Microprocessor Initialization and Configuration 17

This chapter covers microprocessor initialization and configuration information for both uni-processor and dual-processor implementations of the embedded Pentium® processor family. For configuration information on symmetric dual-processing mode, refer to "Managing and Designing with the Symmetrical Dual Processing Configuration" on page 17-231.

Before normal operation of the processor can begin, the processor must be initialized by driving the RESET pin active. The RESET pin forces the processor to begin execution in a known state. Several features are optionally invoked at the falling edge of RESET: Built-in-Self-Test (BIST), Functional Redundancy Checking and Three-state Test Mode.

In addition to the standard RESET pin, the processor has implemented an initialization pin (INIT) that allows the processor to begin execution in a known state without disrupting the contents of the internal caches or the floating-point state.

This chapter describes the embedded Pentium processor power up and initialization procedures, and the test and configuration features enabled at the falling edge of RESET.

17.1 Power Up Specifications

During power up, RESET must be asserted while V_{CC} is approaching nominal operating voltage to prevent internal bus contention, which could negatively affect the reliability of the processor.

It is recommended that CLK begin toggling within 150 ms after V_{CC} reaches its proper operating level. For the embedded Pentium® processor with MMX™ technology, it is recommended that the CLK signal begin toggling within 150 ms after the last V_{CC} plane stabilizes. This recommendation is only to ensure long term reliability of the device.

In order for RESET to be recognized, the CLK input needs to be toggling. RESET must remain asserted for 1 millisecond after V_{CC} and CLK have reached their AC/DC specifications.

17.2 Test and Configuration Features

The INIT, FLUSH#, and FRCMC# inputs are sampled when RESET transitions from high to low to determine if BIST will be run, or if three-state test mode, or checker mode will be entered (respectively). If RESET is driven synchronously, these signals must be at their valid level and meet setup and hold times on the clock before the falling edge of RESET. If RESET is asserted asynchronously, these signals must be at their valid level two clocks before and after RESET transitions from high to low.

17.2.1 Built-in Self-Test

Self-test is initiated by driving the INIT pin high when RESET transitions from high to low. No bus cycles are run by the processor during self test. The duration of self test is approximately 2^{19} core clocks. Approximately 70% of the devices in the processor are tested by BIST. The embedded Pentium processor BIST consists of two parts: hardware self-test and microcode self-test. During the hardware portion of BIST, the microcode ROM and all large PLAs are tested. All possible input combinations of the microcode ROM and PLAs are tested.

The constant ROMs, BTB, TLBs, and all caches are tested by the microcode portion of BIST. The array tests (caches, TLBs and BTB) have two passes. On the first pass, data patterns are written to arrays, read back, and checked for mismatches. The second pass writes the complement of the initial data pattern, reads it back, and checks for mismatches. The constant ROMs are tested by using the microcode to add various constants and check the result against a stored value.

Upon successful completion of BIST, the cumulative result of all tests are stored in the EAX register. If EAX contains 0H, then all checks passed; any non-zero result indicates a faulty unit. Note that when an internal parity error is detected during BIST, the processor asserts the IERR# pin and attempts to shutdown.

17.2.2 Three-state Test Mode

When the FLUSH# pin is sampled low when RESET transitions from high to low, the processor enters three-state test mode. The processor floats all of its output pins and bidirectional pins, including pins that are never floated during normal operation (except TDO). Three-state test mode can be initiated to facilitate testing board interconnects. The processor remains in three-state test mode until the RESET pin is asserted again.

17.2.3 Functional Redundancy Checking

The functional redundancy checking (FRC) master/checker configuration input is sampled when RESET is high to determine whether the processor is configured in master mode (FRCMC# high) or checker mode (FRCMC# low). Note, the embedded Pentium processor with MMX technology does not support FRC mode.

The final master/checker configuration of the processor is determined the clock before the falling edge of RESET. When configured as a master, the processor drives its output pins as required by the bus protocol. When configured as a checker, the processor three-states all outputs (except IERR#, PICD0, PICD1 and TDO) and samples the output pins (that would normally be driven in master mode). If the sampled value differs from the value computed internally, the processor asserts IERR# to indicate an error. Note that IERR# is not asserted due to an FRC mismatch until two clocks after the ADS# of the first bus cycle (or in the third clock of the bus cycle).

To avoid an FRC error caused by differences in the unitialized FPU state, FINIT/FNINIT must be used to initialize the FPU state prior to using FSAVE/FNSAVE in FRC mode. The initialization should be done before other FPU activity so that it does not corrupt the previous state.

17.2.4 Lock Step APIC Operation

Lock Step operation is entered by holding BE4# high during the falling edge of RESET. Lock Step operation is not supported by the embedded Pentium processor with MMX technology.

Lock Step operation guarantees recognition of an interrupt on a specific clock by two processors operating together that are using the APIC as the interrupt controller. This functionality is related to FRC operation, but FRC on the APIC pins is not fully supported in this way. There is no FRC comparator on the APIC pins, but mismatches on these pins result in a mismatch on other pins of the processor.

Fault tolerant systems implemented with multiple processors that run identical code sequences and generate identical bus cycles on all clocks may utilize Lock Step operation.

Setup and Hold time specifications PICCLK (in relation to CLK) are added for this functionality. Additionally, there is a requirement to sustain specific integer ratios between the frequencies of PICCLK and CLK. This ratio should support both the maximum bus frequency of the device and the maximum frequency of PICCLK. Details of these specifications can be found in Chapter 7, "Electrical Differences Between Family Members."

17.3 Initialization with RESET, INIT and BIST

Two pins, RESET and INIT, are used to reset the processor in different manners. A "cold" or "power on" RESET refers to the assertion of RESET while power is initially being applied to the processor. A "warm" RESET refers to the assertion of RESET or INIT while V_{CC} and CLK remain within specified operating limits.

Table 17-1 shows the effect of asserting RESET and/or INIT.

Table 17-1. Pentium® Processor Reset Modes

RESET	INIT	BIST Run?	Effect on Code and Data Caches	Effect on FP Registers	Effect on BTB and TLBs
0	0	No	n/a	n/a	n/a
0	1	No	None	None	Invalidated
1	0	No	Invalidated	Initialized	Invalidated
1	1	Yes	Invalidated	Initialized	Invalidated

Toggling either the RESET pin or the INIT pin individually forces the processor to begin execution at address FFFFFFF0H. The internal instruction cache and data cache are invalidated when RESET is asserted (modified lines in the data cache are NOT written back). The instruction cache and data cache are not altered when the INIT pin is asserted without RESET. In both cases, the branch target buffer (BTB) and translation lookaside buffers (TLBs) are invalidated.

After RESET (with or without BIST) or INIT, the processor starts executing instructions at location FFFFFFF0H. When the first Intersegment Jump or Call instruction is executed, address lines A20-A31 are driven low for CS-relative memory cycles and the processor only executes instructions in the lower 1 Mbyte of physical memory. This allows the system designer to use a ROM at the top of physical memory to initialize the system.

RESET is internally hardwired and forces the processor to terminate all execution and bus cycle activity within two clocks. No instruction or bus activity occurs as long as RESET is active. INIT is implemented as an edge triggered interrupt and is recognized when an instruction boundary is reached. As soon as the processor completes the INIT sequence, instruction execution and bus cycle activity continues at address FFFFFFF0H even if the INIT pin is not deasserted.

At the conclusion of RESET (with or without self-test) or INIT, the DX register contains a component identifier. The upper byte contain 05H and the lower byte contains a stepping identifier.

Table 17-2 defines the processor state after RESET, INIT, and RESET with BIST (built in self-test).

Table 17-2. Register State after RESET, INIT and BIST (Sheet 1 of 2)

Storage Element	RESET (No BIST) (Note 1)	RESET (BIST) (Note 1)	INIT
EAX	0	0 if pass	0
EDX	0500+stepping	0500+stepping	0500+stepping
ECX, EBX, ESP, EBP, ESI, EDI	0	0	0
EFLAGS	2	2	2
EIP	0FFF0	0FFF0	0FFF0
CS	selector = F000	selector = F000	selector = F000
	AR = P, R/W, A	AR = P, R/W, A	AR = P, R/W, A
	base = FFFF0000	base = FFFF0000	base = FFFF0000
	limit = FFFF	limit = FFFF	limit = FFFF
DS, ES, FE, GS, SS	selector = 0	selector = 0	selector = 0
	AR = P, R/W, A	AR = P, R/W, A	AR = P, R/W, A
	base = 0	base = 0	base = 0
	limit = FFFF	limit = FFFF	limit = FFFF
(I/G/L)DTR, TSS	selector = 0	selector = 0	selector = 0
	base = 0	base = 0	base = 0
	AR = P, R/W	AR = P, R/W	AR = P, R/W
	limit = FFFF	limit = FFFF	limit = FFFF
CR0	60000010	60000010	Note 2
CR2, 3, 4	0	0	0
DR3–DR0	0	0	0
DR6	FFFF0FF0	FFFF0FF0	FFFF0FF0
DR7	00000400	00000400	00000400
Time Stamp Counter	0	0	Unchanged
Control and Event Select	0	0	Unchanged
TR12	0	0	Unchanged
All other MSR's	Undefined	Undefined	Unchanged
CW	0040	0040	Unchanged
SW	0	0	Unchanged
TW	5555	5555	Unchanged
FIP, FEA, FCS, FDS, FOP	0	0	Unchanged
FSTACK	0	0	Unchanged
SMBASE	30000	30000	Unchanged
Data and Code Cache	Invalid	Invalid	Unchanged

NOTES:
1. Register States are given in hexadecimal format.
2. CD and NW are unchanged, bit 4 is set to 1, all other bits are cleared.

Table 17-2. Register State after RESET, INIT and BIST (Sheet 2 of 2)

Storage Element	RESET (No BIST) (Note 1)	RESET (BIST) (Note 1)	INIT
Code Cache TLB, Data Cache TLB, BTB, SDC	Invalid	Invalid	Invalid

NOTES:
1. Register States are given in hexadecimal format.
2. CD and NW are unchanged, bit 4 is set to 1, all other bits are cleared.

17.3.1 Recognition of Interrupts after RESET

To guarantee recognition of the edge sensitive interrupts (FLUSH#, NMI, R/S#, SMI#) after RESET or after RESET with BIST, the interrupt input must not be asserted until four clocks after RESET is deasserted, regardless of whether or not BIST is run.

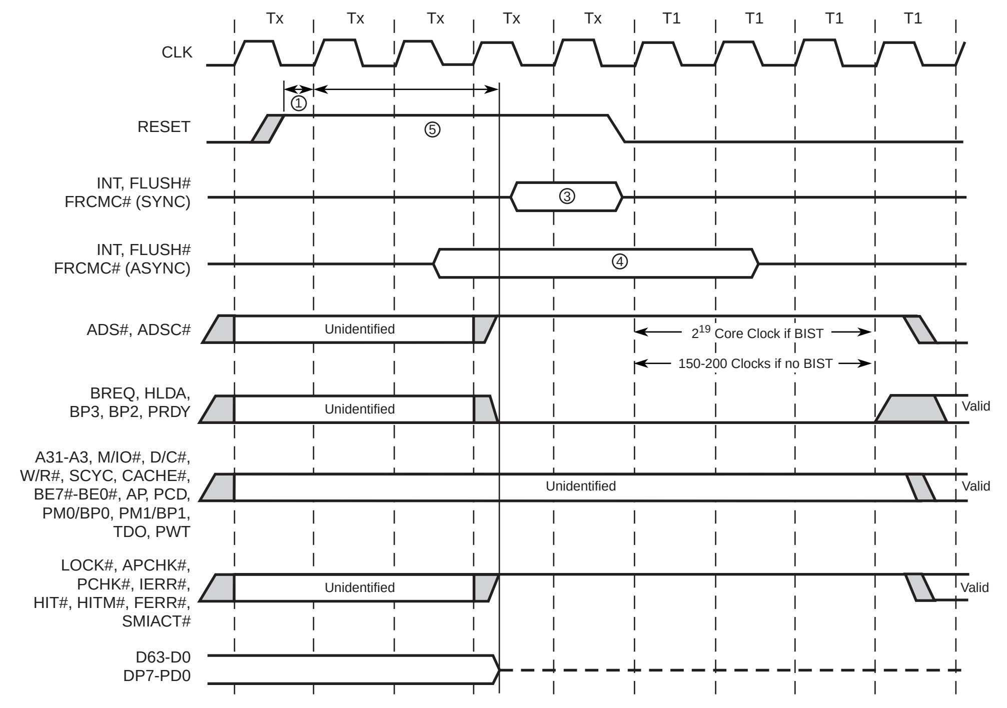
17.3.2 Pin State During/After RESET

The processor recognizes and responds to HOLD, AHOLD, and BOFF# during RESET. Figure 17-1 shows the processor state during and after a power on RESET if HOLD, AHOLD, and BOFF# are inactive. Note that the address bus pins (A31–A3, AP, BE7#–BE0#) and cycle definition pins
(M/IO#, D/C#, W/R#, CACHE#, SCYC, PCD, PWT, PM0/BP0, PM1/BP1 and LOCK#) are undefined from the time RESET is asserted until the start of the first bus cycle.

The following lists the state of the output pins after RESET assuming HOLD, AHOLD, and BOFF# are inactive, boundary scan is not invoked, and no internal parity error is detected.

• High:	LOCK#, ADS#, ADSC#, APCHK#, PCHK#, IERR#, HIT#, HITM#, FERR#, SMIACT#
• Low:	HLDA, BREQ, BP3, BP2, PRDY
• High Independence:	D63–D0, DP7–DP0
• Undefined:	A31–A3, AP, BE7#–BE0#, W/R#, M/IO#, D/C#, PCD, PWT, CACHE#, TDO, SCYC, PM0/BP0, PM1/BP1

Figure 17-1. Pin States during Reset



NOTES:

1. RESET must meet setup and hold times to guarantee recognition on a specific clock edge. If RESET does not need to be recognized on a specific clock edge, it may be asserted asynchronously.
2. At power up, RESET needs to be asserted for 1 ms after V_{CC} and CLK have reached their AC/DC specifications. For warm reset, RESET needs to be asserted for at least 15 clocks while V_{CC} and CLK remain within specified operating limits.
3. If RESET is driven synchronously, FLUSH#, FRCMC# and INIT must be at their valid level and meet setup and hold times to the clock before the falling edge of RESET.
4. If RESET is driven asynchronously, FLUSH#, FRCMC# and INIT must be at their valid level two clocks before and after the falling edge of RESET.
5. An assertion of RESET takes at least two clocks to affect the pins.

A6127-01

17.4 Managing and Designing with the Symmetrical Dual Processing Configuration

17.4.1 Dual Processor Bootup Protocol

17.4.1.1 Bootup Overview

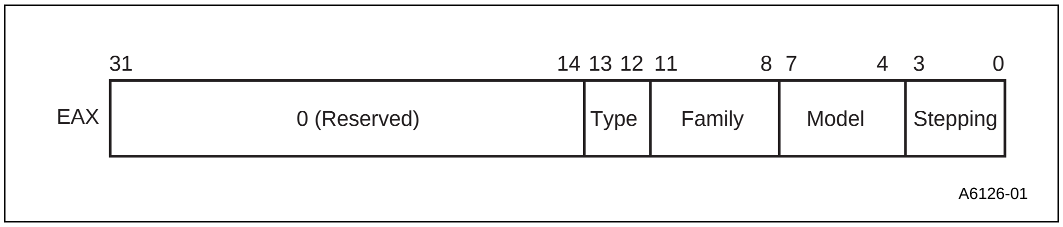
Systems using the embedded Pentium processor may be equipped with a second processor socket. For correct system operation, the processor must be able to identify the presence and type of the second processor (such as a Dual processor). Furthermore, since upgrade processors typically are installed in the field by end users, system configuration may change between any two consecutive power-down/up sequences. The system must therefore have a mechanism to ascertain the system configuration during boot time. The boot up handshake protocol provides this mechanism.

17.4.1.2 BIOS/Operating System Requirements

The BIOS or HAL (hardware abstraction layer) of the operating system software should be generic, independent of the kind of OEM or upgrade processor present in the system. BIOS/HAL are specific to the system hardware, and should not need any change when an upgrade processor is installed. For dual processors, if the BIOS is not DP-ready, it will be up to the operating system to initialize and configure the dual processor appropriately.

The CPUID instruction is used to deliver processor-specific information. The embedded Pentium processor CPUID status has been extended to supply the processor type information which includes “turbo-upgrade” classification (“type” field: bits 13-12 = 0-1). For upgradability with a future Pentium Overdrive processor, system software must allow the type field of the EAX register following the CPUID instruction to contain the values for both the embedded Pentium processor and the Pentium Overdrive processor. Note also that the model field of the CPUID is different for a Pentium OverDrive processor.

Figure 17-2. EAX Bit Assignments for CPUID



17.4.1.3 System Requirements

The number of Dual processors per Primary processor is limited to 1.

This bootup handshake protocol requires enabling the local APIC module using the APICEN pin. The startup IPI must be sent via the local APICs. Once the Dual processor has been initialized, software can later disable the local APIC module using several methods. These methods and their considerations are discussed in “APIC Interrupt Controller” on page 16-213.

The protocol does not preclude more generic multiprocessing systems where multiple pairs of Primary and Dual processors may exist on the system bus.

17.4.1.4 Start-up Behavior

On RESET and INIT (message or pin), the processor begins execution at the reset vector (0FFFFFFF0H). The Dual processor waits for a startup IPI from the BIOS or operating system via the local APIC of the processor. The INIT IPI can be used to put the embedded Pentium processor or Dual processor to sleep (once the INIT IPI is received, the processor must wait for the startup IPI).

The startup IPI is specifically provided to start the Dual processor's execution from a location other than the reset vector, although it also can be used for the processor. The startup IPI is sent by the system software via the local APIC by using a delivery mode of 110B. The startup IPI must include an 8-bit vector that defines the starting address. The starting address = *000 VV 000H*, where *VV* indicates the vector field (in hex) passed through the IPI.

The 8-bit vector defines the address of a 4 Kbyte page in the Intel architecture Real Mode Space (1 Mbyte space). For example, a vector of 0CDH specifies a startup memory address of 000CD000H. This value is used by the processor to initialize the segment descriptor for the upgrade's CS register as follows:

- The CS selector is set to the startup memory address/16 (real mode addressing)
- The CS base is set to the startup memory address
- The CS limit is set to 64 Kbytes
- The current privilege level (CPL) and instruction pointer (IP) are set to 0

Note: Vectors of 0A0H to 0BFH are reserved by Intel. Do *not* use them.

The benefit of the startup IPI is that it does not require the APIC to be software enabled (the APIC must be hardware enabled via the APICEN pin) and does not require the interrupt table to be programmed. Startup IPIs are non-maskable and can be issued at any time to the embedded Pentium processor or Dual processor. If the startup IPI message is not preceded by a RESET or INIT (message or pin), it is ignored.

It is the responsibility of the system software to resend the startup IPI message if there is an error in the IPI message delivery. Although the APIC need not be enabled in order to send the startup IPI, the advantage to enabling the APIC prior to sending the startup IPI is to allow APIC error handling to occur via the APIC error handling entry of the local vector table (ERROR INT or LVT3 at APIC address 0FEE00370H). Otherwise, the system software would have to poll the delivery status bit of the interrupt command register to determine if the IPI is pending (Bit 12 of the ICR=1) and resend the startup IPI if the IPI remains pending after an appropriate amount of time.

17.4.1.5 Dual-Processor Presence Indication

The bootup handshake protocol becomes aware that an additional processor is present through the DPEN# pin. The second processor is guaranteed to drive this signal low during RESETs falling edge. If the system needs to remember the presence of a second processor for future use, it must latch the state of the DPEN# pin during the falling edge of RESET.

17.4.2 Dual-Processor Arbitration

The embedded Pentium processor incorporates a private arbitration mechanism that allows the Primary and Dual processors to arbitrate for the shared processor bus without assistance from a bus controller. The arbitration scheme is architected in such a way that the dual processor pair appears as a single processor to the system.

The processor arbitration logic uses a fair arbitration scheme. The arbitration state machine is designed to efficiently use the processor bus bandwidth. The dual processor pair supports inter-processor pipelining of most bus transactions. Furthermore, the arbitration mechanism does not introduce any dead clocks on bus transactions.

17.4.2.1 Basic Dual-Processor Arbitration Mechanism

The basic set of arbitration premises requires that the processor check the second socket (Socket 7) for a processor every time the processor enters reset. To perform the checking of the Socket 7 and to perform the actual boot sequence, the processor in the 296-pin socket always comes out of reset as the most-recent master (MRM). This requires the part in the Socket 7 to always come out of reset as the least-recent master (LRM).

The LRM processor requests ownership of the processor bus by asserting the private arbitration request pin, PBREQ#. The processor that is currently the MRM and owns the bus grants the bus to the LRM as soon as any pending bus transactions have completed. The MRM grants the bus to the LRM immediately if that processor has a pipelined cycle to issue. The MRM notifies that the LRM can assume ownership by asserting the private arbitration grant pin, PBGNT#. The PBREQ# pin is always the output of the LRM and the PBGNT# is always an input to the LRM.

A processor can park on the processor bus if there are no requests from the LRM. A parked processor can be running cycles or just sitting idle on the bus. If a processor just ran a cycle on the bus and has another cycle pending without an LRM request, the processor runs the second cycle on the bus.

Locked cycles present an exception to the simple arbitration rules. All locked cycles are performed as atomic operations without interrupt from the LRM. An exception to this rule is when a locked access causes an assertion of PHITM# by the LRM. In this case, the MRM grants the bus to the LRM and allows the writeback to complete.

The normal system arbitration pins (HOLD, HLDA, BOFF#) functions the same as in uni-processor mode. Thus, the dual-processor pair always factors the state of the processor bus as well as the state of the local arbitration before actually running a cycle on the processor bus.

17.4.2.2 Dual-Processor Arbitration Interface

Figure 17-3 details the hardware arbitration interface**.**

Note: For proper operation, PBREQ# and PBGNT# must not be loaded by the system.

Figure 17-3. Dual-Processor Arbitration Interface

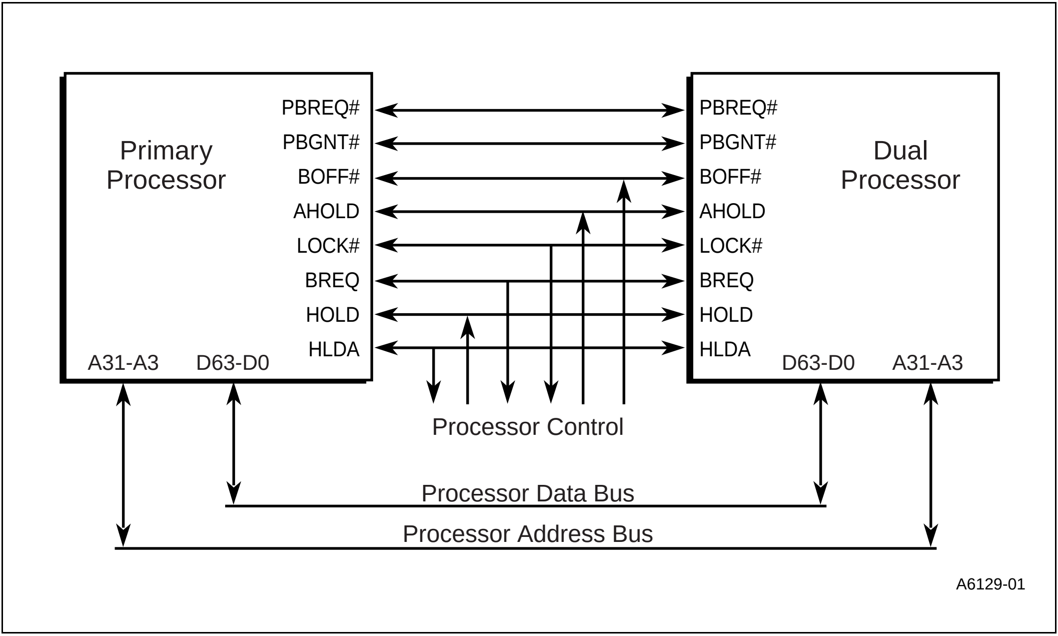


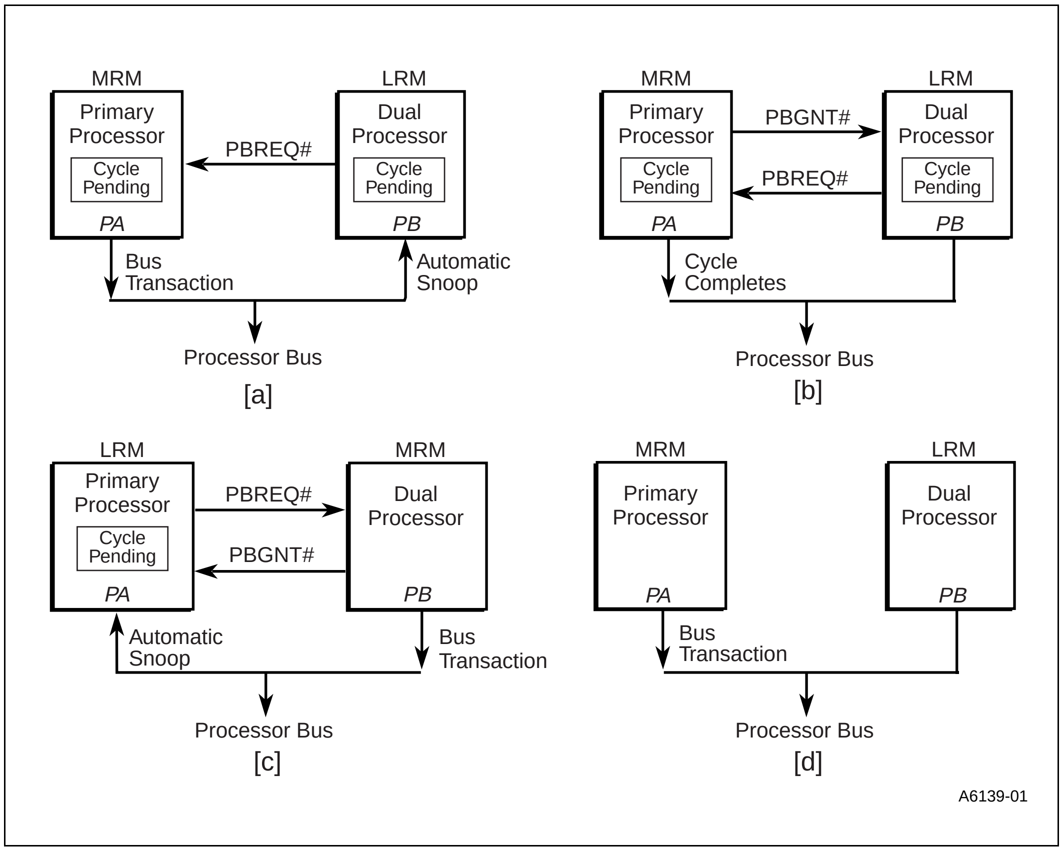
Figure 17-4 shows a typical arbitration exchange.

Diagram (a) of Figure 17-4 shows *PA* running a cycle on the processor bus with a transaction pending. At the same time, *PB* has a cycle pending and has asserted the PBREQ# pin to notify *PA* that *PB* needs the bus.

Diagram (b) of Figure 17-4 shows *PA's* cycle completing with an NA# or the last BRDY#. Note here that *PA* does not run the pending cycle, instead, *PA* grants the bus to *PB* to allow *PB* to run its pending cycle.

In Diagram (c) of Figure 17-4, *PB* is running the pending transaction on the processor bus, and *PA* asserts a request for the bus to *PB*. The bus is granted to *PA*, and Diagram (d) of Figure 17-4 shows *PA* running the last pending cycle on the bus.

Figure 17-4. Typical Dual-Processor Arbitration Example



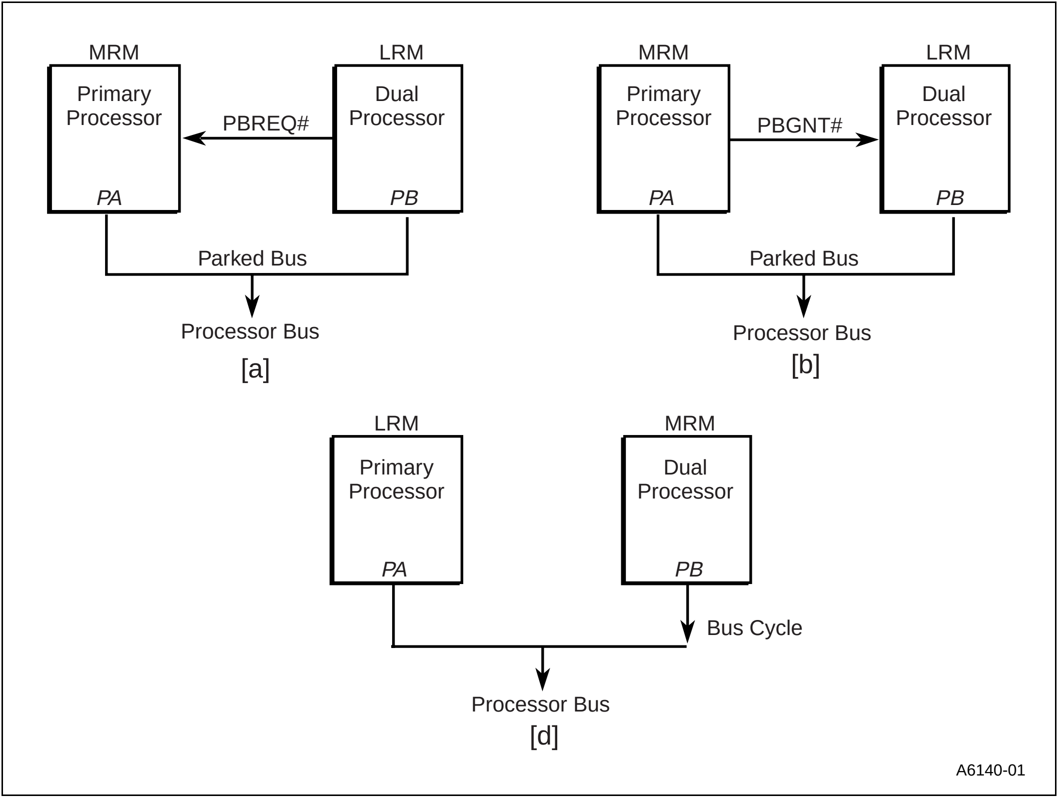
17.4.2.3 Dual-Processor Arbitration from a Parked Bus

When both processors are idle on the processor bus, and the LRM wants to issue an ADS#, there is an arbitration delay in order that it may become the MRM. Figure 17-5 shows how the embedded Pentium processor dual-processor arbitration mechanism handles this case.

This example shows the arbitration necessary for the LRM to gain control of the idle processor bus in order to drive a cycle. In this example, *PA* is the Primary processor, and *PB* is the Dual processor.

Diagram (a) of Figure 17-5 shows *PB* requesting the bus from the MRM (*PA*). Diagram (b) of Figure 17-5 shows *PA* granting control of the bus to *PB*. Diagram (c) of Figure 17-5 shows *PB*, now the MRM, issuing a cycle.

Figure 17-5. Arbitration from LRM to MRM when Bus is Parked



17.4.3 Dual-Processor Cache Consistency

The embedded Pentium processor incorporates a mechanism to maintain cache coherency with the Dual processor. The mechanism allows a dual processor to be inserted into the upgrade socket without special considerations for the system hardware or software. The presence or absence of the dual processor is totally transparent to the system.

17.4.3.1 Basic Cache Consistency Mechanism

A private snoop interface has been added to the embedded Pentium processor. The interface consists of two pins (PHIT#, PHITM#) that only connect between the two sockets. The dual processors arbitrate for the system bus via two private arbitration pins (PBREQ#, PBGNT#).

The LRM processor initiates a snoop sequence for all ADS# cycles to memory that are initiated by the MRM. The LRM processor asserts the private hit indication (PHIT#) if the data accessed (read or written) by the MRM matches a valid cache line in the LRM. In addition, if the data requested by the MRM matches a valid cache line in the LRM that is in the modified state, the LRM asserts the PHITM# signal. The system snooping indication signals (HIT#, HITM#) do not change state as a result of a private snoop.

The processor supports system snooping via the EADS# pin in the same manner in which the processor supports system snooping.

The private snoop interface is bidirectional. The processor that is currently the MRM samples the private snoop interface, while the processor that is the LRM drives the private snoop signals.

The MRM initiates a self backoff sequence if the MRM detects an assertion of the PHITM# signal while running a bus cycle. The self backoff sequence involves the following steps:

1. The MRM allows the cycle that was requested on the bus to finish. However, the MRM ignores the data returned by the system.
2. The MRM-LRM exchanges ownership of the bus (as well as MRM-LRM state) to allow the LRM to write the modified data back to the system.
3. The bus ownership will exchange one more time to allow the original bus master ownership of the bus. At this point the MRM retries the cycle, receiving the fresh data from the system or writing the data again.

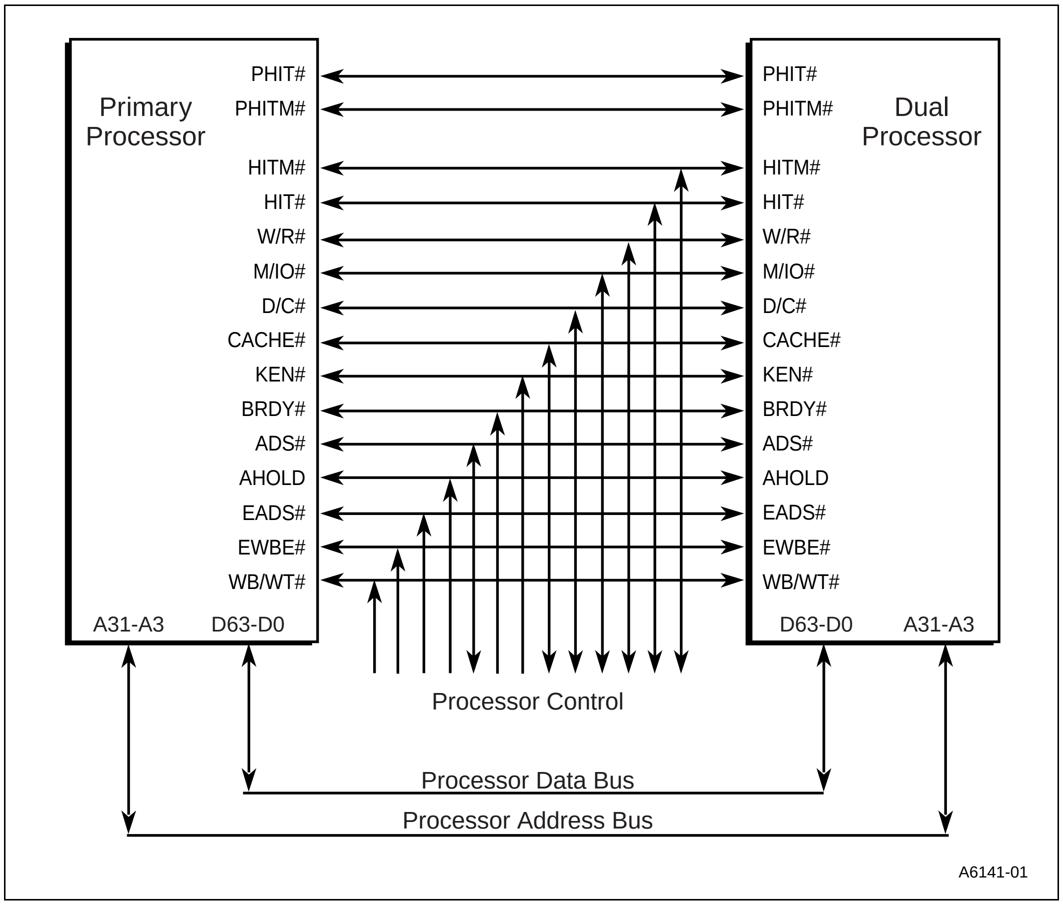
The MRM uses an assertion of the PHIT# signal as an indication that the requested data is being shared with the LRM. Independent of the WB/WT# pin, a cache line is placed in the cache in the shared state if PHIT# is asserted. This makes all subsequent writes to that line externally visible until the state of the line becomes exclusive (E or M states). In a uniprocessor system, the line may have been placed in the cache in the E state. In this situation, all subsequent writes to that line are not visible on the bus until the state is changed to I.

17.4.3.2 Cache Consistency Interface

Figure 17-6 details the hardware cache consistency interface**.**

Note: For proper operation, PHIT# and PHITM# must not be loaded by the system.

Figure 17-6. Cache Consistency Interface



17.4.3.3 Pin Modifications Due to the Dual-Processor

The processor, when operating in dual processing mode, modifies the functionality of the following signals:

- A20M#, ADS#, BE4#–BE0#, CACHE#, D/C#, FERR#, FLUSH#, HIT#, HITM#, HLDA, IGNNE#, LOCK#, M/IO#, PCHK#, RESET, SCYC, SMIACT#, W/R#

Table 17-10 on page 17-251 summarizes the functional changes of all the pins in dual processor mode.

17.4.3.4 Locked Cycles

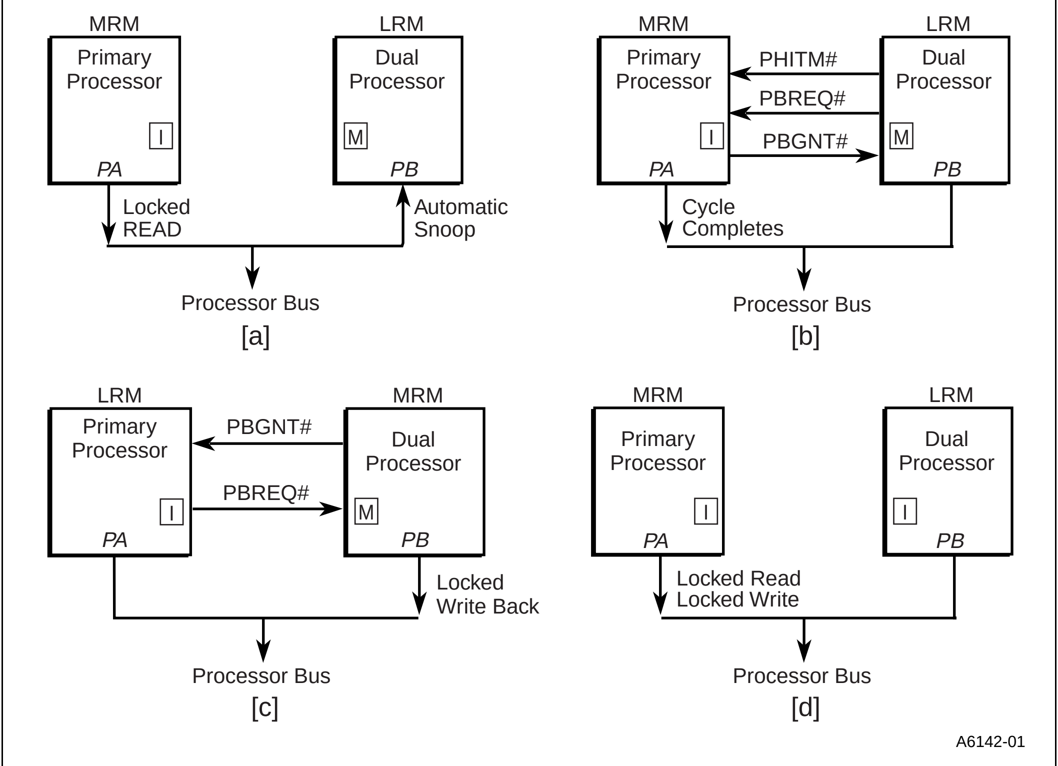
The processor implements atomic bus transactions by asserting the LOCK# pin. Atomic transactions can be initiated explicitly in software by using a LOCK prefix on specific instructions. In addition, atomic cycles may be initiated implicitly for instructions or transactions that perform locked read-modify-write cycles. By asserting the LOCK# pin, the processor indicates to the system that the bus transaction in progress cannot be interrupted.

Lock cycles adhere to the following sequence:

1. An unlocked writeback occurs when a cache line is in the modified state in the MRM processor. Two unlocked write back cycles may be required if the locked item spans two cache lines that are both in the modified state.
2. A locked read to a cache line that is in the shared, exclusive or invalid state is always run on the system bus. The cache line always is moved to the invalid state at the completion of the cycle. A locked read cycle that is run by the MRM could hit a line that is in the modified state in the LRM. In this case, the LRM asserts the PHITM# signal, indicating that the requested data is modified in the LRM data cache. The MRM completes the locked read, but ignores the data returned by the system. The components exchange ownership of the bus, allowing the Modified cache line to be written back with LOCK# still active. The sequence completes with the original bus owner re-running the locked read followed by a locked write. The sequence is as shown in Figure 17-7.

In Figure 17-7, the small box inside each processor indicates the state of an individual cache line in the sequence shown above. Diagram (c) of Figure 17-7 shows the locked writeback occurring as a result of the inter-processor snoop hit to the M-state line.

Figure 17-7. Dual-Processor Cache Consistency for Locked Accesses



17.4.3.5 External Snoop Examples

Example 17-1. During a Write to an M-State Line

The following set of diagrams illustrates the actions performed when one processor attempts a write to a line that is contained in the cache of the other processor. In this situation, the cached line is in the M state in the LRM processor. The external snoop and the write are to the same address in this example. In this example, *PA* is the Primary processor, and *PB* is the Dual processor.

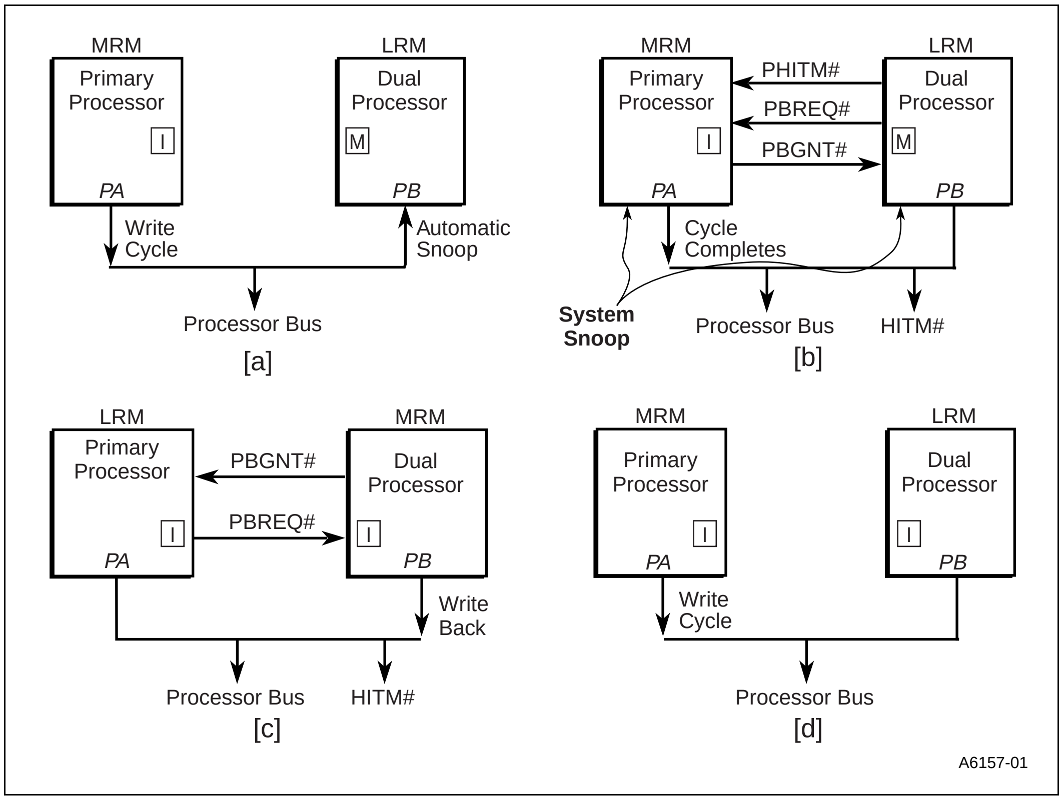
In diagram (a) of Figure 17-8, processor *PA* starts a write cycle on the bus to a line that is in the M state in processor *PB*. Processor *PB* notifies *PA* that the write transaction has hit an M-state line in diagram (b) of Figure 17-8 by asserting the PHITM# signal. The MRM (*PA*) completes the write cycle on the bus as if the LRM processor did not exist.

In this example, an external snoop happens just as the write cycle completes on the bus, but before *PB* has a chance to write the modified data back to the system memory. Diagram (b) of Figure 17-8 shows *PB* asserting the HITM# signal, informing the system that the snoop address is cached in the dual processing pair and is in the modified state. The external snoop in this example is hitting the same line that caused the PHITM# signal to be asserted.

Diagram (c) of Figure 17-8 shows that an arbitration exchange has occurred on the bus, and *PB* is now the MRM. Processor *PB* writes back the M state line; it appears to the system as if a single processor was completing a snoop transaction.

Finally, diagram (d) of Figure 17-8 shows processor *PA* re-running the original write cycle after *PB* has granted the bus back to *PA*.

Figure 17-8. Dual-Processor Cache Consistency for External Snoops



Example 17-2. During an MRM Self-Backoff

The following diagrams show an example in which an external snoop hits an M-state line during a self backoff sequence.

In this example, *PA* is the Primary processor, and *PB* is the Dual processor.

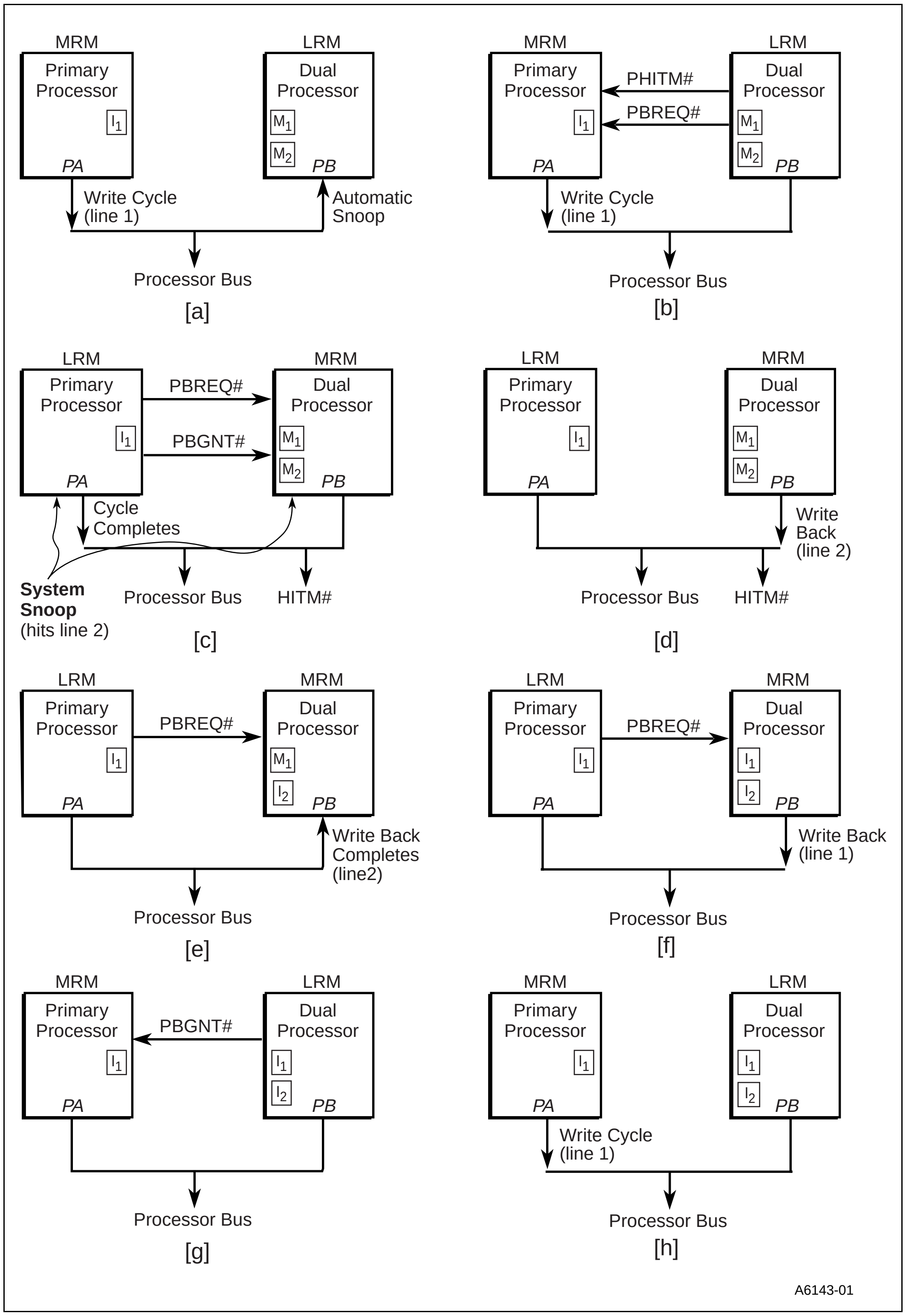
In diagram (a) of Figure 17-9 processor *PA* initiates a write cycle that hits a line that is modified in processor *PB*. In diagram of (b) of Figure 17-9, processor *PB* notifies *PA* that the line is modified in its cache by asserting the PHITM# signal.

Diagram (c) of Figure 17-9 shows an external snoop occurring just as the bus arbitration has exchanged ownership of the bus. Processor *PB* asserts the HITM# signal to notify the system that the external snoop has hit a line in the cache. In this example, the external snoop hits a different line that was just hit on the private snoop.

In diagram (d) of Figure 17-9, processor *PB* takes ownership of the processor bus from *PA*. Processor *PB* initiates a writeback of the data just hit on the external snoop even though a writeback due to the private snoop is pending. The external snoop causes processor *PB* to delay the writeback that was initiated by the private snoop (to line 1).

Diagram (f) of Figure 17-9 shows the writeback of the modified data hit during the initial private snoop. Processor *PA* then restarts the write cycle for the second time, and completes the write cycle in Diagram (h) of Figure 17-9.

Figure 17-9. Dual-Processor Cache Consistency for External Snoops



17.4.3.6 State Transitions Due to Dual-Processor Cache Consistency

The following tables outline the state transitions that a cache line can encounter during various conditions.

Table 17-3. Read Cycle State Transitions Due to Dual-Processor

Present State	Pin Activity	Next State	Description
M	n/a	M	Read hit. Data is provided to the processor core by the cache. No bus activity.
E	n/a	E	Read hit. Data is provided to the processor core by the cache. No bus activity.
S	n/a	S	Read hit. Data is provided to the processor core by the cache. No bus activity.
I	CACHE#(L) & KEN#(L) & WB/WT#(H) & PHIT#(H) & PWT(L)	E	Cache miss. The cacheability information indicates that the data is cacheable. A bus cycle is requested to fill the cache line. PHIT#(H) indicates that the data is not shared by the LRM processor.
I	CACHE#(L) & KEN#(L) & [WB/WT#(L) + PHIT#(L) + PWT(H)]	S	Cache miss. The line is cacheable and a bus cycle is requested to fill the cache line. In this case, either the system or the LRM is sharing the requested data.
I	CACHE#(H) + KEN#(h)	I	Cache miss. The system or the processor indicates that the line is not cacheable.

NOTE: The assertion of PHITM# would cause the requested cycle to complete as normal, with the requesting processor ignoring the data returned by the system. The LRM processor would write the data back and the MRM would retry the cycle. This is called a self backoff cycle.

Table 17-4. Write Cycle State Transitions Due to Dual-Processor

Present State	Pin Activity	Next State	Description
M	n/a	M	Write hit. Data is written directly to the cache. No bus activity.
E	n/a	M	Write hit. Data is written directly to the cache. No bus activity.
S	PWT(L) & WB/WT#(H)	E	Write hit. Data is written directly to the cache. A write-through cycle is generated on the bus to update memory and invalidate the contents of other caches. The LRM invalidates the line if it is sharing the data. The state transition from S to E occurs AFTER the write completes on the processor bus.
S	PWT(H) + WB/WT#(L)	S	Write hit. Data is written directly to the cache. A write-through cycle is generated on the bus to update memory and invalidate the contents of other caches. The LRM invalidates the line if it is sharing the data.
I	n/a	I	Write miss (the Pentium® processor does not support write allocate). The LRM invalidates the line if it is sharing the data.

Table 17-5. Inquire Cycle State Transitions Due to External Snoop

Present State	Next State (INV=1)	Next State (INV=0)	Description
M	I	S	Snoop hit to an M-state line. HIT# and HITM# are asserted, followed by a writeback of the line.
E	I	S	Snoop hit. HIT# will be asserted.
S	I	S	Snoop hit. HIT# will be asserted.
I	I	I	Snoop miss.

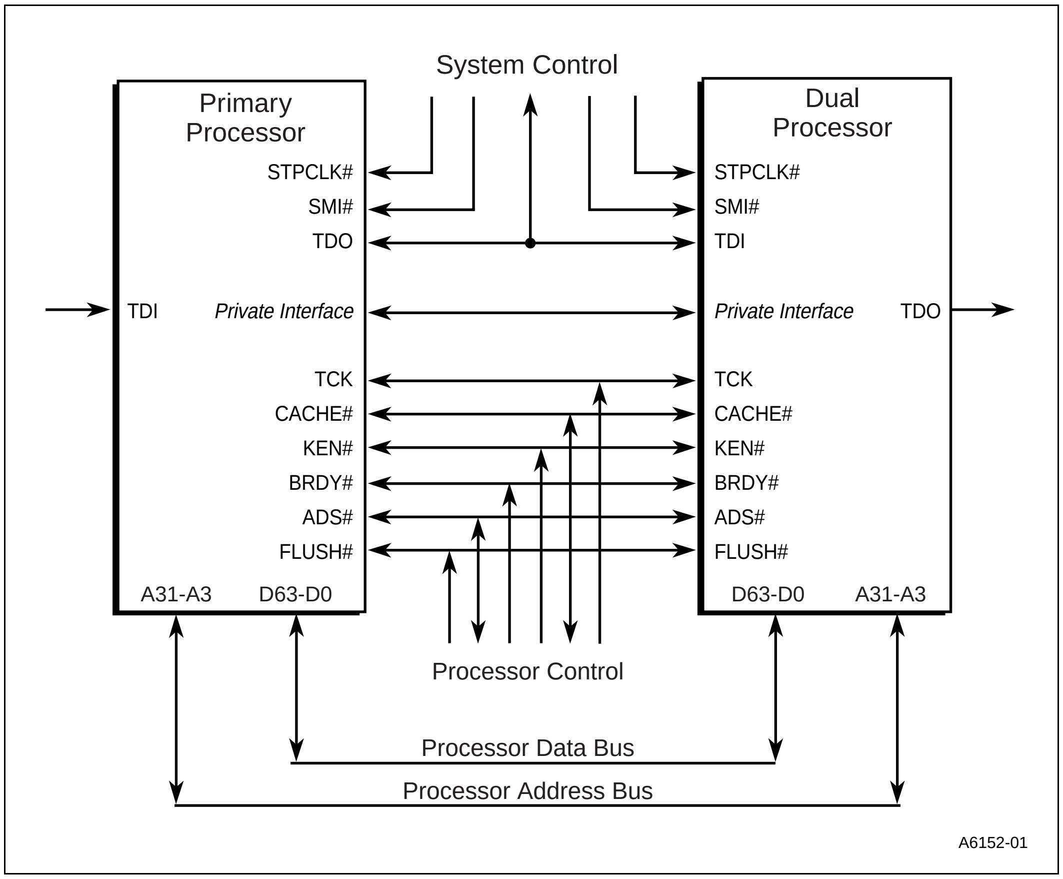
Table 17-6. State Transitions in the LRM Due to Dual-Processor "Private" Snooping

Present State	Next State (MRM Write)	Next State (MRM Read)	Description
M	I	S	Snoop hit to an M state line. PHIT# and PHITM# are asserted, followed by a write-back of the line. Note that HIT# and HITM# are NOT asserted.
E	I	S	Snoop hit. PHIT# is asserted.
S	I	S	Snoop hit. PHIT# is asserted.
I	I	I	Snoop miss.

17.5 Designing with Symmetrical Dual Processors

Figure 17-10 shows how a typical system might be configured to support the Dual processor.

Figure 17-10. Dual-Processor Configuration



Refer to Table 17-10 on page 17-251 for a complete list of dual processor signal connection requirements.

17.5.1 Dual Processor Bus Interface

The processor in the dual-processor configuration is designed to have an identical bus interface to a standard processor system. The processor in dual processor mode has the capability to run the following types bus of cycles:

- Single reads and writes from one processor.
- Burst reads and writes from one processor.
- Address pipelining with up to two outstanding bus cycles from one processor.
- Inter-processor address pipelining with up to two outstanding bus cycles, one from each processor.

All cycles run by the two processors are clock-accurate to corresponding processor bus cycles.

17.5.1.1 Intra- and Inter-Processor Pipelining

In uni-processor mode, the embedded Pentium processor supports bus pipelining with the use of the NA# pin. The bus pipelining concept has been extended to the dual processor pair by allowing inter-processor pipelining. This mechanism allows an exchange between LRM and MRM on assertions of NA#.

When NA# is sampled low, the current MRM processor may drive one more cycle onto the bus or it may grant the address bus and the control bus to the LRM. The MRM gives the bus to the LRM only if its current cycle can have another cycle pipelined into it.

The cacheability (KEN#) and cache policy (WB/WT#) indicators for the current cycle are sampled either in the same clock in which NA# is sampled or with the first BRDY# of the current cycle, whichever comes first.

There are no restrictions on NA# due to dual processing mode.

Inter-processor pipelining is not supported in some situations, as shown in Table 17-7.

Table 17-7. Primary and Dual Processor Pipelining

Cycle Types		Primary and Dual Processor Pipelining		
		Inter-processor	Intra-processor	
First Cycle	**Pipelined Cycle**	**Primary<>Dual**	**Primary<>Primary**	**Dual<>Dual**
Write Back	X	No	No	No
LOCK#	X	No	No	No
X	Write Back	No	No	No
X	LOCK#	No	No	No
Write	**Write**	**No**	**Yes**	**Yes**
Write	Read	Yes	Yes	Yes
Read	Write	Yes	Yes	Yes
Read	Read	Yes	Yes	Yes
I/O	I/O†	**Yes**	**No**	**No**

† I/O write cycles may not be inter-processor pipelined into I/O write cycles

The table indicates that, unlike the uni-processor system, back-to-back write cycles are never pipelined between the two processors.

The processor alone may pipeline I/O cycles into non-I/O cycles, non-I/O cycles into I/O cycles, and I/O cycles into I/O cycles only for OUTS or INS (e.g., string instructions). I/O cycles may be pipelined in any combination (barring writes into writes) between the Primary and Dual processors.

17.5.1.2 FLUSH# Cycles

The on-chip caches can be flushed by asserting the FLUSH# pin. The FLUSH# pin must be connected to both the Primary and Dual processor parts. All cache lines in the instruction cache and all lines in the data cache that are not in the modified state are invalidated when the FLUSH# pin is asserted. All modified lines in the data cache are written back to system memory and then marked as invalid in the data cache. The processor runs a special bus cycle to indicate that the flush process has completed.

The embedded Pentium processor incorporates the following mechanism to present to the system a unified view of the cache flush operation when used with a Dual processor part:

1. FLUSH# is asserted by the system.
2. The Dual processor requests the bus (if it is not already MRM when FLUSH# is recognized). The Dual processor will always perform the cache flush operation first, but will not run a flush special cycle on the system bus.
3. The Dual processor completes writebacks of modified cache lines, and invalidates all others.
4. Once the Dual processor caches are completely invalid, the processor grants the bus to the Primary processor.
5. The Primary processor completes any pending cycles. The Primary processor may have outstanding cycles if the Dual processor initiated its flush operation prior to the Primary processor completing pending operations.
6. Primary processor flushes both of its internal caches and runs the cache flush special cycle. The Primary processor maintains its status of MRM. The Dual processor halts all code execution while the Primary processor is flushing its caches, and does not begin executing code until it recognizes the flush acknowledge special cycle.

The atomic flush operation assumes that the system can tolerate potentially longer interrupt latency during flush operations. The interrupt latency in a dual processor system can be double the interrupt latency in a single processor system during flush operations.

The processor primary cache can be flushed using the WBINVD instruction. In a dual processor system, the WBINVD instruction only flushes the cache in the processor that executed the instruction. The other processor’s cache will be intact.

If the FLUSH# signal is deasserted before the corresponding Flush Acknowledge cycle, the FLUSH# signal *must* not be asserted again until the Flush Acknowledge cycle is completed. Similarly, if the FLUSH# signal is asserted in dual processing mode, it must be deasserted at least one clock prior to BRDY# of the Flush Acknowledge cycle to avoid dual-processor arbitration problems. This requirement does not apply to a uni-processor system. In a dual processor system, a single Flush Acknowledge cycle is generated after the caches in both processors have been flushed.

Warning: If FLUSH# is recognized active a second time by the Primary and Dual processors prior to the completion of the Flush Acknowledge special cycle, the private bus arbitration state machines will be corrupted.

17.5.1.3 Arbitration Exchange with Bus Parking

The dual processor pair supports a number of different types of bus cycles. Each processor can run single-transfer cycles or burst-transfer cycles. A processor can only initiate bus cycles if it is the MRM. To gain ownership of the bus, the LRM processor requests the bus from the MRM processor by asserting PBREQ#.

In response to PBREQ# the MRM grants the address and control buses to the LRM by asserting PBGNT#. If NA# is not asserted or if the current cycle on the bus is not capable of being pipelined, the MRM waits until the end of the active cycle before granting the bus to the LRM. Once PBGNT# is asserted, since the bus is idling, the LRM immediately becomes the MRM. While the MRM, the processor owns the address and the control buses and can therefore start a new cycle.

17.5.1.4 BOFF#

If BOFF# is asserted, the dual-processor pair immediately (in the next clock) floats the address, control, and data buses. Any bus cycles in progress are aborted, and any data returned to the processor in the clock in which BOFF# is asserted is ignored. In response to BOFF#, Primary and Dual processors float the same pins as when HOLD is active.

The Primary and Dual processors may reorder cycles after a BOFF#. The reordering occurs if there is inter-processor pipelining at the time of the BOFF#, but the system cannot change the cacheability of the cycles after the BOFF#. Note that there could be a change of bus ownership transparent to the system while the processors are in the backed-off state. Table 17-8 illustrates the flow of events which would result in cycle reordering due to BOFF#:

Table 17-8. Cycle Reordering Due to BOFF#

Time†	Processor A	System	Processor B
0	ADS# driven	--	--
1	--	NA# active	--
2	--	--	ADS# driven
3	Bus float	BOFF# active	Bus float
4	--	EADS# active	--
5	--	--	HITM# driven
6	--	BOFF# inactive	--
7	--	--	Write back ‘M’ data
8	--	BRDY#s	--
9	--	--	Restart ADS#
10	Restart ADS#	--	--

† Time is merely sequential, NOT measured in CLKs.

17.5.1.5 Bus Hold

The processor supports a bus hold/hold acknowledge protocol using the HOLD and HLDA signals. When the processor completes all outstanding bus cycles, it releases the bus by floating the external bus, and driving HLDA active. HLDA normally is driven two clocks after the later of the last BRDY# or HOLD being asserted, but may be up to six clocks due to active internal APIC cycles. Because of this, it is possible that an additional cycle may begin after HOLD is asserted but before HLDA is driven. Therefore, asserting HOLD does not prevent a dual-processor arbitration from occurring before HLDA is driven out. Even if an arbitration switch occurs, no new cycles are started after HOLD has been active for two clocks.

17.5.2 Dual Processing Power Management

17.5.2.1 STPCLK#

The Primary and Dual processor STPCLK# signals may be tied together or left separate. Refer to Chapter 24, “Power Management.” for more information on stop clock and Autohalt.

17.5.2.2 System Management Mode

The embedded Pentium processor supports system management mode (SMM) with a processor inserted in the upgrade socket. SMM provides a means to implement power management functions and operating system independent functions. SMM consists of an interrupt (SMI), an alternate address space and an instruction (RSM). SMM is entered by asserting the SMI# pin or delivering the SMI interrupt via the local APIC.

Although SMM functions the same when a Dual processor is inserted in Socket 5/Socket 7, the dual processor operation of the system must be carefully considered. The SMI# pins may be tied together or not, depending upon the power management features supported.

17.5.3 Other Dual-Processor Considerations

17.5.3.1 Strong Write Ordering

The ordering of write cycles in the processor can be controlled with the EWBE# pin. During uniprocessor operation, the EWBE# pin is sampled by the processor with each BRDY# assertion during a write cycle. The processor stalls all subsequent write operations to E or M state lines if EWBE# is sampled inactive. If the EWBE# pin is sampled inactive, it continues to be sampled on every clock until it is found to be active.

In dual processing mode, each processor tracks EWBE# independently of bus ownership. EWBE# is sampled and handled independently between the two processors. Only the processor that owns the bus (MRM) samples EWBE#. Once sampled inactive, the processor stalls subsequent write operations.

17.5.3.2 Bus Snarfing

The dual processor pair does not support cache-to-cache transfers (bus snarfing). If a processor *PB* requires data that is modified in processor *PA*, processor *PA* writes the data back to memory. After *PA* has completed the data transfer, *PB* runs a read cycle to memory. Where *PA* is either the Primary or the Dual processor, and *PB* is the other processor.

17.5.3.3 Interrupts

A processor may need to arbitrate for the use of the bus as a result of an interrupt. However, from the simple arbitration model used by the embedded Pentium processor, an interrupt is not a special case. There is no interaction between dual-processor support and the interrupt model in the embedded Pentium processor.

17.5.3.4 INIT Sequences

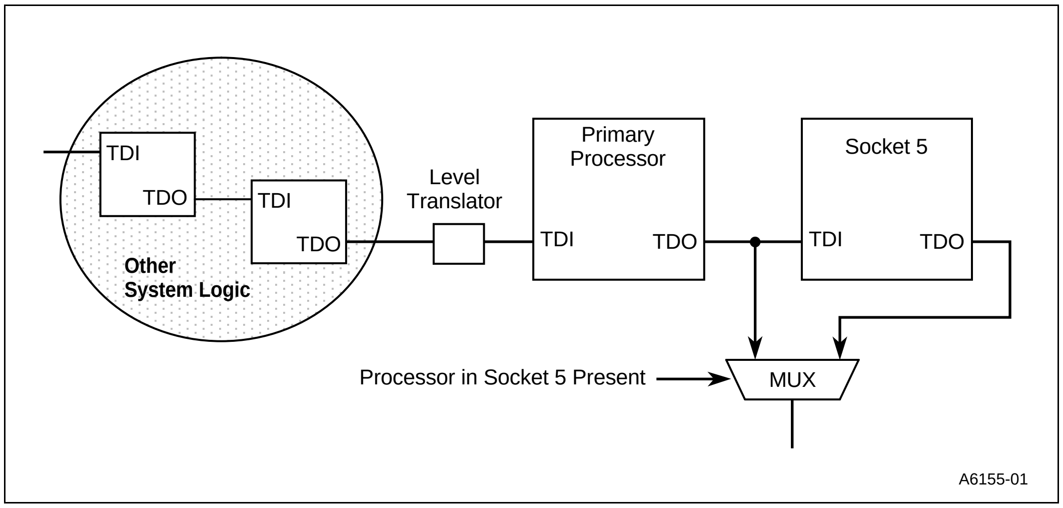
The INIT operation in dual-processor mode is exactly the same as in uni-processor mode. The two INIT pins must be tied together. However, in dual processor mode, the Primary processor must send an IPI and a starting vector to the Dual processor via the local APIC modules.

17.5.3.5 Boundary Scan

The embedded Pentium processor supports the full IEEE JTAG specification. The system designer is responsible to configure an upgrade ready system in such a way that the addition of a Dual processor in Socket 7 allows the boundary scan chain to functional as normal. This could be implemented with a jumper in Socket 7 that connects the TDI and TDO pins. The jumper would then be removed when the dual processor is inserted.

Alternatively, Socket 7 could be placed near the end of the boundary scan chain in the system. A multiplexer in the system boundary scan logic could switch between the TDO of the Primary and the dual processors as a Dual processor part is inserted. An illustration of this approach is shown in Figure 17-11.

Figure 17-11. Dual-Processor Boundary Scan Connections



17.5.3.6 Presence of a Processor in Socket 7

The Dual processor drives the DPEN# signal low during RESET to indicate to the Primary processor that a processor is present in Socket 7. The processor samples this line during RESETs falling edge.

DPEN# shares a pin with the APIC PICD0 signal.

17.5.3.7 MRM Processor Indication

In a dual-processor system, the D/P# (Dual processor/Primary processor Indication) signal indicates which processor is running a cycle on the bus. Table 17-9 shows how the external hardware can determine which processor is the MRM.

Table 17-9. Using D/P# to Determine MRM

D/P#	Bus Owner
0	Primary processor is MRM
1	Dual processor is MRM

D/P# can be sampled by the system with ADS# to determine which processor is driving the cycle on the bus. D/P# is driven only by the processor when operating as the Primary processor. Because of this, this signal is never driven by the Dual processor.

17.5.4 Dual-Processor Pin Functions

All the inputs pins are sampled with bus clock or test clock, and therefore, must meet setup and hold times with respect to the rising edge of the appropriate clock. In the dual-processor configuration, the RESET and FLUSH# pins have been changed to be synchronous (i.e., to meet setup and hold times). There have been no changes to the other existing input pins.

If the FLUSH# signal is deasserted before the corresponding FLUSH ACK cycle, the FLUSH# signal must not be asserted again until the FLUSH ACK cycle is generated. This requirement does not apply to a uni-processor system. In a dual processor system, a single FLUSH ACK cycle is generated after the caches in both processors have been flushed.

All system output pins are driven from the rising edge of the bus clock and meet maximum and minimum valid delays with respect to the bus clock. TDO is driven with respect to the rising edge of TCK and PICD0–PICD1 are driven with respect to the rising edge of PICCLK.

Table 17-10 summarizes the functional changes of all the pins in dual-processor mode.

Table 17-10. Dual-Processor Pin Functions vs. Pentium® Processor (Sheet 1 of 4)

Pin Name	I/O	Load (Note 1)	Same? (Note 2)	Tied Together? (Note 3)	Comments
A31–A3	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it three-states these signals for one CLK.
A20M#	I	Y	Y	Yes	Used in virtual mode and possibly in real mode by DOS and DOS extenders. Internally masked by the Dual processor.
ADS#, ADSC#	I/O O	Y	N	Yes	ADS# and ADSC# are three-stated by the LRM processor in order to allow the MRM processor to begin driving them. There are no system implications.
AHOLD	I	Y	Y	Yes	
AP	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it three-states this signal for one CLK.

NOTES:

1. “Load” indicates whether the pin would introduce a capacitive load to the system board due to the presence of the dual processor.
2. “N” indicates that there is a minor functional change to the pin(s) either as an enhancement to the embedded Pentium processor or due to dual processor operation.
3. “Yes” means that both processors must see the same value on the pin(s) for proper dual-processor operation. “No” means that the system must provide the signal to each processor independently. “May be” means that the system designer can choose to provide the signal to both processors or provide independent signals to each processor.

Table 17-10. Dual-Processor Pin Functions vs. Pentium® Processor (Sheet 2 of 4)

Pin Name	I/O	Load (Note 1)	Same? (Note 2)	Tied Together? (Note 3)	Comments
APCHK#	O	N	Y	No	Requires a system OR function.
BE7–BE5# BE4# –BE0#	O I/O	Y Y	N N	Yes Yes	When the MRM becomes the LRM (and issues PBGNT#), it three-states these signals for one CLK. BE3#–BE0# are used by the local APIC modules to load the APIC_ID at RESET. BE3#–BE0# will be three-stated by the Primary and Dual processors during RESET.
BF	I	Y	n/a	Yes	
BOFF#	I	Y	Y	Yes	
BP3–BP0	O	N	N	No	BP3–BP0 now only indicates breakpoint match in the I/O clock. Each processor must have different breakpoints. Note that BP1–BP0 are muxed with PM1–PM0.
BRDY#, BRDYC#	I	Y	Y	Yes	
BREQ	O	Y	N	Yes	The MRM drives this signal as a combined bus cycle request for itself and the LRM.
BUSCHK#	I	Y	Y	Yes	
CACHE#	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it three-states this signal for one CLK.
CLK	I	Y	Y	Yes	Both processors must use the same system clock.
CPUTYP	I	Y	n/a	No	
D/C#	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it three-states this signal for one CLK.
D/P#	O	n/a	n/a	No	The Primary processor always drives this signal. This output is not defined on the Dual processor.
D63–D0	I/O	Y	Y	Yes	
DP7–DP0	I/O	Y	Y	Yes	
EADS#	I	Y	Y	Yes	
EWBE#	I	Y	Y	Yes	This signal is sampled active with BRDY#, but inactive asynchronously. For optimized performance (minimum number of write E/M stalls) the chip set/platform should allow a dead clock between buffer going empty to buffer going full. This allows this signal to be completely independent between the two processors, rather than having one stall internal cache writes due to the other filling the external buffer.
FERR#	O	Y	Y	Yes	Used for DOS floating-point compatibility. The Primary processor drives this signal. The Dual processor never drives this signal.

NOTES:

1. “Load” indicates whether the pin would introduce a capacitive load to the system board due to the presence of the dual processor.
2. “N” indicates that there is a minor functional change to the pin(s) either as an enhancement to the embedded Pentium processor or due to dual processor operation.
3. “Yes” means that both processors must see the same value on the pin(s) for proper dual-processor operation. “No” means that the system must provide the signal to each processor independently. “May be” means that the system designer can choose to provide the signal to both processors or provide independent signals to each processor.

Table 17-10. Dual-Processor Pin Functions vs. Pentium® Processor (Sheet 3 of 4)

Pin Name	I/O	Load (Note 1)	Same? (Note 2)	Tied Together? (Note 3)	Comments
FLUSH#	I	Y	Y	Yes	In a dual-processor system, the flush operation is atomic with a single flush acknowledge bus cycle. Therefore, FLUSH# must not be re-asserted until the corresponding FLUSH ACK cycle is generated.
FRCMC#	I	N	Y	Yes	Both processors must be in Master mode. A processor in the Socket 7 cannot be used as a Checker.
HIT#	I/O	Y	N	Yes	This signal is asserted by the MRM based on the combined outcome of the inquire cycle between the two processors.
HITM#	I/O	Y	N	Yes	See HIT#.
HLDA	I/O	Y	N	Yes	Driven by the MRM.
HOLD	I	Y	Y	Yes	
IERR#	O	N	Y	No	
IGNNE#	I	Y	Y	Yes	The Dual processor ignores this signal.
INIT	I	N	N	Yes	In dual-processor mode, the Dual processor requires an IPI during initialization.
INTR/LINT0	I	N	N	May be	If the APIC is enabled, this pin is a local interrupt. If the APIC is hardware disabled, this pin function is not changed.
INV	I	Y	Y	Yes	
KEN#	I	Y	Y	Yes	
LOCK#	I/O	Y	N	Yes	The LRM samples the value of LOCK#, and drives the sampled value in the clock in which it gets ownership of the dual-processor bus. If sampled active, then the LRM keeps driving the LOCK# signal until ownership changes again.
M/IO#	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it three-states this signal for one CLK.
NA#	I	Y	Y	Yes	
NC	n/a	N	Y	No	
NMI/LINT1	I	N	Y	May be	If the APIC is enabled, then this pin is a local interrupt. If the APIC is hardware disabled, this pin function is not changed.
PBGNT#	I/O	n/a	n/a	Yes	This signal is always driven by one of the processors.
PBREQ#	I/O	n/a	n/a	Yes	This signal is always driven by one of the processors.
PCD	O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it three-states this signal for one CLK.

NOTES:

1. "Load" indicates whether the pin would introduce a capacitive load to the system board due to the presence of the dual processor.
2. "N" indicates that there is a minor functional change to the pin(s) either as an enhancement to the embedded Pentium processor or due to dual processor operation.
3. "Yes" means that both processors must see the same value on the pin(s) for proper dual-processor operation. "No" means that the system must provide the signal to each processor independently. "May be" means that the system designer can choose to provide the signal to both processors or provide independent signals to each processor.

Table 17-10. Dual-Processor Pin Functions vs. Pentium® Processor (Sheet 4 of 4)

Pin Name	I/O	Load (Note 1)	Same? (Note 2)	Tied Together? (Note 3)	Comments
PCHK#	O	N	Y	May be	May be wire-ANDed together in the system, tied together, or the chip set may have two PCHK# inputs for dual-processor data parity.
PEN#	I	Y	Y	Yes	
PHIT#	I/O	n/a	n/a	Yes	This signal is always driven by one of the processors.
PHITM#	I/O	n/a	n/a	Yes	This signal is always driven by one of the processors.
PHITM#	I/O	n/a	n/a	Yes	This signal is always driven by one of the processors.
PICCLK	I	Y	n/a	Yes	
PICD1–PICD0	I/O	Y	n/a	Yes	
PM1–PM0	O	N	N	No	Each processor may track different performance monitoring events. Note that PM1–PM0 are mux'd with BP1–BP0.
PRDY	O	N	Y	No	
PWT	O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it three-states this signal for one CLK.
R/S#	I	N	Y	No	
RESET	I	Y	Y	Yes	In dual-processor mode, RESET must be synchronous to the processor CLK that goes to the Primary and Dual processors.
SCYC	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it three-states this signal for one CLK.
SMI#	I	N	Y	May be	Refer to Chapter 24.
SMIACT#	O	N	Y	Yes	Refer to Chapter 24.
STPCLK#	I	n/a	n/a	May be	Refer to Chapter 24.
TCK	I	n/a	n/a	May be	System dependent
TDI	I	n/a	n/a	No	System dependent
TDO	O	n/a	n/a	No	System dependent
TMS	I	n/a	n/a	May be	System dependent
TRST#	I	n/a	n/a	May be	System dependent
V_{CC}	I	N	N	Yes	V_{CC} on the processor must be connected to 3.3 V.
V_{SS}	I	N	Y	Yes	
W/R#	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it three-states this signal for one CLK.
WB/WT#	I	Y	Y	Yes	

NOTES:

1. "Load" indicates whether the pin would introduce a capacitive load to the system board due to the presence of the dual processor.
2. "N" indicates that there is a minor functional change to the pin(s) either as an enhancement to the embedded Pentium processor or due to dual processor operation.
3. "Yes" means that both processors must see the same value on the pin(s) for proper dual-processor operation. "No" means that the system must provide the signal to each processor independently. "May be" means that the system designer can choose to provide the signal to both processors or provide independent signals to each processor.