSR fields that switch them from standard boot bus and UART pins to their SPI/MPI functionality.

Table 36–1 SPI/MPI Signals

Ball Name	Description	Enable	SPI/MPI Signals
BOOT_CE_N<6>/SPI_CS0_L	Chip select	MPI_CFG[CSENA0] = 1 MPI_CFG[ENABLE] = 1	SPI_CS0_L
BOOT_CE_N<7>/SPI_CS1_L	Chip select	MPI_CFG[CSENA1] = 1 MPI_CFG[ENABLE] = 1	SPI_CS1_L
UART1_CTS_L/SPI_CS2_L	Chip select	MPI_CFG[CSENA2] = 1 MPI_CFG[ENABLE] = 1	SPI_CS2_L
UART1_RTS_L/SPI_CS3_L	Chip select	MPI_CFG[CSENA3] = 1 MPI_CFG[ENABLE] = 1	SPI_CS3_L
UART0_DTR_L/SPI_DO	Transmit data wire	MPI_CFG[ENABLE] = 1	SPI_DO
UART0_DCD_L/SPI_DI	Receive data wire	MPI_CFG[ENABLE] = 1	SPI_DI

For the remainder of the chapter, the SPI/MPI signals are referred to by the SPI/MPI part of their names.

36.2 SPI/MPI Configuration

36.2.1 Clock Generation

The SPI/MPI unit operates in master mode at all times. As such, it is responsible for generating SPI_CK for the SPI/MPI interface, which is accomplished through a simple clock divider.

MPI_CFG[CLKDIV] determines the SPI_CK frequency using the following relation:

$$\text{SPI_CK} = \frac{\text{SCLK}}{2 \times \text{MPI_CFG[CLKDIV]}}$$

In addition to the clock frequency, the SPI/MPI interface has two types of clocking strategies.

- When MPI_CFG[CLK_CONT] = 0, SPI_CK is stopped between SPI/MPI transactions. In this mode, SPI_CS*n*_L is not required to deassert between transactions.
- When MPI_CFG[CLK_CONT] = 1, SPI_CK runs continuously. In this mode, deassertion of SPI_CS*n*_L is the only notification to the slave device that a transaction has completed.

MPI_CFG[IDLELO] specifies the value at which SPI_CK idles when the clock is stopped.

Cavium Confidential For Bowen Yang ALU 02/27/2014

36.2.2 Chip Select Signals

CN71XX supports up to four chip-select signals (SPI_CS<3:0>_L). The active select signal is determined by MPI_TX[CSID].

Two fields in MPI_CFG control the handling of the SPI_CS<3:0>_L signals: [CSLATE] and [CSHI].

- [CSHI] determines whether SPI_CS*n*_L is treated as a low-asserted chip select ([CSHI] = 0) or a high-asserted chip select ([CSHI] = 1).
- [CSLATE] is provided to handle different device setup requirements for chip select.
  - If [CSLATE] = 0, the SPI/MPI engine asserts chip select one-half coprocessor-clock cycle before SPI_DO is driven.
  - If [CSLATE] = 1, chip select asserts coincident with SPI_DO driving.

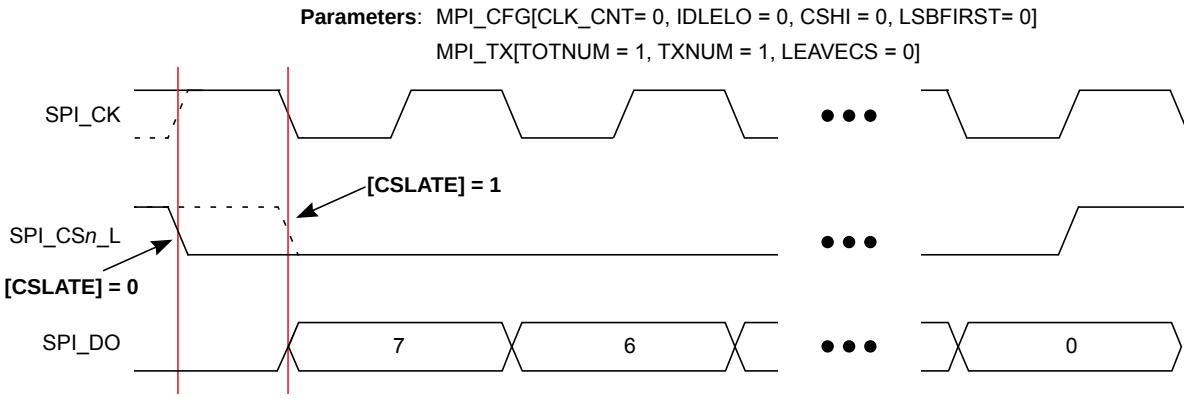
Five fields in MPI_CFG control the enabling of the SPI_CS<3:0>_L signals: [ENABLE], [CSENA3], [CSENA2], [CSENA1], and [CSENA0].

- [CSENA3] determines whether the MPI/SPI engine drives SPI_CS3_L during MPI/SPI transactions.
  - If [CSENA3] = 0, the MPI/SPI engine never drives SPI_CS3_L.
  - If [CSENA3] = 1 and [ENABLE] = 1, then SPI_CS3_L is asserted at the start of every MPI/SPI transaction based on the settings of MPI_CFG[CSHI,CSLATE] and deasserted (or not) based on the value written to MPI_TX[LEAVECS].
- [CSENA2] determines whether the MPI/SPI engine drives SPI_CS2_L during MPI/SPI transactions.
  - If [CSENA2] = 0, the MPI/SPI engine never drives SPI_CS2_L.
  - If [CSENA2] = 1 and [ENABLE] = 1, then SPI_CS2_L is asserted at the start of every MPI/SPI transaction based on the settings of MPI_CFG[CSHI,CSLATE] and deasserted (or not) based on the value written to MPI_TX[LEAVECS].
- [CSENA1] determines whether the SPI/MPI engine drives SPI_CS1_L during SPI/MPI transactions.
  - If [CSENA1] = 0, the SPI/MPI engine never drives SPI_CS1_L.
  - If [CSENA1] = 1 and [ENABLE] = 1, then SPI_CS1_L is asserted at the start of every SPI/MPI transaction based on the settings of MPI_CFG[CSHI,CSLATE] and deasserted (or not) based on the value written to MPI_TX[LEAVECS].
- [CSENA0] determines whether the SPI/MPI engine drives SPI_CS0_L during SPI/MPI transactions.
  - If [CSENA0] = 0, the SPI/MPI engine never drives SPI_CS0_L.
  - If [CSENA0] = 1 and [ENABLE] = 1, then SPI_CS0_L is asserted at the start of every SPI/MPI transaction based on the settings of MPI_CFG[CSHI,CSLATE] and deasserted (or not) based on the value written to MPI_TX[LEAVECS].

A timing diagram displaying the relationship between SPI_CS<1:0>_L and MPI_CFG[CSLATE] is shown in Figure 36–2.

Cavium Confidential For Bowen Yang ALU 04/21/2014

Figure 36–2 SPI_CS<3:0> and MPI_CFG[CSLATE]



36.2.3 Polling/Interrupt-Based Reception

MPI_CFG[INT_ENA] provides a mechanism to interrupt the processor upon completion of an SPI/MPI transaction.

1 = the SPI/MPI engine interrupts at the completion of an SPI/MPI transaction.

0 = software must poll MPI_STS[BUSY] to determine when the SPI/MPI transaction is complete.

36.2.4 Other Fields in MPI_CFG

MPI_CFG[IDLECLKS] specifies the number of idle coprocessor-clock cycles to be inserted between SPI/MPI transactions.

- When cleared to 0, it allows two transactions to follow as closely to each other as the protocol allows (i.e. it does not insert any idle cycles).
- When set to a non-zero value (1, 2, or 3), that many idle coprocessor-clock cycles are inserted between SPI/MPI transactions.

MPI_CFG[LSBFIRST] determines whether the LSB or MSB of a byte is sent first.

MPI_CFG[ENABLE] enables the SPI/MPI engine. When this field is cleared to 0, the engine is disabled.

36.3 SPI/MPI Usage

Sending an SPI/MPI transaction requires the following steps:

1. Configure the interface via MPI_CFG.
2. Write transmission bytes to MPI_DAT0–8 or MPI_WIDE_DAT.
3. Write MPI_TX with transaction information.
4. Wait for interrupt/poll MPI_STS.
5. Read MPI_DAT0–8 or MPI_WIDE_DAT to get received data

Cavium Confidential For Bowen Yang ALU 02/27/2014

36.3.1 MPI_DAT(0..8) and MPI_WIDE_DAT Registers

The MPI_DAT*n* registers hold the bytes to transmit and are written with the received bytes for any transaction performed by the SPI/MPI engine.

The value in MPI_DAT0 is the first byte transmitted, as well as the first register written with received data.

The MPI_WIDE_DAT register provides simultaneous access to MPI_DAT(0..7) and can be used to reduce operations.

36.3.2 Using the MPI_TX Register

The MPI_TX register is written with transaction information to tell the SPI/MPI engine to start a transaction.

- MPI_TX[CSID] specifies the chip-select line to assert for this transaction.
- MPI_TX[TOTNUM] specifies the number of bytes in the transaction (transmit plus receive), while MPI_TX[TXNUM] specifies the number of bytes to transmit. MPI_TX[TOTNUM] should not exceed nine and typically is greater than one.
  - A SPI/MPI transaction intended to write a register in a slave device would typically have [TOTNUM] = [TXNUM].
  - A SPI/MPI transaction intended to read a register in a slave device would typically have [TOTNUM] > [TXNUM].

There are some devices that can support multibyte read/write operations larger than MPI_DAT0–8 can handle. For those devices, setting MPI_TX[LEAVECS] tells the SPI/MPI engine to leave SPI_CS*n*_L asserted after completing the transaction. This allows a subsequent MPI_TX write operation to continue the transaction from the slave device's perspective. This method only works in noncontinuous clocking mode (i.e. MPI_CFG[CLK_CONT] = 0).

36.3.3 Using the MPI_STS Register

Software may read the MPI_STS register to determine the status of the SPI/MPI engine.

- MPI_STS[BUSY] reads 1 if the SPI/MPI engine is currently processing a transaction.
- MPI_STS[RXNUM] reflects the number of bytes received during the transaction. If [BUSY] = 1, [RXNUM] reflects the number of bytes received so far.

Note that the SPI/MPI engine receives bytes the entire time it is shifting. Because of this MPI_STS[RXNUM] always equals MPI_TX[TOTNUM] at the end of a transaction.

Cavium Confidential For Bowen Yang ALU 02/27/2014

36.4 SPI as a Boot Device

Using SPI to boot is selected by the BOOT_ALE<3:1> signals at reset (refer to Table 28–3). When one of SPI16, SPI24, or SPI32 is selected, there must be a SPI device connected to BOOT_CE6_L/SPI_CS0, and that device must have boot code in its lower blocks. SPI boot should not be selected in REMOTE_BOOT mode.

The SPI_CK signal starts two ms after reset is deasserted. The rate of SPI_CK is determined by the mode: SPI16 drives the clock at 25 MHz, while SPI24 and SPI32 use 50 MHz. The hardware then issues an RDSR command (read status register, 0x5), with SPI_CS<0> asserted for up to 1024 bytes, looking for bit<0> to be deasserted (Ready). If the device does not become ready within the 1024 bytes, the boot fails and MIO_EMM_CFG[BOOT_FAIL] is set. Otherwise the hardware issues the READ command (0x3) starting at either address 0 (Authentik mode) or 2000 (not Authentik mode). The number of address bytes sent with the command is determined by the SPI16 (2), SPI24 (3) or SPI32 (4) mode. At least 8K bytes of data (16K + 64 bytes in Authentik mode) is then read in and verified and then executed by core 0. This operation is similar to the eMMC boot operation.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

36.5 Examples

The following four examples show the steps involved in reading and writing the SPI/MPI unit.

36.5.1 Example 1: Reading a Single Byte From Device Address 0x04

Assumptions: device expects first byte transmitted to be a command byte. Bit[7] of the command byte determines whether the transaction is a read (0) or write (1).

1. Write 0x04 to MPI_DAT0
   a. [7] = 0 indicates read (device specific)
   b. [6:0] = 0x04 indicates address 0x04
2. Write MPI_TX with [TOTNUM] = 0x2, [TXNUM] = 0x1, [LEAVECS] = 0, [CSID] = 0

   Meaning: shift for a total of two bytes, transmitting during the first byte only, deassert SPI_CS0_L after the last data bit is received.
3. Read MPI_STS until [BUSY] = 0 (or wait for an interrupt)
4. Read MPI_DAT1 to get the data byte sent from the slave device.

NOTE: MPI_DAT0 contains junk because it was latched while shifting the read command into the device. The second byte shifted from the device is the response to the read command.

36.5.2 Example 2: Writing a Single Byte to Register 0x04

Assumptions: device expects first byte transmitted to be a command byte. Bit[7] of the command byte determines whether the transaction is a read (0) or write (1).

1. Write 0x84 to MPI_DAT0
   a. [7] = 1 indicates write.
   b. [6:0] = 0x04 indicates address 0x04.
2. Write a byte to MPI_DAT1. This is the value to write to address 0x04.
3. Write MPI_TX with [TOTNUM] = 0x2, [TXNUM] = 0x2, [LEAVECS] = 0, [CSID] = 0.

   Meaning: shift for a total of two bytes, transmitting during both bytes, deassert SPI_CS0_L after the last data bit.
4. Read MPI_STS until [BUSY] = 0 (or wait for an interrupt).

Cavium Confidential For Bowen Yang ALU 8/27/2014

36.5.3 Example 3: Writing Ten Bytes to Registers 0x09–0x00

This is a mode used by some devices that allows reading/writing from register *N* down to register 0 with one SPI/MPI command.

The first data byte is for register *N*, the second is for register *N*–1, etc. In this case, an MPI_CS*n*_L deassertion aborts the transaction, so special care is taken here to not deassert MPI_CS*n*_L during the transfer. Note that a maximum of eight bytes can be written at once because there are only nine MPI_DAT*n* registers, and one must be used for the command to start the transaction. So this operation must be split into two transactions.

NOTE: This cannot work in continuous-clocking mode. The coprocessor clock must idle between the two transactions or the data will be shifted out before the second transaction is ready to receive it. In continuous clocking mode, a maximum of eight bytes may be read or written with one SPI/MPI command.

Assumptions: device expects first byte transmitted to be a command byte. Bit[7] of the command byte determines whether the transaction is a read (0) or write (1).

1. Write 0x89 to MPI_DAT0.
   a. [7] = 1 indicates write.
   b. [6:0] = 0x09 indicates address 0x09.
2. Write eight bytes to MPI_DAT1–8. These are the bytes to write to addresses 0x09, 0x08, 0x07, 0x06, 0x05, 0x04, 0x03, and 0x02.
3. Write MPI_TX with [TOTNUM] = 0x9, [TXNUM] = 0x9, [LEAVECS] = 1, [CSID] = 1.

   Meaning: shift for a total of nine bytes, transmitting during all bytes, do not deassert SPI_CS1_L after the completion of the nine bytes.
4. Read MPI_STS until [BUSY] = 0 (or wait for an interrupt).
5. Write two bytes to MPI_DAT0–1. These are the bytes to write to addresses 0x01 and 0x00.
6. Write MPI_TX with [TOTNUM] = 0x2, [TXNUM] = 0x2, [LEAVECS] = 0, [CSID] = 1.

   Meaning: shift for a total of two bytes, transmitting during both bytes, deassert SPI_CS1_L after the last data bit.
7. Read MPI_STS until [BUSY] = 0 (or wait for an interrupt).

Cavium Confidential For Bowen Yang ALD 02/17/2014

36.5.4 Example 4: Reading 17 Bytes From Registers 0x11–0x00

This is the reading equivalent of example 3.

NOTE: This can not work in continuous clocking mode. The coprocessor clock must idle between the two transactions or the data will be shifted out before the second transaction is ready to receive it. In continuous clocking mode, a maximum of eight bytes may be read or written with one SPI/MPI command.

Assumptions: device expects first byte transmitted to be a command byte. Bit[7] of the command byte determines whether the transaction is a read (0) or write (1).

1. Write 0x11 to MPI_DAT0
   a. [7] = 0 indicates read.
   b. [6:0] = 0x11 indicates address 0x11 (17).
2. Write MPI_TX with [TOTNUM] = 0x9, [TXNUM] = 0x1, [LEAVECS] = 1, [CSID] = 0.

   Meaning: shift for a total of nine bytes, transmitting only during the first byte, do not deassert SPI_CS0_L after completion of the nine bytes.
3. Read MPI_STS until [BUSY] = 0 (or wait for an interrupt).
4. Read MPI_DAT1–8 to get the first eight data bytes.

NOTE: MPI_DAT0 contains junk because it was latched while shifting the read command into the device. The second byte shifted from the device is the response to the read command.

5. Write MPI_TX with [TOTNUM] = 0x9, [TXNUM] = 0x0, [LEAVECS] = 0, [CSID] = 0.

   Meaning: shift for a total of nine bytes, never transmitting, deassert SPI_CS0_L after completion of the transaction.
6. Read MPI_STS until [BUSY] = 0 (or wait for an interrupt).
7. Read MPI_DAT0–8 to get the next nine bytes. Because this transaction was simply receiving data from the end of a previously started command, all nine data bytes are valid.

Cavium Confidential For Bowen Yang ALU 01/27/2014

36.6 SPI/MPI Registers

The SPI/MPI registers are listed in Table 36–2.

Table 36–2 SPI/MPI Registers

Register	Address	CSR Type[1]	Detailed Description
MPI_CFG	0x0001070000001000	NCB	See page 1555
MPI_STS	0x0001070000001008	NCB	See page 1557
MPI_TX	0x0001070000001010	NCB	See page 1557
MPI_WIDE_DAT	0x0001070000001040	NCB	See page 1557
MPI_DAT(0..8)	0x0001070000001080 + α * 0x8	NCB	See page 1558

1. NCB-type registers are accessed directly across the I/O Bus.

SPI/MPI Configuration Register MPI_CFG

This register provides configuration for the SPI/MPI interface. Note that some of the SPI/MPI pins are multiplexed with UART and BOOT pins.

SPI_CK — SPI/MPI clock; dedicated pin.

SPI_DI — SPI/MPI input; shared with UART0_DCD_L/SPI_DI; enabled when MPI_CFG[ENABLE] = 1.

SPI_DO — SPI/MPI output; mux to UART0_DTR_L/SPI_DO; enabled when MPI_CFG[ENABLE] = 1.

SPI_CS0_L — Chip select 0, mux to BOOT_CE_<6>_L/SPI_CS0_L pin, enabled when MPI_CFG[CSENA0] = 1 and MPI_CFG[ENABLE] = 1.

SPI_CS1_L — Chip select 1, mux to BOOT_CE_<7>_L/SPI_CS1_L pin, enabled when MPI_CFG[CSENA1] = 1 and MPI_CFG[ENABLE] = 1.

SPI_CS2_L — Chip select 0, mux to UART1_CTS_L/SPI_CS0_L pin, enabled when MPI_CFG[CSENA2] = 1 and MPI_CFG[ENABLE] = 1.

SPI_CS3_L — Chip select 1, mux to UART1_RTS_L/SPI_CS1_L pin, enabled when MPI_CFG[CSENA3] = 1 and MPI_CFG[ENABLE] = 1.

See Table 36–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:29>	—	RAZ	—	—	Reserved
<28:16>	CLKDIV	R/W	0x0	0x0	Clock divisor. SPI_CK = coprocessor clock / (2 × CLKDIV) CLKDIV = coprocessor clock / (2 × SPI_CK)
<15>	CSENA3	R/W	0	0	SPI_CS3_L enable: 0 = UART1_RTS_L/SPI_CS1_L pin is UART pin. 1 = UART1_RTS_L/SPI_CS1_L pin is SPI/MPI pin. SPI_CS1_L drives BOOT_CE_<7>_L/SPI_CS1_L.
<14>	CSENA2	R/W	0	0	SPI_CS2_L enable: 0=UART1_CTS_L/SPI_CS0_L pin is UART pin 1=UART1_CTS_L/SPI_CS0_L pin is SPI/MPI pin SPI_CS0_L drives BOOT_CE_<6>_L/SPI_CS0_L.

Cavium Confidential For Bowen Fang ALD 02/27/2014

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<13>	CSENA1	R/W	0	0	SPI_CS1_L enable: 0 = BOOT_CE_<7>_L/SPI_CS1_L pin is boot-bus pin. 1 = BOOT_CE_<7>_L/SPI_CS1_L pin is SPI/MPI pin. SPI_CS1_L drives BOOT_CE_<7>_L/SPI_CS1_L.
<12>	CSENA0	R/W	0	0	SPI_CS0_L enable: 0=BOOT_CE_<6>_L/SPI_CS0_L pin is boot-bus pin 1=BOOT_CE_<6>_L/SPI_CS0_L pin is SPI/MPI pin SPI_CS0_L drives BOOT_CE_<6>_L/SPI_CS0_L.
<11>	CSLATE	R/W	0	0	SPI_CS*n*_L late. 0 = SPI_CS*n*_L asserts ½ coprocessor-clock cycle before the transaction. 1 = SPI_CS*n*_L asserts coincident with the transaction.
<10>	TRITX	R/W	0	0	Tristate TX. Used only when WIREOR = 1 0 = SPI_DO pin is driven when slave is not expected to be driving. 1 = SPI_DO pin is tristated when not transmitting.
<9:8>	IDLECLKS	R/W	0x0	0x0	Idle clocks. When set, guarantees idle coprocessor-clock cycles between commands.
<7>	CSHI	R/W	0	0	SPI_CS*n*_L high: 1 = SPI_CS*n*_L is asserted high, 0 = SPI_CS*n*_L is asserted low.
<6>	—	RAZ	—	—	Reserved
<5>	INT_ENA	R/W	0	0	SPI/MPI interrupt enable: 0 = polling is required 1 = SPI/MPI engine interrupts at the end of the transaction
<4>	LSBFIRST	R/W	0	0	Shift LSB first: 0 = shift MSB first, 1 = shift LSB first.
<3>	WIREOR	R/W	0	0	Wire-OR DO and DI. 0 = SPI_DO and SPI_DI are separate wires (SPI). SPI_DO pin is always driven. 1 = SPI_DO/DI is all from SPI_DO pin (MPI). SPI_DO pin is tristated when not transmitting. **NOTE:** If WIREOR = 1, SPI_DI pin is not used by the SPI/MPI engine.
<2>	CLK_CONT	R/W	0	0	Clock control. 0 = clock idles to value given by IDLELO after completion of SPI/MPI transaction. 1 = clock never idles, requires SPI_CS*n*_L deassertion/assertion between commands.
<1>	IDLELO	R/W	0	0	Clock idle low/clock invert. 0 = SPI_CK idles high, first transition is high-to-low. 1 = SPI_CK idles low, first transition is low-to-high.
<0>	ENABLE	R/W	0	0	SPI/MPI enable. 0 = UART0_DTR_L/SPI_DO, UART0_DCD_L/SPI_DI, BOOT_CE_<7:6>_L/SPI_CS*x*_L pins are UART/boot-bus pins. 1 = UART0_DTR_L/SPI_DO and UART0_DCD_L/SPI_DI pins are SPI/MPI pins. BOOT_CE_<7:6>_L/SPI_CS<1:0>0_L are SPI/MPI pins if CSENA1/CSENA0 = 1 UART1_CTS_L/SPI_CS2_L is an SPI/MPI pin if CSENA2 = 1 UART1_RTS_L/SPI_CS3_L is an SPI/MPI pin if CSENA3 = 1

Cavium Confidential For Bowen Yang ALL 02/27/2014

SPI/MPI STS Register
MPI_STS

See Table 36–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:13>	—	RAZ	—	—	Reserved
<12:8>	RXNUM	RO/H	0x0	0x0	Number of bytes written for the transaction.
<7:1>	—	RAZ	—	—	Reserved
<0>	BUSY	RO/H	0	0	Busy. 0 = no SPI/MPI transaction in progress. 1 = SPI/MPI engine is processing a transaction.

SPI/MPI Transmit Register
MPI_TX

See Table 36–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:22>	—	WO	—	—	Reserved
<21:20>	CSID	WO	—	0x0	Which CS to assert for this transaction
<19:17>	—	WO	—	—	Reserved
<16>	LEAVECS	WO	—	0	Leave SPI_CS*n*_L asserted. 0 = deassert SPI_CS*n*_L after the transaction is done. 1 = leave SPI_CS*n*_L asserted after the transaction is done.
<15:13>	—	WO	—	—	Reserved
<12:8>	TXNUM	WO	—	0x1	Number of bytes to transmit.
<7:5>	—	WO	—	—	Reserved
<4:0>	TOTNUM	WO	—	0x2	Total number of bytes to shift (transmit and receive).

MPI/SPI Wide Data Register
MPI_WIDE_DAT

This register aliases operations to MPI_DAT(0..7) in a single access. See Table 36–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:0>	DATA	R/W/H	—	—	Data to transmit/receive.

Cavium Confidential For Bowen Yang At U 02/27/2014

SPI/MPI Data Registers
MPI_DAT(0..8)

See Table 36–2 for the address.

Bit Pos	Field Name	Access	Reset Value	Typical Value	Field Description
<63:8>	—	RAZ	—	—	Reserved
<7:0>	DATA	R/W/H	—	—	Data to transmit/receive.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Section 4: OCTEON III Electrical/Mechanical/Physical

This section contains chapters that describe the electrical, mechanical, and physical specifications of the CN71XX. They are the following chapters:

- Electrical Specifications
- AC Specifications
- Mechanical Specifications
- CN71XX Signal Descriptions
- CN71XX Ball Assignments

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 37

Electrical Specifications

This chapter contains the following subjects:

- Overview
- Absolute-Maximum Ratings
- Recommended Operating Conditions
- Power Sequencing
- Power Consumption
- DC Electrical Characteristics

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

This chapter provides absolute maximum ratings, recommended operating conditions, power consumption, and DC characteristics for all I/O types present on CN71XX.

37.1 Absolute-Maximum Ratings

If absolute-maximum ratings are exceeded, the device may fail permanently. Device operation at or above these limits is not recommended. Exposure to absolute-maximum ratings for extended periods of time may also diminish device function. .

Table 37–1 Voltage Absolute Maximum Ratings

Parameter	Description	Min	Max	Unit
VDD	Core supply voltage (0.85V, 0.88V, or 0.92V)	–0.3	0.945	V
VDD_DDR	DDR3 (1.5V or 1.35V) supply voltages	–0.3	1.65/1.49	V
VDD_IO	Miscellaneous I/O supply voltage	–0.3	2.75/3.6	V
VDD_DLM*n*	Dual-lane module supply voltages (0.85V)	–0.3	0.945	V
VDDH_DLM*n*	DLM high-voltage analog supply voltage (1.8V)	–0.3	1.96	V
VDDTX_DLM*n*	DLM transmit supply voltage	–0.3	0.945	V
VDD_DLMC	DLM common reference-clock supply voltage	–0.3	0.945	V
VDD_SMI	SMI/MDIO (2.5V or 3.3V) supply voltage	–0.3	2.75/3.6	V
VDD_RGM	RGMII (1.8V or 2.5V) supply voltage	–0.3	1.98/2.75	V
USB*n*_VDD33	USB (3.3V) supply voltage	–0.3	3.6	V
USB*n*_DVDD	USB 2.0 low-voltage supply	–0.3	0.945	V
USB*n*_VP	USB 3.0 low-voltage supply	–0.3	0.945	V
USB*n*_VPTX	USB 3.0 transmit supply voltage	–0.3	0.945	V
DDR_PLL_VDD	DDR3 PLL supply voltages	–0.3	0.945	V
CORE_PLL_VDD	Core PLL supply voltage	–0.3	0.945	V
SYS_PLL_VDD	Miscellaneous PLL supply voltage	–0.3	0.945	V

37.1.1 Absolute Maximum Storage Temperatures

Table 37–2 shows CN71XX's maximum storage temperature range.

Table 37–2 Absolute Maximum Storage Temperatures

Parameter	Description	Min	Max	Unit
$T_{STORAGE}$	Storage temperature	–40	120	°C

Cavium Confidential For Bowen Yang ALU 02/27/2014

37.2 Recommended Operating Conditions

Table 37–3 shows recommended operating conditions, defining the upper and lower limits between which the device is tested to function.

Table 37–3 Recommended Operating Temperatures

Frequency	Junction Temp. (T_J)		Case Temp. (T_C)		Unit
	Min	Max	Min	Max	
1.6 GHz	0	105	0	88	°C
1.5 GHz	0	105	0	88	°C
1.2 GHz	0	110	0	96	°C
1.0 GHz	0	110	0	100	°C
800 MHz	0	110	0	100	°C

NOTE: The CN71XX chips are tested to the specified case temperatures (T_C). The junction temperature (T_J) is provided for guidance purposes. When you design the system thermal solution, the maximum case temperature must not be exceeded.

37.2.1 Supply Voltages for the Chip-Core Voltage and External Interfaces

Table 37–4 shows CN71XX's recommended supply voltages for the chip-core voltage and external interfaces.

Table 37–4 Recommended Operating Supply Voltages

Parameter	Description	Min	Typical	Max	Unit
VDD	Core supply voltage for 800 MHz and 1.0 GHz	0.82	0.85	0.88	V
VDD	Core supply voltage for 1.2 GHz	0.85	0.88	0.91	V
VDD	Core supply voltage for 1.5 and 1.6 GHz	0.89	0.92	0.95	V
VDD_DDR	DDR3 (1.5V) interface supply voltages	1.425	1.5	1.575	V
VDD_DDR	DDR3L (1.35V) interface supply voltages	1.283	1.35	1.418	V
VDD_IO	Miscellaneous I/O (2.5V) supply voltage	2.375	2.5	2.625	V
VDD_IO	Miscellaneous I/O (3.3V) supply voltage	3.14	3.3	3.46	V
VDD_15	Miscellaneous 1.5V analog supply voltage	1.425	1.5	1.575	V
USB*n*_VDD33[1]	USB (3.3V) supply voltage	3.14	3.3	3.46	V
USB*n*_DVDD[1,2,3]	USB 2.0 low-voltage supply	0.82	0.85	0.935	V
USB*n*_VP[1,3]	USB 3.0 low-voltage supply	0.82	0.85	0.935	V
USB*n*_VPTX[1]	USB 3.0 transmit supply voltage	0.82	0.85	0.935	V
VDD_DLM*n*	DLM analog supply voltages	0.82	0.85	0.935	V
VDDH_DLM*n*	DLM high-voltage analog supply voltages	1.71	1.8	1.89	V
VDDTX_DLM*n*	DLM transmit supply voltage	0.82	0.85	0.935	V
VDD_DLMC	Common DLM analog supply voltage	0.82	0.85	0.935	V
VDD_SMI	SMI/MDIO interface (2.5V) supply voltage	2.375	2.5	2.625	V
VDD_SMI	SMI/MDIO interface (3.3V) supply voltage	3.14	3.3	3.46	V
VDD_RGM	RGMII interface (1.8V) supply voltage	1.71	1.8	1.89	V
VDD_RGM	RGMII interface (2.5V) supply voltage	2.375	2.5	2.625	V

1. If a USB interface is not used, all power-supply pads should be tied to ground.
2. If a USB interface is configured in device mode and does not need to support 2.0 (HS) operation, the USB*n*_DVDD pin can be grounded.
3. For power-supply source information, refer to Table 40–19.

Cavium Confidential For Bowen Yang ALU 02/27/2024

37.2.2 Supply Voltage for the On-Chip PLLs

Table 37–5 shows CN71XX's supply voltage for the on-chip PLLs. Clean PLL supplies PLL_VDD, SYS_PLL_VDD and DDR_PLL_VDD can be filtered versions of an existing DLM supply.

Table 37–5 Power Supplies for Multiple PLLs

Supply	Description	Min	Typical[1]	Max	Unit
CORE_PLL_VDD	Core PLL supply voltage.	0.82	0.85	0.935	V
DDR_PLL_VDD	DDR3/4 PLL supply voltage.	0.82	0.85	0.935	V
SYS_PLL_VDD	Miscellaneous PLL supply voltage.	0.82	0.85	0.935	V

1. Typical voltage matches VDD_DLM*n*..

37.3 Power Sequencing

CN71XX has two power-sequencing modes: default mode and SerDes common-clock mode.

- Default mode uses the PLL_REF_CLK (50 MHz) signal as the PLL clock source.
- SerDes common-clock mode uses the DLMC_REF_CLK1 (must be set as 100 MHz) signal divided-by-2 as the clock source.

The mode is determined at power-up at the leading edge of PLL_DC_OK. The UART0_RTS_L signal is sampled, and a 1 enables default mode and a 0 enables SerDes common-clock mode (see the Note in Section 34.1).

CN71XX has a core voltage, multiple interface-supply voltages, and several PLL voltages. The power-up should be done as specified in Sections 37.3.1 and 37.3.2 and the power-down should be done as specified in Section 37.3.3 to ensure device functionality, stability, and reliability.

Cavium Confidential For Bowen Yang ALU 02/27/2014

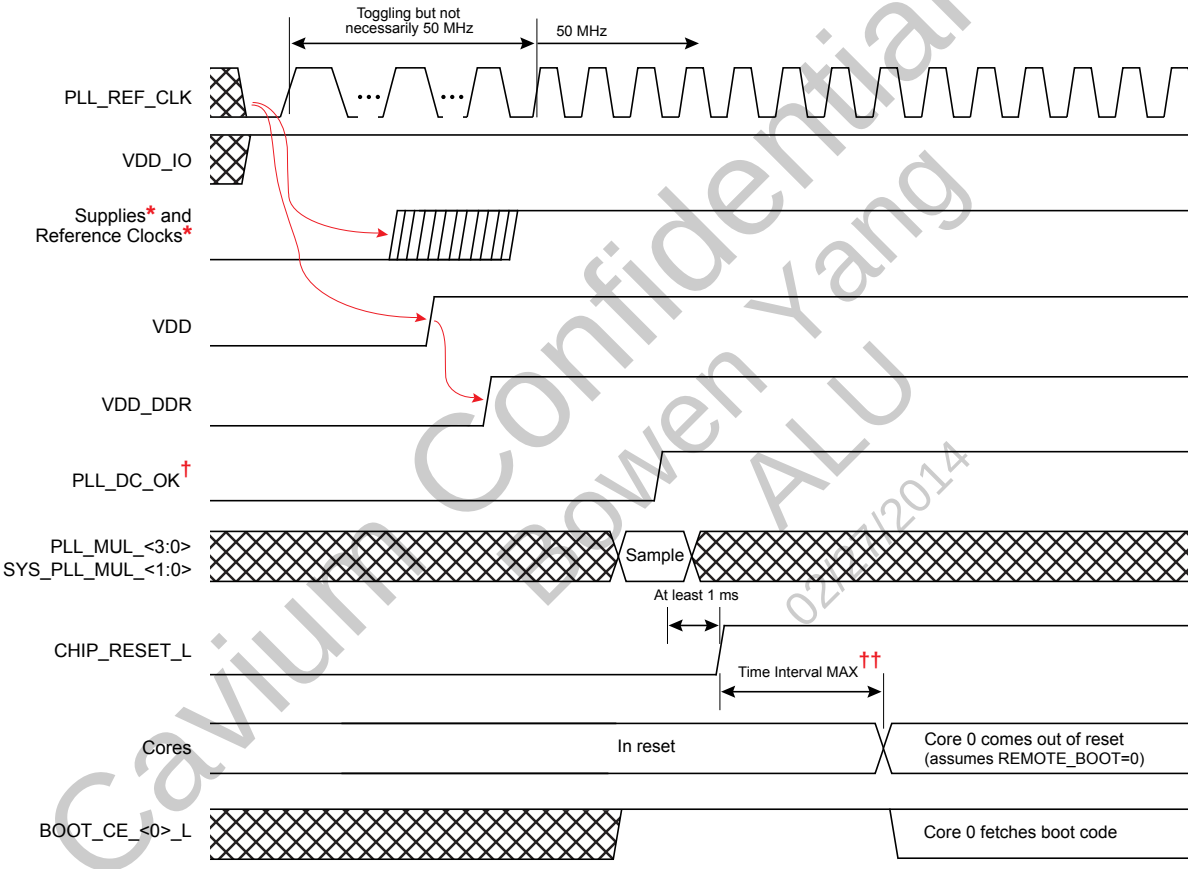
37.3.1 Power Up in Default Mode

Before the start of the power-up sequence, PLL_DC_OK must be deasserted low, and CHIP_RESET_L must be asserted low. The power-up sequence is specified in the following steps.

1. Enable PLL_REF_CLK and VDD_IO. PLL_REF_CLK should be enabled before or at the same time as VDD_IO.
2. Bring up all other supplies and enable reference clocks in any order, except for VDD_DDR.
3. Bring up VDD_DDR after the core VDD has been brought up and is within specifications.
4. Assert PLL_DC_OK high when all power supplies are stable (i.e. meet the specifications). CN71XX samples PLL_MUL_<3:0>and SYS_ PLL_MUL_<1:0> on the rising edge of PLL_DC_OK to determine the internal core-clock and coprocessor-clock multiplier ratios.
5. Deassert signal CHIP_RESET_L high at least 1 ms after PLL_DC_OK is asserted.

Figure 37–2 shows the power-up sequence for default mode.

Figure 37–1 PROPOSED Power Sequencing for Default Mode



† PLL_DC_OK rise time must not exceed 200 ns.

†† MAX (200k core-clock cycles, 150k coprocessor-clock cycles, 152k core-clock cycles + 92k coprocessor-clock cycles)

* Supplies: CORE_PLL_VDD, DDR_PLL_VDD, SYS_PLL_VDD
VDD*_DLM*n*, USBn_*, VDD_RGM, VDD_SMI, VDD_15

Reference Clocks: DLM0_REF_CLK, DLMC_REF_CLK*n*, RGM_REF_CLK,

Cavium Confidential For Bowen Yang ALU 02/27/2014

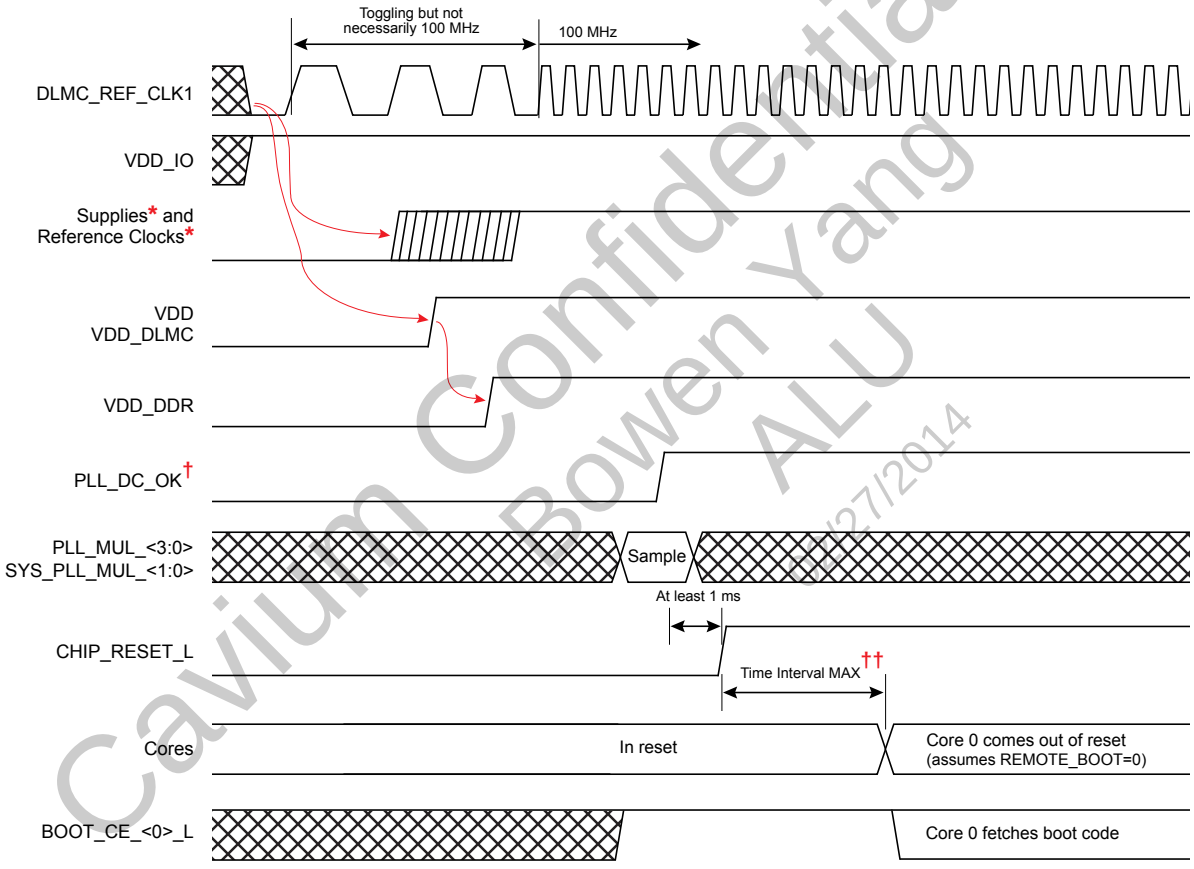
37.3.2 Power Up in SerDes Common-Clock Mode

Before the start of the power-up sequence, PLL_DC_OK must be deasserted low, and CHIP_RESET_L must be asserted low. The power-up sequence is specified in the following steps.

1. Enable DLMC_REF_CLK1 and VDD_IO. DLMC_REF_CLK1 should be enabled before or at the same time as VDD_IO.
2. Bring up all other supplies and enable reference clocks in any order, except for VDD_DDR and VDD_DLMC.
3. Bring up core VDD and VDD_DLMC at the same time.
4. Bring up VDD_DDR after the core VDD has been brought up and is within specifications.
5. Assert PLL_DC_OK high when all power supplies are stable (i.e. meet the specifications). CN71XX samples PLL_MUL_<3:0>and SYS_ PLL_MUL_<1:0> on the rising edge of PLL_DC_OK to determine the internal core-clock and coprocessor-clock multiplier ratios.
6. Deassert signal CHIP_RESET_L high at least 1 ms after PLL_DC_OK is asserted.

Figure 37–2 shows the power-up sequence for SerDes common-clock mode.

Figure 37–2 PROPOSED Power Sequencing for SerDes Common-Clock Mode



† PLL_DC_OK rise time must not exceed 200 ns.

†† MAX (200k core-clock cycles, 150k coprocessor-clock cycles, 152k core-clock cycles + 92k coprocessor-clock cycles)

* Supplies: CORE_PLL_VDD, DDR_PLL_VDD, SYS_PLL_VDD
VDD*_DLM*n*, USBn_*, VDD_RGM, VDD_SMI, VDD_15

Reference Clocks: DLM0_REF_CLK, DLMC_REF_CLK0, RGM_REF_CLK,

Cavium Confidential For Bowen Yang ALU 12/27/2014

37.3.3 Power Down

Deassert PLL_DC_OK first. Then the power supplies can be removed in any order without any restrictions.

NOTE: When powering down, the reference clock must not be stopped until PLL_DC_OK has been deasserted.

37.4 Power Consumption

Table 37–6 lists the worst-case power consumption for CN71XX.

Table 37–6 CN71XX Maximum Core Power Specification[1]

Frequency @Voltage (Coprocessor Clock)	Parameter	Description	Number of Cores			Units
			2	3	4	
800 MHz @ 0.85V (coprocessor clock = 400 MHz max)	ICC	I_{CC} for VDD supply	2.9	3.1	3.3	A
1.0 GHz @ 0.85V (coprocessor clock = 400 MHz max)	ICC	I_{CC} for VDD supply	3.1	3.4	3.7	A
1.2 GHz @ 0.88V (coprocessor clock = 400 MHz max)	ICC	I_{CC} for VDD supply	4.5	4.9	5.4	A
1.5 GHz @ 0.92V (coprocessor clock = 600 MHz max)	ICC	I_{CC} for VDD supply	5.4	5.8	6.3	A
1.6 GHz @ 0.92V (coprocessor clock = 600 MHz max)	ICC	I_{CC} for VDD supply	5.5	6.0	6.5	A

1. This table is preliminary.

Table 37–7 shows typical DDR3 memory-interface supply current for use in thermal design. Table 37–8 shows worst-case DDR3 memory-interface supply current for use in power-supply design. All the current numbers assume address/command terminations to 39Ω, in addition to ODT being enabled on the controller and SDRAMs according to JEDEC recommended matrices. CN71XX has one 36-bit controller.

Table 37–7 Typical DDR3 Memory-Interface Supply Current[1]

Parameter	Description	Data Rate (MHz)			Units
		800	1066	1333	
I_{DDR} (@1.5V)	I_{CC} for DDR3 supply (one rank)	0.4	0.4	0.4	A
I_{DDR} (@1.5V)	I_{CC} for DDR3 supply (two ranks)	0.4	0.4	0.5	A
I_{DDR} (@1.35V)	I_{CC} for DDR3L supply (one rank)	0.4	0.4	0.4	A
I_{DDR} (@1.35V)	I_{CC} for DDR3L supply (two ranks)	0.4	0.4	0.5	A

1. For thermal design, use values in this table along with core current specified in Table 37–6.

Table 37–8 Worst-Case DDR3 Memory-Interface Supply Current[1]

Parameter	Description	Data Rate (MHz)			Units
		800	1066	1333	
I_{DDR} (@1.5V)	I_{CC} for DDR3 supply (one rank)	0.7	0.8	0.9	A
I_{DDR} (@1.5V)	I_{CC} for DDR3 supply (two ranks)	0.9	0.9	1.0	A
I_{DDR} (@1.35V)	I_{CC} for DDR3L supply (one rank)	0.7	0.7	0.8	A
I_{DDR} (@1.35V)	I_{CC} for DDR3L supply (two ranks)	0.8	0.9	0.9	A

1. For power-supply design, use values in this table along with core current specified in Table 37–6.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 37–9 shows CN71XX's miscellaneous supply current.

Table 37–9 Miscellaneous Supply Current

Parameter	Description	Supply Current	Units
VDD_IO	Miscellaneous I/O supply (3.3V)	100	mA
VDD_IO	Miscellaneous I/O supply (2.5V)	65	mA
CORE_PLL_VDD	Core PLL supply voltage	15	mA
DDR_PLL_VDD	DDR3 PLL supply voltage	15	mA
SYS_PLL_VDD	Miscellaneous PLL supply voltage	15	mA
VDD_DLM0/ VDDTX_DLM0	DLM0 sum of analog supply voltage plus transmit supply voltage	125	mA
VDD_DLM1/ VDDTX_DLM1	DLM1 sum of analog supply voltage plus transmit supply voltage	100	mA
VDD_DLM2/ VDDTX_DLM2	DLM2 sum of analog supply voltage plus transmit supply voltage	125	mA
VDD_DLMC	DLM common reference-clock supply voltage	25	mA
VDDH_DLM0	DLM0 high-voltage analog supply voltage	35	mA
VDDH_DLM1	DLM1 high-voltage analog supply voltage	30	mA
VDDH_DLM2	DLM2 high-voltage analog supply voltage	35	mA
VDD_15	Miscellaneous analog supply voltage (1.5V)	5	mA
VDD_RGM	RGMII supply (2.5V)	20	mA
VDD_SMI	SMI supply (3.3V)	8	mA
VDD_SMI	SMI supply (2.5V)	5	mA
USB0_VDD33	USB0_VDD33 supply voltage	28	mA
USB1_VDD33	USB1_VDD33 supply voltage	28	mA
USB0_VP	USB0_VP USB 3.0 low-voltage supply	51	mA
USB1_VP	USB1_VP USB 3.0 low-voltage supply	51	mA
USB0_DVDD	USB0_DVDD USB 2.0 low-voltage supply	14	mA
USB1_DVDD	USB1_DVDD USB 2.0 low-voltage supply	14	mA
USB0_VPTX	USB0_VPTX 3.0 transmitter supply voltage	22	mA
USB1_VPTX	USB1_VPTX 3.0 transmitter supply voltage	22	mA

Table 37–10 shows CN71XX's DLM interface thermal power per DLM.

Table 37–10 DLM Interface Thermal Supply Current

Dual-Lane Module	Supply @ 0.85V	Supply @ 1.8V	Units
DLM0	100	30	mA
DLM1	80	25	mA
DLM2	100	30	mA

Cavium Confidential For ALL Bowen Yang 02/21/2014

37.5 DC Electrical Characteristics

The following tables describe the DC electrical characteristics for different types and combinations of input receivers and output drivers on the CN71XX chip.

37.5.1 DDR3 Bidirectional I/O for the DDR3 Memory Interface

Table 37–11 lists CN71XX's DDR3 memory-interface electrical characteristics for input receivers.

NOTE: Additional nominal settings will be supported, and detailed specifications will be provided in future releases.

Table 37–11 Input-Receiver Specifications

Symbol	Parameter	Min	Nom	Max	Unit
Single-Ended Specifications					
VDD_DDR	DDR3 interface supply voltage	1.425	1.5	1.575	V
VDD_DDR	DDR3L interface supply voltage	1.283	1.35	1.418	V
VIH (AC)	AC input logic high (single-ended)	VDD_DDR/2 + 0.138	—	VDD_DDR + 3	V
VIL (AC)	AC input logic low (single-ended	–0.3	—	VDD_DDR/2 – 0.138	V
Differential Signals					
VIH_{diff} (AC)	AC input logic high (differential)	0.275	—	—	V
VIL_{diff} (AC)	AC input logic low (differential)	—	—	–0.275	V

The DDR3 output-driver resistance follows the DC specification in Table 37–12 for a 34Ω nominal setting.

Table 37–12 Output-Driver Specifications

V_{OUT}	Min	Nom	Max	Unit
Pull-down Resistor				
0.2 × VDD	20.4	34	37.4	Ω
0.5 × VDD	30.6	34	37.4	Ω
0.8 × VDD	30.6	34	47.6	Ω
Pull-up Resistor				
0.2 × VDD	30.6	34	47.6	Ω
0.5 × VDD	30.6	34	37.4	Ω
0.8 × VDD	20.4	34	37.4	Ω

Cavium Confidential For Bowen Yang ALU 02/27/2014

37.5.2 3.3V CMOS Bidirectional and Point-to-Point I/O for the Miscellaneous Interface

Table 37–13 lists CN71XX's 3.3V/2.5V CMOS bidirectional and point-to-point I/O for the miscellaneous interfaces' electrical characteristics.

Table 37–13 3.3V/2.5V CMOS Point-to-Point I/O for Miscellaneous Interfaces

Symbol	Parameter	Min	Nom	Max	Unit
VDD_IO @ 3.3V	Miscellaneous I/O supply voltage @3.3V	3.14	3.3	3.46	V
VDD_IO @ 2.5V	Miscellaneous I/O supply voltage @2.5V	2.375	2.5	2.625	V
VIH (DC @ 3.3V)	DC input logic high @ 3.3V	1.2	—	—	V
VIL (DC @ 3.3V)	DC input logic low @ 3.3V	—	—	0.66	V
VIH (DC @ 2.5V)	DC input logic high @ 2.5V	1.0	—	—	V
VIL (DC @ 2.5V)	DC input logic low @ 2.5V	—	—	0.5	V
VOH (DC @ 3.3V)[1]	DC output logic high @ 3.3V	VDD_IO – 0.7	—	—	V
VOL (DC @ 3.3V)[1]	DC output logic low @ 3.3V	—	—	0.7	V
VOH (DC @ 2.5V)[1]	DC output logic high @ 2.5V	VDD_IO – 0.7	—	—	V
VOL (DC @ 2.5V)[1]	DC output logic low @ 2.5V	—	—	0.7	V
I_{LEAK} (DC)	DC input leakage current	—	—	250	µA

1. Measured at I_{OUT} = 10 mA.

37.5.3 1.8V/2.5V CMOS Bidirectional I/O for RGMII Interface

Table 37–14 lists CN71XX's 1.8V/2.5V CMOS bidirectional I/O for the RGMII interface's electrical characteristics.

Table 37–14 2.5V/3.3V CMOS for RGMII/MII/SMI/MDIO Interface

Symbol	Parameter	Min	Nom	Max	Unit
VDD_RGM (2.5V)	Interface supply voltage	2.375	2.5	2.625	V
VDD_RGM (1.8V)	Interface supply voltage	1.71	1.8	1.89	V
VIH (DC @ 2.5V)	DC input logic high	1.2	—	—	V
VIL (DC @ 2.5V)	DC input logic low	—	—	0.5	V
VIH (DC @ 1.8V)	DC input logic high	1.3	—	—	V
VIL (DC @ 1.8V)	DC input logic low	—	—	0.5	V
VOH (DC @ 2.5V)[1]	DC output logic high	VDD_RGM – 0.7	—	—	V
VOL (DC @ 2.5V)[1]	DC output logic low	—	—	0.7	V
VOH (DC @ 1.8V)[1]	DC output logic high	VDD_RGM – 0.5	—	—	V
VOL (DC @ 1.8V)[1]	DC output logic low	—	—	0.5	V
I_{LEAK} (DC)	DC input leakage current	—	—	1	µA

1. Measured at I_{OUT} = 10 mA.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Chapter 38

AC Characteristics

This chapter contains the following subjects:

- Input Clocks
- DDR3 SDRAM Interface Timing Specifications
- DLM Common Reference Clock
- PCIe Interface (Gen1) Timing Specifications
- PCIe Interface (Gen2) Timing Specifications
- SGMII Interface Timing Specifications
- RXAUI Interface Timing Specifications
- SATA Interface Timing Specifications
- RGMII Interface
- EEPROM Interface
- Boot Bus Interface
- EMMC Interface
- JTAG Interface
- TDM/PCM Interface
- MPI/SPI Interface
- TWSI Interface
- SMI/MDIO Interface
- USB Interface

Cavium Confidential For Bowen Yang ALU 02/27/2014

38.1 Input Clocks

38.1.1 Inputs for Core/Coprocessor/DDR Reference-Clock

Table 38–1 lists CN71XX's non-SerDes reference-clock input electrical characteristics.

The 50-MHz core/coprocessor/DDR reference clock can be internally derived from the DLM common reference clock inputs via UART0_RTS_L (see Table 40–16).

Restrictions for using the DLM common reference clock inputs to internally derive the core 50-MHz reference clock:

- **The DLM common reference clock frequency must be 100 MHz.**
- **The DLM common reference clock cannot be spread spectrum modulated.**

Table 38–1 Core/Coprocessor/DDR Reference-Clock

Clock Name	Frequency	Frequency Tolerance	Duty Cycle	Phase Jitter (rms)[1]	Phase Jitter (p-to-p)	Edge Rate [2]	Voltage Levels
PLL_REF_CLK	50 MHz	±150 ppm	60/40	4 ps	110 ps	4 ns	0–3.3V single-ended

1. From 12 kHz to 20 MHz.
2. Edge rate, defined as 20%–80% value, specifies worst case.

Cavium Confidential For Bowen Yang ALU 02/27/2014

38.2 DDR3 SDRAM Interface Timing Specifications

38.2.1 DDR3 AC Test Load

Standard JEDEC test load (Figure 87 of JEDEC Standard JESD79-3A) is assumed.

38.2.2 DDR3 Output-Clock Timing Specifications

Table 38–2 lists CN71XX's DDR3 output-clock timing specifications.

Table 38–2 DDR3 Output-Clock Timing Specifications

Parameter	Description	Min	Typ	Max	Units
667 MT/s					
T_{CK} (average) [1]	Average clock period	—	3.0	—	ns
T_{CH} (average) [2]	Average high-pulse width	1.41	1.50	1.59	ns
T_{CL} (average) [3]	Average low-pulse width	1.41	1.50	1.59	ns
T_{JITPER}	Clock-period jitter	–0.10	—	0.10	ns
800 MT/s					
T_{CK} (average) [1]	Average clock period	—	2.5	—	ns
T_{CH} (average) [2]	Average high-pulse width	1.18	1.25	1.33	ns
T_{CL} (average) [3]	Average low-pulse width	1.18	1.25	1.33	ns
T_{JITPER}	Clock-period jitter	–0.10	—	0.10	ns
1066 MT/s					
T_{CK} (average) [1]	Average clock period	—	1.875	—	ns
T_{CH} (average) [2]	Average high-pulse width	0.88	0.94	0.99	ns
T_{CL} (average) [3]	Average low-pulse width	0.88	0.94	0.99	ns
T_{JITPER}	Clock-period jitter	–0.09	—	0.09	ns
1333 MT/s					
T_{CK} (average) [1]	Average clock period	—	1.5	—	ns
T_{CH} (average) [2]	Average high-pulse width	0.71	0.75	0.80	ns
T_{CL} (average) [3]	Average low-pulse width	0.71	0.75	0.80	ns
T_{JITPER}	Clock-period jitter	–0.08	—	0.08	ns

1. T_{CK} (average) is the average clock period across any consecutive 200-cycle window (rising edge to rising edge).
2. T_{CH} (average) is the average high pulse width across any consecutive 200-cycle window (rising edge to falling edge).
3. T_{CL} (average) is the average low pulse width across any consecutive 200-cycle window (falling edge to rising edge).

Cavium Confidential For Bowen Yang ALU 02/27/2014

38.2.3 DDR3 Output Timing Specifications

Table 38–3 lists CN71XX's DDR3 output timing specifications. Figure 38–1 shows the relationship between DDR_CK*n*_P/N and the DDR commands. Figure 38–2 shows the write timing.

Table 38–3 DDR3 Output Timing Specifications

Parameter	Description	Min	Typ	Max	Units
667 MT/s					
T_A [1]	Address output delay from DDR_CK_*n*_N	–0.125	0.000	0.125	ns
T_{CMD} [1]	Command output delay from DDR_CK_*n*_N	–0.125	0.000	0.125	ns
T_{BA} [1]	Bank address output delay from DDR_CK_*n*_N	–0.125	0.000	0.125	ns
T_{DQSH} [2]	DDR_DQS_*n*_P differential high-pulse width	1.363	1.500	1.643	ns
T_{DQSL} [2]	DDR_DQS_*n*_P differential low-pulse width	1.363	1.500	1.643	ns
T_{DQSS} [3]	DDR_DQS_*n*_P/N rising to DDR_CK_*n*_P/N rising	–0.125	0.000	0.125	ns
T_{DSS} [3]	DDR_DQS_*n*_P/N falling to DDR_CK_*n*_P/N rising	1.355	1.500	1.645	ns
T_{DSH} [3]	DDR_CK_*n*_P/N rising to DDR_DQS_*n*_P/N falling	1.355	1.500	1.645	ns
T_{DQ} [2,4]	DDR_DQ*n* to DDR_DQS_*n*_P/N output delay	0.625	0.750	0.875	ns
800 MT/s					
T_A [1]	Address output delay from DDR_CK_*n*_N	–0.125	0.000	0.125	ns
T_{CMD} [1]	Command output delay from DDR_CK_*n*_N	–0.125	0.000	0.125	ns
T_{BA} [1]	Bank address output delay from DDR_CK_*n*_N	–0.125	0.000	0.125	ns
T_{DQSH} [2]	DDR_DQS_*n*_P differential high-pulse width	1.128	1.25	1.375	ns
T_{DQSL} [2]	DDR_DQS_*n*_P differential low-pulse width	1.128	1.25	1.375	ns
T_{DQSS} [3]	DDR_DQS_*n*_P/N rising to DDR_CK_*n*_P/N rising	–0.125	0.000	0.125	ns
T_{DSS} [3]	DDR_DQS_*n*_P/N falling to DDR_CK_*n*_P/N rising	1.105	1.25	1.395	ns
T_{DSH} [3]	DDR_CK_*n*_P/N rising to DDR_DQS_*n*_P/N falling	1.105	1.25	1.395	ns
T_{DQ} [2,4]	DDR_DQ*n* to DDR_DQS_*n*_P/N output delay	0.500	0.625	0.750	ns
1066 MT/s					
T_A [1]	Address output delay from DDR_CK_*n*_N	–0.125	0.000	0.125	ns
T_{CMD} [1]	Command output delay from DDR_CK_*n*_N	–0.125	0.000	0.125	ns
T_{BA} [1]	Bank address output delay from DDR_CK_*n*_N	–0.125	0.000	0.125	ns
T_{DQSH} [2]	DDR_DQS_*n*_P differential high-pulse width	0.839	0.938	1.041	ns
T_{DQSL} [2]	DDR_DQS_*n*_P differential low-pulse width	0.839	0.938	1.041	ns
T_{DQSS} [3]	DDR_DQS_*n*_P/N rising to DDR_CK_*n*_P/N rising	–0.125	0.000	0.125	ns
T_{DSS} [3]	DDR_DQS_*n*_P/N falling to DDR_CK_*n*_P/N rising	0.793	0.938	1.083	ns
T_{DSH} [3]	DDR_CK_*n*_P/N rising to DDR_DQS_*n*_P/N falling	0.793	0.938	1.083	ns
T_{DQ} [2,4]	DDR_DQ*n* to DDR_DQS_*n*_P/N output delay	0.369	0.469	0.569	ns

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 38–3 DDR3 Output Timing Specifications (Continued)

Parameter	Description	Min	Typ	Max	Units
1333 MT/s					
T_A [1]	Address output delay from DDR_CK_*n*_N	–0.125	0.000	0.125	ns
T_{CMD} [1]	Command output delay from DDR_CK_*n*_N	–0.125	0.000	0.125	ns
T_{BA} [1]	Bank address output delay from DDR_CK_*n*_N	–0.125	0.000	0.125	ns
T_{DQSH} [2]	DDR_DQS_*n*_P differential high-pulse width	0.667	0.750	0.837	ns
T_{DQSL} [2]	DDR_DQS_*n*_P differential low-pulse width	0.667	0.750	0.837	ns
T_{DQSS} [3]	DDR_DQS_*n*_P/N rising to DDR_CK_*n*_P/N rising	–0.125	0.000	0.125	ns
T_{DSS} [3]	DDR_DQS_*n*_P/N falling to DDR_CK_*n*_P/N rising	0.605	0.750	0.895	ns
T_{DSH} [3]	DDR_CK*n*_P/N rising to DDR_DQS_*n*_P/N falling	0.605	0.750	0.895	ns
T_{DQ} [2,4]	DDR_DQ*n* to DDR_DQS_*n*_P/N output delay	0.285	0.375	0.465	ns

1. Refer to Figure 38–1. For more details, see the OCTEON III DDR Application Note.
2. Refer to Figure 38–2. For more details, see the OCTEON III DDR Application Note.
3. These specifications are defined at the CN71XX balls. On the memory side, they will be met after the write-levelling procedure.
4. The latching DDR_DQS_*n*_P/N strobe is delayed by 90° (nominal delay) inside the CN71XX.

Figure 38–1 CK vs Command Timing Diagram

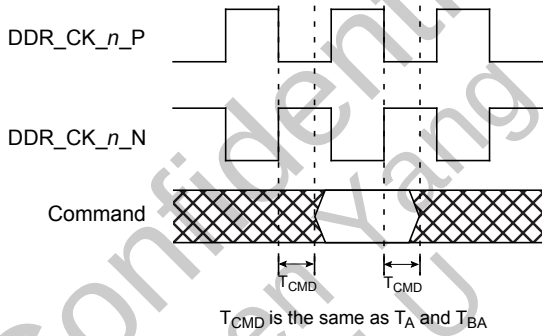
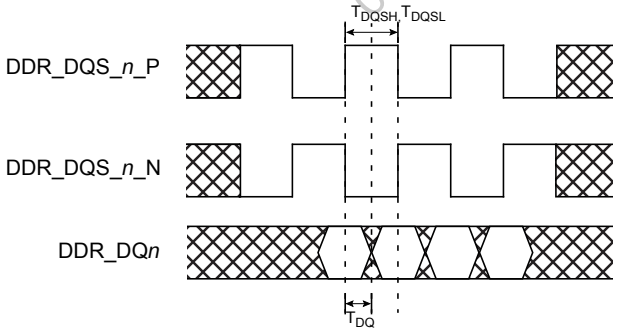


Figure 38–2 WRITE Timing Diagram



Cavium Confidential For Bowen Yang ALU 02/27/2014

38.2.4 DDR3 Input Timing Specifications

Table 38–4 lists CN71XX's DDR3 input timing specifications.

Table 38–4 DDR3 Input Timing Specifications

Parameter	Description	Min	Typ	Max	Units
667 MT/s					
T_{DS} [1]	DDR_DQ*n* setup time relative to DDR_DQS*n*_P/N	–0.585	0.000	—	ns
T_{DH} [2]	DDR_DQ*n* hold time relative to DDR_DQS*n*_P/N	0.895	1.499	—	ns
800 MT/s					
T_{DS} [1]	DDR_DQ*n* setup time relative to DDR_DQS*n*_P/N	–0.460	0.000	—	ns
T_{DH} [2]	DDR_DQ*n* hold time relative to DDR_DQS*n*_P/N	0.770	1.250	—	ns
1066 MT/s					
T_{DS} [1]	DDR_DQ*n* setup time relative to DDR_DQS*n*_P/N	–0.304	0.000	—	ns
T_{DH} [2]	DDR_DQ*n* hold time relative to DDR_DQS*n*_P/N	0.614	0.938	—	ns
1333 MT/s					
T_{DS} [1]	DDR_DQ*n* setup time relative to DDR_DQS*n*_P/N	–0.210	0.000	—	ns
T_{DH} [2]	DDR_DQ*n* hold time relative to DDR_DQS*n*_P/N	0.520	0.750	—	ns

1. Internal to CN71XX, DDR_DQS*n*_P/N is delayed by 90° (nominal). A negative setup time means that DDR_DQ*n* can be valid later than the DDR_DQS*n*_P/N latching edge.
2. Internal to CN71XX, DDR_DQS*n*_P/N is delayed by 90°. The total DDR_DQ*n* valid window is equal to $T_{DS} + T_{DH}$ (T_{DS} being negative).

Cavium Confidential For Bowen Yang ALU 02/27/2014

38.3 SerDes Timing Specifications

The CN71XX contains three SerDes dual-lane modules (DLMs) that are shared between the GMX (QSGMII/SGMII/RXAUI), PEM (PCIe), and SATA interfaces. The following sections contain the SerDes interfaces timing specifications.

38.3.1 DLM Common Reference Clock

The three DLMs have their clocks established in the following way:

- Derived from one of the two common DLM reference clocks DLMC_REF_CLK<1:0>_P/N (see Table 40–4) via GSER(0)_DLM(0..2)_REFCLK_SEL[REFCLK_SEL]. Table 38–5 lists CN71XX's SerDes common reference-clock input electrical characteristics.
- DLM0 only can also choose to use the DLM0_REF_CLK_P/N signals.

There are several restrictions for using the DLM common reference-clock input:

- The common reference clock should be chosen according to the specifications listed in table Table 38–5.
- PCIe requires a 100-MHz reference clock as per PCIe-SIG requirements. See Table 38–9 for more information on particular specifications.
- In links where a spread-spectrum-modulated reference clock is used, both ends of the link must share the same master reference clock source, except in the case of SATA links. When the common reference clock is spread-spectrum-modulated, this means that all DLMs using the common clock are SSC modulated, and therefore share this requirement.

Table 38–5 SerDes Common Reference-Clock Inputs

Clock Name	Frequency	Frequency Tolerance	Duty Cycle	Phase Jitter (rms)[1]	Phase Jitter (p-to-p)	Edge Rate[2]	V_{HIGH} Levels			V_{LOW} Levels		
							Min	Nom	Max	Min	Nom	Max
DLMC_REF_CLK<1:0>_N/P	100 MHz [3]	Note[4]	60/40	Note[4]	Note[4]	840ps	325mV	700mV	850mV	–150mV	0V	175mV
DLMC_REF_CLK<1:0>_N/P	125 MHz	±100 ppm	60/40	2 ps	44 ps	840ps	325mV	700mV	850mV	–150mV	0V	175mV
DLMC_REF_CLK<1:0>_N/P	156.25 MHz	±100 ppm	60/40	2 ps	44 ps	840ps	325mV	700mV	850mV	–150mV	0V	175mV

1. From 12 kHz to 20 MHz.
2. Edge rate is defined as 20%–80% value.
3. If PCIe EP mode is used, DLMC_REF_CLK0_N/P must be 100 MHz.
4. Use the specification for each individual standard (PCIe/SGMII/QSGMII/RXAUI) as specified in Tables 38–6, 38–9, 38–12, and 38–15.

Cavium Confidential For Bowen Yang ALU 02/27/2014

38.3.2 PCIe Interface (Gen1) Timing Specifications

38.3.2.1 PCIe (Gen1) Clock Input Timing

Table 38–6 lists CN71XX's DLM reference-clock input (PCIe Gen1) electrical characteristics.

Table 38–6 DLM Reference-Clock Input: PCIe (Gen1)

Clock Name	Frequency	Frequency Tolerance	Duty Cycle	Phase Jitter (rms)	Phase Jitter (p-to-p)	Edge Rate[1]	CN71XX V_{HIGH} Levels			V_{LOW} Levels		
							Min	Nom	Max	Min	Nom	Max
DLMC_REF_CLK*n*_N/P	100 MHz	±300 ppm[2]	60/40	see spec[3]	see spec[3]	840ps	325mV	700mV	850mV	–150mV	0V	175mV

1. Edge rate is defined as 20%–80% value.
2. In links where a spread spectrum-modulated reference clock is used, both ends of the link must share the same master reference-clock source.
3. Refer to PCIe Base 2.1 Specifications Under Common Clock Architecture, Table 4-16.

38.3.2.2 PCIe (Gen1) AC Specifications

Table 38–7 shows the PCIe TX specifications, and Table 38–8 shows the PCIe RX specifications.

Table 38–7 PCIe (Gen1) TX Specifications

Parameter [1]	Min	Typical	Max	Units
$V_{TX\text{-}DIFF\ pp}$	0.8	1	1.2	V
$V_{TX\text{-}DE\text{-}RATIO}$	–3.0	–3.5	–4.0	dB
$T_{TX\text{-}EYE}$	0.75	—	—	UI (400 ps)
$Z_{TX\text{-}DIFF\text{-}DC}$	80	—	120	Ω

1. For an explanation of the PCIe parameters, refer to the PCI Express Base Specification, Rev. 1.1, pp 222–230.

Table 38–8 PCIe (Gen1) RX Specifications

Parameter [1]	Min	Typical	Max	Units
$V_{RX\text{-}DIFF\ pp}$	0.175	—	1.2	V
$T_{RX\text{-}EYE}$	0.4	—	—	UI
$Z_{RX\text{-}DIFF\text{-}DC}$	80	—	120	Ω

1. For an explanation of the PCIe parameters, refer to the PCI Express Base Specification, Rev. 1.1, pp 222–230.

Cavium Confidential For Bowen Yang ALD 02/27/2014

38.3.3 PCIe Interface (Gen2) Timing Specifications

38.3.3.1 PCIe (Gen2) Clock Input Timing

Table 38–9 lists CN71XX's DLM reference-clock input (PCIe Gen2) electrical characteristics.

Table 38–9 DLM Reference-Clock Input: PCIe (Gen2)

Clock Name	Frequency	Frequency Tolerance	Duty Cycle	Phase Jitter (rms)	Phase Jitter (p-to-p)	Edge Rate[1]	V_{HIGH} Levels			V_{LOW} Levels		
							Min	Nom	Max	Min	Nom	Max
DLMC_REF_CLK*n*_N/P	100 MHz	±300 ppm[2]	60/40	see spec[3]	see spec[3]	840ps	325mV	700mV	850mV	–150mV	0V	175mV

1. Edge rate is defined as 20%–80% value.
2. In links where a spread spectrum-modulated reference clock is used, both ends of the link must share the same master reference-clock source.
3. Refer to PCIe Base 2.1 Specifications Under Common Clock Architecture, Table 4-16.

38.3.3.2 PCIe (Gen2) AC Specifications

Table 38–10 shows the PCIe TX specifications, and Table 38–11 shows the PCIe RX specifications.

Table 38–10 PCIe (Gen2) TX Specifications

Parameter [1]	Min	Typical	Max	Units
$V_{TX\text{-}DIFF\ pp}$	0.8	1	1.2	V
$V_{TX\text{-}DE\text{-}RATIO}$	–3.0	–3.5	–4.0	dB
$T_{TX\text{-}EYE}$	0.75	—	—	UI (200 ps)
$Z_{TX\text{-}DIFF\text{-}DC}$	80	—	120	Ω

1. For an explanation of the PCIe parameters, refer to the PCI Express Base Specification, Rev. 2.0, pp 241–272.

Table 38–11 PCIe (Gen2) RX Specifications

Parameter [1]	Min	Typical	Max	Units
$V_{RX\text{-}DIFF\ pp}$	0.12	—	1.2	V
$T_{RX\text{-}EYE}$	0.4	—	—	UI
$Z_{RX\text{-}DIFF\text{-}DC}$	80	—	120	Ω

1. For an explanation of the PCIe parameters, refer to the PCI Express Base Specification, Rev. 2.0, pp 241–272.

Cavium Confidential For Bowen Yang ALU 02/27/2014

38.3.4 SGMII Interface Timing Specifications

38.3.4.1 SGMII Clock Input Timing

Table 38–12 lists CN71XX's DLM reference-clock input (SGMII) electrical characteristics.

Table 38–12 DLM Reference-Clock Input: SGMII

Clock Name	Frequency	Frequency Tolerance	Duty Cycle	Phase Jitter (rms)[1]	Phase Jitter (p-to-p)	Edge Rate[2]	V_{HIGH} Levels			V_{LOW} Levels		
							Min	Nom	Max	Min	Nom	Max
DLMC_REF_CLK*n*_N/P	100 MHz	±100 ppm[3]	60/40	3 ps	70 ps	840ps	325mV	700mV	850mV	–150mV	0V	175mV
DLM0_REF_CLK_N/P	100 MHz	±100 ppm[3]	60/40	3 ps	70 ps	840ps	325mV	700mV	850mV	–150mV	0V	175mV

1. From 12 kHz to 20 MHz.
2. Edge rate is defined as 20%–80% value.
3. In links where a spread spectrum-modulated reference clock is used, both ends of the link must share the same master reference-clock source.

38.3.4.2 SGMII AC Specifications

Table 38–13 shows the SGMII TX specifications, and Table 38–14 shows the SGMII RX specifications.

Table 38–13 SGMII TX Specifications

Parameter[1]	Min	Typical	Max	Units
VOD	0.150	0.325	0.400	V
Ro	40	—	60	Ω
Delta Ro	—	—	10	%
Delta VOD	—	—	25	mV
Delta VOS	—	—	25	mV
Duty cycle @ 625	48	—	52	%

1. Refer to Serial-GMII Specification Revision 1.8 for parameter definitions.

Table 38–14 SGMII RX Specifications

Parameter	Min	Typical	Max	Units
V_I [1]	–0.6	—	0.6	V
V_{IDTH}	—	—	100	mV_{pp}
R_{IN}	80	—	120	Ω

1. SGMII DC specifications are not met. A coupling capacitor is required.

Cavium Confidential For Bowen Yang ALU 02/27/2014

38.3.5 RXAUI Interface Timing Specifications

38.3.5.1 RXAUI Clock Input Timing

Table 38–15 lists CN71XX's DLM reference-clock input (RXAUI) electrical characteristics.

Table 38–15 DLM Reference-Clock Input: RXAUI

Clock Name	Frequency	Frequency Tolerance	Duty Cycle	Phase Jitter (rms)[1]	Phase Jitter (p-to-p)	Edge Rate[2]	V_{HIGH} Levels			V_{LOW} Levels		
							Min	Nom	Max	Min	Nom	Max
DLMC_REF_CLK*n*_N/P	156.25 MHz	±100 ppm[3]	60/40	2 ps	44 ps	840ps	325mV	700mV	850mV	–150mV	0V	175mV

1. From 12 kHz to 20 MHz.
2. Edge rate is defined as 20%–80% value.
3. In links where a spread spectrum-modulated reference clock is used, both ends of the link must share the same master reference-clock source.

38.3.5.2 RXAUI AC Specifications

Table 38–16 shows the RXAUI TX specifications, and Table 38–17 shows the RXAUI RX specifications.

Table 38–16 RXAUI TX Specifications

Parameter	Min	Typical	Max	Units
Data rate	—	6.25	—	GBd
Differential p-p Amplitude	—	600	—	mV
Total jitter[1]	—	—	0.30	UIpp

1. Refer to the CEI6 spec (CEI-6G-SR standard of the Common Electrical I/O - Electrical and Jitter Interoperability agreement for 6G+ bps and 11G+ bps I/O, aka OIF-CEI-02.0).

Table 38–17 RXAUI RX Specifications

Parameter	Min	Typical	Max	Units
Data rate	6.25 – 100 ppm	6.25	6.25 + 100 ppm	GBd
Sensitivity (pp)	125	—	—	mV
Total jitter tolerance	0.65	—	—	UIpp
Differential resistance	80	—	120	Ω

Cavium Confidential For Bowen Yang ALU 02/27/2014

38.3.6 SATA Interface Timing Specifications

38.3.6.1 SATA Clock-Input Timing

Table 38–18 lists CN71XX's DLM reference-clock input (SATA) electrical characteristics.

Table 38–18 DLM Reference-Clock Input: SATA

Clock Name	Frequency	Frequency Tolerance[1]	Duty Cycle	Phase Jitter (rms)	Phase Jitter (p-to-p)	Edge Rate[2]	V_{HIGH} Levels			V_{LOW} Levels		
							Min	Nom	Max	Min	Nom	Max
DLMC_REF_CLK*n*_N/P	100 MHz	±350 ppm	60/40	2 ps	44 ps	840ps	325mV	700mV	850mV	–150mV	0V	175mV

1. In links where a spread-spectrum-modulated reference clock is used, you must ensure the internal SATA SSC generator is disabled, and the DLMC_REF_CLKn_N/P input that is used to drive the SATA PLL is not shared on a link that does not support SSC.
2. Edge rate is defined as 20%–80% value.

38.3.6.2 SATA AC Specifications

Table 38–19 shows the SATA TX specifications, and Table 38–20 shows the SATA RX specifications.

Table 38–19 SATA TX Specifications

Parameter [1]	Min	Typical	Max	Units
$V_{TX\text{-}DIFF\ pp}$	240	—	900	mV
T_j[2]	—	—	0.52	UI
$Z_{TX\text{-}DIFF\text{-}DC}$	80	—	120	Ω
f_{SSC}	30	—	33	kHz

1. For an explanation of the SATA parameters, refer to the Serial ATA Specification, Rev. 3.1.
2. Jitter is specified at 10^{-12} BER before the CIC.

Table 38–20 SATA RX Specifications

Parameter [1]	Min	Typical	Max	Units
$V_{RX\text{-}DIFF\ pp}$	240	—	1000	mV
$Z_{RX\text{-}DIFF\text{-}DC}$	80	—	120	Ω

1. For an explanation of the SATA parameters, refer to the Serial ATA Specification, Rev. 3.1.

Cavium Confidential For Bowen Yang ALU 02/27/2014

38.4 RGMII Interface

38.4.1 RGMII Clock Input Timing

Table 38–21 lists CN71XX's RGMII/MII reference-clock input electrical characteristics. Table 38–22 lists the clock periods for the specified clock signals.

Table 38–21 RGMII Reference-Clock Input

Clock Name	Frequency	Frequency Tolerance	Duty Cycle	Period Jitter (P–P)[1]	Edge Rate[2]	Levels
RGM_REF_CLK	125 MHz	±150 ppm	45/55	100 ps	2 ns	0.0–VDD_RGM

1. Cycle to cycle, peak to peak.
2. Edge rate is 10%–90% (rise) and 90%–10% (fall).

Table 38–22 RGMII Clock Periods

Parameter	Description	Min	Typical	Max	Units
T_{PERIOD}	RGM_REF_CLK period @ 10 Mb/s		400		ns
T_{PERIOD}	RGM0_RXC period @ 10 Mb/s		400		ns
T_{PERIOD}	RGM_REF_CLK period @ 100 Mb/s		40		ns
T_{PERIOD}	RGM0_RXC period @ 100 Mb/s		40		ns

38.4.2 RGMII Signal Timing

Timing for the RGMII interface is such that the clock and data are generated simultaneously by the source of the signals and therefore skew between the clock and data is critical to proper operation. CN71XX has a DLL circuit to automatically compute the quadrature settings. The computed value is available at AGL_PRT0/1_CTL[CLK_SET].

Table 38–23 lists CN71XX's timing parameters for RGMII when the DLL circuit is not engaged, while Table 38–24 lists the timing parameters when the DLL circuit is engaged. Figure 38–3 shows transmit timing and Figure 38–4 shows receive timing.

Table 38–23 is used when DLL circuit is not engaged. Table 38–24 is used when DLL is engaged. (For more info about engaging the DLL, refer to the note with step 3 in Section 25.7.1)

Table 38–23 RGMII Timing Parameters Without DLL Circuit Engaged

Parameter	Description	Min	Typical	Max	Units
$T_{SKEW}T$	Data to clock output skew (at transmitter)[1]	–0.5	0	0.5	ns
$T_{SETUP}R$	Data to clock input setup	1.0	—	—	ns
$T_{HOLD}R$	Data to clock input hold	1.0	—	—	ns
T_{CYC}	Clock cycle duration[2]	7.2	8	8.8	ns
Duty G	Duty cycle for Gigabit[3]	45	50	55	%
Duty T	Duty cycle for 10/100T[3]	40	50	60	%

1. This implies that the PC board design will require clocks to be routed such that an additional trace delay of greater than 1.5 ns and less than 2.0 ns will be added to the associated clock signal. For 10/100T, the max value is unspecified.
2. For 10 Mb/s and 100 Mb/s, T_{CYC} scales to 400 ns + (–40 ns) and 40 ns + (–4 ns) respectively.
3. Duty cycle may be stretched/shrunk during speed changes or while transmitting to a received packet's clock domain, as long as minimum duty cycle is not violated and stretching occurs for no more than three T_{CYC} of the lowest speed transitioned between.

Cavium Confidential For Bowen Yang 02/27/2014

Table 38–24 RGMII Timing Parameters With DLL Circuit Engaged

Parameter	Description	Min	Typical	Max	Units
$T_{SKEW}T$	Data to clock output skew (at transmitter)	1.5	2.0	2.5	ns
$T_{SETUP}R$	Data to clock input setup	–0.5	0	0.5	ns
$T_{HOLD}R$	Data to clock input hold	3.5	4.0	4.5	ns
T_{CYC}	Clock cycle duration[1]	7.2	8	8.8	ns
Duty G	Duty cycle for Gigabit[2]	45	50	55	%
Duty T	Duty cycle for 10/100T[3]	40	50	60	%

1. For 10 Mb/s and 100 Mb/s, T_{CYC} scales to 400 ns + (–40 ns) and 40 ns + (–4 ns) respectively.
2. Duty cycle may be stretched/shrunk during speed changes or while transmitting to a received packet's clock domain, as long as minimum duty cycle is not violated and stretching occurs for no more than three T_{CYC} of the lowest speed transitioned between.

Figure 38–3 RGMII Transmit Multiplexing and Timing Diagram

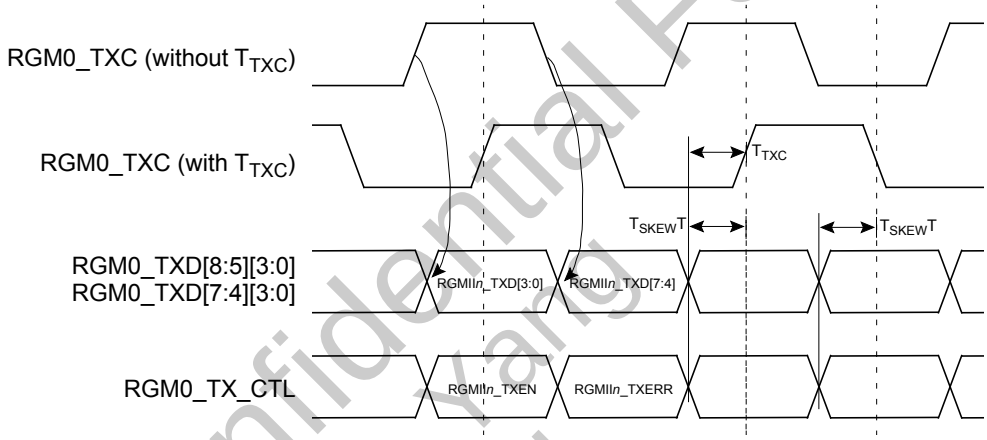
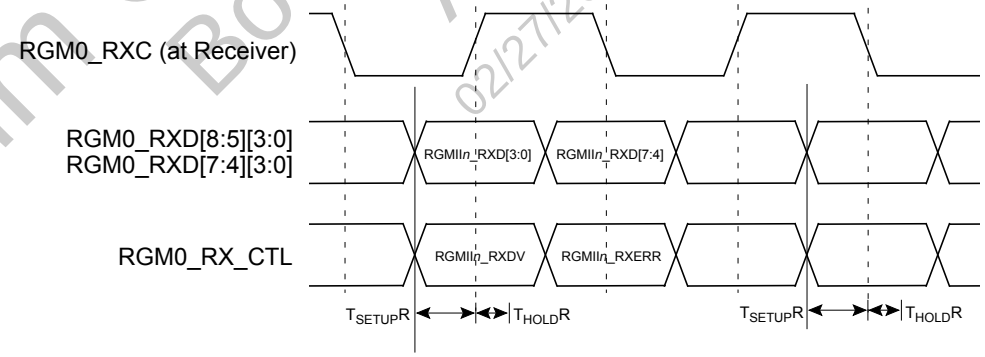


Figure 38–4 RGMII Receive Multiplexing and Timing Diagram



Cavium Confidential For Bowen Yang ALU 02/27/2014

38.5 EEPROM Interface

The CN71XX EEPROM interface is a serial interface to an external EEPROM, which can be used to initialize configuration data. See Section 22.4. At power-on, the PCIe controller can load its read-only configuration registers from a standard SPI serial EEPROM.

Following is a list of acceptable EEPROM part numbers[1] and a reference to the appropriate functional timing diagrams in the specs:

- Atmel AT25010B, AT25020B, AT25040B

  See Figures 4-2 through 4-7 of the Atmel SPI Serial EEPROM data-sheet. (8707D-SEEPROM-04/2013)

- ISSI IS25C02, IS25C04

  See Figures 2 through 7 of the ISSI SPI Serial Electrically Erasable PROM data-sheet. (8707C-SEEPR-6/11)

- ST M95010, M95020, M95040

  See Figures 7 through 13 of the ST M950x0 SPI bus EEPROM with high-speed clock data-sheet. (DocID6512 Rev 12, October 2013)

- Microchip 25xx010, 25xx020, 25xx040

  See Figures 2-1 through 2-7 of the Microchip SPI Bus Serial EEPROM data-sheet. (DS21832H, 11/27/12)

38.5.1 EEPROM Read Cycle

Table 38–25 lists CN71XX's EEPROM read-cycle timing and Figure 38–5 provides the timing diagram.

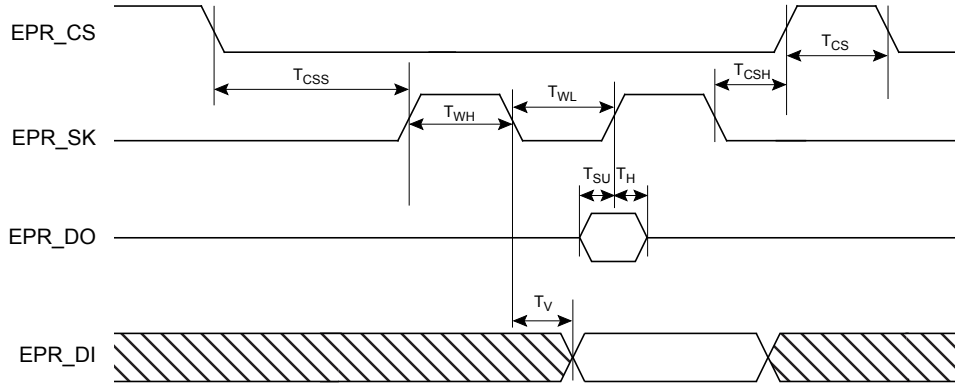
Table 38–25 EEPROM Cycle Timing

Parameter	Description	Min	Typical	Max	Units
T_{CSS}	EPR_CS setup time	200	—	—	ns
T_{WH}	EPR_SK high time	200	256	—	ns
T_{WL}	EPR_SK low time	200	256	—	ns
T_{SU}	EPR_DO setup time	80	—	—	ns
T_{H}	EPR_DO hold time	80	—	—	ns
T_{V}	EPR_DI valid	0	—	80	ns
T_{CS}	EPR_CS high time	200	—	—	ns
T_{CSH}	EPR_CS hold time	200	—	—	ns

Cavium Confidential For Bowen Yang ALU 02/27/2014

1. The SPI serial EEPROM must support 64-bit read operations and support a standard instruction set with standard encodings (i.e., set write enable latch, reset write enable latch, read/write status register, read/write data from/to memory array).

Figure 38–5 EEPROM AC Timing



In order to save power, EPR_SK only runs when the EEPROM is accessed.

Cavium Confidential For Bowen Yang ALU 02/27/2014

38.6 Boot Bus Interface

For timing diagrams, refer to Section 28.4 in the Boot Bus chapter.

38.7 EMMC Interface

Table 38–26 lists the eMMC command-timing parameters.

Table 38–26 EMMC Command-Timing Parameters

Parameter	Description	Min	Typ	Max	Units
T_{CH} [1]	eMMC clock high	2		65535	coprocessor-clock cycles
T_{CL} [1]	eMMC clock low	2		65535	coprocessor-clock cycles
T_{CHLD}	eMMC clock-command hold	1		$T_{CH}/4$	coprocessor-clock cycles
T_{CSET}	eMMC clock-command setup	1		$T_{CLK} - T_{CH}/2$	coprocessor-clock cycles

1. T_{CH} and T_{CL} are set by writing to MIO_EMM_SWITCH[CLK_HI, CLK_LO].

Figure 38–6 EMMC Command Timing Diagram

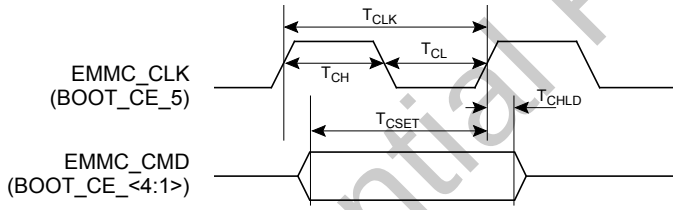
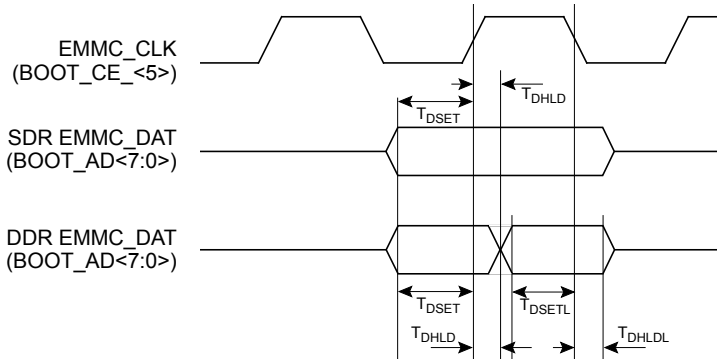


Table 38–27 lists the eMMC data-timing parameters.

Table 38–27 EMMC Data-Timing Parameters

Parameter	Description	Min	Typ	Max	Units
T_{DHLD}	eMMC data hold	1		$T_{CH}/4$	coprocessor-clock cycles
T_{DSET}	eMMC data set	1		$T_{CL} - T_{CL}/4$	coprocessor-clock cycles
T_{DHLDL}	eMMC data hold low	1		$T_{CL}/4$	coprocessor-clock cycles
T_{DSETL}	eMMC data set low	1		$T_{CH} - T_{CH}/4$	coprocessor-clock cycles

Figure 38–7 EMMC Data Timing Diagram



Cavium Confidential For Bowen Yang ALU 02/27/2014

38.8 JTAG Interface

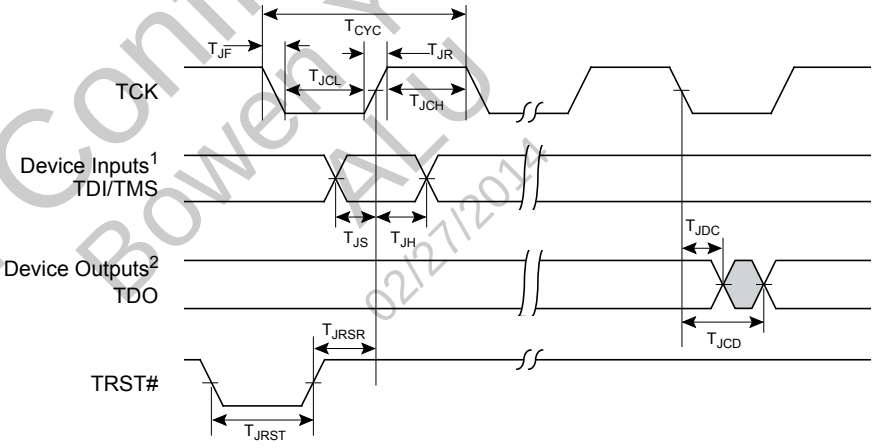
Table 38–28 lists CN71XX's JTAG signals I/O timing parameters.

Table 38–28 JTAG Signal I/O Timing Parameters

Parameter	Description	Min	Typ	Max	Units
T_{JCYC}	TCK cycle time	100		—	ns
T_{JCH}	TCK clock HIGH	40		—	ns
T_{JCL}	TCK clock LOW	40		—	ns
T_{JR} [1]	TCK rise time	—		5	ns
T_{JF} [1]	TCK fall time	—		5	ns
T_{JRST} [2]	TRST# assert time	50		—	ns
T_{JRSR}	TRST# recovery time	50			
T_{JCD} [3]	TCK to output data valid	5		50	ns
T_{JDC} [1, 3]	TCK to output data hold	5		—	ns
T_{JS} [4]	TCK to input setup time	50		—	ns
T_{JH} [5]	TCK to input hold time	50		—	ns

1. Guaranteed by design and characterization.
2. RST is an asynchronous level sensitive signal. Setup time for test purpose only.
3. Output = All device outputs including TDO.
4. Input = All device inputs include TDI and TMS.
5. Input = All device inputs include TDI and TMS.

Figure 38–8 JTAG Signal I/O Timing Diagram



38.9 TDM/PCM Interface

TBD

Cavium Confidential For Bowen Yang 02/27/2014

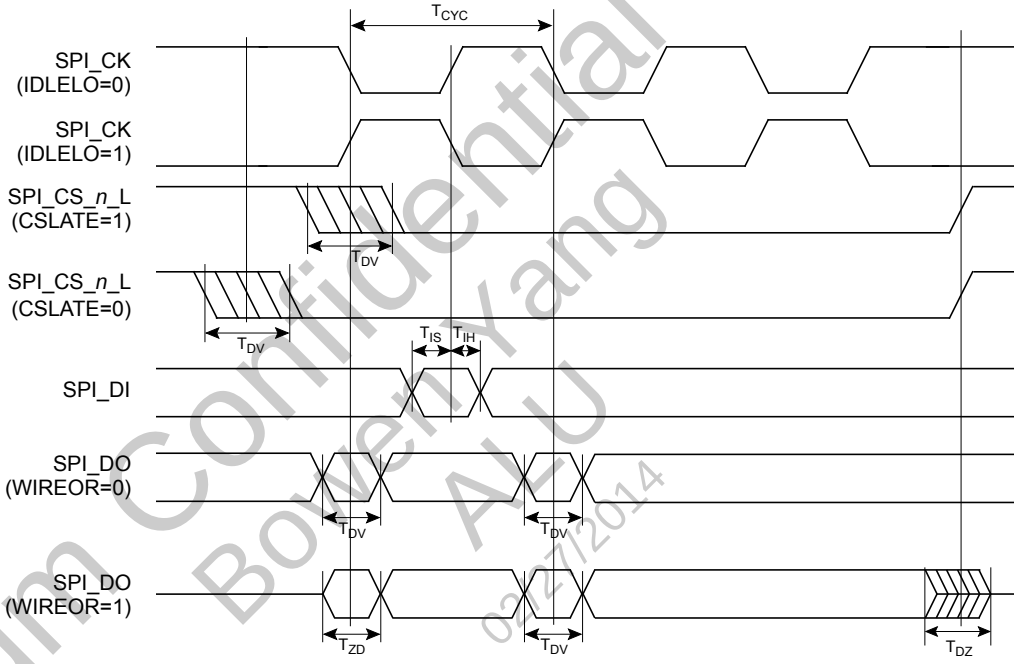
38.10 MPI/SPI Interface

Table 38–28 lists CN71XX's MPI/SPI signals I/O timing parameters and Figure 38–9 shows the signal timing.

Table 38–29 MPI/SPI Signal I/O Timing Parameters

Parameter	Description	Min	Max	Units
T_{CYC}	SPI_CK period	10	8191	coprocessor-clock cycles
T_{CYC}	SPI_CK period	20	—	ns
T_{DV}	SPI_DO data valid	–1	1	ns
T_{ZD}	SPI_DO turn-on delay	–1	1	ns
T_{DZ}	SPI_DO turn-off delay	–1	1	ns
T_{IS}	SPI_DI/SPI_DO input setup	7	—	ns
T_{IH}	SPI_DI/SPI_DO input hold	0	—	ns

Figure 38–9 MPI/SPI Signal I/O Timing Diagram



Cavium Confidential For Bowen Yang ALU 8/27/2014

38.11 TWSI Interface

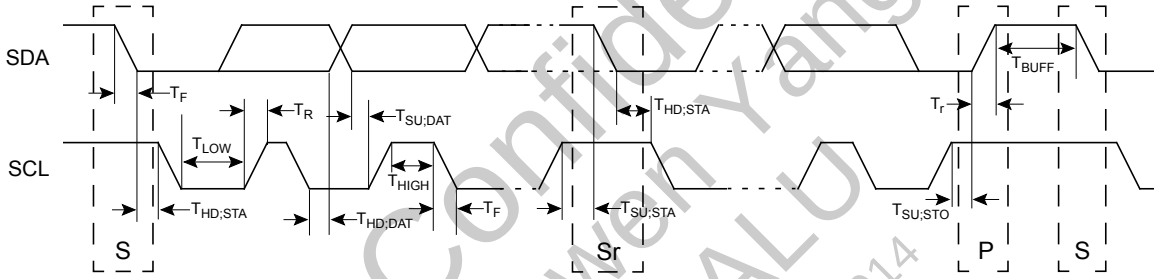
Table 38–28 lists CN71XX's TWSI signals I/O timing parameters.

Table 38–30 TWSI Signal Parameters

Parameter	Description	Standard Mode		Fast Mode		Units
		Min	Max	Min	Max	
F_{SCL}	SCL clock frequency	0	100	0	400	kHz
$T_{HD;STA}$[1]	Hold time (repeated) START condition	4.0	—	0.6	—	µs
T_{LOW}	Low period of the SCL clock	4.7	—	1.3	—	µs
T_{HIGN}	High period of the SCL clock	4.0	—	0.6	—	µs
$T_{SU;STA}$	Set-up time for a repeated START condition	4.7	—	0.6	—	µs
$T_{HD;DAT}$	Data hold time	0	—	0	—	µs
$T_{SU;DAT}$	Data set-up time	250	—	100	—	ns
T_R	Rise time of both SDA and SCL signals	—	1000	20 + 0.1Cb	300	ns
T_F	Fall time of both SDA and SCL signals	—	300	20 + 0.1Cb	300	ns
$T_{SU;STO}$	Set-up time for STOP condition	4.0	—	0.6	—	µs
T_{BUF}	Bus free time between a STOP and START condition	4.7	—	1.3	—	µs

1. After this period, the first clock pulse is generated.

Figure 38–10 TWSI Signal Parameters



38.12 SMI/MDIO Interface

For timing diagrams, refer to Figures 32–2 and 32–3 in the SMI chapter.

Cavium Confidential For Bowen Yang ALU 02/27/2014

38.13 USB Interface

38.13.1 USB Clock Input Timing

Depending on whether SS or HS operation is needed, CN71XX provides different options of reference-clock. Refer to Section 31.5.2 for more detail.

For the 50MHz reference-clock input, which uses clock source from PLL_REF_CLK, refer to Table 38–1 for reference-clock input electrical characteristics.

For the 100MHz reference-clock input, which uses clock source from DLMC_REF_CLK<1:0>_N/P, refer to Table 38–5 for reference-clock input electrical characteristics.

38.13.2 USB AC Specifications

Table 38–31 shows the USB 3.0 TX specifications.

Table 38–31 USB 3.0 TX Specifications

Parameter[1]	Min	Max	Units
$V_{TX\text{-}DIFF\ pp}$	0.8	1.2	V
$V_{TX\text{-}DE\text{-}RATIO}$	–3.0	–4.0	dB
$T_{TX\text{-}EYE}$	0.625	—	UI (200 ps)

1. For an explanation of USB parameters, refer to *Universal Serial Bus (USB) 3.0 Specification*, Revision 1.0, pp 6–20, 21.

USB 3.0 RX AC characteristics are validated in accordance with the *Universal Serial Bus (USB) 3.0 Specification*, Revision 1.0. For more information regarding TX and RX configuration, refer to Section 31.5.1 and Section 31.5.2.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 39

Mechanical Specifications

This chapter contains the following subjects:

- Overview
- Landed Ball-Grid-Array Package Diagram
- Package Thermal Specifications

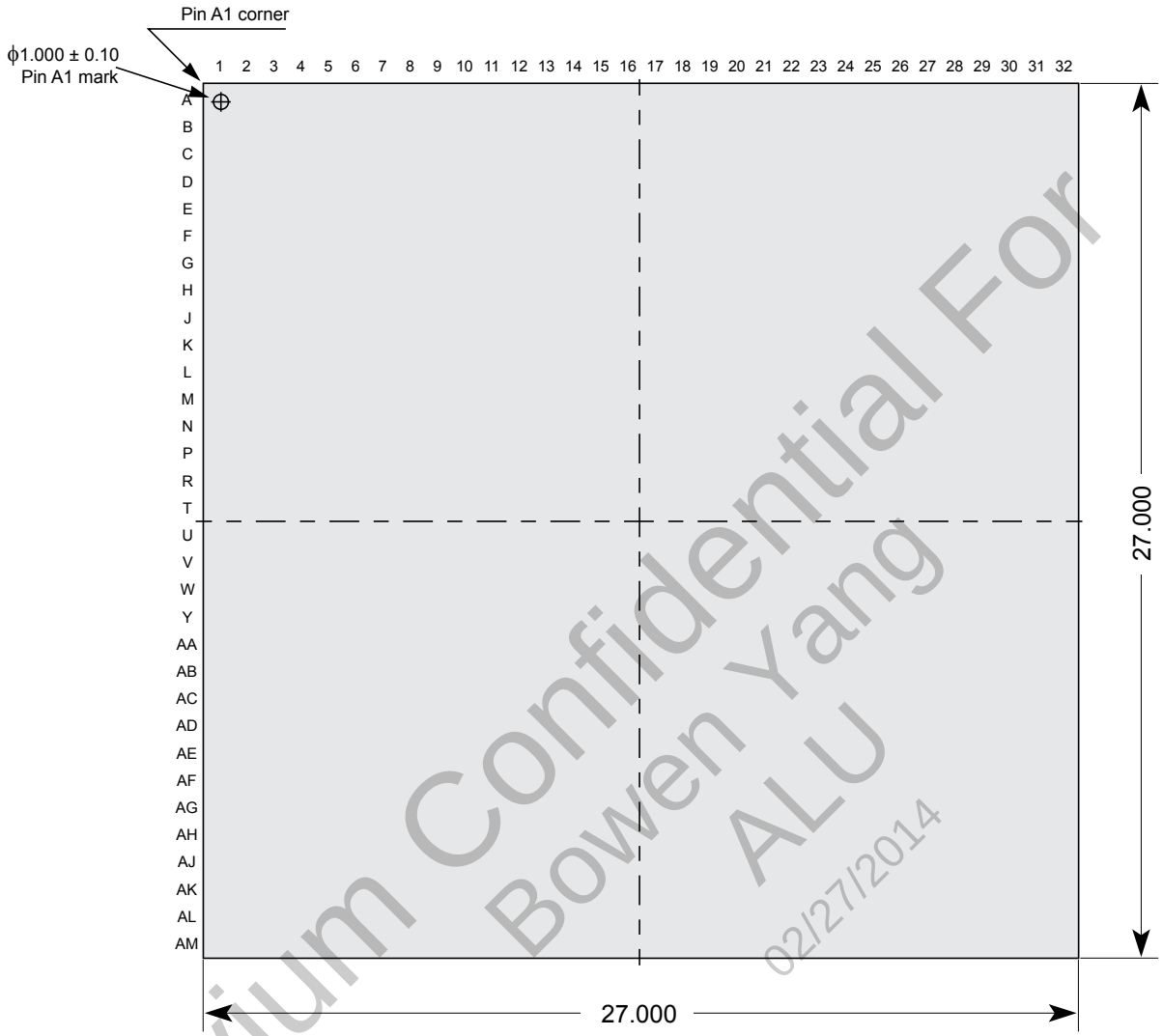
Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

This chapter contains the CN71XX package dimensions and ball grid array.

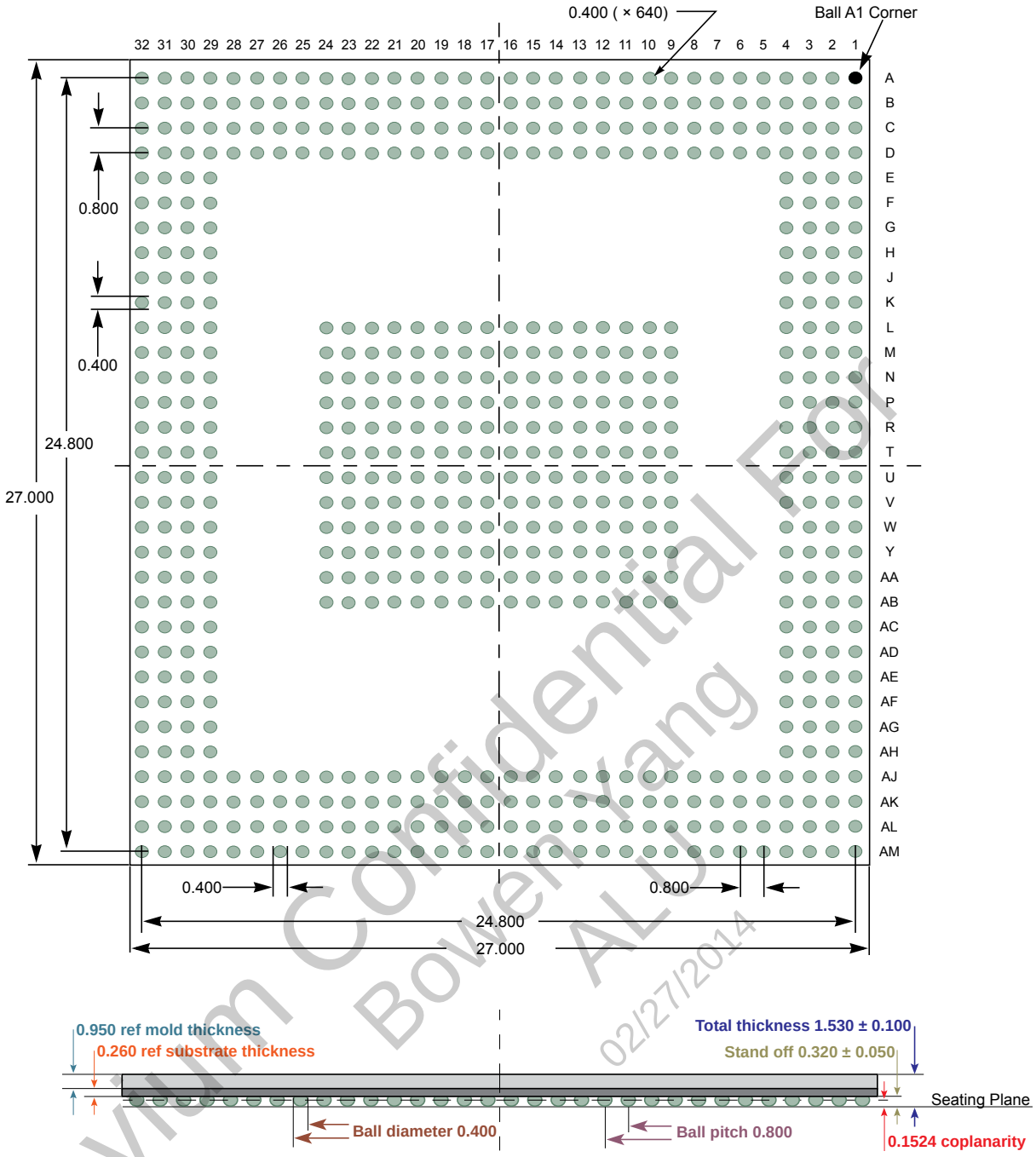
39.1 Landed Ball-Grid-Array Package Diagram

Figure 39–1 640-LBGA Package Diagram (Top View)



Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 39–2 640-LBGA Package Diagram (Bottom and Side Views)



Cavium Confidential For Bowen Yang ALU 02/27/2014

39.2 Package Thermal Specifications

Table 39–1 lists CN71XX's thermal packaging specifications.

Table 39–1 Thermal Package Specification for CN71XX

Parameter	Description	Max	Units
θ_{JA0}	Thermal resistance – junction to ambient at 0 m/s or 0 LFM	16.3	°C/W
θ_{JA1}	Thermal resistance – junction to ambient at 1 m/s or 200 LFM	14.9	°C/W
θ_{JA2}	Thermal resistance – junction to ambient at 2 m/s or 400 LFM	13.8	°C/W
θ_{JC}	Thermal resistance – junction to case	1.9	°C/W
θ_{JB}	Thermal resistance – junction to board	8.5	°C/W

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Chapter 40

CN71XX Signal Descriptions

This chapter provides descriptions of the CN71XX I/O signals. The signals are grouped according to the I/O interfaces, as shown in Table 40–1.

Table 40–1 CN71XX Interfaces

Interface	Number of Signals	Number of Balls	Table
DDR memory interface signals	81	81	Table 40–3 on page 1599
SerDes signals			
DLM common signals	8	8	Table 40–4 on page 1600,
PCIe-specific signals	4	4	Table 40–5 on page 1600
DLM signals	29	29	Table 40–6 on page 1601
RGMII interface signals	16	16	Table 40–7 on page 1601
USB interface signals	16	16	Table 40–8 on page 1602
SMI/MDIO interface signals	5	5	Table 40–9 on page 1602
GPIO interface signals	20	20[1]	Table 40–10 on page 1603
TDM/PCM interface signals	8	0[2]	Table 40–11 on page 1603
TWSI signals	4	4	Table 40–12 on page 1604
Clock signals	8	8	Table 40–13 on page 1604
Boot/eMMC signals	25	25	Table 40–14 on page 1605
SPI/MPI interface signals	7	1[3]	Table 40–15 on page 1605
UART interface signals	10	10	Table 40–16 on page 1606
eJTAG/JTAG interface signals	6	6	Table 40–17 on page 1606
Miscellaneous signals	14	14	Table 40–18 on page 1607
Power/Ground/Reserved/NoConnect	393	393	Table 40–19 on page 1607
Total number of signals	654	640	

1. Some balls support multiple signals, depending on the configuration.
2. The TDM/PCM signals share balls with GPIO_<19:12> and therefore are listed as zero here.
3. The SPI/MPI signals share balls with UART and boot signals, and therefore are listed as one here.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Overview

The CN71XX I/O signals are grouped by interface and described in tables with the following column headings:

Ball Name: Provides the name of the field or signal.

Reset: Indicates the value of the field or signal at reset.

If Not Used: Indicates what should be done externally with the balls if the field or signal is not used.

Dir: Indicates the direction of the field or signal: input, output, input/output or analog input.

Type: Indicates what type of signal. Refer to Table 40–2.

Up/Down: Indicates the direction of CN71XX's weak internal pull-up/pull-down resistor for the signal, as well as resistor values. If blank, CN71XX does not have a weak internal pull-up/pull-down resistor on the signal.

Description: Describes the function of the signal or field.

Signal Ball Types

Table 40–2 describes the connections of the I/O types used in the Type column of the signals list:

Table 40–2 CN71XX Signal Types

I/O Type	Description
USB_ANALOG_IO	3.3V tolerant USB analog input
CMOS25	2.5V CMOS I/O
CMOS25_50	2.5V RGMII I/O
CMOS33	3.3V CMOS I/O
SERDES_I	High-speed SerDes differential input
SERDES_O	High-speed SerDes differential output
SERDES_CLK	High-speed SerDes reference input clock
USB_SERDES_I	High-speed USB differential input
USB_SERDES_O	High-speed USB differential output
USB_SERDES_CLK	High-speed USB reference input clock
SSTL15/135	1.5V/1.35V SSTL I/O for DDR3/DDR3L interfaces
SUPPLY	Power supply input

NOTE: When using a passive pull-up resistor on the board on CMOS25- and CMOS33-type pins, please use a value of 2.2kΩ

When using a passive pull-up resistor on the board on CMOS25_50-type pins, please use a value of 1.5k Ω

Cavium Confidential For Bowen Yang 02/12/2014

40.1 DRAM Interface Signals

The CN71XX DRAM interface communicates across a 36-bit data bus (32 bits for data and four bits for ECC).

Table 40–3 describes the signals for each DRAM interface.

Table 40–3 DRAM Interface Signals

Ball Name	Reset	If Not Used	Dir	Type	Up/ Down	Description
DDR_CS<1:0>_L	1	NC	O	SSTL15/135		DDR3 DIMM chip select. CS0 indicates rank 0, CS1 indicates rank 1.
DDR_CAS_L	1	NC	O	SSTL15/135		DDR3 column address select.
DDR_RAS_L	1	NC	O	SSTL15/135		DDR3 row address select.
DDR_A<15:0>	0	NC	O	SSTL15/135		DDR3 address lines.
DDR_BA<2:0>	0	NC	O	SSTL15/135		DDR3 bank address lines.
DDR_DQ<31:0>	Z	NC	I/O	SSTL15/135		DDR3 data (bytes <3:0>).
DDR_CB<3:0>	Z	NC	I/O	SSTL15/135		DDR3 check bits.
DDR_DQS_<3:0>_P DDR_DQS_<3:0>_N	Z	NC	I/O	SSTL15/135		Positive and negative differential data strobes for DDR_DQ<31:0>. DDR_DQS_0 for byte 0,..., DDR_DQS_3 for byte 3.
DDR_CBS_0_P DDR_CBS_0_N	Z Z	NC	I/O	SSTL15/135		Positive and negative differential data strobes for DDR_CB<3:0>
DDR_CK_<1:0>_P DDR_CK_<1:0>_N	1 0	NC	O	SSTL15/135		DDR3 differential address clocks (positive and negative).
DDR_CKE<1:0>	0	NC	O	SSTL15/135		DDR3 clock-enable signals.
DDR_WE_L	1	NC	O	SSTL15/135		DDR3 write enable.
DDR_ODT_<1:0>	0	NC	O	SSTL15/135		DDR3 DIMM on-die termination enable.
DDR_RESET_L	1	NC	O	SSTL15/135		DDR3 reset signal.
DDR_COMP_UP		NC	O	SSTL15/135		DDR output-impedance compensation. Connect to 240Ω ±1% resistor to ground.
DDR_COMP_DN		NC	O	SSTL15/135		DDR output-impedance compensation. Connect to 60.4Ω ±1% resistor to VDD_DDR.
Total = 81						

Cavium Confidential For Bowen Yang ALU 02/27/2014

40.2 SerDes Signals

The CN71XX contains three SerDes dual-lane modules (DLMs) that are shared between PCIe, SGMII, QSGMII, RXAUI, and SATA interfaces. For configuration options, refer to Figure 1–1 or .

Table 40–4 describes the DLM common signals that are used to reset the SerDes links. Table 40–5 describes the signals that are PCIe-only signals. Table 40–6 describes the signals associated with the appropriate DLMs.

Note that the DLM0_REF_CLK_P/N and DLMC_REF_CLK*n*_P/N frequencies should be:

- 100 MHz for PCIe, SATA, or USB, and possibly SGMII or QSGMII.
- 125 MHz for SGMII, QSGMII, or RXAUI.
- 156.25 MHz for RXAUI.

For users who may not need to use the full capacity of the DLMs, refer to the following guidelines:

- **Powered DLMs Connected to Unused Slots**

  The DLM is connected as if the slot were used. Nothing special needs to be done.

- **Unused (Powered-Down) DLMs**

You may choose to not use a DLM and wish to power it down. In this case, the recommended signal connections are described in Table 40–6 in the If Not Used column.

Table 40–4 DLM Common Signals

Ball Name	Reset	If Not Used	Dir	Type	Up/ Down	Description
PERST0_L PERST1_L PERST2_L	N/A	NC	I/O	CMOS25/CMOS33		TBD
DLMC_REF_CLK0_P DLMC_REF_CLK0_N DLMC_REF_CLK1_P DLMC_REF_CLK1_N		GND	I	SERDES_CLK		SerDes common reference clock signals (differential inputs). Either reference clock can be selected through CSRs as the reference clock for each of the USB and DLM blocks. In PCIe EP mode, DLMC_REF_CLK0 must be selected for PEM0.
DLMC_VPH_SELECT_18			I	CMOS25/CMOS33		Selects VPH level for DLMs.
Total = 8						

Table 40–5 PCIe-Specific Signals

Ball Name	Reset	If Not Used	Dir	Type	Up/ Down	Description
PCIE0_REV_LANES	N/A	NC	I	CMOS25/ CMOS33	DN	Pull up to reverse the order of the lanes for PCIe0 (DLM2–DLM5). This signal is pulled down internally.
PCIE0_MODE<2:0>			I	CMOS25/ CMOS33		Specifies the PEM0 configuration. 0x0 = EP mode, ×2, Gen2 0x4 = EP mode, ×2, Gen1 0x1 = EP mode, ×1, Gen2 0x5 = EP mode, ×1, Gen1 0x2 = EP mode, ×4, Gen2 0x6 = EP mode, ×4, Gen1 0x3 = RC mode, Gen2 0x7 = RC mode, Gen1
Total = 4						

Cavium Confidential For Bowen Yang 02/27/2014

Table 40–6 DLM Signals

Signal Name	Reset	If Not Used	Dir	Type	Up/ Down	Description
DLM 0						
DLM0_TXP<1:0>	N/A	NC	O	SERDES_O		Output data (positive).
DLM0_TXN<1:0>	N/A	NC	O	SERDES_O		Output data (negative).
DLM0_RXP<1:0>	N/A	GND	I	SERDES_I		Input data (positive).
DLM0_RXN<1:0>	N/A	GND	I	SERDES_I		Input data (negative).
DLM0_RESREF	N/A	NC	I/O	ANALOG I/O		DLM0 external resistor connect.
DLM0_REF_CLK_P DLM0_REF_CLK_N	N/A	GND	I	SERDES_CLK		SerDes reference-clock signals (differential inputs) for DLM0 only; cannot be shared with other blocks. Note that DLM0 can use one of the common reference clocks DLMC_REF_CLK<1:0>_P/N instead of these signals. The clock is selected through CSRs.
DLM 1						
DLM1_TXP<1:0>	N/A	NC	O	SERDES_O		Output data (positive).
DLM1_TXN<1:0>	N/A	NC	O	SERDES_O		Output data (negative).
DLM1_RXP<1:0>	N/A	GND	I	SERDES_I		Input data (positive).
DLM1_RXN<1:0>	N/A	GND	I	SERDES_I		Input data (negative).
DLM1_RESREF	N/A	NC	I/O	ANALOG I/O		DLM1 external resistor connect.
DLM 2						
DLM2_TXP<1:0>	N/A	NC	O	SERDES_O		Output data (positive).
DLM2_TXN<1:0>	N/A	NC	O	SERDES_O		Output data (negative).
DLM2_RXP<1:0>	N/A	GND	I	SERDES_I		Input data (positive).
DLM2_RXN<1:0>	N/A	GND	I	SERDES_I		Input data (negative).
DLM2_RESREF	N/A	NC	I/O	ANALOG I/O		DLM2 external resistor connect.
Total = 29						

40.3 RGMII Interface Signals

The CN71XX has one packet interface for RGMII. The AGL interface has compensation pins that are used by the attached interface, shown in Table 40–7. The signals are described in Table 40–7.

Table 40–7 RGMII Signals

Signal Name	Reset	If Not Used	Dir	Type	Up/ Down	Description
RGM0_RXD_<3:0>		NC	I	CMOS25_50		RGM RX data.
RGM0_RXCTL		NC	I	CMOS25_50		RGM RX control.
RGM0_RXC		NC	I	CMOS25_50		RGM RX clock.
RGM0_TXD_<3:0>	0	NC	O	CMOS25_50		RGM TX data.
RGM0_TXCTL	0	NC	O	CMOS25_50		RGM TX control.
RGM0_TXC	0	NC	O	CMOS25_50		RGM TX clock.
RGM_VDD_SELECT		PU[1]	I	Analog		Selects VDD level for RGMII: enable 2.5V operation: 1kΩ pull-up resistor to VDD (nom = 0.85V)
RGM_REF_CLK				CMOS25_50		RGM reference clock.
RGM_COMP_DN[2]	1	NC	I/O	CMOS25_50		Down compensation resistor to VDD_RGM, with a value of 50 Ω ± 5%.
RGM_COMP_UP[2]	0	NC	I/O	CMOS25_50		Up compensation resistor to GND, with a value of 50 Ω ± 5%.
Total = 16						

1. Recommended external pull-up resistor value is 1kΩ pull-up to VDD.
2. RGM_COMP_UP and RGM_COMP_DN are used to specify the driving strength of the RGMII signals.

Cavium Confidential For Bowen Yang ALU 02/27/2014

40.4 USB Interface Signals

Table 40–8 describes the signals associated with the USB interfaces.

Table 40–8 USB Interface Signals

Ball Name	Reset	If Not Used	Dir	Type	Up/ Down	Description
USB0_SSRX_P		NC	I	USB SERDES_ I		USB 3.0 input data (plus)
USB0_SSRX_M		NC	I	USB SERDES_ I		USB 3.0 input data (minus)
USB0_SSTX_P		NC	O	USB SERDES_ O		USB 3.0 output data (plus)
USB0_SSTX_M		NC	O	USB SERDES_ O		USB 3.0 output data (minus)
USB0_VBUS			I	USB SERDES_ I		
USB0_RESREF		NC	I/O	USB Analog I/O		External resistor connect (200 Ω ±1%). If the USB interface is needed, connect a pull-down 200 Ω resistor (nominal value) to ground.
USB0_DP		NC	I/O	USB Analog I/O		D+ analog signal from the USB cable (USB 2.0 and below)
USB0_DM		NC	I/O	USB Analog I/O		D– analog signal from the USB cable (USB 2.0 and below)
USB1_SSRX_P		NC	I	USB SERDES_ I		USB 3.0 input data (plus)
USB1_SSRX_M		NC	I	USB SERDES_ I		USB 3.0 input data (minus)
USB1_SSTX_P		NC	O	USB SERDES_ O		USB 3.0 output data (plus)
USB1_SSTX_M		NC	O	USB SERDES_ O		USB 3.0 output data (minus)
USB1_VBUS			I	USB SERDES_ I		
USB1_RESREF		NC	I/O	USB Analog I/O		External resistor connect (200 Ω ±1%). If the USB interface is needed, connect a pull-down 200 Ω resistor (nominal value) to ground.
USB1_DP		NC	I/O	USB Analog I/O		D+ analog signal from the USB cable (USB 2.0 and below)
USB1_DM		NC	I/O	USB Analog I/O		D– analog signal from the USB cable (USB 2.0 and below)
Total = 16						

40.5 SMI/MDIO Interface Signals

Table 40–9 describes the signals associated with the SMI/MDIO interface.

Table 40–9 SMI/MDIO Interface Signals

Ball Name	Reset	If Not Used	Dir	Type[1]	Up/ Down	Description
SMI0_MDIO	0	NC	I/O	CMOS25/CMOS33		SMI0/MDIO serial data
SMI0_MDC	0	NC	O	CMOS25/CMOS33		SMI0/MDIO clock
SMI1_MDIO	0	NC	I/O	CMOS25/CMOS33		SMI1/MDIO serial data
SMI1_MDC	0	NC	O	CMOS25/CMOS33		SMI1/MDIO clock
SMI_VDD_SELECT			I	Analog		Selects VDD level for SMI/MDIO: • enable 3.3V operation: NC • enable 2.5V operation: 1kΩ pull-up resistor to VDD (nom = 0.85V)
Total = 5						

1. The SMI/MDIO interface is powered by the VDD_SMI power supply.

Cavium Confidential For Bowen Yang ALU 02/27/2014

40.6 General Purpose I/O (GPIO) Interface Signals

The GPIO interface has 20 signals available for I/O. Some of these signals are shared with the boot bus, and the PCM/TDM interface:

- bits<11:0> are strictly GPIO bits
- bits <19:12> can be used with the PCM/TDM interface, or as GPIO bits

Table 40–10 describes the GPIO interface signals.

Table 40–10 GPIO Interface Signals

Pin Name	Reset	If Not Used	Dir	Type	Up/ Down	Description	Notes
GPIO_<19:12>		NC	I/O	CMOS25/ CMOS33	DN[1]	General purpose I/O.	These signals share balls on the chip with TDM/ PCM interface signals. Refer to Table 40–11.
GPIO_<11:0>		NC	I/O	CMOS25/ CMOS33	DN[1]	General purpose I/O.	
Total = 20							

1. Value is 50kΩ ± 25%.

40.7 TDM/PCM Interface Signals

Table 40–11 describes the signals available to the TDM/PCM interface.

Table 40–11 PCM/TDM Interface Signals

Pin Name	Reset	If Not Used	Dir	Type	Up/ Down	Description	Notes
PCM_BCLK0		NC	I/O	CMOS25/ CMOS33	DN[1]	TDM/PCM bit clock for clock pair 0	This signal and GPIO_19 share a ball on the chip. Refer to Table 40–10. This signal is controlled by PCM_CLK0_GEN[N].
PCM_FSYNC0		NC	I/O	CMOS25/ CMOS33	DN[1]	TDM/PCM sync clock for clock pair 0	This signal and GPIO_18 share a ball on the chip. Refer to Table 40–10. This signal is controlled by PCM_CLK0_GEN[N].
PCM_DATA<3:0>		NC	I/O	CMOS25/ CMOS33	DN[1]	TDM/PCM engine data bits	These signals and GPIO_<12,13,16,17> share balls on the chip. Refer to Table 40–10. These signals are controlled by PCM*n*_TDM_CFG[ENABLE].
PCM_BCLK1		NC	I/O	CMOS25/ CMOS33	DN[1]	TDM/PCM bit clock for clock pair 1	This signal and GPIO_15 share a ball on the chip. Refer to Table 40–10. This signal is controlled by PCM_CLK1_GEN[N].
PCM_FSYNC1		NC	I/O	CMOS25/ CMOS33	DN[1]	TDM/PCM sync clock for clock pair 1	This signal and GPIO_14 share a ball on the chip. Refer to Table 40–10. This signal is controlled by PCM_CLK1_GEN[N].
Total = 8							

1. Value is 50kΩ ± 25%.

Cavium Confidential For Bowen Yang 02/11/14

40.8 Two-Wire Serial Interface (TWSI) Signals

Table 40–12 describes the signals associated with the CN71XX TWSI interfaces.

Table 40–12 TWSI Signals

Ball Name	Reset	If Not Used	Dir	Type	Up/ Down	Description
TWSI Interface 0						
TWS0_SCL	Z	PU[1]	I/O (OD)	CMOS25/CMOS33		TWSI serial clock
TWS0_SDA	Z	PU[1]	I/O (OD)	CMOS25/CMOS33		TWSI serial data
TSWI Interface 1						
TWS1_SCL	Z	PU[1]	I/O (OD)	CMOS25/CMOS33		TWSI serial clock
TWS1_SDA	Z	PU[1]	I/O (OD)	CMOS25/CMOS33		TWSI serial data
Total = 4						

1. Recommended external pull-up resistor value is 2kΩ ± 10%.

40.9 Clock Signals

Table 40–13 describes the CN71XX clock signals.

Table 40–13 Clock Signals

Ball Name	Reset	If Not Used	Dir	Type	Up/ Down	Description
PLL_REF_CLK		N/A	I	CMOS25/ CMOS33		Reference clock input pin for on-chip core, coprocessor, and memory controller PLLs (50 MHz).
PLL_DC_OK		NC	I	CMOS25/ CMOS33		Power OK. Assert this signal after all the power supplies and clocks are stable (refer to Section 37.3.1).
PLL_MUL_<3:0>		NC	I	CMOS25/ CMOS33	DN[1]	Core-clock PLL frequency selectors. 0x4 = 800 MHz 0x5 = 1.0 GHz 0x7 = 1.2 GHz 0xA = 1.5 GHz 0xB = 1.6 GHz All other values are reserved. For more information on the CN71XX core clock, see Section 1.1.
SYS_PLL_MUL_<1:0>		NC	I	CMOS25/ CMOS33	DN[1]	Coprocessor-clock PLL frequency selectors 0x0 = 350 MHz 0x1 = 400 MHz 0x2 = 500 MHz 0x3 = 600 MHz For more information on the CN71XX coprocessor clock, see Section 1.1.
Total = 8						

1. Value is 50kΩ ± 25%.

Cavium Confidential For Bowen Yang ALU 07/7/2014

40.10 Boot-Bus/eMMC Signals

Table 40–14 describes the signals associated with the boot bus/eMMC.

Table 40–14 Boot Bus Signals

Ball Name	Reset	If Not Used	Dir	Type	Up/ Down	Description
BOOT_WAIT_L		NC	I	CMOS25/CMOS33		Boot-bus wait/busy signal
BOOT_AD<7:0> (EMMC_DAT<7:0>)	Z	NC	I/O	CMOS25/CMOS33		Boot-bus address. When using eMMC, BOOT_AD<7:0> is EMMC_DAT<7:0>.
BOOT_ALE<3:0>	0	NC	O	CMOS25/CMOS33		Address latch enable
BOOT_CE_<0>_L	1	NC	O	CMOS25/CMOS33		Boot-bus chip enable for boot device
BOOT_CE_<5:1>_L (EMMC_CLK, EMMC_CMD<3:0>)	1	NC	O	CMOS25/CMOS33		Boot-bus chip enable for nonboot devices. When using eMMC: BOOT_CE_<5>_L is EMMC_CLK BOOT_CE_<4:1>_L is EMMC_CMD<3:0>
BOOT_CE_<7:6>_L	1	NC	O	CMOS25/CMOS33		Boot-bus chip enable for nonboot devices. Note that BOOT_CE_L<7:6> and SPI_CS<1:0>_L share balls on the chip. Refer to Table 40–15.
BOOT_OE_L	1	NC	O	CMOS25/CMOS33		Boot-bus output enable
BOOT_CLE	1	NC	O	CMOS25/CMOS33		NAND-flash command latch enable
BOOT_WE_L	1	NC	O	CMOS25/CMOS33		Boot-bus write enable
BOOT_DIR	1	NC	O	CMOS25/CMOS33		Transfer direction
Total = 25						

40.11 SPI Interface Signals

Table 40–15 describes the CN71XX SPI/MPI signals.

Table 40–15 SPI Signals

Ball Name	Reset	If Not Used	Dir	Type	Up/ Down	Description	Notes
SPI_CS<1:0>_L		NC	O	CMOS25/ CMOS33		SPI/MPI chip select	These signals and BOOT_CE_<7:6>_L share balls on the chip. Refer to Table 40–14.
SPI_CS<2>_L		NC	O	CMOS25/ CMOS33		SPI/MPI chip select	These signals and UART1_CTS_L share balls on the chip. Refer to Table 40–16.
SPI_CS<3>_L		NC	O	CMOS25/ CMOS33		SPI/MPI chip select	These signals and UART1_RTS_L share balls on the chip. Refer to Table 40–16.
SPI_CK		NC	O	CMOS25/ CMOS33		SPI/MPI clock	
SPI_DI		NC	I	CMOS25/ CMOS33		SPI/MPI receive data	This signal and UART0_DCD_L share a ball on the chip. Refer to Table 40–16. Controlled by MPI_CFG[ENABLE].
SPI_DO		NC	O	CMOS25/ CMOS33		SPI/MPI transmit data	This signal and UART0_DTR_L share a ball on the chip. Refer to Table 40–16. Controlled by MPI_CFG[ENABLE].
Total = 7							

Cavium Confidential For Bowen Yang ALD 02/27/2018

40.12 UART Interface Signals

Table 40–16 describes the signals associated CN71XX UART interfaces.

Table 40–16 UART Interface Signals

Ball Name	Reset	If Not Used	Dir	Type	Up/ Down	Description	Notes
Universal Asynchronous Receive/Transmit Interface (UART) 0							
UART0_CTS_L		NC	I	CMOS25/CMOS33		UART0 clear to send	
UART0_RTS_L	1	NC	O	CMOS25/CMOS33		UART0 request to send	This signal is used as a bootstrap-select signal that selects between PLL_REF_CLK and DLMC_REF_CLK1 as the input for the core, coprocessor, and DDR PLLs. An external pull-up/pull-down resistor must be added to make this signal either high or low: ● A 2.2kΩ pull-up resistor selects PLL_REF_CLK as the input. ● A 10kΩ pull-down resistor selects DLMC_REF_CLK1 as the input.
UART0_SIN		NC	I	CMOS25/CMOS33		UART0 serial data in	
UART0_SOUT	1	NC	O	CMOS25/CMOS33		UART0 serial data out	
UART0_DCD_L		NC	I	CMOS25/CMOS33		UART0 data carrier detect	This signal and SPI_DI share a ball on the chip. Refer to Table 40–15. Controlled by MPI_CFG[ENABLE].
UART0_DTR_L	1	NC	O	CMOS25/CMOS33		UART0 data terminal ready	This signal and SPI_DO share a ball on the chip. Refer to Table 40–15. Controlled by MPI_CFG[ENABLE].
Universal Asynchronous Receive/Transmit Interface (UART) 1							
UART1_CTS_L		NC	I	CMOS25/CMOS33		UART1 clear to send	This signal and SPI_CS<2>_L share a ball on the chip. Refer to Table 40–15. Controlled by MPI_CFG.
UART1_RTS_L	1	NC	O	CMOS25/CMOS33		UART1 request to send	This signal and SPI_CS<3>_L share a ball on the chip. Refer to Table 40–15. Controlled by MPI_CFG.
UART1_SIN		NC	I	CMOS25/CMOS33		UART1 serial data in	
UART1_SOUT	1	NC	O	CMOS25/CMOS33		UART1 serial data out	
Total = 10							

40.13 EJTAG/JTAG Signals

Table 40–17 describes the CN71XX EJTAG/JTAG signals.

Table 40–17 eJTAG/JTAG Signals

Ball Name	Reset	If Not Used	Dir	Type	Up/ Down	Description
EJTG_TRST_L		NC	I	CMOS25/CMOS33	DN[1]	EJTAG reset
JTG_TCK		NC	I	CMOS25/CMOS33		JTAG clock
JTG_TDI		NC	I	CMOS25/CMOS33		JTAG test data in
JTG_TDO	Z	NC	O	CMOS25/CMOS33		JTAG test data out
JTG_TMS		NC	I	CMOS25/CMOS33		JTAG TMS
JTG_TRST_L		NC	I	CMOS25/CMOS33	DN[1]	JTAG reset
Total = 6						

1. Value is 50kΩ ± 25%.

Cavium Confidential For Bowen Yang ALU 02/27/2014

40.14 Miscellaneous Signals

Table 40–18 describes the CN71XX miscellaneous signals.

Table 40–18 Miscellaneous Signals

Ball Name	Reset	If Not Used	Dir	Type	Up/ Down	Description
MSC_CLKOUT		NC	O	CMOS25/CMOS33		Diagnostic clock out, running at 1/12 core-clock speed
MSC_CLKOUT _ENABLE		NC	I	CMOS25/CMOS33	DN[1]	Enable MSC_CLKOUT
MSC_SYS_CLKOUT		NC	O	CMOS25/CMOS33		Diagnostic clock out, running at 1/12 coprocessor-clock speed
VRM_I2_CLK		NC	I/O	CMOS25/CMOS33		PMBus-compatible clock to/from VRM.
VRM_I2_DATA		NC	I/O	CMOS25/CMOS33		PMBus-compatible data to/from VRM.
EPR_CS		NC	O	CMOS25/CMOS33		PCIe configuration EEPROM chip select
EPR_DI		NC	O	CMOS25/CMOS33		PCIe configuration data to EEPROM
EPR_DO		NC	I	CMOS25/CMOS33		PCIe configuration data from EEPROM
EPR_SK		NC	O	CMOS25/CMOS33		PCIe configuration EEPROM chip clock
THERMAL_D_N		GND	I	Analog		Thermal diode N terminal
THERMAL_D_P		GND	I	Analog		Thermal diode P terminal
CHIP_RESET_L			I	CMOS25/CMOS33		Chip reset input
REMOTE_BOOT		NC	I	CMOS25/CMOS33	DN[1]	PCIe boot select. 1 = boot from PCIe bus, 0 = boot from boot bus.
IO_VDD_SELECT			I	Analog		Selects VDD level for VDD_IO: • enable 3.3V operation: NC • enable 2.5V operation: 1kΩ pull-up resistor to VDD (nom = 0.85V)
Total = 14						

1. Value is 50kΩ ± 25%.

40.15 Power/Ground/Reserved/No Connect Signals

Table 40–19 lists the power/ground/reserved/no connect signals.

Table 40–19 Power/Ground/Reserved/No Connect Signals

Ball Name	Number	If Not Used	Dir	Type	Up/ Down	Description
GND	257	GND		Supply		Ground supply.
CORE_PLL_VDD	1			Supply		Core PLL low-noise supply (0.9V).
DDR_PLL_VDD	1			Supply		DDR PLL low-noise supply (0.9V).
SYS_PLL_VDD	1			Supply		Coprocessor PLL low-noise supply (0.9V).
VDD	39			Supply		Core supply voltage.
VDD_DDR	27			Supply		DDR supply voltage (DDR3 = 1.5V, DDR3L = 1.35V).
VDD_IO	19			Supply		Miscellaneous supply voltage (3.3/2.5V). If VDD_IO is 3.3V, this supply can come from the same off-chip source as the USB*n*_VDD33 supply, but it should be isolated on the board through a ferrite bead.
VDD_15	1			Supply		Miscellaneous 1.5V analog supply voltage. This supply can come from the same off-chip source as the VDD_DDR supply (if and only if it is 1.5V), but should be isolated on the board through a ferrite bead.
USB0_VDD33[1]	2	GND		Supply		USB supply voltage (3.3V).
USB1_VDD33[1]	2	GND		Supply		USB supply voltage (3.3V).
USB0_DVDD[1]	1	GND		Supply		USB 2.0 low-voltage supply. This supply can come from the same off-chip source as the VDD supply, but should be isolated on the board through a ferrite bead.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 40–19 Power/Ground/Reserved/No Connect Signals (Continued)

Ball Name	Number	If Not Used	Dir	Type	Up/ Down	Description
USB1_DVDD[1]	1	GND		Supply		USB 2.0 low-voltage supply. This supply can come from the same off-chip source as the VDD supply, but should be isolated on the board through a ferrite bead.
USB0_VP[1]	1	GND		Supply		USB 3.0 low-voltage supply. This supply can come from the same off-chip source as the VDD supply, but should be isolated on the board through a ferrite bead.
USB1_VP[1]	1	GND		Supply		USB 3.0 low-voltage supply. This supply can come from the same off-chip source as the VDD supply, but should be isolated on the board through a ferrite bead.
USB0_VPTX[1]	1	GND		Supply		USB transmitter supply voltage. This supply can come from the same off-chip source as the VDD supply, but should be isolated on the board through a ferrite bead.
USB1_VPTX[1]	1	GND		Supply		USB transmitter supply voltage. This supply can come from the same off-chip source as the VDD supply, but should be isolated on the board through a ferrite bead.
VDD_RGM	3	GND		Supply		RGMII interface 1.8/2.5 V supply voltage
VDD_SMI	1	GND		Supply		SMI/MDIO interface 2.5/3.3V supply voltage
VDD_DLM0	1	GND		Supply		DLM0 analog supply voltage (0.85V)
VDDH_DLM0	1	GND		Supply		DLM0 high-voltage analog supply voltage (1.8V)
VDDTX_DLM0	1	GND		Supply		DLM0 transmit supply voltage
VDD_DLM1	1	GND		Supply		DLM1 analog supply voltage (0.85V)
VDDH_DLM1	1	GND		Supply		DLM1 high-voltage analog supply voltage (1.8V)
VDDTX_DLM1	1	GND		Supply		DLM1 transmit supply voltage
VDD_DLM2	1	GND		Supply		DLM2 analog supply voltage (0.85V)
VDDH_DLM2	1	GND		Supply		DLM2 high-voltage analog supply voltage (1.8V)
VDDTX_DLM2	1	GND		Supply		DLM2 transmit supply voltage
VDD_DLMC	1	GND		Supply		DLM common reference-clock supply voltage
NC	10			—		Not connected
RSVD	13			—		Reserved; for internal use only. Implementation recommendation: NC with via.
Total = 393						

1. **All** of the USB 2.0 and USB 3.0 power supplies must be powered if the USB interface is being used, regardless of the configuration. If only one USB interface is being used, the other interface's supplies can be grounded.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Chapter 41

Ball Assignments

This chapter contains the following subjects:

- Overview
- CN71XX Landed Ball-Grid Array
- CN71XX Signal Mapping
- Balls Sorted in Alphabetical Order by Signal Name
- Balls Sorted in Numerical Order and Associated Signal Names

Cavium Confidential For Bowen Yang ALU 02/27/2014

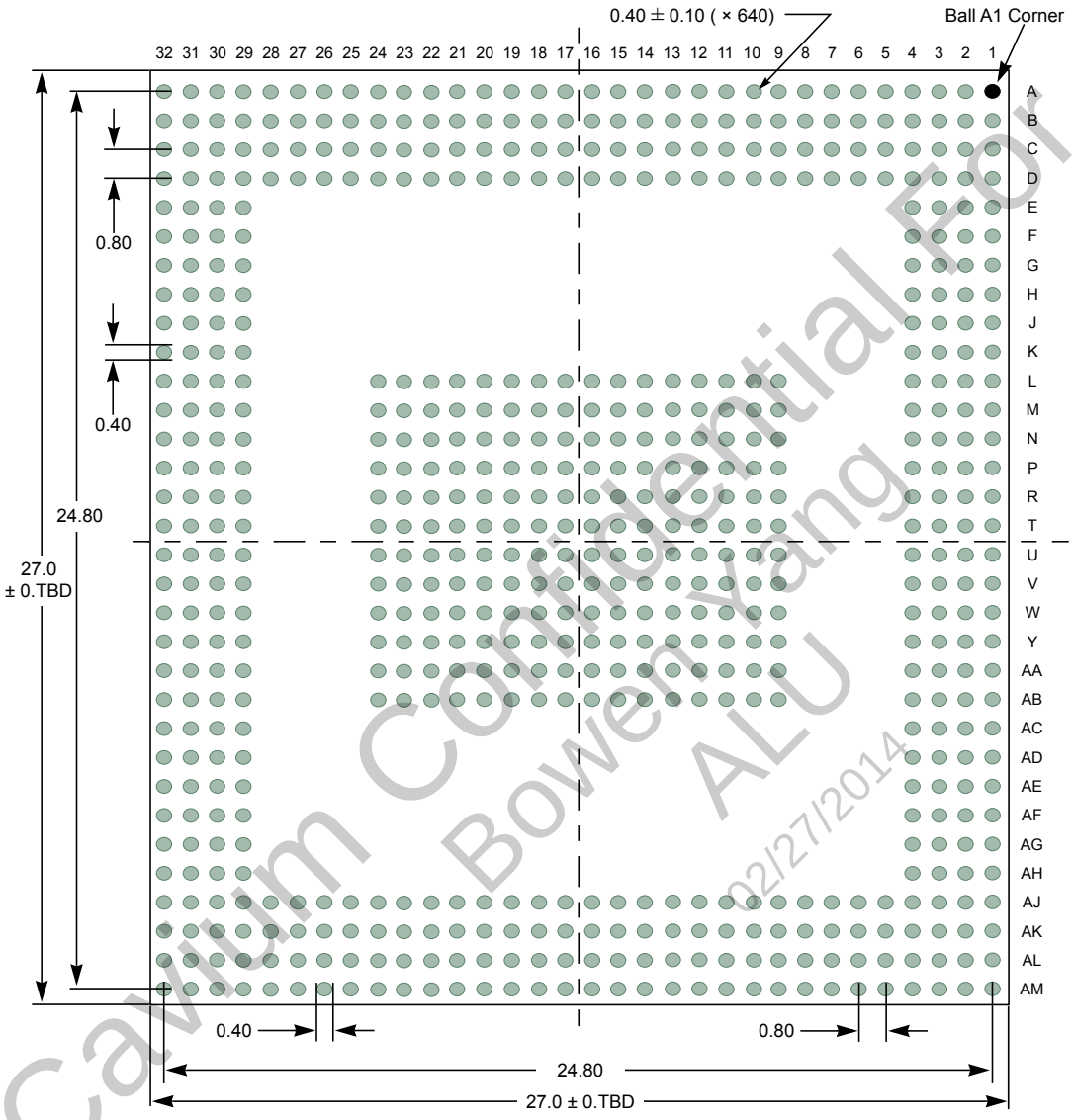
Overview

This chapter provides the signal/ball assignments (pinouts) for the CN71XX family of network processors.

41.1 CN71XX Landed Ball-Grid Array

Figure 41–1 shows the CN71XX landed ball-grid array. The black pin is the A1 pin.

Figure 41–1 640 BGA Ball Assignment Diagram (Bottom View)

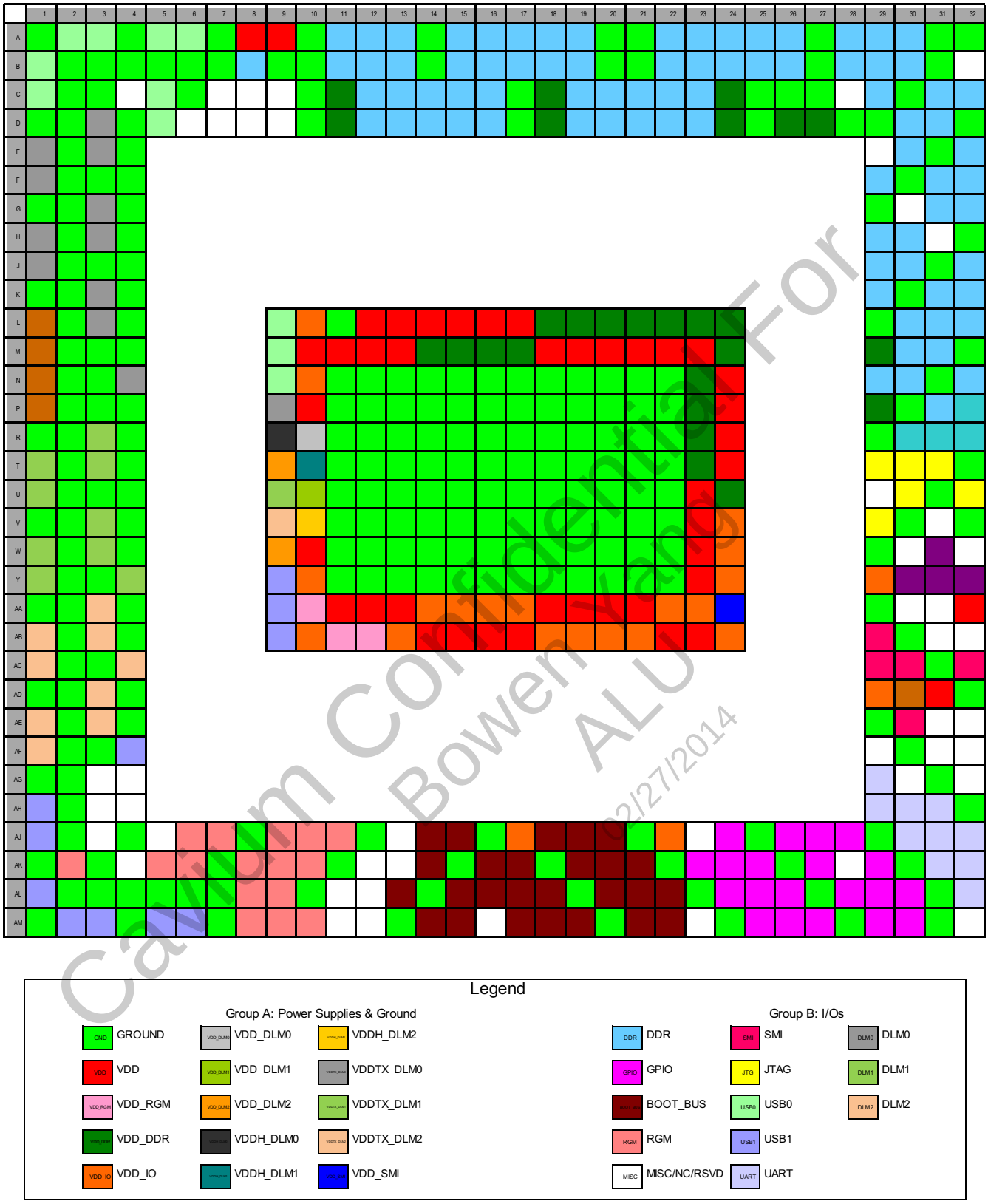


Cavium Confidential For Bowen Yang ALU 02/27/2014

41.2 CN71XX Signal Mapping

Figure 41–2 shows the CN71XX signal map.

Figure 41–2 640 LBGA Preliminary Signal Map (Top View)



Cavium Confidential For Bowen Yang ALU 02/27/2014

The CN71XX signal map with the associated signal names is shown in four quadrants:

- Figure 41–3 shows the upper left quadrant.
- Figure 41–4 shows the upper right quadrant.
- Figure 41–5 shows the lower left quadrant.
- Figure 41–6 shows the lower right quadrant.

Figure 41–3 CN71XX Signal Map, Upper Left Quadrant (Top View)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	GND	USB0_SSTX_M	USB0_SSTX_P	GND	USB0_SSRX_M	USB0_SSRX_P	GND	SYS_PLL_VDD	CORE_PLL_VDD	GND	DDR_CBS_0_P	DDR_CB<2>	DDR_CB<1>	GND	DDR_DQ<22>	DDR_DQ<20>
B	USB0_DP	GND	GND	GND	GND	GND	GND	DDR_PLL_VDD	GND	GND	DDR_CBS_0_N	DDR_CB<3>	DDR_CB<0>	GND	DDR_DQ<23>	DDR_DQ<21>
C	USB0_DM	GND	GND	NC	USB0_VBUS	GND	RSVD	RSVD	RSVD	GND	VDD_DDR	DDR_DQ<30>	DDR_DQ<28>	DDR_DQS_3_N	DDR_DQ<26>	DDR_DQ<25>
D	GND	GND	DLM0_REF_CLK_P	GND	USB0_RESREF	RSVD	RSVD	RSVD	RSVD	GND	VDD_DDR	DDR_DQ<31>	DDR_DQ<29>	DDR_DQS_3_P	DDR_DQ<27>	DDR_DQ<24>
E	DLM0_RXN<0>	GND	DLM0_REF_CLK_N	GND												
F	DLM0_RXP<0>	GND	GND	GND												
G	GND	GND	DLM0_TXN<0>	GND												
H	DLM0_RXP<1>	GND	DLM0_TXP<0>	GND												
J	DLM0_RXN<1>	GND	GND	GND												
K	GND	GND	DLM0_TXP<1>	GND												
L		GND	DLM0_TXN<1>	GND												
M	DLMC_REF_CLK_N	GND	GND	GND												
N		GND	GND	DLM0_RESREF												
P	DLMC_REF_CLK_N	GND	GND	GND												
R	GND	GND	DLM1_TXN<0>	GND												
T	DLM1_RXN<0>	GND	DLM1_TXP<0>	GND												

USB0_VP	USB0_VDD33	GND	VDD	VDD	VDD	VDD	VDD
USB0_VPTX	VDD	VDD	VDD	VDD	VDD_DDR	VDD_DDR	VDD_DDR
USB0_DVDD	USB0_VDD33	GND	GND	GND	GND	GND	GND
VDDTX_DLM0	VDD	GND	GND	GND	GND	GND	GND
VDDH_DLM0	VDD_DLMC	GND	GND	GND	GND	GND	GND
VDD_DLMC	VDDH_DLM1	GND	GND	GND	GND	GND	GND

Cavium Confidential For Bowen Yang ALU 9/27/2014

Figure 41–4 CN71XX Signal Map, Upper Right Quadrant (Top View)

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
DDR_DQS_2_P	DDR_DQ<18>	DDR_DQ<17>	GND	GND	DDR_DQ<6>	DDR_DQ<4>	DDR_DQS_0_P	DDR_DQ<2>	DDR_DQ<1>	GND	DDR_COMP_UP	DDR_CK_1_P	DDR_CK_0_P	GND	GND	A
DDR_DQS_2_N	DDR_DQ<19>	DDR_DQ<16>	GND	GND	DDR_DQ<7>	DDR_DQ<5>	DDR_DQS_0_N	DDR_DQ<3>	DDR_DQ<0>	GND	DDR_COMP_DN	DDR_CK_1_N	DDR_CK_0_N	GND	NC	B
GND	VDD_DDR	DDR_DQ<14>	DDR_DQ<13>	DDR_DQS_1_P	DDR_DQ<10>	DDR_DQ<9>	VDD_DDR	GND	GND	GND	RSVD	DDR_CKE1	GND	DDR_ODT_0	DDR_RAS_L	C
GND	VDD_DDR	DDR_DQ<15>	DDR_DQ<12>	DDR_DQS_1_N	DDR_DQ<11>	DDR_DQ<8>	VDD_DDR	GND	VDD_DDR	VDD_DDR	GND	GND	DDR_ODT_1	DDR_CS1_L	GND	D
												NC	DDR_CKE0	GND	DDR_CAS_L	E
												DDR_CS0_L	GND	DDR_WE_L	DDR_BA<2>	F
												GND	NC	DDR_A<15>	DDR_A<0>	G
												DDR_A<10>	DDR_A<12>	RSVD	GND	H
												DDR_A<5>	DDR_BA<0>	GND	DDR_A<3>	J
												DDR_A<2>	GND	DDR_A<1>	DDR_A<9>	K
VDD	VDD_DDR	VDD_DDR	VDD_DDR	VDD_DDR	VDD_DDR	VDD_DDR	VDD_DDR					GND	DDR_A<4>	DDR_BA<1>	DDR_A<11>	L
VDD_DDR	VDD	VDD	VDD	VDD	VDD	VDD	VDD_DDR					VDD_DDR	DDR_A<6>	DDR_A<7>	GND	M
GND	GND	GND	GND	GND	GND	VDD_DDR	VDD					DDR_A<8>	DDR_A<14>	GND	DDR_A<13>	N
GND	GND	GND	GND	GND	GND	VDD_DDR	VDD					VDD_DDR	GND	DDR_RESET_L	TWS0_SCL	P
GND	GND	GND	GND	GND	GND	VDD_DDR	VDD					GND	TWS0_SDA	TWS1_SDA	TWS1_SCL	R
GND	GND	GND	GND	GND	GND	VDD_DDR	VDD					JTG_TRST_L	JTG_TDO	JTG_TCK	GND	T

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 41–5 CN71XX Signal Map, Lower Left Quadrant (Top View)

U	DLM1_RXP<0>	GND	GND	GND												
V	GND	GND	DLM1_TXP<1>	GND												
W	DLM1_RXP<1>	GND	DLM1_TXN<1>	GND												
Y	DLM1_RXN<1>	GND	GND	DLM1_RESREF												
AA	GND	GND	DLM2_TXN<0>	GND												
AB	DLM2_RXN<0>	GND	DLM2_TXP<0>	GND												
AC	DLM2_RXP<0>	GND	GND	DLM2_RESREF												
AD	GND	GND	DLM2_TXP<1>	GND												
AE	DLM2_RXP<1>	GND	DLM2_TXN<1>	GND												
AF	DLM2_RXN<1>	GND	GND	USB1_VBUS												
AG	GND	GND	RSVD	PLL_MUL_0												
AH	USB1_DM	GND	PLL_MUL_1	PLL_MUL_2												
AJ	USB1_DP	GND	PLL_MUL_3	GND	SYS_PLL_MUL_1	RGM0_TXD_2	RGM0_TXD_1	GND	RGM_REF_CLK	RGM0_RXD_1	RGM0_RXCTL	GND	PERST0_L	BOOT_AD<6>	BOOT_AD<2>	GND
AK	GND	RGM_VDD_SELECT	GND	SYS_PLL_MUL_0	RGM0_TXD_3	RGM0_TXC	GND	RGM0_TXCTL	RGM0_RXD_3	RGM0_RXC	GND	PCIE_MODE0	PERST2_L	BOOT_AD<7>	GND	BOOT_ALE_2
AL	USB1_RESREF	GND	GND	GND	GND	GND	GND	RGM0_TXD_0	RGM0_RXD_2	GND	PCIE_MODE2	PCIE_MODE1	REMOTE_BOOT	GND	BOOT_AD<3>	BOOT_AD<0>
AM	GND	USB1_SSTX_M	USB1_SSTX_P	GND	USB1_SSRX_M	USB1_SSRX_P	GND	RGM_COMP_UP	RGM_COMP_DN	RGM0_RXD_0	PCIE0_REV_LANES	PERST1_L	GND	BOOT_AD<5>	BOOT_AD<4>	NC
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

VDDTX_DLM1	VDD_DLM1	GND	GND	GND	GND	GND	GND
VDDTX_DLM2	VDDH_DLM2	GND	GND	GND	GND	GND	GND
VDD_DLM2	VDD	GND	GND	GND	GND	GND	GND
USB1_DVDD	USB1_VDD33	GND	GND	GND	GND	GND	GND
USB1_VPTX	VDD_RGM	VDD	VDD	VDD	VDD_IO	VDD_IO	VDD_IO
USB1_VP	USB1_VDD33	VDD_RGM	VDD_RGM	VDD_IO	VDD	VDD	VDD

Cavium Confidential For Bowen Yang ALU 02/27/2014

Figure 41–6 CN71XX Signal Map, Lower Right Quadrant (Top View)

GND	GND	GND	GND	GND	GND	VDD	VDD_DDR					NC	JTG_TDI	GND	EJTG_TRST_L	U
GND	GND	GND	GND	GND	GND	VDD	VDD_IO					JTG_TMS	GND	NC	GND	V
GND	GND	GND	GND	GND	GND	VDD	VDD_IO					GND	NC	EPR_CS	SPI_CK	W
GND	GND	GND	GND	GND	GND	VDD	VDD_IO					VDD_IO	EPR_SK	EPR_DO	EPR_DI	Y
VDD_IO	VDD	VDD	VDD	VDD	VDD_IO	VDD_IO	VDD_SMI					GND	MSC_CLKOUT	MSC_SYS_CLKOUT	IO_VDD_SELECT	AA
VDD	VDD_IO	VDD_IO	VDD_IO	VDD_IO	VDD	VDD	VDD_IO					SMI0_MDIO	GND	RSVD	MSC_CHIP_RMII	AB
												SMI1_MDIO	SMI1_MDC	GND	SMI0_MDC	AC
												VDD_IO	MSC_PH_SELECT_N	VDD_15	GND	AD
												GND	SMI_VDD_SELECT	THERMAL_D_P	THERMAL_D_N	AE
												VRM_I2_DATA	GND	RSVD	PLL_DC_OK	AF
												UART1_SIN	VRM_I2_CLK	GND	RSVD	AG
												UART0_SIN	UART1_CTS_L	UART1_SOUT	GND	AH
VDD_IO	BOOT_CE_7_L	BOOT_CE_4_L	BOOT_CE_0_L	GND	VDD_IO	PLL_REF_CLK	GPIO_9	GND	GPIO_14	GPIO_16	GPIO_19	GND	UART0_CTS_L	UART0_SOUT	UART1_RTS_L	AJ
BOOT_ALE_0	GND	BOOT_CE_6_L	BOOT_CE_3_L	BOOT_CLE	GND	GPIO_4	GPIO_3	GPIO_8	GND	GPIO_13	NC	GPIO_18	GND	UART0_DCD_L	UART0_RTS_L	AK
BOOT_ALE_3	BOOT_WAIT_L	GND	BOOT_WE_L	BOOT_CE_2_L	BOOT_DIR	GND	GPIO_0	GPIO_2	GPIO_6	GND	GPIO_11	GPIO_12	GPIO_17	GND	UART0_DTR_L	AL
BOOT_AD<1>	BOOT_ALE_1	BOOT_OE_L	GND	BOOT_CE_5_L	BOOT_CE_1_L	CHIP_RESET_L	GND	GPIO_1	GPIO_5	GPIO_10	GND	GPIO_7	GPIO_15	GND	NC	AM
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	

Cavium Confidential For Bowen Yang ATO 02/27/2014

41.3 Balls Sorted in Alphabetical Order by Signal Name

Signal Name	Ball
BOOT_AD<0>	AL16
BOOT_AD<1>	AM17
BOOT_AD<2>	AJ15
BOOT_AD<3>	AL15
BOOT_AD<4>	AM15
BOOT_AD<5>	AM14
BOOT_AD<6>	AJ14
BOOT_AD<7>	AK14
BOOT_ALE_0	AK17
BOOT_ALE_1	AM18
BOOT_ALE_2	AK16
BOOT_ALE_3	AL17
BOOT_CE_0_L	AJ20
BOOT_CE_1_L	AM22
BOOT_CE_2_L	AL21
BOOT_CE_3_L	AK20
BOOT_CE_4_L	AJ19
BOOT_CE_5_L	AM21
BOOT_CE_6_L	AK19
BOOT_CE_7_L	AJ18
BOOT_CLE	AK21
BOOT_DIR	AL22
BOOT_OE_L	AM19
BOOT_WAIT_L	AL18
BOOT_WE_L	AL20
CHIP_RESET_L	AM23
CORE_PLL_VDD	A9
DDR_A<0>	G32
DDR_A<1>	K31
DDR_A<2>	K29
DDR_A<3>	J32
DDR_A<4>	L30
DDR_A<5>	J29
DDR_A<6>	M30
DDR_A<7>	M31
DDR_A<8>	N29
DDR_A<9>	K32
DDR_A<10>	H29
DDR_A<11>	L32
DDR_A<12>	H30
DDR_A<13>	N32

Signal Name	Ball
DDR_A<14>	N30
DDR_A<15>	G31
DDR_BA<0>	J30
DDR_BA<1>	L31
DDR_BA<2>	F32
DDR_CAS_L	E32
DDR_CB<0>	B13
DDR_CB<1>	A13
DDR_CB<2>	A12
DDR_CB<3>	B12
DDR_CBS_0_N	B11
DDR_CBS_0_P	A11
DDR_CK_0_N	B30
DDR_CK_0_P	A30
DDR_CK_1_N	B29
DDR_CK_1_P	A29
DDR_CKE0	E30
DDR_CKE1	C29
DDR_COMP_DN	B28
DDR_COMP_UP	A28
DDR_CS0_L	F29
DDR_CS1_L	D31
DDR_DQ<0>	B26
DDR_DQ<1>	A26
DDR_DQ<2>	A25
DDR_DQ<3>	B25
DDR_DQ<4>	A23
DDR_DQ<5>	B23
DDR_DQ<6>	A22
DDR_DQ<7>	B22
DDR_DQ<8>	D23
DDR_DQ<9>	C23
DDR_DQ<10>	C22
DDR_DQ<11>	D22
DDR_DQ<12>	D20
DDR_DQ<13>	C20
DDR_DQ<14>	C19
DDR_DQ<15>	D19
DDR_DQ<16>	B19
DDR_DQ<17>	A19
DDR_DQ<18>	A18

Signal Name	Ball
DDR_DQ<19>	B18
DDR_DQ<20>	A16
DDR_DQ<21>	B16
DDR_DQ<22>	A15
DDR_DQ<23>	B15
DDR_DQ<24>	D16
DDR_DQ<25>	C16
DDR_DQ<26>	C15
DDR_DQ<27>	D15
DDR_DQ<28>	C13
DDR_DQ<29>	D13
DDR_DQ<30>	C12
DDR_DQ<31>	D12
DDR_DQS_0_N	B24
DDR_DQS_0_P	A24
DDR_DQS_1_N	D21
DDR_DQS_1_P	C21
DDR_DQS_2_N	B17
DDR_DQS_2_P	A17
DDR_DQS_3_N	C14
DDR_DQS_3_P	D14
DDR_ODT_0	C31
DDR_ODT_1	D30
DDR_PLL_VDD	B8
DDR_RAS_L	C32
DDR_RESET_L	P31
DDR_WE_L	F31
DLM0_REF_CLK_N	E3
DLM0_REF_CLK_P	D3
DLM0_RESREF	N4
DLM0_RXN<0>	E1
DLM0_RXN<1>	J1
DLM0_RXP<0>	F1
DLM0_RXP<1>	H1
DLM0_TXN<0>	G3
DLM0_TXN<1>	L3
DLM0_TXP<0>	H3
DLM0_TXP<1>	K3
DLM1_RESREF	Y4
DLM1_RXN<0>	T1
DLM1_RXN<1>	Y1

Cavium Confidential For Bowen Yang ALU 02/27/2014

Signal Name	Ball
DLM1_RXP<0>	U1
DLM1_RXP<1>	W1
DLM1_TXN<0>	R3
DLM1_TXN<1>	W3
DLM1_TXP<0>	T3
DLM1_TXP<1>	V3
DLM2_RESREF	AC4
DLM2_RXN<0>	AB1
DLM2_RXN<1>	AF1
DLM2_RXP<0>	AC1
DLM2_RXP<1>	AE1
DLM2_TXN<0>	AA3
DLM2_TXN<1>	AE3
DLM2_TXP<0>	AB3
DLM2_TXP<1>	AD3
DLMC_REF_CLK0_N	M1
DLMC_REF_CLK0_P	L1
DLMC_REF_CLK1_N	P1
DLMC_REF_CLK1_P	N1
DLMC_VPH_SELECT_18	AD30
EJTG_TRST_L	U32
EPR_CS	W31
EPR_DI	Y32
EPR_DO	Y31
EPR_SK	Y30
GND	L4
GND	M4
GND	J3
GND	G1
GND	K1
GND	F3
GND	N3
GND	M3
GND	C26
GND	D28
GND	V32
GND	A1
GND	A4
GND	A10
GND	A14
GND	A20
GND	A21
GND	A27
GND	A31

Signal Name	Ball
GND	A32
GND	B2
GND	B3
GND	B4
GND	B6
GND	B7
GND	B9
GND	B10
GND	B14
GND	B20
GND	B21
GND	B27
GND	B31
GND	C2
GND	C3
GND	C6
GND	C10
GND	C17
GND	C25
GND	C30
GND	D2
GND	D4
GND	D10
GND	D17
GND	D25
GND	D29
GND	D32
GND	E2
GND	E4
GND	E31
GND	F2
GND	F4
GND	F30
GND	G2
GND	G4
GND	G29
GND	H2
GND	H4
GND	H32
GND	J2
GND	J31
GND	K2
GND	K4
GND	K30

Signal Name	Ball
GND	L2
GND	L11
GND	L29
GND	M2
GND	M32
GND	N2
GND	N11
GND	N12
GND	N13
GND	N14
GND	N15
GND	N16
GND	N17
GND	N18
GND	N19
GND	N20
GND	N21
GND	N22
GND	N31
GND	P2
GND	P3
GND	P4
GND	P11
GND	P12
GND	P13
GND	P14
GND	P15
GND	P16
GND	P17
GND	P18
GND	P19
GND	P20
GND	P21
GND	P22
GND	P30
GND	R1
GND	R2
GND	R4
GND	R11
GND	R12
GND	R13
GND	R14
GND	R15
GND	R16

Cavium Confidential For Bowen Yang ALU 03/27/2014

Signal Name	Ball
GND	R17
GND	R18
GND	R19
GND	R20
GND	R21
GND	R22
GND	R29
GND	T2
GND	T4
GND	T11
GND	T12
GND	T13
GND	T14
GND	T15
GND	T16
GND	T17
GND	T18
GND	T19
GND	T20
GND	T21
GND	T22
GND	T32
GND	U2
GND	U3
GND	U4
GND	U11
GND	U12
GND	U13
GND	U14
GND	U15
GND	U16
GND	U17
GND	U18
GND	U19
GND	U20
GND	U21
GND	U22
GND	U31
GND	V1
GND	V2
GND	V4
GND	V11
GND	V12
GND	V13

Signal Name	Ball
GND	V14
GND	V15
GND	V16
GND	V17
GND	V18
GND	V19
GND	V20
GND	V21
GND	V22
GND	V30
GND	W2
GND	W4
GND	W11
GND	W12
GND	W13
GND	W14
GND	W15
GND	W16
GND	W17
GND	W18
GND	W19
GND	W20
GND	W21
GND	W22
GND	W29
GND	Y2
GND	Y3
GND	Y11
GND	Y12
GND	Y13
GND	Y14
GND	Y15
GND	Y16
GND	Y17
GND	Y18
GND	Y19
GND	Y20
GND	Y21
GND	Y22
GND	AA1
GND	AA2
GND	AA4
GND	AA29
GND	AB2

Signal Name	Ball
GND	AB4
GND	AB30
GND	AC2
GND	AC3
GND	AC31
GND	AD1
GND	AD2
GND	AD4
GND	AD32
GND	AE2
GND	AE4
GND	AE29
GND	AF2
GND	AF3
GND	AF30
GND	AG1
GND	AG2
GND	AG31
GND	AH2
GND	AH32
GND	AJ2
GND	AJ4
GND	AJ8
GND	AJ12
GND	AJ16
GND	AJ21
GND	AJ25
GND	AJ29
GND	AK1
GND	AK3
GND	AK7
GND	AK11
GND	AK15
GND	AK18
GND	AK22
GND	AK26
GND	AK30
GND	AL2
GND	AL3
GND	AL4
GND	AL5
GND	AL6
GND	AL7
GND	AL10

Cavium Confidential For Bowen Yang ALU 5/27/2014

Signal Name	Ball
GND	AL14
GND	AL19
GND	AL23
GND	AL27
GND	AL31
GND	AM1
GND	AM4
GND	AM7
GND	AM13
GND	AM20
GND	AM24
GND	AM28
GND	AM31
GND	C27
GND	B5
GND	J4
GND	A7
GND	D1
GPIO_0	AL24
GPIO_1	AM25
GPIO_2	AL25
GPIO_3	AK24
GPIO_4	AK23
GPIO_5	AM26
GPIO_6	AL26
GPIO_7	AM29
GPIO_8	AK25
GPIO_9	AJ24
GPIO_10	AM27
GPIO_11	AL28
GPIO_12	AL29
GPIO_13	AK27
GPIO_14	AJ26
GPIO_15	AM30
GPIO_16	AJ27
GPIO_17	AL30
GPIO_18	AK29
GPIO_19	AJ28
IO_VDD_SELECT	AA32
JTG_TCK	T31
JTG_TDI	U30
JTG_TDO	T30
JTG_TMS	V29
JTG_TRST_L	T29

Signal Name	Ball
MSC_CLKOUT	AA30
MSC_CLKOUT_ENABLE	AB32
MSC_SYS_CLKOUT	AA31
NC	C4
NC	B32
NC	E29
NC	G30
NC	U29
NC	V31
NC	W30
NC	AK28
NC	AM16
NC	AM32
PCIE0_MODE2	AL11
PCIE0_MODE1	AL12
PCIE0_MODE0	AK12
PCIE0_REV_LANES	AM11
PERST0_L	AJ13
PERST1_L	AM12
PERST2_L	AK13
PLL_DC_OK	AF32
PLL_MUL_0	AG4
PLL_MUL_1	AH3
PLL_MUL_2	AH4
PLL_MUL_3	AJ3
PLL_REF_CLK	AJ23
REMOTE_BOOT	AL13
RGM_COMP_DN	AM9
RGM_COMP_UP	AM8
RGM_REF_CLK	AJ9
RGM_VDD_SELECT	AK2
RGM0_RXC	AK10
RGM0_RXCTL	AJ11
RGM0_RXD_0	AM10
RGM0_RXD_1	AJ10
RGM0_RXD_2	AL9
RGM0_RXD_3	AK9
RGM0_TXC	AK6
RGM0_TXCTL	AK8
RGM0_TXD_0	AL8
RGM0_TXD_1	AJ7
RGM0_TXD_2	AJ6
RGM0_TXD_3	AK5
RSVD	D6

Signal Name	Ball
RSVD	AG3
RSVD	H31
RSVD	C28
RSVD	C7
RSVD	C8
RSVD	C9
RSVD	D7
RSVD	D8
RSVD	D9
RSVD	AB31
RSVD	AF31
RSVD	AG32
SMI_VDD_SELECT	AE30
SMI0_MDC	AC32
SMI0_MDIO	AB29
SMI1_MDC	AC30
SMI1_MDIO	AC29
SPI_CK	W32
SYS_PLL_MUL_0	AK4
SYS_PLL_MUL_1	AJ5
SYS_PLL_VDD	A8
THERMAL_D_N	AE32
THERMAL_D_P	AE31
TWS0_SCL	P32
TWS0_SDA	R30
TWS1_SCL	R32
TWS1_SDA	R31
UART0_CTS_L	AJ30
UART0_DCD_L	AK31
UART0_DTR_L	AL32
UART0_RTS_L	AK32
UART0_SIN	AH29
UART0_SOUT	AJ31
UART1_CTS_L	AH30
UART1_RTS_L	AJ32
UART1_SIN	AG29
UART1_SOUT	AH31
USB0_DM	C1
USB0_DP	B1
USB0_DVDD	N9
USB0_RESREF	D5
USB0_SSRX_M	A5
USB0_SSRX_P	A6
USB0_SSTX_M	A2

Cavium Confidential For Bowen Yang ALU 12/27/2014

Signal Name	Ball
USB0_SSTX_P	A3
USB0_VBUS	C5
USB0_VDD33	L10
USB0_VDD33	N10
USB0_VP	L9
USB0_VPTX	M9
USB1_DM	AH1
USB1_DP	AJ1
USB1_DVDD	Y9
USB1_RESREF	AL1
USB1_SSRX_M	AM5
USB1_SSRX_P	AM6
USB1_SSTX_M	AM2
USB1_SSTX_P	AM3
USB1_VBUS	AF4
USB1_VDD33	Y10
USB1_VDD33	AB10
USB1_VP	AB9
USB1_VPTX	AA9
VDD	L12
VDD	L13
VDD	L14
VDD	L15
VDD	L16
VDD	L17
VDD	M10
VDD	M11
VDD	M12
VDD	M13
VDD	M18
VDD	M19
VDD	M20
VDD	M21
VDD	M22
VDD	M23
VDD	N24
VDD	P10
VDD	P24
VDD	R24
VDD	T24
VDD	U23
VDD	V23
VDD	W10
VDD	W23

Signal Name	Ball
VDD	Y23
VDD	AA11
VDD	AA12
VDD	AA13
VDD	AA18
VDD	AA19
VDD	AA20
VDD	AA21
VDD	AB14
VDD	AB15
VDD	AB16
VDD	AB17
VDD	AB22
VDD	AB23
VDD_15	AD31
VDD_DDR	C11
VDD_DDR	C18
VDD_DDR	C24
VDD_DDR	D11
VDD_DDR	D18
VDD_DDR	D24
VDD_DDR	D26
VDD_DDR	D27
VDD_DDR	L18
VDD_DDR	L19
VDD_DDR	L20
VDD_DDR	L21
VDD_DDR	L22
VDD_DDR	L23
VDD_DDR	L24
VDD_DDR	M14
VDD_DDR	M15
VDD_DDR	M16
VDD_DDR	M17
VDD_DDR	M24
VDD_DDR	M29
VDD_DDR	N23
VDD_DDR	P23
VDD_DDR	P29
VDD_DDR	R23
VDD_DDR	T23
VDD_DDR	U24
VDD_DLM0	R10
VDD_DLM1	U10

Signal Name	Ball
VDD_DLM2	W9
VDD_DLMC	T9
VDD_IO	V24
VDD_IO	W24
VDD_IO	Y24
VDD_IO	Y29
VDD_IO	AA14
VDD_IO	AA15
VDD_IO	AA16
VDD_IO	AA17
VDD_IO	AA22
VDD_IO	AA23
VDD_IO	AB13
VDD_IO	AB18
VDD_IO	AB19
VDD_IO	AB20
VDD_IO	AB21
VDD_IO	AB24
VDD_IO	AD29
VDD_IO	AJ17
VDD_IO	AJ22
VDD_RGM	AA10
VDD_RGM	AB11
VDD_RGM	AB12
VDD_SMI	AA24
VDDH_DLM0	R9
VDDH_DLM1	T10
VDDH_DLM2	V10
VDDTX_DLM0	P9
VDDTX_DLM1	U9
VDDTX_DLM2	V9
VRM_I2_CLK	AG30
VRM_I2_DATA	AF29

Cavium Confidential For Bowen Yang ALU 12/27/2014

41.4 Balls Sorted in Numerical Order and Associated Signal Names

Ball	Signal Name
A1	GND
A2	USB0_SSTX_M
A3	USB0_SSTX_P
A4	GND
A5	USB0_SSRX_M
A6	USB0_SSRX_P
A7	GND
A8	SYS_PLL_VDD
A9	CORE_PLL_VDD
A10	GND
A11	DDR_CBS_0_P
A12	DDR_CB<2>
A13	DDR_CB<1>
A14	GND
A15	DDR_DQ<22>
A16	DDR_DQ<20>
A17	DDR_DQS_2_P
A18	DDR_DQ<18>
A19	DDR_DQ<17>
A20	GND
A21	GND
A22	DDR_DQ<6>
A23	DDR_DQ<4>
A24	DDR_DQS_0_P
A25	DDR_DQ<2>
A26	DDR_DQ<1>
A27	GND
A28	DDR_COMP_UP
A29	DDR_CK_1_P
A30	DDR_CK_0_P
A31	GND
A32	GND
B1	USB0_DP
B2	GND
B3	GND
B4	GND
B5	GND
B6	GND
B7	GND
B8	DDR_PLL_VDD
B9	GND

Ball	Signal Name
B10	GND
B11	DDR_CBS_0_N
B12	DDR_CB<3>
B13	DDR_CB<0>
B14	GND
B15	DDR_DQ<23>
B16	DDR_DQ<21>
B17	DDR_DQS_2_N
B18	DDR_DQ<19>
B19	DDR_DQ<16>
B20	GND
B21	GND
B22	DDR_DQ<7>
B23	DDR_DQ<5>
B24	DDR_DQS_0_N
B25	DDR_DQ<3>
B26	DDR_DQ<0>
B27	GND
B28	DDR_COMP_DN
B29	DDR_CK_1_N
B30	DDR_CK_0_N
B31	GND
B32	NC
C1	USB0_DM
C2	GND
C3	GND
C4	NC
C5	USB0_VBUS
C6	GND
C7	RSVD
C8	RSVD
C9	RSVD
C10	GND
C11	VDD_DDR
C12	DDR_DQ<30>
C13	DDR_DQ<28>
C14	DDR_DQS_3_N
C15	DDR_DQ<26>
C16	DDR_DQ<25>
C17	GND
C18	VDD_DDR

Ball	Signal Name
C19	DDR_DQ<14>
C20	DDR_DQ<13>
C21	DDR_DQS_1_P
C22	DDR_DQ<10>
C23	DDR_DQ<9>
C24	VDD_DDR
C25	GND
C26	GND
C27	GND
C28	RSVD
C29	DDR_CKE1
C30	GND
C31	DDR_ODT_0
C32	DDR_RAS_L
D1	GND
D2	GND
D3	DLM0_REF_CLK_P
D4	GND
D5	USB0_RESREF
D6	RSVD
D7	RSVD
D8	RSVD
D9	RSVD
D10	GND
D11	VDD_DDR
D12	DDR_DQ<31>
D13	DDR_DQ<29>
D14	DDR_DQS_3_P
D15	DDR_DQ<27>
D16	DDR_DQ<24>
D17	GND
D18	VDD_DDR
D19	DDR_DQ<15>
D20	DDR_DQ<12>
D21	DDR_DQS_1_N
D22	DDR_DQ<11>
D23	DDR_DQ<8>
D24	VDD_DDR
D25	GND
D26	VDD_DDR
D27	VDD_DDR

Cavium Confidential For Bowen Yang ALU 02/27/2014

Ball	Signal Name
D28	GND
D29	GND
D30	DDR_ODT_1
D31	DDR_CS1_L
D32	GND
E1	DLM0_RXN<0>
E2	GND
E3	DLM0_REF_CLK_N
E4	GND
E29	NC
E30	DDR_CKE0
E31	GND
E32	DDR_CAS_L
F1	DLM0_RXP<0>
F2	GND
F3	GND
F4	GND
F29	DDR_CS0_L
F30	GND
F31	DDR_WE_L
F32	DDR_BA<2>
G1	GND
G2	GND
G3	DLM0_TXN<0>
G4	GND
G29	GND
G30	NC
G31	DDR_A<15>
G32	DDR_A<0>
H1	DLM0_RXP<1>
H2	GND
H3	DLM0_TXP<0>
H4	GND
H29	DDR_A<10>
H30	DDR_A<12>
H31	RSVD
H32	GND
J1	DLM0_RXN<1>
J2	GND
J3	GND
J4	GND
J29	DDR_A<5>
J30	DDR_BA<0>
J31	GND

Ball	Signal Name
J32	DDR_A<3>
K1	GND
K2	GND
K3	DLM0_TXP<1>
K4	GND
K29	DDR_A<2>
K30	GND
K31	DDR_A<1>
K32	DDR_A<9>
L1	DLMC_REF_CLK0_P
L2	GND
L3	DLM0_TXN<1>
L4	GND
L9	USB0_VP
L10	USB0_VDD33
L11	GND
L12	VDD
L13	VDD
L14	VDD
L15	VDD
L16	VDD
L17	VDD
L18	VDD_DDR
L19	VDD_DDR
L20	VDD_DDR
L21	VDD_DDR
L22	VDD_DDR
L23	VDD_DDR
L24	VDD_DDR
L29	GND
L30	DDR_A<4>
L31	DDR_BA<1>
L32	DDR_A<11>
M1	DLMC_REF_CLK0_N
M2	GND
M3	GND
M4	GND
M9	USB0_VPTX
M10	VDD
M11	VDD
M12	VDD
M13	VDD
M14	VDD_DDR
M15	VDD_DDR

Ball	Signal Name
M16	VDD_DDR
M17	VDD_DDR
M18	VDD
M19	VDD
M20	VDD
M21	VDD
M22	VDD
M23	VDD
M24	VDD_DDR
M29	VDD_DDR
M30	DDR_A<6>
M31	DDR_A<7>
M32	GND
N1	DLMC_REF_CLK1_P
N2	GND
N3	GND
N4	DLM0_RESREF
N9	USB0_DVDD
N10	USB0_VDD33
N11	GND
N12	GND
N13	GND
N14	GND
N15	GND
N16	GND
N17	GND
N18	GND
N19	GND
N20	GND
N21	GND
N22	GND
N23	VDD_DDR
N24	VDD
N29	DDR_A<8>
N30	DDR_A<14>
N31	GND
N32	DDR_A<13>
P1	DLMC_REF_CLK1_N
P2	GND
P3	GND
P4	GND
P9	VDDTX_DLM0
P10	VDD
P11	GND

Cavium Confidential For Bowen Yang ALU 02/27/2014

Ball	Signal Name
P12	GND
P13	GND
P14	GND
P15	GND
P16	GND
P17	GND
P18	GND
P19	GND
P20	GND
P21	GND
P22	GND
P23	VDD_DDR
P24	VDD
P29	VDD_DDR
P30	GND
P31	DDR_RESET_L
P32	TWS0_SCL
R1	GND
R2	GND
R3	DLM1_TXN<0>
R4	GND
R9	VDDH_DLM0
R10	VDD_DLM0
R11	GND
R12	GND
R13	GND
R14	GND
R15	GND
R16	GND
R17	GND
R18	GND
R19	GND
R20	GND
R21	GND
R22	GND
R23	VDD_DDR
R24	VDD
R29	GND
R30	TWS0_SDA
R31	TWS1_SDA
R32	TWS1_SCL
T1	DLM1_RXN<0>
T2	GND
T3	DLM1_TXP<0>

Ball	Signal Name
T4	GND
T9	VDD_DLMC
T10	VDDH_DLM1
T11	GND
T12	GND
T13	GND
T14	GND
T15	GND
T16	GND
T17	GND
T18	GND
T19	GND
T20	GND
T21	GND
T22	GND
T23	VDD_DDR
T24	VDD
T29	JTG_TRST_L
T30	JTG_TDO
T31	JTG_TCK
T32	GND
U1	DLM1_RXP<0>
U2	GND
U3	GND
U4	GND
U9	VDDTX_DLM1
U10	VDD_DLM1
U11	GND
U12	GND
U13	GND
U14	GND
U15	GND
U16	GND
U17	GND
U18	GND
U19	GND
U20	GND
U21	GND
U22	GND
U23	VDD
U24	VDD_DDR
U29	NC
U30	JTG_TDI
U31	GND

Ball	Signal Name
U32	EJTG_TRST_L
V1	GND
V2	GND
V3	DLM1_TXP<1>
V4	GND
V9	VDDTX_DLM2
V10	VDDH_DLM2
V11	GND
V12	GND
V13	GND
V14	GND
V15	GND
V16	GND
V17	GND
V18	GND
V19	GND
V20	GND
V21	GND
V22	GND
V23	VDD
V24	VDD_IO
V29	JTG_TMS
V30	GND
V31	NC
V32	GND
W1	DLM1_RXP<1>
W2	GND
W3	DLM1_TXN<1>
W4	GND
W9	VDD_DLM2
W10	VDD
W11	GND
W12	GND
W13	GND
W14	GND
W15	GND
W16	GND
W17	GND
W18	GND
W19	GND
W20	GND
W21	GND
W22	GND
W23	VDD

Cavium Confidential For Bowen Yang ALU 02/27/2014

Ball	Signal Name
W24	VDD_IO
W29	GND
W30	NC
W31	EPR_CS
W32	SPI_CK
Y1	DLM1_RXN<1>
Y2	GND
Y3	GND
Y4	DLM1_RESREF
Y9	USB1_DVDD
Y10	USB1_VDD33
Y11	GND
Y12	GND
Y13	GND
Y14	GND
Y15	GND
Y16	GND
Y17	GND
Y18	GND
Y19	GND
Y20	GND
Y21	GND
Y22	GND
Y23	VDD
Y24	VDD_IO
Y29	VDD_IO
Y30	EPR_SK
Y31	EPR_DO
Y32	EPR_DI
AA1	GND
AA2	GND
AA3	DLM2_TXN<0>
AA4	GND
AA9	USB1_VPTX
AA10	VDD_RGM
AA11	VDD
AA12	VDD
AA13	VDD
AA14	VDD_IO
AA15	VDD_IO
AA16	VDD_IO
AA17	VDD_IO
AA18	VDD
AA19	VDD

Ball	Signal Name
AA20	VDD
AA21	VDD
AA22	VDD_IO
AA23	VDD_IO
AA24	VDD_SMI
AA29	GND
AA30	MSC_CLKOUT
AA31	MSC_SYS_CLKOUT
AA32	IO_VDD_SELECT
AB1	DLM2_RXN<0>
AB2	GND
AB3	DLM2_TXP<0>
AB4	GND
AB9	USB1_VP
AB10	USB1_VDD33
AB11	VDD_RGM
AB12	VDD_RGM
AB13	VDD_IO
AB14	VDD
AB15	VDD
AB16	VDD
AB17	VDD
AB18	VDD_IO
AB19	VDD_IO
AB20	VDD_IO
AB21	VDD_IO
AB22	VDD
AB23	VDD
AB24	VDD_IO
AB29	SMI0_MDIO
AB30	GND
AB31	RSVD
AB32	MSC_CLKOUT_ENABLE
AC1	DLM2_RXP<0>
AC2	GND
AC3	GND
AC4	DLM2_RESREF
AC29	SMI1_MDIO
AC30	SMI1_MDC
AC31	GND
AC32	SMI0_MDC
AD1	GND
AD2	GND
AD3	DLM2_TXP<1>

Ball	Signal Name
AD4	GND
AD29	VDD_IO
AD30	DLMC_VPH_SELECT_18
AD31	VDD_15
AD32	GND
AE1	DLM2_RXP<1>
AE2	GND
AE3	DLM2_TXN<1>
AE4	GND
AE29	GND
AE30	SMI_VDD_SELECT
AE31	THERMAL_D_P
AE32	THERMAL_D_N
AF1	DLM2_RXN<1>
AF2	GND
AF3	GND
AF4	USB1_VBUS
AF29	VRM_I2_DATA
AF30	GND
AF31	RSVD
AF32	PLL_DC_OK
AG1	GND
AG2	GND
AG3	RSVD
AG4	PLL_MUL_0
AG29	UART1_SIN
AG30	VRM_I2_CLK
AG31	GND
AG32	RSVD
AH1	USB1_DM
AH2	GND
AH3	PLL_MUL_1
AH4	PLL_MUL_2
AH29	UART0_SIN
AH30	UART1_CTS_L
AH31	UART1_SOUT
AH32	GND
AJ1	USB1_DP
AJ2	GND
AJ3	PLL_MUL_3
AJ4	GND
AJ5	SYS_PLL_MUL_1
AJ6	RGM0_TXD_2
AJ7	RGM0_TXD_1

Cavium Confidential For Bowen Yang ALU 02/27/2014

Ball	Signal Name
AJ8	GND
AJ9	RGM_REF_CLK
AJ10	RGM0_RXD_1
AJ11	RGM0_RXCTL
AJ12	GND
AJ13	PERST0_L
AJ14	BOOT_AD<6>
AJ15	BOOT_AD<2>
AJ16	GND
AJ17	VDD_IO
AJ18	BOOT_CE_7_L
AJ19	BOOT_CE_4_L
AJ20	BOOT_CE_0_L
AJ21	GND
AJ22	VDD_IO
AJ23	PLL_REF_CLK
AJ24	GPIO_9
AJ25	GND
AJ26	GPIO_14
AJ27	GPIO_16
AJ28	GPIO_19
AJ29	GND
AJ30	UART0_CTS_L
AJ31	UART0_SOUT
AJ32	UART1_RTS_L
AK1	GND
AK2	RGM_VDD_SELECT
AK3	GND
AK4	SYS_PLL_MUL_0
AK5	RGM0_TXD_3
AK6	RGM0_TXC
AK7	GND
AK8	RGM0_TXCTL
AK9	RGM0_RXD_3
AK10	RGM0_RXC
AK11	GND
AK12	PCIE0_MODE0
AK13	PERST2_L
AK14	BOOT_AD<7>
AK15	GND
AK16	BOOT_ALE_2
AK17	BOOT_ALE_0
AK18	GND
AK19	BOOT_CE_6_L

Ball	Signal Name
AK20	BOOT_CE_3_L
AK21	BOOT_CLE
AK22	GND
AK23	GPIO_4
AK24	GPIO_3
AK25	GPIO_8
AK26	GND
AK27	GPIO_13
AK28	NC
AK29	GPIO_18
AK30	GND
AK31	UART0_DCD_L
AK32	UART0_RTS_L
AL1	USB1_RESREF
AL2	GND
AL3	GND
AL4	GND
AL5	GND
AL6	GND
AL7	GND
AL8	RGM0_TXD_0
AL9	RGM0_RXD_2
AL10	GND
AL11	PCIE0_MODE2
AL12	PCIE0_MODE1
AL13	REMOTE_BOOT
AL14	GND
AL15	BOOT_AD<3>
AL16	BOOT_AD<0>
AL17	BOOT_ALE_3
AL18	BOOT_WAIT_L
AL19	GND
AL20	BOOT_WE_L
AL21	BOOT_CE_2_L
AL22	BOOT_DIR
AL23	GND
AL24	GPIO_0
AL25	GPIO_2
AL26	GPIO_6
AL27	GND
AL28	GPIO_11
AL29	GPIO_12
AL30	GPIO_17
AL31	GND

Ball	Signal Name
AL32	UART0_DTR_L
AM1	GND
AM2	USB1_SSTX_M
AM3	USB1_SSTX_P
AM4	GND
AM5	USB1_SSRX_M
AM6	USB1_SSRX_P
AM7	GND
AM8	RGM_COMP_UP
AM9	RGM_COMP_DN
AM10	RGM0_RXD_0
AM11	PCIE0_REV_LANES
AM12	PERST1_L
AM13	GND
AM14	BOOT_AD<5>
AM15	BOOT_AD<4>
AM16	NC
AM17	BOOT_AD<1>
AM18	BOOT_ALE_1
AM19	BOOT_OE_L
AM20	GND
AM21	BOOT_CE_5_L
AM22	BOOT_CE_1_L
AM23	CHIP_RESET_L
AM24	GND
AM25	GPIO_1
AM26	GPIO_5
AM27	GPIO_10
AM28	GND
AM29	GPIO_7
AM30	GPIO_15
AM31	GND
AM32	NC

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Appendix A

Cavium-Specific Core Instructions

This appendix describes Cavium-specific instructions, which are grouped in the categories listed below. Table A–1 lists the individual instructions.

- CPU Arithmetic (and Move) Instructions
  - CPU Insert/Extract Instructions
  - Large Multiply Instructions
  - Population Count Instructions
  - Set Equals/Set Not-Equals Instructions
  - Q15 Multiply and Accumulate Instructions
- CPU Branch and Jump Instructions
  - Bit-test Branches Instructions
- CPU Load, Store, Memory, and Control Instructions
  - Store Atomic Add Instructions
  - SYNC Instructions
  - Unaligned Load/Store Instructions
  - Integer Indexed Load Instructions
  - Memory Atomic Fetch and Add Instructions
  - Memory Atomic Increment Instructions
  - Zero Cache Block Instructions
- CPU Move and Privileged Instructions
- Coprocessor Move Instructions
  - CRC Instructions
  - Security Accelerator Instructions
    - 3DES
    - AES
    - Galois Field Multiplier
    - Hash (MD5/SHA-1/SHA-256/SHA-512)
    - KASUMI
    - SNOW 3G
    - SMS4
    - CAMELLIA
    - ZUC

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table A–1 Cavium-Specific Instructions

Category	Subcategory	Instruction	Detailed Description
CPU Arithmetic (and Move) Instructions		BADDU	See page 1633
		DMUL	See page 1634
	CPU Insert/Extract Instructions	EXTS	See page 1635
		EXTS32	See page 1636
		CINS	See page 1637
		CINS32	See page 1638
	Large Multiply Instructions	MTM0	See page 1639
		MTM1	See page 1641
		MTM2	See page 1642
		MTP0	See page 1643
		MTP1	See page 1644
		MTP2	See page 1645
		V3MULU	See page 1646
		VMM0	See page 1648
		VMULU	See page 1650
	Population Count	DPOP	See page 1652
		POP	See page 1653
	Set Equals/Set Not-Equals	SEQ	See page 1654
		SEQI	See page 1655
		SNE	See page 1656
		SNEI	See page 1657
	Q15 Multiply and Accumulate	QMAC.00	See page 1658
		QMAC.01	
		QMAC.02	
		QMAC.03	
		QMACS.00	See page 1660
		QMACS.01	
		QMACS.02	
		QMACS.03	
CPU Branch and Jump Instructions	**Bit-test Branches**	BBIT0	See page 1662
		BBIT032	See page 1663
		BBIT1	See page 1664
		BBIT132	See page 1665

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table A–1 Cavium-Specific Instructions (Continued)

Category	Subcategory	Instruction	Detailed Description	
CPU Load, Store, Memory, and Control Instructions	**PREF**	PREF 4	See page 1666	
		PREF 5		
		PREF 24		cnMIPS III only
		PREF 25		
		PREF 26		
		PREF 27		
		PREF 28		
		PREF 29		
		PREF 30		
		PREF 31		
	PREFX	PREF 4	See page 1669	cnMIPS III only
		PREF 5		
		PREFX 24		
		PREFX 25		
		PREFX 26		
		PREFX 27		
		PREFX 28		
		PREFX 29		
		PREFX 30		
		PREFX 31		
	Store Atomic Add	SAA	See page 1671	
		SAAD	See page 1673	
	SYNC	SYNCIOBDMA	See page 1675	
		SYNCS	See page 1676	
		SYNCW	See page 1678	
		SYNCWS	See page 1680	
	Unaligned Load/ Stores	ULD	See page 1682	
		ULW	See page 1685	
		USD	See page 1688	
		USW	See page 1691	
	Integer Indexed Loads	LBX	See page 1694	
		LBUX	See page 1695	
		LHX	See page 1696	
		LHUX	See page 1698	
		LWX	See page 1700	
		LWUX	See page 1702	
		LDX	See page 1704	
	Memory Atomic Fetch and Add	LAA	See page 1706	
		LAAD	See page 1708	
	Memory Atomic Swap	LAW	See page 1710	
		LAWD	See page 1712	
	Memory Atomic Increment	LAI	See page 1714	
		LAID	See page 1716	
	Memory Atomic Decrement	LAD	See page 1718	
		LADD	See page 1720	
	Memory Atomic Set	LAS	See page 1722	
		LASD	See page 1724	
	Memory Atomic Clear	LAC	See page 1726	
		LACD	See page 1728	
	Zero Cache Block	ZCB	See page 1730	
		ZCBT	See page 1731	

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table A–1 Cavium-Specific Instructions (Continued)

Category	Subcategory	Instruction	Detailed Description	
CPU Move and Privileged Instructions		CACHE	See page 1732	
		RDHWR30	See page 1737	
		RDHWR31	See page 1737	
	Read µTLB	TLBR when CvmMemCtl2[UTLBRW] = 1	See page 1739	cnMIPS III only
	Write µTLB	TLBWI when CvmMemCtl2[UTLBRW] = 1	See page 1740	cnMIPS III only
Coprocessor Move Instructions	**CRC**	CVM_MF_CRC_IV	See page 1741	
		CVM_MF_CRC_IV_REFLECT	See page 1742	
		CVM_MF_CRC_LEN	See page 1743	
		CVM_MF_CRC_POLYNOMIAL	See page 1744	
		CVM_MT_CRC_BYTE	See page 1745	
		CVM_MT_CRC_BYTE_REFLECT	See page 1746	
		CVM_MT_CRC_DWORD	See page 1747	
		CVM_MT_CRC_DWORD_REFLECT	See page 1748	
		CVM_MT_CRC_HALF	See page 1749	
		CVM_MT_CRC_HALF_REFLECT	See page 1750	
		CVM_MT_CRC_IV	See page 1751	
		CVM_MT_CRC_IV_REFLECT	See page 1752	
		CVM_MT_CRC_LEN	See page 1753	
		CVM_MT_CRC_POLYNOMIAL	See page 1754	
		CVM_MT_CRC_POLYNOMIAL_REFLECT	See page 1755	
		CVM_MT_CRC_VAR	See page 1756	
		CVM_MT_CRC_VAR_REFLECT	See page 1757	
		CVM_MT_CRC_WORD	See page 1758	
		CVM_MT_CRC_WORD_REFLECT	See page 1759	
	3DES	CVM_MF_3DES_IV	See page 1760	
		CVM_MF_3DES_KEY	See page 1761	
		CVM_MF_3DES_RESULT	See page 1762	
		CVM_MT_3DES_DEC	See page 1763	
		CVM_MT_3DES_DEC_CBC	See page 1764	
		CVM_MT_3DES_ENC	See page 1765	
		CVM_MT_3DES_ENC_CBC	See page 1766	
		CVM_MT_3DES_IV	See page 1767	
		CVM_MT_3DES_KEY	See page 1768	
		CVM_MT_3DES_RESULT	See page 1769	

Cavium Confidential For Binsen Yang 02/27/2014

Table A–1 Cavium-Specific Instructions (Continued)

Category	Subcategory	Instruction	Detailed Description
Coprocessor Move Instructions (Continued)	**AES**	CVM_MF_AES_INP0	See page 1770
		CVM_MF_AES_IV	See page 1771
		CVM_MF_AES_KEY	See page 1772
		CVM_MF_AES_KEYLENGTH	See page 1773
		CVM_MF_AES_RESINP	See page 1774
		CVM_MT_AES_DEC_CBC0	See page 1775
		CVM_MT_AES_DEC_CBC1	See page 1776
		CVM_MT_AES_DEC0	See page 1778
		CVM_MT_AES_DEC1	See page 1779
		CVM_MT_AES_ENC_CBC0	See page 1781
		CVM_MT_AES_ENC_CBC1	See page 1782
		CVM_MT_AES_ENC0	See page 1784
		CVM_MT_AES_ENC1	See page 1785
		CVM_MT_AES_IV	See page 1787
		CVM_MT_AES_KEY	See page 1788
		CVM_MT_AES_KEYLENGTH	See page 1789
		CVM_MT_AES_RESINP	See page 1790
	Galois Field Multiplier	CVM_MF_GFM_MUL	See page 1791
		CVM_MF_GFM_POLY	See page 1792
		CVM_MF_GFM_RESINP	See page 1793
		CVM_MT_GFM_MUL	See page 1794
		CVM_MT_GFM_POLY	See page 1795
		CVM_MT_GFM_RESINP	See page 1796
		CVM_MT_GFM_XOR0	See page 1797
		CVM_MT_GFM_XORMUL1	See page 1798
	Galois Field Multiplier REFLECT	CVM_MF_GFM_RESINP_REFLECT	See page 1800
		CVM_MT_GFM_MUL_REFLECT	See page 1801
		CVM_MT_GFM_XOR0_REFLECT	See page 1802
		CVM_MT_GFM_XORMUL1_ REFLECT	See page 1803
	Hash (MD5/SHA-1/ SHA-256/SHA-512)	CVM_MF_HSH_DAT	See page 1805
		CVM_MF_HSH_DATW	See page 1807
		CVM_MF_HSH_IV	See page 1809
		CVM_MF_HSH_IVW	See page 1811
		CVM_MT_HSH_DAT	See page 1813
		CVM_MT_HSH_DATW	See page 1815
		CVM_MT_HSH_IV	See page 1817
		CVM_MT_HSH_IVW	See page 1819
		CVM_MT_HSH_STARTMD5	See page 1821
		CVM_MT_HSH_STARTSHA	See page 1823
		CVM_MT_HSH_STARTSHA256	See page 1825
		CVM_MT_HSH_STARTSHA512	See page 1827
	KASUMI	CVM_MF_KAS_KEY	See page 1829
		CVM_MF_KAS_RESULT	See page 1830
		CVM_MT_KAS_KEY	See page 1831
		CVM_MT_KAS_RESULT	See page 1832
		CVM_MT_KAS_ENC	See page 1833
		CVM_MT_KAS_ENC_CBC	See page 1834

Cavium Confidential For Bowen Yang ALU 02/27/2024

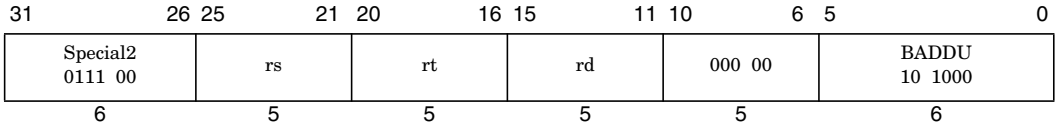
Table A–1 Cavium-Specific Instructions (Continued)

Category	Subcategory	Instruction	Detailed Description	
Coprocessor Move Instructions (Continued)	**SNOW 3G**	CVM_MF_SNOW3G_LFSR	See page 1835	
		CVM_MF_SNOW3G_FSM	See page 1837	
		CVM_MF_SNOW3G_RESULT	See page 1838	
		CVM_MT_SNOW3G_LFSR	See page 1839	
		CVM_MT_SNOW3G_FSM	See page 1841	
		CVM_MT_SNOW3G_RESULT	See page 1842	
		CVM_MT_SNOW3G_START	See page 1843	
		CVM_MT_SNOW3G_MORE	See page 1846	
	SMS4	CVM_MF_SMS4_IV	See page 1848	
		CVM_MF_SMS4_KEY	See page 1849	
		CVM_MF_SMS4_RESINP	See page 1850	
		CVM_MT_SMS4_DEC_CBC0	See page 1851	
		CVM_MT_SMS4_DEC_CBC1	See page 1852	
		CVM_MT_SMS4_DEC0	See page 1854	
		CVM_MT_SMS4_DEC1	See page 1855	
		CVM_MT_SMS4_ENC_CBC0	See page 1857	
		CVM_MT_SMS4_ENC_CBC1	See page 1858	
		CVM_MT_SMS4_ENC0	See page 1860	
		CVM_MT_SMS4_ENC1	See page 1861	
		CVM_MT_SMS4_IV	See page 1863	
		CVM_MT_SMS4_KEY	See page 1864	
		CVM_MT_SMS4_RESINP	See page 1865	
	CAMELLIA	CVM_MF_CAMELLIA_RESINP	See page 1866	cnMIPS III only
		CVM_MT_CAMELLIA_FL	See page 1867	
		CVM_MT_CAMELLIA_FLINV	See page 1869	
		CVM_MF_CAMELLIA_RESINP	See page 1876	
		CVM_MT_CAMELLIA_ROUND	See page 1871	
	ZUC	CVM_MF_ZUC_LFSR	See page 1877	cnMIPS III only
		CVM_MF_ZUC_FSM	See page 1879	
		CVM_MF_ZUC_RESULT	See page 1880	
		CVM_MF_ZUC_TRESULT	See page 1881	
		CVM_MT_ZUC_LFSR	See page 1882	
		CVM_MT_ZUC_FSM	See page 1884	
		CVM_MT_ZUC_RESULT	See page 1885	
		CVM_MT_ZUC_START	See page 1886	
		CVM_MT_ZUC_MORE	See page 1889	

Cavium Confidential For Bowei Yang ALU 02/27/2014

A.1 CPU Arithmetic (and Move) Instructions

Unsigned Byte Add **BADDU**



Format: BADDU rd, rs, rt **CVM**

Purpose:

To do an unsigned byte (8-bit) add.

Description:

The 8-bit byte value in GPR rt is added to the 8-bit byte value in GPR rs producing an 8-bit result that is zero-extended.

No integer overflow or any other exception occurs under any circumstance.

Operation:

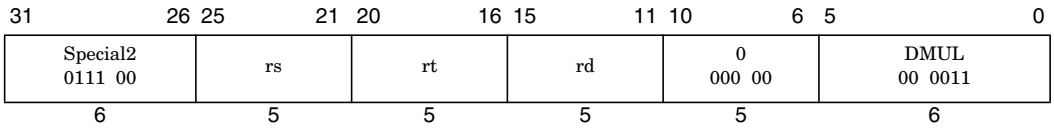
```
GPR[rd] = (GPR[rs] + GPR[rt]) & 0xFF
```

Exceptions:

None.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Multiply Doubleword to GPR — DMUL



Format: DMUL rd, rs, rt **CVM**

Purpose:

To multiply 64-bit signed integers and write the result to a GPR.

Description: rd = rs × rt

The 64-bit doubleword value in GPR rt is multiplied by the 64-bit value in GPR rs, treating both operands as signed values, to produce a 128-bit result. The low-order 64-bit doubleword of the result is written to GPR rd. The contents of HI, LO, P0, P1, P2, P3, P4, and P5 are unpredictable after the operation.

No arithmetic exception occurs under any circumstances.

Restrictions:

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

```
if Are64bitOperationsEnabled() then
   prod = GPR[rs] * GPR[rt]
   GPR[rd] = prod<63:0>
   HI = LO = P0 = P1 = P2 = unpredictable
else
   SignalException(ReservedInstruction)
endif
```

Exceptions:

Reserved Instruction

Programming Notes:

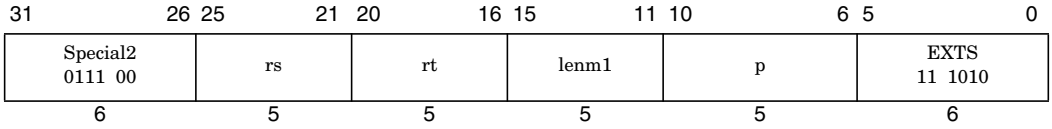
The integer multiply operation proceeds asynchronously and other CPU instructions can execute before it is complete. An attempt to read GPR rd before the results are written interlocks until the results are ready. Asynchronous execution does not affect the program result, but offers an opportunity for performance improvement by scheduling the multiply so that other instructions can execute in parallel.

Programs that require overflow detection must check for it explicitly.

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.1.1 CPU Insert/Extract Instructions

Extract a Signed Bit Field **EXTS**



Format: EXTS rt, rs, p, lenm1 **CVM**

Purpose:

To extract and sign-extend a bit field that starts from the lower 32 bits of a register.

Description: rt = sign-extend(rs<p+lenm1:p>, lenm1)

Bit locations p + lenm1 to p are extracted from rs and the result is written into the lowest bits of destination register rt. The remaining bits in rt are a sign-extension of the most-significant bit of the bit field (i.e. rt<63:lenm1> are all duplicates of the source-register bit rs<p+lenm1>).

Restrictions:

The p and lenm1 fields are only 5-bits, so the widest allowed bit field is 32-bits.

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

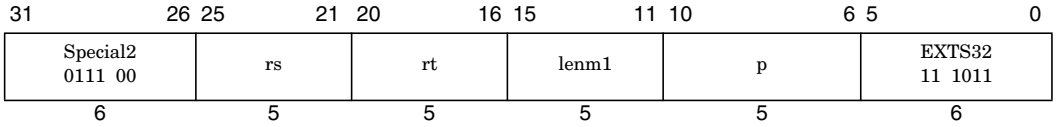
```
if Are64bitOperationsEnabled() then
  GPR[rt] ← sign-extend(GPR[rs]<p+lenm1:p>, lenm1)
else
  SignalException(ReservedInstruction)
endif
```

Exceptions:

Reserved Instruction.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Extract a Signed Bit Field Plus 32 — EXTS32



Format: EXTS32 rt, rs, p, lenm1 **CVM**

Purpose:

To extract and sign-extend a bit field that starts from the upper 32 bits of a register.

Description: rt = sign-extend(rs<p+32+lenm1:p+32>, lenm1)

Bit locations p + 32 + lenm1 to p + 32 are extracted from rs and the result is written into the lowest bits of destination register rt. The remaining bits in rt are a sign-extension of the most-significant bit of the bit field (i.e. rt<63:lenm1> are all duplicates of the source-register bit rs<p+32+lenm1>).

Restrictions:

The p and lenm1 fields are only 5-bits, so the widest allowed bit field is 32-bits.

The result register GPR[rt] is unpredictable when p+32+lenm1 is larger than 63.

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

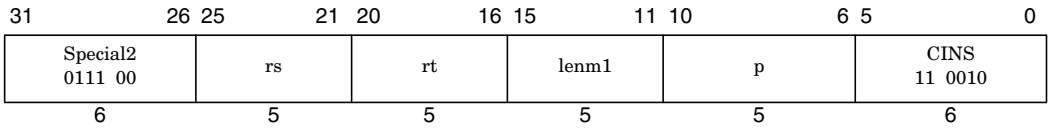
```
if Are64bitOperationsEnabled() then
  GPR[rt] ← sign-extend(GPR[rs]<p+32+lenm1:p+32>, lenm1)
else
  SignalException(ReservedInstruction)
endif
```

Exceptions:

Reserved Instruction.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Clear and Insert a Bit Field — CINS



Format: CINS rt, rs, p, lenm1 **CVM**

Purpose:

To insert a bit field that starts in the lower 32 bits of a register and clear the rest of the register.

Description: rt = rs<lenm1:0> << p

The contents of general-purpose register rt from bit location p + lenm1 to bit location p are filled with the contents of general-purpose register rs from bit location lenm1 to bit location 0. The remaining bits in rt are zeroed by this instruction.

Restrictions:

The p and lenm1 fields are only 5-bits, so the widest allowed bit field is 32-bits.

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

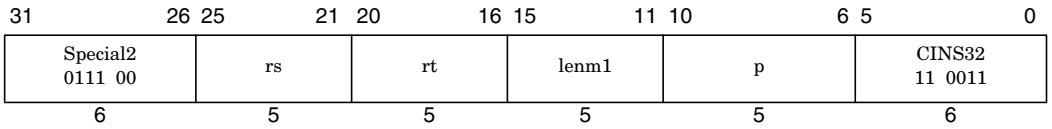
```
if Are64bitOperationsEnabled() then
  GPR[rt] ← GPR[rs]<lenm1:0> << p
else
  SignalException(ReservedInstruction)
endif
```

Exceptions:

Reserved Instruction.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Clear and Insert a Bit Field Plus 32 — CINS32



Format: CINS32 rt, rs, p, lenm1 **CVM**

Purpose:

To insert a bit field that starts in the upper 32 bits of a register and clear the rest of the register.

Description: rt = rs<lenm1:0> << (p + 32)

The contents of general-purpose register rt from bit location p + 32 + lenm1 to bit location p + 32 are filled with the contents of general-purpose register rs from bit location lenm1 to bit location 0. The remaining bits in rt are zeroed by this instruction.

Restrictions:

The p and lenm1 fields are only 5-bits, so the widest allowed bit field is 32-bits.

The result register GPR[rt] is unpredictable when p+32+lenm1 is larger than 63.

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

```
if Are64bitOperationsEnabled() then
  GPR[rt] ← GPR[rs]<lenm1:0> << (p + 32)
else
  SignalException(ReservedInstruction)
endif
```

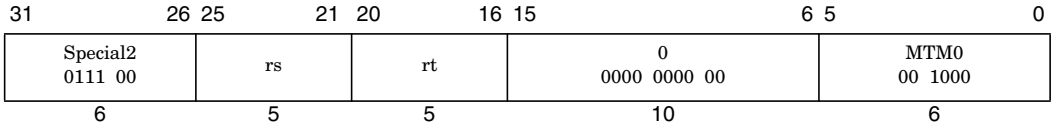
Exceptions:

Reserved Instruction.

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.1.2 Large Multiply Instructions

Load Multiplier Register MPL0 and MPL3 **MTM0**



Format: MTM0 rs, rt **CVM**

Purpose:

To load the multiplier registers MPL0 and MPL3 and zero MPL1, P0, P1, P2, P3, P4, and P5.

Description: MPL0 = GPR[rs]; MPL3 = GPR[rt]; MPL1 = 0; P0, P1, P2, P3, P4, P5 = 0

The 64-bit double-word value in GPR rs is loaded into the multiplier register MPL0, the 64-bit double-word value in GPR rt is loaded into the multiplier register MPL3, and MPL1 as well as P0-P5 are zeroed. MPL2, MPL4, and MPL5 are unpredictable after this operation executes. .

Register	MPL0	MPL1	MPL2	MPL3	MPL4	MPL5
State after execution	GPR[rs]	0	Unpredictable	GPR[rt]	Unpredictable	Unpredictable

Register	P0	P1	P2	P3	P4	P5
State after execution	0	0	0	0	0	0

Restrictions:

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

```
if Are64bitOperationsEnabled() and !CvmCtl[NOMUL] then
  MPL0<63:0> = GPR[rs]
  MPL3<63:0> = GPR[rt]
  MPL1<63:0> = 0;
  MPL2 = unpredictable
  MPL4 = unpredictable
  MPL5 = unpredictable
  P0 = 0
  P1 = 0
  P2 = 0
  P3 = 0
  P4 = 0
  P5 = 0
else
  SignalException(ReservedInstruction)
endif
```

Exceptions:

Reserved Instruction

Programming Notes:

Cavium Confidential For Bowen Yang ALU 02/27/2014

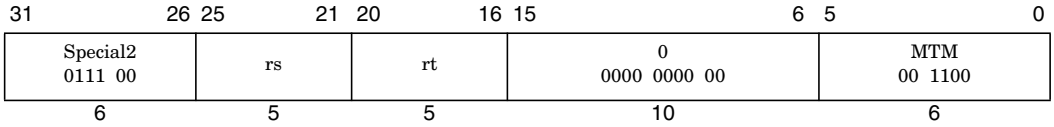
See “Special MUL Topics” on page 203 for a method to read out the MPL0 and MPL3 values.

MTM0 is backward-compatible with cnMIPS II MTM0. cnMIPS II “MTM0 rs” maps to “MTM0 rs, $0” in cnMIPS III.

MTM0 sets MPL1 to zero in hardware for VMULU to be backward-compatible with cnMIPS II VMULU definition. The cnMIPS II MTM0 makes MPL1 unpredictable.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Load Multiplier Register MPL1 and MPL4 — MTM1



Format: MTM1 rs, rt **CVM**

Purpose:

To load the multiplier register MPL1 and MPL3 and zero P0, P1, P2, P3, P4, and P5.

Description: MPL1 = GPR[rs]; MPL4 = GPR[rt]; P0, P1, P2, P3, P4, P5 = 0

The 64-bit doubleword value in GPR rs is loaded into the multiplier register MPL1, the 64-bit doubleword value in GPR rt is loaded into the multiplier register MPL2, and P0–P5 are zeroed. MPL2 and MPL5 are unpredictable after this operation.

Register	MPL0	MPL1	MPL2	MPL3	MPL4	MPL5
State after execution	Unchanged	GPR[rs]	Unpredictable	Unchanged	GPR[rt]	Unpredictable

Register	P0	P1	P2	P3	P4	P5
State after execution	0	0	0	0	0	0

Restrictions:

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

```
if Are64bitOperationsEnabled() and !CvmCtl[NOMUL] then
  MPL1<63:0> = GPR[rs]
  MPL4<63:0> = GPR[rt]
  MPL2 = unpredictable
  MPL5 = unpredictable
  P0 = 0
  P1 = 0
  P2 = 0
  P3 = 0
  P4 = 0
  P5 = 0
else
  SignalException(ReservedInstruction)
endif
```

Exceptions:

Reserved Instruction

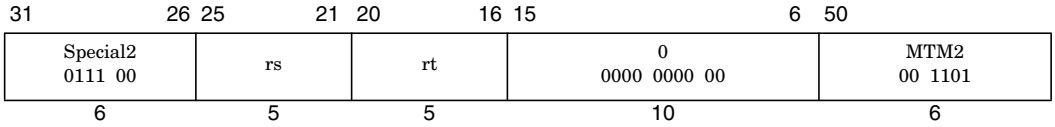
Programming Notes:

See “Special MUL Topics” on page 203 for a method to read out the MPL1 and MPL4 values.

MTM1 is backward-compatible with cnMIPS II MTM1. cnMIPS II “MTM1 rs” maps to “MTM1 rs, $0” in cnMIPS III.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Multiplier Register MPL2 and MPL5 — MTM2



Format: MTM2 rs, rt **CVM**

Purpose:

To load the multiplier register MPL2 and MPL5 and zero P0, P1, P2, P3, P4, and P5.

Description: MPL2 = GPR[rs]; MPL5 = GPR[rt]; P0, P1, P2, P3, P4, P5 = 0

The 64-bit doubleword value in GPR rs is loaded into the multiplier register MPL2, the 64-bit doubleword value in GPR rt is loaded into the multiplier register MPL5, and P0–P5 are zeroed.

Register	MPL0	MPL1	MPL2	MPL3	MPL4	MPL5
State after execution	Unchanged	Unchanged	GPR[rs]	Unchanged	Unchanged	GPR[rt]

Register	P0	P1	P2	P3	P4	P5
State after execution	0	0	0	0	0	0

Restrictions:

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

```
if Are64bitOperationsEnabled() and !CvmCtl[NOMUL] then
  MPL2<63:0> = GPR[rs]
  MPL5<63:0> = GPR[rt]
  P0 = 0
  P1 = 0
  P2 = 0
  P3 = 0
  P4 = 0
  P5 = 0
else
  SignalException(ReservedInstruction)
endif
```

Exceptions:

Reserved Instruction

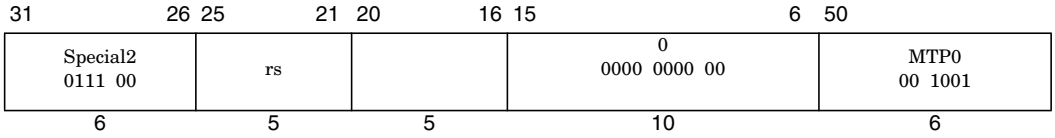
Programming Notes:

See “Special MUL Topics” on page 203 for a method to read out the MPL2 and MPL5 values.

MTM2 is backward-compatible with cnMIPS II MTM2. cnMIPS II “MTM2 rs” maps to “MTM2 rs, $0” in cnMIPS III.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Multiplier Register P0 and P3 — MTP0



Format: MTP0 rs, rt **CVM**

Purpose:

To load the product registers P0 and P3.

Description: P0 = GPR[rs]; P1 = 0; P3 = GPR[rt]

The 64-bit doubleword value in GPR rs is loaded into the product register P0. The 64-bit doubleword value in GPR rt is loaded into the product register P3. P1, P2, P4, and P5 are unpredictable after this operation.

Register	P0	P1	P2	P3	P4	P5
State after execution	GPR[rs]	0	Unpredictable	GPR[rt]	Unpredictable	Unpredictable

Restrictions:

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

```
if Are64bitOperationsEnabled() and !CvmCtl[NOMUL] then
  P0 = GPR[rs]
  P3 = GPR[rt]
  P1 = unpredictable
  P2 = unpredictable
  P4 = unpredictable
  P5 = unpredictable
else
  SignalException(ReservedInstruction)
endif
```

Exceptions:

Reserved Instruction

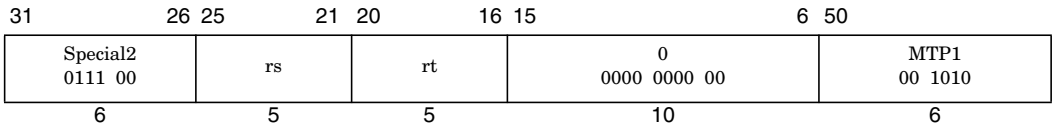
Programming Notes:

See "Special MUL Topics" on page 203 for a method to read out the P0 and P3 values.

MTP0 is backward-compatible with cnMIPS II MTP0. cnMIPS II "MTP0 rs" maps to "MTP0 rs, $0" in cnMIPS III.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Multiplier Register P1 and P4 — MTP1



Format: MTP1 rs, rt **CVM**

Purpose:

To load the product registers P1 and P4.

Description: P1 = GPR[rs]; P4 = GPR[rt]

The 64-bit doubleword value in GPR rs is loaded into the product register P1. The 64-bit doubleword value in GPR rs is loaded into the product register P4. P2 and P5 are unpredictable after this operation. .

Register	P0	P1	P2	P3	P4	P5
State after execution	Unchanged	GPR[rs]	Unpredictable	Unchanged	GPR[rt]	Unpredictable

Restrictions:

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

```
if Are64bitOperationsEnabled() and !CvmCtl[NOMUL] then
  P1 = GPR[rs]
  P4 = GPR[rt]
  P2 = unpredictable
  P5 = unpredictable
else
  SignalException(ReservedInstruction)
endif
```

Exceptions:

Reserved Instruction

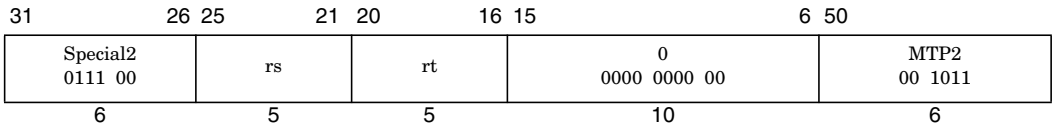
Programming Notes:

See "Special MUL Topics" on page 203 for a method to read out the P1 and P4 values.

MTP1 is backward-compatible with cnMIPS II MTP1. cnMIPS II "MTP1 rs" maps to "MTP1 rs, $0" in cnMIPS III.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Multiplier Register P2 and P5 — MTP2



Format: MTP2 rs, rt **CVM**

Purpose:

To load the product registers P2 and P5.

Description: P2 = GPR[rs]; P5 = GPR[rt]

The 64-bit doubleword value in GPR rs is loaded into the product register P2. The 64-bit doubleword value in GPR rt is loaded into the product register P5.

Register	P0	P1	P2	P3	P4	P5
State after execution	Unchanged	Unchanged	GPR[rs]	Unchanged	Unchanged	GPR[rt]

Restrictions:

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

```
if Are64bitOperationsEnabled() and !CvmCtl[NOMUL] then
  P2 = GPR[rs]
  P5 = GPR[rt]
else
  SignalException(ReservedInstruction)
endif
```

Exceptions:

Reserved Instruction

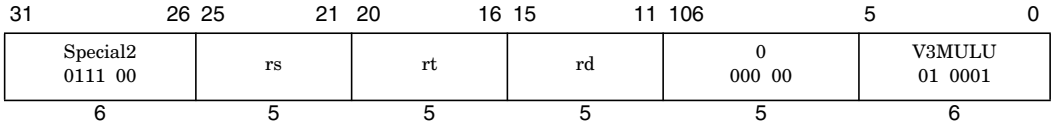
Programming Notes:

See “Special MUL Topics” on page 203 for a method to read out the P2 and P5 values.

MTP2 is backward-compatible with cnMIPS II MTP2. cnMIPS II “MTP2 rs” maps to “MTP2 rs, $0” in cnMIPS III.

Cavium Confidential For Bowen Yang ALU 02/27/2014

384-bit × 64-bit Unsigned Multiply and Add — V3MULU



Format: V3MULU rd, rs, rt **CVM**

Purpose:

To perform a 384-bit x 64-bit unsigned multiply and add yielding a 448-bit result.

Description: $(P5 \| P4 \| P3 \| P2 \| P1 \| P0 \| GPR[rd]) = (0^{64} \| P5 \| P4 \| P3 \| P2 \| P1 \| P0) + (0^{384} \| GPR[rt]) + GPR[rs] \times (MPL5 \| MPL4 \| MPL3 \| MPL2 \| MPL1 \| MPL0)$

The 64-bit doubleword value in GPR rs is multiplied by the 384-bit multiplier registers MPL0-MPL5, treating both as unsigned values, producing a 448-bit result. The 64-bit doubleword value in GPR rt and the 64-bit product registers P0–P5 are zero-extended and added to the 448-bit result. The least-significant 64 bits of the result is placed in GPR rd. The most-significant 384 bits of the result is placed into the product registers P0–P5.

Register	MPL0	MPL1	MPL2	MPL3	MPL4	MPL5
State after execution	Unchanged	Unchanged	Unchanged	Unchanged	Unchanged	Unchanged

Register	P0	P1	P2	P3	P4	P5
State after execution	Updated	Updated	Updated	Updated	Updated	Updated

Restrictions:

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

If a V3MULU is preceded by an MTP* instruction (without an intervening MTM*/VMM0, VMULU/V3MULU instruction), the MTP* instruction must be preceded by an MTM* or VMM0 instruction (without any intervening VMULU/V3MULU instructions).

If a V3MULU precedes a VMULU/VMM0, there must be an intervening MTM* between the two.

No overflow or other arithmetic exception occurs under any circumstances.

Like cnMIPS II, cnMIPS III requires that MTM* and MTP* be performed in order. That is, for correct operation in both cnMIPS II and cnMIPS III, MTM0 must precede MTM1, which must precede MTM2. Similarly, MTP0 must precede MTP1, which must precede MTP2.

Cavium Confidential For Bowen Yang ALU 02/7/2014

Operation:

```
 if Are64bitOperationsEnabled() and !CvmCtl[NOMUL] then
   product_register<383:0> = MPL5<63:0> || MPL4<63:0> || MPL3<63:0> || MPL2<63:0> || MPL1<63:0> ||
MPL0<63:0>
   product<447:0> = GPR[rs]<63:0> * product_register<383:0>
   Pext<447:0> = 0^64 || P5<63:0> || P4<63:0> || P3<63:0> || P2<63:0> || P1<63:0> || P0<63:0>
   rtext<447:0> = 0^384 || GPR[rt]<63:0>
   sum<447:0> = Pext<447:0> + rtext<447:0> + product<447:0>
   GPR[rd] = sum<63:0>
   P5 = sum<447:384>
   P4 = sum<383:320>
   P3 = sum<319:256>
   P2 = sum<255:192>
   P1 = sum<191:128>
   P0 = sum<127:64>
 else
   SignalException(ReservedInstruction)
 endif
```

Exceptions:

Reserved Instruction

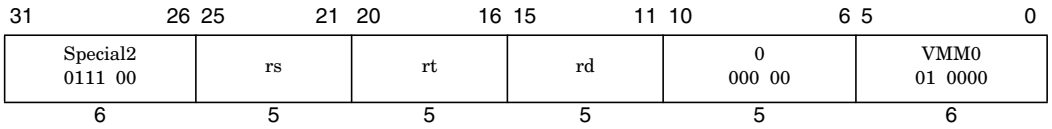
Programming Notes:

The cnMIPS III V3MULU performs a 384-bit x 64-bit unsigned multiplication, unlike the cnMIPS II V3MULU that performs a 192-bit x 64-bit multiplication. Hence, cnMIPS III requires three extra multiplier registers (MPL3, MPL4, and MPL5) and three extra product registers (P3, P4, and P5).

The cnMIPS III V3MULU is backward-compatible with cnMIPS II V3MULU. This is because when MPL3-MPL5 and P3-P5 are zero, so you will get the 256-bit unsigned product.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

64-bit Unsigned Multiply and Add Move — VMM0



Format: VMM0 rd, rs, rt **CVM**

Purpose:

To perform a 64-bit × 64-bit unsigned multiply and add yielding a 64-bit result.

Description:

```
temp = P0 + GPR[rt] + GPR[rs] × MPL0
MPL0 = temp
GPR[rd] = temp
P0, P1, P2, P3, P4, P5 = 0
MPL1 = 0
```

The 64-bit doubleword value in GPR rs is multiplied by the 64-bit product register MPL0, treating both as unsigned values. The 64-bit doubleword value in GPR rt and the 64-bit product register P0 are added to the result. The 64-bit result is placed in GPR rd and the multiplier register MPL0. The multiplier register MPL0 and product registers P0–P5 are zeroed.

MPL2, MPL3, MPL4, and MPL5 are unpredictable after this operation executes.

Register	MPL0	MPL1	MPL2	MPL3	MPL4	MPL5
State after execution	Updated	0	Unpredictable	Unpredictable	Unpredictable	Unpredictable

Register	P0	P1	P2	P3	P4	P5
State after execution	0	0	0	0	0	0

Restrictions:

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

If a VMM0 instruction is preceded by an MTP* instruction (without an intervening MTM*/VMM0, VMULU/V3MULU instruction), the MTP* instruction must be preceded by an MTM* or VMM0 instruction (without any intervening VMULU/V3MULU instructions).

If a VMM0 is preceded by a V3MULU, there must be an intervening MTM* between the two.

Following a VMM0 instruction, the results of an MTP* instruction are unpredictable until after an MTM* instruction executes.

No overflow or other arithmetic exception occurs under any circumstances.

Operation:

```
if Are64bitOperationsEnabled() and !CvmCtl[NOMUL] then
  product<63:0> = GPR[rs]<63:0> * MPL0<63:0>
  sum<63:0> = P0<63:0> + GPR[rt]<63:0> + product<63:0>
  GPR[rd] = sum<63:0>
  MPL0 = sum<63:0>
```

Cavium Confidential For Bowen Yang ALU 09/27/2014

```
  MPL1 = 0
  MPL2 = unpredictable
  MPL3 = unpredictable
  MPL4 = unpredictable
  MPL5 = unpredictable
  P0 = 0
  P1 = 0
  P2 = 0
  P3 = 0
  P4 = 0
  P5 = 0
else
  SignalException(ReservedInstruction)
endif
```

Exceptions:

Reserved Instruction

Programming Notes:

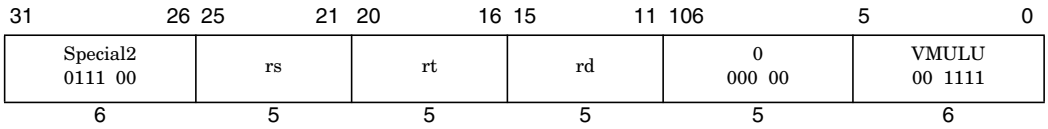
VMM0 rd, rs, rt is functionally identical to the two-instruction sequence:

```
VMULU rd, rs, rt
MTM0 rd
```

The cnMIPS III VMM0 is backward-compatible with cnMIPS II VMM0. Although cnMIPS III VMM0 sets MPL1 to zero in hardware, MPL1 becomes unpredictable after a cnMIPS II VMM0. Hence, cnMIPS III VMM0 is backward-compatible with cnMIPS II VMM0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

128-bit × 64-bit Unsigned Multiply and Add — VMULU



Format: VMULU rd, rs, rt **CVM**

Purpose:

To perform a 64-bit x 64-bit unsigned multiply and add yielding a 128-bit result.

Description: $(P1 \| P0 \| GPR[rd]) = (0^{64} \| P1 \| P0) + (0^{128} \| GPR[rt]) + GPR[rs] \times (MPL1 \| MPL0)$

The 64-bit doubleword value in GPR rs is multiplied by the 128-bit multiplier registers MPL0-MPL1, treating both as unsigned values, producing a 192-bit result. The 64-bit doubleword value in GPR rt and the 64-bit product registers P0-P1 are zero-extended and added to the 192-bit result. The least-significant 64 bits of the result is placed in GPR rd. The most-significant 128 bits of the result is placed into the product registers P0-P1.

P2-P5 and MPL2-MPL5 are unpredictable after this operation executes.

Register	MPL0	MPL1	MPL2	MPL3	MPL4	MPL5
State after execution	Unchanged	Unchanged	Unpredictable	Unpredictable	Unpredictable	Unpredictable

Register	P0	P1	P2	P3	P4	P5
State after execution	Updated	Updated	Unpredictable	Unpredictable	Unpredictable	Unpredictable

Restrictions:

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

If a VMULU is preceded by an MTP1 instruction (without an intervening MTM*/ VMM0, VMULU/V3MULU instruction), the MTP1 instruction must be preceded by an MTM*, MTP0, or VMM0 instruction (without any intervening VMULU/V3MULU instructions).

If a VMULU is preceded by a V3MULU, there must be an intervening MTM* between the two.

No overflow or other arithmetic exception occurs under any circumstances.

Operation:

```
if Are64bitOperationsEnabled() and !CvmCtl[NOMUL] then
  product_register<127:0> = MPL1<63:0> || MPL0<63:0>
  product<191:0> = GPR[rs]<63:0> * product_register<127:0>
  Pext<191:0> = 0^64 || P1<63:0> || P0<63:0>
  rtext<191:0> = 0^128 || GPR[rt]<63:0>
  sum<191:0> = P2ext<127:0> + rtext<127:0> + product<127:0>
  GPR[rd] = sum<63:0>
  MPL2 = unpredictable
  MPL3 = unpredictable
  MPL4 = unpredictable
  MPL5 = unpredictable

  P0 = sum<127:64>
  P1 = sum<191:128>
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

```
    P2 = unpredictable
    P3 = unpredictable
    P4 = unpredictable
    P5 = unpredictable
  else
    SignalException(ReservedInstruction)
  endif
```

Exceptions:

Reserved Instruction

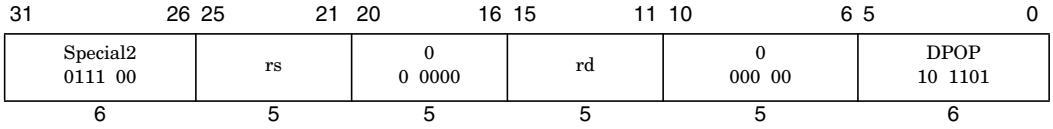
Programming Notes:

The cnMIPS III VMULU is backward-compatible with cnMIPS II VMULU. This is because when MPL1-MPL5 and P1-P5 are zero, so you will get the 128-bit unsigned product.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

A.1.3 Population Count Instructions

Count Ones in a Doubleword **DPOP**



Format: DPOP rd, rs **CVM**

Purpose

Count the number of ones in a doubleword

Description: rd = count_ones(rs)

Bits 63..0 of GPR rs are scanned. The number of ones is counted and the result is written to GPR rd.

Restrictions:

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

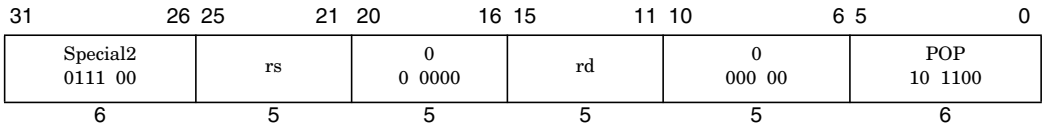
```
if Are64bitOperationsEnabled() then
  temp = 0
  for i in 63 .. 0
    if GPR[rs]<i> = 1 then
      temp++
    endif
  endfor
  GPR[rd] = temp
else
  SignalException(ReservedInstruction)
endif
```

Exceptions:

Reserved Instruction.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Count Ones in a Word — POP



Format: POP rd, rs **CVM**

Purpose

Count the number of ones in a word

Description: rd = count_ones(rs)

Bits 31..0 of GPR rs are scanned. The number of ones is counted and the result is written to GPR rd.

Restrictions:

None.

Operation:

```
temp = 0
for i in 31 .. 0
  if GPR[rs]<i> = 1 then
    temp++
  endif
endfor
GPR[rd] = temp
```

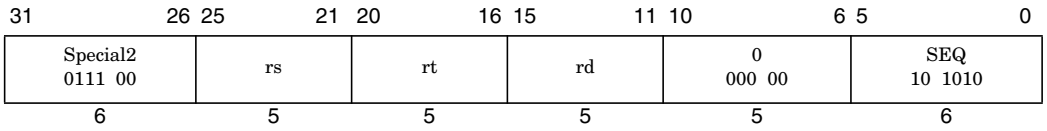
Exceptions:

None.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

A.1.4 Set Equals/Set Not-Equals Instructions

Set on Equal **SEQ**



Format: SEQ rd, rs, rt **CVM**

Purpose:

To record the result of an equals comparison.

Description: rd = (rs == rt)

Compare the contents of GPR rs and GPR rt and record the Boolean result of the comparison in GPR rd. If GPR rs equals GPR rt, the result is 1 (true); otherwise, it is 0 (false).

The arithmetic comparison does not cause an Integer Overflow exception.

Restrictions:

None.

Operation:

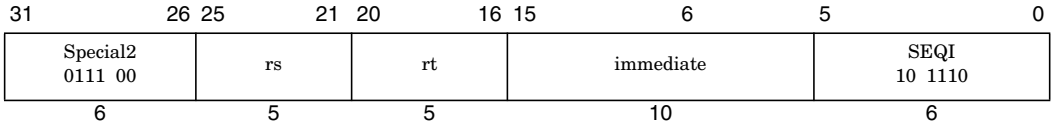
```
if GPR[rs] == GPR[rt] then
  GPR[rd] = 0^63 || 1
else
  GPR[rd] = 0^64
endif
```

Exceptions:

None.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Set on Equal Immediate SEQI



Format: SEQI rt, rs, immediate **CVM**

The immediate is a 10-bit sign-extended value contained in <15:6> of the instruction.

Purpose:

To record the result of an equals comparison with a constant.

Description: rt = (rs == immediate)

Compare the contents of GPR rs and the 10-bit signed immediate and record the Boolean result of the comparison in GPR rt. If GPR rs equals immediate, the result is 1 (true); otherwise, it is 0 (false).

The arithmetic comparison does not cause an Integer Overflow exception.

Restrictions:

None.

Operation:

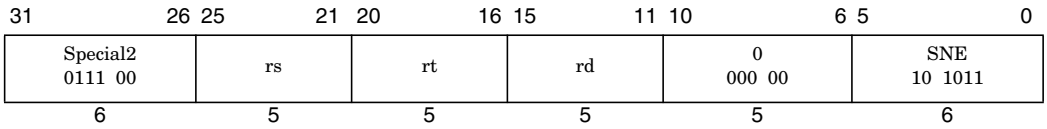
```
if GPR[rs] == sign_extend(immediate) then
   GPR[rt] = 0^63 || 1
else
   GPR[rt] = 0^64
endif
```

Exceptions:

None.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Set on Not Equal — SNE



Format: SNE rd, rs, rt **CVM**

Purpose:

To record the result of a not equals comparison.

Description: rd = (rs != rt)

Compare the contents of GPR rs and GPR rt and record the Boolean result of the comparison in GPR rd. If GPR rs equals GPR rt, the result is 0 (false); otherwise, it is 1 (true).

The arithmetic comparison does not cause an Integer Overflow exception.

Restrictions:

None.

Operation:

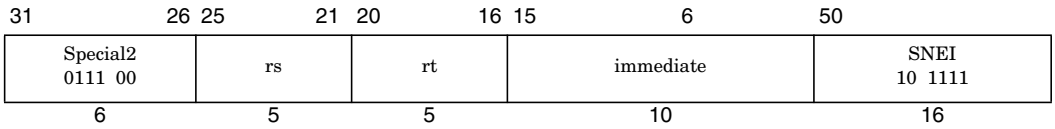
```
if GPR[rs] != GPR[rt] then
   GPR[rd] = 0^63 || 1
else
   GPR[rd] = 0^64
endif
```

Exceptions:

None.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Set on Not Equal Immediate — SNEI



Format: SNEI rt, rs, immediate **CVM**

The immediate is a 10-bit sign-extended value contained in <15:6> of the instruction.

Purpose:

To record the result of a not equals comparison with a constant.

Description: rt = (rs != immediate)

Compare the contents of GPR rs and the 10-bit signed immediate and record the Boolean result of the comparison in GPR rt. If GPR rs equals immediate, the result is 0 (false); otherwise, it is 1 (true).

The arithmetic comparison does not cause an Integer Overflow exception.

Restrictions:

None.

Operation:

```
if GPR[rs] != sign_extend(immediate) then
   GPR[rt] = 0^63 || 1
else
   GPR[rt] = 0^64
endif
```

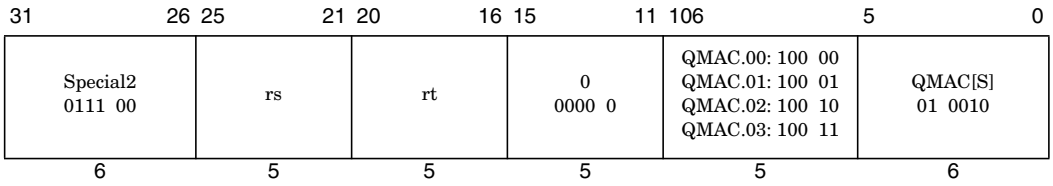
Exceptions:

None.

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.1.5 Q15 Multiply and Accumulate Instructions

Q15 Multiply and Accumulate **QMAC.0x**



Format: QMAC.00 rs, rt
QMAC.01 rs, rt
QMAC.02 rs, rt
QMAC.03 rs, rt **CVMv2**

Purpose:

Q15 multiply and accumulate in HI/LO.

To multiply one Q15 value from a quad halfword with another Q15 value and accumulate the result, with saturation on the multiply.

Description:

```
HI,LO ← HI,LO + sat32(rt<15:0> × rs<15:0> × 2)  // QMAC.00
HI,LO ← HI,LO + sat32(rt<15:0> × rs<31:16> × 2) // QMAC.01
HI,LO ← HI,LO + sat32(rt<15:0> × rs<47:32> × 2) // QMAC.02
HI,LO ← HI,LO + sat32(rt<15:0> × rs<63:48> × 2) // QMAC.03
```

The Q15 value in rt is multiplied by one of the Q15 values from the quad halfword rs, and the product is left-shifted by one bit position to generate a Q31 fractional format intermediate result. If both multiplicands are equal to –1.0 in Q15 fractional format (0x8000), the intermediate result is saturated to the maximum positive Q31 fractional value (0x7FFFFFFF). This multiply result is then added with a 64-bit value in the lower 32 bits of HI and LO.

No arithmetic exception occurs under any circumstances.

Restrictions:

The HI and LO contents are unpredictable after the QMAC instruction execution if rt is not a sign-extended 16-bit value or if LO is not a sign-extended 32-bit value prior to execution of the QMAC instruction. The HI and LO contents are also unpredictable after QMAC.02 or QMAC.03 instruction execution if 64-bit operations are not enabled.

HI is unpredictable after QMAC instruction if HI is not a sign-extended 32-bit value prior to execution of the QMAC instruction.

Cavium Confidential For Bowen Yang ALI 02/27/2014

Operation:

```
case QMAC.00: temprs<15:0> = GPR[rs]<15:0>
case QMAC.01: temprs<15:0> = GPR[rs]<31:16>
case QMAC.02: temprs<15:0> = GPR[rs]<47:32>
case QMAC.03: temprs<15:0> = GPR[rs]<63:48>

tempmul<31:0> = mulQ15Q15noovr(GPR[rt]<15:0>, temprs<15:0>)
tempres<63:0> = (HI<31:0> || LO<31:0>) + sign_extend(tempmul<31:0>)
LO<63:0> = sign_extend(tempres<31:0>)
HI<63:0> = sign_extend(tempres<63:32>)

function mulQ15Q15noovr(a<15:0>, b<15:0>)
  if ((a<15:0> == 0x8000) and (b<15:0> == 0x8000)) then
    temp<31:0> ← 0x7FFFFFFF
  else
    temp<31:0> ← (a<15:0> * b<15:0>) << 1
  endif
  return temp<31:0>
endfunction mulQ15Q15
```

Exceptions:

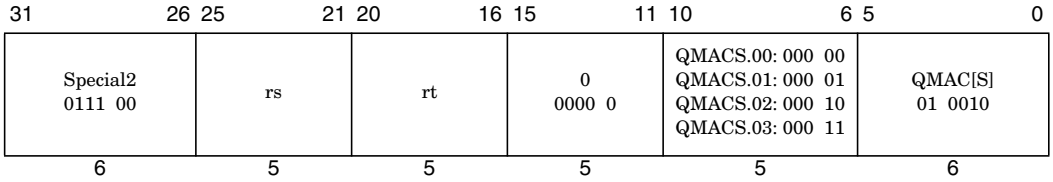
None.

Programming Notes:

The EXTS/EXTS32 instructions may be useful to generate the rt value from a 64-bit quad halfword that contains four Q15s.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Q15 Multiply and Accumulate With Saturation — QMACS.0x



Format: QMACS.00 rs, rt
QMACS.01 rs, rt
QMACS.02 rs, rt
QMACS.03 rs, rt **CVMv2**

Purpose:

Q15 multiply and accumulate in LO with saturation and overflow indication.

To multiply one Q15 value from a quad halfword with another Q15 value and accumulate the result, with saturation on both the multiply and add, and overflow indication.

Description:

```
LO ← sat32(LO + sat32(rt<15:0> × rs<15:0> × 2)) ; HI ← overflow // QMACS.00
LO ← sat32(LO + sat32(rt<15:0> × rs<31:16> × 2)); HI ← overflow // QMACS.01
LO ← sat32(LO + sat32(rt<15:0> × rs<47:32> × 2)); HI ← overflow // QMACS.02
LO ← sat32(LO + sat32(rt<15:0> × rs<63:48> × 2)); HI ← overflow // QMACS.03
```

The Q15 value in rt is multiplied by one of the Q15 values from the quad halfword rs, and the product is left-shifted by one bit position to generate a Q31 fractional format intermediate result. If both multiplicands are equal to –1.0 in Q15 fractional format (0x8000), overflow occurs and the intermediate result is saturated to the maximum positive Q31 fractional value (0x7FFFFFFF). This multiply result is then added with the 32-bit signed value in LO. If this final result cannot be represented by a 32-bit signed value, overflow occurs and the LO result is saturated to either the maximum positive 32-bit value (0x7FFFFFFF) or the minimum negative 32-bit value (0x80000000).

HI<0> is set if overflow occurs on either the multiply or the accumulate, and can accumulate (i.e. is sticky) to indicate overflow in any one or more of any number of QMACS instruction executions.

No arithmetic exception occurs under any circumstances.

Restrictions:

The HI and LO contents are unpredictable after the QMACS instruction execution if rt is not a sign-extended 16-bit value or if LO is not a sign-extended 32-bit value prior to execution of the QMACS instruction. The HI and LO contents are also unpredictable after the QMACS instruction if a MADD, MADDU, MSUB, MSUBU, DDIV, DIV, DDIVU, DIVU, QMAC, MULT, MULTU, DMULT, or DMULTU instruction immediately precedes the QMACS instruction without intervening MT HI and MT LO instructions. The HI and LO contents are also unpredictable after QMACS.02 or QMACS.03 instruction execution if 64-bit operations are not enabled.

Cavium Confidential For Bowie Yang Audit 07/27/2014

HI is unpredictable after QMACS instruction if HI<63:1> are not zero prior to execution of the QMACS instruction, or if the QMACS instruction is followed by a QMAC, MADD, MADDU, MSUB, or MSUBU instruction without an intervening MT/MF HI/LO instruction.

Operation:

```
case QMACS.00: temprs<15:0> = GPR[rs]<15:0>
case QMACS.01: temprs<15:0> = GPR[rs]<31:16>
case QMACS.02: temprs<15:0> = GPR[rs]<47:32>
case QMACS.03: temprs<15:0> = GPR[rs]<63:48>

tempmul<31:0> = mulQ15Q15(GPR[rt]<15:0>, temprs<15:0>)
LO<63:0> = sign_extend(sat32AccQ31(tempmul<31:0>))

function mulQ15Q15(a<15:0>, b<15:0>)
  if ((a<15:0> == 0x8000) and (b<15:0> == 0x8000)) then
    temp<31:0> ← 0x7FFFFFFF
    HI<0> ← 1  // indicate overflow
  else
    temp<31:0> ← (a<15:0> × b<15:0>) << 1
    HI<0> ← HI<0> // HI<0> is sticky
  endif
  return temp<31:0>
endfunction mulQ15Q15

function sat32AccQ31( a<31:0> )
  temp<32:0> ← LO<31> || LO<31:0>
  temp<32:0> ← temp + (a<31> || a<31:0>)
  if (temp<32> != temp<31>) then
    if (temp<31> == 0) then
       temp<31:0> ← 0x80000000
    else
       temp<31:0> ← 0x7FFFFFFF
    endif
    HI<0> ← 1 // indicate overflow
  else
    HI<0> ← HI<0>// HI<0> is sticky
  endif
  return temp<31:0>
endfunction sat32AccQ31
```

Exceptions:

None.

Programming Notes:

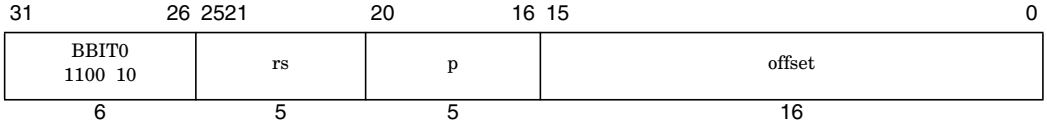
The EXTS/EXTS32 instructions may be useful to generate the rt value from a 64-bit quad halfword that contains four Q15s.

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.2 CPU Branch and Jump Instructions

A.2.1 Bit-test Branches Instructions

Branch on Bit Clear **BBIT0**



Format: BBIT0 rs, p, offset **CVM**

Purpose:

To do a PC-relative conditional branch if one of the lower 32-bits is clear.

Description: if !rs<p> then branch

An 18-bit signed offset (the 16-bit offset field shifted left 2-bits) is added to the address of the instruction following the branch (not the branch itself), in the branch delay slot, to form a PC-relative effective target address.

If bit rs<p> is clear, branch to the effective target address after the instruction in the delay slot is executed.

Restrictions:

Processor operation is unpredictable if a branch, jump, ERET, DERET, WAIT, or PAUSE instruction is placed in the delay slot of a branch or jump.

Operation:

```
I:    target_offset ← sign_extend(offset || 00)
      condition ← !GPR[rs]<p>
I+1:  if condition then
        PC ← PC + target_offset
        endif
```

Exceptions:

None.

Programming Notes:

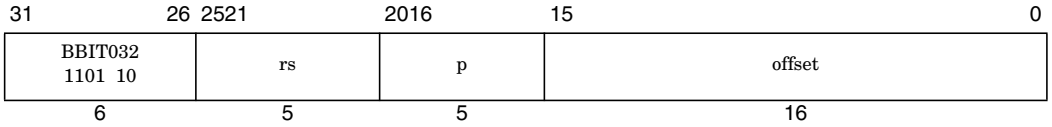
With the 18-bit signed instruction offset, the conditional branch range is ± 128 KB. Use jump (J) or jump register (JR) instructions to branch to addresses outside this range.

BBIT0 consumes the major opcode of the MIPS LWC2 instruction. BBIT0 is not considered a CO

instruction so does not take an exception when COP2 is not enabled.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Branch on Bit Clear Plus 32 — BBIT032



Format: BBIT032 rs, p, offset **CVM**

Purpose:

To do a PC-relative conditional branch if one of the upper 32 bits is clear.

Description: if !rs<p+32> then branch

An 18-bit signed offset (the 16-bit offset field shifted left 2-bits) is added to the address of the instruction following the branch (not the branch itself), in the branch delay slot, to form a PC-relative effective target address.

If bit rs<p+32> is clear, branch to the effective target address after the instruction in the delay slot is executed.

Restrictions:

Processor operation is unpredictable if a branch, jump, ERET, DERET, WAIT, or PAUSE instruction is placed in the delay slot of a branch or jump.

Processor operation is unpredictable if 64-bit operations are not enabled during execution of a BBIT032 instruction.

Operation:

```
I:   target_offset ← sign_extend(offset || 00)
     condition ← !GPR[rs]<p+32>
I+1: if condition then
       PC ← PC + target_offset
     endif
```

Exceptions:

None.

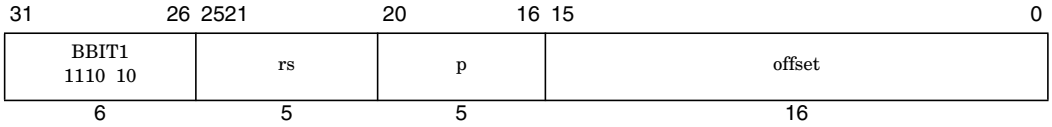
Programming Notes:

With the 18-bit signed instruction offset, the conditional branch range is ± 128 KB. Use jump (J) or jump register (JR) instructions to branch to addresses outside this range.

BBIT032 consumes the major opcode of the MIPS LDC2 instruction. BBIT032 is not considered a COP2 instruction so does not take an exception when COP2 is not enabled.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Branch on Bit Set — BBIT1



Format: BBIT1 rs, p, offset **CVM**

Purpose:

To do a PC-relative conditional branch if one of the lower 32 bits is set.

Description: if rs<p> then branch

An 18-bit signed offset (the 16-bit offset field shifted left 2-bits) is added to the address of the instruction following the branch (not the branch itself), in the branch delay slot, to form a PC-relative effective target address.

If-bit rs<p> is set, branch to the effective target address after the instruction in the delay slot is executed.

Restrictions:

Processor operation is unpredictable if a branch, jump, ERET, DERET, WAIT, or PAUSE instruction is placed in the delay slot of a branch or jump.

Operation:

```
I:    target_offset ← sign_extend(offset || 00)
      condition ← GPR[rs]<p>
I+1:  if condition then
        PC ← PC + target_offset
        endif
```

Exceptions:

None.

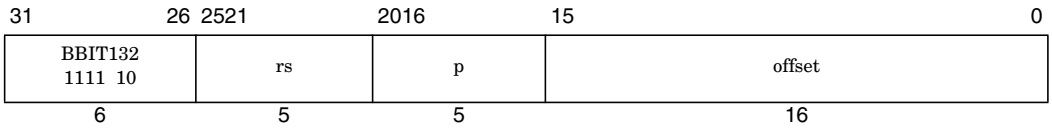
Programming Notes:

With the 18-bit signed instruction offset, the conditional branch range is ± 128 KB. Use jump (J) or jump register (JR) instructions to branch to addresses outside this range.

BBIT1 consumes the major opcode of the MIPS SWC2 instruction. BBIT1 is not considered a COP2 instruction so does not take an exception when COP2 is not enabled.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Branch on Bit Set Plus 32 — BBIT132



Format: BBIT132 rs, p, offset **CVM**

Purpose:

To do a PC-relative conditional branch if one of the upper 32 bits is set.

Description: if rs<p+32> then branch

An 18-bit signed offset (the 16-bit offset field shifted left 2-bits) is added to the address of the instruction following the branch (not the branch itself), in the branch delay slot, to form a PC-relative effective target address.

If bit rs<p+32> is set, branch to the effective target address after the instruction in the delay slot is executed.

Restrictions:

Processor operation is unpredictable if a branch, jump, ERET, DERET, WAIT, or PAUSE instruction is placed in the delay slot of a branch or jump.

Processor operation is unpredictable if 64-bit operations are not enabled during execution of a BBIT132 instruction.

Operation:

```
I:   target_offset ← sign_extend(offset || 00)
     condition ← GPR[rs]<p+32>
I+1: if condition then
       PC ← PC + target_offset
     endif
```

Exceptions:

None.

Programming Notes:

With the 18-bit signed instruction offset, the conditional branch range is ± 128 KB. Use jump (J) or jump register (JR) instructions to branch to addresses outside this range.

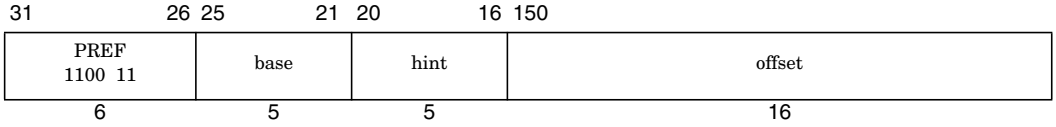
BBIT132 consumes the major opcode of the MIPS SDC2 instruction. BBIT132 is not considered a COP2 instruction so does not take an exception when COP2 is not enabled.

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.3 CPU Load, Store, Memory, and Control Instructions

A.3.1 Prefetch Instructions

Prefetch **PREF**



Format: PREF hint, offset(base) **MIPS32/CVM/CVMv2/CVMv3**

Purpose:

To move data between memory and cache/TLB.

Description: prefetch_memory(GPR[base]+offset)

PREF adds the 16-bit signed offset to the contents of GPR base to form an effective byte address. PREF operates on the surrounding naturally-aligned cache block, typically prefetching it into a cache. The cache block size is 128 bytes on the CN71XX . With the exception of the Prepare-For-Store and Don't-Write-Back hints, PREF instruction execution does not change architectural state. The hint field selects the particular operation.

PREF does not cause addressing-related exceptions. If the address specified would cause an addressing exception, the exception condition is ignored and no data movement occurs.

PREF never generates a memory operation for I/O memory locations, DSEG locations, and CVMSEG locations.

Possible values of the hint field for the PREF Instruction are listed in Table 42–2.

NOTE: This is a standard MIPS32 instruction for which the CN71XX implements all the functionality defined in the MIPS standard instruction set, as well as additional, cnMIPS III-specific operations. The hint field is cnMIPS III-specific.

Table 42–2 Hint-Field Values

Hint Value	Name	Data Use and Desired Prefetch Action
0–1, 6[1], 7	NORMAL	The block will be prefetched into the L1 cache and the L2 cache. PREF 0–1, 6, 7 generates an LDD CMI transaction – refer to Section 3.1.
2–3, 8–23	Reserved	Results are unpredictable.
4[1]	NOTL2	The block will be prefetched into the L1 cache without putting it into the L2 cache. PREF 4 generates a PSL1 CMI transaction – refer to Section 3.1.
5, 28[2]	L2	The block will be prefetched into the L2 cache without putting it into the L1 cache. PREF 5, 28 generate a PL2 CMI transaction – refer to Section 3.1.
24[1]	μTLB	The translation corresponding to the virtual address specified in the instruction will be prefetched from the TLB into the μTLB. The operation is a NOP if the address is not in the TLB.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Table 42–2 Hint-Field Values

Hint Value	Name	Data Use and Desired Prefetch Action
25[2]	NUDGE/WRITEBACK_INVALIDATE	Clear the used bit in the L2 cache and, if the block is dirty, schedule an immediate write-back of the block (clearing the dirty bit). The block is not invalidated in the L2 cache but will be evicted soon. PREF 25 generates a WBL2 CMI transaction – refer to Section 3.1. **NOTE:** The operation of PREF 25 is similar to the operation of CACHE 27.
26[2]	NUDGE_WB	If the cache block is in the L1 write buffer, send it out now to the L2 cache.
27[2]	EVICT_SOON	Clear the used bit in the L2 cache so that the block gets evicted soon. PREF 27 generates a RPL2 CMI transaction – refer to Section 3.1.
29[2]	DONT_WRITE_BACK	The block will not be written back to memory (i.e. the cache dirty bit is cleared). The value of the bytes in the cache block are unpredictable and may change value unpredictably until they are later stored to. The DONT_WRITE_BACK operation can be used to avoid unnecessary write backs from the L2 cache (to DRAM) for memory locations that are not being used. TLB treats this as a store, as it can change state. PREF 29 generates a DWB CMI transaction – refer to Section 3.1.
30[2]	PREPARE_FOR_STORE	A write buffer entry is created. If the write buffer entry misses in the L2 cache, the CN71XX will not read the prior block from DRAM into the L2 cache. The value of the bytes in the cache block are unpredictable and may change value unpredictably until they are later stored to. The PREPARE_FOR_STORE operations can be used to avoid unnecessary DRAM read operations for memory locations whose current value does not matter. TLB treats this as a store, as it can change state. PREF 30 generates a STF CMI transaction – refer to Section 3.1.
31[2]	PREPARE_FOR_STORE_THROUGH	Same as PREPARE_FOR_STORE, except that the block write operation will not allocate the block in L2C when it is not already present in L2C. TLB treats this as a store, as it can change state. PREF 31 generates a STT CMI transaction – refer to Section 3.1.

1. When Root.CvmMemCtl[DISLDPREF] = 1, Hint Values 0, 1, 4, 6, 7, 24 are NOPs.
2. When Root.CvmMemCtl[DISSTPREF] = 1, Hint Values 5, 25–31 are NOPs.

Restrictions:

None.

Operation:

```
vAddr , GPR[base] + sign_extend(offset)
(pAddr, CCA) , AddressTranslation(vAddr, DATA, LOAD)
Prefetch(CCA, pAddr, vAddr, DATA, hint)
```

Exceptions:

Cache Error, Watch, Breakpoint

Prefetch does not take any TLB-related or address-related exceptions under any circumstances.

CN71XX Core watchpoints and (EJTAG) breakpoints match against the PREF 29–31 instructions as if they were stores to every byte in the surrounding cache block. These PREFs always match a value compare.

None of the PREF 0–28 instruction ever match against watchpoints/breakpoints.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Programming Notes:

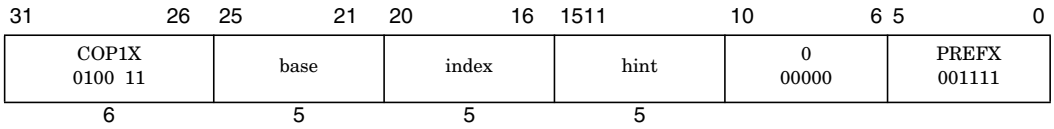
Prefetch cannot prefetch data from a mapped location unless the translation for that location is present in the TLB. Locations in memory pages that have not been accessed recently may not have translations in the TLB, so prefetch may not be effective for such locations.

Prefetch will act as a NOP if it misses in the TLB. When CvmMemCtl2[UTLBRW] = 0, a Prefetch will always replay if it misses in the μTLB. When CvmMemCtl2[UTLBRW] = 1, a Prefetch will become a NOP (except for hint 24) if it misses in the μTLB.

Prefetch does not cause addressing exceptions, so a legal PREF instruction can have any effective address.In many circumstances, ZCB/ZCBT instructions should be used rather than PREF 30/31. PREF 30/31 have offset addressing and will leave the block in the L1 cache after the instruction if it were already present in the cache.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Prefetch Indexed — PREFX



Format: PREFX hint, index(base) **CVMv3**

Purpose:

To move data between memory and cache/TLB.

Description: prefetch_memory(GPR[base]+GPR[index])

PREFX adds contents of GPR index to GPR base to form an effective byte address. PREFX operates on the surrounding naturally-aligned cache block, typically prefetching it into a cache. The cache block size is 128 bytes on the CN71XX. With the exception of the Prepare-For-Store and Don't-Write-Back hints, PREFX instruction execution does not change architectural state. The hint field selects the particular operation.

PREFX does not cause addressing-related exceptions. If the address specified would cause an addressing exception, the exception condition is ignored and no data movement occurs.

PREFX never generates a memory operation for I/O memory locations, DSEG locations, and CVMSEG locations.

Possible values of the hint field for the PREFX Instruction are same as PREF instruction and are listed in Table 42–2.

Restrictions:

If access to Coprocessor 1 is not enabled, a Coprocessor Unusable Exception is signaled.

Operation:

```
vAddr , GPR[base] + sign_extend(offset)
(pAddr, CCA) , AddressTranslation(vAddr, DATA, LOAD)
Prefetch(CCA, pAddr, vAddr, DATA, hint)
```

Exceptions:

Coprocessor Unusable Exception

Cache Error, Watch, Breakpoint

Prefetch Indexed does not take any TLB-related or address-related exceptions under any circumstances.

CN71XX Core watchpoints and (EJTAG) breakpoints match against the PREF 29–31 instructions as if they were stores to every byte in the surrounding cache block. These PREFXs always match a value compare.

None of the PREFX 0–28 instruction ever match against watchpoints/breakpoints.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Programming Notes:

Prefetch Indexed cannot prefetch data from a mapped location unless the translation for that location is present in the TLB. Locations in memory pages that have not been accessed recently may not have translations in the TLB, so prefetch indexed may not be effective for such locations.

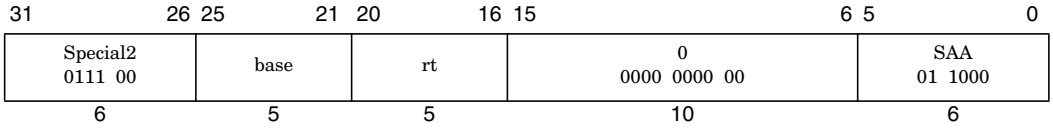
Prefetch Indexed does not cause addressing exceptions, so a legal PREFX instruction can have any effective address.

In many circumstances, ZCB/ZCBT instructions should be used rather than PREFX 30/31. PREFX 30/31 have offset addressing and will leave the block in the L1 cache after the instruction if it were already present in the cache.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

A.3.2 Store Atomic Add Instructions

Store Atomic Add Word **SAA**



Format: SAA rt, (base) **CVM**

Purpose:

To atomically add a word to a memory location.

Description: memory[base] = memory[base] + rt //atomic

The least-significant 32-bit word of register rt is added to the memory at the aligned address specified by the register base.

The memory read, add, and memory store are not interrupted by any other instructions on this or any other processor. No Integer Overflow exception occurs under any circumstance.

TLB accesses, watches, and breakpoints treat SAA instructions as a software instruction, except that a breakpoint value compare always matches regardless of the DBV and memory/add byte values.

SAA generates a SAA32 CMI transaction – refer to Section 3.1.

Restrictions:

The address in the base register must be naturally-aligned, else an address error occurs.

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results.

Operation:

```
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
      unpredictable
else if (vAddr/pAddr has an address error) then
      SignalException(AddressError)
endif
amsk2 = 0^61 || CvmCtl[LE] || 0^2
memdoubleword ← LoadMemory (CCA, WORD, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
bit ← 64 - ((vAddr<2> XOR CvmCtl[LE]) || 0^5)
resultdoubleword = memdoubleword + (GPR[rt]<63:0> << (bit-32))
StoreMemory (CCA, WORD, resultdoubleword, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint

Programming Notes:

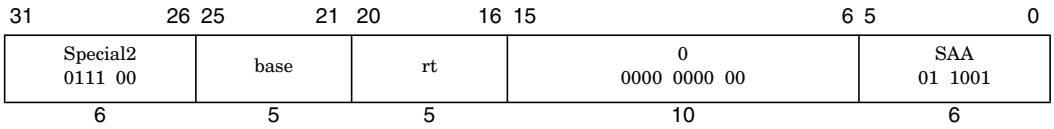
The SAA instruction is performance-optimized for memory locations that are present in L2/DRAM, not the level-one data cache. The cache block that contains the memory location is removed from the level-one data cache (if present) during execution of the SAA instruction.

SAA execution is very similar to LAA/LAI/LAD, except that the previous value of the memory location is not returned in a register. Different instructions can be used in different circumstances:

- SAA should always be used rather than LAA/LAI/LAD if the memory location need not be read near the operation.
- If the memory location needs to be read atomically as part of the operation, then LAA/LAI/LAD must be used rather than SAA.
- If the memory location needs to be read sometime near the operation, but not atomically with the update:
  - If the block is likely to be in the L1 cache before the operation, LW+SAA may perform better than LAA/LAI/LAD.
  - If the block is not likely to be in the L1 cache before the operation, either LAA/LAI/LAD or SAA+LW or SAA+PREF+LW sequences could perform best, depending on how useful it is to put the block into the L1 after the operation, and whether the PREF can hide the cache miss latency.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Store Atomic Add Doubleword — SAAD



Format: SAAD rt, (base) **CVM**

Purpose:

To atomically add a doubleword to a memory location.

Description: memory[base] = memory[base] + rt //atomic

The 64-bit register rt is added to the memory at the aligned address specified by register base.

The memory read, add, and memory store are not interrupted by any other instructions on this or any other processor. No Integer Overflow exception occurs under any circumstance.

TLB accesses, watches, and breakpoints treat SAAD instructions as a SD instruction, except that a breakpoint value compare always matches regardless of the DBV and memory/add byte values.

SAAD generates a SAA64 CMI transaction – refer to Section 3.1.

Restrictions:

The address in the base register must be naturally-aligned, else an address error occurs.

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results. A Reserved Instruction Exception is signalled if access to 64-bit operations is not enabled.

Operation:

```
if !Are64bitOperationsEnabled() then
    SignalException(ReservedInstruction)
endif
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
    unpredictable
else if (vAddr/pAddr has an address error) then
    SignalException(AddressError)
endif
memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr, vAddr, DATA)
resultdoubleword = memdoubleword + GPR[rt]
StoreMemory (CCA, WORD, resultdoubleword, pAddr, vAddr, DATA)
```

Exceptions:

Reserved Instruction, TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint

Cavium Confidential For Bowen Yang ALU 02/27/2014

Programming Notes:

The SAAD instruction is performance-optimized for memory locations that are present in L2/DRAM, not the level-one data cache. The cache block that contains the memory location is removed from the level-one data cache (if present) during execution of the SAAD instruction on the CN5XXX series.

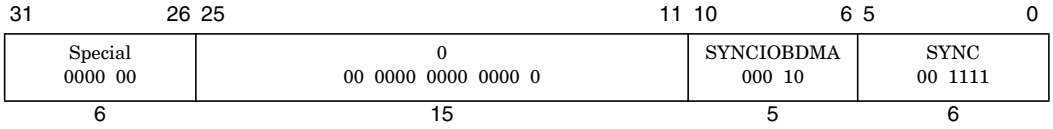
SAAD execution is very similar to LAAD/LAID/LADD, except that the previous value of the memory location is not returned in a register. Different instructions can be used in different circumstances:

- SAAD should always be used rather than LAAD/LAID/LADD if the memory location need not be read near the operation.
- If the memory location needs to be read atomically as part of the operation, then LAAD/LAID/LADD must be used rather than SAAD.
- If the memory location needs to be read sometime near the operation, but not atomically with the update:
  - If the block is likely to be in the L1 cache before the operation, LD+SAAD may perform better than LAAD/LAID/LADD.
  - If the block is not likely to be in the L1 cache before the operation, either LAAD/LAID/LADD or SAAD+LD or SAAD+PREF+LD sequences could perform best, depending on how useful it is to put the block into the L1 after the operation, and whether the PREF can hide the cache miss latency.

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.3.3 SYNC Instructions

Synchronize IOBDMAs **SYNCIOBDMA**



Format: SYNCIOBDMA **CVM**

Purpose:

To complete IOBDMA operations.

Description:

SYNCIOBDMA execution guarantees that there are no in-flight IOBDMA operations; a subsequent CVMSEG LM load operation will get results updated for all outstanding IOBDMA operations.

Restrictions:

None.

Operation:

```
SyncOperation(stype)
```

Exceptions:

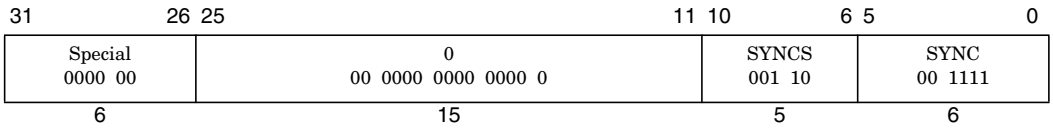
None.

Programming Notes:

IOBDMA operations (see 2.7 on page 84) can be initiated without waiting for their CVMSEG LM result. SYNCIOBDMA operations are necessary to complete them. (SYNC/SYNCS instructions also guarantee that in-flight IOBDMA operations complete, but are more costly than SYNCIOBDMA operations.)

Cavium Confidential For Bowen Yang ALU 03/19/2014

Synchronize Special SYNCS



Format: SYNCS **CVM**

Purpose:

To order unmarked L2/DRAM and I/O load/store operations and IOBDMA operations. SYNCS is identical to SYNC, except that SYNCS does not order marked L2/DRAM load/store operations.

Description:

- SYNCS affects the ordering of all unmarked L2/DRAM load/store operations, I/O load/store operations, and IOBDMA operations. The unmarked L2/DRAM load/store operations, I/O load/store operations, and IOBDMA operations that occur before the SYNCS are completed before any unmarked L2/DRAM load/store operations, I/O load/store operations, or IOBDMA operations after the SYNCS are allowed to start.
- SYNCS does not affect the order of marked L2/DRAM load/store operations. Marked L2/DRAM load/store operations are those L2/DRAM load/store operations whose cache coherency attribute equals 7.
- Loads are complete when the destination register is written.
- L2/DRAM stores are complete when the stored value is visible to every other core and all CN71XX IO units.
- I/O stores are posted on the CN71XX, and are “complete” when they reach an ordering point for the device.
- SYNCS execution guarantees that there are no in-flight IOBDMA operations; a subsequent CVMSEG LM load gets results updated for all outstanding IOBDMA operations.
- SYNCS does not guarantee the order in which instruction fetches are performed.
- SYNCS is identical to SYNC when CvmMemCtl[DISSYNCWS] is set.
- Refer to Section 2.8 for more discussion of ordering.

Restrictions:

None.

Operation:

```
SyncOperation(stype)
```

Exceptions:

None.

Cavium Confidential For Bowen Yang ALU 01/27/2014

Programming Notes:

On the CN71XX, SYNCW and SYNCWS are much faster than SYNC or SYNCS, so should be used when possible.

A processor executing load and store instructions observes the order in which loads and stores occur in the instruction stream; this is known as program order.

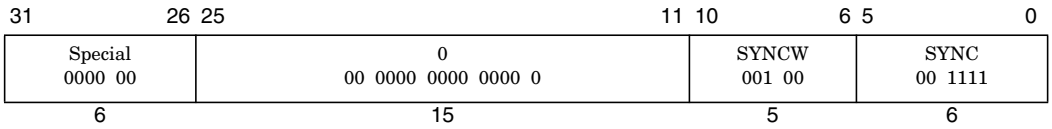
A parallel program has multiple instruction streams that can execute simultaneously on different processors. In the CN71XX, the order in which the effects of loads and stores are observed by other processors - the global order of the loads and stores - determines the actions necessary to reliably share data in parallel programs.

In the CN71XX, the effects of store instructions executed by one processor may be observed out of program order by other processors, so parallel programs must take explicit actions to reliably share data. At critical points in the program, the effects of stores from an instruction stream must occur in the same order for all processors. SYNCS separates the unmarked L2/DRAM and I/O load/store operations executed on the processor into two groups, and the effect of all references in one group is seen by all processors before the effect of any reference in the subsequent group.

IOBDMA operations (see 2.7 on page 84) can be initiated without waiting for their CVMSEG LM result. SYNCS operations complete them. (SYNCIOBDMA instructions also guarantee that in-flight IOBDMA operations complete, and are less costly than SYNC/SYNCS operations.)

Cavium Confidential For Bowen Yang ALU 02/27/2014

Synchronize Stores — SYNCW



Format: SYNCW **CVM**

Purpose:

To order stores. SYNCW is similar to SYNCWS, but SYNCW orders more stores.

Description:

- SYNCW affects only stores. All the stores that occur before the SYNCW are completed before any of the stores after the SYNC are allowed to start.
- L2/DRAM stores are complete when the stored value is visible to every other core and all CN71XX units.
- I/O stores are posted on the CN71XX, and are “complete” when they reach an ordering point for the device.
- SYNCW does not affect any ordering between load/IOBDMA operations and stores. Furthermore, SYNCW operations are queued and complete out of order with respect to load/IOBDMA operations.
- Refer to Section 2.8 for more discussion of ordering.

Restrictions:

None.

Operation:

```
SyncOperation(stype)
```

Exceptions:

None.

Programming Notes:

On the CN71XX, SYNCW and SYNCWS are much faster than SYNC or SYNCS, so should be used whenever possible. SYNCWS can produce substantially better system performance than SYNCW, as it orders fewer stores.

A processor executing load and store instructions observes the order in which loads and stores occur in the instruction stream; this is known as program order.

A parallel program has multiple instruction streams that can execute simultaneously on different processors. In the CN71XX, the order in which the effects of loads and stores are observed by other processors - the global order of the loads and stores - determines the actions necessary to reliably share data in parallel programs.

Cavium Confidential For Bowen Yang ALU 02/27/2014

In the CN71XX, the effects of store instructions executed by one processor may be observed out of program order by other processors, so parallel programs must take explicit actions to reliably share data. At critical points in the program, the effects of stores from an instruction stream must occur in the same order for all processors. SYNCW separates the load operations and store operations executed on the processor into two groups, and the effect of all stores in one group is seen by all processors before the effect of any store operation in the subsequent group. In effect, SYNCW causes the CN71XX to be strongly ordered for the Core at the instant that the SYNCW is executed.

Conditions at entry: The value 0 has been stored in FLAG and that value is observable by core B

Core A (writer)

```
SW R1, DATA#            change shared DATA value
LI R2, 1
SYNCW# (or SYNCWS) Perform DATA store before performing FLAG store
SW R2, FLAG#            say that the shared DATA value is valid
SYNCW# (or SYNCWS) Force the FLAG store soon (CN71XX-specific)
```

Core B (reader)

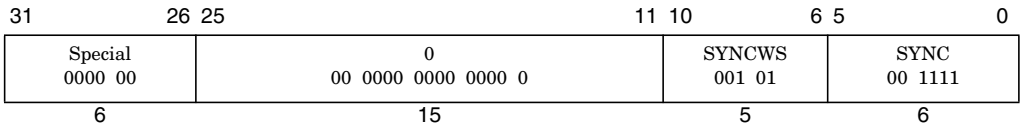
```
  LI R2, 1
1: LW R1, FLAG    # Get FLAG
  BNE R2, R1, 1B  # if it says that DATA is not valid, poll again
  NOP
  LW R1, DATA     # Read (valid) shared DATA value
```

The code fragments above show how SYNCW can be used to coordinate the use of shared data between separate writer and reader instruction streams in the CN71XX. The FLAG location is used by the instruction streams to determine whether the shared data item DATA is valid. The first SYNCW instruction executed by core A forces the store of DATA to be performed globally before the store to FLAG is performed. This is necessary for correct behavior on the CN71XX.

The second SYNCW instruction executed by core A is not necessary for correctness, but has very important performance effects on the CN71XX. Without it, the store to FLAG may linger in core A's write buffer before it becomes visible to other cores. (If core A is not performing many stores, this may add hundreds of thousands of cycles to the flag release time since the CN71XX core nominally retains stores to attempt to merge them before sending the store on the CMI.) Applications should include this second SYNCW instruction after flag or lock releases.

Cavium Confidential For Bowen Yang 02/12/2014

Synchronize Stores Special — SYNCWS



Format: SYNCWS **CVM**

Purpose:

To order unmarked L2/DRAM and I/O stores. SYNCWS is identical to SYNCW, except that SYNCWS does not order marked L2/DRAM stores.

Description:

- SYNCWS affects the ordering of unmarked L2/DRAM store operations and all I/O store operations. The unmarked L2/DRAM and I/O store operations that occur before the SYNCWS are completed before the unmarked L2/DRAM and I/O store operations after the SYNC are allowed to start.
- SYNCWS does not affect the order of marked L2/DRAM store operations. Marked L2/DRAM store operations are those L2/DRAM store operations whose cache coherency attribute equals 7.
- L2/DRAM store operations are completed when the stored value is visible to every other core and all CN71XX I/O units.
- I/O store operations are posted on the CN71XX, and are "complete" when they reach the coherent memory interconnect.
- SYNCWS does not affect any ordering between load operations and store operations, nor between IOBDMA operations and store operations. Furthermore, SYNCWS operations may execute out of order with respect to load operations and IOBDMA operations.
- SYNCWS is identical to SYNCW when CvmMemCtl[DISSYNCWS] is set.
- Refer to the cnMIPS III Cores chapter ("cnMIPS III Cores" on page 57) for more discussion of ordering.

Restrictions:

None.

Operation:

```
SyncOperation(stype)
```

Exceptions:

None.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Programming Notes:

On the CN71XX, SYNCW and SYNCWS are much faster than SYNC or SYNCS, so should be used whenever possible. SYNCWS can produce substantially better system performance than SYNCW, as it orders fewer stores.

A processor executing load and store instructions observes the order in which loads and stores occur in the instruction stream; this is known as program order.

A parallel program has multiple instruction streams that can execute simultaneously on different processors. In the CN71XX, the order in which the effects of loads and stores are observed by other processors - the global order of the loads and stores - determines the actions necessary to reliably share data in parallel programs.

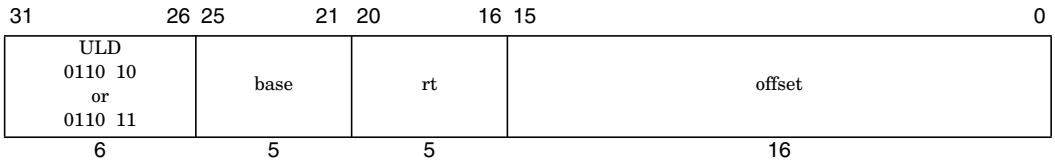
In the CN71XX, the effects of store instructions executed by one processor may be observed out of program order by other processors, so parallel programs must take explicit actions to reliably share data. At critical points in the program, the effects of stores from an instruction stream must occur in the same order for all processors. SYNCWS separates the unmarked memory and I/O stores executed on the processor into two groups, and the effect of all store operations in one group is seen by all processors before the effect of any store in the subsequent group.

Refer to the SYNCW Programming Notes: on page 1678 for a code fragment of a writer/reader application and optimization on the CN71XX.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

A.3.4 Unaligned Load/Store Instructions

Unaligned Load Doubleword **ULD**



Format: ULD rt, offset(base) **CVM**

The ULD instruction does not exist when CvmCtl[USEUN] is clear. (The MIPS LDL and LDR instructions do not exist when CvmCtl[USEUN] is set.)

The ULD instruction (when enabled by CvmCtl[USEUN]) consumes either the MIPS LDL or LDR major opcodes according to the following table:

CvmCtl[USELY]	BigEndianCPU (CvmCtl[$\overline{LE}$])	MIPS LDL opcode is:	MIPS LDR opcode is:
0	0	NOP	ULD
0	1	ULD	NOP
1	X	ULD	NOP

When CvmCtl[USEUN] is set, one of MIPS LDL and LDR major opcodes execute as NOPs, as specified in the table above.

Purpose:

To load a doubleword from a (potentially-unaligned) memory location.

Description: rt = memory[base+offset]

The contents of the 64-bit doubleword at the memory location specified by the effective address are fetched and placed in GPR rt. The 16-bit signed offset is added to the contents of GPR base to form the effective address.

Restrictions:

Whenever any of the bytes required to service the load reside outside of normal/cacheable memory and CVMSEG LM, the effective address must be naturally-aligned, else an address error exception occurs. Note that these alignment address error exceptions are lower priority than TLB refill and invalid exceptions.

CVMSEG I/O references cause address errors. CVMSEG LM references by these load operations cause an address error when outside the legal range allowed by CvmMemCtl[LMEMSZ] (assuming CVMSEG LM is enabled by CvmMemCtl[CVMSEGENA*]).

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Operation:

```
if !Are64bitOperationsEnabled() then
    SignalException(ReservedInstruction)
endif
vAddr ← sign_extend(offset) + GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr has address error) then
    SignalException(AddressError)
endif
amsk = 1^61 | 0^3
memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr AND amsk, vAddr AND amsk, DATA)
if(vAddr<2:0> >= 1) then
    vAddr2 ← vAddr + 8
    (pAddr2, CCA2) ← AddressTranslation (vAddr2, DATA, LOAD)
    if (vAddr2/pAddr2 has address error)
        SignalException(AddressError) // vAddr+8 appears in BadVAddr in this case
    endif
    // either vAddr2 is CVMSEG LM or paddr2 must be L2/DRAM
    memdoubleword2 ← LoadMemory (CCA2, DOUBLEWORD, pAddr2 AND amsk, vAddr2 AND amsk, DATA)
endif
if CvmCtl[LE] then
    memquadword<127..0> = memdoubleword2<63..0> || memdoubleword<63..0>
    bit ← (vAddr<2..0> || 0^3)
    GPR[rd] ← sign_extend(memquadword<bit+63 .. bit>)
else
    memquadword<127..0> = memdoubleword<63..0> || memdoubleword2<63..0>
    bit ← 128 - (vAddr<2..0> || 0^3)
    GPR[rd] ← sign_extend(memquadword<bit-1 .. bit-64>)
endif
```

Exceptions:

TLB Refill, TLB Invalid, Bus Error, Address Error, Reserved Instruction, Watch, Breakpoint

Notes:

CN71XX cores execute naturally-aligned LD/LDXs one cycle faster than naturally-aligned ULDs, so the LD/LDX instructions should be used rather than ULD when an address is known to be naturally-aligned.

ULD is an assembler macro that converts to MIPS LDL/LDR sequences on most MIPS assemblers.

Cavium Confidential For Bowen Yang ALU 02/27/2014

The following table indicates the byte lanes for 64-bit references in big-endian mode:

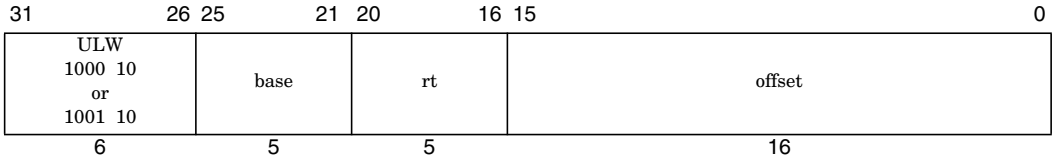
VA<2:0>	Register File Byte Positions in the two memory doublewords							
0	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>
	X	X	X	X	X	X	X	X
1	X	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>
	< 7: 0>		X	X	X	X	X	X
2	X	X	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>
	<15: 8>	< 7: 0>		X	X	X	X	X
3	X	X	X	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>
	<23:16>	<15: 8>	< 7: 0>		X	X	X	X
4	X	X	X	X	<63:56>	<55:48>	<47:40>	<39:32>
	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X	X
5	X	X	X	X	X	<63:56>	<55:48>	<47:40>
	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X
6	X	X	X	X	X	X	<63:56>	<55:48>
	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X
7	X	X	X	X	X	X	X	<63:56>
	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>	X

The following table indicates the byte lanes for 64-bit references in little-endian mode:

VA<2:0>	Register File Byte Positions in the two memory doublewords							
0	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>
	X	X	X	X	X	X	X	X
1	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>	X
	X	X	X	X	X	X	X	<63:56>
2	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X
	X	X	X	X	X	X	<63:56>	<55:48>
3	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X
	X	X	X	X	X	<63:56>	<55:48>	<47:40>
4	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X	X
	X	X	X	X	<63:56>	<55:48>	<47:40>	<39:32>
5	<23:16>	<15: 8>	< 7: 0>	X	X	X	X	X
	X	X	X	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>
6	<15: 8>	< 7: 0>	X	X	X	X	X	X
	X	X	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>
7	< 7: 0>	X	X	X	X	X	X	X
	X	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>

Cavium Confidential For Bowen Yang ALU 02/27/2014

Unaligned Load Word ULW



Format: ULW rt, offset(base) **CVM**

The ULW instruction does not exist when CvmCtl[USEUN] is clear. (The MIPS LWL and LWR instructions do not exist when CvmCtl[USEUN] is set.)

The ULW instruction (when enabled by CvmCtl[USEUN]) consumes either the MIPS LWL or LWR major opcodes according to the following table:

CvmCtl[USELY]	BigEndianCPU (CvmCtl[$\overline{LE}$])	MIPS LWL opcode is:	MIPS LWR opcode is:
0	0	NOP	ULW
0	1	ULW	NOP
1	X	ULW	NOP

When CvmCtl[USEUN] is set, one of MIPS LWL and LWR major opcodes execute as NOPs, as specified in the table above.

Purpose:

To load a word from a (potentially-unaligned) memory location

Description: rt = memory[base+offset]

The contents of the 32-bit word at the memory location specified by the effective address are fetched, sign-extended to 64-bits, and placed in GPR rt. The 16-bit signed offset is added to the contents of GPR base to form the effective address.

Restrictions:

Whenever any of the bytes required to service the load reside outside of normal/ cacheable memory and CVMSEG LM, the effective address must be naturally aligned, else an address error exception occurs. Note that these alignment address error exceptions are lower priority than TLB refill and invalid exceptions.

CVMSEG IO references cause address errors. CVMSEG LM references by these loads cause an address error when outside the legal range allowed by CvmMemCtl[LMEMSZ] (assuming CVMSEG LM is enabled by CvmMemCtl[CVMSEGENA*]).

Cavium Confidential For Power Yang ALD 02/7/2014

Operation:

```
if !Are64bitOperationsEnabled() then
    SignalException(ReservedInstruction)
endif
vAddr ← sign_extend(offset) + GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr has address error) then
    SignalException(AddressError)
endif
amsk = 1^61 | 0^3
if (vAddr<1:0> == 0) then
    amsk2 = 0^61 || CvmCtl[LE] || 0^2
    memdoubleword ← LoadMemory (CCA, WORD, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
else
    memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr AND amsk, vAddr AND amsk, DATA)
endif
if (vAddr<2:0> >= 5) then
    vAddr2 ← vAddr + 8
    (pAddr2, CCA2) ← AddressTranslation (vAddr2, DATA, LOAD)
    if (vAddr2/pAddr2 has address error)
        SignalException(AddressError) // vAddr+8 appears in BadVAddr in this case
    endif
    // either vAddr2 is CVMSEG LM or paddr2 must be L2/DRAM
    memdoubleword2 ← LoadMemory (CCA2, DOUBLEWORD, pAddr2 AND amsk, vAddr2 AND amsk, DATA)
endif
if CvmCtl[LE] then
    memquadword<127..0> = memdoubleword2<63..0> || memdoubleword<63..0>
    bit ← (vAddr<2..0> || 0^3)
    GPR[rd] ← sign_extend(memquadword<bit+31 .. bit>)
else
    memquadword<127..0> = memdoubleword<63..0> || memdoubleword2<63..0>
    bit ← 128 - (vAddr<2..0> || 0^3)
    GPR[rd] ← sign_extend(memquadword<bit-1 .. bit-32>)
endif
```

Exceptions:

TLB Refill, TLB Invalid, Bus Error, Address Error, Watch, Breakpoint

Notes:

CN71XX cores execute naturally-aligned LW/LWS instructions one cycle faster than naturally-aligned ULW instructions, so the LW/LWS instructions should be used rather than ULW when an address is known to be naturally-aligned.

ULW is an assembler macro that converts to MIPS LWL/LWR sequences on most MIPS assemblers.

Cavium Confidential For Bowen Yang ALU 9/2/2014

The following table indicates the byte lanes for 32-bit references in big-endian mode:

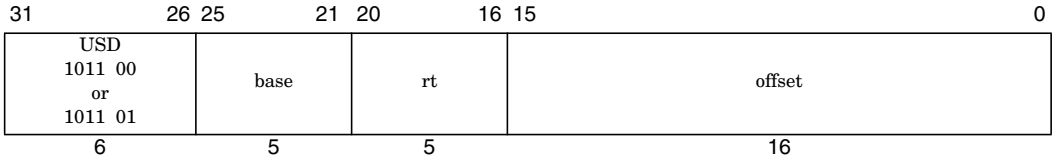
VA<2:0>	Register File Byte Positions in the two memory doublewords							
0	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X	X
	X	X	X	X	X	X	X	X
1	X	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X
	X	X	X	X	X	X	X	X
2	X	X	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X
	X	X	X	X	X	X	X	X
3	X	X	X	<31:24>	<23:16>	<15: 8>	< 7: 0>	X
	X	X	X	X	X	X	X	X
4	X	X	X	X	<31:24>	<23:16>	<15: 8>	< 7: 0>
	X	X	X	X	X	X	X	X
5	X	X	X	X	X	<31:24>	<23:16>	<15: 8>
	< 7: 0>	X	X	X	X	X	X	X
6	X	X	X	X	X	X	<31:24>	<23:16>
	<15: 8>	< 7: 0>	X	X	X	X	X	X
7	X	X	X	X	X	X	X	<31:24>
	<23:16>	<15: 8>	< 7: 0>	X	X	X	X	X

The following table indicates the byte lanes for 32-bit references in little-endian mode:

VA<2:0>	Register File Byte Positions in the two memory doublewords							
0	X	X	X	X	<31:24>	<23:16>	<15: 8>	< 7: 0>
	X	X	X	X	X	X	X	X
1	X	X	X	<31:24>	<23:16>	<15: 8>	< 7: 0>	X
	X	X	X	X	X	X	X	X
2	X	X	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X
	X	X	X	X	X	X	X	X
3	X	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X
	X	X	X	X	X	X	X	X
4	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X	X
	X	X	X	X	X	X	X	X
5	<23:16>	<15: 8>	< 7: 0>	X	X	X	X	X
	X	X	X	X	X	X	X	<31:24>
6	<15: 8>	< 7: 0>	X	X	X	X	X	X
	X	X	X	X	X	X	<31:24>	<23:16>
7	< 7: 0>	X	X	X	X	X	X	X
	X	X	X	X	X	<31:24>	<23:16>	<15: 8>

Cavium Confidential For Bowen Yang ALU 02/27/2014

Unaligned Store Doubleword USD



Format: USD rt, offset(base) **CVM**

The USD instruction does not exist when CvmCtl[USEUN] is clear. (The MIPS SDL and SDR instructions do not exist when CvmCtl[USEUN] is set.)

The USD instruction (when enabled by CvmCtl[USEUN]) consumes either the MIPS SDL or SDR major opcodes according to the following table:

CvmCtl[USELY]	BigEndianCPU (CvmCtl[$\overline{LE}$])	MIPS SDL opcode is:	MIPS SDR opcode is:
0	0	NOP	USD
0	1	USD	NOP
1	X	USD	NOP

When CvmCtl[USEUN] is set, one of MIPS SDL and SDR major opcodes execute as NOPs, as specified in the table above.

Purpose:

To store a doubleword to a (potentially-unaligned) memory location.

Description: memory[base+offset] = rt

The 64-bit doubleword in GPR rt is stored in memory at the location specified by the effective address. The 16-bit signed offset is added to the contents of GPR base to form the effective address.

If the effective address is not naturally-aligned and the second of the two memory stores required to complete the unaligned reference encounters an Address Error, TLB Refill, TLB Invalid, or TLB Modified exception, the first of the two memory references will still complete. Effectively, the USD may partially complete.

If the effective address is not naturally-aligned, the two memory stores required to complete the unaligned reference may complete in any order (as seen by other CN71XX Cores and I/O devices).

Restrictions:

Whenever any of the bytes required to service the store reside outside of normal/ cacheable memory and CVMSEG LM, the effective address must be naturally aligned, else an address-error exception occurs. Note that these alignment-address error exceptions are lower priority than TLB refill and invalid exceptions.CVMSEG I/O references cause address errors.

CVMSEG LM references by these store operations cause an address error when outside the legal range allowed by CvmMemCtl[LMEMSZ] (assuming CVMSEG LM is enabled by CvmMemCtl[CVMSEGENA*]).

Cavium Confidential For Bowen Yang 03/28/2024

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

```
 if !Are64bitOperationsEnabled() then
    SignalException(ReservedInstruction)
 endif
 vAddr ← sign_extend(offset) + GPR[base]
 (pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
 if (vAddr/pAddr has address error) then
    SignalException(AddressError)
 endif
 datadoubleword = GPR[rt]
 if CvmCtl[LE] then
    StoreMemory (CCA, DOUBLEWORD-vAddr<2:0>, datadoubleword << ((vAddr<2:0>) *
8), pAddr, vAddr, DATA)
 else
    StoreMemory (CCA, DOUBLEWORD-vAddr<2:0>, datadoubleword >> ((vAddr<2:0>) *
8), pAddr, vAddr, DATA)
 endif
 if(vAddr<2:0> >= 1) then
    vAddr2 ← vAddr + 8
    (pAddr2, CCA2) ← AddressTranslation (vAddr2, DATA, LOAD)
    if (vAddr2/pAddr2 has address error)
       SignalException(AddressError) // vAddr+8 appears in BadVAddr in this case
    endif
    amsk = 1^61 | 0^3
    if CvmCtl[LE] then
       StoreMemory (CCA2, vAddr<2:0>, datadoubleword >> ((8-vAddr<2:0>) * 8), pAddr2 AND
amsk, vAddr2 AND amsk, DATA)
    else
       StoreMemory (CCA2, vAddr<2:0>, datadoubleword << ((8-vAddr<2:0>) * 8), pAddr2 AND
amsk, vAddr2 AND amsk, DATA)
    endif
endif
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Reserved Instruction, Watch, Breakpoint

Notes:

CN71XX cores execute naturally-aligned SD instructions one cycle faster than naturally-aligned USD instructions, so the SD instructions should be used rather than USD when an address is known to be naturally-aligned.

USD is an assembler macro that converts to MIPS SDL/SDR sequences on most MIPS assemblers.

Cavium Confidential For Bowen Yang ALU 07/7/2014

The following table indicates the byte lanes for 64-bit references in big-endian mode:

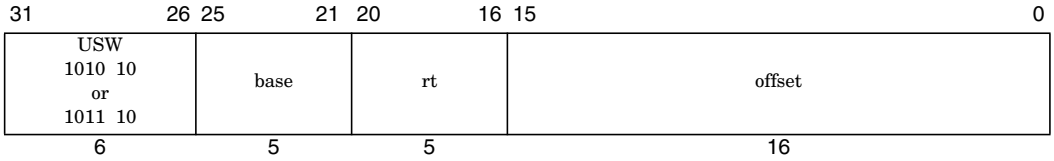
VA<2:0>	Register File Byte Positions in the two memory doublewords							
0	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>
	X	X	X	X	X	X	X	X
1	X	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>
	< 7: 0>	X	X	X	X	X	X	X
2	X	X	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>
	<15: 8>	< 7: 0>	X	X	X	X	X	X
3	X	X	X	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>
	<23:16>	<15: 8>	< 7: 0>	X	X	X	X	X
4	X	X	X	X	<63:56>	<55:48>	<47:40>	<39:32>
	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X	X
5	X	X	X	X	X	<63:56>	<55:48>	<47:40>
	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X
6	X	X	X	X	X	X	<63:56>	<55:48>
	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X
7	X	X	X	X	X	X	X	<63:56>
	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>	X

The following table indicates the byte lanes for 64-bit references in little-endian mode:

VA<2:0>	Register File Byte Positions in the two memory doublewords							
0	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>
	X	X	X	X	X	X	X	X
1	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>	X
	X	X	X	X	X	X	X	<63:56>
2	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X
	X	X	X	X	X	X	<63:56>	<55:48>
3	<39:32>	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X
	X	X	X	X	X	<63:56>	<55:48>	<47:40>
4	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X	X
	X	X	X	X	<63:56>	<55:48>	<47:40>	<39:32>
5	<23:16>	<15: 8>	< 7: 0>	X	X	X	X	X
	X	X	X	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>
6	<15: 8>	< 7: 0>	X	X	X	X	X	X
	X	X	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>
7	< 7: 0>	X	X	X	X	X	X	X
	X	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15: 8>

Cavium Confidential For Bowen Yang ALU 02/27/2014

Unaligned Store Word USW



Format: USW rt, offset(base) **CVM**

The USW instruction does not exist when CvmCtl[USEUN] is clear. (The MIPS SWL and SWR instructions do not exist when CvmCtl[USEUN] is set.)

The USW instruction (when enabled by CvmCtl[USEUN]) consumes either the MIPS SWL or SWR major opcodes according to the following table:

CvmCtl[USELY]	BigEndianCPU (CvmCtl[$\overline{LE}$])	MIPS SWL opcode is:	MIPS SWR opcode is:
0	0	NOP	USW
0	1	USW	NOP
1	X	USW	NOP

When CvmCtl[USEUN] is set, one of MIPS SWL and SWR major opcodes execute as NOPs, as specified in the table above.

Purpose:

To store a word to a (potentially-unaligned) memory location.

Description: memory[base+offset] = rt

The 64-bit word in GPR rt is stored in memory at the location specified by the effective address. The 16-bit signed offset is added to the contents of GPR base to form the effective address.

If the effective address is not naturally-aligned and two memory stores are required to complete the unaligned reference and the second of the two memory stores encounters an Address Error, TLB Refill, TLB Invalid, or TLB Modified exception, the first of the two memory references will still complete. Effectively, the USW may partially complete.

If the effective address is not naturally-aligned and two memory stores are required to complete the unaligned reference, the two memory stores may complete in any order (as seen by other CN71XX Cores and I/O devices).

Restrictions:

Whenever any of the bytes required to service the store reside outside of normal/ cacheable memory and CVMSEG LM, the effective address must be naturally aligned, else an address error exception occurs. Note that these alignment address error exceptions are lower priority than TLB refill and invalid exceptions.

CVMSEG I/O references cause address errors. CVMSEG LM references by these store operations cause an address error when outside the legal range allowed by CvmMemCtl[LMEMSZ] (assuming CVMSEG LM is enabled by CvmMemCtl[CVMSEGENA*]).

Cavium Confidential - For Boren Yang 02/2/2012

Operation:

```
 if !Are64bitOperationsEnabled() then
    SignalException(ReservedInstruction)
 endif
 vAddr ← sign_extend(offset) + GPR[base]
 (pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
 if (vAddr/pAddr has address error) then
    SignalException(AddressError)
 endif
 datadoubleword = GPR[rt]
 if(vAddr<2:0> >= 5) then
    if CvmCtl[LE] then
       StoreMemory (CCA, WORD-vAddr<1:0>, datadoubleword << (vAddr<2:0> * 8),
pAddr, vAddr, DATA)
    else
       StoreMemory (CCA, WORD-vAddr<1:0>, datadoubleword >> ((vAddr<2:0>-4) * 8),
pAddr, vAddr, DATA)
    endif
    vAddr2 ← vAddr + 8
    (pAddr2, CCA2) ← AddressTranslation (vAddr2, DATA, LOAD)
    if (vAddr2/pAddr2 has address error)
       SignalException(AddressError) // vAddr+8 appears in BadVAddr in this case
    endif
    amsk = 1^61 | 0^3
    if CvmCtl[LE] then
       StoreMemory (CCA2, vAddr<1:0>, datadoubleword >> ((8-vAddr<2:0>) * 8), pAddr2 AND
amsk, vAddr2 AND amsk, DATA)
    else
       StoreMemory (CCA2, vAddr<1:0>, datadoubleword << ((12-vAddr<2:0>) * 8), pAddr2 AND
amsk, vAddr2 AND amsk, DATA)
    endif
 else
    if CvmCtl[LE] then
       StoreMemory (CCA, WORD, datadoubleword << (vAddr<2:0> * 8), pAddr, vAddr,
DATA)
    else
       StoreMemory (CCA, WORD, datadoubleword << ((4-vAddr<2:0>) * 8), pAddr,
vAddr, DATA)
    endif
 endif
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint

Notes:

CN71XX cores execute naturally-aligned SWs one cycle faster than naturally-aligned USWs, so the SW instructions should be used rather than USW when an address is known to be naturally-aligned.

USW is an assembler macro that converts to MIPS SWL/SWR sequences on most MIPS assemblers.

Cavium Confidential For Bowen Yang ALU 08/27/2014

The following table indicates the byte lanes for 32-bit references in big-endian mode:

VA<2:0>	Register File Byte Positions in the two memory doublewords							
0	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X	X
	X	X	X	X	X	X	X	X
1	X	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X
	X	X	X	X	X	X	X	X
2	X	X	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X
	X	X	X	X	X	X	X	X
3	X	X	X	<31:24>	<23:16>	<15: 8>	< 7: 0>	X
	X	X	X	X	X	X	X	X
4	X	X	X	X	<31:24>	<23:16>	<15: 8>	< 7: 0>
	X	X	X	X	X	X	X	X
5	X	X	X	X	X	<31:24>	<23:16>	<15: 8>
	< 7: 0>	X	X	X	X	X	X	X
6	X	X	X	X	X	X	<31:24>	<23:16>
	<15: 8>	< 7: 0>	X	X	X	X	X	X
7	X	X	X	X	X	X	X	<31:24>
	<23:16>	<15: 8>	< 7: 0>	X	X	X	X	X

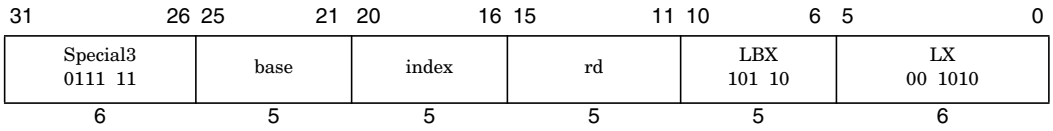
The following table indicates the byte lanes for 32-bit references in little-endian mode:

VA<2:0>	Register File Byte Positions in the two memory doublewords							
0	X	X	X	X	<31:24>	<23:16>	<15: 8>	< 7: 0>
	X	X	X	X	X	X	X	X
1	X	X	X	<31:24>	<23:16>	<15: 8>	< 7: 0>	X
	X	X	X	X	X	X	X	X
2	X	X	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X
	X	X	X	X	X	X	X	X
3	X	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X
	X	X	X	X	X	X	X	X
4	<31:24>	<23:16>	<15: 8>	< 7: 0>	X	X	X	X
	X	X	X	X	X	X	X	X
5	<23:16>	<15: 8>	< 7: 0>	X	X	X	X	X
	X	X	X	X	X	X	X	<31:24>
6	<15: 8>	< 7: 0>	X	X	X	X	X	X
	X	X	X	X	X	X	<31:24>	<23:16>
7	< 7: 0>	X	X	X	X	X	X	X
	X	X	X	X	X	<31:24>	<23:16>	<15: 8>

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.3.5 Integer Indexed Load Instructions

Load Byte Indexed **LBX**



Format: LBX rd, index(base) **CVMv2**

Purpose:

To load a byte from memory as a signed value with register+register addressing.

Description: GPR[rd] ← Mem[GPR[index] + GPR[base]]

The contents of the 8-bit byte at the memory location specified by the effective address are fetched, sign-extended, and placed in GPR rd. GPR base is added to GPR index to form the effective address.

Restrictions:

None.

Operation:

```
vAddr ← GPR[base] + GPR[index]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
amsk2 = 0^61 || CvmCtl[LE]^3
memdoubleword ← LoadMemory (CCA, BYTE, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
bit ← 64 - ((vAddr<2..0> XOR CvmCtl[LE]^3) || 0^3)
GPR[rd] ← sign_extend(memdoubleword bit-1 .. bit-8)
```

Exceptions:

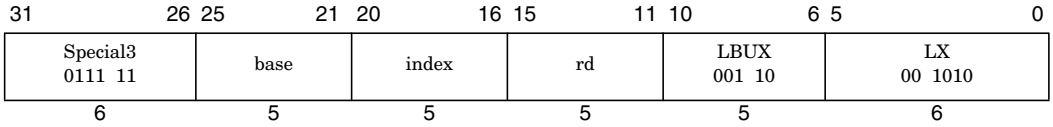
TLB Refill, TLB Invalid, Bus Error, Address Error, Watch, Breakpoint

Notes:

LBX instruction execution is exactly the same as LB instruction execution, except that the address calculation is register + register rather than register + offset.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Byte Unsigned Indexed LBUX



Format: LBUX rd, index(base) **CVMv2/MIPS DSP**

Purpose:

To load a byte from memory as an unsigned value with register+register addressing.

Description: GPR[rd] ← Mem[GPR[index] + GPR[base]]

The contents of the 8-bit byte at the memory location specified by the effective address are fetched, zero-extended, and placed in GPR rd. GPR base is added to GPR index to form the effective address.

Restrictions:

None.

Operation:

```
vAddr ← GPR[base] + GPR[index]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
amsk2 = 0^61 || CvmCtl[LE]^3
memdoubleword ← LoadMemory (CCA, BYTE, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
bit ← 64 - ((vAddr<2..0> XOR CvmCtl[LE]^3) || 0^3)
GPR[rd] ← zero_extend(memdoubleword<bit-1 .. bit-8>)
```

Exceptions:

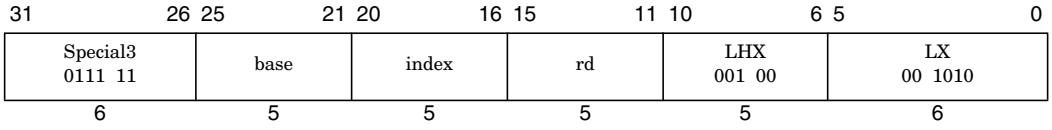
TLB Refill, TLB Invalid, Bus Error, Address Error, Watch, Breakpoint

Notes:

LBUX instruction execution is exactly the same as LBU instruction execution, except that the address calculation is register + register rather than register + offset.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Halfword Indexed — LHX



Format: LHX rd, index(base) **CVMv2/MIPS DSP**

Purpose:

To load a halfword from memory as a signed value with register+register addressing.

Description: GPR[rd] ← Mem[GPR[index] + GPR[base]]

The contents of the 16-bit halfword at the memory location specified by the aligned effective address are fetched, sign-extended, and placed in GPR rd. GPR base is added to GPR index to form the effective address.

Restrictions:

When CvmCtl[REPUN] is clear, the effective address must be naturally aligned, else an Address Error exception occurs.

Whenever any of the bytes required to service the load reside outside of normal/cacheable memory and CVMSEG LM, the effective address must be naturally aligned, else an address error exception occurs in all modes. When CvmCtl[REPUN] is set, note that these alignment address error exceptions are lower priority than TLB refill and invalid exceptions.

References to CVMSEG I/O cause address errors. CVMSEG LM references by these loads cause an address error when outside the legal range allowed by CvmMemCtl[LMEMSZ] (assuming CVMSEG LM is enabled by CvmMemCtl[CVMSEGENA*]).

Cavium Confidential For BowenYang ALU 02/27/2014

Operation:

```
vAddr ← GPR[base] + GPR[index]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr has address error) then
  SignalException(AddressError)
endif
amsk = 1^61 | 0^3
if (vAddr<0> == 0) then
  amsk2 = 0^61 || CvmCtl[LE]^2 || 0
  memdoubleword ← LoadMemory (CCA, HALFWORD, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
else
  memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr AND amsk, vAddr AND amsk, DATA)
endif
if(vAddr<2:0> >= 7) then
  vAddr2 ← vAddr + 8
  (pAddr2, CCA2) ← AddressTranslation (vAddr2, DATA, LOAD)
  if (vAddr2/pAddr2 has address error)
    SignalException(AddressError) // vAddr+8 appears in BadVAddr in this case
  endif
  // either vAddr2 is CVMSEG LM or paddr2 must be L2/DRAM
  memdoubleword2 ← LoadMemory (CCA2, DOUBLEWORD, pAddr2 AND amsk, vAddr2 AND amsk, DATA)
endif
if CvmCtl[LE] then
  memquadword<127..0> = memdoubleword2<63..0>|| memdoubleword<63..0>
  bit ← vAddr<2..0> || 0^3
  GPR[rd] ← sign_extend(memquadword<bit+15 .. bit>)
else
  memquadword<127..0> = memdoubleword<63..0> || memdoubleword2<63..0>
  bit ← 128 - (vAddr<2..0> || 0^3)
  GPR[rd] ← sign_extend(memquadword<bit-1 .. bit-16>)
endif
```

Exceptions:

TLB Refill, TLB Invalid, Bus Error, Address Error, Watch, Breakpoint

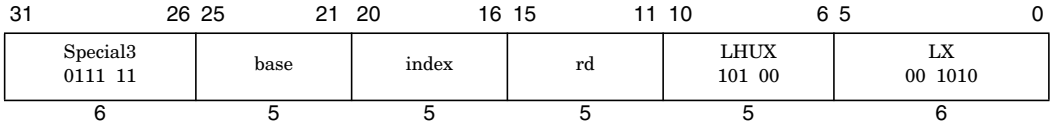
Notes:

LHX instruction execution is exactly the same as LH instruction execution, except that the address calculation is register + register rather than register + offset.

When CvmCtl[REPUN] is set, unaligned LHX instructions may be slow (approximately 8-12 cycles to execute) when the halfword crosses an aligned 64-bit boundary - other load methods may be faster.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Halfword Unsigned Indexed — LHUX



Format: LHUX rd, index(base) **CVMv2**

Purpose:

To load a halfword from memory as an unsigned value with register+register addressing.

Description: GPR[rd] ← Mem[GPR[index] + GPR[base]]

The contents of the 16-bit halfword at the memory location specified by the aligned effective address are fetched, zero-extended, and placed in GPR rd. GPR base is added to GPR index to form the effective address.

Restrictions:

When CvmCtl[REPUN] is clear, the effective address must be naturally aligned, else an Address Error exception occurs.

Whenever any of the bytes required to service the load reside outside of normal/cacheable memory and CVMSEG LM, the effective address must be naturally aligned, else an address error exception occurs in all modes. When CvmCtl[REPUN] is set, note that these alignment address error exceptions are lower priority than TLB refill and invalid exceptions.

References to CVMSEG I/O cause address errors. CVMSEG LM references by these loads cause an address error when outside the legal range allowed by CvmMemCtl[LMEMSZ] (assuming CVMSEG LM is enabled by CvmMemCtl[CVMSEGENA*]).

Cavium Confidential For Bowen Yang ALU 02/27/2014

Operation:

```
vAddr ← GPR[base] + GPR[index]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr has address error) then
  SignalException(AddressError)
endif
amsk = 1^61 | 0^3
if (vAddr<0> == 0) then
  amsk2 = 0^61 || CvmCtl[LE]^2 || 0
  memdoubleword ← LoadMemory (CCA, HALFWORD, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
else
  memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr AND amsk, vAddr AND amsk, DATA)
endif
if(vAddr<2:0> >= 7) then
  vAddr2 ← vAddr + 8
  (pAddr2, CCA2) ← AddressTranslation (vAddr2, DATA, LOAD)
  if (vAddr2/pAddr2 has address error)
    SignalException(AddressError) // vAddr+8 appears in BadVAddr in this case
  endif
  // either vAddr2 is CVMSEG LM or paddr2 must be L2/DRAM
  memdoubleword2 ← LoadMemory (CCA2, DOUBLEWORD, pAddr2 AND amsk, vAddr2 AND amsk, DATA)
endif
if CvmCtl[LE] then
  memquadword<127..0> = memdoubleword2<63..0> || memdoubleword<63..0>
  bit ← vAddr<2..0> || 0^3
  GPR[rd] ← zero_extend(memquadword<bit+15 .. bit>)
else
  memquadword<127..0> = memdoubleword<63..0> || memdoubleword2<63..0>
  bit ← 128 - (vAddr<2..0> || 0^3)
  GPR[rd] ← zero_extend(memquadword<bit-1 .. bit-16>)
endif
```

Exceptions:

TLB Refill, TLB Invalid, Bus Error, Address Error, Watch, Breakpoint

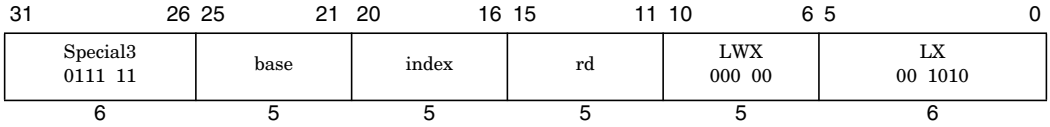
Notes:

LHUX instruction execution is exactly the same as LHU instruction execution, except that the address calculation is register + register rather than register + offset.

When CvmCtl[REPUN] is set, unaligned LHUX instructions may be slow (approximately 8-12 cycles to execute) when the halfword crosses an aligned 64-bit boundary - other load methods may be faster.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Word Indexed — LWX



Format: LWX rd, index(base) **CVMv2/MIPS DSP**

Purpose:

To load a word from memory as a signed value with register+register addressing.

Description: GPR[rd] ← Mem[GPR[index] + GPR[base]]

The contents of the 32-bit word at the memory location specified by the effective address are fetched, sign-extended to the GPR register length if necessary, and placed in GPR rd. GPR base is added to GPR index to form the effective address.

Restrictions:

When CvmCtl[REPUN] is clear, the effective address must be naturally aligned, else an Address Error exception occurs.

Whenever any of the bytes required to service the load reside outside of normal/cacheable memory and CVMSEG LM, the effective address must be naturally aligned, else an address error exception occurs in all modes. When CvmCtl[REPUN] is set, note that these alignment address error exceptions are lower priority than TLB refill and invalid exceptions.

References to CVMSEG I/O cause address errors. CVMSEG LM references by these loads cause an address error when outside the legal range allowed by CvmMemCtl[LMEMSZ] (assuming CVMSEG LM is enabled by CvmMemCtl[CVMSEGENA*]).

Cavium Confidential For Bowen Yang ALU 02/27/2014

Operation:

```
vAddr ← GPR[base] + GPR[index]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr has address error) then
  SignalException(AddressError)
endif
amsk = 1^61 | 0^3
if (vAddr<1:0> == 0) then
  amsk2 = 0^61 || CvmCtl[LE] || 0^2
  memdoubleword ← LoadMemory (CCA, WORD, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
else
  memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr AND amsk, vAddr AND amsk, DATA)
endif
if(vAddr<2:0> >= 5) then
  vAddr2 ← vAddr + 8
  (pAddr2, CCA2) ← AddressTranslation (vAddr2, DATA, LOAD)
  if (vAddr2/pAddr2 has address error)
    SignalException(AddressError) // vAddr+8 appears in BadVAddr in this case
  endif
  // either vAddr2 is CVMSEG LM or paddr2 must be L2/DRAM
  memdoubleword2 ← LoadMemory (CCA2, DOUBLEWORD, pAddr2 AND amsk, vAddr2 AND amsk, DATA)
endif
if CvmCtl[LE] then
  memquadword<127..0> = memdoubleword2<63..0> || memdoubleword<63..0>
  bit ← vAddr<2..0> || 0^3
  GPR[rd] ← sign_extend(memquadword<bit+31 .. bit>)
else
  memquadword<127..0> = memdoubleword<63..0> || memdoubleword2<63..0>
  bit ← 128 - (vAddr<2..0> || 0^3)
  GPR[rd] ← sign_extend(memquadword<bit-1 .. bit-32>)
endif
```

Exceptions:

TLB Refill, TLB Invalid, Bus Error, Address Error, Watch, Breakpoint

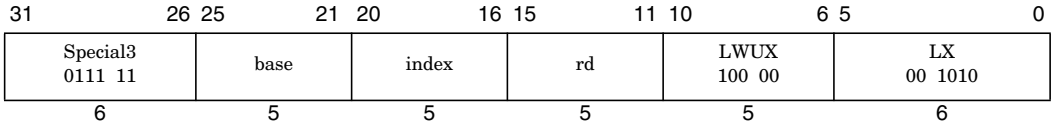
Notes:

LWX instruction execution is exactly the same as LW instruction execution, except that the address calculation is register + register rather than register + offset.

When CvmCtl[REPUN] is set, unaligned LWX instructions may be slow (approximately 8-12 cycles to execute) when the word crosses an aligned 64-bit boundary - other load methods, like ULW, may be faster.

Cavium Confidential For Bowen Yang ALU 02/17/2014

Load Word Unsigned Indexed — LWUX



Format: LWUX rd, index(base) **CVMv2**

Purpose:

To load a word from memory as an unsigned value with register+register addressing.

Description: GPR[rd] ← Mem[GPR[index] + GPR[base]]

The contents of the 32-bit word at the memory location specified by the effective address are fetched, zero-extended, and placed in GPR index. GPR base is added to GPR index to form the effective address.

Restrictions:

When CvmCtl[REPUN] is clear, the effective address must be naturally aligned, else an Address Error exception occurs.

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Whenever any of the bytes required to service the load reside outside of normal/ cacheable memory and CVMSEG LM, the effective address must be naturally aligned, else an address error exception occurs in all modes. When CvmCtl[REPUN] is set, note that these alignment address error exceptions are lower priority than TLB refill and invalid exceptions.

References to CVMSEG I/O cause address errors. CVMSEG LM references by these loads cause an address error when outside the legal range allowed by CvmMemCtl[LMEMSZ] (assuming CVMSEG LM is enabled by CvmMemCtl[CVMSEGENA*]).

Cavium Confidential For Bowen Yang A.U 02/27/2018

Operation:

```
if !Are64bitOperationsEnabled() then
  SignalException(ReservedInstruction)
endif
vAddr ← GPR[base] + GPR[index]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr has address error) then
  SignalException(AddressError)
endif
amsk = 1^61 | 0^3
if (vAddr<1:0> == 0) then
  amsk2 = 0^61 || CvmCtl[LE] || 0^2
  memdoubleword ← LoadMemory (CCA, WORD, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
else
  memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr AND amsk, vAddr AND amsk, DATA)
endif
if(vAddr<2:0> >= 5) then
  vAddr2 ← vAddr + 8
  (pAddr2, CCA2) ← AddressTranslation (vAddr2, DATA, LOAD)
  if (vAddr2/pAddr2 has address error)
    SignalException(AddressError) // vAddr+8 appears in BadVAddr in this case
  endif
  // either vAddr2 is CVMSEG LM or paddr2 must be L2/DRAM
  memdoubleword2 ← LoadMemory (CCA2, DOUBLEWORD, pAddr2 AND amsk, vAddr2 AND amsk, DATA)
endif
if CvmCtl[LE] then
  memquadword<127..0> = memdoubleword2<63..0> || memdoubleword<63..0>
  bit ← vAddr<2..0> || 0^3
  GPR[rd] ← zero_extend(memquadword<bit+31 .. bit>)
else
  memquadword<127..0> = memdoubleword<63..0> || memdoubleword2<63..0>
  bit ← 128 - (vAddr<2..0> || 0^3)
  GPR[rd] ← zero_extend(memquadword<bit-1 .. bit-32>)
endif
```

Exceptions:

TLB Refill, TLB Invalid, Bus Error, Address Error, Watch, Breakpoint, Reserved Instruction.

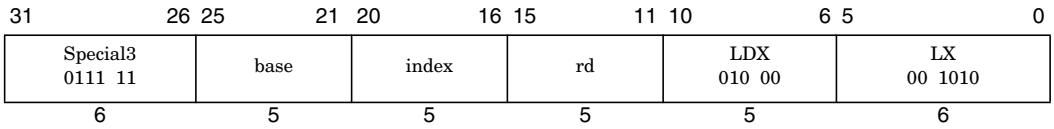
Notes:

LWUX instruction execution is exactly the same as LWU instruction execution, except that the address calculation is register + register rather than register + offset.

When CvmCtl[REPUN] is set, unaligned LWUX instructions may be slow (approximately 8-12 cycles to execute) when the word crosses an aligned 64-bit boundary - other load methods, like ULW, may be faster.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Doubleword Indexed — LDX



Format: LDX rd, index(base) **CVMv2**

Purpose:

To load a doubleword from memory with register+register addressing.

Description: GPR[rd] ← Mem[GPR[index] + GPR[base]]

Restrictions:

When CvmCtl[REPUN] is clear, the effective address must be naturally aligned, else an Address Error exception occurs.

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Whenever any of the bytes required to service the load reside outside of normal/cacheable memory and CVMSEG LM, the effective address must be naturally aligned, else an address error exception occurs in all modes. When CvmCtl[REPUN] is set, note that these alignment address error exceptions are lower priority than TLB refill and invalid exceptions.

References to CVMSEG I/O cause address errors. CVMSEG LM references by these loads cause an address error when outside the legal range allowed by CvmMemCtl[LMEMSZ] (assuming CVMSEG LM is enabled by CvmMemCtl[CVMSEGENA*]).

Cavium Confidential For Bowen Yang ALU 02/27/2014

Operation:

```
 if !Are64bitOperationsEnabled() then
    SignalException(ReservedInstruction)
 endif
 vAddr ← GPR[base] + GPR[index]
 (pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
 if (vAddr/pAddr has address error) then
    SignalException(AddressError)
 endif
 amsk = 1^61 | 0^3
 memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr AND amsk, vAddr AND amsk,
DATA)
 if (vAddr<2:0> != 0) then
    vAddr2 ← vAddr + 8
    (pAddr2, CCA2) ← AddressTranslation (vAddr2, DATA, LOAD)
    if (vAddr2/pAddr2 has address error)
       SignalException(AddressError) // vAddr+8 is in BadVAddr in this case
    endif
    // either vAddr2 is CVMSEG LM or paddr2 must be L2/DRAM
    memdoubleword2 ← LoadMemory (CCA2, DOUBLEWORD, pAddr2 AND amsk, vAddr2 AND
amsk, DATA)
 endif
if CvmCtl[LE] then
  memquadword<127..0> = memdoubleword2<63..0> || memdoubleword<63..0>
  bit ← vAddr<2..0> || 0^3
  GPR[rd] ← memquadword<bit+31 .. bit>
else
  memquadword<127..0> = memdoubleword<63..0> || memdoubleword2<63..0>
  bit ← 128 - (vAddr<2..0> || 0^3)
  GPR[rd] ← memquadword<bit-1 .. bit-32>
 endif
```

Exceptions:

TLB Refill, TLB Invalid, Bus Error, Address Error, Watch, Breakpoint, Reserved Instruction.

Notes:

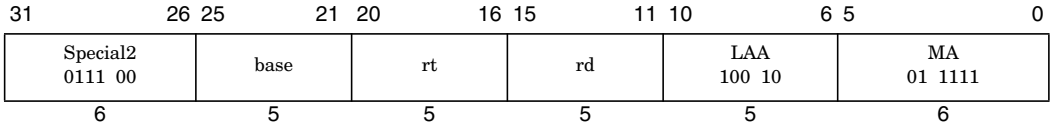
LDX instruction execution is exactly the same as LD instruction execution, except that the address calculation is register + register rather than register + offset.

When CvmCtl[REPUN] is set, unaligned LDX instructions may be slow (approximately 8-12 cycles to execute) when the word crosses an aligned 64-bit boundary - other load methods, like ULD, may be faster.

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.3.6 Memory Atomic Fetch and Add Instructions

Load Atomic Add Word **LAA**



Format: LAA rd, (base), rt **CVMv2**

Purpose:

To load from and add to a word of memory atomically.

Description:
```
GPR[rd] ← Mem[GPR[base]];                          // atomic
Mem[GPR[base]] = Mem[GPR[base]] + GPR[rt];         // atomic
```

The least-significant 32-bit word of register rt is added to the memory at the 32-bit aligned address specified by register base after the memory location is read. Register rd gets the sign-extended memory read result.

The memory read, add, and memory store operations are not interrupted by any other instructions on this or any other processor. No Integer Overflow exception occurs under any circumstance.

TLB accesses, watches, and breakpoints commonly treat these instructions as both a load operation and a store operation. The specifics of the instruction are:

- For address errors, Cause[ExcCode] is never AdEL (it is AdES always).
- For TLB exceptions, this instruction is both a load operation and a store operation. The details are:
  - Cause[ExcCode] is TLBL only when the sole cause of the TLB exception is a set RI bit (it is typically TLBS)
  - A TLB Modified exception is higher priority than a TLB Invalid caused solely by a set RI bit.
- For watchpoints, this instruction is both a load operation and a store operation.
- For breakpoints, this instruction is both a load operation and a store operation. The details are:
  - Any of DDBS, DDBL, and DDBLImpr exceptions are possible.
  - A DDBL Exception will not occur whenever a DDBS Exception could occur instead.
  - Load-value breakpoints check against the actual value returned from the memory load operation (i.e. the old contents of the memory location).
  - Store-value breakpoints match regardless of the DBV and add/memory byte values.

LAA generates a FAA32 CMI transaction – refer to Section 3.1.

Cavium Confidential For Bowen Yang ALU 02/7/2014

Restrictions:

The address in the base register must be naturally aligned, else an address error occurs.

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results.

Operation:

```
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
amsk2 = 0^61 || CvmCtl[LE] || 0^2
memdoubleword ← LoadMemory (CCA, WORD, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
bit ← 64 - ((vAddr<2> XOR CvmCtl[LE]) || 0^5)
resultdoubleword = memdoubleword + (GPR[rt]<63:0> << (bit-32))
StoreMemory (CCA, WORD, resultdoubleword, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
GPR[rd] ← sign_extend(memdoubleword<bit-1 .. bit-32>)
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint

Programming Notes:

This instruction is performance-optimized for memory locations that are present in the L2 cache, not the L1 data cache. The cache block that contains the memory location is removed from the L1 data cache (if present) during execution of the instruction.

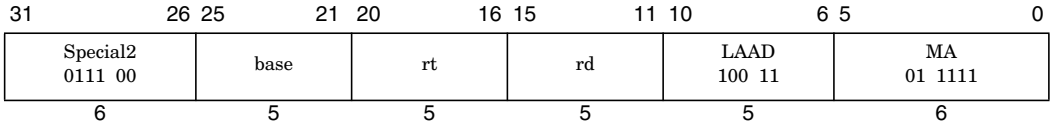
LAI or LAD should be used rather than LAA whenever rt is statically known to be 1 or all 1s (i.e. –1), respectively.

LAA/LAI/LAD execution is very similar to SAA, except that the previous value of the memory location is returned in a register. Different instructions can be used in different circumstances:

- SAA should always be used rather than LAA/LAI/LAD if the memory location need not be read near the operation.
- If the memory location needs to be read atomically as part of the operation, then LAA/LAI/LAD must be used rather than SAA.
- If the memory location needs to be read sometime near the operation, but not atomically with the update, then:
  - If the block is likely to be in the L1 cache before the operation, LW+SAA may perform better than LAA/LAI/LAD.
  - If the block is not likely to be in the L1 cache before the operation, either LAA/LAI/LAD or SAA+LW or SAA+PREF+LW sequences could perform best, depending on how useful it is to put the block into the L1 after the operation, and whether the PREF can hide the cache-miss latency.

Cavium Confidential For Bowen Yong ALU 02/27/2014

Load Atomic Add Doubleword — LAAD



Format: LAAD rd, (base), rt **CVMv2**

Purpose:

To load from and add to a doubleword of memory atomically.

Description:
```
GPR[rd] ← Mem[GPR[base]];                          // atomic
Mem[GPR[base]] = Mem[GPR[base]] + GPR[rt];         // atomic
```

Register rt is added to the memory at the 64-bit-aligned address specified by register base after the memory location is read. Register rd gets the memory read result.

The memory read, add, and memory store operations are not interrupted by any other instructions on this or any other processor. No Integer Overflow exception occurs under any circumstance.

TLB accesses, watches, and breakpoints commonly treat these instructions as both a load operation and a store operation. The specifics of the instruction are:

- For address errors, Cause[ExcCode] is never AdEL (it is AdES always).
- For TLB exceptions, this instruction is both a load operation and a store operation. The details are:
  - Cause[ExcCode] is TLBL only when the sole cause of the TLB exception is a set RI bit (it is typically TLBS)
  - A TLB Modified exception is higher priority than a TLB Invalid caused solely by a set RI bit.
- For watchpoints, this instruction is both a load operation and a store operation.
- For breakpoints, this instruction is both a load operation and a store operation. The details are:
  - Any of DDBS, DDBL, and DDBLImpr exceptions are possible.
  - A DDBL exception will not occur whenever a DDBS exception could occur instead.
  - Load-value breakpoints check against the actual value returned from the memory load operation (i.e. the old contents of the memory location).
  - Store-value breakpoints match regardless of the DBV and add/memory byte values.

LAAD generates a FAA64 CMI transaction – refer to Section 3.1.

Cavium Confidential For Bowen Yang At U 02/27/2014

Restrictions:

The address in the base register must be naturally aligned, else an address error occurs.

A Reserved Instruction exception is signaled if access to 64-bit operations is not enabled. The address must refer to an L2/DRAM location. Any other address may produce unpredictable results.

Operation:

```
if !Are64bitOperationsEnabled() then
   SignalException(ReservedInstruction)
endif
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr, vAddr, DATA)
resultdoubleword = memdoubleword + GPR[rt]
StoreMemory (CCA, DOUBLEWORD, resultdoubleword, pAddr, vAddr, DATA)
GPR[rd] ← memdoubleword
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint, Reserved Instruction.

Programming Notes:

This instruction is performance-optimized for memory locations that are present in the L2 cache, not the L1 data cache. The cache block that contains the memory location is removed from the L1 data cache (if present) during execution of the instruction.

LAID or LADD should be used rather than LAAD whenever rt is statically known to be 1 or all 1s (i.e. –1), respectively.

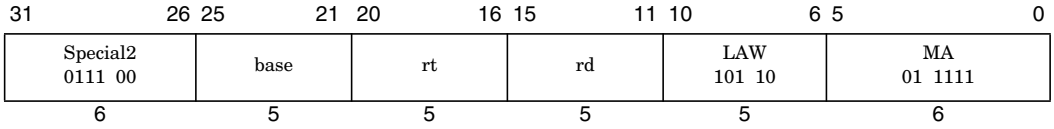
LAAD/LAID/LADD execution is very similar to SAAD, except that the previous value of the memory location is returned in a register. Different instructions can be used in different circumstances:

- SAAD should always be used rather than LAAD/LAID/LADD if the memory location need not be read near the operation.
- If the memory location needs to be read atomically as part of the operation, then LAAD/LAID/LADD must be used rather than SAAD.
- If the memory location needs to be read sometime near the operation, but not atomically with the update, then:
  - If the block is likely to be in the L1 cache before the operation, LW+SAAD may perform better than LAAD/LAID/LADD
  - If the block is not likely to be in the L1 cache before the operation, either LAAD/LAIDLADD or SAAD+LD or SAAD+PREF+LD sequences could perform best, depending on how useful it is to put the block into the L1 after the operation, and whether the PREF can hide the cache-miss latency

Cavium Confidential For Rowen Yang ALD 02/20/2014

A.3.7 Memory Atomic Swap Instructions

Load Atomic Swap Word **LAW**



Format: LAW rd, (base), rt **CVMv2**

Purpose:

To load from and write to a word of memory atomically (i.e. swap).

Description:
```
GPR[rd] ← Mem[GPR[base]];          // atomic
Mem[GPR[base]] = GPR[rt];          // atomic
```

The least-significant 32-bit word of register rt is written to the memory at the 32-bit aligned address specified by register base after the memory location is read. Register rd gets the sign-extended memory read result.

The memory read and memory store operations are not interrupted by any other instructions on this or any other processor. No Integer Overflow exception occurs under any circumstance.

TLB accesses, watches, and breakpoints commonly treat these instructions as both a load operation and a store operation. The specifics of the instruction are:

- For address errors, Cause[ExcCode] is never AdEL (it is AdES always).
- For TLB exceptions, this instruction is both a load operation and a store operation. The details are:
  - Cause[ExcCode] is TLBL only when the sole cause of the TLB exception is a set RI bit (it is typically TLBS)
  - A TLB Modified exception is higher priority than a TLB Invalid exception caused solely by a set RI bit.
- For watchpoints, this instruction is both a load operation and a store operation.
- For breakpoints, this instruction is both a load operation and a store operation. The details are:
  - Any of DDBS, DDBL, and DDBLImpr exceptions are possible.
  - A DDBL Exception will not occur whenever a DDBS Exception could occur instead.
  - Load-value breakpoints check against the actual value returned from the memory load operation (i.e. the old contents of the memory location).
  - Store-value breakpoints match when the rt bytes match the appropriate DBV bytes.

LAW generates a FAS32 CMI transaction – refer to Section 3.1.

Cavium Confidential For Bowen Yang AL U 02/12/2014

Restrictions:

The address in the base register must be naturally aligned, else an address error occurs.

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results.

Operation:

```
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
amsk2 = 0^61 || CvmCtl[LE] || 0^2
memdoubleword ← LoadMemory (CCA, WORD, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
bit ← 64 - ((vAddr<2> XOR CvmCtl[LE]) || 0^5)
writedoubleword = (GPR[rt]<63:0> << (bit-32))
StoreMemory (CCA, WORD, writedoubleword, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
GPR[rd] ← sign_extend(memdoubleword<bit-1 .. bit-32>)
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint

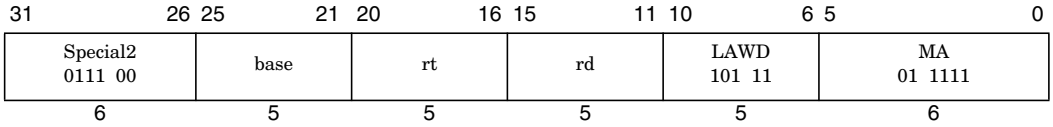
Programming Notes:

This instruction is performance-optimized for memory locations that are present in the L2 cache, not the L1 data cache. The cache block that contains the memory location is removed from the L1 data cache (if present) during execution of the instruction.

LAC or LAS should be used rather than LAW whenever rt is statically known to be 0 or all 1s, respectively.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Atomic Swap Doubleword — LAWD



Format: LAWD rd, (base), rt **CVMv2**

Purpose:

To load from and write to a doubleword of memory atomically (i.e. swap).

Description:
```
GPR[rd] ← Mem[GPR[base]];          // atomic
Mem[GPR[base]] = GPR[rt];          // atomic
```

Register rt is written to the memory at the 64-bit-aligned address specified by register base after the memory location is read. Register rd gets the memory read result.

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

The memory read and memory store operations are not interrupted by any other instructions on this or any other processor.

TLB accesses, watches, and breakpoints commonly treat these instructions as both a load operation and a store operation. The specifics of the instruction are:

- For address errors, Cause[ExcCode] is never AdEL (it is AdES always).
- For TLB exceptions, this instruction is both a load operation and a store operation. The details are:
  - Cause[ExcCode] is TLBL only when the sole cause of the TLB exception is a set RI bit (it is typically TLBS)
  - A TLB Modified exception is higher priority than a TLB Invalid caused solely by a set RI bit.
- For watchpoints, this instruction is both a load operation and a store operation.
- For breakpoints, this instruction is both a load operation and a store operation. The details are:
  - Any of DDBS, DDBL, and DDBLImpr exceptions are possible.
  - A DDBL exception will not occur whenever a DDBS exception could occur instead.
  - Load-value breakpoints check against the actual value returned from the memory load operation (i.e. the old contents of the memory location).
  - Store-value breakpoints match when the rt bytes match the appropriate DBV bytes.

LAWD generates a FAS64 CMI transaction – refer to Section 3.1.

Restrictions:

The address in the base register must be naturally aligned, else an address error occurs.

Cavium Confidential For Bowen Yang ALU 02/27/2014

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results.

Operation:

```
if !Are64bitOperationsEnabled() then
   SignalException(ReservedInstruction)
endif
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr, vAddr, DATA)
StoreMemory (CCA, DOUBLEWORD, GPR[rt], pAddr, vAddr, DATA)
GPR[rd] ← memdoubleword
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint, Reserved Instruction.

Programming Notes:

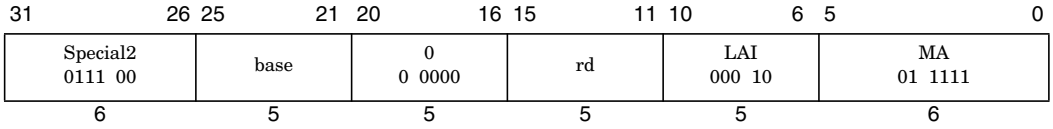
This instruction is performance-optimized for memory locations that are present in the L2 cache, not the L1 data cache. The cache block that contains the memory location is removed from the L1 data cache (if present) during execution of the instruction.

LACD or LASD should be used rather than LAW whenever rt is statically known to be 0 or all 1s, respectively.

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.3.8 Memory Atomic Increment Instructions

Load Atomic Increment Word **LAI**



Format: LAI rd, (base) **CVMv2**

Purpose:
To load and increment a word of memory atomically.

Description:
```
GPR[rd] ← Mem[GPR[base]];                 // atomic
Mem[GPR[base]] = Mem[GPR[base]] + 1;      // atomic
```

The memory at the 32-bit-aligned address specified by register base is incremented after being read. Register rd gets the sign-extended memory read result.

The memory read, increment, and memory store operations are not interrupted by any other instructions on this or any other processor. No Integer Overflow exception occurs under any circumstance.

TLB accesses, watches, and breakpoints commonly treat these instructions as both a load operation and a store operation. The specifics of the instruction are:

- For address errors, Cause[ExcCode] is never AdEL (it is AdES always).
- For TLB exceptions, this instruction is both a load operation and a store operation. The details are:
  - Cause[ExcCode] is TLBL only when the sole cause of the TLB exception is a set RI bit (it is typically TLBS)
  - A TLB Modified exception is higher priority than a TLB Invalid exception caused solely by a set RI bit.
- For watchpoints, this instruction is both a load operation and a store operation.
- For breakpoints, this instruction is both a load operation and a store operation. The details are:
  - Any of DDBS, DDBL, and DDBLImpr exceptions are possible.
  - A DDBL exception will not occur whenever a DDBS exception could occur instead.
  - Load-value breakpoints check against the actual value returned from the memory load operation (i.e. the old contents of the memory location).
  - Store-value breakpoints match regardless of the DBV byte values.

LAI generates an INCR32 CMI transaction – refer to Section 3.1.

Restrictions:

The address in the base register must be naturally aligned, else an address error occurs. The address must refer to an L2/DRAM location. Any other address may produce unpredictable results.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Operation:

```
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
amsk2 = 0^61 || CvmCtl[LE] || 0^2
memdoubleword ← LoadMemory (CCA, WORD, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
bit ← 64 - ((vAddr<2> XOR CvmCtl[LE]) || 0^5)
resultdoubleword = memdoubleword + (1 << (bit-32))
StoreMemory (CCA, WORD, resultdoubleword, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
GPR[rd] ← sign_extend(memdoubleword<bit-1 .. bit-32>)
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint, Reserved Instruction.

Programming Notes:

This instruction is performance-optimized for memory locations that are present in the L2 cache, not the L1 data cache. The cache block that contains the memory location is removed from the L1 data cache (if present) during execution of the instruction.

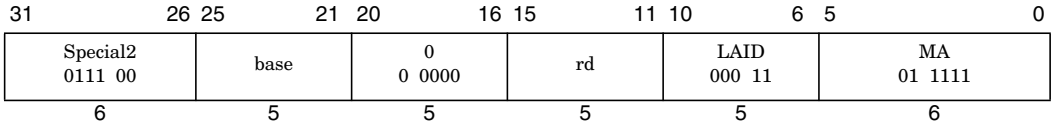
When a more general memory atomic add is needed rather than a increment, LAA can be used rather than LAI. LAI should be used rather than LAA whenever it is statically known that an increment is needed.

LAA/LAI/LAD execution is very similar to SAA, except that the previous value of the memory location is returned in a register. Different instructions can be used in different circumstances:

- SAA should always be used rather than LAA/LAI/LAD if the memory location need not be read near the operation.
- If the memory location needs to be read atomically as part of the operation, then LAA/LAI/LAD must be used rather than SAA.
- If the memory location needs to be read sometime near the operation, but not atomically with the update, then:
  - If the block is likely to be in the L1 cache before the operation, LW+SAA may perform better than LAA/LAI/LAD
  - If the block is not likely to be in the L1 cache before the operation, either LAA/LAI/LAD or SAA+LW or SAA+PREF+LW sequences could perform best, depending on how useful it is to put the block into the L1 cache after the operation, and whether the PREF can hide the cache-miss latency

Cavium Confidential For Bowen Yang 2/27/2014

Load Atomic Increment Doubleword — LAID



Format: LAID rd, (base) **CVMv2**

Purpose:
To load and increment a doubleword of memory atomically.

Description:
```
GPR[rd] ← Mem[GPR[base]];                    // atomic
Mem[GPR[base]] = Mem[GPR[base]] + 1;         // atomic
```

The 64-bit-aligned address specified by register base is incremented after being read. Register rd gets the memory read result.

The memory read, increment, and memory store operations are not interrupted by any other instructions on this or any other processor. No Integer Overflow exception occurs under any circumstance.

TLB accesses, watches, and breakpoints commonly treat these instructions as both a load operation and a store operation. The specifics of the instruction are:

- For address errors, Cause[ExcCode] is never AdEL (it is AdES always).
- For TLB exceptions, this instruction is both a load operation and a store operation. The details are:
  - Cause[ExcCode] is TLBL only when the sole cause of the TLB exception is a set RI bit (it is typically TLBS)
  - A TLB Modified exception is higher priority than a TLB Invalid exception caused solely by a set RI bit.
- For watchpoints, this instruction is both a load operation and a store operation.
- For breakpoints, this instruction is both a load operation and a store operation. The details are:
  - Any of DDBS, DDBL, and DDBLImpr exceptions are possible.
  - A DDBL exception will not occur whenever a DDBS exception could occur instead.
  - Load-value breakpoints check against the actual value returned from the memory load operation (i.e. the old contents of the memory location).
  - Store-value breakpoints match regardless of the DBV values.

LAID generates an INCR64 CMI transaction – refer to Section 3.1.

Restrictions:

The address in the base register must be naturally aligned, else an address error occurs. A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results.

Cavium Confidential For Bowen Yang ALU 2/27/2014

Operation:

```
if !Are64bitOperationsEnabled() then
   SignalException(ReservedInstruction)
endif
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr, vAddr, DATA)
resultdoubleword = memdoubleword + 1
StoreMemory (CCA, DOUBLEWORD, resultdoubleword, pAddr, vAddr, DATA)
GPR[rd] ← memdoubleword
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint, Reserved Instruction.

Programming Notes:

This instruction is performance-optimized for memory locations that are present in the L2 cache, not the L1 data cache. The cache block that contains the memory location is removed from the L1 data cache (if present) during execution of the instruction.

When a more general memory atomic add is needed rather than a increment, LAAD can be used rather than LAID. LAID should be used rather than LAAD whenever it is statically known that an increment is needed.

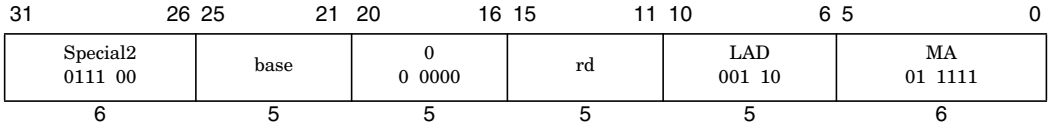
LAAD/LAID/LADD execution is very similar to SAAD, except that the previous value of the memory location is returned in a register. Different instructions can be used in different circumstances:

- SAAD should always be used rather than LAAD/LAID/LADD if the memory location need not be read near the operation.
- If the memory location needs to be read atomically as part of the operation, then LAAD/LAID/LADD must be used rather than SAAD.
- If the memory location needs to be read sometime near the operation, but not atomically with the update, then:
  - If the block is likely to be in the L1 cache before the operation, LD+SAAD may perform better than LAAD/LAID/LADD
  - If the block is not likely to be in the L1 cache before the operation, either LAAD/LAID/LADD or SAAD+LD or SAAD+PREF+LD sequences could perform best, depending on how useful it is to put the block into the L1 cache after the operation, and whether the PREF can hide the cache-miss latency.

Cavium Confidential For Bowen Yang ALD 02/17/2014

A.3.9 Memory Atomic Decrement Instructions

Load Atomic Decrement Word **LAD**



Format: LAD rd, (base) **CVMv2**

Purpose:
To load and decrement a word of memory atomically.

Description:
```
GPR[rd] ← Mem[GPR[base]];                  // atomic
Mem[GPR[base]] = Mem[GPR[base]] – 1;       // atomic
```

The memory at the 32-bit-aligned address specified by register base is decremented after being read. Register rd gets the sign-extended memory read result.

The memory read, decrement, and memory store operations are not interrupted by any other instructions on this or any other processor. No Integer Overflow exception occurs under any circumstance.

TLB accesses, watches, and breakpoints commonly treat these instructions as both a load operation and a store operation. The specifics of the instruction are:

- For address errors, Cause[ExcCode] is never AdEL (it is AdES always).
- For TLB exceptions, this instruction is both a load operation and a store operation. The details are:
  - Cause[ExcCode] is TLBL only when the sole cause of the TLB exception is a set RI bit (it is typically TLBS)
  - A TLB Modified exception is higher priority than a TLB Invalid exception caused solely by a set RI bit.
- For watchpoints, this instruction is both a load operation and a store operation.
- For breakpoints, this instruction is both a load operation and a store operation. The details are:
  - Any of DDBS, DDBL, and DDBLImpr exceptions are possible.
  - A DDBL exception will not occur whenever a DDBS exception could occur instead.
  - Load-value breakpoints check against the actual value returned from the memory load operation (i.e. the old contents of the memory location).
  - Store-value breakpoints match regardless of the DBV byte values.

LAD generates a DECR32 CMI transaction – refer to Section 3.1.

Restrictions:

The address in the base register must be naturally aligned, else an address error occurs.

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results.

Cavium Confidential For Bowei Yang ALU 02/21/2014

Operation:

```
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
amsk2 = 0^61 || CvmCtl[LE] || 0^2
memdoubleword ← LoadMemory (CCA, WORD, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
bit ← 64 - ((vAddr<2> XOR CvmCtl[LE]) || 0^5)
resultdoubleword = memdoubleword + (1^64 << (bit-32))
StoreMemory (CCA, WORD, resultdoubleword, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
GPR[rd] ← sign_extend(memdoubleword<bit-1 .. bit-32>)
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint

Programming Notes:

This instruction is performance-optimized for memory locations that are present in the L2 cache, not the L1 data cache. The cache block that contains the memory location is removed from the L1 data cache (if present) during execution of the instruction.

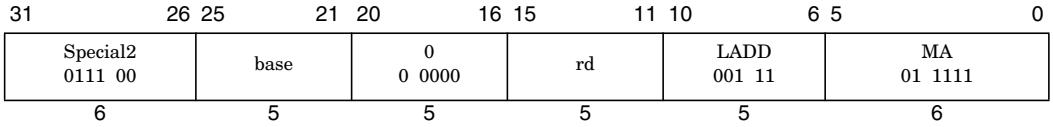
When a more general memory atomic add is needed rather than a increment, LAA can be used rather than LAD. LAD should be used rather than LAA whenever it is statically known that a decrement is needed.

LAA/LAI/LAD execution is very similar to SAA, except that the previous value of the memory location is returned in a register. Different instructions can be used in different circumstances:

- SAA should always be used rather than LAA/LAI/LAD if the memory location need not be read near the operation.
- If the memory location needs to be read atomically as part of the operation, then LAA/LAI/LAD must be used rather than SAA.
- If the memory location needs to be read sometime near the operation, but not atomically with the update, then:
  - If the block is likely to be in the L1 cache before the operation, LW+SAA may perform better than LAA/LAI/LAD
  - If the block is not likely to be in the L1 cache before the operation, either LAA/LAI/LAD or SAA+LW or SAA+PREF+LW sequences could perform best, depending on how useful it is to put the block into the L1 cache after the operation, and whether the PREF can hide the cache-miss latency.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Atomic Decrement Doubleword — LADD



Format: LADD rd, (base) **CVMv2**

Purpose:
To load and decrement a doubleword of memory atomically.

Description:
```
GPR[rd] ← Mem[GPR[base]];                    // atomic
Mem[GPR[base]] = Mem[GPR[base]] – 1;         // atomic
```

The 64-bit aligned address specified by register base is decremented after being read. Register rd gets the memory read result.

The memory read, decrement, and memory store operations are not interrupted by any other instructions on this or any other processor. No Integer Overflow exception occurs under any circumstance.

TLB accesses, watches, and breakpoints commonly treat these instructions as both a load operation and a store operation. The specifics of the instruction are:

- For address errors, Cause[ExcCode] is never AdEL (it is AdES always).
- For TLB exceptions, this instruction is both a load operation and a store operation. The details are:
  - Cause[ExcCode] is TLBL only when the sole cause of the TLB exception is a set RI bit (it is typically TLBS)
  - A TLB Modified exception is higher priority than a TLB Invalid exception caused solely by a set RI bit.
- For watchpoints, this instruction is both a load operation and a store operation.
- For breakpoints, this instruction is both a load operation and a store operation. The details are:
  - Any of DDBS, DDBL, and DDBLImpr exceptions are possible.
  - A DDBL exception will not occur whenever a DDBS exception could occur instead.
  - Load-value breakpoints check against the actual value returned from the memory load operation (i.e. the old contents of the memory location).
  - Store-value breakpoints match regardless of the DBV values.

LADD generates a DECR64 CMI transaction – refer to Section 3.1.

Restrictions:

The address in the base register must be naturally aligned, else an address error occurs.

A Reserved Instruction exception is signaled if access to 64-bit operations is not enabled.

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Operation:

```
if !Are64bitOperationsEnabled() then
   SignalException(ReservedInstruction)
endif
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr, vAddr, DATA)
resultdoubleword = memdoubleword + 1^64
StoreMemory (CCA, DOUBLEWORD, resultdoubleword, pAddr, vAddr, DATA)
GPR[rd] ← memdoubleword
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint, Reserved Instruction.

Programming Notes:

This instruction is performance-optimized for memory locations that are present in the L2 cache, not the L1 data cache. The cache block that contains the memory location is removed from the L1 data cache (if present) during execution of the instruction.

When a more general memory atomic add is needed rather than a increment, LAAD can be used rather than LADD. LADD should be used rather than LAAD whenever it is statically known that a decrement is needed.

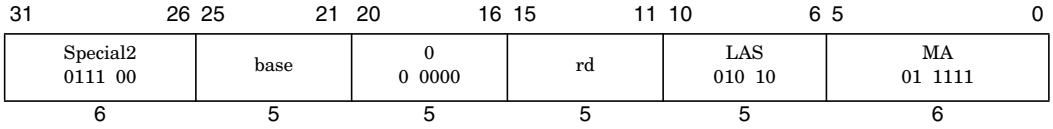
LAAD/LAID/LADD execution is very similar to SAAD, except that the previous value of the memory location is returned in a register. Different instructions can be used in different circumstances:

- SAAD should always be used rather than LAAD/LAID/LADD if the memory location need not be read near the operation.
- If the memory location needs to be read atomically as part of the operation, then LAAD/LAID/LADD must be used rather than SAAD.
- If the memory location needs to be read sometime near the operation, but not atomically with the update, then:
  - If the block is likely to be in the L1 cache before the operation, LD+SAAD may perform better than LAAD/LAID/LADD
  - If the block is not likely to be in the L1 cache before the operation, either LAAD/LAID/LADD or SAAD+LD or SAAD+PREF+LD sequences could perform best, depending on how useful it is to put the block into the L1 cache after the operation, and whether the PREF can hide the cache-miss latency.

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.3.10 Memory Atomic Set Instructions

Load Atomic Set Word **LAS**



Format: LAS rd, (base) **CVMv2**

Purpose:
To load from and write ones to a word of memory atomically.

Description:
```
GPR[rd] ← Mem[GPR[base]];          // atomic
Mem[GPR[base]] = 1s;               // atomic
```

Ones are written to the memory at the 32-bit aligned address specified by register base after the memory location is read. Register rd gets the sign-extended memory read result.

The memory read and memory store operations are not interrupted by any other instructions on this or any other processor.

TLB accesses, watches, and breakpoints commonly treat these instructions as both a load operation and a store operation. The specifics of the instruction are:

- For address errors, Cause[ExcCode] is never AdEL (it is AdES always).
- For TLB exceptions, this instruction is both a load operation and a store operation. The details are:
  - Cause[ExcCode] is TLBL only when the sole cause of the TLB exception is a set RI bit (it is typically TLBS)
  - A TLB Modified exception is higher priority than a TLB Invalid exception caused solely by a set RI bit.
- For watchpoints, this instruction is both a load operation and a store operation.
- For breakpoints, this instruction is both a load operation and a store operation. The details are:
  - Any of DDBS, DDBL, and DDBLImpr exceptions are possible.
  - A DDBL exception will not occur whenever a DDBS exception could occur instead.
  - Load-value breakpoints check against the actual value returned from the memory load operation (i.e. the old contents of the memory location).
  - Store-value breakpoints match when the appropriate DBV bytes are 1s.

LAS generates a SET32 CMI transaction – refer to Section 3.1.

Restrictions:

The address in the base register must be naturally aligned, else an address error occurs.

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results..

Cavium Confidential For Bowei Yang ALU 02/2/2014

Operation:

```
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
amsk2 = 0^61 || CvmCtl[LE] || 0^2
memdoubleword ← LoadMemory (CCA, WORD, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
bit ← 64 - ((vAddr<2> XOR CvmCtl[LE]) || 0^5)
StoreMemory (CCA, WORD, 1^64, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
GPR[rd] ← sign_extend(memdoubleword<bit-1 .. bit-32>)
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint.

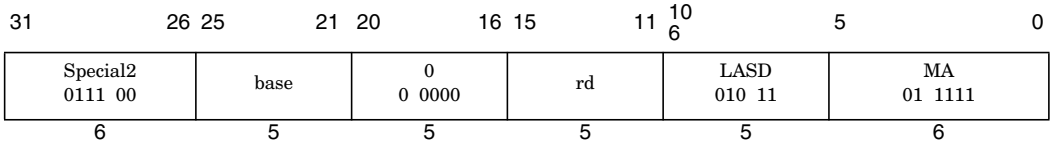
Programming Notes:

This instruction is performance-optimized for memory locations that are present in the L2 cache, not the L1 data cache. The cache block that contains the memory location is removed from the L1 data cache (if present) during execution of the instruction.

When a more general atomic memory write value is needed rather than just 1s, LAW can be used rather than LAS. LAS should be used rather than LAW whenever it is statically known that the write value is all 1s.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Atomic Set Doubleword — LASD



Format: LASD rd, (base) **CVMv2**

Purpose:
To load from and write ones to a doubleword of memory atomically.

Description:
```
GPR[rd] ← Mem[GPR[base]];          // atomic
Mem[GPR[base]] = 1s;               // atomic
```

Ones are written to the memory at the 64-bit aligned address specified by register base after the memory location is read. Register rd gets the memory read result.

The memory read and memory store operations are not interrupted by any other instructions on this or any other processor.

TLB accesses, watches, and breakpoints commonly treat these instructions as both a load operation and a store operation. The specifics of the instruction are:

- For address errors, Cause[ExcCode] is never AdEL (it is AdES always).
- For TLB exceptions, this instruction is both a load operation and a store operation. The details are:
  - Cause[ExcCode] is TLBL only when the sole cause of the TLB exception is a set RI bit (it is typically TLBS)
  - A TLB Modified exception is higher priority than a TLB Invalid exception caused solely by a set RI bit.
- For watchpoints, this instruction is both a load operation and a store operation.
- For breakpoints, this instruction is both a load operation and a store operation. The details are:
  - Any of DDBS, DDBL, and DDBLImpr exceptions are possible.
  - A DDBL exception will not occur whenever a DDBS exception could occur instead.
  - Load-value breakpoints check against the actual value returned from the memory load operation (i.e. the old contents of the memory location).
  - Store-value breakpoints match when the appropriate DBV bytes are 1s.

LASD generates a SET64 CMI transaction – refer to Section 3.1.

Restrictions:

The address in the base register must be naturally aligned, else an address error occurs.

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results..

Cavium Confidential For Bowen Yang ALU 02/7/2014

Operation:

```
if !Are64bitOperationsEnabled() then
   SignalException(ReservedInstruction)
endif
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr, vAddr, DATA)
StoreMemory (CCA, DOUBLEWORD, 1^64, pAddr, vAddr, DATA)
GPR[rd] ← memdoubleword
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint, Reserved Instruction.

Programming Notes:

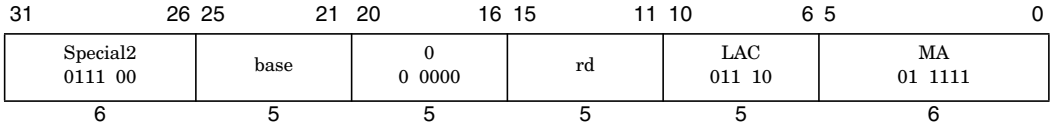
This instruction is performance-optimized for memory locations that are present in the L2 cache, not the L1 data cache. The cache block that contains the memory location is removed from the L1 data cache (if present) during execution of the instruction.

When a more general atomic memory write value is needed rather than just 1s, LAWD can be used rather than LASD. LASD should be used rather than LAWD whenever it is statically known that the write value is all 1s.

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.3.11 Memory Atomic Clear Instructions

Load Atomic Clear Word **LAC**



Format: LAC rd, (base) **CVMv2**

Purpose:
To load from and write 0s to a word of memory atomically.

Description:
```
GPR[rd] ← Mem[GPR[base]];          // atomic
Mem[GPR[base]] = 0s;               // atomic
```

Zeroes are written to the memory at the 32-bit aligned address specified by register base after the memory location is read. Register rd gets the sign-extended memory read result.

The memory read and memory store operations are not interrupted by any other instructions on this or any other processor.

TLB accesses, watches, and breakpoints commonly treat these instructions as both a load operation and a store operation. The specifics of the instruction are:

- For address errors, Cause[ExcCode] is never AdEL (it is AdES always).
- For TLB exceptions, this instruction is both a load operation and a store operation. The details are:
  - Cause[ExcCode] is TLBL only when the sole cause of the TLB exception is a set RI bit (it is typically TLBS)
  - A TLB Modified exception is higher priority than a TLB Invalid exception caused solely by a set RI bit.
- For watchpoints, this instruction is both a load operation and a store operation.
- For breakpoints, this instruction is both a load operation and a store operation. The details are:
  - Any of DDBS, DDBL, and DDBLImpr exceptions are possible.
  - A DDBL exception will not occur whenever a DDBS exception could occur instead.
  - Load-value breakpoints check against the actual value returned from the memory load operation (i.e. the old contents of the memory location).
  - Store-value breakpoints match when the appropriate DBV bytes are 0s.

LAC generates a CLR32 CMI transaction – refer to Section 3.1.

Restrictions:

The address in the base register must be naturally aligned, else an address error occurs.

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results.

Cavium Confidential For Bowei Yang ALU 02/2/2014

Operation:

```
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
amsk2 = 0^61 || CvmCtl[LE] || 0^2
memdoubleword ← LoadMemory (CCA, WORD, pAddr XOR amsk2, vAddr XOR amsk2, DATA)
bit ← 64 - ((vAddr<2> XOR CvmCtl[LE]) || 0^5)
StoreMemory (CCA, WORD, 0^64, pAddr XOR amsk2, vAddr XOR amsk
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint

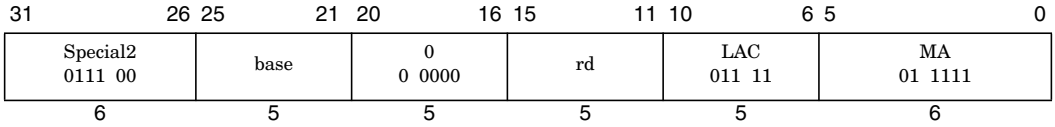
Programming Notes:

This instruction is performance-optimized for memory locations that are present in the L2 cache, not the L1 data cache. The cache block that contains the memory location is removed from the L1 data cache (if present) during execution of the instruction.

When a more general atomic memory write value is needed rather than just ones, LAW can be used rather than LAC. LAC should be used rather than LAW whenever it is statically known that the write value is 0x0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Atomic Clear Doubleword LACD



Format: LACD rd, (base) **CVMv2**

Purpose:
To load from and write 0s to a doubleword of memory atomically.

Description:
```
GPR[rd] ← Mem[GPR[base]];          // atomic
Mem[GPR[base]] = 0s;               // atomic
```

Zeroes are written to the memory at the 64-bit aligned address specified by register base after the memory location is read. Register rd gets the memory read result.

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

The memory read and store are not interrupted by any other instructions on this or any other processor.

TLB accesses, watches, and breakpoints commonly treat these instructions as both a load operation and a store operation. The specifics of the instruction are:

- For address errors, Cause[ExcCode] is never AdEL (it is AdES always).
- For TLB exceptions, this instruction is both a load operation and a store operation. The details are:
  - Cause[ExcCode] is TLBL only when the sole cause of the TLB exception is a set RI bit (it is typically TLBS)
  - A TLB Modified exception is higher priority than a TLB Invalid exception caused solely by a set RI bit.
- For watchpoints, this instruction is both a load operation and a store operation.
- For breakpoints, this instruction is both a load operation and a store operation. The details are:
  - Any of DDBS, DDBL, and DDBLImpr exceptions are possible.
  - A DDBL exception will not occur whenever a DDBS exception could occur instead.
  - Load-value breakpoints check against the actual value returned from the memory load operation (i.e. the old contents of the memory location).
  - Store-value breakpoints match when the appropriate DBV bytes are 0s.

LACD generates a CLR64 CMI transaction – refer to Section 3.1.

Restrictions:

The address in the base register must be naturally aligned, else an address error occurs.

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results.

Cavium Confidential For Bowen Yang ALD 01/27/2014

Operation:

```
if !Are64bitOperationsEnabled() then
   SignalException(ReservedInstruction)
endif
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
memdoubleword ← LoadMemory (CCA, DOUBLEWORD, pAddr, vAddr, DATA)
StoreMemory (CCA, DOUBLEWORD, 0^64, pAddr, vAddr, DATA)
GPR[rd] ← memdoubleword
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint, Reserved Instruction

Programming Notes:

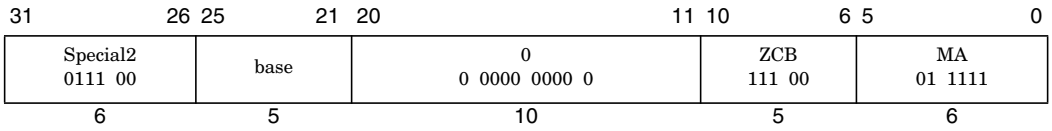
This instruction is performance-optimized for memory locations that are present in the L2 cache, not the L1 data cache. The cache block that contains the memory location is removed from the L1 data cache (if present) during execution of the instruction.

When a more general atomic memory write value is needed rather than just ones, LAWD can be used rather than LACD. LACD should be used rather than LAWD whenever it is statically known that the write value is 0x0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.3.12 Zero Cache Block Instructions

Zero Cache Block **ZCB**



Format: ZCB (base) **CVMv2**

Purpose:
To zero all the bytes in a (128-byte) cache block and allocate the block in the L2 cache.

Description: `zero_cache_block(base)`

base is an effective byte address. ZCB zeroes the surrounding naturally aligned 128-byte cache block.

TLB accesses, watches, and breakpoints treat ZCB as a store of 0 to every byte in the cache block.

ZCB generates a STFIL1 CMI transaction – refer to Section 3.1.

Restrictions:

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results.

Operation:

```
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
zero_cache_block(pAddr)
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint

Programming Notes:

PREF 30 is similar to ZCB, but ZCB takes TLB exceptions and guarantees that the cache block is 0, unlike PREF 30. PREF 30 is typically followed by store operations that write the cache block to known values, but ZCB need not be.

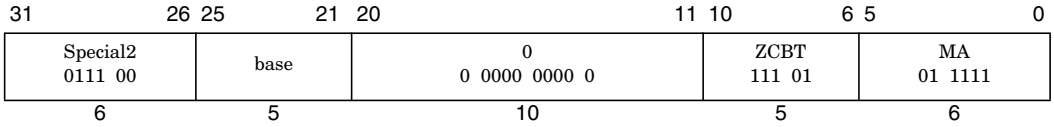
The cache block is invalid in the L1 cache after ZCB execution.

ZCB allocates the block into the L2 cache. ZCBT should be used instead of ZCB when the cache block should not be allocated in the L2 cache.

In many circumstances, ZCB/ZCBT instructions should be used rather than PREF 30/31. PREF 30/31 have offset addressing and will leave the block in the L1 cache after the instruction if it were already present in the cache.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Zero Cache Block Through ZCBT



Format: ZCBT (base) **CVMv2**

Purpose:
To zero all the bytes in a (128-byte) cache block without allocating the block in any cache.

Description: `zero_cache_block_through(base)`

base is an effective byte address. ZCBT zeroes the surrounding naturally aligned 128-byte cache block.

TLB accesses, watches, and breakpoints treat ZCBT as a store of 0 to every byte in the cache block.

ZCBT generates a STTIL1 CMI transaction – refer to Section 3.1.

Restrictions:

The address must refer to an L2/DRAM location. Any other address may produce unpredictable results.

Operation:

```
vAddr ← GPR[base]
(pAddr, CCA) ← AddressTranslation (vAddr, DATA, LOAD)
if (vAddr/pAddr is not L2/DRAM) then
   unpredictable
else if (vAddr/pAddr has an address error) then
   SignalException(AddressError)
endif
zero_cache_block_through(pAddr)
```

Exceptions:

TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Watch, Breakpoint

Programming Notes:

PREF 31 is similar to ZCBT, but ZCBT takes TLB exceptions and guarantees that the cache block is 0, unlike PREF 31. PREF 31 is typically followed by store operations that write the cache block to known values, but ZCBT need not be.

The cache block is invalid in the L1 cache after ZCBT execution.

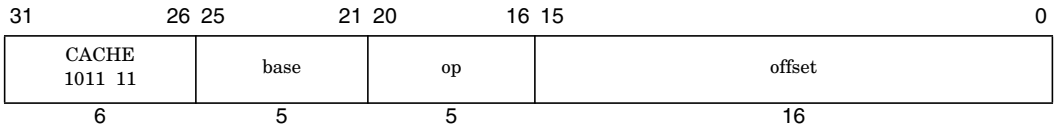
ZCBT does not allocate the block into the L2 cache. ZCB should be used instead of ZCBT when the cache block should be allocated in the L2 cache.

In many circumstances, ZCB/ZCBT instructions should be used rather than PREF 30/31. PREF 30/31 have offset addressing and will leave the block in the L1 cache after the instruction if it were already present in the cache.

Cavium Confidential For Bowen Yang ALU 03/27/2014

A.4 CPU Move and Privileged Instructions

Perform Cache Operation **CACHE**



Format: CACHE op, offset(base) **MIPS32/CVM**

Purpose:

To perform the cache operation specified by op.

Description:

The operation depends on the value of the op field. The possible values are shown in the table below.

NOTE: This is a standard MIPS32 instruction for which CN71XX chips implement all the functionality defined in the MIPS standard instruction set, as well as additional, cnMIPS III-specific operations.

Op Value	Name	Description
0, 8, 16, 20	L1 Icache Invalidate	Clears the valid bit for all blocks in the entire L1 Icache.
1, 9	L1 Dcache Invalidate	Clears the valid bit for all blocks in the entire L1 Dcache.
2, 6, 10, 13–15, 18, 22, 24–26, 28–30		No operation.
3	L2 Cache Index Writeback Invalidate	The 16-bit signed offset is added to the contents of the base register to form an effective address. • Bits<18:17> select the way of the cache block involved in the operation. • Bits<16:7> of the effective address select the index. These bits are not aliased during the execution of the instruction, so may differ from <19:7> of addresses that map to the index. (Refer to Section 3.2.2.) Section 3.2.3If the VALID and DIRTY bits are set for the L2 cache block at the specified index, hardware writes the block back to the memory address. After that operation is completed, the VALID, USED, and LOCK bits for the block are cleared. Refer to Section 3.2.2, L2 Cache Indexing (Set Selection). CACHE 3 generates a WBIL2I CMI transaction – refer to Section 3.1. A SYNCW should be used prior to the CACHE 3 if updates to the affected cache block that need to be processed prior to the WBIL2I may be present in the write buffer of the core when the CACHE 3 issues. Also note that the CACHE 3 has no effect on the L1 data or instruction cache contents in any core. It may also be necessary to invalidate the L1 data and/or instruction cache(s) to remove all copies of a cache block from all caches.

Cavium Confidential For Bowen Yang ALU 08/27/2014

Op Value	Name	Description
4	L1 Icache Index Load Tag	The 16-bit signed offset is added to the contents of the base register to form an effective address. • Bits<16:11> select the way of the cache block involved in the operation. • Bits<10:7> of the effective address select the set. • Bits<6:3> of the effective address select a 64-bit word within the cache block. • Bits<2:0> must be zero. The hardware loads the virtual tag for the Icache block into the Icache TagLo register and 65 bits of Icache data into the Icache DataLo and DataHi registers. If the instruction cache is perfect (i.e. if no repair is required), then: • bits<63:0> of the Icache data contain the two instructions in the word in big-endian format; • bit<64> is the parity bit for the instructions; This operation does not cause a cache error exception.
5	L1 Dcache Index Load Tag	The 16-bit signed offset is added to the contents of the base register to form an effective address. • Bits<14:10> of the effective address select the particular cache block involved in the operation (i.e. the way) • Bits<9:7> select the set. • Bits<6:3> selects a 64-bit word within the cache block. The hardware loads the virtual and physical tags for the Dcache block into the Dcache TagLo, TagLo1, and TagHi registers and the 64-bit data word and corresponding parity bits into the Dcache DataLo and DataHi registers. This operation does not cause a cache error exception.
7	L2 Cache Index Load Tag	The 16-bit signed offset is added to the contents of the base register to form an effective address. • Bits<18:17> select the way of the cache block involved in the operation. • Bits<16:7> of the effective address select the index. These bits are not aliased during the execution of the instruction, so may differ from <19:7> of addresses that map to the index. (Refer to Section 3.2.2.) Section 3.2.3The hardware loads the tag for the L2 cache block into the L2C_TAD*t*_TAG register. If the tag ECC (i.e. L2C_TAD*t*_TAG[ECC]) is required to be valid after execution of the L2 Cache Index Load Tag instruction, then L2C_CTL[DISECC] must be set prior to the execution of the instruction. (An L2C_CTL read prior to the L2 Cache Index Load Tag instruction ensures that L2C uses the L2C_CTL[DISECC] value most recently written.) A SYNC/SYNCS instruction after the L2 cache index-load tag CACHE instruction ensures L2C_TAD*t*_TAG has been updated by the CACHE instruction. Refer to Section 3.2.2, L2 Cache Indexing (Set Selection). CACHE 7 generates a LTGL2I CMI transaction – refer to Section 3.1. A SYNCW should be used prior to the CACHE 7 if updates to the affected cache block that need to be processed prior to the LTGL2I may be present in the write buffer of the core when the CACHE 7 issues.

Cavium Confidential For Bowen Yang 08/12/2024

Op Value	Name	Description
11	L2 Cache Index Store Tag	The 16-bit signed offset is added to the contents of the base register to form an effective address. • Bits<18:17> select the way of the cache block involved in the operation. • Bits<16:7> of the effective address select the index. These bits are not aliased during the execution of the instruction, so may differ from <19:7> of addresses that map to the index. (Refer to Section 3.2.2.) Section 3.2.3The hardware fills the virtual tag for the L2 cache block with the contents of the L2C_TAD*t*_TAG register. This operation does not cause a cache error exception. An L2C_TAD*t*_TAG read prior to the L2 cache index-store tag CACHE instruction ensures that the CACHE instruction uses the L2C_TAD*t*_TAG value most recently written. A SYNC/SYNCS instruction after the CACHE instruction ensures that the CACHE instruction executes before a subsequent L2C_TAD*t*_TAG write. Refer to Section 3.2.2, L2 Cache Indexing (Set Selection). CACHE 11 generates a STGL2I CMI transaction – refer to Section 3.1. A SYNCW should be used prior to the CACHE 11 if updates to the affected cache block that need to be processed prior to the STGL2I may be present in the write buffer of the core when the CACHE 11 issues. Also note that the CACHE 11 has no effect on the L1 data or instruction cache contents in any core.
12	L1 Icache Bitmap Store	The 16-bit signed offset is added to the contents of the base register to form an effective address. • Bits<16:11> selects the way of the cache block involved in the operation. • Bits<10:7> of the effective address selects the set. • Bits<6:5> of the effective address selects a 260-bit word within the cache block. (Bits <4:3> are not used.) • Bits<2:0> must be zero. The hardware fills the virtual tag for the Icache block with the contents of the Icache TagLo register and fills 260 bits of data into the selected Icache word. The 260 bits written are 33 duplicates of DataLo<1:0>, so only values 0x00000000000000000, 0x15555555555555555, 0x2AAAAAAAAAAAAAAAA, and 0x3FFFFFFFFFFFFFFFF are possible. If the instruction cache is perfect, then: • bits<63:0> of the 260 bits written correspond to the two instructions of the word in big-endian format; • bit<64> is the parity bit.
17, 21	L1 Dcache Virtual Tag Invalidate	All valid bits for all virtual tags in the entire L1 Dcache are cleared. The valid bits for the physical tags in the Dcache remain unchanged.
19	L2 Cache Hit Invalidate	The 16-bit signed offset is added to the contents of the base register to form an effective address. The VALID, USED, and LOCK bits are cleared for the L2 cache block at the specified address. Dirty data is lost. The value of the bytes in the selected cache block are unpredictable after this operation and may change value unpredictably until they are later stored to. Bits <41:7> of the effective (virtual) address is the physical address of the selected cache block, and must reside in a DR0 DRAM or DR1 DRAM range (Section 2.6 describes DR0 and DR1 DRAM ranges). Cavium recommends that the effective address is always an XKPHYS virtual address. Note specifically that KSEG0 virtual addresses should not be used. CACHE 19 generates an INVL2 CMI transaction – refer to Section 3.1. Also note that the CACHE 19 has no effect on the L1 data or instruction cache contents in any core. It may also be necessary to invalidate the L1 data and/or instruction cache(s) to remove all copies of a cache block from all caches.

Cavium Confidential For Buwen Yang ALU 02/27/2014

Op Value	Name	Description
23	L2 Cache Hit Writeback Invalidate	The 16-bit signed offset is added to the contents of the base register to form an effective address. If the VALID and DIRTY bits are set for the L2 cache block at the specified address, hardware writes the block back to the memory address. After that operation is completed, the VALID, USED, and LOCK bits for the block are cleared. Bits <41:7> of the effective (virtual) address is the physical address of the selected cache block, and must reside in a DR0 DRAM or DR1 DRAM range (Section 2.6 describes DR0 and DR1 DRAM ranges). Cavium recommends that the effective address is always an XKPHYS virtual address. Note specifically that KSEG0 virtual addresses should not be used. CACHE 23 generates a WBIL2 CMI transaction – refer to Section 3.1. Also note that the CACHE 23 has no effect on the L1 data or instruction cache contents in any core. It may also be necessary to invalidate the L1 data and/or instruction cache(s) to remove all copies of a cache block from all caches.
27	L2 Cache Hit Writeback	The 16-bit signed offset is added to the contents of the base register to form an effective address. If the VALID and DIRTY bits are set for the L2 cache block at the specified address, hardware writes the block back to the memory address (clearing the DIRTY and USED bits, with no effect on the LOCK bit for the block). The VALID bit is not cleared for the block, but may be evicted soon. Bits <41:7> of the effective (virtual) address is the physical address of the selected cache block, and must reside in a DR0 DRAM or DR1 DRAM range (Section 2.6 describes DR0 and DR1 DRAM ranges). Cavium recommends that the effective address is always an XKPHYS virtual address. Note specifically that KSEG0 virtual addresses should not be used. CACHE 27 generates a WBL2 CMI transaction – refer to Section 3.1. **NOTE:** The operation of CACHE 27 is similar to the operation of PREF 25.
31	L2 Cache Fetch and Lock	The 16-bit signed offset is added to the contents of the base register to form an effective address. If the L2 cache does not contain the specified address and there is an available cache way, hardware fills the block from memory into the L2 cache, performing a writeback if required, and sets the VALID, LOCK, and USE bits. If the L2 cache does not contain the specified address and there are no available ways, this instruction is a NOP. If the L2 cache already contains the specified address, hardware sets the LOCK and USE bits for the block. Bits <41:7> of the effective (virtual) address is the physical address of the selected cache block, and must reside in a DR0 DRAM or DR1 DRAM range (Section 2.6 describes DR0 and DR1 DRAM ranges). Cavium recommends that the effective address is always an XKPHYS virtual address. Note specifically that KSEG0 virtual addresses should not be used. CACHE 27 generates a LCKL2 CMI transaction – refer to Section 3.1. L2C sets L2C_TAD*_INT[NOWAY] when it turns a CACHE 31 into a NOP. Software can use L2C_TAD*_INT[NOWAY] to determine whether the CACHE 31 succeeded or failed in controlled circumstances where there are no other L2C transactions that could possibly set L2C_TAD*_INT[NOWAY]. In general conditions, Cavium recommends that CACHE 7 instructions instead be used to confirm whether a CACHE 31 instruction succeeded.

Restrictions:

If access to Coprocessor 0 is not enabled, a Coprocessor Unusable Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(0) then
  vAddr = GPR[base] + sign_extend(offset)
  CacheOp(op, vAddr)
else
  SignalException(CoprocessorUnusable, 0)
endif
```

Exceptions:

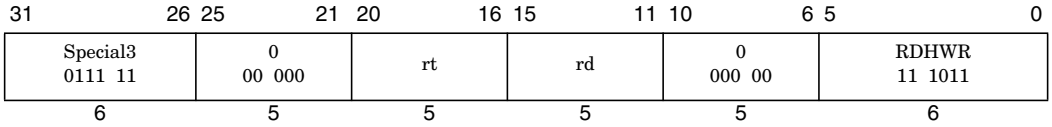
Coprocessor Unusable Exception

Cavium Confidential For Broadcom Use Only 02/10/14

CACHE instructions never match watches/breakpoints on the CN71XX.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Read Hardware Register — RDHWR



Format: RDHWR rt, rd **MIPS32 Release 2/CVM**

Purpose:

To move the contents of a hardware register to a general purpose register (GPR) if that operation is enabled by privileged software.

Description: rt= HWR[rd]

If access is allowed to the specified hardware register, the contents of the register specified by rd is sign-extended and loaded into general register rt. Access control for each register is selected by the bits in the coprocessor 0 HWREna register.

The available hardware registers, and the encoding of the rd field for each, is as shown in the table below.

NOTE: This is a standard MIPS32 instruction for which CN71XX chips implement all the functionality defined in the MIPS standard instruction set, as well as additional, cnMIPS III-specific operations. The register numbers 30 and 31 are cnMIPS III-specific. <31:30> are present in the HWREna register on CN71XX chips.

Register Number (rd Value)	Register Name	Contents
0	CPUNum	Number of the Core on which the program is currently running. This comes directly from the coprocessor 0 EBaseCPUNum field.
1	SYNCI_Step	Address step size to be used with the SYNCI instruction. See that instruction's description for the use of this value. A SYNCI flushes the entire instruction cache on the CN71XX, so CN71XX's SYNCI_Step is 4096.
2	CC	High-resolution cycle counter. This comes directly from the coprocessor 0 Count register.
3	CCRes	Resolution of the CC register. This value denotes the number of cycles between update of the register. This is one for the CN71XX.
29	ULR	User Local Register. This register provides read access to the coprocessor 0 UserLocal register.
30	ChOrd	The current value of the pending tag switch bit local to this core: 1 = the local bit indicates that there is no pending SSO tag switch 0 = the local bit indicates that there is a pending SSO tag switch See Section 9.3 for a discussion of the pending tag switch bit that is local to each core. Note that the local pending switch bit is NOT the actual SSO state, but should match in normal circumstances.
31	CvmCount	The current value of the 64-bit Cavium-specific coprocessor 0 CvmCount register. Note that the full 64-bit CvmCount value is provided when 64-bit operations are enabled, and only the lower 32 bits of CvmCount are provided (sign-extended) when 64-bit operations are not enabled.
All others		Access results in a Reserved Instruction Exception

Cavium Confidential For Bowen Yang ALU 06/27/2014

Restrictions:

In implementations of Release 1 of the Architecture, this instruction resulted in a Reserved Instruction Exception.

Operation:

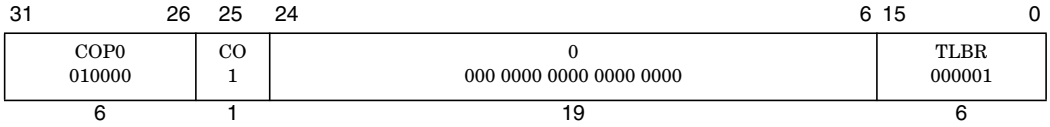
```
if ((HWREna(rd) = 1) or IsCoprocessorEnabled(0)) then
    case rd
      16#00: temp = EBase[CPUNum]
      16#01: temp = 4096
      16#02: temp = sign_extend(Count)
      16#03: temp = 1
      16#1d: temp = UserLocal
      16#1e: temp = local pending switch bit
      16#1f: if Are64bitOperationsEnabled() then
               temp = CvmCount<63:0>
             else
               temp = sign_extend(CvmCount<31:0>)
     otherwise: SignalException(ReservedInstruction)
   endcase
   GPR[rt] = temp
 else
   SignalException(ReservedInstruction)
 endif
```

Exceptions:

Reserved Instruction

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Read a µTLB Entry — TLBR



Format: TLBR **CVMv3**

Purpose:

To read an entry from the µTLB when CvmMemCtl2[UTLBRW] = 1. When CvmMemCtl2[UTLBRW] = 0, TLBR acts as the normal MIPS-specified TLBR instruction.

Description:
The UtlbEntry0, UtlbEntry1, and UtlbEntry2 registers are loaded with the contents of the µTLB entry pointed to the UtlbIndex register.

Restrictions:

The opreation is UNDEFINED if the index specified in the UtlbIndex register is greater than or equal to the number of µTLB entries initialized in the processor.

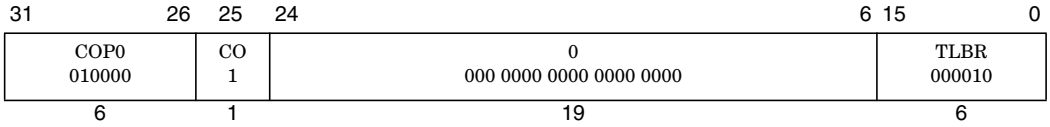
Operation:

```
if IsCoprocessorEnabled(0) then
  i = Index into µTLB from UtlbIndex register
  if i > CvMemCtl2[UTLBNUMEN] then
   UNDEFINED
  else
    {UtlbEntry0, UtlbEntry1, UtlbEntry} ← µTLB[i]
  endif
else
  SignalException(CoprocessorUnusable, 0)
endif
```

Exceptions:

Cavium Confidential For Bowen Yang ALU 02/27/2014

Write a μTLB Entry — TLBWI



Format: TLBWI **CVMv3**

Purpose:

To write an entry into the μTLB when CvmMemCtl2[UTLBRW] = 1. When CvmMemCtl2[UTLBRW] = 0, TLBWI acts as the normal MIPS-specified TLBWI instruction.

Description:
The contents of UtlbEntry0, UtlbEntry1, and UtlbEntry2 registers are written to the μTLB entry pointed to by the UtlbIndex register.

Restrictions:

The operation is UNDEFINED if the index specified in the UtlbIndex register is greater than or equal to the number of μTLB entries initialized in the processor.

Processor execution is UNDEFINED if TLBWI writes a μTLB entry that does not exist in the TLB.

Operation:

```
if IsCoprocessorEnabled(0) then
  i = Index into μTLB from UtlbIndex register
  if i > CvMemCtl2[UTLBNUMEN] then
    UNDEFINED
  else
     μTLB[i]←{UtlbEntry0, UtlbEntry1, UtlbEntry}
  endif
else
  SignalException(CoprocessorUnusable, 0)
endif
```

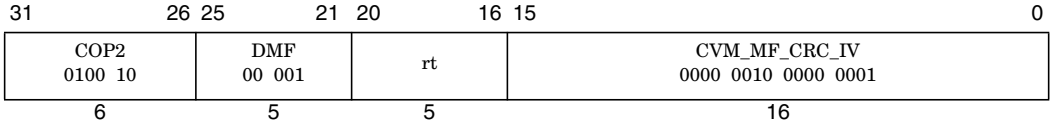
Exceptions:

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.5 Coprocessor Move Instructions

A.5.1 CRC Instructions

Load IV from CRC Unit **CVM_MF_CRC_IV**



Format: DMFC2 rt, 0x0201 **CVM**

Purpose:

To read CRCIV from the CRC coprocessor.

Description: rt = sign_extend(CRCIV<31:0>)

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) then
    SignalException(ReservedInstruction)
  else
    GPR[rt] = sign_extend(CRCIV<31:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

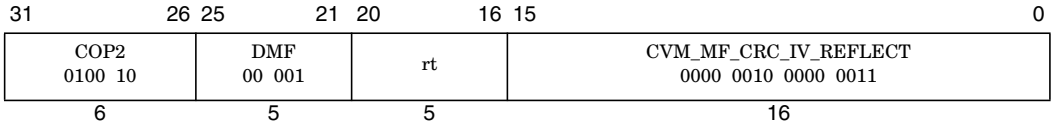
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load IV from CRC Unit Reflected — CVM_MF_CRC_IV_REFLECT



Format: DMFC2 rt, 0x0203 **CVM**

Purpose:

To read CRCIV from the CRC coprocessor. The bits within the entire CRCIV word are reflected.

Description: rt = sign_extend(word_bit_reflect(CRCIV<31:0>))

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) then
    SignalException(ReservedInstruction)
  else
    GPR[rt] = sign_extend(word_bit_reflect(CRCIV<31:0>))
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

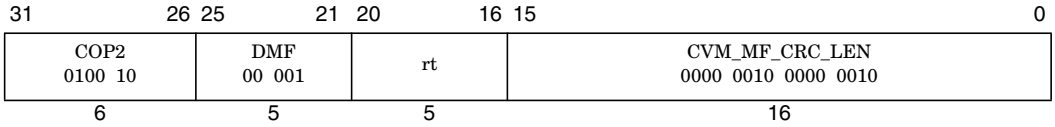
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Length from CRC Unit — CVM_MF_CRC_LEN



Format: DMFC2 rt, 0x0202 **CVM**

Purpose:

To read CRCLEN from the CRC coprocessor.

Description: rt = CRCLEN<3:0>

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) then
    SignalException(ReservedInstruction)
  else
    GPR[rt] = 0^60 || CRCLEN<3:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

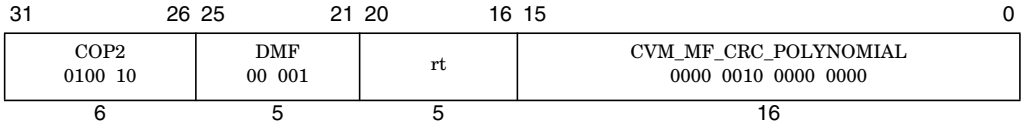
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Polynomial from CRC Unit — CVM_MF_CRC_POLYNOMIAL



Format: DMFC2 rt, 0x0200 **CVM**

Purpose:

To read CRCPOLY from the CRC coprocessor.

Description: rt = sign_extend(CRCPOLY<31:0>)

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) then
    SignalException(ReservedInstruction)
  else
    GPR[rt] = sign_extend(CRCPOLY<31:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

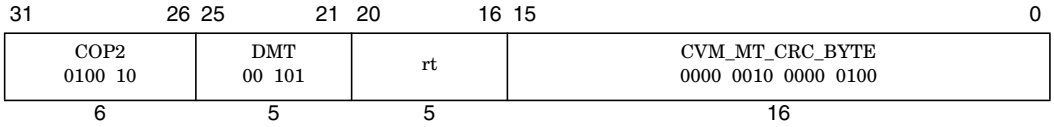
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

CRC for a Byte — CVM_MT_CRC_BYTE



Format: DMTC2 rt, 0x0204 **CVM**

Purpose:

To calculate CRC for a byte. The least-significant byte of rt is the input to the CRC engine (using CRCIV and CRCPOLY). CRCIV is updated with the CRC result for the given CRCPOLY.

Description: CRCIV<31:0> = CRC(rt<7:0>, CRCIV<31:0>, CRCPOLY<31:0>)

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled())) then
    SignalException(ReservedInstruction)
  else
    CRCIV<31:0> = CRC_BYTE(GPR[rt]<7:0>, CRCIV<31:0>, CRCPOLY<31:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

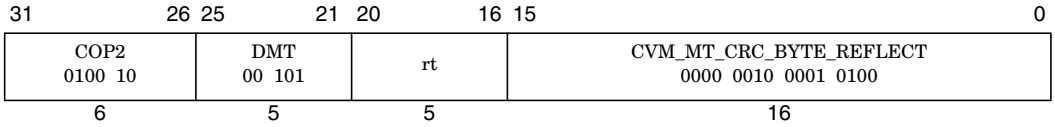
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

CRC for a Byte Reflected — CVM_MT_CRC_BYTE_REFLECT



Format: DMTC2 rt, 0x0214 **CVM**

Purpose:

To calculate CRC for a byte. The bits in each byte are reflected. The least-significant byte of rt is the input to the CRC engine (using CRCIV and CRCPOLY) after bit reflection. CRCIV is updated with the CRC result for the given CRCPOLY.

Description: CRCIV<31:0> = CRC(byte_bit_reflect(rt<7:0>), CRCIV<31:0>, CRCPOLY<31:0>)

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled())) then
    SignalException(ReservedInstruction)
  else
    CRCIV<31:0> = CRC_BYTE(byte_bit_reflect(GPR[rt]<7:0>),
                                        CRCIV<31:0>, CRCPOLY<31:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

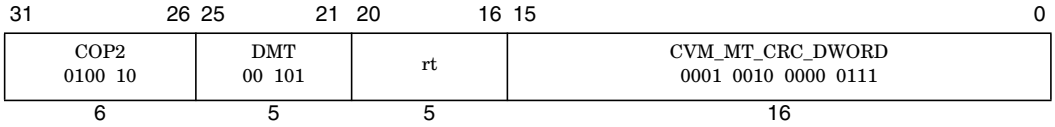
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

CRC for a Double-word — CVM_MT_CRC_DWORD



Format: DMTC2 rt, 0x1207 **CVM**

Purpose:

To calculate CRC for a double-word. The eight bytes of rt is the input to the CRC engine (using CRCIV and CRCPOLY). CRCIV is updated with the CRC result for the given CRCPOLY.

Description: CRCIV<31:0> = CRC(rt<63:0>, CRCIV<31:0>, CRCPOLY<31:0>)

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled())) then
    SignalException(ReservedInstruction)
  else
    for(i = 0; i < 8; i++)
      CRCIV<31:0> = CRC_BYTE(GPR[rt]<63-i*8:56-i*8>, CRCIV<31:0>, CRCPOLY<31:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

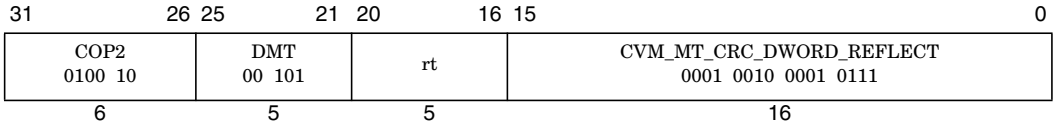
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

CRC for a Double-word Reflected — CVM_MT_CRC_DWORD_REFLECT



Format: DMTC2 rt, 0x1217 **CVM**

Purpose:

To calculate CRC for a double-word. The bits in each byte are reflected. The eight bytes of rt is the input to the CRC engine (using CRCIV and CRCPOLY) after bit reflection. CRCIV is updated with the CRC result for the given CRCPOLY.

Description: CRCIV<31:0> = CRC(byte_bit_reflect(rt<63:0>), CRCIV<31:0>, CRCPOLY<31:0>)

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
 if IsCoprocessorEnabled(2) then
    if (not Are64bitOperationsEnabled())) then
       SignalException(ReservedInstruction)
    else
       for(i = 0; i < 8; i++)
          CRCIV<31:0> = CRC_BYTE(byte_bit_reflect(GPR[rt]<63-i*8:56-i*8>),
CRCIV<31:0>, CRCPOLY<31:0>)
    endif
 else
    SignalException(CoprocessorUnusable, 2)
 endif
```

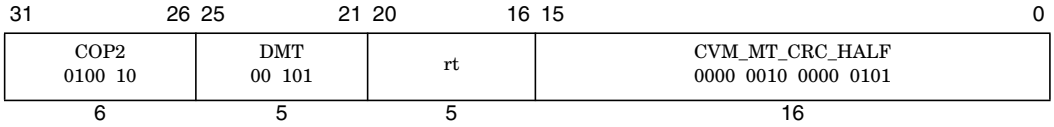
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

CRC for a Halfword — CVM_MT_CRC_HALF



Format: DMTC2 rt, 0x0205 **CVM**

Purpose:

To calculate CRC for a halfword. The least-significant two bytes of rt is the input to the CRC engine (using CRCIV and CRCPOLY). CRCIV is updated with the CRC result for the given CRCPOLY.

Description: CRCIV<31:0> = CRC(rt<15:0>, CRCIV<31:0>, CRCPOLY<31:0>)

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled())) then
    SignalException(ReservedInstruction)
  else
    for(i = 0; i < 2; i++)
      CRCIV<31:0> = CRC_BYTE(GPR[rt]<15-i*8:8-i*8>, CRCIV<31:0>, CRCPOLY<31:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

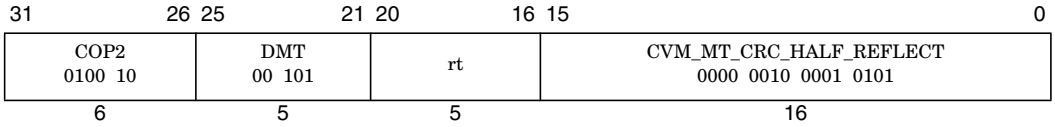
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

CRC for a Halfword Reflected — CVM_MT_CRC_HALF_REFLECT



Format: DMTC2 rt, 0x0215 **CVM**

Purpose:

To calculate CRC for a halfword. The bits in each byte are reflected. The least-significant two bytes of rt is the input to the CRC engine (using CRCIV and CRCPOLY) after bit reflection. CRCIV is updated with the CRC result for the given CRCPOLY.

Description: CRCIV<31:0> = CRC(byte_bit_reflect(rt<15:0>), CRCIV<31:0>, CRCPOLY<31:0>)

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled())) then
    SignalException(ReservedInstruction)
  else
    for(i = 0; i < 2; i++)
      CRCIV<31:0> = CRC_BYTE(byte_bit_reflect(GPR[rt]<15-i*8:8-i*8>), CRCIV<31:0>, CRCPOLY<31:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

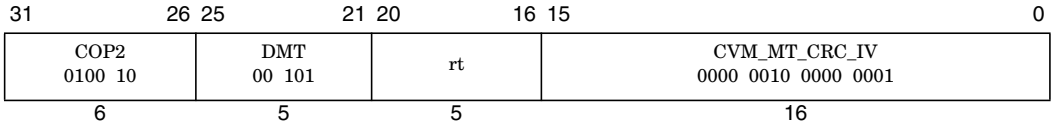
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load IV into CRC Unit — CVM_MT_CRC_IV



Format: DMTC2 rt, 0x0201 **CVM**

Purpose:

To load a value into CRCIV.

Description: CRCIV<31:0> = rt<31:0>

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled())) then
    SignalException(ReservedInstruction)
  else
    CRCIV<31:0> = GPR[rt]<31:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

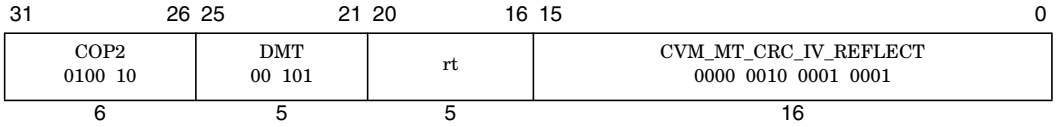
Reserved Instruction

Programming Notes:

CRC polynomials smaller than 32-bits should be supported by placing the IV in the most-significant CRCIV bits. For example, the IV with a 16-bit polynomial should reside in CRCIV<31:16>, and CRCIV<15:0> should be zero.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load IV into CRC Unit Reflected — CVM_MT_CRC_IV_REFLECT



Format: DMTC2 rt, 0x0211 **CVM**

Purpose:

To load a value into CRCIV. The bits in each byte are reflected.

Description: CRCIV<31:0> = byte_bit_reflect(rt<31:0>)

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled())) then
    SignalException(ReservedInstruction)
  else
    CRCIV<31:0> = byte_bit_reflect(GPR[rt]<31:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

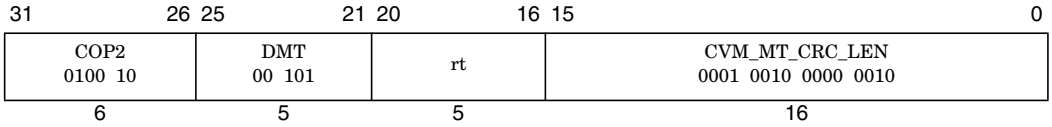
Reserved Instruction

Programming Notes:

CRC polynomials smaller than 32-bits should be supported by placing the IV in the most-significant CRCIV bits. For example, the IV with a 16-bit polynomial should reside in CRCIV<31:16>, and CRCIV<15:0> should be zero.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Length into CRC Unit — CVM_MT_CRC_LEN



Format: DMTC2 rt, 0x1202 **CVM**

Purpose:

To load a value into CRCLEN.

Description: CRCLEN<3:0> = rt<3:0>

Restrictions:

The results of CVM_MT_CRC_LEN are unpredictable when GPR[rt]<63:4> is not zero.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) then
    SignalException(ReservedInstruction)
  else
    CRCLEN<3:0> = GPR[rt]<3:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

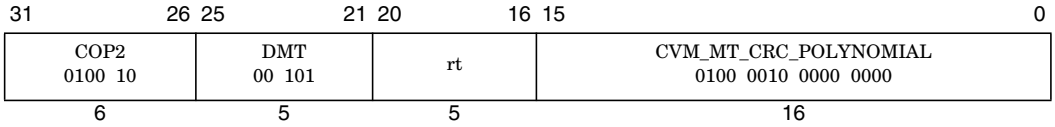
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Polynomial into CRC Unit — CVM_MT_CRC_POLYNOMIAL



Format: DMTC2 rt, 0x4200 **CVM**

Purpose:

To load a value into CRCPOLY.

Description: CRCPOLY<31:0> = rt<31:0>

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled())) then
    SignalException(ReservedInstruction)
  else
    CRCPOLY<31:0> = GPR[rt]<31:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

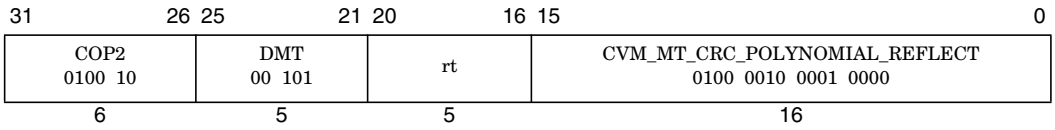
Reserved Instruction

Programming Notes:

CRC polynomials smaller than 32-bits should be supported by placing the polynomial in the most-significant CRCPOLY bits. For example, a 16-bit polynomial should reside in CRCPOLY<31:16>, and CRCPOLY<15:0> should be zero.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Polynomial into CRC Unit Reflected **CVM_MT_CRC_POLYNOMIAL_REFLECT**



Format: DMTC2 rt, 0x4210 **CVM**

Purpose:

To load a value into CRCPOLY. The bits in each byte are reflected.

Description: CRCPOLY<31:0> = byte_bit_reflect(rt<31:0>)

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled())) then
    SignalException(ReservedInstruction)
  else
    CRCPOLY<31:0> = byte_bit_reflect(GPR[rt]<31:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

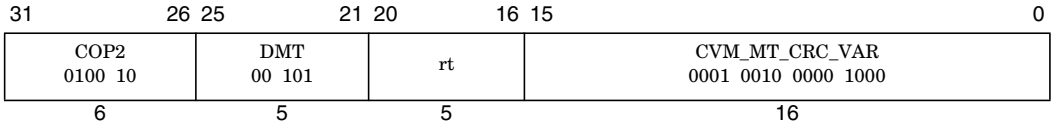
Reserved Instruction

Programming Notes:

CRC polynomials smaller than 32-bits should be supported by placing the polynomial in the most-significant CRCPOLY bits. For example, a 16-bit polynomial should reside in CRCPOLY<31:16>, and CRCPOLY<15:0> should be zero.

Cavium Confidential For Bowen Yang ALU 02/27/2014

CRC for Variable Length — CVM_MT_CRC_VAR



Format: DMTC2 rt, 0x1208 **CVM**

Purpose:

To calculate CRC for CRCLEN bytes. The CRCLEN most-significant bytes of rt is the input to the CRC engine (using CRCIV and CRCPOLY). CRCIV is updated with the CRC result for the given CRCPOLY.

Description: CRCIV<31:0> = CRC(rt<63:0>, CRCLEN<3:0>, CRCIV<31:0>, CRCPOLY<31:0>)

Restrictions:

The results of CVM_MT_CRC_VAR are unpredictable when CRCLEN<3:0> is larger than eight.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) then
    SignalException(ReservedInstruction)
  else
    len = CRCLEN<3:0>;
    for(i = 0; i < len; i++)
      CRCIV<31:0> = CRC_BYTE(GPR[rt]<63-i*8:56-i*8>, CRCIV<31:0>, CRCPOLY<31:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

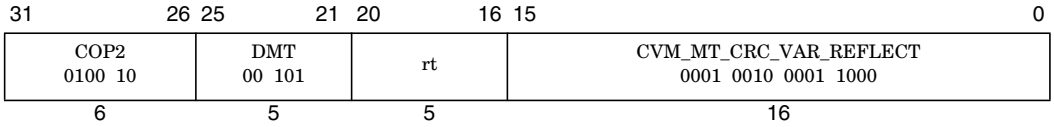
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

CRC for Variable Length Reflected **CVM_MT_CRC_VAR_REFLECT**



Format: DMTC2 rt, 0x1218 **CVM**

Purpose:

To calculate CRC for length bytes. The bits in each byte are reflected. The CRCLEN most-significant bytes of rt is the input to the CRC engine (using CRCIV and CRCPOLY) after bit reflection. CRCIV is updated with the CRC result for the given CRCPOLY.

Description: CRCIV<31:0> = CRC(byte_bit_reflect(rt<63:0>), CRCLEN<3:0>, CRCIV<31:0>, CRCPOLY<31:0>)

Restrictions:

The results of CVM_MT_CRC_VAR_REFLECT are unpredictable when CRCLEN<3:0> is larger than eight.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled())) then
    SignalException(ReservedInstruction)
  else
    len = CRCLEN<3:0>;
    for(i = 0; i < len; i++)
      CRCIV<31:0> = CRC_BYTE(byte_bit_reflect(GPR[rt]<63-i*8:56-i*8>), CRCIV<31:0>, CRCPOLY<31:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

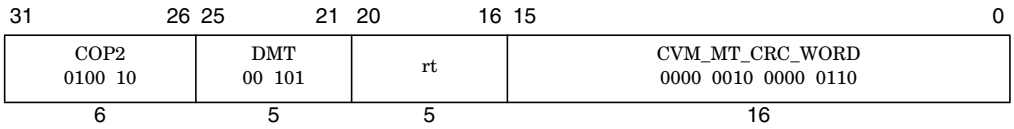
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

CRC for a Word — CVM_MT_CRC_WORD



Format: DMTC2 rt, 0x0206 **CVM**

Purpose:

To calculate CRC for a word. The least-significant four bytes of rt is the input to the CRC engine (using CRCIV and CRCPOLY). CRCIV is updated with the CRC result for the given CRCPOLY.

Description: CRCIV<31:0> = CRC(rt<31:0>, CRCIV<31:0>, CRCPOLY<31:0>)

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled())) then
    SignalException(ReservedInstruction)
  else
    for(i = 0; i < 4; i++)
      CRCIV<31:0> = CRC_BYTE(GPR[rt]<31-i*8:24-i*8>, CRCIV<31:0>, CRCPOLY<31:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

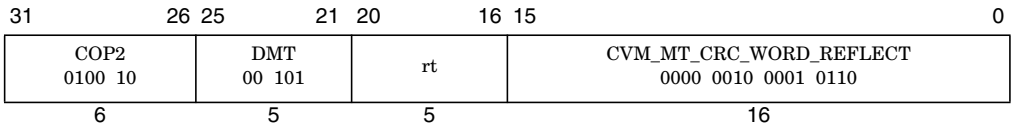
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

CRC for a Word Reflected — CVM_MT_CRC_WORD_REFLECT



Format: DMTC2 rt, 0x0216 **CVM**

Purpose:

To calculate CRC for a word. The bits in each byte are reflected. The least-significant four bytes of rt is the input to the CRC engine (using CRCIV and CRCPOLY) after bit reflection. CRCIV is updated with the CRC result for the given CRCPOLY.

Description: CRCIV<31:0> = CRC(byte_bit_reflect(rt<31:0>), CRCIV<31:0>, CRCPOLY<31:0>)

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled())) then
    SignalException(ReservedInstruction)
  else
    for(i = 0; i < 4; i++)
      CRCIV<31:0> = CRC_BYTE(byte_bit_reflect(GPR[rt]<31-i*8:24-i*8>), CRCIV<31:0>, CRCPOLY<31:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

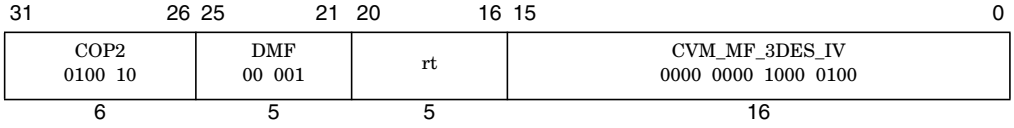
Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

A.5.2 Security Accelerator Instructions

3DES

Load IV from 3DES Unit **CVM_MF_3DES_IV**



Format: DMFC2 rt, 0x0084 **CVM**

Purpose:

To read 3DESIV from the 3DES coprocessor.

Description: rt = 3DESIV<63:0>

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CVMCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    GPR[rt] = 3DESIV<63:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
 endif
```

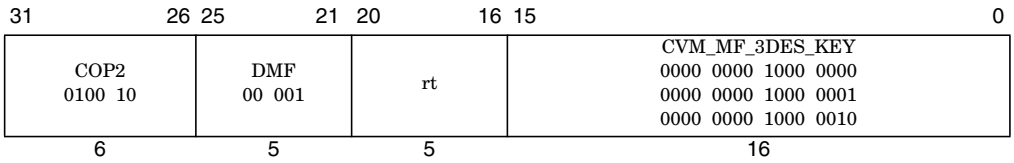
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Key from 3DES Unit — CVM_MF_3DES_KEY



Note: These instructions are the same as Load Key from KASUMI Unit instructions (CVM_MF_KAS_KEY), except that 3DESKEY[2:0] are used by the 3DES unit and only 3DESKEY[1:0] are used by the KASUMI unit.

Format: DMFC2 rt, 0x0080 **CVM**

DMFC2 rt, 0x0081

DMFC2 rt, 0x0082

Purpose:

To read 3DESKEY values from the 3DES coprocessor. 3DESKEY[2:0] are used by the 3DES unit.

Description:

```
rt = 3DESKEY[0]  // DMFC2 rt, 0x0080
rt = 3DESKEY[1]  // DMFC2 rt, 0x0081
rt = 3DESKEY[2]  // DMFC2 rt, 0x0082
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0080: GPR[rt] = 3DES_KEY[0]
    case 0x0081: GPR[rt] = 3DES_KEY[1]
    case 0x0082: GPR[rt] = 3DES_KEY[2]
  endif
else
  SignalException(CoprocessorUnusable, 2)
 endif
```

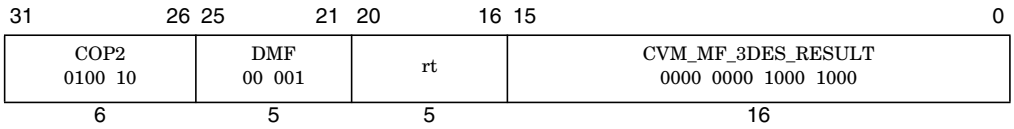
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Result from 3DES Unit — CVM_MF_3DES_RESULT



Note: This instruction is the same as Load Result from KASUMI Unit (CVM_MF_KAS_RESULT).

Format: DMFC2 rt, 0x0088 **CVM**

Purpose:

To read 3DESRESULT from the 3DES coprocessor.

Description: rt = 3DESRESULT<63:0>

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
    if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
      SignalException(ReservedInstruction)
    else
      GPR[rt] = 3DESRESULT<63:0>
    endif
  else
    SignalException(CoprocessorUnusable, 2) endif
endif
```

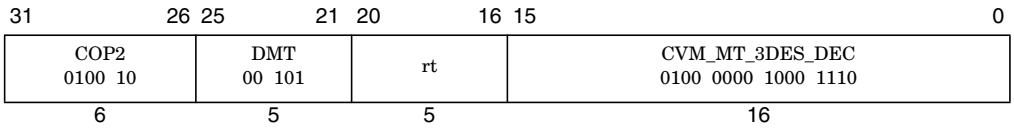
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

3DES Decrypt **CVM_MT_3DES_DEC**



Format: DMTC2 rt, 0x408E **CVM**

Purpose:

To do a 3DES EDE decrypt. GPR[rt] is the input to a 3DES EDE decrypt (using 3DESKEY). 3DESRESULT is set to the result of the decrypt, and 3DESIV is unpredictable.

Description: 3DESRESULT<63:0> = 3DES_EDE_DEC(rt, 3DESKEY[2:0])

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    3DESRESULT<63:0> = 3DES_EDE_DEC(GPR[rt], 3DESKEY[2:0])
    3DESIV<63:0> = unpredictable
    KASUMIRESULT<63:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

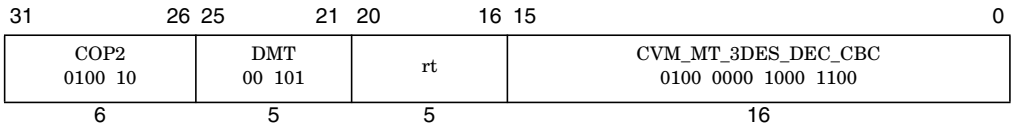
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

3DES CBC Decrypt — CVM_MT_3DES_DEC_CBC



Format: DMTC2 rt, 0x408C **CVM**

Purpose:

To do a 3DES EDE CBC decrypt. GPR[rt] is the input to a 3DES EDE decrypt (using 3DESKEY). 3DESRESULT is set to the result of the decrypt XOR 3DESIV, and 3DESIV is set to GPR[rt].

Description:
```
3DESRESULT<63:0> = 3DES_EDE_DEC(rt, 3DESKEY[2:0]) ⊕ 3DESIV<63:0>
3DESIV<63:0> = rt
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    3DESRESULT<63:0> = 3DES_EDE_DEC(GPR[rt], 3DESKEY[2..0]) ⊕ 3DESIV<63:0>
    3DESIV<63:0> = GPR[rt]
    KASUMIRESULT<63:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

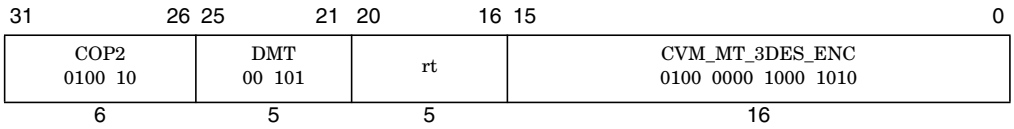
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALD 02/27/2014

3DES Encrypt — CVM_MT_3DES_ENC



Format: DMTC2 rt, 0x408A **CVM**

Purpose:

To do a 3DES EDE encrypt. GPR[rt] is the input to a 3DES EDE encrypt (using 3DESKEY). 3DESRESULT is set to the result of the encrypt and 3DESIV is unpredictable.

Description: 3DESRESULT<63:0> = 3DES_EDE_ENC(rt, 3DESKEY[2:0])

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    3DESRESULT<63:0> = 3DES_EDE_ENC(GPR[rt], 3DESKEY[2:0])
    3DESIV<63:0> = unpredictable
    KASUMIRESULT<63:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

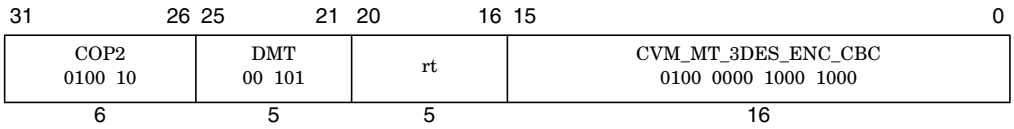
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

3DES CBC Encrypt — CVM_MT_3DES_ENC_CBC



Format: Format: DMTC2 rt, 0x4088 **CVM**

Purpose:

To do a 3DES EDE CBC encrypt. The XOR of GPR[rt] and 3DESIV is the input to a 3DES EDE encrypt (using 3DESKEY). 3DESRESULT and 3DESIV are set to the result of the encrypt.

Description: 3DESRESULT<63:0> = 3DESIV<63:0> =
3DES_EDE_ENC(rt ⊕ 3DESIV<63:0>, 3DESKEY[2:0])

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    3DESRESULT<63:0> = 3DESIV<63:0> = 3DES_EDE_ENC(GPR[rt] ⊕ 3DESIV<63:0>, 3DESKEY[2:0])
    KASUMIRESULT<63:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

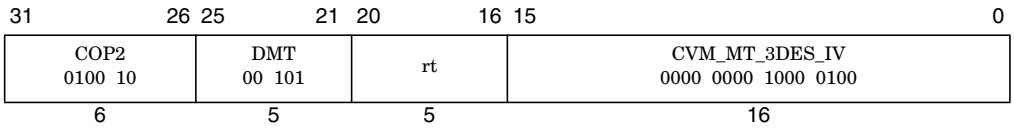
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2015

Load IV into 3DES Unit — CVM_MT_3DES_IV



Format: DMTC2, rt, 0x0084 **CVM**

Purpose:

To load a value into 3DESIV.

Description: 3DESIV<63:0> = rt

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
    if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
      SignalException(ReservedInstruction)
    else
      3DESIV<63:0> = GPR[rt]
    endif
 else
    SignalException(CoprocessorUnusable, 2)
 endif
```

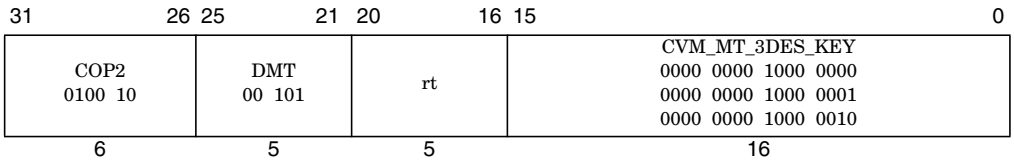
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Key into 3DES Unit — CVM_MT_3DES_KEY



Note: These instructions are the same as Load Key into KASUMI Unit instructions (CVM_MT_KAS_KEY), except that 3DESKEY[2:0] are used by the 3DES unit and only 3DESKEY[1:0] are used by the KASUMI unit.

Format: DMTC2 rt, 0x0080 **CVM**

DMTC2 rt, 0x0081

DMTC2 rt, 0x0082

Purpose:

To load values into 3DESKEY. 3DESKEY[2:0] are used by the 3DES unit.

Description:

```
3DESKEY[0]  = rt // DMTC2 rt,0x0080
3DESKEY[1]  = rt // DMTC2 rt,0x0081
3DESKEY[2]  = rt // DMTC2 rt,0x0082
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0080: 3DESKEY[0] = GPR[rt]
    case 0x0081: 3DESKEY[1] = GPR[rt]
    case 0x0082: 3DESKEY[2] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

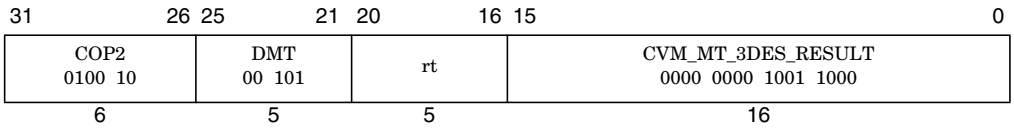
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Result into 3DES Unit — CVM_MT_3DES_RESULT



Note: This instruction is the same as Load Result into KASUMI Unit (CVM_MT_KAS_RESULT).

Format: DMTC2 rt, 0x0098 **CVM**

Purpose:

To load a value into 3DESRESULT.

Description: 3DESRESULT<63:0> = rt

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    3DESRESULT<63:0> = GPR[rt]
    KASUMIRESULT<63:0> = 3DESRESULT<63:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

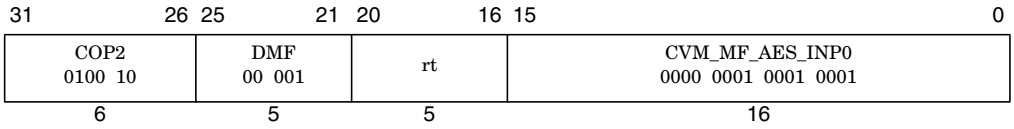
Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

AES

Load INP0 from AES Unit — CVM_MF_AES_INP0



Format: DMFC2 rt, 0x0111 **CVM**

Purpose:

To read AES input from the AES coprocessor.

Description: rt = AESRESINP[0]

CVM_MF_AES_RESINP0 should be used instead of this instruction. This operation is deprecated, and is present only for backward compatibility with earlier revisions of the architecture.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

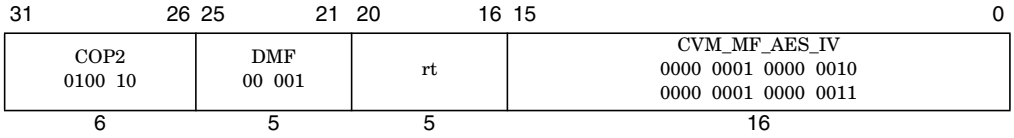
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    GPR[rt] = AESRESINP[0]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load IV from AES Unit — CVM_MF_AES_IV



Note: These instructions are the same as Load IV from the SMS4 Unit instructions (CVM_MF_SMS4_IV 0–1).

Format: DMFC2 rt, 0x0102 **CVM**

DMFC2 rt, 0x0103

Purpose:

To read AESIV from the AES coprocessor.

Description:
```
rt = AESIV[0]   // DMFC2 rt, 0x0102
rt = AESIV[1]   // DMFC2 rt, 0x0103
```

Restrictions:

CVM_MF_AES_IV1 (i.e. “DMFC2 rt, 0x0103”) must not be the first AES-related instruction that immediately follows a prior CVM_MT_AES_ENC_CBC1, CVM_MT_AES_DEC_CBC1, CVM_MT_SMS4_ENC_CBC1, or CVM_MT_SMS4_DEC_CBC1 instruction. CVM_MF_AES_IV1 should always follow CVM_MF_AES_IV0.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

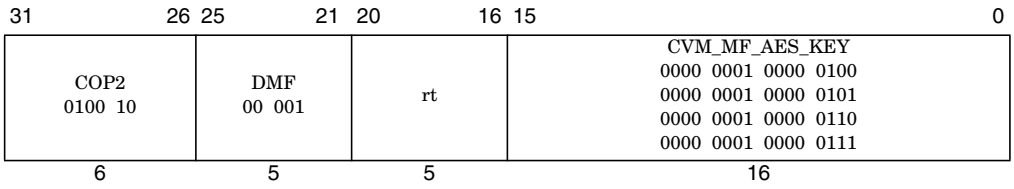
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0102: GPR[rt] = AESIV[0]
    case 0x0103: GPR[rt] = AESIV[1]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Key from AES Unit — CVM_MF_AES_KEY



Note: The instructions 0–1 are the same as Load Key from the SMS4 Unit instructions (CVM_MF_SMS4_KEY 0–1).

Format: DMFC2 rt, 0x0104 **CVM**

DMFC2 rt, 0x0105

DMFC2 rt, 0x0106

DMFC2 rt, 0x0107

Purpose:

To read AESKEY from the AES coprocessor.

Description:

```
rt = AESKEY[0]  // DMFC2 rt, 0x0104
rt = AESKEY[1]  // DMFC2 rt, 0x0105
rt = AESKEY[2]  // DMFC2 rt, 0x0106
rt = AESKEY[3]  // DMFC2 rt, 0x0107
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

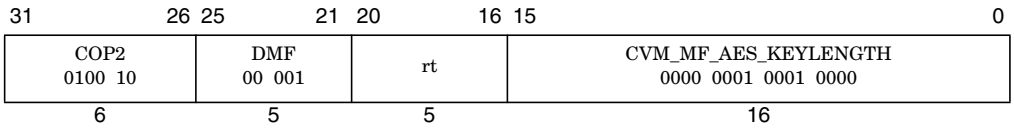
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0104: GPR[rt] = AESKEY[0]
    case 0x0105: GPR[rt] = AESKEY[1]
    case 0x0106: GPR[rt] = AESKEY[2]
    case 0x0107: GPR[rt] = AESKEY[3]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Keylength from AES Unit — CVM_MF_AES_KEYLENGTH



Format: DMFC2 rt, 0x0110 **CVM**

Purpose:

To read AESKEYLEN from the AES coprocessor.

Description: rt = AESKEYLEN<1:0>

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

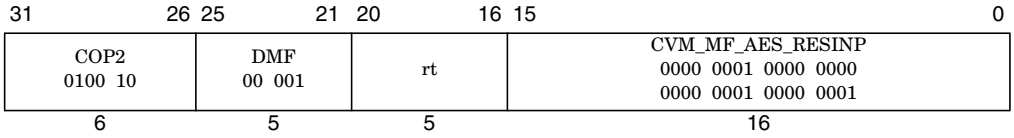
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    GPR[rt] = 0^62 || AESKEYLEN<1:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Result/Input from AES Unit — CVM_MF_AES_RESINP



Note: These instructions are the same as Load Result/Input from the SMS4 or CAMELLIA Unit instructions (CVM_MF_SMS4_RESINP 0–1 and CVM_MF_CAMELLIA_RESINP 0–1, respectively).

Format: DMFC2 rt, 0x0100 **CVM**

DMFC2 rt, 0x0101

Purpose:

To read AESRESINP from the AES coprocessor.

Description:

```
rt = AESRESINP[0]  // DMFC2 rt, 0x0100
rt = AESRESINP[1]  // DMFC2 rt, 0x0101
```

Restrictions:

CVM_MF_AES_RESINP1 (i.e. “DMFC2 rt, 0x0101”) must not be the first AES-related instruction that immediately follows a prior CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC1, CVM_MT_AES_DEC_CBC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC1, or CVM_MT_SMS4_DEC_CBC1 instruction. CVM_MF_AES_RESINP1 should always follow CVM_MF_AES_RESINP0.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

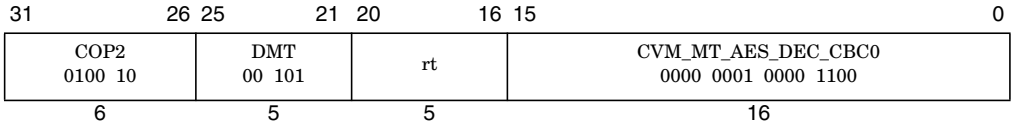
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0100: GPR[rt] = AESRESINP[0]
    case 0x0101: GPR[rt] = AESRESINP[1]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ASU 02/11/2014

AES CBC Decrypt (part 1) CVM_MT_AES_DEC_CBC0



Note: This instruction is the same as SMS4 instruction (CVM_MF_SMS4_DEC_CBC0).

Format: DMTC2 rt, 0x010C **CVM**

Purpose:

To load AESRESINP[0] in preparation for a subsequent CVM_MT_AES_DEC_CBC1.

Description: AESRESINP[0] = rt

Restrictions:

AESIV[0] must be predictable prior to the CVM_MT_AES_DEC_CBC0 instruction and the AES unit cannot use an IV produced by the SMS4 or CAMELLIA unit. This means that AESIV[0] must be written between execution of this CVM_MT_AES_DEC_CBC0 and the execution of the immediately preceding CVM_MT_AES_ENC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

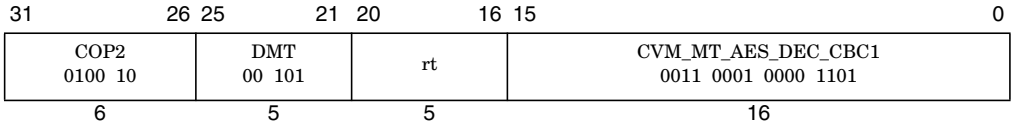
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    AESRESINP[0] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

AES CBC Decrypt (part 2) **CVM_MT_AES_DEC_CBC1**



Format: DMTC2 rt, 0x310D **CVM**

Purpose:

To perform an AES CBC decrypt operation. GPR[rt] and AESRESINP[0] are the inputs to an AES decryption operation (using AESKEY and AESKEYLEN). AESRESINP is updated with the result of the decryption XORed with AESIV. AESIV is set to the inputs of the decryption. Unused AESKEY locations become unpredictable.

Description:
```
TIN[1:0] = rt || AESRESINP[0]
AESRESINP[1:0] = AES_DEC(TIN, AESKEY[3:0], AESKEYLEN<1:0>) ⊕ AESIV[1:0]
AESIV[1:0] = TIN[1:0]
```

Restrictions:

CVM_MT_AES_DEC_CBC0 (or CVM_MT_AES_RESINP0 in the save/restore case) must be executed between the execution of this CVM_MT_AES_DEC_CBC1 and the immediately preceding CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1 instruction.

AESIV[1:0] must be predictable for AES prior to this CVM_MT_AES_DEC_CBC1 instruction and a CVM_MT_AES_DEC_CBC1 cannot use an IV produced by the SMS4 unit. This means that AESIV[1:0] must be written (with CVM_MT_AES_IV* instructions) between execution of this CVM_MT_AES_DEC_CBC1 and the execution of the immediately preceding CVM_MT_AES_ENC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1.

AESKEY[AESKEYLEN<1:0>:0] must be loaded (with CVM_MT_AES_KEY* instructions) between this CVM_MT_AES_DEC_CBC1 and any prior CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1 instruction.

AESKEY[AESKEYLEN<1:0>:0] must be predictable prior to this CVM_MT_AES_DEC_CBC1 instruction. This means that when AESKEYLEN<1:0> increases, AESKEY[AESKEYLEN<1:0>:X] must be written (with CVM_MT_AES_KEY* instructions) between the execution of this CVM_MT_AES_DEC_CBC1 and any immediately preceding CVM_MT_AES_DEC_CBC1 or CVM_MT_AES_DEC1 instruction that used the smaller AESKEYLEN<1:0> value of X.

Results are unpredictable when AESKEYLEN<1:0> is zero with the CVM_MT_AES_DEC_CBC1 instruction.

Cavium Confidential For Boren Yang 2012-7-27 11:1

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    TIN[1:0]<63:0> = GPR[rt] || AESRESINP[0]
    AESRESINP[1:0] = AES_DEC(TIN[1:0], AESKEY[3:0], AESKEYLEN<1:0>) ⊕ AESIV[1:0]
    AESIV[1:0] = TIN[1:0]
    for(i = AESKEYLEN<1:0>+1; i < 4; i++)
      AESKEY[i] = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

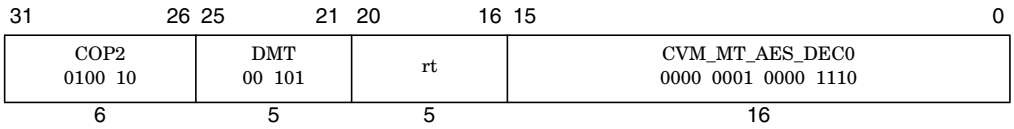
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

AES Decrypt (part 1) **CVM_MT_AES_DEC0**



Note: This instruction is the same as SMS4 instruction (CVM_MT_SMS4_DEC0).

Format: DMTC2 rt, 0x010E **CVM**

Purpose:

To load AESRESINP[0] in preparation for a subsequent CVM_MT_AES_DEC1.

Description: AESRESINP[0] = rt

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    AESRESINP[0] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

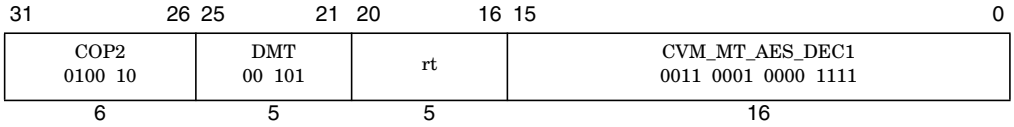
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

AES Decrypt (part 2) **CVM_MT_AES_DEC1**



Format: DMTC2 rt, 0x310F **CVM**

Purpose:

To perform an AES decrypt operation. GPR[rt] and AESRESINP[0] are the inputs to an AES decryption operation (using AESKEY and AESKEYLEN). AESRESINP is updated with the result of the decryption. AESIV and unused AESKEY locations become unpredictable.

Description: AESRESINP[1:0] = AES_DEC((rt || AESRESINP[0]), AESKEY[3:0], AESKEYLEN<1:0>)

Restrictions:

CVM_MT_AES_DEC0 (or CVM_MT_AES_RESINP0 in the save/restore case) must be executed between the execution of this CVM_MT_AES_DEC1 and the immediately preceding CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1 instruction.

AESKEY[AESKEYLEN<1:0>:0] must be loaded (with CVM_MT_AES_KEY* instructions) between this CVM_MT_AES_DEC1 and any prior CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1 instruction.

AESKEY[AESKEYLEN<1:0>:0] must be predictable prior to this CVM_MT_AES_DEC1 instruction. This means that when AESKEYLEN<1:0> increases, AESKEY[AESKEYLEN<1:0>:X] must be written (with CVM_MT_AES_KEY* instructions) between the execution of this CVM_MT_AES_DEC1 and any immediately preceding CVM_MT_AES_DEC_CBC1 or CVM_MT_AES_DEC1 instruction that used the smaller AESKEYLEN<1:0> value of X.

Results are unpredictable when AESKEYLEN<1:0> is zero with the CVM_MT_AES_DEC1 instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Cavium Confidential For Broadcom Use Only 02/17/2012

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
   AESRESINP[1..0] = AES_DEC((GPR[rt] || AESRESINP[0]), AESKEY[3..0], AESKEYLEN<1:0>)
    AESIV[1..0] = unpredictable
    for(i = AESKEYLEN<1:0>+1; i < 4; i++)
      AESKEY[i] = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

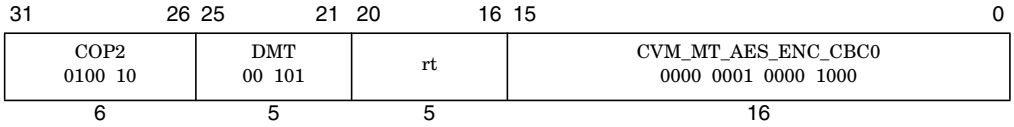
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

AES CBC Encrypt (part 1) CVM_MT_AES_ENC_CBC0



Note: This instruction is the same as SMS4 instruction (CVM_MT_SMS4_ENC_CBC0).

Format: DMTC2 rt, 0x0108 **CVM**

Purpose:

To load AESRESINP[0] in preparation for a subsequent CVM_MT_AES_ENC_CBC1 instruction. The implementation may either deposit (rt XOR AESIV[0]) or rt into AESRESINP[0].

Description: AESRESINP[0] = rt ⊕ AESIV[0], or AESRESINP[0] = rt

Restrictions:

AESIV[0] must be predictable prior to the CVM_MT_AES_ENC_CBC0 instruction and the AES unit cannot use an IV produced by the SMS4 unit. This means that AESIV[0] must be written between execution of this CVM_MT_AES_ENC_CBC0 and the execution of the immediately preceding CVM_MT_AES_ENC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

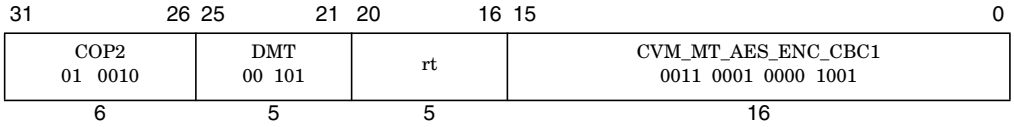
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    AESRESINP[0] = GPR[rt] ⊕ AESIV[0], or
    AESRESINP[0] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yuan ALU 02/27/2014

AES CBC Encrypt (part 2) CVM_MT_AES_ENC_CBC1



Format: DMTC2 rt, 0x3109 **CVM**

Purpose:

To perform an AES CBC encrypt operation. GPR[rt] XOR AESIV[1] is the second 64-bit input to an AES CBC encryption operation (using AESKEY and AESKEYLEN). Either AESRESINP[0] or (AESRESINP[0] XOR AESIV[0]) is the first 64-bit input to the AES operation, depending on whether the CVM_MT_AES_ENC_CBC0 implementation pre-XORed AESIV[0] or not, respectively. AESRESINP and AESIV are updated with the result. Unused AESKEY locations become unpredictable.

Description: AESRESINP[1:0] = AESIV[1..0] =
AES_ENC((rt ⊕ AESIV[1]) || AESRESINP[0], AESKEY[3:0], AESKEYLEN<1:0>), or
AESRESINP[1:0] = AESIV[1..0] =
AES_ENC((rt || AESRESINP[0]) ⊕ AESIV[1:0], AESKEY[3..0], AESKEYLEN<1:0>)

Restrictions:

CVM_MT_AES_ENC_CBC0 (or CVM_MT_AES_RESINP0 in the save/restore case) must be executed between the execution of this CVM_MT_AES_ENC_CBC1 and the immediately preceding CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1 instruction.

AESIV[1:0] must be predictable prior to this CVM_MT_AES_ENC_CBC1 instruction and a CVM_MT_AES_ENC_CBC1 cannot use an IV created by the SMS4 unit. This means that AESIV[1:0] must be written (with CVM_MT_AES_IV* instructions) between execution of this CVM_MT_AES_ENC_CBC1 and the execution of the immediately preceding CVM_MT_AES_ENC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1.

AESKEY[AESKEYLEN<1:0>:0] must be loaded (with CVM_MT_AES_KEY* instructions) between this CVM_MT_AES_ENC_CBC1 and any prior CVM_MT_AES_DEC_CBC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1 instruction.

AESKEY[AESKEYLEN<1:0>:0] must be predictable prior to this CVM_MT_AES_ENC_CBC1 instruction. This means that when AESKEYLEN<1:0> increases, AESKEY[AESKEYLEN<1:0>:X] (at least) must be written (with CVM_MT_AES_KEY* instructions) between the execution of this CVM_MT_AES_ENC_CBC1 and any immediately preceding CVM_MT_AES_ENC_CBC1 or CVM_MT_AES_ENC1 instruction that used the smaller AESKEYLEN<1:0> value of X.

Cavium Confidential For Power ALU Review

Results are unpredictable when AESKEYLEN<1:0> is zero with the CVM_MT_AES_ENC_CBC1 instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    TIN[1..0]<63:0> = (GPR[rt] ⊕ AESIV[1]) || AESRESINP[0], or
    TIN[1..0]<63:0> = (GPR[rt] || AESRESINP[0]) ⊕ AESIV[1:0]
    AESRESINP[1:0] = AES_ENC(TIN[1:0], AESKEY[3:0], AESKEYLEN<1:0>)
    AESIV[1:0] = AESRESINP[1:0]
    for(i = AESKEYLEN<1:0>+1; i < 4; i++)
      AESKEY[i] = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

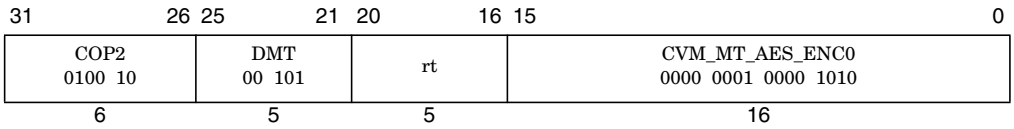
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

AES Encrypt (part 1) — CVM_MT_AES_ENC0



Note: This instruction is the same as SMS4 instruction (CVM_MT_SMS4_ENC0).

Format: DMTC2 rt, 0x010A **CVM**

Purpose:

To load AESRESINP[0] in preparation for a subsequent CVM_MT_AES_ENC1.

Description: AESRESINP[0] = rt

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

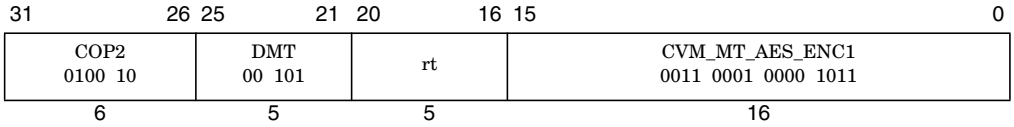
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    AESRESINP[0] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

AES Encrypt (part 2) **CVM_MT_AES_ENC1**



Format: DMTC2 rt, 0x310B **CVM**

Purpose:

To perform an AES encrypt operation. GPR[rt] and AESRESINP[0] are the inputs to an AES encryption operation (using AESKEY and AESKEYLEN). AESRESINP is updated with the result. AESIV and unused AESKEY locations become unpredictable.

Description: AESRESINP[1:0] =
AES_ENC((rt || AESRESINP[0]), AESKEY[3:0], AESKEYLEN<1:0>)

Restrictions:

CVM_MT_AES_ENC0 (or CVM_MT_AES_RESINP0 in the save/restore case) must be executed between the execution of this CVM_MT_AES_ENC1 and the immediately preceding CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1 instruction.

AESKEY[AESKEYLEN<1:0>:0] must be loaded (with CVM_MT_AES_KEY* instructions) between this CVM_MT_AES_ENC_CBC1 and any prior CVM_MT_AES_DEC_CBC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1 instruction.

AESKEY[AESKEYLEN<1:0>:0] must be predictable prior to this CVM_MT_AES_ENC_CBC1 instruction. This means that when AESKEYLEN<1:0> increases, AESKEY[AESKEYLEN<1:0>:X] (at least) must be written (with CVM_MT_AES_KEY* instructions) between the execution of this CVM_MT_AES_ENC1 and any immediately preceding CVM_MT_AES_ENC_CBC1 or CVM_MT_AES_ENC1 instruction that used the smaller AESKEYLEN<1:0> value of X.

Results are unpredictable when AESKEYLEN<1:0> is zero with the CVM_MT_AES_ENC1 instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Cavium Confidential For Beverly Hu 02/7/2011

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    AESRESINP[1:0] = AES_ENC((GPR[rt] || AESRESINP[0]), AESKEY[3:0], AESKEYLEN<1:0>)
    AESIV[1..0] = unpredictable
    for(i = AESKEYLEN<1:0>+1; i < 4; i++)
      AESKEY[i] = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

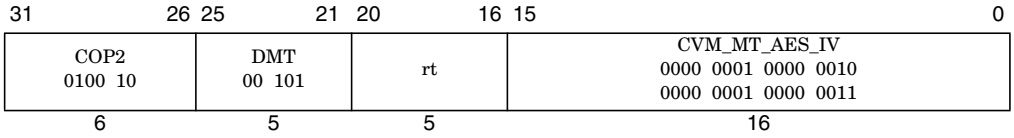
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load IV into AES Unit — CVM_MT_AES_IV



Note: These instructions are the same as Load IV into SMS4 Unit instructions (CVM_MT_SMS4_IV 0–1).

Format: DMTC2 rt, 0x0102 **CVM**

DMTC2 rt, 0x0103

Purpose:

To load AESIV.

Description:
```
AESIV[0] = rt // DMTC2 rt, 0x0102
AESIV[1] = rt // DMTC2 rt, 0x0103
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

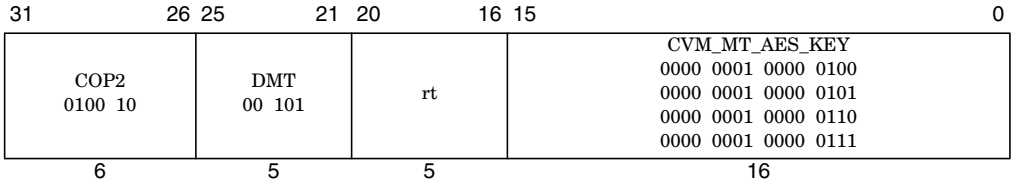
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0102: AESIV[0] = GPR[rt]
    case 0x0103: AESIV[1] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Key into AES Unit — CVM_MT_AES_KEY



Note: The first two instructions (CVM_MT_AES_KEY 0–1) are the same as Load Key into SMS4 Unit instructions (CVM_MT_SMS4_KEY 0–1).

Format: DMTC2 rt, 0x0104 **CVM**
DMTC2 rt, 0x0105
DMTC2 rt, 0x0106
DMTC2 rt, 0x0107

Purpose:

To load AESKEY.

Description:
AESKEY[0] = rt // DMTC2 rt, 0x0104
AESKEY[1] = rt // DMTC2 rt, 0x0105
AESKEY[2] = rt // DMTC2 rt, 0x0106
AESKEY[3] = rt // DMTC2 rt, 0x0107

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

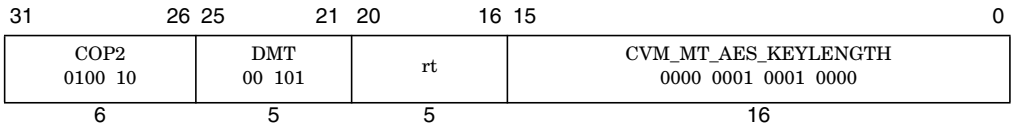
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0104: AESKEY[0] = GPR[rt]; AESKEYLEN<2:0> = unpredictable
    case 0x0105: AESKEY[1] = GPR[rt]; AESKEYLEN<2:0> = 1 // 1 is 128-bit key
    case 0x0106: AESKEY[2] = GPR[rt]; AESKEYLEN<2:0> = 2 // 2 is 192-bit key
    case 0x0107: AESKEY[3] = GPR[rt]; AESKEYLEN<2:0> = 3 // 3 is 256-bit key
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Key Length into AES Unit — CVM_MT_AES_KEYLENGTH



Format: DMTC2 rt, 0x0110 **CVM**

Purpose:

To load AESKEYLEN.

Description: AESLEYLEN<1:0> = rt<1:0>

AESKEYLEN	Length of AES key
00	unpredictable
01	128 bits
10	192 bits
11	256 bits

Restrictions:

Results are unpredictable when an AESKEYLEN value of 0 is used.

Results are unpredictable when GPR[rt]<63:2> are not zero.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    AESLEYLEN<1:0> = GPR[rt]<1:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

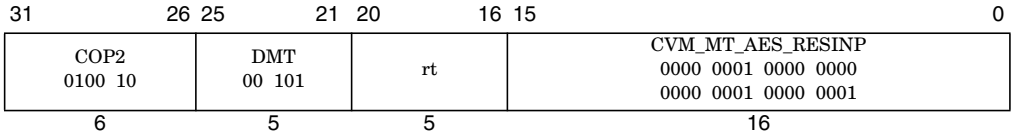
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 03/27/2014

Load Result/Input into AES Unit — CVM_MT_AES_RESINP



Note: These instructions are the same as Load Result/Input into SMS4 or CAMELLIA Unit instructions (CVM_MT_SMS4_RESINP 0–1 and CVM_MT_CAMELLIA_RESINP 0–1, respectively).

Format: DMTC2 rt, 0x0100 **CVM**
DMTC2 rt, 0x0101

Purpose:

To load AESRESINP.

Description:
```
AESRESINP[0]  = rt // DMTC2 rt, 0x0100
AESRESINP[1]  = rt // DMTC2 rt, 0x0101
```

Restrictions:

Either CVM_MF_AES_RESINP0 or CVM_MF_AES_RESINP1 must execute between this CVM_MT_AES_RESINP* instruction and the last CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC1, CVM_MT_AES_DEC_CBC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC1, or CVM_MT_SMS4_DEC_CBC1 instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0102: AESRESINP[0] = GPR[rt]
    case 0x0103: AESRESINP[1] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

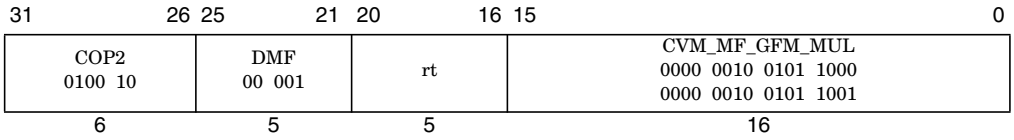
Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yan ALU 02/27/2014

Galois Field Multiplier

Load Multiplier from GFM Unit — CVM_MF_GFM_MUL



Format: DMFC2 rt, 0x0258 **CVM**

DMFC2 rt, 0x0259

Purpose:

To read GFMMUL from the GFM coprocessor.

Description: rt = GFMMUL[0] // DMFC2 rt, 0x0258
rt = GFMMUL[1] // DMFC2 rt, 0x0259

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0258: GPR[rt]<63:0> = GFMMUL[0]<63:0>)
    case 0x0259: GPR[rt]<63:0> = GFMMUL[1]<63:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

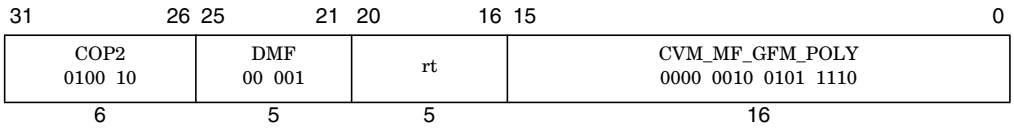
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Polynomial from GFM Unit — CVM_MF_GFM_POLY



Format: DMFC2 rt, 0x025E **CVM**

Purpose:

To read GFMPOLY from the GFM coprocessor.

Description: rt = GFMPOLY<15:0>

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    GPR[rt]<63:0> = 0^48 || GFMPOLY<15:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

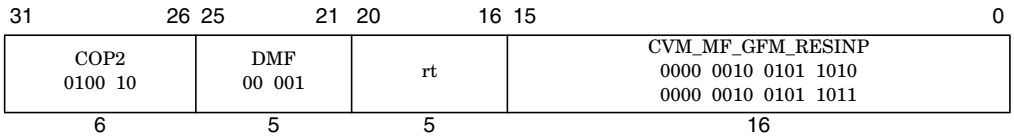
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Result/Input from GFM Unit — CVM_MF_GFM_RESINP



Format: DMFC2 rt, 0x025A **CVM**
DMFC2 rt, 0x025B

Purpose:

To read GFMRESINP from the GFM coprocessor.

Description:
```
rt = GFMRESINP[0] // DMFC2 rt, 0x025A
rt = GFMRESINP[1] // DMFC2 rt, 0x025B
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x025A: GPR[rt]<63:0> = GFMRESINP[0]<63:0>
    case 0x025B: GPR[rt]<63:0> = GFMRESINP[1]<63:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

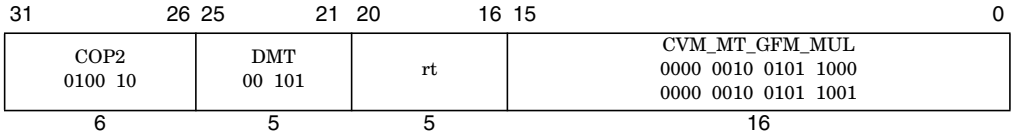
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Multiplier into GFM Unit — CVM_MT_GFM_MUL



Format: DMTC2 rt, 0x0258 **CVM**

DMTC2 rt, 0x0259

Purpose:

To load GFMMUL.

Description:
```
GFMMUL[0] = rt  // DMTC2 rt, 0x0258
GFMMUL[1] = rt  // DMTC2 rt, 0x0259
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x258: GFMMUL[0] = GPR[rt]
    case 0x259: GFMMUL[1] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

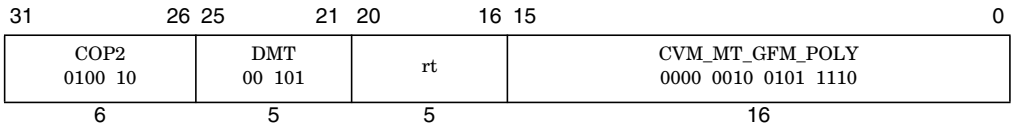
Reserved Instruction

Programming Notes:

For GF(2^{64}) multiplication, GFMMUL[1] should be set to zero and GFMMUL[0] should contain the 64-bit multiplier.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Polynomial into GFM Unit — CVM_MT_GFM_POLY



Format: DMTC2 rt, 0x025E **CVM**

Purpose:

To load GFMPOLY.

Description: GFMPOLY<15:0> = rt<15:0>

Restrictions:

Results are unpredictable when GPR[rt]<63:16> are not zero.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    GFMPOLY<15:0> = GPR[rt]<15:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

Reserved Instruction

Programming Notes:

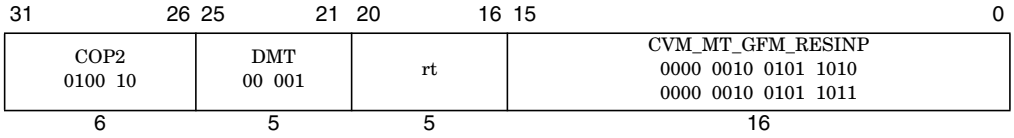
There are at least two interesting GFMPOLY values, corresponding to the two polynomials covered in “The Galois/Counter Mode of Operation” by McGrew and Viega:

GFMPOLY = 0xE100 for $1 + \alpha + \alpha^2 + \alpha^7 + \alpha^{128}$ in $GF(2^{128})$

GFMPOLY = 0x00D8 for $1 + \alpha + \alpha^3 + \alpha^4 + \alpha^{64}$ in $GF(2^{64})$

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Result/Input into GFM Unit — CVM_MT_GFM_RESINP



Format: DMTC2 rt, 0x025A **CVM**
DMTC2 rt, 0x025B

Purpose:

To load GFMRESINP.

Description:
```
GFMRESINP[0] = rt  // DMFC2 rt, 0x025A
GFMRESINP[1] = rt  // DMFC2 rt, 0x025B
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x25A: GFMRESINP[0] = GPR[rt]
    case 0x25B: GFMRESINP[1] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

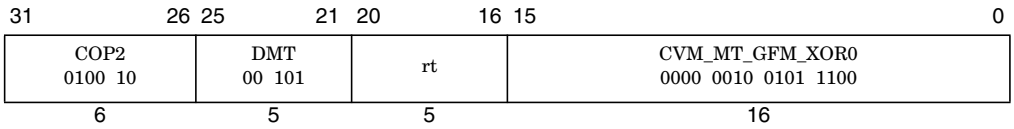
Reserved Instruction

Programming Notes:

For GF(2^{64}) multiplication, GFMRESINP[0] should be set to zero and GFMRESINP[1] should contain the 64-bit value.

Cavium Confidential For Bowen Yang ALU 02/27/2014

XOR into GFM Unit — CVM_MT_GFM_XOR0



Format: DMTC2 rt, 0x025C **CVM**

Purpose:

To XOR the contents of GFMRESINP[0] with rt.

Description: GFMRESINP[0] = GFMRESINP[0] ⊕ rt

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    GFMPRESINP[0] = GFMPRESINP[0] ⊕ GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

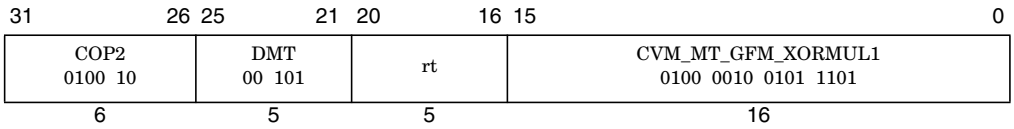
Reserved Instruction

Programming Notes:

CVM_MT_GFM_XOR0 should not be used for GF(2^{64}) multiplications. (GFMRESINP[0] should be zero.)

Cavium Confidential For Bowen Yang ALU 02/27/2014

XOR and GF Multiply — CVM_MT_GFM_XORMUL1



Format: DMTC2 rt, 0x425D **CVM**

Purpose:

To XOR the contents of GFMRESINP[1] with rt and perform a GF(2^{128}) multiplication of GFMRESINP and GFMMUL (using the polynomial selected by GFMPOLY), depositing the result in GFMRESINP.

Description: GFMRESINP[1] = GFMRESINP[1] $\oplus$ rt
GFMRESINP[1:0] = GFMMULT(GFMRESINP, GFMMUL, GFMPOLY)

Restrictions:

If GFMPOLY<15:8> is 0x0, this instruction executes a GF(2^{64}) XOR and multiplication (rather than GF(2^{128})) involving rt, GFMRESINP[1], and GFMMUL[0]. When GFMPOLY<15:8> is 0x0, both GFMRESINP[0]<63:0> and GFMMUL[1]<63:0> must be 0x0, or else both GFMRESINP[1:0]<63:0> and GFMMUL[1:0]<63:0> will be unpredictable after the instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    Z[1:0]<63:0> = 0
    V[0]<63:0> = doubleword_bit_reflect(GFMRESINP[0]<63:0>)
    V[1]<63:0> = doubleword_bit_reflect(GFMRESINP[1]<63:0> ⊕ GPR[rt])
    Y[0]<63:0> = doubleword_bit_reflect(GFMMUL[0]<63:0>)
    Y[1]<63:0> = doubleword_bit_reflect(GFMMUL[1]<63:0>)
    T<15:0> = halfword_bit_reflect(GFMPOLY<15:0>);
    POLY[0]<63:0> = 0^56 || T<7:0>
    POLY[1]<63:0> = 0^56 || T<15:8>
    for(i = 0; i < 128; i++) {
      if(Y[i >> 6]<i & 0x3F>)
        Z[1:0] = Z[1:0] ⊕ V[1:0]
      R[1:0]<63:0> = V[1]<63> ? POLY[1:0]<63:0> : 0^128
      V[1]<63:0> || V[0]<63:0> = V[1]<62:0> || V[0]<63:0> || 0^1
      V[1:0] = V[1:0] ⊕ R[1:0]
    }
    GFMRESINP[0]<63:0> = doubleword_bit_reflect(Z[0]<63:0>)
    GFMRESINP[1]<63:0> = doubleword_bit_reflect(Z[1]<63:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

Reserved Instruction

Programming Notes:

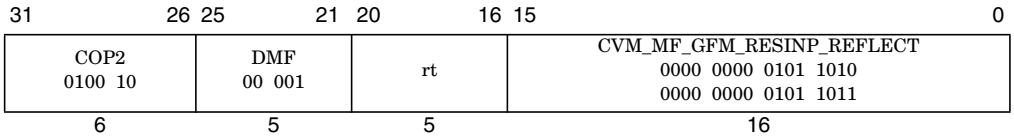
CVM_MT_GFM_XORMUL1 should be used to initiate GF(2^{64}) multiplications, as well as GF(2^{128}). GPR[rt] is XORed with GFMRESINP[1] to create the actual 64-bit input for the GF(2^{64}) multiplication. Assuming that GFMRESINP[0] and GFMMUL[1] are zero, GFMRESINP[1] will contain the GF(2^{64}) multiplication result:

$$GFMRESINP[1] = (GPR[rt] \oplus GFMRESINP[1]) \times GFMMUL[0]$$

Cavium Confidential For Bowen Yang ALU 02/27/2014

Galois Field Multiplier REFLECT

Load Result/Input from GFM Unit Reflected CVM_MF_GFM_RESINP_REFLECT



Format: DMFC2 rt, 0x005A **CVMv2**

DMFC2 rt, 0x005B

Purpose:

To read GFMRESINP from the GFM coprocessor. The bits within the entire GFMRESINP word are reflected.

Description: rt = doubleword_bit_reflect(GFMRESINP[0]) // DMFC2 rt, 0x005A
rt = doubleword_bit_reflect(GFMRESINP[1]) // DMFC2 rt, 0x005B

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
   if (not Are64bitOperationsEnabled()) or !CvmCtl[SNOW3G] then
      SignalException(ReservedInstruction)
   else
      case 0x005A: GPR[rt]<63:0> = doubleword_bit_reflect(GFMRESINP[0]<63:0>)
      case 0x005B: GPR[rt]<63:0> = doubleword_bit_reflect(GFMRESINP[1]<63:0>)
   endif
else
   SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

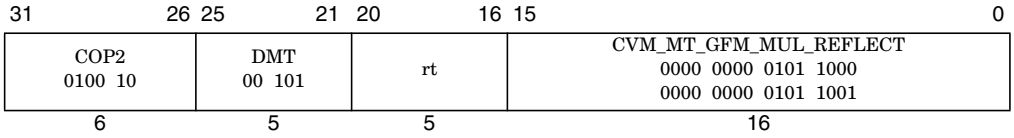
Programming notes:

CVM_MF_GFM_RESINP_REFLECT is the same as CVM_MF_GFM_RESINP, except the rt bits are reflected/swapped (bit 63 becomes bit 0, bit 62 becomes bit 1, ...).

The CVM_MF_GFM_RESINP_REFLECT instruction first appeared on the CN6XXX series parts. The behavior of the CVM_MF_GFM_RESINP_REFLECT instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX series parts. CvmCtl[SNOW3G] is always 0 on CN3XXX/CN5XXX parts, so accurately portrays the lack of GFM reflect instructions. However, there is no guarantee of a Reserved Instruction exception when CVM_MF_GFM_RESINP_REFLECT is executed on these earlier parts. GFM reflect code that is portable to all of CN6XXX (and successors) and CN3XXX/CN5XXX must not execute CVM_MF_GFM_RESINP_REFLECT when CvmCtl[SNOW3G] is clear.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Multiplier into GFM Unit Reflect **CVM_MT_GFM_MUL_REFLECT**



Format: DMTC2 rt, 0x0058 **CVMv2**

DMTC2 rt, 0x0059

Purpose:

To load GFMMUL. The bits in the entire GFMMUL word are reflected.

Description:
```
GFMMUL[0] = doubleword_bit_reflect(rt)  // DMTC2 rt, 0x0058
GFMMUL[1] = doubleword_bit_reflect(rt)  // DMTC2 rt, 0x0059
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
   if (not Are64bitOperationsEnabled()) or !CvmCtl[SNOW3G] then
      SignalException(ReservedInstruction)
   else
      case 0x058: GFMMUL[0] = doubleword_bit_reflect(GPR[rt])
      case 0x059: GFMMUL[1] = doubleword_bit_reflect(GPR[rt])
   endif
else
   SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Programming Notes:

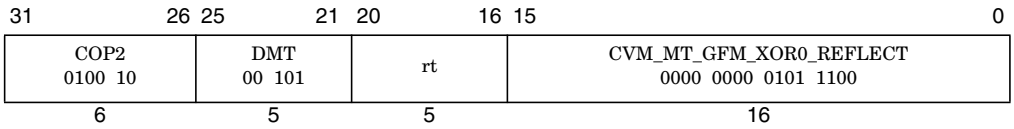
CVM_MT_GFM_MUL_REFLECT is the same as CVM_MT_GFM_MUL, except the rt bits are reflected/swapped (bit 63 becomes bit 0, bit 62 becomes bit 1, ...).

The CVM_MT_GFM_MUL_REFLECT instruction first appeared on the CN6XXX series parts. The behavior of the CVM_MT_GFM_MUL_REFLECT instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX series parts. CvmCtl[SNOW3G] is always 0 on CN3XXX/CN5XXX parts, so accurately portrays the lack of GFM reflect instructions. However, there is no guarantee of a Reserved Instruction exception when CVM_MT_GFM_MUL_REFLECT is executed on these earlier parts. GFM reflect code that is portable to all of CN6XXX (and successors) and CN3XXX/CN5XXX must not execute CVM_MT_GFM_MUL_REFLECT when CvmCtl[SNOW3G] is clear.

For GF(2^{64}) multiplication, GFMMUL[1] should be set to 0 and GFMMUL[0] should contain the 64-bit multiplier.

Cavium Confidential For Bowen Yang ALU 02/27/2014

XOR into GFM Unit Reflected — CVM_MT_GFM_XOR0_REFLECT



Format: DMTC2 rt, 0x005C **CVMv2**

Purpose:

To XOR the contents of GFMRESINP[0] with the reflected bits of rt.

Description: GFMRESINP[0] = GFMRESINP[0] ⊕ doubleword_bit_reflect(rt)

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
   if (not Are64bitOperationsEnabled()) or !CvmCtl[SNOW3G] then
      SignalException(ReservedInstruction)
   else
      GFMPRESINP[0] = GFMPRESINP[0] XOR doubleword_bit_reflect(GPR[rt])
   endif
else
   SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Programming Notes:

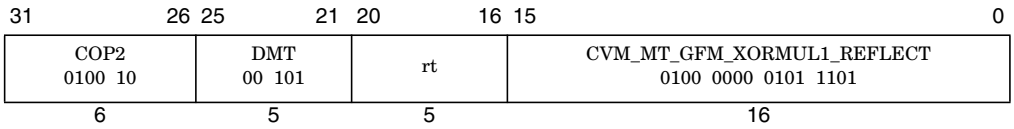
CVM_MT_GFM_XOR0_REFLECT is the same as CVM_MT_GFM_XOR0, except the rt bits are reflected/swapped (bit 63 becomes bit 0, bit 62 becomes bit 1, ...).

The CVM_MT_GFM_XOR0_REFLECT instruction first appeared on the 6 series parts. The behavior of the CVM_MT_GFM_XOR0_REFLECT instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX series parts. CvmCtl[SNOW3G] is always 0 on CN3XXX/CN5XXX parts, so accurately portrays the lack of GFM reflect instructions. However, there is no guarantee of a Reserved Instruction exception when CVM_MT_GFM_XOR0_REFLECT is executed on these earlier parts. GFM reflect code that is portable to all of CN6XXX (and successors) and CN3XXX/CN5XXX must not execute CVM_MT_GFM_XOR0_REFLECT when CvmCtl[SNOW3G] is clear.

CVM_MT_GFM_XOR0_REFLECT should not be used for GF(2^{64}) multiplications. (GFMRESINP[0] should be 0.)

Cavium Confidential For Bowen Yang ALU 02/27/2014

XOR and GF Multiply Reflected CVM_MT_GFM_XORMUL1_REFLECT



Format: DMTC2 rt, 0x405D **CVMv2**

Purpose:

To XOR the contents of GFMRESINP[1] with the reflected bits of rt and perform a GF(2^{128}) multiplication of GFMRESINP and GFMMUL (using the polynomial selected by GFMPOLY), depositing the result in GFMRESINP.

Description: GFMRESINP[1] = GFMRESINP[1] $\oplus$ doubleword_bit_reflect(rt)
GFMRESINP[1:0] = GFMMULT(GFMRESINP, GFMMUL, GFMPOLY)

Restrictions:

If GFMPOLY<15:8> is 0x0, this instruction executes a GF(2^{64}) XOR and multiplication (rather than GF(2^{128})) involving rt, GFMRESINP[1], and GFMMUL[0]. When GFMPOLY<15:8> is 0x0, both GFMRESINP[0]<63:0> and GFMMUL[1]<63:0> must be 0x0, or else both GFMRESINP[1:0]<63:0> and GFMMUL[1:0]<63:0> will be unpredictable after the instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    Z[1:0]<63:0> = 0
    V[0]<63:0> = doubleword_bit_reflect(GFMRESINP[0]<63:0>)
    V[1]<63:0> = doubleword_bit_reflect(GFMRESINP[1]<63:0>) ⊕ GPR[rt]
    Y[0]<63:0> = doubleword_bit_reflect(GFMMUL[0]<63:0>)
    Y[1]<63:0> = doubleword_bit_reflect(GFMMUL[1]<63:0>)
    T<15:0> = halfword_bit_reflect(GFMPOLY<15:0>);
    POLY[0]<63:0> = 0^56 || T<7:0>
    POLY[1]<63:0> = 0^56 || T<15:8>
    for(i = 0; i < 128; i++) {
      if(Y[i >> 6]<i & 0x3F>)
        Z[1:0] = Z[1:0] ⊕ V[1:0]
      R[1:0]<63:0> = V[1]<63> ? POLY[1:0]<63:0> : 0^128
      V[1]<63:0> || V[0]<63:0> = V[1]<62:0> || V[0]<63:0> || 0^1
      V[1:0] = V[1:0] ⊕ R[1:0]
    }
    GFMRESINP[0]<63:0> = doubleword_bit_reflect(Z[0]<63:0>)
    GFMRESINP[1]<63:0> = doubleword_bit_reflect(Z[1]<63:0>)
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Programming Notes:

CVM_MT_GFM_XORMUL1_REFLECT should be used to initiate GF(2^{64}) multiplications, as well as GF(2^{128}). GPR[rt] is reflected and XORed with GFMRESINP[1] to create the actual 64-bit input for the GF(2^{64}) multiplication. Assuming that GFMRESINP[0] and GFMMUL[1] are 0x0, GFMRESINP[1] contains the GF(2^{64}) multiplication result:

$$GFMRESINP[1] = (doubleword_bit_reflect(GPR[rt]) \oplus GFMRESINP[1]) \times GFMMUL[0]$$

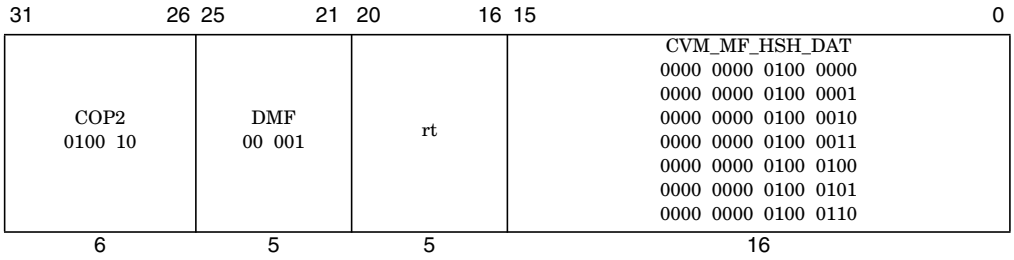
CVM_MT_GFM_XORMUL1_REFLECT is the same as CVM_MT_GFM_XORMUL1, except the rt bits are reflected/swapped (bit 63 becomes bit 0, bit 62 becomes bit 1, ...).

The CVM_MT_GFM_XORMUL1_REFLECT instruction first appeared on the OCTEON CN6XXX series parts. The behavior of the CVM_MT_GFM_XORMUL1_REFLECT instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX series parts. CvmCtl[SNOW3G] is always 0 on CN3XXX/CN5XXX parts, so accurately portrays the lack of GFM reflect instructions. However, there is no guarantee of a Reserved Instruction exception when CVM_MT_GFM_XORMUL1_REFLECT is executed on these earlier parts. GFM reflect code that is portable to all of CN6XXX (and successors) and CN3XXX/CN5XXX must not execute CVM_MT_GFM_XORMUL1_REFLECT when CvmCtl[SNOW3G] is clear.

Cavium Confidential For Bowen Yang ALU 2/27/2014

Hash (MD5/SHA-1/ SHA-256/SHA-512)

Load Data from HSH Unit (narrow mode) CVM_MF_HSH_DAT



Format: DMFC2 rt, 0x0040 **CVM**

DMFC2 rt, 0x0041

DMFC2 rt, 0x0042

DMFC2 rt, 0x0043

DMFC2 rt, 0x0044

DMFC2 rt, 0x0045

DMFC2 rt, 0x0046

Purpose:

To read HASHDAT from the HSH coprocessor in narrow mode. MD5, SHA-1, and SHA-256 use the narrow mode. The most-significant and least-significant 32 bits of rt are written with the least-significant 32 bits of separate 64-bit HASHDAT double-words.

Description:

```
rt = HASHDAT[0]<31:0>  || HASHDAT[1]<31:0>  // DMFC2 rt, 0x0040
rt = HASHDAT[2]<31:0>  || HASHDAT[3]<31:0>  // DMFC2 rt, 0x0041
rt = HASHDAT[4]<31:0>  || HASHDAT[5]<31:0>  // DMFC2 rt, 0x0042
rt = HASHDAT[6]<31:0>  || HASHDAT[7]<31:0>  // DMFC2 rt, 0x0043
rt = HASHDAT[8]<31:0>  || HASHDAT[9]<31:0>  // DMFC2 rt, 0x0044
rt = HASHDAT[10]<31:0> || HASHDAT[11]<31:0> // DMFC2 rt, 0x0045
rt = HASHDAT[12]<31:0> || HASHDAT[13]<31:0> // DMFC2 rt, 0x0046
```

Restrictions:

This instruction returns unpredictable results after a SNOW3G or ZUC operation and renders the value of any results from SNOW3G and ZUC instructions unpredictable. See the programming notes of CVM_MF_SNOW3G_LFSR, CVM_MF_SNOW3G_FSM, CVM_MF_SNOW3G_RESULT, CVM_MT_SNOW3G_LFSR, CVM_MT_SNOW3G_FSM, and CVM_MT_SNOW3G_RESULT in section SNOW 3G on page 1835 and CVM_MF_ZUC_LFSR, CVM_MF_ZUC_FSM, CVM_MF_ZUC_RESULT, CVM_MF_ZUC_TRESULT, CVM_MT_ZUC_LFSR, CVM_MT_ZUC_FSM, and CVM_MT_ZUC_RESULT in section ZUC on page 1877.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Cavium Confidential For Bowen Yang ARB 07/27/2015

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0040: GPR[rt]<63:32> = HASHDAT[ 0]<31:0>
                 GPR[rt]<31:0>  = HASHDAT[ 1]<31:0>
    case 0x0041: GPR[rt]<63:32> = HASHDAT[ 2]<31:0>
                 GPR[rt]<31:0>  = HASHDAT[ 3]<31:0>
    case 0x0042: GPR[rt]<63:32> = HASHDAT[ 4]<31:0>
                 GPR[rt]<31:0>  = HASHDAT[ 5]<31:0>
    case 0x0043: GPR[rt]<63:32> = HASHDAT[ 6]<31:0>
                 GPR[rt]<31:0>  = HASHDAT[ 7]<31:0>
    case 0x0044: GPR[rt]<63:32> = HASHDAT[ 8]<31:0>
                 GPR[rt]<31:0>  = HASHDAT[ 9]<31:0>
    case 0x0045: GPR[rt]<63:32> = HASHDAT[10]<31:0>
                 GPR[rt]<31:0>  = HASHDAT[11]<31:0>
    case 0x0046: GPR[rt]<63:32> = HASHDAT[12]<31:0>
                 GPR[rt]<31:0>  = HASHDAT[13]<31:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

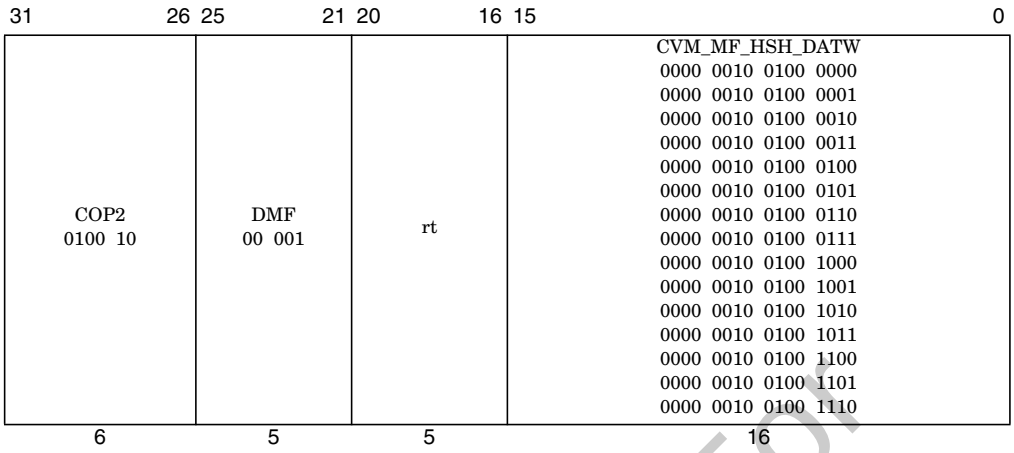
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Load Data from HSH Unit (wide mode) **CVM_MF_HSH_DATW**



Note: These instructions 0–7 are the same as Load LFSR from SNOW3G Unit and ZUC Unit (CVM_MF_SNOW3G_LFSR 0–7 and CVM_MF_ZUC_LFSR 0–7, respectively) instructions.

Format: DMFC2 rt, 0x0240 **CVM**

DMFC2 rt, 0x0241

DMFC2 rt, 0x0242

DMFC2 rt, 0x0243

DMFC2 rt, 0x0244

DMFC2 rt, 0x0245

DMFC2 rt, 0x0246

DMFC2 rt, 0x0247

DMFC2 rt, 0x0248

DMFC2 rt, 0x0249

DMFC2 rt, 0x024A

DMFC2 rt, 0x024B

DMFC2 rt, 0x024C

DMFC2 rt, 0x024D

DMFC2 rt, 0x024E

Purpose:

To read HASHDAT from the HSH coprocessor in wide mode. SHA-512 uses the wide mode.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Description:

```
rt = HASHDAT[0]  // DMFC2 rt, 0x0240
rt = HASHDAT[1]  // DMFC2 rt, 0x0241
rt = HASHDAT[2]  // DMFC2 rt, 0x0242
rt = HASHDAT[3]  // DMFC2 rt, 0x0243
rt = HASHDAT[4]  // DMFC2 rt, 0x0244
rt = HASHDAT[5]  // DMFC2 rt, 0x0245
rt = HASHDAT[6]  // DMFC2 rt, 0x0246
rt = HASHDAT[7]  // DMFC2 rt, 0x0247
rt = HASHDAT[8]  // DMFC2 rt, 0x0248
rt = HASHDAT[9]  // DMFC2 rt, 0x0249
rt = HASHDAT[10] // DMFC2 rt, 0x024A
rt = HASHDAT[11] // DMFC2 rt, 0x024B
rt = HASHDAT[12] // DMFC2 rt, 0x024C
rt = HASHDAT[13] // DMFC2 rt, 0x024D
rt = HASHDAT[14] // DMFC2 rt, 0x024E
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0240: GPR[rt] = HASHDAT[0]
    case 0x0241: GPR[rt] = HASHDAT[1]
    case 0x0242: GPR[rt] = HASHDAT[2]
    case 0x0243: GPR[rt] = HASHDAT[3]
    case 0x0244: GPR[rt] = HASHDAT[4]
    case 0x0245: GPR[rt] = HASHDAT[5]
    case 0x0246: GPR[rt] = HASHDAT[6]
    case 0x0247: GPR[rt] = HASHDAT[7]
    case 0x0248: GPR[rt] = HASHDAT[8]
    case 0x0249: GPR[rt] = HASHDAT[9]
    case 0x024A: GPR[rt] = HASHDAT[10]
    case 0x024B: GPR[rt] = HASHDAT[11]
    case 0x024C: GPR[rt] = HASHDAT[12]
    case 0x024D: GPR[rt] = HASHDAT[13]
    case 0x024E: GPR[rt] = HASHDAT[14]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

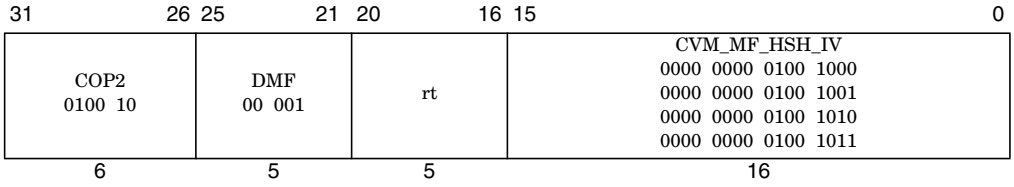
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Brian Yang ALU 02/27/2014

Load IV from HSH Unit (narrow mode) CVM_MF_HSH_IV



Format: DMFC2 rt, 0x0048 **CVM**

DMFC2 rt, 0x0049

DMFC2 rt, 0x004A

DMFC2 rt, 0x004B

Purpose:

To read HASHIV from the HSH coprocessor in narrow mode. MD5, SHA-1, and SHA-256 use the narrow mode. The most-significant and least-significant 32 bits of rt are written with the least-significant 32 bits of separate 64-bit HASHIV double-words.

Description:

```
rt = HASHIV[0]<31:0> || HASHIV[1]<31:0> // DMFC2 rt, 0x0048
rt = HASHIV[2]<31:0> || HASHIV[3]<31:0> // DMFC2 rt, 0x0049
rt = HASHIV[4]<31:0> || HASHIV[5]<31:0> // DMFC2 rt, 0x004A
rt = HASHIV[6]<31:0> || HASHIV[7]<31:0> // DMFC2 rt, 0x004B
```

Restrictions:

This instruction returns unpredictable results after a SNOW3G and a ZUC operation and renders the value of any results from SNOW3G and ZUC instructions unpredictable. See the programming notes of CVM_MF_SNOW3G_LFSR, CVM_MF_SNOW3G_FSM, CVM_MF_SNOW3G_RESULT, CVM_MT_SNOW3G_LFSR, CVM_MT_SNOW3G_FSM, and CVM_MT_SNOW3G_RESULT in section SNOW 3G on page 1835 and CVM_MF_ZUC_LFSR, CVM_MF_ZUC_FSM, CVM_MF_ZUC_RESULT, CVM_MF_ZUC_TRESULT, CVM_MT_ZUC_LFSR, CVM_MT_ZUC_FSM, and CVM_MT_ZUC_RESULT in section ZUC on page 1877.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Cavium Confidential For Bowen Yang 2012/12/21

Operation:

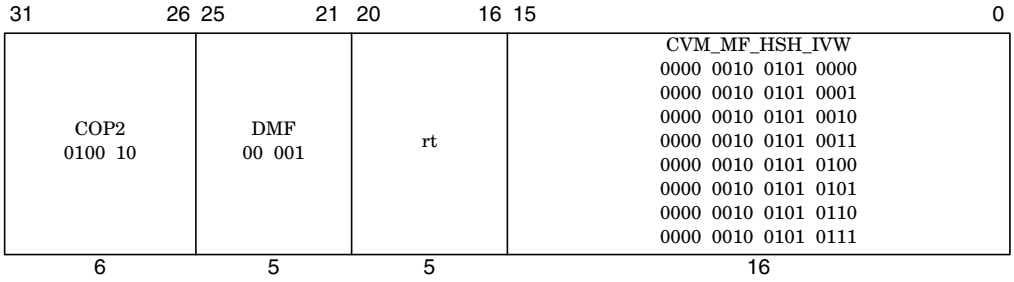
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0048: GPR[rt]<63:32> = HASHIV[0]<31:0>
                 GPR[rt]<31:0>  = HASHIV[1]<31:0>
    case 0x0049: GPR[rt]<63:32> = HASHIV[2]<31:0>
                 GPR[rt]<31:0>  = HASHIV[3]<31:0>
    case 0x004A: GPR[rt]<63:32> = HASHIV[4]<31:0>
                 GPR[rt]<31:0>  = HASHIV[5]<31:0>
    case 0x004B: GPR[rt]<63:32> = HASHIV[6]<31:0>
                 GPR[rt]<31:0>  = HASHIV[7]<31:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable
Reserved Instruction

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Load IV from HSH Unit (wide mode) **CVM_MF_HSH_IVW**



Note: This instruction 0 is the same as Load Result from SNOW 3G Unit and ZUC Unit (CVM_MF_SNOW3G_RESULT and CVM_MF_ZUC_RESULT, respectively) instructions.

These instructions 1-2 are the same as Load FSM from SNOW3G and ZUC Unit (CVM_MF_SNOW3G_FSM 0-1 and CVM_MF_ZUC_FSM 0-1, respectively) instructions.

This instruction 3 is the same as CVM_MF_SNOW3G_FSM 2 and CVM_MF_ZUC_TRESULT.

Format: DMFC2 rt, 0x0250 **CVM**

DMFC2 rt, 0x0251

DMFC2 rt, 0x0252

DMFC2 rt, 0x0253

DMFC2 rt, 0x0254

DMFC2 rt, 0x0255

DMFC2 rt, 0x0256

DMFC2 rt, 0x0257

Purpose:

To read HASHIV from the HSH coprocessor in wide mode. SHA-512 use the wide mode.

Description:

```
rt = HASHIV[0] // DMFC2 rt, 0x0250
rt = HASHIV[1] // DMFC2 rt, 0x0251
rt = HASHIV[2] // DMFC2 rt, 0x0252
rt = HASHIV[3] // DMFC2 rt, 0x0253
rt = HASHIV[4] // DMFC2 rt, 0x0254
rt = HASHIV[5] // DMFC2 rt, 0x0255
rt = HASHIV[6] // DMFC2 rt, 0x0256
rt = HASHIV[7] // DMFC2 rt, 0x0257
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0250: GPR[rt] = HASHIV[0]
    case 0x0251: GPR[rt] = HASHIV[1]
    case 0x0252: GPR[rt] = HASHIV[2]
    case 0x0253: GPR[rt] = HASHIV[3]
    case 0x0254: GPR[rt] = HASHIV[4]
    case 0x0255: GPR[rt] = HASHIV[5]
    case 0x0256: GPR[rt] = HASHIV[6]
    case 0x0257: GPR[rt] = HASHIV[7]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

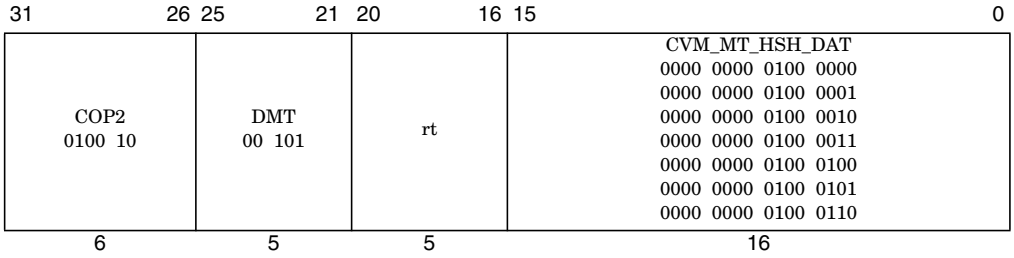
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Load Data into HSH Unit (narrow mode) **CVM_MT_HSH_DAT**



Format: DMTC2 rt, 0x0040 **CVM**

DMTC2 rt, 0x0041

DMTC2 rt, 0x0042

DMTC2 rt, 0x0043

DMTC2 rt, 0x0044

DMTC2 rt, 0x0045

DMTC2 rt, 0x0046

Purpose:

To load values into HASHDAT via the narrow mechanism. The MD5, SHA-1, and SHA-256 algorithms use the narrow HASHDAT format. The most-significant and least-significant 32 bits of rt are written into the least-significant 32 bits of separate 64-bit double-words HASHDAT.

Description:

```
HASHDAT[0]<31:0>  || HASHDAT[1]<31:0>  = rt // DMTC2 rt,0x0040
HASHDAT[2]<31:0>  || HASHDAT[3]<31:0>  = rt // DMTC2 rt,0x0041
HASHDAT[4]<31:0>  || HASHDAT[5]<31:0>  = rt // DMTC2 rt,0x0042
HASHDAT[6]<31:0>  || HASHDAT[7]<31:0>  = rt // DMTC2 rt,0x0043
HASHDAT[8]<31:0>  || HASHDAT[9]<31:0>  = rt // DMTC2 rt,0x0044
HASHDAT[10]<31:0> || HASHDAT[11]<31:0> = rt // DMTC2 rt,0x0045
HASHDAT[12]<31:0> || HASHDAT[13]<31:0> = rt // DMTC2 rt,0x0046
```

Restrictions:

This instruction renders the value of any results from SNOW3G and ZUC instructions unpredictable. See the programming notes of CVM_MF_SNOW3G_LFSR, CVM_MF_SNOW3G_FSM, CVM_MF_SNOW3G_RESULT, CVM_MT_SNOW3G_LFSR, CVM_MT_SNOW3G_FSM, and CVM_MT_SNOW3G_RESULT in section SNOW 3G on page 1835 and and CVM_MF_ZUC_LFSR, CVM_MF_ZUC_FSM, CVM_MF_ZUC_RESULT, CVM_MF_ZUC_TRESULT, CVM_MT_ZUC_LFSR, CVM_MT_ZUC_FSM, and CVM_MT_ZUC_RESULT in section ZUC on page 1877.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Cavium Confidential For Bowen Jiang ALU 7/7/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0040: HASHDAT[0]<31:0>       = GPR[rt]<63:32>
                 HASHDAT[1]<31:0>       = GPR[rt]<31:0>
                 HASHDAT[0..1]<63:32>   = unpredictable
    case 0x0041: HASHDAT[2]<31:0>       = GPR[rt]<63:32>
                 HASHDAT[3]<31:0>       = GPR[rt]<31:0>
                 HASHDAT[2..3]<63:32>   = unpredictable
    case 0x0042: HASHDAT[4]<31:0>       = GPR[rt]<63:32>
                 HASHDAT[5]<31:0>       = GPR[rt]<31:0>
                 HASHDAT[4..5]<63:32>   = unpredictable
    case 0x0043: HASHDAT[6]<31:0>       = GPR[rt]<63:32>
                 HASHDAT[7]<31:0>       = GPR[rt]<31:0>
                 HASHDAT[6..7]<63:32>   = unpredictable
    case 0x0044: HASHDAT[8]<31:0>       = GPR[rt]<63:32>
                 HASHDAT[9]<31:0>       = GPR[rt]<31:0>
                 HASHDAT[8..9]<63:32>   = unpredictable
    case 0x0045: HASHDAT[10]<31:0>      = GPR[rt]<63:32>
                 HASHDAT[11]<31:0>      = GPR[rt]<31:0>
                 HASHDAT[10..11]<63:32> = unpredictable
    case 0x0046: HASHDAT[12]<31:0>      = GPR[rt]<63:32>
                 HASHDAT[13]<31:0>      = GPR[rt]<31:0>
                 HASHDAT[12..13]<63:32> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

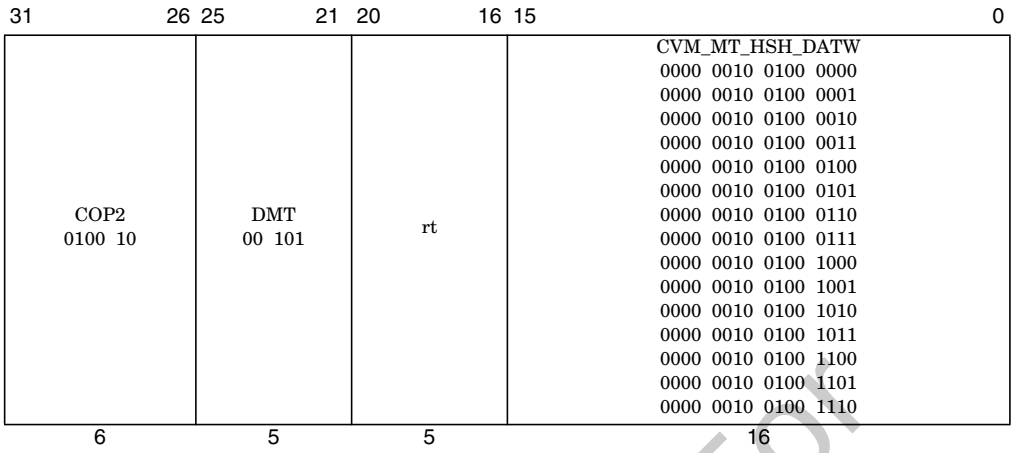
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Load Data into HSH Unit (wide mode) CVM_MT_HSH_DATW



Note: These instructions 0–7 are the same as Load LFSR to SNOW3G Unit and ZUC Unit (CVM_MT_SNOW3G_LFSR 0–7 and CVM_MT_ZUC_LFSR 0–7, respectively) instructions.

Format: DMTC2 rt, 0x0240 **CVM**

DMTC2 rt, 0x0241
DMTC2 rt, 0x0242
DMTC2 rt, 0x0243
DMTC2 rt, 0x0244
DMTC2 rt, 0x0245
DMTC2 rt, 0x0246
DMTC2 rt, 0x0247
DMTC2 rt, 0x0248
DMTC2 rt, 0x0249
DMTC2 rt, 0x024A
DMTC2 rt, 0x024B
DMTC2 rt, 0x024C
DMTC2 rt, 0x024D
DMTC2 rt, 0x024E

Purpose:

To load values into HASHDAT via the wide mechanism. The SHA-512 algorithm uses the wide HASHDAT format.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Description:

```
HASHDAT[0]  = rt // DMTC2 rt,0x0240
HASHDAT[1]  = rt // DMTC2 rt,0x0241
HASHDAT[2]  = rt // DMTC2 rt,0x0242
HASHDAT[3]  = rt // DMTC2 rt,0x0243
HASHDAT[4]  = rt // DMTC2 rt,0x0244
HASHDAT[5]  = rt // DMTC2 rt,0x0245
HASHDAT[6]  = rt // DMTC2 rt,0x0246
HASHDAT[7]  = rt // DMTC2 rt,0x0247
HASHDAT[8]  = rt // DMTC2 rt,0x0248
HASHDAT[9]  = rt // DMTC2 rt,0x0249
HASHDAT[10] = rt // DMTC2 rt,0x024A
HASHDAT[11] = rt // DMTC2 rt,0x024B
HASHDAT[12] = rt // DMTC2 rt,0x024C
HASHDAT[13] = rt // DMTC2 rt,0x024D
HASHDAT[14] = rt // DMTC2 rt,0x024E
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0240: HASHDAT[ 0] = GPR[rt]
    case 0x0241: HASHDAT[ 1] = GPR[rt]
    case 0x0242: HASHDAT[ 2] = GPR[rt]
    case 0x0243: HASHDAT[ 3] = GPR[rt]
    case 0x0243: HASHDAT[ 4] = GPR[rt]
    case 0x0245: HASHDAT[ 5] = GPR[rt]
    case 0x0246: HASHDAT[ 6] = GPR[rt]
    case 0x0247: HASHDAT[ 7] = GPR[rt]
    case 0x0248: HASHDAT[ 8] = GPR[rt]
    case 0x0249: HASHDAT[ 9] = GPR[rt]
    case 0x024a: HASHDAT[10] = GPR[rt]
    case 0x024b: HASHDAT[11] = GPR[rt]
    case 0x024c: HASHDAT[12] = GPR[rt]
    case 0x024d: HASHDAT[13] = GPR[rt]
    case 0x024e: HASHDAT[14] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

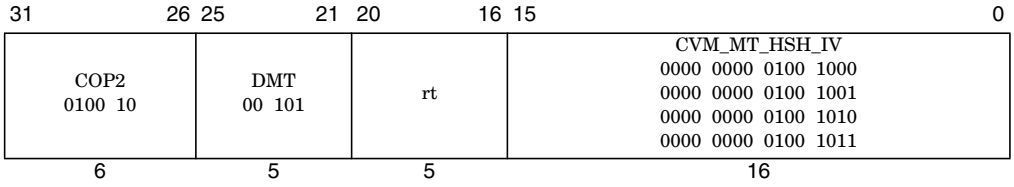
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load IV into HSH Unit (narrow mode) **CVM_MT_HSH_IV**



Format: DMTC2 rt, 0x0048 **CVM**

DMTC2 rt, 0x0049

DMTC2 rt, 0x004A

DMTC2 rt, 0x004B

Purpose:

To load values into HASHIV in the narrow mode. MD5, SHA-1, and SHA-256 use the narrow mode. The least-significant 32 bits of separate 64-bit HASHDAT double-words are written with the most-significant and least-significant 32 bits of rt.

Description:
```
HASHDAT[0]<31:0> || HASHDAT[1]<31:0> = rt // DMTC2 rt,0x0048
HASHDAT[2]<31:0> || HASHDAT[3]<31:0> = rt // DMTC2 rt,0x0049
HASHDAT[4]<31:0> || HASHDAT[5]<31:0> = rt // DMTC2 rt,0x004A
HASHDAT[6]<31:0> || HASHDAT[7]<31:0> = rt // DMTC2 rt,0x004B
```

Restrictions:

If this instruction follows a CVM_MT_HSH_STARTMD5, CVM_MT_HSH_STARTSHA, CVM_MT_HSH_STARTSHA256, or CVM_MT_HSH_STARTSHA512 instruction without a CVM_MF_HSH_IV* instruction between the two, the HASHIV result of the previous CVM_MT_HSH_START* instruction is unpredictable (but this CVM_MT_HSH_IV* instruction will successfully complete).

This instruction renders the value of any results from SNOW3G and ZUC instructions unpredictable. See the programming notes of CVM_MF_SNOW3G_LFSR, CVM_MF_SNOW3G_FSM, CVM_MF_SNOW3G_RESULT, CVM_MT_SNOW3G_LFSR, CVM_MT_SNOW3G_FSM, and CVM_MT_SNOW3G_RESULT in section SNOW 3G on page 1835 and CVM_MF_ZUC_LFSR, CVM_MF_ZUC_FSM, CVM_MF_ZUC_RESULT, CVM_MF_ZUC_TRESULT, CVM_MT_ZUC_LFSR, CVM_MT_ZUC_FSM, and CVM_MT_ZUC_RESULT in section ZUC on page 1877.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Cavium Confidential For Review Only Aug 2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0048: HASHIV[0]<31:0>     = GPR[rt]<63:32>
                 HASHIV[1]<31:0>     = GPR[rt]<31:0>
                 HASHIV[1..0]<63:32> = unpredictable
    case 0x0049: HASHIV[2]<31:0>     = GPR[rt]<63:32>
                 HASHIV[3]<31:0>     = GPR[rt]<31:0>
                 HASHIV[3..2]<63:32> = unpredictable
    case 0x004a: HASHIV[4]<31:0>     = GPR[rt]<63:32>
                 HASHIV[5]<31:0>     = GPR[rt]<31:0>
                 HASHIV[5..4]<63:32> = unpredictable
    case 0x004a: HASHIV[6]<31:0>     = GPR[rt]<63:32>
                 HASHIV[7]<31:0>     = GPR[rt]<31:0>
                 HASHIV[7..6]<63:32> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

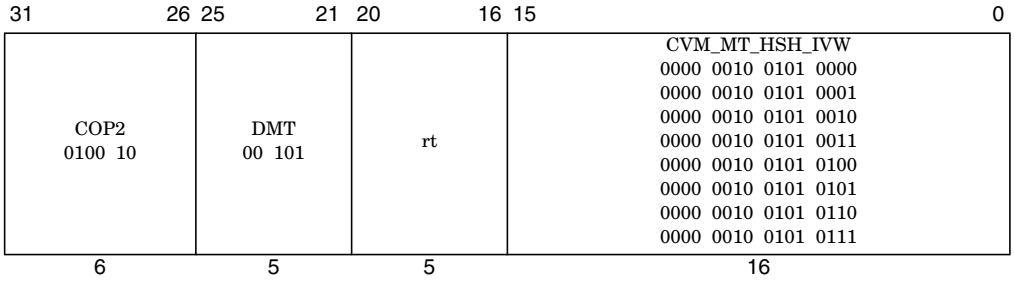
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Load IV into HSH Unit (wide mode) **CVM_MT_HSH_IVW**



Note: This instruction 0 is the same as Load Result to SNOW 3G Unit and ZUC Unit (CVM_MT_SNOW3G_RESULT and CVM_MT_ZUC_RESULT, respectively) instructions.

These instructions 1-2 are the same as Load FSM to SNOW3G Unit and ZUC Unit (CVM_MT_SNOW3G_FSM 0-1 and CVM_MT_ZUC_FSM 0-1, respectively) instructions.

This instruction 3 is the same as CVM_MT_SNOW3G_FSM2.

Format: DMTC2 rt, 0x0250 **CVM**

DMTC2 rt, 0x0251

DMTC2 rt, 0x0252

DMTC2 rt, 0x0253

DMTC2 rt, 0x0254

DMTC2 rt, 0x0255

DMTC2 rt, 0x0256

DMTC2 rt, 0x0257

Purpose:

To load values into HASHIV in the wide mode. SHA-512 uses the wide mode.

Description:

```
HASHIV[0] = rt // DMTC2 rt,0x0250
HASHIV[1] = rt // DMTC2 rt,0x0251
HASHIV[2] = rt // DMTC2 rt,0x0252
HASHIV[3] = rt // DMTC2 rt,0x0253
HASHIV[4] = rt // DMTC2 rt,0x0254
HASHIV[5] = rt // DMTC2 rt,0x0255
HASHIV[6] = rt // DMTC2 rt,0x0256
HASHIV[7] = rt // DMTC2 rt,0x0257
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

Restrictions:

If this instruction follows a CVM_MT_HSH_STARTMD5, CVM_MT_HSH_STARTSHA, CVM_MT_HSH_STARTSHA256, or CVM_MT_HSH_STARTSHA512 instruction without a CVM_MF_HSH_IV* instruction between the two, the HASHIV result of the previous CVM_MT_HSH_START* instruction is unpredictable (but this CVM_MT_HSH_IV* instruction will successfully complete).

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0250: HASHIV[0] = GPR[rt]
    case 0x0251: HASHIV[1] = GPR[rt]
    case 0x0252: HASHIV[2] = GPR[rt]
    case 0x0253: HASHIV[3] = GPR[rt]
    case 0x0254: HASHIV[4] = GPR[rt]
    case 0x0255: HASHIV[5] = GPR[rt]
    case 0x0256: HASHIV[6] = GPR[rt]
    case 0x0257: HASHIV[7] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

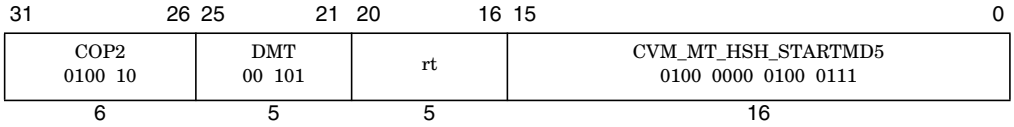
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

MD5 Hash **CVM_MT_HSH_STARTMD5**



Format: DMTC2 rt, 0x4047 **CVM**

Purpose:

To start an MD5 hash. The least-significant 32 bits of the first 14 HASHDAT words together with rt is hashed using the IV from the least-significant 32 bits of the first four HASHIV words. The least-significant 32 bits of the first four HASHIV words are updated with the result (in narrow mode).

Description: HASHIV[3:0]<31:0> = MD5(rt || HASHDAT[13:0]<31:0>, HASHIV[3:0]<31:0>)
Restrictions:

This instruction returns unpredictable results after a SNOW3G and a ZUC operation and renders the value of any results from SNOW3G and ZUC instructions unpredictable. See the programming notes of CVM_MF_SNOW3G_LFSR, CVM_MF_SNOW3G_FSM, CVM_MF_SNOW3G_RESULT, CVM_MT_SNOW3G_LFSR, CVM_MT_SNOW3G_FSM, and CVM_MT_SNOW3G_RESULT in section SNOW 3G on page 1835 and and CVM_MF_ZUC_LFSR, CVM_MF_ZUC_FSM, CVM_MF_ZUC_RESULT, CVM_MF_ZUC_TRESULT, CVM_MT_ZUC_LFSR, CVM_MT_ZUC_FSM, and CVM_MT_ZUC_RESULT in section ZUC on page 1877.

HASHDAT[0..13]<31:0> must be predictable. This means that HASHDAT[0..13]<31:0> must be written between execution of this CVM_MT_HSH_STARTMD5 and the execution of the immediately preceding CVM_MT_HSH_STARTMD5, CVM_MT_HSH_STARTSHA, CVM_MT_HSH_STARTSHA256, or CVM_MT_HSH_STARTSHA512.

All of HASHIV[3:0]<31:0> must be predictable prior to this CVM_MT_HSH_STARTMD5.

If this instruction is followed by a CVM_MT_HSH_IV* instruction without any CVM_MF_HSH_IV* instructions between the two, the HASHIV results of this instruction are unpredictable.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Cavium Confidential For Bowei Yang ALD 02/27/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    for(i = 0; i < 2; i++)
      TIV[i]<63:0> = HASHIV[2*i]<31:0> || HASHIV[2*i+1]<31:0>
    for(i = 0; i < 7; i++)
      TDAT[i]<63:0> = HASHDAT[2*i]<31:0> || HASHDAT[2*i+1]<31:0>
    TDAT[7]<63:0> = GPR[rt]
    TIV = MD5(TDAT, TIV)
    for(i = 0; i < 2; i++) {
      HASHIV[2*i]<31:0>   = TIV[i]<63:32>
      HASHIV[2*i+1]<31:0> = TIV[i]<31:0>
    }
    HASHIV[3..0]<63:32> = unpredictable
    HASHDAT[14..0]<63:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

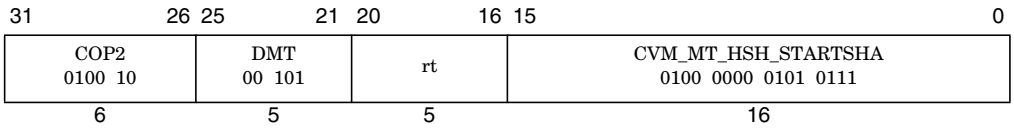
Reserved Instruction

Notes:

The MD5 algorithm is little-endian and the HASHIV and HASHDAT data are big-endian, so both HASHIV and HASHDAT are byte-swapped in and out of the MD5() function above.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

SHA-1 Hash **CVM_MT_HSH_STARTSHA**



Format: DMTC2 rt, 0x4057 **CVM**

Purpose:

To start a SHA-1 hash. The least-significant 32 bits of the first 14 HASHDAT words together with rt is hashed using the IV from the least-significant 32 bits of the first five HASHIV words. The least-significant 32 bits of the first five HASHIV words are updated with the result (in narrow mode).

Description: HASHIV[4:0]<31:0> = SHA-1(rt || HASHDAT[13:0]<31:0>, HASHIV[4:0]<31:0>)
Restrictions:

This instruction returns unpredictable results after a SNOW3G and ZUC operation and renders the value of any results from SNOW3G and ZUC instructions unpredictable. See the programming notes of CVM_MF_SNOW3G_LFSR, CVM_MF_SNOW3G_FSM, CVM_MF_SNOW3G_RESULT, CVM_MT_SNOW3G_LFSR, CVM_MT_SNOW3G_FSM, and CVM_MT_SNOW3G_RESULT in section SNOW 3G on page 1835 and and CVM_MF_ZUC_LFSR, CVM_MF_ZUC_FSM, CVM_MF_ZUC_RESULT, CVM_MF_ZUC_TRESULT, CVM_MT_ZUC_LFSR, CVM_MT_ZUC_FSM, and CVM_MT_ZUC_RESULT in section ZUC on page 1877.

HASHDAT[13:0]<31:0> must be predictable. This means that HASHDAT[13:0]<31:0> must be written between execution of this CVM_MT_HSH_STARTSHA and the execution of the immediately preceding CVM_MT_HSH_STARTMD5, CVM_MT_HSH_STARTSHA, CVM_MT_HSH_STARTSHA256, or CVM_MT_HSH_STARTSHA512.

All of HASHIV[4:0]<31:0> must be predictable prior to this CVM_MT_HSH_STARTSHA.

If this instruction is followed by a CVM_MT_HSH_IV* instruction without any CVM_MF_HSH_IV* instructions between the two, the HASHIV results of this instruction are unpredictable.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Cavium Confidential For Bowei Yang ALI 2015/7/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    for(i = 0; i < 2; i++)
      TIV[i]<63:0> = HASHIV[2*i]<31:0> || HASHIV[2*i+1]<31:0>
    TIV[2]<63:0> = TIV[4]<31:0> || 0^32
    for(i = 0; i < 7; i++)
      TDAT[i]<63:0> = HASHDAT[2*i]<31:0> || HASHDAT[2*i+1]<31:0>
    TDAT[7]<63:0> = GPR[rt]
    TIV = SHA-1(TDAT, TIV)
    for(i = 0; i < 2; i++) {
      HASHIV[2*i]<31:0>   = TIV[i]<63:32>
      HASHIV[2*i+1]<31:0> = TIV[i]<31:0>
    }
    HASHIV[4]<31:0> = TIV[2]<63:32>
    HASHIV[4..0]<63:32> = unpredictable
    HASHDAT[14..0]<63:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

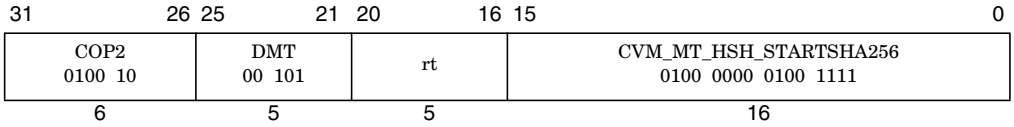
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

SHA-256 Hash **CVM_MT_HSH_STARTSHA256**



Format: DMTC2 rt, 0x404F **CVM**

Purpose:

To start a SHA-256 hash. The least-significant 32 bits of the first 14 HASHDAT words together with rt is hashed using the IV from the least-significant 32 bits of the first five HASHIV words. The least-significant 32 bits of the first five HASHIV words are updated with the result (in narrow mode).

Description: HASHIV[7:0]<31:0> = SHA-256(rt || HASHDAT[13:0]<31:0>, HASHIV[7:0]<31:0>)

Restrictions:

This instruction returns unpredictable results after a SNOW3G and ZUC operation and renders the value of any results from SNOW3G and ZUC instructions unpredictable. See the programming notes of CVM_MF_SNOW3G_LFSR, CVM_MF_SNOW3G_FSM, CVM_MF_SNOW3G_RESULT, CVM_MT_SNOW3G_LFSR, CVM_MT_SNOW3G_FSM, and CVM_MT_SNOW3G_RESULT in section SNOW 3G on page 1835 and CVM_MF_ZUC_LFSR, CVM_MF_ZUC_FSM, CVM_MF_ZUC_RESULT, CVM_MF_ZUC_TRESULT, CVM_MT_ZUC_LFSR, CVM_MT_ZUC_FSM, and CVM_MT_ZUC_RESULT in section ZUC on page 1877.

HASHDAT[13:0]<31:0> must be predictable. This means that HASHDAT[13:0]<31:0> must be written between execution of this CVM_MT_HSH_STARTSHA256 and the execution of the immediately preceding CVM_MT_HSH_STARTMD5, CVM_MT_HSH_STARTSHA, CVM_MT_HSH_STARTSHA256, or CVM_MT_HSH_STARTSHA512.

All of HASHIV[7:0]<31:0> must be predictable prior to this CVM_MT_HSH_STARTSHA256.

If this instruction is followed by a CVM_MT_HSH_IV* instruction without any CVM_MF_HSH_IV* instructions between the two, the HASHIV results of this instruction are unpredictable.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Cavium Confidential For Lowen Yang Only 02/24/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    for(i = 0; i < 4; i++)
      TIV[i]<63:0> = HASHIV[2*i]<31:0> || HASHIV[2*i+1]<31:0>
    for(i = 0; i < 7; i++)
      TDAT[i]<63:0> = HASHDAT[2*i]<31:0> || HASHDAT[2*i+1]<31:0>
    TDAT[7]<63:0> = GPR[rt]
    TIV = SHA-256(TDAT, TIV)
    for(i = 0; i < 4; i++) {
      HASHIV[2*i]<31:0>   = TIV[i]<63:32>
      HASHIV[2*i+1]<31:0> = TIV[i]<31:0>
    }
    HASHIV[7..0]<63:32> = unpredictable
    HASHDAT[14..0]<63:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

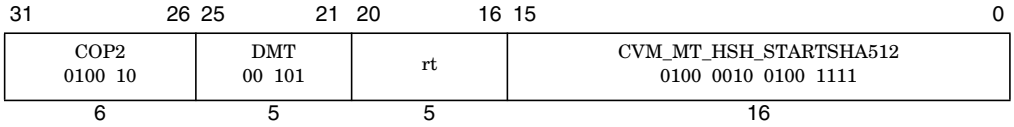
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

SHA-512 Hash **CVM_MT_HSH_STARTSHA512**



Format: DMTC2 rt, 0x424F **CVM**

Purpose:

To start a SHA-512 hash. The HASHDAT words together with rt are hashed using HASHIV. HASHIV is updated with the result in wide mode.

Description: HASHIV[7:0] = SHA-512(rt || HASHDAT[14:0], HASHIV[7:0])

Restrictions:

This instruction returns unpredictable results after a SNOW3G and ZUC operation and renders the value of any results from SNOW3G and ZUC instructions unpredictable. See the programming notes of CVM_MF_SNOW3G_LFSR, CVM_MF_SNOW3G_FSM, CVM_MF_SNOW3G_RESULT, CVM_MT_SNOW3G_LFSR, CVM_MT_SNOW3G_FSM, and CVM_MT_SNOW3G_RESULT in section SNOW 3G on page 1835 and and CVM_MF_ZUC_LFSR, CVM_MF_ZUC_FSM, CVM_MF_ZUC_RESULT, CVM_MF_ZUC_TRESULT, CVM_MT_ZUC_LFSR, CVM_MT_ZUC_FSM, and CVM_MT_ZUC_RESULT in section ZUC on page 1877.

HASHDAT[14:0]<63:0> must be predictable. This means that HASHDAT[14:0] must be written between execution of this CVM_MT_HSH_STARTSHA512 and the execution of the immediately preceding CVM_MT_HSH_STARTMD5, CVM_MT_HSH_STARTSHA, CVM_MT_HSH_STARTSHA256, or CVM_MT_HSH_STARTSHA512.

All of HASHIV[7..0]<31:0> must be predictable prior to this CVM_MT_HSH_STARTSHA512. This means that HASHIV[7:0] must have been written since the last CVM_MT_HSH_STARTMD5, CVM_MT_HSH_STARTSHA, or CVM_MT_HSH_STARTSHA256.

If this instruction is followed by a CVM_MT_HSH_IV* instruction without any CVM_MF_HSH_IV* instructions between the two, the HASHIV results of this instruction are unpredictable.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Cavium Confidential For Development Use Only 02/27/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    HASHIV[7:0] = SHA-512(rt || HASHDAT[14:0], HASHIV[7:0])
    HASHDAT[14:0]<63:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

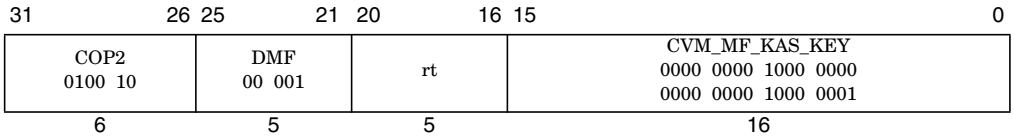
Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

KASUMI

Load Key from KASUMI Unit — CVM_MF_KAS_KEY



Note: These instructions are the same as Load Key from 3DES Unit instructions (CVM_MF_3DES_KEY), except that 3DESKEY[2:0] are used by the 3DES unit and only 3DESKEY[1:0] are used by the KASUMI unit.

Format: DMFC2 rt, 0x0080 **CVM**

DMFC2 rt, 0x0081

Purpose:

To read 3DESKEY values from the KASUMI coprocessor. 3DESKEY[1:0] are used by the KASUMI unit.

Description:
```
rt = 3DESKEY[0]  // DMFC2 rt, 0x0080
rt = 3DESKEY[1]  // DMFC2 rt, 0x0081
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0080: GPR[rt] = 3DES_KEY[0]
    case 0x0081: GPR[rt] = 3DES_KEY[1]
  endif
else
  SignalException(CoprocessorUnusable, 2)
 endif
```

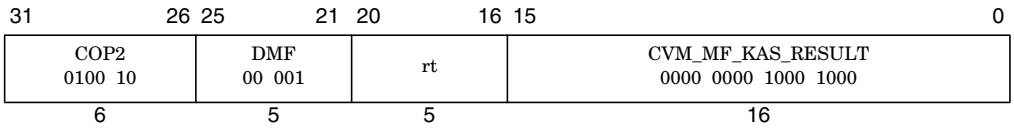
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Result from KASUMI Unit **CVM_MF_KAS_RESULT**



Note: This instruction is the same as Load Result from 3DES Unit (CVM_MF_3DES_RESULT).

Format: DMFC2 rt, 0x0088 **CVM**

Purpose:

To read 3DESRESULT from the KASUMI coprocessor.

Description: rt = 3DESRESULT<63:0>

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
    if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
      SignalException(ReservedInstruction)
    else
      GPR[rt] = 3DESRESULT<63:0>
    endif
  else
    SignalException(CoprocessorUnusable, 2) endif
endif
```

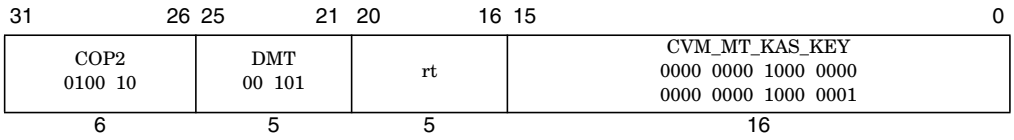
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Key into KASUMI Unit — CVM_MT_KAS_KEY



Note: These instructions are the same as Load Key into 3DES Unit instructions (CVM_MT_3DES_KEY), except that 3DESKEY[2:0] are used by the 3DES unit and only 3DESKEY[1:0] are used by the KASUMI unit.

Format: DMTC2 rt, 0x0080 **CVM**

DMTC2 rt, 0x0081

Purpose:

To load values into 3DESKEY. 3DESKEY[1:0] are used by the KASUMI unit.

Description:
```
3DESKEY[0]  = rt // DMTC2 rt,0x0080
3DESKEY[1]  = rt // DMTC2 rt,0x0081
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0080: 3DESKEY[0] = GPR[rt]
    case 0x0081: 3DESKEY[1] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

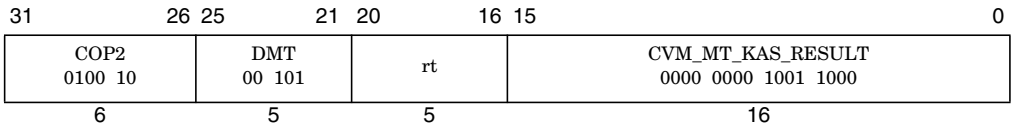
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Result into KASUMI Unit — CVM_MT_KAS_RESULT



Note: This instruction is the same as Load Result into 3DES Unit (CVM_MT_3DES_RESULT).

Format: DMTC2 rt, 0x0098 **CVM**

Purpose:

To load a value into 3DESRESULT.

Description: 3DESRESULT<63:0> = rt

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    3DESRESULT<63:0> = GPR[rt]
    KASUMIRESULT<63:0> = 3DESRESULT<63:0>
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

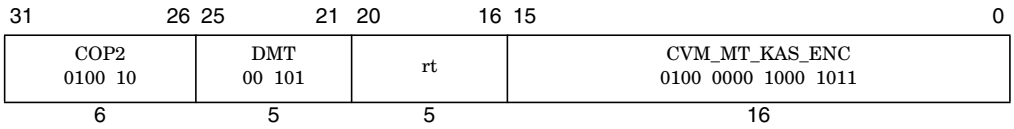
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

KASUMI Encrypt **CVM_MT_KAS_ENC**



Format: DMTC2 rt, 0x408B **CVM**

Purpose:

To do a KASUMI encrypt. GPR[rt] is the input to a KASUMI encrypt (using 3DESKEY[1:0]). 3DESRESULT is set to the result of the encrypt.

Description: KASUMIRESULT<63:0> = 3DESRESULT<63:0> = KASUMI(rt, 3DESKEY[1:0])

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or (not CvmCtl[KASUMI]) then
    SignalException(ReservedInstruction)
  else
    KASUMIRESULT<63:0> = KASUMI(GPR[rt]<63:0>, 3DESKEY[1:0])
    3DESRESULT<63:0> = KASUMIRESULT<63:0>
    3DESKEY[2]<63:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

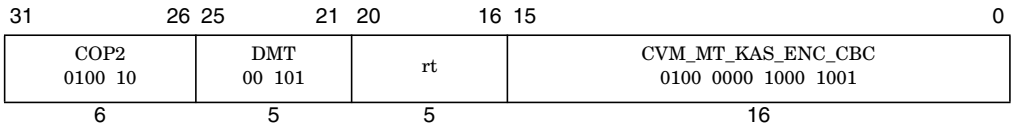
Reserved Instruction

Notes:

The CVM_MT_KAS_ENC instruction first appeared on the OCTEON Plus CN5XXX series parts. The behavior of the CVM_MT_KAS_ENC instruction is not predictable on the earlier OCTEON CN3XXX series parts. CvmCtl[KASUMI] is always 0 on CN3XXX parts, so accurately portrays the lack of a KASUMI unit. However, there is no guarantee of a Reserved Instruction exception when CVM_MT_KAS_ENC is executed. KASUMI code that is portable to both CN5XXX (and successors) and CN3XXX must not execute CVM_MT_KAS_ENC when CvmCtl[KASUMI] is clear.

Cavium Confidential For Bowen Yang ALU 02/27/2014

KASUMI CBC Encrypt — CVM_MT_KAS_ENC_CBC



Format: DMTC2 rt, 0x4089 **CVM**

Purpose:

To do a KASUMI CBC encrypt. GPR[rt] XOR KASUMIRESULT is the input to a KASUMI encrypt (using 3DESKEY[1:0]). 3DESRESULT is set to the result of the encrypt.

Description: KASUMIRESULT<63:0> = 3DESRESULT<63:0> = KASUMI(rt ⊕ KASUMIRESULT, 3DESKEY[1:0])

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or (not CvmCtl[KASUMI]) then
    SignalException(ReservedInstruction)
  else
    KASUMIRESULT<63:0> = KASUMI(GPR[rt]<63:0> ⊕ KASUMIRESULT<63:0>, 3DESKEY[1:0])
    3DESRESULT<63:0> = KASUMIRESULT<63:0>
    3DESKEY[2]<63:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

Reserved Instruction

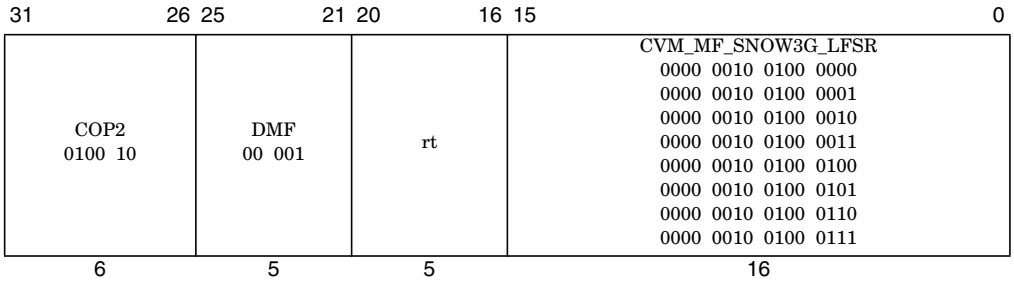
Notes:

The CVM_MT_KAS_ENC_CBC instruction first appeared on the OCTEON Plus CN5XXX series parts. The behavior of the CVM_MT_KAS_ENC_CBC instruction is not predictable on the earlier OCTEON CN3XXX series parts. CvmCtl[KASUMI] is always 0 on CN3XXX parts, so accurately portrays the lack of a KASUMI unit. However, there is no guarantee of a Reserved Instruction exception when CVM_MT_KAS_ENC_CBC is executed. KASUMI code that is portable to both CN5XXX (and successors) and CN3XXX must not execute CVM_MT_KAS_ENC_CBC when CvmCtl[KASUMI] is clear.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SNOW 3G

Load LFSR from SNOW 3G Unit **CVM_MF_SNOW3G_LFSR**



Note: These instructions are the same as Load Data from HSH Unit (wide mode) and ZUC instructions (CVM_MF_HSH_DATW 0–7 and CVM_MF_ZUC_LFSR 0–7, respectively).

Format: DMFC2 rt, 0x0240 **CVMv2**
DMFC2 rt, 0x0241
DMFC2 rt, 0x0242
DMFC2 rt, 0x0243
DMFC2 rt, 0x0244
DMFC2 rt, 0x0245
DMFC2 rt, 0x0246
DMFC2 rt, 0x0247

Purpose:

To read the LFSR from the SNOW 3G coprocessor.

Description:
```
rt = HASHDAT[0]   // DMTC2 rt,0x0240
rt = HASHDAT[1]   // DMTC2 rt,0x0241
rt = HASHDAT[2]   // DMTC2 rt,0x0242
rt = HASHDAT[3]   // DMTC2 rt,0x0243
rt = HASHDAT[4]   // DMTC2 rt,0x0244
rt = HASHDAT[5]   // DMTC2 rt,0x0245
rt = HASHDAT[6]   // DMTC2 rt,0x0246
rt = HASHDAT[7]   // DMTC2 rt,0x0247
```

When executing SNOW3G, HASHDAT[i] has SNOW3G LFSR_S(2×i) in <63:32>, LFSR_S(2×i+1) in <31:0>. For example, HASHDAT[3] has SNOW3G LFSR_S6 in <63:32>, and LFSR_S7 in <31:0>.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Cavium Confidential For Bowen Yang 05/07/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0040: GPR[rt] = SNOW3GLFSR[0] = HASHDAT[0]
    case 0x0041: GPR[rt] = SNOW3GLFSR[1] = HASHDAT[1]
    case 0x0042: GPR[rt] = SNOW3GLFSR[2] = HASHDAT[2]
    case 0x0043: GPR[rt] = SNOW3GLFSR[3] = HASHDAT[3]
    case 0x0044: GPR[rt] = SNOW3GLFSR[4] = HASHDAT[4]
    case 0x0045: GPR[rt] = SNOW3GLFSR[5] = HASHDAT[5]
    case 0x0046: GPR[rt] = SNOW3GLFSR[6] = HASHDAT[6]
    case 0x0047: GPR[rt] = SNOW3GLFSR[7] = HASHDAT[7]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

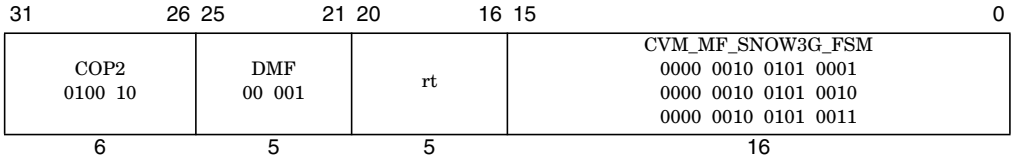
Reserved Instruction

Programming Notes:

CVM_MF_SNOW3G_LFSR may be rarely used when executing SNOW3G - typically only to save SNOW3G state.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Load FSM from SNOW3G Unit — CVM_MF_SNOW3G_FSM



Note: These instructions are the same as Load IV from HSH Unit (wide mode) and ZUC Unit instructions (CVM_MF_HSH_IVW 1-3, CVM_MF_ZUC_FSM 0-1, and CVM_MF_ZUC_TRESULT).

Format: DMFC2 rt, 0x0251 **CVMv2**

DMFC2 rt, 0x0252

DMFC2 rt, 0x0253

Purpose:

To read the FSM from the SNOW 3G coprocessor.

Description:

```
rt = HASHIV[1] // DMFC2 rt, 0x0251
rt = HASHIV[2] // DMFC2 rt, 0x0252
rt = HASHIV[3] // DMFC2 rt, 0x0253
```

When executing SNOW3G, HASHDAT[i] has SNOW3G FSM_Ri in <31:0> and bits <63:32> are unpredictable. For example, HASHIV[2] has SNOW3G FSM_R2 in <31:0>.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0251: GPR[rt] = HASHIV[1]
    case 0x0252: GPR[rt] = HASHIV[2]
    case 0x0253: GPR[rt] = HASHIV[3]

  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

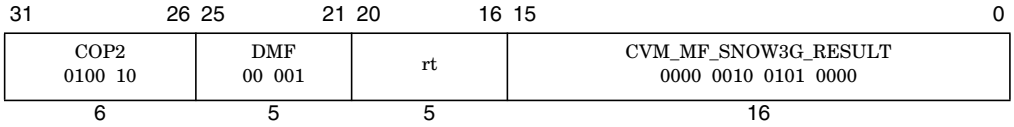
Coprocessor Unusable, Reserved Instruction

Programming Notes:

CVM_MF_SNOW3G_FSM may be rarely used when executing SNOW3G - typically only to save SNOW3G state.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Result from SNOW3G Unit **CVM_MF_SNOW3G_RESULT**



Note: This instruction is the same as Load IV from HSH Unit (wide mode) and ZUC Unit instructions (CVM_MF_HSH_IVW 0 and CVM_MF_ZUC_RESULT, respectively).

Format: DMFC2 rt, 0x0250 **CVMv2**

Purpose:

To read the 64 bits of SNOW3 keystream from the SNOW3G coprocessor.

Description: rt = HASHIV[0] // DMFC2 rt, 0x0250

When executing SNOW3G, HASHIV[0] has the last two 32-bit keystream words produced by the hardware. HASHIV[0]<63:32> contains the earlier 32-bit keystream word, and HASHIV[0]<31:0> contains the later 32-bit keystream word.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0250: GPR[rt] = HASHIV[0]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

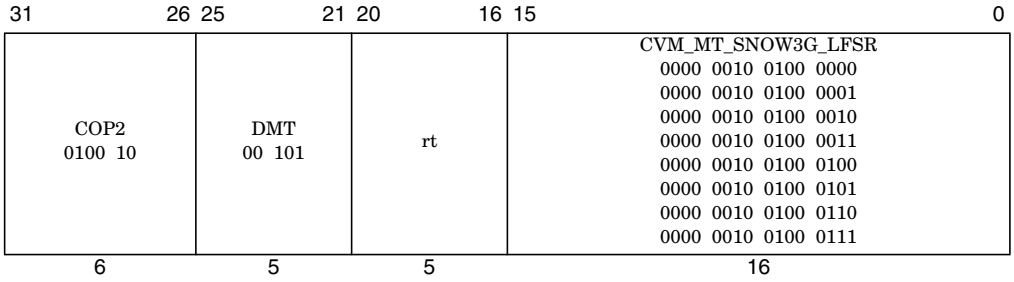
Coprocessor Unusable, Reserved Instruction

Programming Notes:

CVM_MF_SNOW3G_RESULT is commonly used when executing SNOW3G to extract the SNOW3G keystream words.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load LFSR to SNOW3G Unit — CVM_MT_SNOW3G_LFSR



Note: These instructions are the same as Load Data to HSH Unit (wide mode) and ZUC Unit instructions (CVM_MT_HSH_DATW 0–7 and CMV_MT_ZUC_LFSR 0–7, respectively).

Format: DMTC2 rt, 0x0240 **CVMv2**

DMTC2 rt, 0x0241

DMTC2 rt, 0x0242

DMTC2 rt, 0x0243

DMTC2 rt, 0x0244

DMTC2 rt, 0x0245

DMTC2 rt, 0x0246

DMTC2 rt, 0x0247

Purpose:

To load values into the SNOW3G LFSR.

Description:

```
HASHDAT[0] = rt // DMTC2 rt,0x0240
HASHDAT[1] = rt // DMTC2 rt,0x0241
HASHDAT[2] = rt // DMTC2 rt,0x0242
HASHDAT[3] = rt // DMTC2 rt,0x0243
HASHDAT[4] = rt // DMTC2 rt,0x0244
HASHDAT[5] = rt // DMTC2 rt,0x0245
HASHDAT[6] = rt // DMTC2 rt,0x0246
HASHDAT[7] = rt // DMTC2 rt,0x0247
```

When executing SNOW3G, HASHDAT[i] has SNOW3G LFSR_S(2×i) in <63:32>, LFSR_S(2×i+1) in <31:0>. For example, HASHDAT[3] has SNOW3G LFSR_S6 in <63:32>, and LFSR_S7 in <31:0>.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0240: HASHDAT[0] = GPR[rt]
    case 0x0241: HASHDAT[1] = GPR[rt]
    case 0x0242: HASHDAT[2] = GPR[rt]
    case 0x0243: HASHDAT[3] = GPR[rt]
    case 0x0244: HASHDAT[4] = GPR[rt]
    case 0x0245: HASHDAT[5] = GPR[rt]
    case 0x0246: HASHDAT[6] = GPR[rt]
    case 0x0247: HASHDAT[7] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Programming Notes:

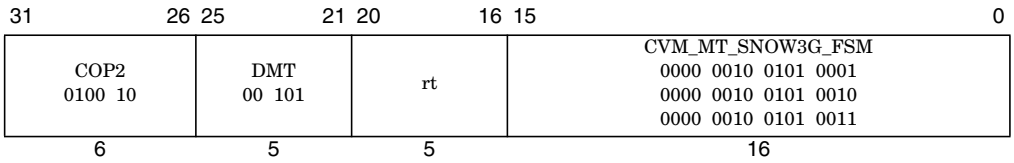
Applications commonly use CVM_MT_SNOW3G_LFSR to initialize the SNOW3G LFSR in before initiating SNOW3G keystream generation. This initialization must follow the convention established in the SNOW3G specifications. For example, the following code can initialize and instruct the hardware to generate the first two 32-bit SNOW3G keywords in as few as nine cycles:

```
// assume $6 has  k0 in <63:32>,  k1 in <31:0>
// assume $7 has  k2 in <63:32>,  k3 in <31:0>
// assume $9 has IV0 in <63:32>, IV1 in <31:0>
// assume $8 has IV2 in <63:32>, IV3 in <31:0>
dmtc2   $6, 0x0242     // CVM_MT_SNOW3G_LFSR(2) ... LFSR_S4 =  k0,     LFSR_S5 =  k1
nor     $2, $0, $6
dmtc2   $2, 0x0240     // CVM_MT_SNOW3G_LFSR(0) ... LFSR_S0 = ~k0,     LFSR_S1 = ~k1
nor     $3, $0, $7
dmtc2   $7, 0x0243     // CVM_MT_SNOW3G_LFSR(3) ... LFSR_S6 =  k2,     LFSR_S7 =  k3
dext    $4, $8, 0, 32
dmtc2   $3, 0x0241     // CVM_MT_SNOW3G_LFSR(1) ... LFSR_S2 = ~k2,     LFSR_S3 = ~k3
xor     $2, $2, $4
dins    $8, $0, 0, 32
dmtc2   $2, 0x0244     // CVM_MT_SNOW3G_LFSR(4) ... LFSR_S8 = ~k0,     LFSR_S9 = ~k1+IV3
xor     $3, $3, $8
dsll32  $2, $9, 0
dmtc2   $3, 0x0245     // CVM_MT_SNOW3G_LFSR(5) ... LFSR_S10= ~k2+IV2, LFSR_S11= ~k3
xor     $6, $6, $2
dsrl32  $9, $9, 0
dmtc2   $6, 0x0246     // CVM_MT_SNOW3G_LFSR(6) ... LFSR_S12=  k0+IV1, LFSR_S13=  k1
xor     $7, $7, $9
dmtc2   $7, 0x404d     // CVM_MT_SNOW3G_START   ... LFSR_S14=  k2,     LFSR_S15=  k3+IV0
```

After a sequence like this, the first two 32-bit SNOW3G keystream words are present in HASHIV[0] and can be extracted via a CVM_MF_SNOW3G_RESULT.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load FSM to SNOW3G Unit — CVM_MT_SNOW3G_FSM



Note: The first two instructions are the same as Load IV to HSH Unit (wide mode) and ZUC Unit instructions (CVM_MT_HSH_IVW 1-2 and CVM_MT_ZUC_FSM 0-1). The last instruction is the same as CVM_MT_HSH_IVW 3.

Format: DMTC2 rt, 0x0251 **CVMv2**

DMTC2 rt, 0x0252

DMTC2 rt, 0x0253

Purpose:

To load values into SNOW3G FSM.

Description:

```
HASHIV[1] = rt  // DMTC2 rt, 0x0251
HASHIV[2] = rt  // DMTC2 rt, 0x0252
HASHIV[3] = rt  // DMTC2 rt, 0x0253
```

When executing SNOW3G, HASHDAT[i] has SNOW3G FSM_Ri in <31:0> and bits <63:32> are unpredictable. For example, HASHIV[2] has SNOW3G FSM_R2 in <31:0>.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0251: HASHIV[1] = GPR[rt]
    case 0x0252: HASHIV[2] = GPR[rt]
    case 0x0253: HASHIV[3] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

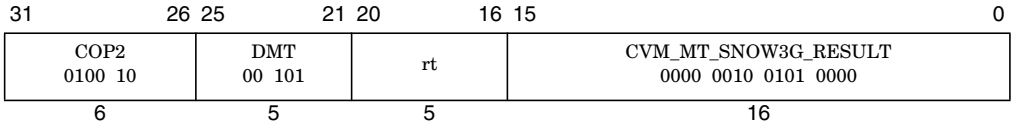
Coprocessor Unusable, Reserved Instruction

Programming Notes:

CVM_MT_SNOW3G_FSM may be rarely used when executing SNOW3G - typically only to restore SNOW3G state.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Data to SNOW3G Unit **CVM_MT_SNOW3G_RESULT**



Note: This instruction is the same as Load Data to HSH Unit (wide mode) and ZUC Unit instructions (CVM_MT_HSH_IVW 0 and CMV_MT_ZUC_RESULT, respectively).

Format: DMTC2 rt, 0x0250 **CVMv2**

Purpose:

To load a value into SNOW3GRESULT.

Description: HASHIV[0] = rt // DMTC2 rt, 0x0250

When executing SNOW3G, HASHIV[0] has the last two 32-bit keystream words produced by the hardware. HASHIV[0]<63:32> contains the earlier 32-bit keystream word, and HASHIV[0]<31:0> contains the later 32-bit keystream word.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0250: HASHIV[0] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

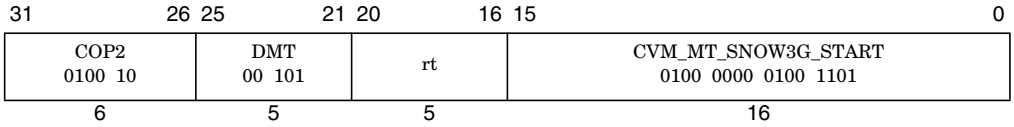
Reserved Instruction

Programming Notes:

CVM_MT_SNOW3G_RESULT may be rarely used when executing SNOW3G - typically only to restore SNOW3G state.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Start SNOW3G Keystream **CVM_MT_SNOW3G_START**



Format: DMFC2 rt, 0x404D **CVMv2**

Purpose:

To complete SNOW3G initialization and compute the first two 32-bit words of SNOW3G keystream. The contents of GPR rt are put into HASHDAT[7]. (HASHDAT[7]<63:32> has the SNOW3G LFSR_S14 state, and HASHDAT[7]<31:0> has the SNOW3G LFSR_S15 state.) HASHIV[1:3]<31:0> are zeroed. (HASHIV[1:3]<31:0> has the SNOW3G FSM_R1, FSM_R2, and FSM_R3, respectively.) The remaining SNOW3G initialization sequence is executed, and then the keystream result is generated. (HASHIV[0]<63:32> has the first 32-bit keystream word and HASHIV[0]<31:0> has the second 32-bit keystream word.)

Description:
```
HASHDAT[7] = rt
HASHIV[1:3]<31:0> = 0
SNOW3G_Initialize()
HASHIV[0]<63:32> = SNOW3G_Loop_Iteration()
HASHIV[0]<31:0>  = SNOW3G_Loop_Iteration()
```

Restrictions:

HASHDAT[0:6]<63:0> must have been previously initialized before the CVM_MT_SNOW3G_START, or else the keystream result, LFSR, and FSM become unpredictable.

HASHDAT[8:14]<63:0>, HASHIV[4:7]<63:0>, and HASHIV[1:3]<63:32> are unpredictable after the instruction.

This instruction renders the value of any results of prior hash instructions (CVM_MT_HSH_STARTMD5, CVM_MT_HSH_STARTSHA, CVM_MT_HSH_STARTSHA256, CVM_MT_HSH_STARTSHA512) and prior ZUC instructions (CVM_MT_ZUC_START and CVM_MT_ZUC_MORE) unpredictable.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Cavium Confidential For Bowen Yang 02/21/14

Operation:

```
if IsCoprocessorEnabled(2) then
   if (not Are64bitOperationsEnabled()) or !CvmCtl[SNOW3G] then
      SignalException(ReservedInstruction)
   else
      HASHDAT[7]<63:0> = GPR[rt]<63:0>
      for (i=0; i<8; i++) {
          LFSR_S(2*i)   = HASHDAT[i]<63:32>
          LFSR_S(2*i+1) = HASHDAT[i]<31:0>
      }
      HASHDAT[14:0]<63:0> = unpredictable
      HASHIV[7:0]<63:0> = unpredictable
      // end of SNOW3G spec section 4.1 "Initialization"
      FSM_R1 = FSM_R2 = FSM_R3 = 0
      for (i=0; i<32; i++) {
          F = ClockFSM(FSM_R*, LFSR_S*)         // STEP 1
          ClockLFSRInitMode(F, FSM_R*, LFSR_S*) // STEP 2
      }
      // first paragraph of SNOW3G spec section 4.2 "Generation of Keystream"
      ClockFSM(FSM_R*, LFSR_S*)
      ClockLFSRKeyStreamMode(FSM_R*, LFSR_S*)
      // two loops from SNOW3G spec section 4.2 "Generation of Keystream"
      for (i=0; i<2; i++) {
          F = ClockFSM(FSM_R*, LFSR_S*)            // STEP 1
          HASHIV[0]<63-i*32:32-i*32> = F XOR LFSR_S0 // STEP 2
          ClockLFSRKeyStreamMode(FSM_R*, LFSR_S*)  // STEP 3
      }
      for (i=0; i<8; i++) {
          HASHDAT[i]<63:32> = LFSR_S(2*i)
          HASHDAT[i]<31:0>  = LFSR_S(2*i+1)
      }
      HASHIV[1]<31:0> = FSM_R1
      HASHIV[2]<31:0> = FSM_R2
      HASHIV[3]<31:0> = FSM_R3
   endif
else
   SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Programming Notes:

Applications commonly use CVM_MT_SNOW3G_START to complete the SNOW3G LFSR initialization and initiate SNOW3G keystream generation. This initialization must follow the convention established in the SNOW3G specifications. For example, the following code can initialize and instruct the hardware to generate the first two 32-bit SNOW3G keywords in as few as nine cycles:

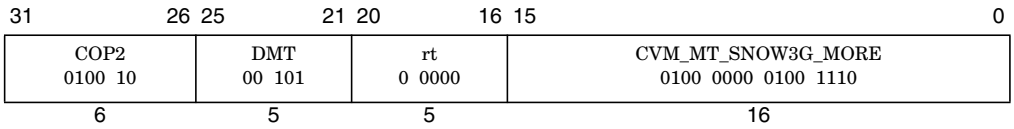
```
// assume $6 has  k0 in <63:32>,  k1 in <31:0>
// assume $7 has  k2 in <63:32>,  k3 in <31:0>
// assume $9 has IV0 in <63:32>, IV1 in <31:0>
// assume $8 has IV2 in <63:32>, IV3 in <31:0>
dmtc2   $6, 0x0242     // CVM_MT_SNOW3G_LFSR(2) ... LFSR_S4 =  k0,     LFSR_S5 =  k1
nor     $2, $0, $6
dmtc2   $2, 0x0240     // CVM_MT_SNOW3G_LFSR(0) ... LFSR_S0 = ~k0,     LFSR_S1 = ~k1
nor     $3, $0, $7
dmtc2   $7, 0x0243     // CVM_MT_SNOW3G_LFSR(3) ... LFSR_S6 =  k2,     LFSR_S7 =  k3
dext    $4, $8, 0, 32
dmtc2   $3, 0x0241     // CVM_MT_SNOW3G_LFSR(1) ... LFSR_S2 = ~k2,     LFSR_S3 = ~k3
xor     $2, $2, $4
dins    $8, $0, 0, 32
dmtc2   $2, 0x0244     // CVM_MT_SNOW3G_LFSR(4) ... LFSR_S8 = ~k0,     LFSR_S9 = ~k1+IV3
xor     $3, $3, $8
dsll32  $2, $9, 0
dmtc2   $3, 0x0245     // CVM_MT_SNOW3G_LFSR(5) ... LFSR_S10= ~k2+IV2, LFSR_S11= ~k3
xor     $6, $6, $2
dsrl32  $9, $9, 0
dmtc2   $6, 0x0246     // CVM_MT_SNOW3G_LFSR(6) ... LFSR_S12=  k0+IV1, LFSR_S13=  k1
xor     $7, $7, $9
dmtc2   $7, 0x404d     // CVM_MT_SNOW3G_START   ... LFSR_S14=  k2,     LFSR_S15=  k3+IV0
```

After a sequence like this, the first two 32-bit SNOW3G keystream words are present in HASHIV[0] and can be extracted via a CVM_MF_SNOW3G_RESULT.

The CVM_MT_SNOW3G_START instruction first appeared on the CN6XXX series parts. The behavior of the CVM_MT_SNOW3G_START instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX series parts. CvmCtl[SNOW3G] is always 0 on CN3XXX/CN5XXX parts, so accurately portrays the lack of a SNOW3G unit. However, there is no guarantee of a Reserved Instruction exception when CVM_MT_SNOW3G_START is executed on these earlier parts. SNOW3G code that is portable to all of CN6XXX (and successors) and CN3XXX/CN5XXX must not execute CVM_MT_SNOW3G_START when CvmCtl[SNOW3G] is clear.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Continue SNOW3G Keystream **CVM_MT_SNOW3G_MORE**



Format: DMTC2 rt, 0x404E **CVMv2**

Purpose:

To compute two more 32 bit words of SNOW3G keystream.

Description: HASHIV[0]<63:32> = SNOW3G_Loop_Iteration()
HASHIV[0]<31:0> = SNOW3G_Loop_Iteration()

Restrictions:

The result of CVM_MT_SNOW3G_MORE is unpredictable unless it is preceded by a CVM_MT_SNOW3G_START operation or all of the following operations CVM_MT_HSH_DATW 0–7 and CVM_MT_HSH_IVW 1–3 (without any intervening ZUC START or MORE).

HASHDAT[0:7]<63:0> and HASHIV[1:3]<31:0> must have been previously initialized with SNOW3G LFSR and FSM information, respectively, typically via a prior CVM_MT_SNOW3G_START or CVM_MT_SNOW3G_MORE, or else the keystream result, LFSR, and FSM become unpredictable.

HASHDAT[8:14]<63:0>, HASHIV[4:7]<63:0>, and HASHIV[1:3]<63:32> are unpredictable after the instruction.

This instruction renders the value of any results of prior hash instructions (CVM_MT_HSH_STARTMD5, CVM_MT_HSH_STARTSHA, CVM_MT_HSH_STARTSHA256, CVM_MT_HSH_STARTSHA512) unpredictable.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Cavium Confidential For Bowen Yang ALL 03/27/2014

Operation:

```
if IsCoprocessorEnabled(2) then
   if (not Are64bitOperationsEnabled()) or !CvmCtl[SNOW3G] then
      SignalException(ReservedInstruction)
   else
      for (i=0; i<8; i++) {
          LFSR_S(2*i)   = HASHDAT[i]<63:32>
          LFSR_S(2*i+1) = HASHDAT[i]<31:0>
      }
      FSM_R1 = HASHIV[1]<31:0>
      FSM_R2 = HASHIV[2]<31:0>
      FSM_R3 = HASHIV[3]<31:0>
      HASHDAT[14:0]<63:0> = unpredictable
      HASHIV[7:0]<63:0> = unpredictable
      // two loops from SNOW3G spec section 4.2 "Generation of Keystream"
      for (i=0; i<2; i++) {
          F = ClockFSM(FSM_R*, LFSR_S*)              // STEP 1
          HASHIV[0]<63-i*32:32-i*32> = F XOR LFSR_S0 // STEP 2
          ClockLFSRKeyStreamMode(FSM_R*, LFSR_S*)    // STEP 3
      }
      for (i=0; i<8; i++) {
          HASHDAT[i]<63:32> = LFSR_S(2*i)
          HASHDAT[i]<31:0>  = LFSR_S(2*i+1)
      }
      HASHIV[1]<31:0> = FSM_R1
      HASHIV[2]<31:0> = FSM_R2
      HASHIV[3]<31:0> = FSM_R3
   endif
else
   SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Programming Notes:

Applications commonly use CVM_MT_SNOW3G_MORE to obtain 64 bits more of SNOW3G keystream data.

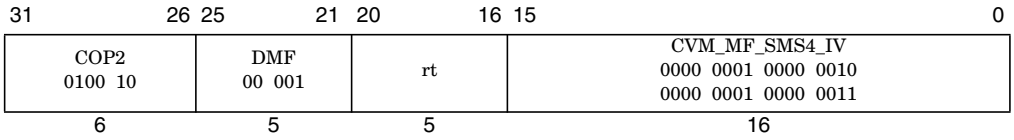
High-performance software may typically execute CVM_MT_SNOW3G_MORE the next cycle after a CVM_MF_SNOW3G_RESULT that extracts the last 64 bits of SNOW3G keystream data.

The CVM_MT_SNOW3G_MORE instruction first appeared on the CN6XXX series parts. The behavior of the CVM_MT_SNOW3G_MORE instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX series parts. CvmCtl[SNOW3G] is always 0 on CN3XXX/CN5XXX parts, so accurately portrays the lack of a SNOW3G unit. However, there is no guarantee of a Reserved Instruction exception when CVM_MT_SNOW3G_MORE is executed on these earlier parts. SNOW3G code that is portable to all of CN6XXX (and successors) and CN3XXX/CN5XXX must not execute CVM_MT_SNOW3G_MORE when CvmCtl[SNOW3G] is clear.

Cavium Confidential For Bowen Yang ALU 2/27/2014

SMS4

Load IV from SMS4 Unit — CVM_MF_SMS4_IV



Note: These instructions are the same as the Load IV from the AES Unit instructions (CVM_MF_AES_IV 0–1).

Format: DMFC2 rt, 0x0102 **CVMv2**

DMFC2 rt, 0x0103

Purpose:

To read AESIV from the SMS4 coprocessor.

Description:
```
rt = AESIV[0]   // DMFC2 rt, 0x0102
rt = AESIV[1]   // DMFC2 rt, 0x0103
```

Restrictions:

CVM_MF_SMS4_IV1 (i.e. “DMFC2 rt, 0x0103”) must not be the first SMS4-related instruction that immediately follows a prior CVM_MT_AES_ENC_CBC1, CVM_MT_AES_DEC_CBC1, CVM_MT_SMS4_ENC_CBC1, or CVM_MT_SMS4_DEC_CBC1 instruction. CVM_MF_SMS4_IV1 should always follow CVM_MF_SMS4_IV0.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0102: GPR[rt] = AESIV[0]
    case 0x0103: GPR[rt] = AESIV[1]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

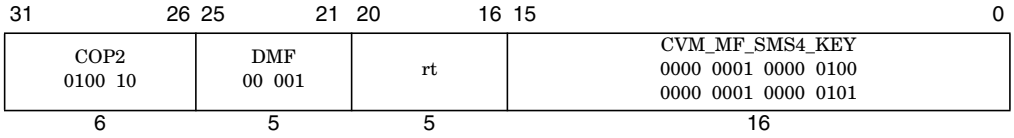
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Kang ALD 02/27/2014

Load Key from SMS4 Unit — CVM_MF_SMS4_KEY



Note: These instructions are the same as Load Key from the AES Unit instructions (CVM_MF_AES_KEY 0–1).

Format: DMFC2 rt, 0x0104 **CVMv2**

DMFC2 rt, 0x0105

Purpose:

To read AESKEY from the SMS4 coprocessor.

Description:
```
rt = AESKEY[0]  // DMFC2 rt, 0x0104
rt = AESKEY[1]  // DMFC2 rt, 0x0105
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0104: GPR[rt] = AESKEY[0]
    case 0x0105: GPR[rt] = AESKEY[1]
  endif
else
  SignalException(CoprocessorUnusable, 2)
 endif
```

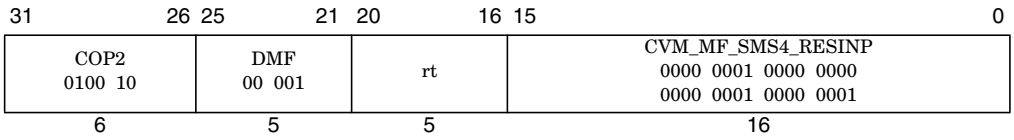
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Result/Input from SMS4 Unit — CVM_MF_SMS4_RESINP



Note: These instructions are the same as Load Result/Input from the AES or CAMELLIA Unit instructions (CVM_MF_AES_RESINP 0–1 and CVM_MF_CAMELLIA_RESINP 0–1, respectively).

Format: DMFC2 rt, 0x0100 **CVMv2**
DMFC2 rt, 0x0101

Purpose:

To read AESRESINP from the SMS4 coprocessor.

Description:

```
rt = AESRESINP[0]  // DMFC2 rt, 0x0100
rt = AESRESINP[1]  // DMFC2 rt, 0x0101
```

Restrictions:

CVM_MF_SMS4_RESINP1 (i.e. “DMFC2 rt, 0x0101”) must not be the first SMS4-related instruction that immediately follows a prior CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC1, CVM_MT_AES_DEC_CBC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC1, or CVM_MT_SMS4_DEC_CBC1 instruction. CVM_MF_SMS4_RESINP1 should always follow CVM_MF_SMS4_RESINP0.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0100: GPR[rt] = AESRESINP[0]
    case 0x0101: GPR[rt] = AESRESINP[1]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

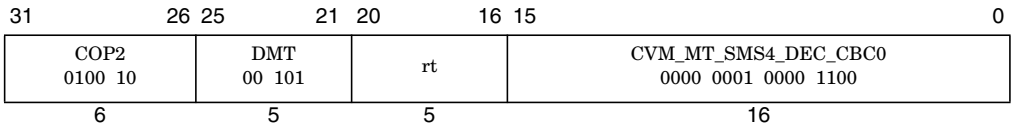
Exceptions:

Coprocessor Unusable

Reserved Instruction

Cavium Confidential For Bowen Hsu ATU 02/27/2014

SMS4 CBC Decrypt (part 1) — CVM_MT_SMS4_DEC_CBC0



Note: This instruction is the same as AES instruction (CVM_MT_AES_DEC_CBC0).

Format: DMTC2 rt, 0x010C **CVMv2**

Purpose:

To load AESRESINP[0] in preparation for a subsequent CVM_MT_SMS4_DEC_CBC1.

Description: AESRESINP[0] = rt

Restrictions:

AESIV[0] must be predictable prior to the CVM_MT_SMS4_DEC_CBC0 instruction and the SMS4 unit cannot use an IV produced by the AES or CAMELLIA unit. This means that AESIV[0] must be written between execution of this CVM_MT_SMS4_DEC_CBC0 and the execution of the immediately preceding CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC1, CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, or CVM_MT_AES_DEC1.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

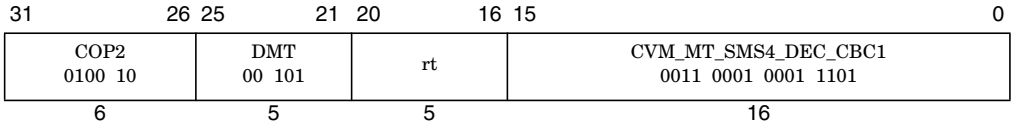
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    AESRESINP[0] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang, ALU 02/27/2014

SMS4 CBC Decrypt (part 2) **CVM_MT_SMS4_DEC_CBC1**



Format: DMTC2 rt, 0x311D **CVMv2**

Purpose:

To perform an SMS4 CBC decrypt operation. GPR[rt] and AESRESINP[0] are the inputs to an SMS4 decryption operation (using AESKEY[1:0] as the key). AESRESINP is updated with the result of the decryption XORed with AESIV. AESIV is set to the inputs of the decryption. AESKEY[3:2] become unpredictable.

Description: TIN[1:0] = rt || AESRESINP[0]
AESRESINP[1:0] = SMS4_DEC(TIN, AESKEY[1:0]) ⊕ AESIV[1:0]
AESIV[1:0] = TIN[1:0]

Restrictions:

CVM_MT_SMS4_DEC_CBC0 (or CVM_MT_SMS4_RESINP0 in the save/restore case) must be executed between the execution of this CVM_MT_SMS4_DEC_CBC1 and the immediately preceding CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1 instruction.

AESIV[1:0] must be predictable for SMS4 prior to this CVM_MT_SMS4_DEC_CBC1 instruction and a CVM_MT_SMS4_DEC_CBC1 cannot use an IV produced by the AES unit. This means that AESIV[1:0] must be written (with CVM_MT_SMS4_IV* instructions) between execution of this CVM_MT_SMS4_DEC_CBC1 and the execution of the immediately preceding CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC1, CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, or CVM_MT_AES_DEC1.

AESKEY[1:0] must be loaded (with CVM_MT_SMS4_KEY* instructions) between this CVM_MT_SMS4_DEC_CBC1 and any prior CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, or CVM_MT_AES_DEC1 instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Cavium Confidential For Bowei Yang 2012/12/21

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or !CvmCtl[SMS4] then
    SignalException(ReservedInstruction)
  else
    TIN[1:0]<63:0> = GPR[rt] || AESRESINP[0]
    AESRESINP[1:0] = SMS4_DEC(TIN[1:0], AESKEY[1:0]) ⊕ AESIV[1:0]
    AESIV[1:0] = TIN[1:0]
    AESKEY[3:2]<63:0> = unpredictable
    AESKEYLEN<1:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

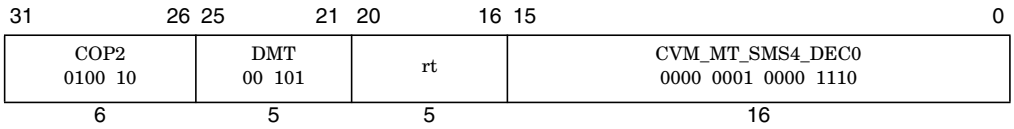
Coprocessor Unusable, Reserved Instruction

Notes:

The CVM_MT_SMS4_DEC_CBC1 instruction first appeared on the CN6XXX series parts. The behavior of the CVM_MT_SMS4_DEC_CBC1 instruction is not predictable on the earlier CN3XXX/CN5XXX series parts. CvmCtl[SMS4] is always 0 on CN3XXX/CN5XXX parts, so accurately portrays the lack of a SMS4 unit. However, there is no guarantee of a Reserved Instruction exception when CVM_MT_SMS4_DEC_CBC1 is executed on these earlier parts. SMS4 code that is portable to all of CN6XXX (and successors) and CN3XXX/CN5XXX must not execute CVM_MT_SMS4_DEC_CBC1 when CvmCtl[SMS4] is clear.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SMS4 Decrypt (part 1) **CVM_MT_SMS4_DEC0**



Note: This instruction is the same as AES instruction (CVM_MT_AES_DEC0).

Format: DMTC2 rt, 0x010E **CVMv2**

Purpose:

To load AESRESINP[0] in preparation for a subsequent CVM_MT_SMS4_DEC1.

Description: AESRESINP[0] = rt

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

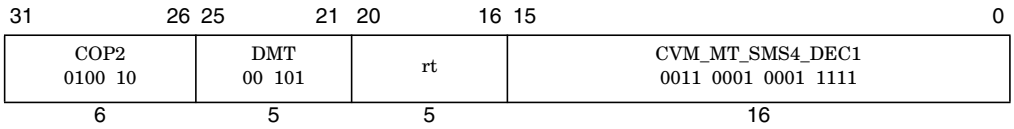
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    AESRESINP[0] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

SMS4 Decrypt (part 2) **CVM_MT_SMS4_DEC1**



Format: DMTC2 rt, 0x311F **CVMv2**

Purpose:

To perform an SMS4 decrypt operation. GPR[rt] and AESRESINP[0] are the inputs to an SMS4 decryption operation (using AESKEY[1:0] as the key). AESRESINP is updated with the result of the decryption. AESKEY[3:2] and AESIV become unpredictable.

Description:
```
TIN[1:0] = rt || AESRESINP[0]
AESRESINP[1:0] = SMS4_DEC(TIN, AESKEY[1:0])
```

Restrictions:

CVM_MT_SMS4_DEC0 (or CVM_MT_SMS4_RESINP0 in the save/restore case) must be executed between the execution of this CVM_MT_SMS4_DEC1 and the immediately preceding CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1 instruction.

AESKEY[1:0] must be loaded (with CVM_MT_SMS4_KEY* instructions) between this CVM_MT_SMS4_DEC1 and any prior CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, or CVM_MT_AES_DEC1 instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Cavium Confidential For Power Long ALD 02/27/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or !CvmCtl[SMS4] then
    SignalException(ReservedInstruction)
  else
    TIN[1:0]<63:0> = GPR[rt] || AESRESINP[0]
    AESRESINP[1:0] = SMS4_DEC(TIN[1:0], AESKEY[1:0])
    AESIV[1:0] = unpredictable
    AESKEY[3:2]<63:0> = unpredictable
    AESKEYLEN<1:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

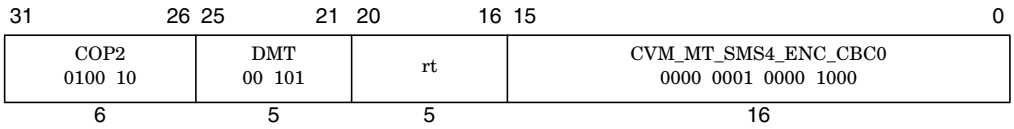
Coprocessor Unusable, Reserved Instruction

Notes:

The CVM_MT_SMS4_DEC1 instruction first appeared on the OCTEON CN6XXX series parts. The behavior of the CVM_MT_SMS4_DEC1 instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX series parts. CvmCtl[SMS4] is always 0 on CN3XXX/CN5XXX parts, so accurately portrays the lack of a SMS4 unit. However, there is no guarantee of a Reserved Instruction exception when CVM_MT_SMS4_DEC1 is executed on these earlier parts. SMS4 code that is portable to all of CN6XXX (and successors) and CN3XXX/CN5XXX must not execute CVM_MT_SMS4_DEC1 when CvmCtl[SMS4] is clear.

Cavium Confidential For Bowen Yang ALU 02/27/2014

SMS4 CBC Encrypt (part 1) **CVM_MT_SMS4_ENC_CBC0**



Note: This instruction is the same as AES instruction (CVM_MT_AES_ENC_CBC0).

Format: DMTC2 rt, 0x0108 **CVMv2**

Purpose:

To load AESRESINP[0] in preparation for a subsequent CVM_MT_SMS4_ENC_CBC1 instruction. The implementation may either deposit (rt XOR AESIV[0]) or rt into AESRESINP[0].

Description: AESRESINP[0] = rt ⊕ AESIV[0], or AESRESINP[0] = rt

Restrictions:

AESIV[0] must be predictable prior to the CVM_MT_SMS4_ENC_CBC0 instruction and the SMS4 unit cannot use an IV produced by the AES unit. This means that AESIV[0] must be written between execution of this CVM_MT_SMS4_ENC_CBC0 and the execution of the immediately preceding CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC1, CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, or CVM_MT_AES_DEC1.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

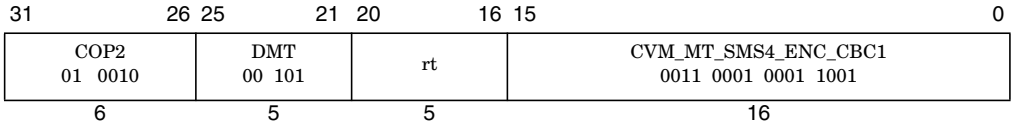
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    AESRESINP[0] = GPR[rt] ⊕ AESIV[0], or
    AESRESINP[0] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang AL U 02/27/2014

SMS4 CBC Encrypt (part 2) **CVM_MT_SMS4_ENC_CBC1**



Format: DMTC2 rt, 0x3119 **CVMv2**

Purpose:

To perform an SMS4 CBC encrypt operation. GPR[rt] XOR AESIV[1] is the second 64-bit input to an SMS4 CBC encryption operation (using AESKEY[1:0] as the key). Either AESRESINP[0] or (AESRESINP[0] XOR AESIV[0]) is the first 64-bit input to the SMS4 operation, depending on whether the CVM_MT_SMS4_ENC_CBC0 implementation pre-XORed AESIV[0] or not, respectively. AESRESINP and AESIV are updated with the result. AESKEY[3:2] become unpredictable.

Description:

AESRESINP[1:0] = AESIV[1..0] =
SMS4_ENC((rt ⊕ AESIV[1]) || AESRESINP[0], AESKEY[1:0]), or
AESRESINP[1:0] = AESIV[1:0] =
SMS4_ENC((rt || AESRESINP[0]) ⊕ AESIV[1:0], AESKEY[1:0])

Restrictions:

CVM_MT_SMS4_ENC_CBC0 (or CVM_MT_SMS4_RESINP0 in the save/restore case) must be executed between the execution of this CVM_MT_SMS4_ENC_CBC1 and the immediately preceding CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1 instruction.

AESIV[1:0] must be predictable for SMS4 prior to this CVM_MT_SMS4_ENC_CBC1 instruction and a CVM_MT_SMS4_ENC_CBC1 cannot use an IV produced by the AES unit. This means that AESIV[1:0] must be written (with CVM_MT_SMS4_IV* instructions) between execution of this CVM_MT_SMS4_ENC_CBC1 and the execution of the immediately preceding CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC1, CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, or CVM_MT_AES_DEC1.

AESKEY[1:0] must be loaded (with CVM_MT_SMS4_KEY* instructions) between this CVM_MT_SMS4_ENC_CBC1 and any prior CVM_MT_SMS4_DEC_CBC1, CVM_MT_SMS4_DEC1, CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, or CVM_MT_AES_DEC1 instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
```

Cavium Confidential For Broadcom Use Only 02-27-2014

```
  if (not Are64bitOperationsEnabled()) or !CvmCtl[SMS4] then
    SignalException(ReservedInstruction)
  else
    TIN[1:0]<63:0> = (GPR[rt] ⊕ AESIV[1]) || AESRESINP[0], or
    TIN[1:0]<63:0> = (GPR[rt] || AESRESINP[0]) ⊕ AESIV[1:0]
    AESRESINP[1:0] = SMS4_ENC(TIN[1:0], AESKEY[<1:0])
    AESIV[1:0] = AESRESINP[1:0]
    AESKEY[3:2]<63:0> = unpredictable
    AESKEYLEN<1:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

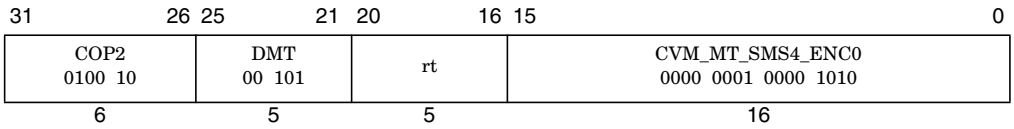
Coprocessor Unusable, Reserved Instruction

Notes:

The CVM_MT_SMS4_ENC_CBC1 instruction first appeared on the OCTEON CN6XXX series parts. The behavior of the CVM_MT_SMS4_ENC_CBC1 instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX series parts. CvmCtl[SMS4] is always 0 on CN3XXX/CN5XXX parts, so accurately portrays the lack of a SMS4 unit. However, there is no guarantee of a Reserved Instruction exception when CVM_MT_SMS4_ENC_CBC1 is executed on these earlier parts. SMS4 code that is portable to all of CN6XXX (and successors) and CN3XXX/CN5XXX must not execute CVM_MT_SMS4_ENC_CBC1 when CvmCtl[SMS4] is clear.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

SMS4 Encrypt (part 1) **CVM_MT_SMS4_ENC0**



Note: This instruction is the same as AES instruction (CVM_MT_AES_ENC0).

Format: DMTC2 rt, 0x010A **CVMv2**

Purpose:

To load AESRESINP[0] in preparation for a subsequent CVM_MT_SMS4_ENC1.

Description: AESRESINP[0] = rt

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

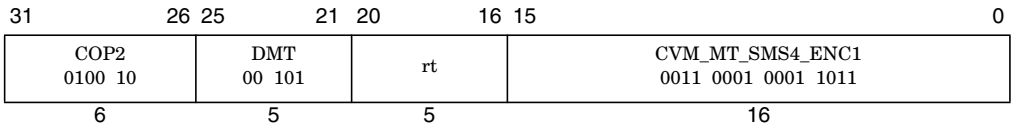
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    AESRESINP[0] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

SMS4 Encrypt (part 2) **CVM_MT_SMS4_ENC1**



Format: DMTC2 rt, 0x311B **CVMv2**

Purpose:

To perform an SMS4 encrypt operation. GPR[rt] and AESRESINP[0] are the input to the SMS4 operation. AESRESINP is updated with the result. AESKEY[3:2] and AESIV become unpredictable.

Description: AESRESINP[1:0] = SMS4_ENC(rt || AESRESINP[0], AESKEY[3:0])

Restrictions:

CVM_MT_SMS4_ENC0 (or CVM_MT_SMS4_RESINP0 in the save/restore case) must be executed between the execution of this CVM_MT_SMS4_ENC1 and the immediately preceding CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, CVM_MT_AES_DEC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC_CBC1, or CVM_MT_SMS4_DEC1 instruction.

AESKEY[1:0] must be loaded (with CVM_MT_SMS4_KEY* instructions) between this CVM_MT_SMS4_ENC1 and any prior CVM_MT_SMS4_DEC_CBC1, CVM_MT_SMS4_DEC1, CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC_CBC1, or CVM_MT_AES_DEC1 instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Cavium Confidential For Bowen Tang ALU 02/27/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or !CvmCtl[SMS4] then
    SignalException(ReservedInstruction)
  else
    TIN[1:0]<63:0> = GPR[rt] || AESRESINP[0]
    AESRESINP[1:0] = SMS4_ENC(TIN[1:0], AESKEY[1:0])
    AESIV[1:0] = unpredictable
    AESKEY[3:2]<63:0> = unpredictable
    AESKEYLEN<1:0> = unpredictable
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

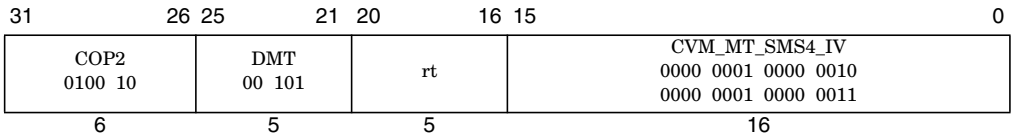
Coprocessor Unusable, Reserved Instruction

Notes:

The CVM_MT_SMS4_ENC1 instruction first appeared on the OCTEON CN6XXX series parts. The behavior of the CVM_MT_SMS4_ENC1 instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX series parts. CvmCtl[SMS4] is always 0 on CN3XXX/CN5XXX parts, so accurately portrays the lack of a SMS4 unit. However, there is no guarantee of a Reserved Instruction exception when CVM_MT_SMS4_ENC1 is executed on these earlier parts. SMS4 code that is portable to all of CN6XXX (and successors) and CN3XXX/CN5XXX must not execute CVM_MT_SMS4_ENC1 when CvmCtl[SMS4] is clear.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load IV into SMS4 Unit — CVM_MT_SMS4_IV



Note: These instructions are the same as Load IV into AES instructions (CVM_MT_AES_IV 0–1).

Format: DMTC2 rt, 0x0102 **CVMv2**

DMTC2 rt, 0x0103

Purpose:

To load AESIV.

Description:

```
AESIV[0] = rt // DMTC2 rt, 0x0102
AESIV[1] = rt // DMTC2 rt, 0x0103
```

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

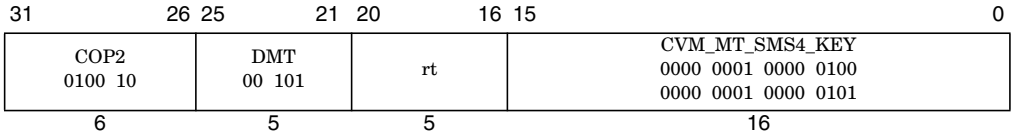
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0102: AESIV[0] = GPR[rt]
    case 0x0103: AESIV[1] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Key into SMS4 Unit — CVM_MT_SMS4_KEY



Note: These instructions are the same as Load Key into AES Unit instructions (CVM_MT_AES_KEY 0–1).

Format: DMTC2 rt, 0x0104 **CVMv2**

DMTC2 rt, 0x0105

Purpose:

To load AESKEY.

Description: AESKEY[0] = rt // DMTC2 rt, 0x0104
AESKEY[1] = rt // DMTC2 rt, 0x0105

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

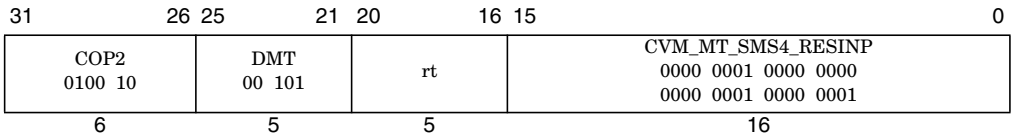
```
if IsCoprocessorEnabled(2) then
   if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
      SignalException(ReservedInstruction)
   else
      case 0x0104: AESKEY[0] = GPR[rt]
      case 0x0105: AESKEY[1] = GPR[rt]
   endif
else
   SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Result/Input into SMS4 Unit — CVM_MT_SMS4_RESINP



Note: These instructions are the same as Load Result/Input into AES or CAMELLIA Unit instructions (CVM_MT_AES_RESINP 0–1 and CVM_MT_CAMELLIA_RESINP 0–1, respectively).

Format: DMTC2 rt, 0x0100 **CVMv2**

DMTC2 rt, 0x0101

Purpose:

To load SMS4RESINP.

Description:
```
AESRESINP[0]  = rt // DMTC2 rt, 0x0100
AESRESINP[1]  = rt // DMTC2 rt, 0x0101
```

Restrictions:

Either CVM_MF_SMS4_RESINP0 or CVM_MF_SMS4_RESINP1 must execute between this CVM_MT_SMS4_RESINP* instruction and the last CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC1, CVM_MT_AES_DEC_CBC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC1, or CVM_MT_SMS4_DEC_CBC1 instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0102: AESRESINP[0] = GPR[rt]
    case 0x0103: AESRESINP[1] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

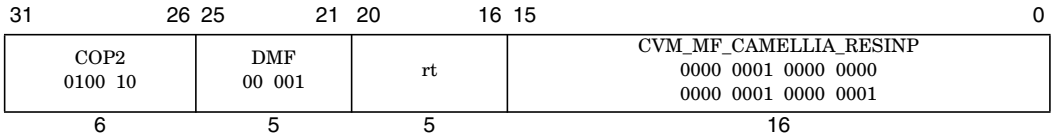
Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang 02/27/2014

CAMELLIA

Load Result/Input from CAMELLIA Unit CVM_MF_CAMELLIA_RESINP



Note: These instructions are the same as Load Result/Input from the AES or SMS4 Unit instructions (CVM_MF_AES_RESINP 0–1 and CVM_MF_SMS4_RESINP 0–1, respectively).

Format: DMFC2 rt, 0x0100 **CVMv3**

DMFC2 rt, 0x0101

Purpose:

To read AESRESINP (a.k.a. CAMELLIARESINP) from the CAMELLIA coprocessor.

Description:

```
rt = AESRESINP[0]  // DMFC2 rt, 0x0100
rt = AESRESINP[1]  // DMFC2 rt, 0x0101
```

Restrictions:

CVM_MF_CAMELLIA_RESINP1 (i.e. “DMFC2 rt, 0x0101”) must not be the first instruction that immediately follows a prior CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC1, CVM_MT_AES_DEC_CBC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC1, or CVM_MT_SMS4_DEC_CBC1 instruction. CVM_MF_CAMELLIA_RESINP1 should always follow CVM_MF_CAMELLIA_RESINP0.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Operation:

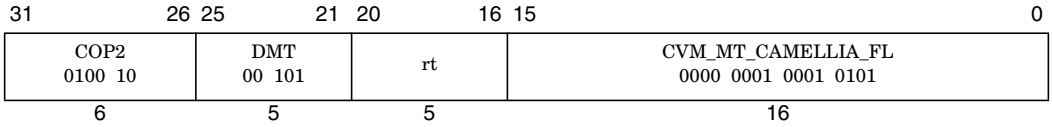
```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or !CvmCtl[CAMELLIA] then
    SignalException(ReservedInstruction)
  else
    case 0x0100: GPR[rt] = AESRESINP[0]
    case 0x0101: GPR[rt] = AESRESINP[1]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Fang A.U 02/27/2021

Perform CAMELLIA FL Function — CVM_MT_CAMELLIA_FL



Format: DMTC2 rt, 0x0115

Purpose:

To execute the Camellia FL function. The key is the contents of GPR rt. AESRESINP[0] is L.

Description: AESRESINP[0] = CAMELLIA_FL(AESRESINP[0], GPR[rt])

Restrictions:

Results are unpredictable if AESRESINP[0] isn't loaded (via CVM_MT_CAMELLIA_RESINP* instructions) or created from the CAMELLIA unit (via a CVM_MT_CAMELLIA_ROUND, CVM_MT_CAMELLIA_FL, or CVM_MT_CAMELLIA_FLINV instruction) prior to execution of this CVM_MT_CAMELLIA_FL instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
 if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
  SignalException(ReservedInstruction)
 else
  L<63:0> = AESRESINP[0]
  CAMELLIAROUNDKEY<63:0> = GPR[rt]<63:0>
  L<63:0> = CAMELLIA_FL(L, CAMELLIAROUNDKEY)
  AESRESINP[0] = L<63:0>
 endif
else
 SignalException(CoprocessorUnusable, 2) endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Programming Notes:

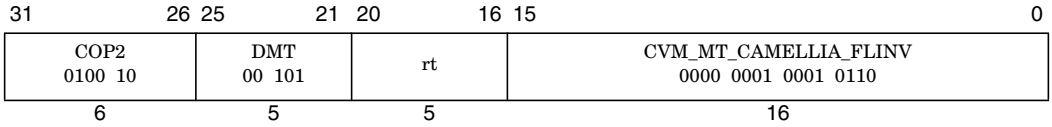
CVM_MT_CAMELLIA_FL can be used in CAMELLIA encryption/decryption. Refer to the Camellia_Encrypt() and Camellia_Decrypt() example C code in the Programming Notes section of the CVM_MT_CAMELLIA_ROUND instruction description.

The CVM_MT_CAMELLIA_FL instruction first appeared on the CN7XXX series parts. The behavior of the CVM_MT_CAMELLIA_FL instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX/CN6XXX series parts. CvmCtl[CAMELLIA] is always 0 on CN3XXX/CN5XXX/CN6XXX parts, so accurately portrays the lack of a CAMELLIA unit.

However, there is no guarantee of a Reserved Instruction exception when CVM_MT_CAMELLIA_FL is executed on these earlier parts. CAMELLIA code that is portable to all of CN7XXX (and successors) and CN3XXX/CN5XXX/CN6XXX must not execute CVM_MT_CAMELLIA_FL when CvmCtl[CAMELLIA] = 0.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Perform CAMELLIA FL^{-1} Function — CVM_MT_CAMELLIA_FLINV



Format: DMTC2 rt, 0x0116

Purpose:

To execute the Camellia FL^{-1} function. The key is the contents of GPR rt. AESRESINP[1] is R.

Description: AESRESINP[1] = CAMELLIA_FL^{-1}(AESRESINP[1], GPR[rt])

Restrictions:

Results are unpredictable if AESRESINP[1] isn't loaded (via CVM_MT_CAMELLIA_RESINP* instructions) or created from the CAMELLIA unit (via a CVM_MT_CAMELLIA_ROUND, CVM_MT_CAMELLIA_FL, or CVM_MT_CAMELLIA_FLINV instruction) prior to execution of this CVM_MT_CAMELLIA_FLINV instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
 if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
  SignalException(ReservedInstruction)
 else
  R<63:0> = AESRESINP[1]
  CAMELLIAROUNDKEY<63:0> = GPR[rt]<63:0>
  R<63:0> = CAMELLIA_FL^-1(R, CAMELLIAROUNDKEY)
  AESRESINP[1] = R<63:0>
 endif
else
 SignalException(CoprocessorUnusable, 2) endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Programming Notes:

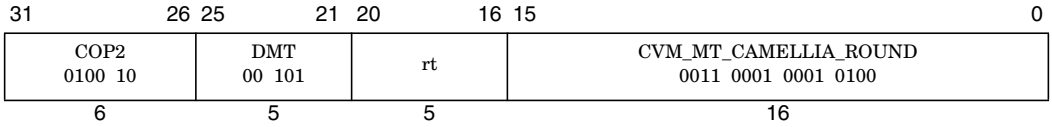
CVM_MT_CAMELLIA_FLINV can be used in CAMELLIA encryption/decryption. Refer to the Camellia_Encrypt() and Camellia_Decrypt() example C code in the Programming Notes section of the CVM_MT_CAMELLIA_ROUND instruction description.

The CVM_MT_CAMELLIA_FLINV instruction first appeared on the CN7XXX series parts. The behavior of the CVM_MT_CAMELLIA_FLINV instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX/CN6XXX series parts. CvmCtl[CAMELLIA] is always 0 on CN3XXX/CN5XXX/CN6XXX parts, so accurately portrays the lack of a CAMELLIA unit.

However, there is no guarantee of a Reserved Instruction exception when CVM_MT_CAMELLIA_FLINV is executed on these earlier parts. CAMELLIA code that is portable to all of CN7XXX (and successors) and CN3XXX/CN5XXX/CN6XXX must not execute CVM_MT_CAMELLIA_FLINV when CvmCtl[CAMELLIA] = 0.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Perform One CAMELLIA Round — CVM_MT_CAMELLIA_ROUND



Format: DMTC2 rt, 0x3114 **CVMv3**

Purpose:

To execute one Camellia round. The keys for the round are the contents of GPR rt. AESRESINP[0] is L and AESRESINP[1] is R.

Description:
```
L = AESRESINP[0]
AESRESINP[0] = AESRESINP[1] XOR CAMELLIA_FEISTEL(L, GPR[rt], AESRESINP[1])
AESRESINP[1] = L
```

Restrictions:

Results are unpredictable if AESRESINP[0] and AESRESINP[1] aren't loaded (via CVM_MT_CAMELLIA_RESINP* instructions) or created from the CAMELLIA unit (via a CVM_MT_CAMELLIA_ROUND, CVM_MT_CAMELLIA_FL, or CVM_MT_CAMELLIA_FLINV instruction) prior to execution of this CVM_MT_CAMELLIA_ROUND instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
 if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
  SignalException(ReservedInstruction)
 else
  L<63:0> = AESRESINP[0]
  R<63:0> = AESRESINP[1]
  CAMELLIAROUNDKEY<63:0> = GPR[rt]<63:0>
  R<63:0> = R<63:0> XOR CAMELLIA_FEISTEL(L, CAMELLIAROUNDKEY, R)
  AESRESINP[0] = R<63:0>
  AESRESINP[1] = L<63:0>
 endif
else
 SignalException(CoprocessorUnusable, 2) endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Power Yang AL 02/27/2014

Programming Notes:

CVM_MT_CAMELLIA_ROUND can be used to generate the CAMELLIA round keys. Example C code to generate the required CAMELLIA round keys:

```
const int KSFT1[26] = {
  0,64,0,64,15,79,15,79,30,94,45,109,45,124,60,124,77,13,
  94,30,94,30,111,47,111,47
};
const int KIDX1[26] = {
  0,0,4,4,0,0,4,4,4,4,0,0,4,0,4,4,0,0,0,0,4,4,0,0,4,4
};
const int KSFT2[34] = {
  0,64,0,64,15,79,15,79,30,94,30,94,45,109,45,109,60,124,
  60,124,60,124,77,13,77,13,94,30,94,30,111,47,111,47
};
const int KIDX2[34] = {
  0,0,12,12,8,8,4,4,8,8,12,12,0,0,4,4,0,0,8,8,12,12,
  0,0,4,4,8,8,4,4,0,0,12,12
};

static inline uint64_t RotBlock( const uint32_t *x, const int n )  {
  uint32_t res0, res1;
  int r;
  if( (r = (n & 31)) ){
     res0 = x[((n>>5)+0)&3]<<r^x[((n>>5)+1)&3]>>(32-r);
     res1 = x[((n>>5)+1)&3]<<r^x[((n>>5)+2)&3]>>(32-r);
  }
  else{
     res0 = x[((n>>5)+0)&3];
     res1 = x[((n>>5)+1)&3];
  }
  return((((uint64_t) res0) << 32) | res1);  }

const uint64_t SIGMA[6] = {
  0xa09e667f3bcc908b,
  0xb67ae8584caa73b2,
  0xc6ef372fe94f82be,
  0x54ff53a5f1d36f1c,
  0x10e527fade682d1d,
  0xb05688c2b3e6c1fd
};

void Camellia_Roundkeygen( const int keylen, const uint64_t key[], uint64_t round_keys[] )  {
  uint64_t t[8];

  if( keylen == 128 ) {
     t[0] = key[0];
     t[1] = key[1];
     t[2] = 0;
     t[3] = 0;
  }
  else if( keylen == 192 ) {
     t[0] = key[0];
     t[1] = key[1];
     t[2] = key[2];
     t[3] = t[2] ^ 0xFFFFFFFFFFFFFFFFull;
  }
  else {
     assert(keylen == 256);
     t[0] = key[0];
     t[1] = key[1];
     t[2] = key[2];
     t[3] = key[3];
  }

  CVM_MT_CAMELLIA_RESINP(t[0] ^ t[2], 0);
  CVM_MT_CAMELLIA_RESINP(t[1] ^ t[3], 1);

  CVM_MT_CAMELLIA_ROUND(SIGMA[0]);
  CVM_MT_CAMELLIA_ROUND(SIGMA[1]);
```

Cavium Confidential For Bowen Yang ALU 02/27/2014

```
CVM_MF_CAMELLIA_RESINP(t[4], 0);
CVM_MF_CAMELLIA_RESINP(t[5], 1);
t[4] ^= t[0];
t[5] ^= t[1];
CVM_MT_CAMELLIA_RESINP(t[4], 0);
CVM_MT_CAMELLIA_RESINP(t[5], 1);

CVM_MT_CAMELLIA_ROUND(SIGMA[2]);
CVM_MT_CAMELLIA_ROUND(SIGMA[3]);

CVM_MF_CAMELLIA_RESINP(t[4], 0);
CVM_MF_CAMELLIA_RESINP(t[5], 1);

uint32_t u[16];
u[0] = (uint32_t)(t[0] >> 32);
u[1] = (uint32_t)(t[0] >>  0);
u[2] = (uint32_t)(t[1] >> 32);
u[3] = (uint32_t)(t[1] >>  0);
u[4] = (uint32_t)(t[4] >> 32);
u[5] = (uint32_t)(t[4] >>  0);
u[6] = (uint32_t)(t[5] >> 32);
u[7] = (uint32_t)(t[5] >>  0);

if( keylen == 128 ) {
  int i;
  for( i=0; i<26; i+=2 ){
    round_keys[i]   = RotBlock( u+KIDX1[i+0], KSFT1[i+0] );
    round_keys[i+1] = RotBlock( u+KIDX1[i+1], KSFT1[i+1] );
  }
}
else {
  CVM_MT_CAMELLIA_RESINP(t[4] ^ t[2], 0);
  CVM_MT_CAMELLIA_RESINP(t[5] ^ t[3], 1);

  CVM_MT_CAMELLIA_ROUND(SIGMA[4]);
  CVM_MT_CAMELLIA_ROUND(SIGMA[5]);

  CVM_MF_CAMELLIA_RESINP(t[6], 0);
  CVM_MF_CAMELLIA_RESINP(t[7], 1);

  u[ 8] = (uint32_t)(t[2] >> 32);
  u[ 9] = (uint32_t)(t[2] >>  0);
  u[10] = (uint32_t)(t[3] >> 32);
  u[11] = (uint32_t)(t[3] >>  0);
  u[12] = (uint32_t)(t[6] >> 32);
  u[13] = (uint32_t)(t[6] >>  0);
  u[14] = (uint32_t)(t[7] >> 32);
  u[15] = (uint32_t)(t[7] >>  0);

  int i;
  for( i=0; i<34; i+=2 ){
    round_keys[i]   = RotBlock( u+KIDX2[i+0], KSFT2[i+0] );
    round_keys[i+1] = RotBlock( u+KIDX2[i+1], KSFT2[i+1] );
  }
}
```

CVM_MT_CAMELLIA_ROUND can be used in combination with CVM_MT_CAMELLIA_FL and CVM_MT_CAMELLIA_FLINV to perform a CAMELLIA encryption. Example C code:

```
void Camellia_Encrypt( const int len128bit, const uint64_t plaintext[],
                       const uint64_t round_keys[], uint64_t ciphertext[] )  {
  CVM_MT_CAMELLIA_RESINP(plaintext[0] ^ *round_keys++, 0); // 0
  CVM_MT_CAMELLIA_RESINP(plaintext[1] ^ *round_keys++, 1); // 1

  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 2
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 3
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 4
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 5
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 6
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 7

  CVM_MT_CAMELLIA_FL   (*round_keys++); // 8
```

Cavium Confidential For Bowen Yang ALU 11/27/2014

```
  CVM_MT_CAMELLIA_FLINV(*round_keys++); // 9

  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 10
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 11
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 12
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 13
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 14
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 15

  CVM_MT_CAMELLIA_FL   (*round_keys++); // 16
  CVM_MT_CAMELLIA_FLINV(*round_keys++); // 17

  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 18
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 19
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 20
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 21
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 22
  CVM_MT_CAMELLIA_ROUND(*round_keys++); // 23

  if( !len128bit ) {
    CVM_MT_CAMELLIA_FL   (*round_keys++); // 24
    CVM_MT_CAMELLIA_FLINV(*round_keys++); // 25

    CVM_MT_CAMELLIA_ROUND(*round_keys++); // 26
    CVM_MT_CAMELLIA_ROUND(*round_keys++); // 27
    CVM_MT_CAMELLIA_ROUND(*round_keys++); // 28
    CVM_MT_CAMELLIA_ROUND(*round_keys++); // 29
    CVM_MT_CAMELLIA_ROUND(*round_keys++); // 30
    CVM_MT_CAMELLIA_ROUND(*round_keys++); // 31
  }

  CVM_MF_CAMELLIA_RESINP(ciphertext[1], 0);
  CVM_MF_CAMELLIA_RESINP(ciphertext[0], 1);
  ciphertext[0] ^= *round_keys++;          // 24 or 32
  ciphertext[1] ^= *round_keys;            // 25 or 33
 }
```

CVM_MT_CAMELLIA_ROUND can also be used in combination with CVM_MT_CAMELLIA_FL and CVM_MT_CAMELLIA_FLINV to perform a CAMELLIA decryption. Example C code:

```
 void Camellia_Decrypt( const int len128bit, const uint64_t ciphertext[],
                        const uint64_t round_keys[], uint64_t plaintext[] )  {
  const uint64_t *roundkeyptr;
  if( len128bit ){
    roundkeyptr = round_keys + 23;
    CVM_MT_CAMELLIA_RESINP(ciphertext[0] ^ roundkeyptr[1], 0); // 24
    CVM_MT_CAMELLIA_RESINP(ciphertext[1] ^ roundkeyptr[2], 1); // 25
  }
  else {
    roundkeyptr = round_keys + 31;
    CVM_MT_CAMELLIA_RESINP(ciphertext[0] ^ roundkeyptr[1], 0); // 32
    CVM_MT_CAMELLIA_RESINP(ciphertext[1] ^ roundkeyptr[2], 1); // 33

    CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 31
    CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 30
    CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 29
    CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 28
    CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 27
    CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 26

    CVM_MT_CAMELLIA_FL   (*roundkeyptr--); // 25
    CVM_MT_CAMELLIA_FLINV(*roundkeyptr--); // 24
  }

  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 23
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 22
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 21
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 20
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 19
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 18

  CVM_MT_CAMELLIA_FL   (*roundkeyptr--); // 17
  CVM_MT_CAMELLIA_FLINV(*roundkeyptr--); // 16
```

Cavium Confidential For Bowen Yang A-U 01/07/2014

```
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 15
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 14
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 13
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 12
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 11
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 10

  CVM_MT_CAMELLIA_FL   (*roundkeyptr--); // 9
  CVM_MT_CAMELLIA_FLINV(*roundkeyptr--); // 8

  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 7
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 6
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 5
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 4
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 3
  CVM_MT_CAMELLIA_ROUND(*roundkeyptr--); // 2

  CVM_MF_CAMELLIA_RESINP(plaintext[1], 0);
  CVM_MF_CAMELLIA_RESINP(plaintext[0], 1);
  plaintext[1] ^= *roundkeyptr--;       // 1
  plaintext[0] ^= *roundkeyptr;         // 0
}
```

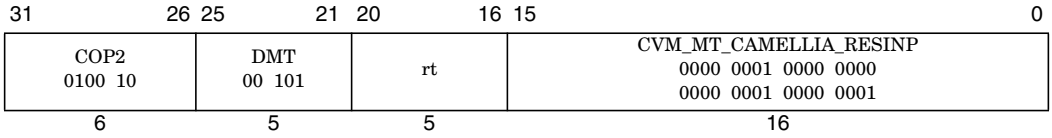
The CVM_MT_CAMELLIA_ROUND instruction first appeared on the CN7XXX series parts.

The behavior of the CVM_MT_CAMELLIA_ROUND instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX/CN6XXX series parts. CvmCtl[CAMELLIA] is always 0 on CN3XXX/CN5XXX/CN6XXX parts, so accurately portrays the lack of a CAMELLIA unit.

However, there is no guarantee of a Reserved Instruction exception when CVM_MT_CAMELLIA_ROUND is executed on these earlier parts. CAMELLIA code that is portable to all of CN7XXX (and successors) and CN3XXX/CN5XXX/CN6XXX must not execute CVM_MT_CAMELLIA_ROUND when CvmCtl[CAMELLIA] = 0.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Result/Input into CAMELLIA Unit CVM_MT_CAMELLIA_RESINP



Note: These instructions are the same as Load Result/Input into AES or SMS4 Unit instructions (CVM_MT_AES_RESINP 0–1 and CVM_MT_SMS4_RESINP 0–1, respectively).

Format: DMTC2 rt, 0x0100 **CVMv3**

DMTC2 rt, 0x0101

Purpose:

To load AESRESINP (a.k.a. CAMELLIARESINP).

Description: AESRESINP[0] = rt // DMTC2 rt, 0x0100
AESRESINP[1] = rt // DMTC2 rt, 0x0101

Restrictions:

Either CVM_MF_CAMELLIA_RESINP0 or CVM_MF_CAMELLIA_RESINP1 must execute between this CVM_MT_CAMELLIA_RESINP* instruction and the last CVM_MT_AES_ENC_CBC1, CVM_MT_AES_ENC1, CVM_MT_AES_DEC1, CVM_MT_AES_DEC_CBC1, CVM_MT_SMS4_ENC_CBC1, CVM_MT_SMS4_ENC1, CVM_MT_SMS4_DEC1, or CVM_MT_SMS4_DEC_CBC1 instruction.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or !CvmCtl[CAMELLIA] then
    SignalException(ReservedInstruction)
  else
    case 0x0102: AESRESINP[0] = GPR[rt]
    case 0x0103: AESRESINP[1] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

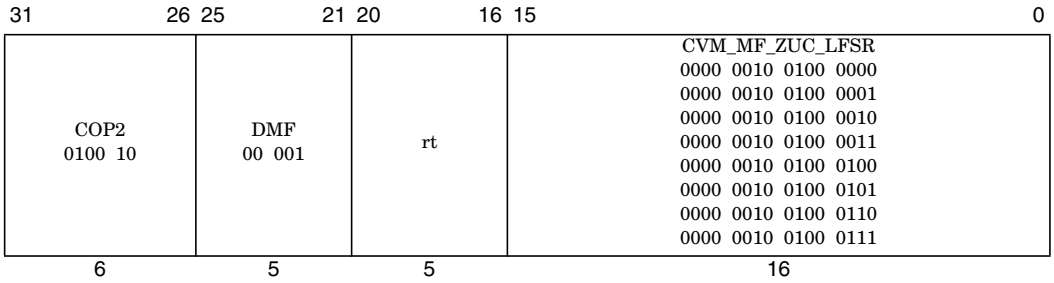
Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowei Yang ALL 02/27/2014

ZUC

Load LFSR from ZUC Unit — CVM_MF_ZUC_LFSR



Note: These instructions are the same as Load Data from HSH Unit (wide mode) and SNOW3G Unit instructions (CVM_MF_HSH_DATW 0–7 and CVM_MF_SNOW3G_LFSR 0–7, respectively).

Format: DMFC2 rt, 0x0240 **CVMv3**
DMFC2 rt, 0x0241
DMFC2 rt, 0x0242
DMFC2 rt, 0x0243
DMFC2 rt, 0x0244
DMFC2 rt, 0x0245
DMFC2 rt, 0x0246
DMFC2 rt, 0x0247

Purpose:

To read the LFSR from the ZUC coprocessor.

Description:

```
rt = HASHDAT[0]   // DMTC2 rt,0x0240
rt = HASHDAT[1]   // DMTC2 rt,0x0241
rt = HASHDAT[2]   // DMTC2 rt,0x0242
rt = HASHDAT[3]   // DMTC2 rt,0x0243
rt = HASHDAT[4]   // DMTC2 rt,0x0244
rt = HASHDAT[5]   // DMTC2 rt,0x0245
rt = HASHDAT[6]   // DMTC2 rt,0x0246
rt = HASHDAT[7]   // DMTC2 rt,0x0247
```

When executing ZUC, HASHDAT[i] has ZUC LFSR S(2×i) in <62:32>, LFSR S(2×i+1) in <30:0>. For example, HASHDAT[3] has ZUC LFSR S6 in <62:32>, and LFSR S7 in <30:0>.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Cavium Confidential For Bowen Yang BLU 07/7/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0040: GPR[rt] = ZUCLFSR[0] = HASHDAT[0]
    case 0x0041: GPR[rt] = ZUCLFSR[1] = HASHDAT[1]
    case 0x0042: GPR[rt] = ZUCLFSR[2] = HASHDAT[2]
    case 0x0043: GPR[rt] = ZUCLFSR[3] = HASHDAT[3]
    case 0x0044: GPR[rt] = ZUCLFSR[4] = HASHDAT[4]
    case 0x0045: GPR[rt] = ZUCLFSR[5] = HASHDAT[5]
    case 0x0046: GPR[rt] = ZUCLFSR[6] = HASHDAT[6]
    case 0x0047: GPR[rt] = ZUCLFSR[7] = HASHDAT[7]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

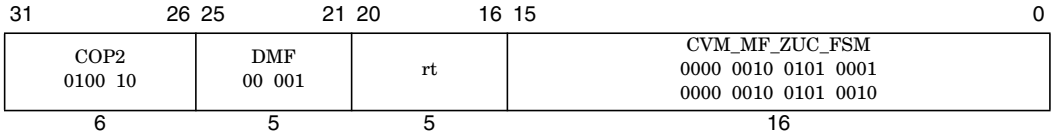
Reserved Instruction

Programming Notes:

CVM_MF_ZUC_LFSR may be rarely used when executing ZUC - typically only to save ZUC state.

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Load FSM from ZUC Unit — CVM_MF_ZUC_FSM



Note: These instructions are the same as Load IV from HSH Unit (wide mode) and SNOW3G Unit instructions (CVM_MF_HSH_IVW 1–2 and CVM_MF_SNOW3G_ZUC_FSM 0–1, respectively).

Format: DMFC2 rt, 0x0251 **CVMv3**

DMFC2 rt, 0x0252

Purpose:

To read the FSM from the ZUC coprocessor.

Description:
```
rt = HASHIV[1] // DMFC2 rt, 0x0251
rt = HASHIV[3] // DMFC2 rt, 0x0252
```

When executing ZUC, HASHIV[i] has ZUC FSM_Ri in <31:0> and bits <63:32> are unpredictable. For example, HASHIV[2] has ZUC FSM_R2 in <31:0>.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[Nocrypto] then
    SignalException(ReservedInstruction)
  else
    case 0x0251: GPR[rt] = HASHIV[1]
    case 0x0252: GPR[rt] = HASHIV[2]

  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

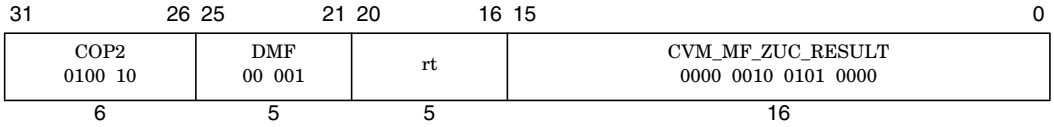
Coprocessor Unusable, Reserved Instruction

Programming Notes:

CVM_MF_ZUC_FSM may be rarely used when executing ZUC - typically only to save ZUC state.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Result from ZUC Unit — CVM_MF_ZUC_RESULT



Note: This instruction is the same as Load IV from HSH Unit (wide mode) and SNOW3G Unit instructions (CVM_MF_HSH_IVW 0 and CVM_MF_SNOW3G_RESULT, respectively).

Format: DMFC2 rt, 0x0250 **CVMv3**

Purpose:

To read the 64 bits of ZUC keystream from the ZUC coprocessor.

Description: rt = HASHIV[0] // DMFC2 rt, 0x0250

When executing ZUC, HASHIV[0] has the last two 32-bit keystream words produced by the hardware. HASHIV[0]<63:32> contains the earlier 32-bit keystream word, and HASHIV[0]<31:0> contains the later 32-bit keystream word.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0250: GPR[rt] = HASHIV[0]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

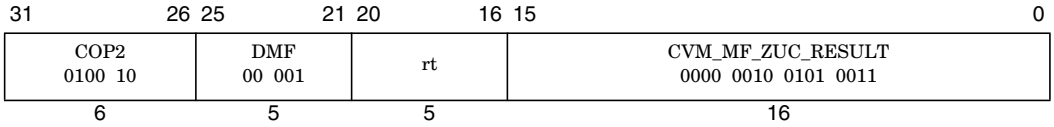
Coprocessor Unusable, Reserved Instruction

Programming Notes:

CVM_MF_ZUC_RESULT is commonly used when executing ZUC to extract the ZUC keystream words.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load 128-EIA3 MAC from ZUC Unit — CVM_MF_ZUC_TRESULT



Note: This instruction is the same as CVM_MF_HSH_IVW3 and CVM_MF_SNOW3G_FSM2 instructions.

Format: DMFC2 rt, 0x0253 **CVMv3**

Purpose:

To read the integrity algorithm 128-EIA3 message authentication code (MAC) from the ZUC unit.

Description: rt = HASHIV[3] // DMFC2 rt, 0x0253

When executing ZUC, HASHIV[3] has the 32-bit MAC T in <31:0> and <63:32> are zeroed.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[Nocrypto] then
    SignalException(ReservedInstruction)
  else
    case 0x0253: GPR[rt] = HASHIV[3]

  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

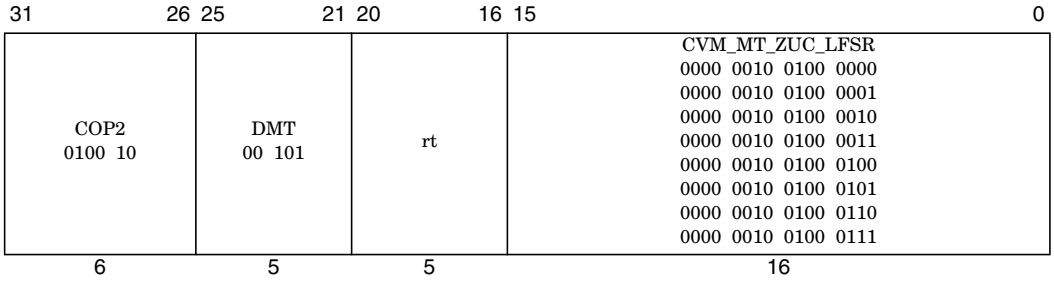
Coprocessor Unusable, Reserved Instruction

Programming Notes:

CVM_MF_ZUC_TRESULT can commonly be used once per 128-EIA3 MAC calculation to extract the MAC.

Cavium Confidential For BoWen Yang ALU 02/27/2014

Load LFSR to ZUC Unit — CVM_MT_ZUC_LFSR



Note: These instructions are the same as Load Data to HSH Unit (wide mode) and SNOW3G Unit instructions (CVM_MT_HSH_DATW 0–7 and CVM_MT_SNOW3G_LFSR 0–7, respectively).

Format: DMTC2 rt, 0x0240 **CVMv3**

DMTC2 rt, 0x0241

DMTC2 rt, 0x0242

DMTC2 rt, 0x0243

DMTC2 rt, 0x0244

DMTC2 rt, 0x0245

DMTC2 rt, 0x0246

DMTC2 rt, 0x0247

Purpose:

To load values into the ZUC LFSR.

Description:

```
HASHDAT[0] = rt // DMTC2 rt,0x0240
HASHDAT[1] = rt // DMTC2 rt,0x0241
HASHDAT[2] = rt // DMTC2 rt,0x0242
HASHDAT[3] = rt // DMTC2 rt,0x0243
HASHDAT[4] = rt // DMTC2 rt,0x0244
HASHDAT[5] = rt // DMTC2 rt,0x0245
HASHDAT[6] = rt // DMTC2 rt,0x0246
HASHDAT[7] = rt // DMTC2 rt,0x0247
```

When executing ZUC, HASHDAT[i] has ZUC LFSR_S(2×i) in <63:32>, LFSR_S(2×i+1) in <31:0>. For example, HASHDAT[3] has ZUC LFSR_S6 in <63:32>, and LFSR_S7 in <31:0>.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Cavium Confidential For Bowen Yang ALU 02/7/2014

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0240: HASHDAT[0] = GPR[rt]
    case 0x0241: HASHDAT[1] = GPR[rt]
    case 0x0242: HASHDAT[2] = GPR[rt]
    case 0x0243: HASHDAT[3] = GPR[rt]
    case 0x0244: HASHDAT[4] = GPR[rt]
    case 0x0245: HASHDAT[5] = GPR[rt]
    case 0x0246: HASHDAT[6] = GPR[rt]
    case 0x0247: HASHDAT[7] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Programming Notes:

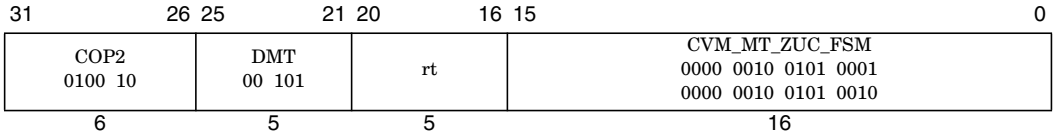
Applications commonly use CVM_MT_ZUC_LFSR to initialize the ZUC LFSR before initiating ZUC keystream generation. This initialization must follow the convention established in the ZUC specifications. For example, the following Initialize_CVM_ZUC() can initialize and instruct the hardware to generate the first two 32-bit ZUC keywords:

```
uint64_t lfsr_s_zeroes[8] = {0x0044d7000026bc00ull, 0x00626b0000135e00ull,
                             0x005789000035e200ull, 0x007135000009af00ull,
                             0x004d7800002f1300ull, 0x006bc400001af100ull,
                             0x005e2600003c4d00ull, 0x00789a000047ac00ull};

#define SLL64(x,n) (((uint64_t) (x)) << (n))
#define LFSR_S64(n,keyb1,keyb2,ivb1,ivb2) \
  (lfsr_s_zeroes[n] | SLL64(keyb1,55) | SLL64(keyb2,23) | SLL64(ivb1,32) | (ivb2))
#define CVM_ZUC_INIT_S64(n) \
  CVM_MT_ZUC_LFSR(LFSR_S64(n,key[(n)*2],key[((n)*2)+1],iv[(n)*2],iv[((n)*2)+1]), n)

void Initialize_CVM_ZUC(uint8_t key[], uint8_t iv[]) {
   CVM_ZUC_INIT_S64(0);
   CVM_ZUC_INIT_S64(1);
   CVM_ZUC_INIT_S64(2);
   CVM_ZUC_INIT_S64(3);
   CVM_ZUC_INIT_S64(4);
   CVM_ZUC_INIT_S64(5);
   CVM_ZUC_INIT_S64(6);
   CVM_MT_ZUC_START(LFSR_S64(7,key[14],key[15],iv[14],iv[15]));
}
```

After this sequence, the first two 32-bit ZUC keystream words are present in HASHIV[0] and can be extracted via a CVM_MF_ZUC_RESULT.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load FSM to ZUC Unit — CVM_MT_ZUC_FSM



Note: These instructions are the same as Load IV to HSH Unit (wide mode) and SNOW3G Unit instructions (CVM_MT_HSH_IVW 1–2 and CVM_MT_SNOW3G_FSM 0–1, respectively).

Format: DMTC2 rt, 0x0251 **CVMv3**

DMTC2 rt, 0x0252

Purpose:

To load values into ZUC FSM.

Description:
```
HASHIV[1] = rt  // DMTC2 rt, 0x0251
HASHIV[2] = rt  // DMTC2 rt, 0x0252
```

When executing ZUC, HASHIV[i] has ZUC FSM_Ri in <31:0> and bits <63:32> are unpredictable. For example, HASHIV[2] has ZUC FSM_R2 in <31:0>.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0251: HASHIV[1] = GPR[rt]
    case 0x0252: HASHIV[2] = GPR[rt]
    endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

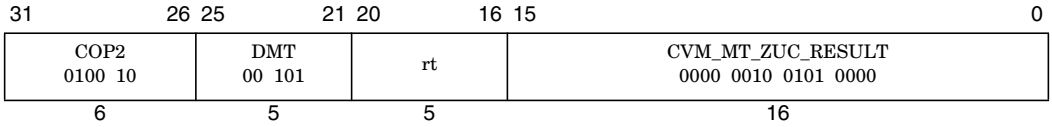
Coprocessor Unusable, Reserved Instruction

Programming Notes:

CVM_MT_ZUC_FSM may be rarely used when executing ZUC - typically only to restore ZUC state.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Load Data to ZUC Unit **CVM_MT_ZUC_RESULT**



Note: This instruction is the same as Load Data to HSH Unit (wide mode) and SNOW3G Unit instructions (CVM_MT_HSH_IVW 0 and CVM_MT_SNOW3G_RESULT, respectively).

Format: DMTC2 rt, 0x0250 **CVMv3**

Purpose:

To load a value into ZUCRESULT.

Description: HASHIV[0] = rt // DMTC2 rt, 0x0250

When executing ZUC, HASHIV[0] has the last two 32-bit keystream words produced by the hardware. HASHIV[0]<63:32> contains the earlier 32-bit keystream word, and HASHIV[0]<31:0> contains the later 32-bit keystream word.

Restrictions:

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

This instruction is not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued this instruction.

Operation:

```
if IsCoprocessorEnabled(2) then
  if (not Are64bitOperationsEnabled()) or CvmCtl[NOCRYPTO] then
    SignalException(ReservedInstruction)
  else
    case 0x0250: HASHIV[0] = GPR[rt]
  endif
else
  SignalException(CoprocessorUnusable, 2)
endif
```

Exceptions:

Coprocessor Unusable

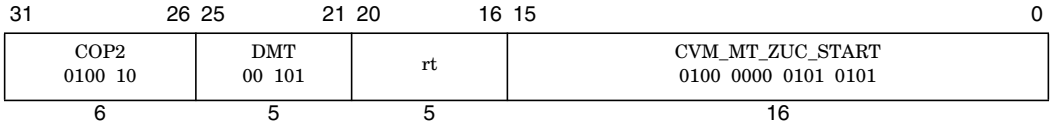
Reserved Instruction

Programming Notes:

CVM_MT_ZUC_RESULT may be rarely used when executing ZUC - typically only to restore ZUC state.

Cavium Confidential For Bowen Yang ALU 02/27/2014

Start ZUC Keystream **CVM_MT_ZUC_START**



Format: DMFC2 rt, 0x4055 **CVMv3**

Purpose:

To complete ZUC initialization and compute the first two 32-bit words of ZUC keystream. The contents of GPR rt are put into HASHDAT[7]. (HASHDAT[7]<63:32> has the ZUC LFSR_S14 state, and HASHDAT[7]<31:0> has the ZUC LFSR_S15 state.) HASHIV[1:3]<31:0> are zeroed. (HASHIV[1:3]<31:0> has the ZUC FSM_R1, FSM_R2, and FSM_R3, respectively.) The remaining ZUC initialization sequence is executed, and then the keystream result is generated. (HASHIV[0]<63:32> has the first 32-bit keystream word and HASHIV[0]<31:0> has the second 32-bit keystream word.

Description:

```
HASHDAT[7] = rt
ZUC_Initialize()
HASHIV[0]<63:32> = ZUC_Loop_Iteration()
HASHIV[0]<31:0> = ZUC_Loop_Iteration()
HASHIV[3]<63:0> = 0
```

Restrictions:

HASHDAT[0:6]<62:32,30:0> must have been initialized before the CVM_MT_ZUC_START, or else the keystream result, LFSR, and FSM become unpredictable. HASHDAT[0:7]<63> and HASHDAT[0:7]<31> are unused and are unpredictable after the CVM_MT_ZUC_START. The keystream result, LFSR, and FSM are unpredictable following a CVM_MT_ZUC_START if any one of the 16 individual 31-bit LFSR S values are zero.

HASHDAT[8:14]<63:0>, HASHIV[4:7]<63:0>, and HASHIV[1:2]<63:32> are unpredictable after the instruction.

This instruction renders the value of any results of prior hash instructions (CVM_MT_HSH_STARTMD5, CVM_MT_HSH_STARTSHA, CVM_MT_HSH_STARTSHA256, CVM_MT_HSH_STARTSHA512) and prior SNOW3G instructions (CVM_MT_SNOW3G_START and

CVM_MT_SNOW3G_MORE) unpredictable.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Cavium Confidential For Bowen Yang ALU 5/27/2014

Operation:

```
if IsCoprocessorEnabled(2) then
 if (not Are64bitOperationsEnabled()) or !CvmCtl[ZUC] then
  SignalException(ReservedInstruction)
 else
  HASHDAT[7]<63:0> = GPR[rt]<63:0>
  for (i=0; i<8; i++) {
   LFSR_S(2*i) = HASHDAT[i]<62:32>
   LFSR_S(2*i+1) = HASHDAT[i]<30:0>
  }
  HASHDAT[14:0]<63:0> = unpredictable
  HASHIV[7:0]<63:0> = unpredictable
  FSM_R1 = FSM_R2 = 0
  for (i=0; i<32; i++) {
   ZUC_BitReorganization()
   w = ZUC_F()
   ZUC_LFSRWithInitialisationMode(w >> 1)
  }
  ZUC_BitReorganization()
  ZUC_F() // discard output
  ZUC_LFSRWithWorkMode()
  HASHIV[0]<63:32> = ZUC_GenerateKeyStream()
  HASHIV[0]<31:0> = ZUC_GenerateKeyStream()
  for (i=0; i<8; i++) {
   HASHDAT[i]<62:32> = LFSR_S(2*i)
   HASHDAT[i]<30:0> = LFSR_S(2*i+1)
  }
  HASHIV[1]<31:0> = FSM_R1
  HASHIV[2]<31:0> = FSM_R2
  HASHIV[3]<63:0> = 0
 endif
else
 SignalException(CoprocessorUnusable, 2) endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Programming Notes:

Applications commonly use CVM_MT_ZUC_START to complete the ZUC LFSR initialization and initiate ZUC keystream generation. This initialization must follow the convention established in the ZUC specifications. For example, the following Initialize_CVM_ZUC() can initialize and instruct the hardware to generate the first two 32-bit ZUC keywords:

```
uint64_t lfsr_s_zeroes[8] = {0x0044d7000026bc00ull, 0x00626b0000135e00ull,
                             0x005789000035e200ull, 0x007135000009af00ull,
                             0x004d7800002f1300ull, 0x006bc400001af100ull,
                             0x005e2600003c4d00ull, 0x00789a000047ac00ull};

#define SLL64(x,n) (((uint64_t) (x)) << (n))
#define LFSR_S64(n,keyb1,keyb2,ivb1,ivb2) \
  (lfsr_s_zeroes[n] | SLL64(keyb1,55) | SLL64(keyb2,23) | SLL64(ivb1,32) | (ivb2))
#define CVM_ZUC_INIT_S64(n) \
  CVM_MT_ZUC_LFSR(LFSR_S64(n,key[(n)*2],key[((n)*2)+1],iv[(n)*2],iv[((n)*2)+1]), n)

void Initialize_CVM_ZUC(uint8_t key[], uint8_t iv[]) {
   CVM_ZUC_INIT_S64(0);
   CVM_ZUC_INIT_S64(1);
   CVM_ZUC_INIT_S64(2);
   CVM_ZUC_INIT_S64(3);
   CVM_ZUC_INIT_S64(4);
   CVM_ZUC_INIT_S64(5);
   CVM_ZUC_INIT_S64(6);
   CVM_MT_ZUC_START(LFSR_S64(7,key[14],key[15],iv[14],iv[15]));
}
```

The CVM_MT_ZUC_START instruction first appeared on the CN7XXX series parts.

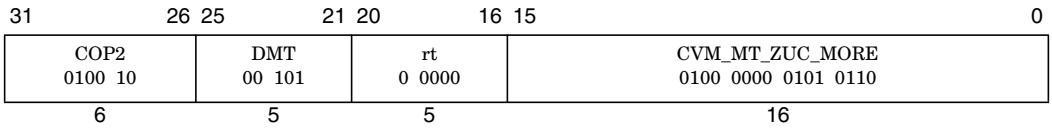
The behavior of the CVM_MT_ZUC_START instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX/CN6XXX series parts. CvmCtl[ZUC] is always 0 on CN3XXX/CN5XXX/CN6XXX parts, so accurately portrays the lack of a ZUC unit.

However, there is no guarantee of a Reserved Instruction exception when CVM_MT_ZUC_START is executed on these earlier parts. ZUC code that is portable to all of CN7XXX (and successors) and CN3XXX/CN5XXX/CN6XXX must not execute CVM_MT_ZUC_START when CvmCtl[ZUC] is clear

.

Cavium Confidential For Bowen Wang ALU 02/27/2014

Continue ZUC Keystream and 128-EIA3 MAC — CVM_MT_ZUC_MORE



Format: DMTC2 rt, 0x4056 **CVMv3**

Purpose:

To compute two more 32 bit words of ZUC keystream and/or calculate the message authentication code (MAC) for the integrity algorithm 128?EIA3 for 64 more bits. (HASHIV[0]<63:32> has the first 32-bit keystream word and HASHIV[0]<31:0> has the second 32-bit keystream word.) The hardware accumulates T, the MAC, from the keystream and the GPR rt contents, which supplies the message data. (HASHIV[3]<63:0> has T in the least-significant 32 bits, 0s in all other upper bits.)

Description:
ZDAT<95:32> = HASHIV[0]<63:0>
ZDAT<31:0> = HASHIV[0]<63:32> = ZUC_Loop_Iteration()
HASHIV[0]<31:0> = ZUC_Loop_Iteration()
HASHIV[3]<63:0> = T = Accumulate_MAC(GPR[rt]<63:0>, T = HASHIV[3]<63:0>, ZDAT<95:0>)

Restrictions:

The result of CVM_MT_ZUC_MORE is unpredictable unless it is preceded by a CVM_MT_ZUC_START operation or all of the following operations: CVM_MT_HSH_DATW 0-7 and CVM_MT_HSH_IVW 1-2 (without any intervening SNOW3G START or MORE).

HASHDAT[0:7]<62:23,30:0> and HASHIV[1:2]<31:0> must have been previously initialized with ZUC LFSR and FSM information, respectively, typically via a prior CVM_MT_ZUC_START or CVM_MT_ZUC_MORE, or else the keystream result, LFSR, and FSM become unpredictable. HASHDAT[0:7]<63> and HASHDAT[0:7]<31> are unused and are unpredictable after the CVM_MT_ZUC_MORE. The keystream result, LFSR, and FSM are unpredictable following a CVM_MT_ZUC_START if any one of the 16 individual 31-bit LFSR S values are zero.

HASHIV[3] becomes unpredictable if HASHIV[3]<63:32> ≠ 0 or if the keystream result, LFSR, or FSM are unpredictable when the message data is not zero. During normal operation, HASHIV[3]<63:32> is 0 and HASHIV[3]<31:0> contains the accumulated MAC.

HASHDAT[8:14]<63:0>, HASHIV[4:7]<63:0>, and HASHIV[1:2]<63:32> are unpredictable after the instruction.

This instruction renders the value of any results of prior hash instructions (CVM_MT_HSH_STARTMD5, CVM_MT_HSH_STARTSHA, CVM_MT_HSH_STARTSHA256, CVM_MT_HSH_STARTSHA512) and prior SNOW3G instructions (CVM_MT_SNOW3G_START and CVM_MT_SNOW3G_MORE) unpredictable.

If access to Coprocessor 2 is not enabled, a Coprocessor Unusable Exception is signaled. If access to Coprocessor 2 is enabled but access to 64-bit operations is not enabled, a Reserved Instruction Exception is signaled.

Cavium Confidential File
Author: Bowen Yang
Date: 2/7/2014

These instructions are not available for the OCTEON III CP family parts, which do not provide encryption functionality. Such parts signal a Reserved Instruction exception when issued these instructions.

Operation:

```
if IsCoprocessorEnabled(2) then
 if (not Are64bitOperationsEnabled()) or !CvmCtl[ZUC] then
  SignalException(ReservedInstruction)
 else
  for (i=0; i<8; i++) {
   LFSR_S(2*i)   = HASHDAT[i]<62:32>
   LFSR_S(2*i+1) = HASHDAT[i]<30:0>
  }
  ZDAT<95:32>    = HASHIV[0]<63:0>
  FSM_R1         = HASHIV[1]<31:0>
  FSM_R2         = HASHIV[2]<31:0>
  T64            = HASHIV[3]<63:0>
  HASHDAT[14:0]<63:0> = unpredictable
  HASHIV[7:0]<63:0> = unpredictable
  ZDAT<31:0> = HASHIV[0]<63:32> = ZUC_GenerateKeyStream()
  HASHIV[0]<31:0>  = ZUC_GenerateKeyStream()
  for (i=0; i<8; i++) {
   HASHDAT[i]<62:32> = LFSR_S(2*i)
   HASHDAT[i]<30:0> = LFSR_S(2*i+1)
  }
  HASHIV[1]<31:0> = FSM_R1
  HASHIV[2]<31:0> = FSM_R2
  message_bits<63:0> = GPR[rt]<63:0>
  for (i=0; i<64; i++) {
   if(message_bits<63>) T64<31:0> = T64<31:0> XOR ZDAT<95:64>
   message_bits<63:1> = message_bits<62:0>
   ZDAT<95:1> = ZDAT<94:0>
  }
  if(T64<63:32> != 0) T64<63:0> = unpredictable
  HASHIV[3]<63:0> = T64
 endif
else
 SignalException(CoprocessorUnusable, 2) endif
```

Exceptions:

Coprocessor Unusable, Reserved Instruction

Cavium Confidential For Bowen Yang ALU 02/27/2014

Programming Notes:

High-performance usage of CVM_MT_ZUC_MORE for the confidentiality algorithm 128-EEA3 may commonly obtain 64 more bits of keystream data by executing DMTC2 $0, 0x4056 the next cycle after a CVM_MF_ZUC_RESULT that extracts the last 64 bits of ZUC keystream data. Software would then exclusive-or the extracted keystream data and the message data.

A series of CVM_MT_ZUC_MORE instructions can implement the per-message-bit (Zsubscript(i)) exclusive-or's required for the integrity algorithm 128-EIA3, with 64 more bits of message data supplied with each CVM_MT_ZUC_MORE. The final two required 32-bit exclusive-or's (Zsubscript(LENGTH) and Zsubscript(32x(L-1))) can either be implemented by introducing more pseudo-message-bits into the CVM_MT_ZUC_MORE stream, or by reading the keystream data and performing the exclusive-or's in software. The message data can be zero-extended to be a multiple of 64-bits - message zero-extension does not modify T.

The CVM_MT_ZUC_MORE instruction first appeared on the CN7XXX series parts.

The behavior of the CVM_MT_ZUC_MORE instruction is not predictable on the earlier OCTEON CN3XXX/CN5XXX/CN6XXX series parts. CvmCtl[ZUC] is always 0 on CN3XXX/CN5XXX/CN6XXX parts, so accurately portrays the lack of a ZUC unit.

However, there is no guarantee of a Reserved Instruction exception when CVM_MT_ZUC_MORE is executed on these earlier parts. ZUC code that is portable to all of CN7XXX (and successors) and CN3XXX/CN5XXX/CN6XXX must not execute CVM_MT_ZUC_MORE when CvmCtl[ZUC] is clear.

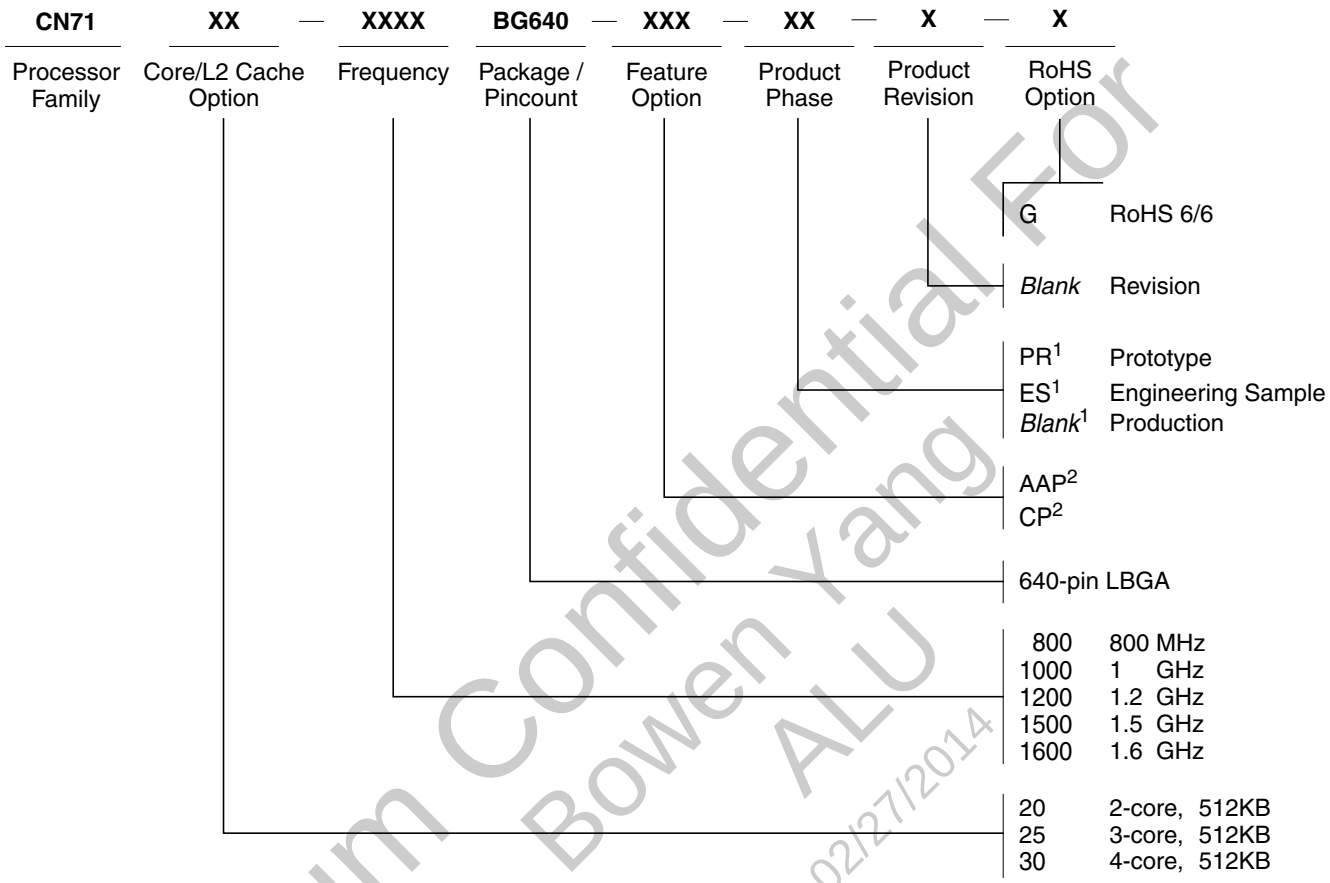
Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Appendix B

Ordering Information

The following shows the breakdown of the CN71XX part numbers.



[1] PR Prototype phase, preliminary testing in place (unless otherwise specified).
ES Engineering samples phase, full production testing in place, official production-release pending.
Blank Production phase, full production testing in place, official production release complete.

[2] AAP Application Acceleration Processor includes RAID, encryption, RegEx acceleration, networking, TCP acceleration, and QoS.
CP Communication Processor includes networking, TCP acceleration, and QoS.

Example Part Numbers:

CN7125-1000BG640-CP-G	3-core, 512KB L2C, 1.0 GHz, CP Option, Production, RoHS6/6
CN7130-1200BG640-AAP-G	4-core, 512KB L2C, 1.2 GHz, AAP Option, Production, RoHS6/6

Cavium Confidential For Bowen Yang ALU 02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Inside Back Cover
Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Cavium Confidential For
Bowen Yang
ALU
02/27/2014

Cavium, Inc.
2315 North 1st St,
San Jose, CA 95131
Telephone: +1-650-623-7000
Fax: +1-650-625-9751
Email: info@cavium.com

CN71XX-HM-0.991E© 2014 Cavium, Inc. All rights reserved