Using a PCI Bus as the I/O Bus on an Am29030™ Microprocessor Design

Advanced Micro Devices

Application Note

by David Stoenner

This application note describes how the Peripheral Component Interconnect (PCI) bus can be used as the I/O bus portion on a two-bus microcontroller design. Additionally, one programmable logic part, the MACH® 220 device, is added for the entire control logic for both the memory control and the signal conversion between the Am29030™ microprocessor and Revision 2.0 of the PCI I/O bus.

INTRODUCTION

The Peripheral Component Interconnect (PCI) local bus is a high-performance, 32-bit or 64-bit bus with multiplexed address and data lines. It is intended for use as an interconnect mechanism between highly integrated peripheral controller components, peripheral add-in boards, and processor/memory systems. Because the bus is multiplexed, the number of pin contacts is reduced, making the peripheral cheaper to implement. To accomplish this, PCI adds a layer between the CPU and the peripherals, resulting in a processor-independent bus that can be used for a variety of CPUs and processor speeds.

The PCI bus is a well-defined interface that is specified for the signals as well as the physical characteristics of the connector and the loading. Since inexpensive peripherals will become available to meet PC needs, it would therefore be advantageous to extend the use of the PCI bus into the embedded processor market. Since the 29K™ Family is one of the premiere RISC families used in embedded control, this application note shows how easily the PCI bus can be adapted to the 29K Family processor to meet the requirement of an I/O bus.

THE PCI BUS

In order to create an industry standard for PCI, the PCI Special Interest Group (SIG) has defined a specification. The *PCI Local Bus Specification*, Revision 2.0, defines the protocol, electrical, mechanical, and configuration specifications for PCI local bus components and expansion boards. (See page 5 for information on ordering the specification.)

Basics of the PCI Bus

The PCI bus is an address/data multiplexed bus. Additionally, the control signals are multiplexed with the byte control signals. The PCI bus supports a burst protocol that accepts an address with multiple data packets. This protocol is identical to the 29K Family, making the interface design easily accomplished.

Arbitration on the PCI bus ranges from a simple design of direct priority to a more complicated design of using arbitration under the existing bus master for the next bus master. The protocol also supports bus master preemption if desired. Additionally, complex caching cycles for multiprocessors also are defined. With forethought, the PCI SIG specified that these features be optional and not required.

Use of the PCI Bus in this Design

This design focuses on the minimum requirements of the PCI bus, with the exception of parity. Currently, parity is not supported on this design because of the complexity of the chips that would need to be added to the design. Bus master arbitration is limited to the Am29030 processor “parked” on the PCI bus being the main master, and the peripheral chips requesting the bus as necessary. No support for time-out and bus master preemption is included, so it is the responsibility of each master to ensure that each peripheral does not use the bus to the exclusion of the other peripherals.

The memory design is taken directly from the original EZ-030 demonstration board (see the *EZ-030 Demonstration Board Theory of Operation* application note), except that the MACH220 device replaces all the discrete PAL® devices of the EZ-030 board design. Since the EZ-030 board supports a memory clock (MEMCLK) of 16 MHz, this application note also focuses at 16 MHz for both the memory and the PCI bus clock. The PCI bus clock is defined from 0 to 33 MHz, so this falls within that limit. Also, 33-MHz clock rates, and therefore the bus, require more careful layout than the 16-MHz clock.

The rest of this application note focuses on the MACH220 device and the extra circuitry needed to implement the control portion of the PCI bus. Schematics for this design are shown in Appendix A and the PAL equations are listed in Appendix B.

Publication# **18468** Rev. **A** Amendment **/0**
Issue Date: **May 1994**

THEORY OF OPERATIONS

As discussed in the introduction, this design is a modification of the original EZ-030 board design. The major change is that the three control PAL devices of the EZ-030 board design are merged into one large control PAL device, the MACH220 device, shown in the schematic on page 8. Pages 9 through 12 show the need of multiplexing the address and data bus together, and pages 13 through 15 depict the three separate PCI connectors.

Figures 1 and 2 show block diagrams for the original and modified board designs, respectively.

The PCI bus is a multiplexed address, data, and control bus to conserve on pins. The Am29030 processor is not a multiplexed address and data bus part, so in the schematic on page 12 , the 12 octal parts required to do the address/data multiplexing are shown. U20 through U23, 74F244s, support placing the initial address in the first clock cycle of the frame. (Note that the upper four bits, A31–A28, are driven onto the PCI bus with 0s. These bits are used on the local Am29030 processor side to decode the various PCI bus cycles that are to be accomplished.)

U16 through U19, 74F245s, are the data buffers used to input or output the data off or onto the PCI bus. These parts will be used for Am29030 processor PCI bus cycles, as well as when another PCI bus master is using the DRAM memory.

If another bus master takes over the PCI bus and wants to perform a DRAM memory cycle, the parts U24 through U27 capture the address on the first clock cycle of the FRAME signal. The parts are 20L8s used to capture addresses A31–A10, A1, and A0 in a clock-enabled register configuration. The equations for the 20L8s are in Appendix B. The address bits A9–A2 are captured in a 22V10, which is a combination clock-enabled register and a counter that supports the burst feature that PCI has defined and enables the maximum throughput allowed. This PAL device also detects an upper address bit carry if the requesting master tries to burst across a 1K address boundary, which is then used by the PCI controller to signal a STOP condition on the PCI bus. This then is the full address and data support for both bus master and bus slave.

The schematics shown on pages 13 through 15 of this document show the PCI connectors. With the exception of the interrupts and the bus request/grant signals being point to point, all the other signals are bused in common.

Each slot is given its own interrupt to the processor from the INTA of the PCI bus. Interrupts on the PCI bus are negative True and level sensitive so they fit nicely to the Am29030 processor interrupt input structure without a need for additional interrupt controllers. Each bus request and grant is given to the MACH220 device to arbitrate the bus. All the power pins of the PCI connector in this design are for 5 V; 3.3 V is not considered.

MACH220 DEVICE

The heart of the design is the MACH220 device, shown in the schematic on page 8. Internally this part can be subdivided into the following categories:

- Bus arbitration
- PCI control signals
- Memory control
- RAS and CAS decode
- Refresh timer

Each of these parts is detailed in the following sections.

Bus Arbitration

The bus arbiter in this design “parks” the Am29030 processor on the bus with a PROC_BGRT unless any of the PCI_BREQx’s become valid. When a PCI_BREQx becomes valid, then the PROC_BGRT is lowered and the arbiter waits for the REQ (Am29030 processor memory request) to be released for at least 2 clock cycles. At that point, the highest level PCI_BREQx is arbitrated and its corresponding PCI_BGRTx is issued. This grant is held until the PCI_BREQx goes away, the PCI_FRAME is released, and the last IRDY and TRDY have been issued. If another PCI_BREQx is active, then the arbiter will move to the highest new PCI_BREQx. If no other PCI_BREQx is active, then the PROC_BGRT is issued again and the whole cycle starts over. No attempt is made to make a rotating-priority arbiter, although that could be accomplished in the buried macros of the MACH220 device. This arbiter simply functions as a fixed-priority level arbiter.

PCI Control Signals

This section has the largest number of equations. Some of them are for support when the Am29030 processor is the bus master, while some of them are for the slave memory interface. Address bit A31 is the magic control bit that determines whether the Am29030 processor stays local to the memory subsystem or goes to the PCI bus. If A31=0, then the Am29030 processor will stay in local memory section; if A31=1, then an external PCI access is started.

When the Am29030 processor is a bus master and goes to the PCI bus to access a peripheral, A31=1 and PROC_BGRT is True. PROC_BGRT controls the three-state of the PCI_FRAME, PCI_IRDY, and PCI_C_BEx control lines. If A31=1 and PROC_BGRT is True, a PCI_FRAME signal is generated on the first line of the equations and is then held True if the Am29030 processor is bursting on the bus (BURST True).

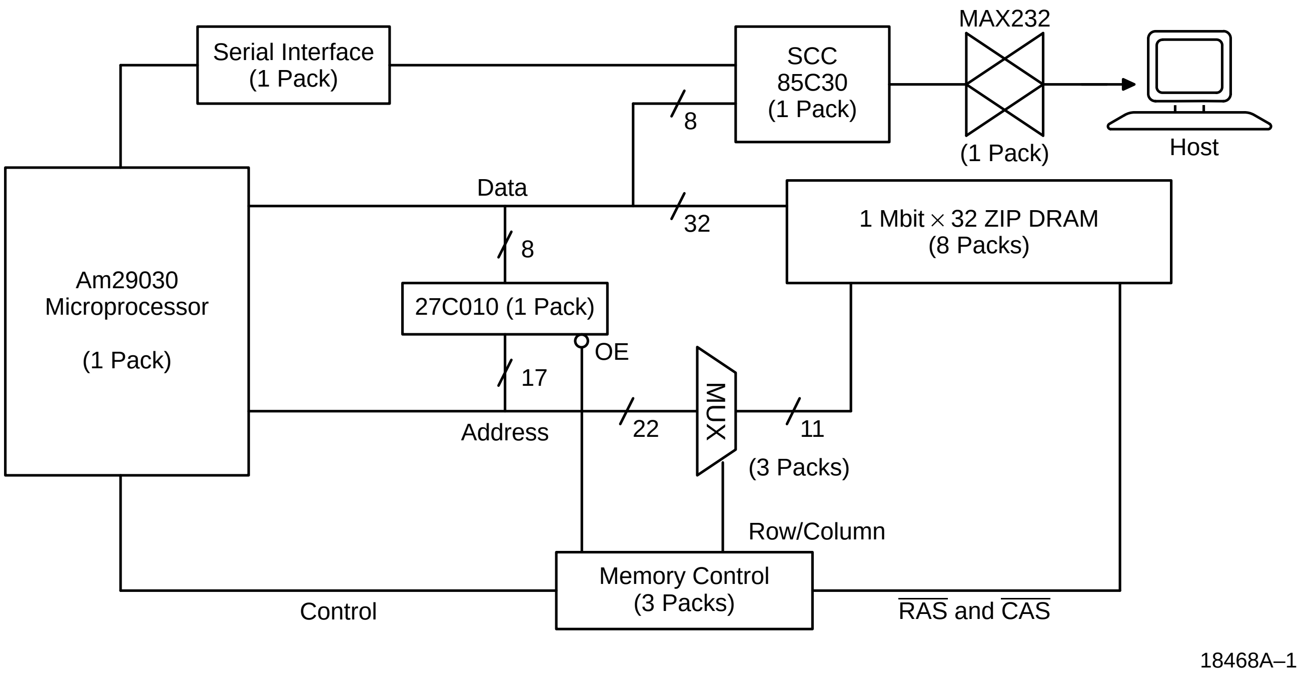

Serial Interface (1 Pack)
SCC 85C30 (1 Pack)
MAX232
(1 Pack)
Host
8
Data
32
1 Mbit × 32 ZIP DRAM (8 Packs)
Am29030 Microprocessor (1 Pack)
8
27C010 (1 Pack)
OE
17
Address
22
MUX
11
(3 Packs)
Row/Column
Memory Control (3 Packs)
Control
$\overline{\text{RAS}}$ and $\overline{\text{CAS}}$
18468A–1

Figure 1. EZ-030 Demonstration Board Block Diagram

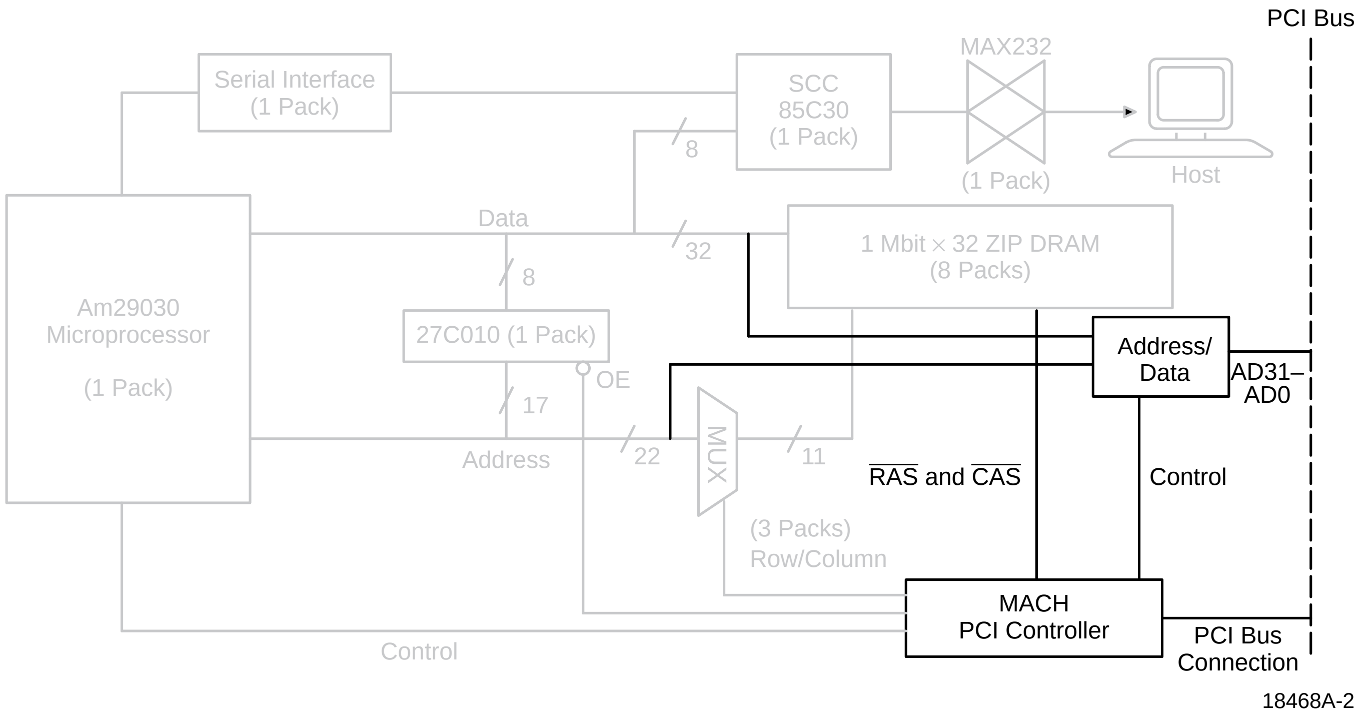


Figure 2. Modified EZ-030 Demonstration Board, With PCI I/O Bus

The Am29030 processor address is enabled onto the PCI bus with the term ADD_EN during the time that PCI_FRAME is True while PCI_FRAME_D (delayed by one clock PCI_FRAME) is False. This represents the first cycle on the PCI bus. During this cycle, the PCI_C_BEx lines output the appropriately decoded control information. The only cycles this design supports in master mode are:

- Memory reads and writes
  (A31=1, A30=0, PROC_IO_MEM=0)
- Configuration reads and writes
  (A31=1, A30=1, PROC_IO_MEM=0)
- I/O reads and writes
  (A31=1, A30=X, PROC_IO_MEM=1)

Configuration reads and writes also assert PCI_IDSELx with the decode performed on the A29 and A28 address bits. When PCI_FRAME_D becomes True, control goes to the data portion of the cycle (PCI_DATA_CYCLE is True) and the data bus is enabled with DATA_EN. PCI_FRAME_D stays True until the last PCI_TRDY is generated.

The data direction is handled by DATA_DIR. All PCI_C_BEx signals are driven True during the data cycle if the cycle is a read, or they are driven with the BEx off the Am29030 processor during a write to the PCI bus. On the PCI bus, a device indicates that it has been selected by asserting a signal PCI_DEVSEL. However, if no device responds, the Am29030 processor does not terminate the transaction because no PROC_RDY is asserted and there is a bus lock. To break this lock, a default timer is put into this design in the form of a 3-bit counter (CT0, CT1, CT2) that gives the peripherals 8 clock cycles to respond. If no PCI_DEVSEL is returned, then an Am29030 processor PROC_ERR is generated. Additionally, PROC_ERR is generated in normal cycles if the responding peripheral also asserts PCI_STOP. During the data cycle, PCI_IRDY is driven True at all times and PCI_TRDY is passed to the processor on PROC_RDY to signal the end of any given cycle on the PCI bus.

For a PCI master access, a similar transaction takes place in reverse if the incoming PCI address bus bit A31=1 and A30=0. Every address issued on the PCI bus is captured in the address register by the ADD_CLK line being active with PCI_FRAME set to True and PCI_FRAME_D set to False. The address register (ADD_REG_EN) is always enabled onto the Am29030 processor address bus if any of the PCI_BGRTx signals are True, which means the Am29030 processor does not control its address bus.

If the PCI cycle is for the local memory system, then the internal node DEV_SEL_NODE goes active, thereby deactivating the three-state of the control for PCI_STOP, PCI_DEVSEL, and PCI_TRDY, and enabling the data bus drivers through the DATA_EN term. Since the PCI bus gives the condition for read and write under the control portion of the PCI bus, and does not keep a write line active through the whole cycle, this fact is captured on an internal node INTERNAL_WRITE. This then controls the direction of the data buffers during a PCI master cycle. INTERNAL_WRITE also drives the WRITE line of the processor for the memory subsystem during this time. PCI_TRDY then becomes MEM_RDY as the state machine for the local memory goes through its paces. If bursting is being done, the ADD_INC signal tells the 22V10 the A9–A2 captured address to increment to the next address. If roll over is detected in the 22V10 by the ADD_STOP signal, then a PCI_STOP is asserted, which says a new bus address is needed.

Memory Control

Memory control becomes almost a direct graft of that described in the *EZ-030 Demonstration Board Theory of Operation* application note. The same state machines are implemented with the exception that accommodation is now made for more than one master to access memory. REF_ACCESS still has top priority to do its job to refresh the memory array. IDLE has more terms added to it to support the processor doing a local memory cycle and the PCI bus doing its master cycle. PCI bus is indicated by the PROC_BGRT set to False and the PCI_DATA_CYCLE set to True. MEM_ACCESS is started in the same manner as before and held as long as burst is True for the Am29030 processor.

PCI, though, indicates that it is on its last cycle by deasserting PCI_FRAME, which has a similar control to the Am29030 processor BURST signal. However, for a complete cycle to take place, both TRDY and IRDY on the PCI bus must be True so this is taken into account in the equations for MEM_ACCESS, MEM_RDY, and ADD_INC. If at any time, the PCI bus suspends the memory access, then reads and writes occur to the same address and are simply repeated because this is a single-cycle memory unit.

RAS and CAS Decode

The RAS and CAS decode portion of the equations are essentially the same as in the EZ-030 board application note, with the exception that a new equation was added to account for the source of the byte enable control. When a local cycle is performed from the Am29030 processor, the byte enables come from the processor on the BEx pins. During a PCI cycle, they are issued on the PCI_C_BEx pins on the PCI bus interface. Internal write is used as the read/write indication (as was mentioned before) because the PCI bus transfers the state of the transfer on the first cycle and then does not retain a write line for the remainder of the transaction.

Refresh Timer

The refresh timer is exactly as described in the *EZ-030 Demonstration Board Theory of Operation* application note, consisting of a 7-bit timer with the eighth bit serving as a REFRESH_REQ bit, which remembers that a refresh needs to be arbitrated by the main memory system arbiter. The counter will continue counting towards the next refresh interval.

Serial Port

If used, the serial port can be exactly the same as described in the *EZ-030 Demonstration Board Theory of Operation* application note.

TIMING AND WORST-CASE ISSUES

This design assumes that the PGA Am29030 processor's Scalable Clocking™ feature is used, resulting in a 32-MHz processor with a 16-MHz external memory and PCI bus interface. The worst-case timing in this design then occurs in the memory system around a single-cycle read at 16 MHz. The delay must be set at 20 ns for the CAS pulse generation while 12 ns is the best time of a MACH220 device for the combinatorial delay, making CAS fall at 32 ns into the cycle. CAS access time is 20 ns. A set up of 9 ns makes an access time of 61 ns for a 62.5-ns cycle at 16 MHz. If additional setup or CAS access time is needed, then the RAS/CAS decode can be accomplished in a separate, faster decode PAL device. The DELAY signal from the delay line can be used in this PAL device, with the other state lines going to the faster decode PAL device. Additionally, faster address multiplexers can be used, such as 74F or AS parts, instead of the 74LS157s. This reduces the delay from 12 ns to 6 ns and decreases the delay line requirement to 15 ns, gaining an additional 5 ns on CAS access. Use of these faster 74F/AS157s then requires 33-ohm series dampening resistors before the DRAMs.

The PCI bus from the initiator side meets the setup times up to 25 MHz, but the CLK to Q delays on the beginning of edges are not to the delay specification. This is really not of consequence to this design because the bus is running slower than the maximum of 33 MHz. This slower speed allows a more relaxed timing as well as board layout in the final analysis. The rest of the paths, including to and from the processor and the surrounding control logic, have wide margins of setup and hold that can be used.

PCI BUS SIGNAL QUALITY

The PCI bus is defined to be a current-driven reflected-wave transmission line. However, all the drivers (74F24x and MACH device) used in this design to drive the PCI bus signals are voltage drivers, and therefore, incident-wave drivers. Reflected-wave drivers have to settle the bus and therefore have two times the transmission line distance to settle, but incident-wave drivers have only one times the transmission line time distance. This makes up for some of the clock-to-Q time of the 74F24x parts. The input threshold levels and the final output drive levels on the PCI bus are TTL-compatible, which means the 74F24x and MACH device drivers are compatible. When laying out the design, the best layout is the MACH device, buffers, and PAL devices on one end of the transmission line, and the sockets for the PCI bus in line towards the opposite end.

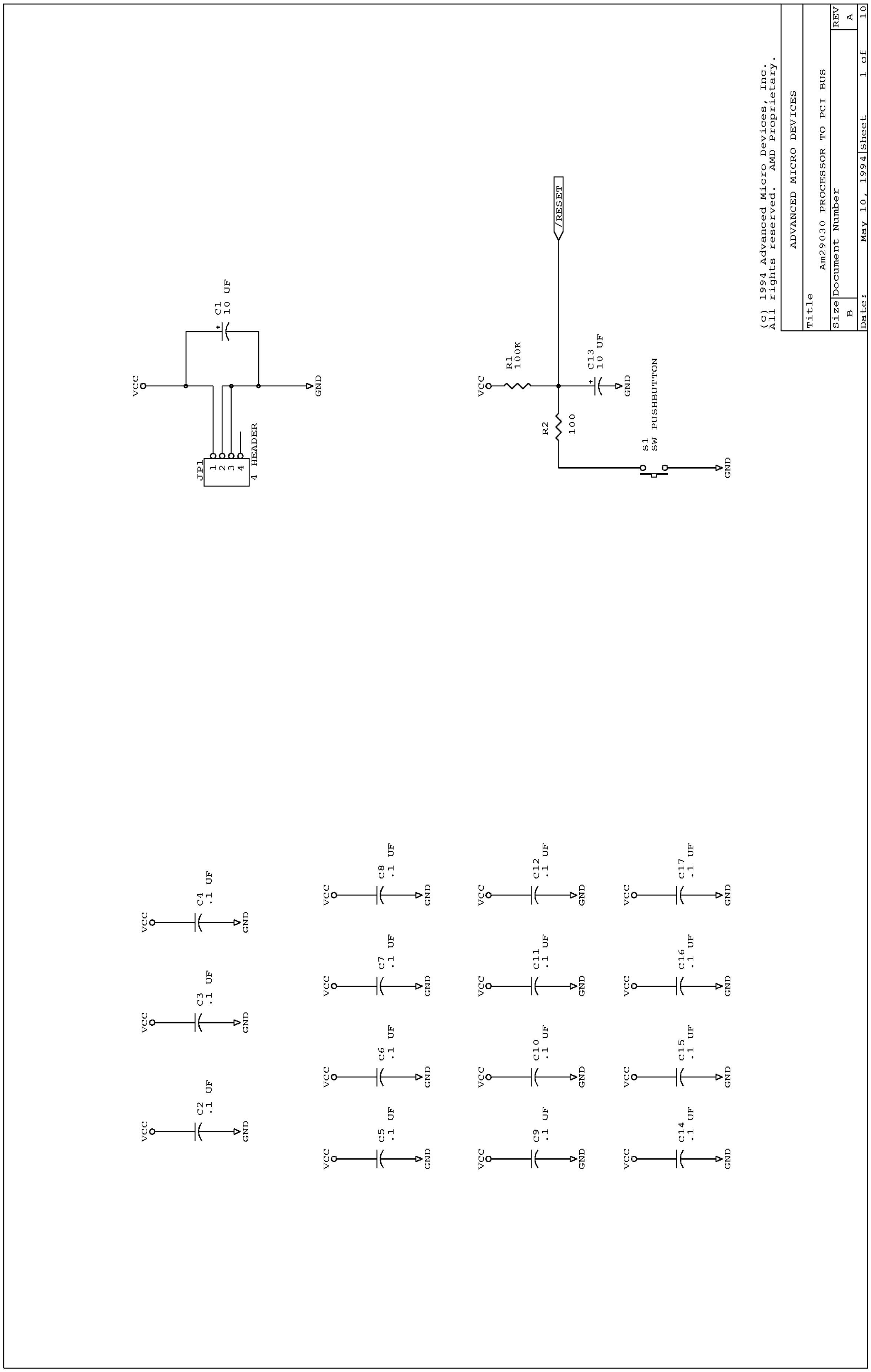
The PCI bus allows 10 loads of 10 pF per load maximum at 33 MHz. This design uses six loads for the three slots. The three-way load of the 74F24x and PAL devices then represent about 35 pF, making up the other four loads. At 16 MHz, though, this requirement could be easily relaxed if needed.

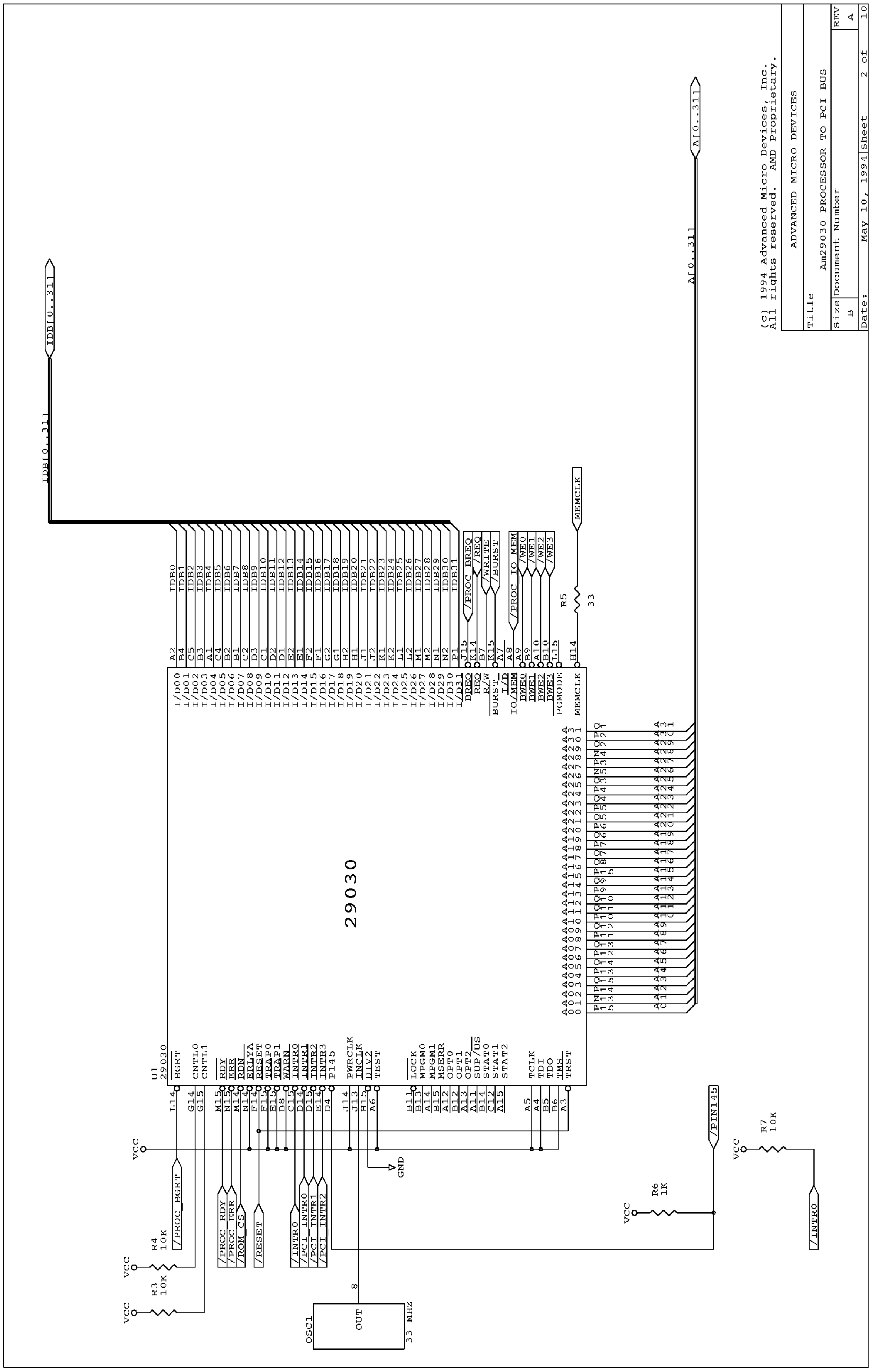
SUGGESTED REFERENCE

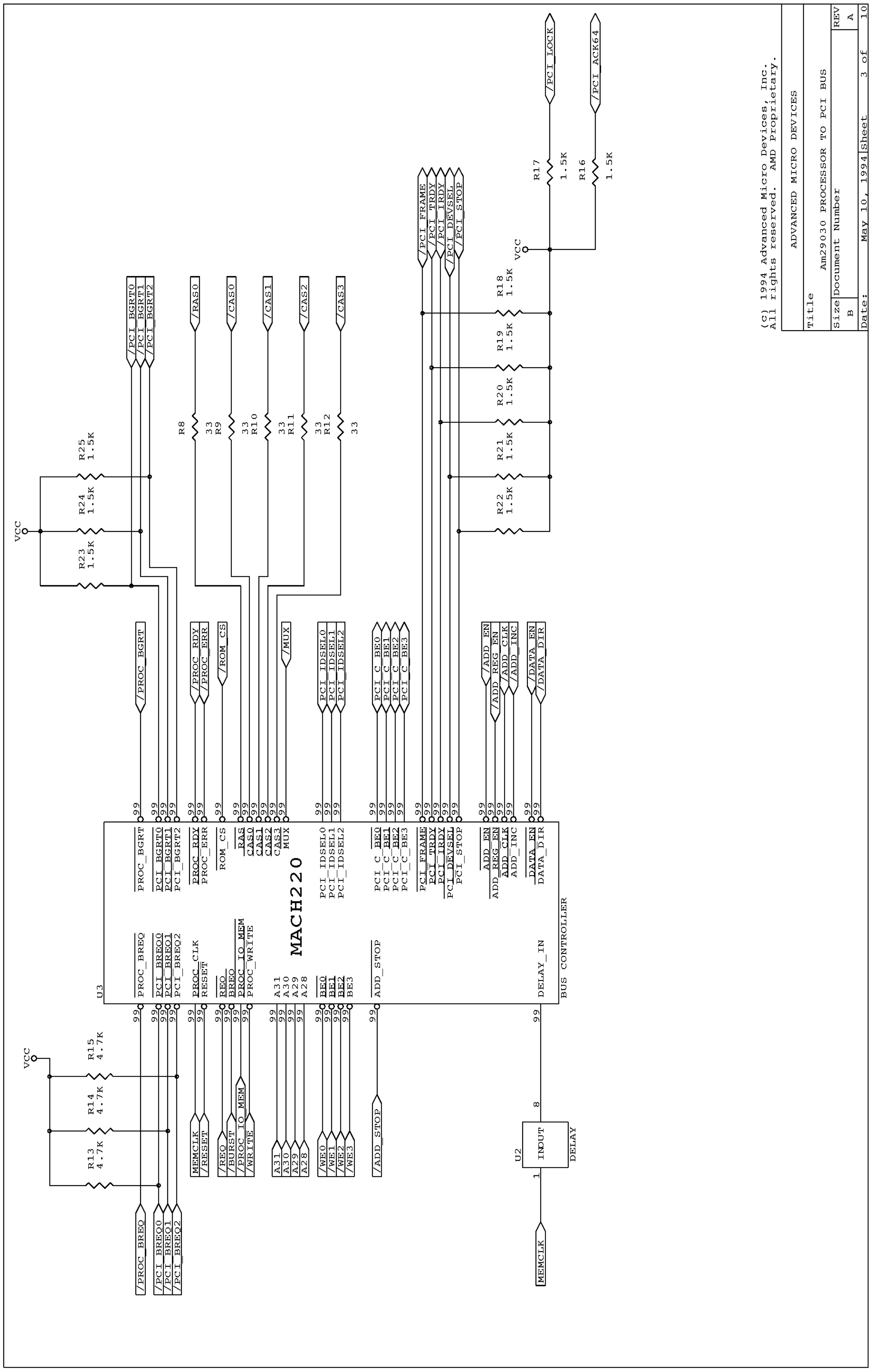
- *Bank Interleaved Memory System for an Am29030 Microprocessor* application note, order# 18478, Advanced Micro Devices
- *EZ-030 Demonstration Board Theory of Operation* application note, order# 17580, Advanced Micro Devices
- *PCI Local Bus Specification*, Revision 2.0
  PCI Special Interest Group
  M/S HF3–15A
  5200 N. E. Elam Young Parkway
  Hillsboro, Oregon 97124–6497
  (503) 696–2000

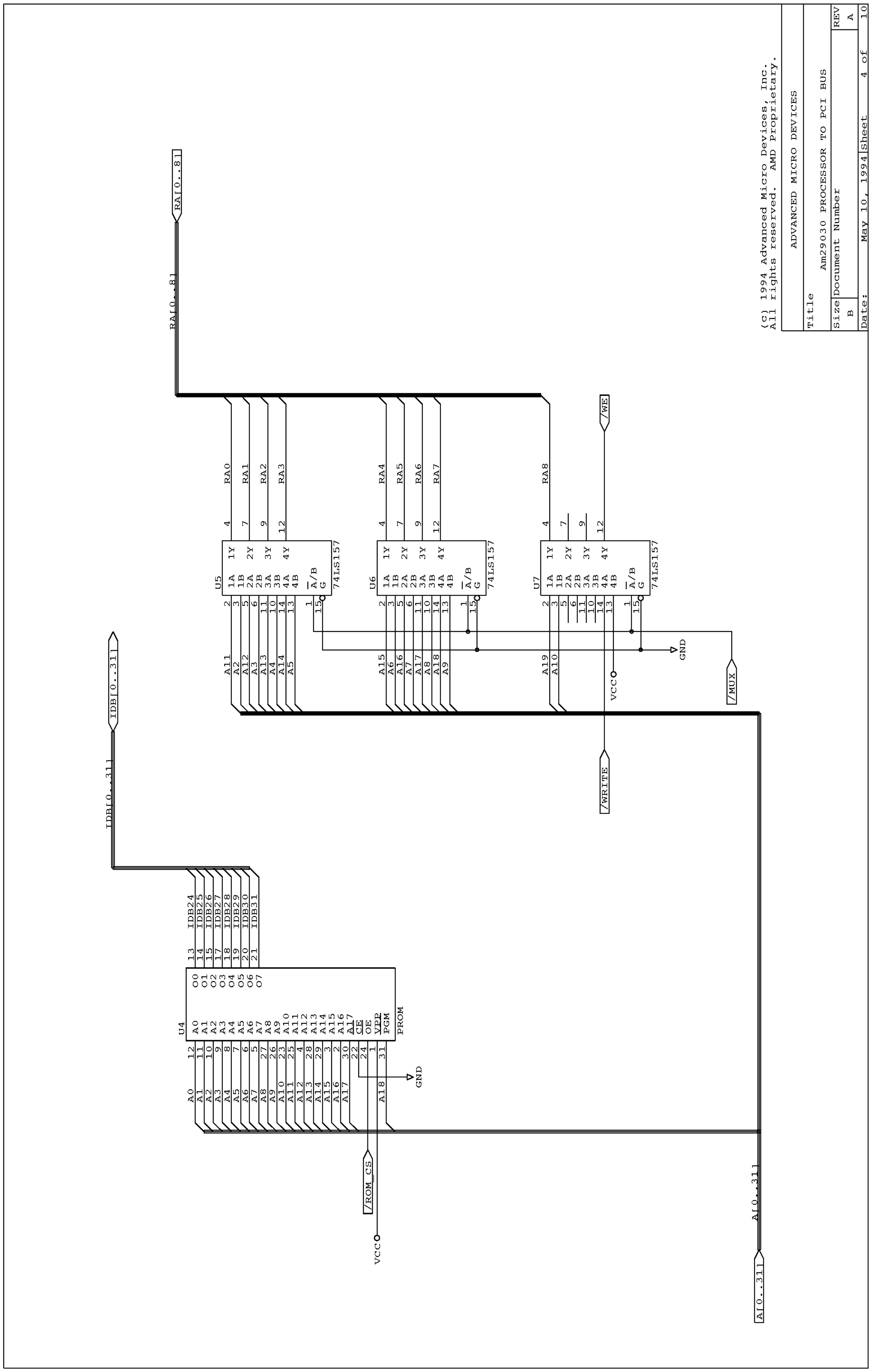
Appendix A. Schematics

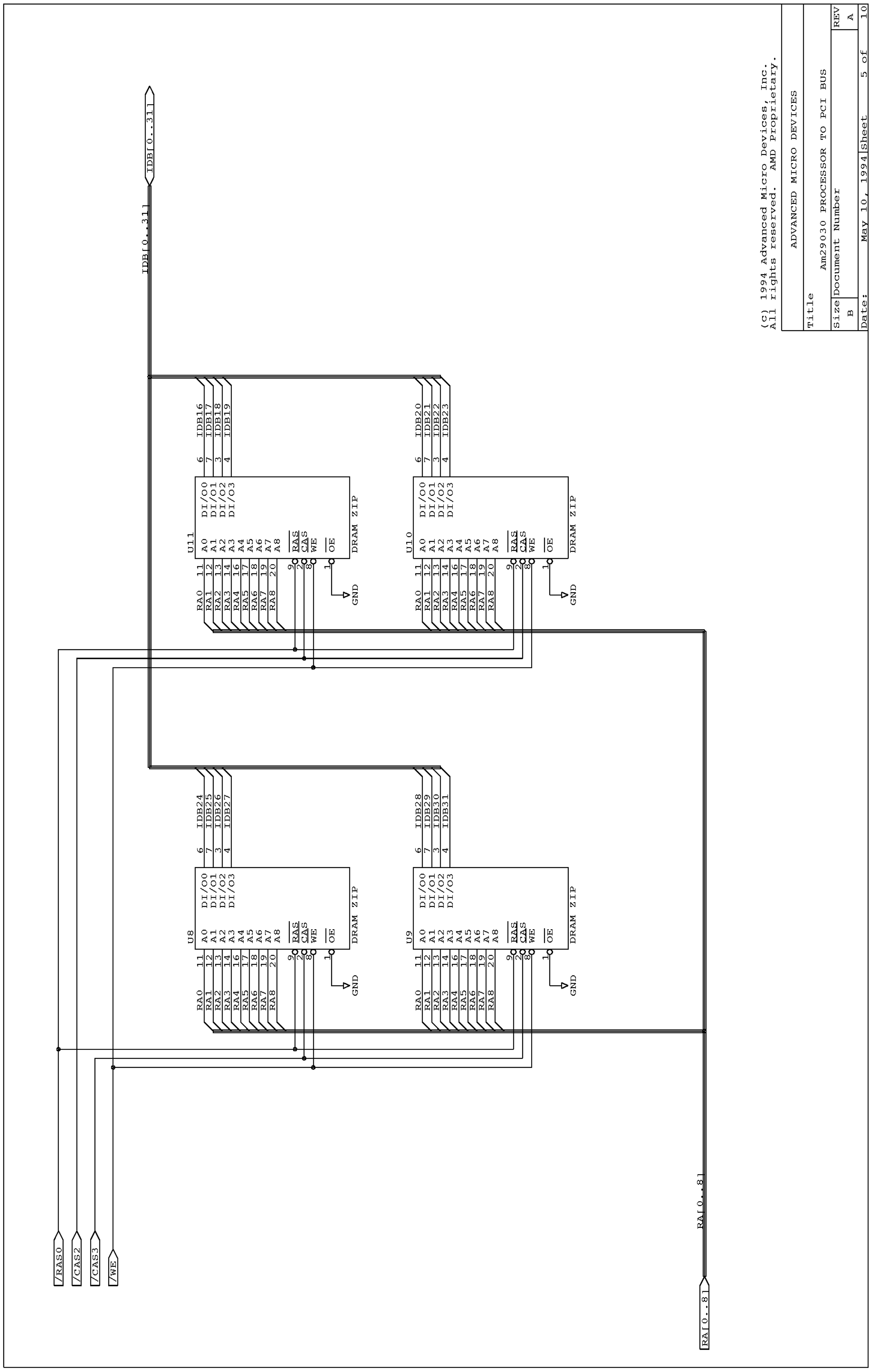
The schematics for this design are shown on the pages that follow.

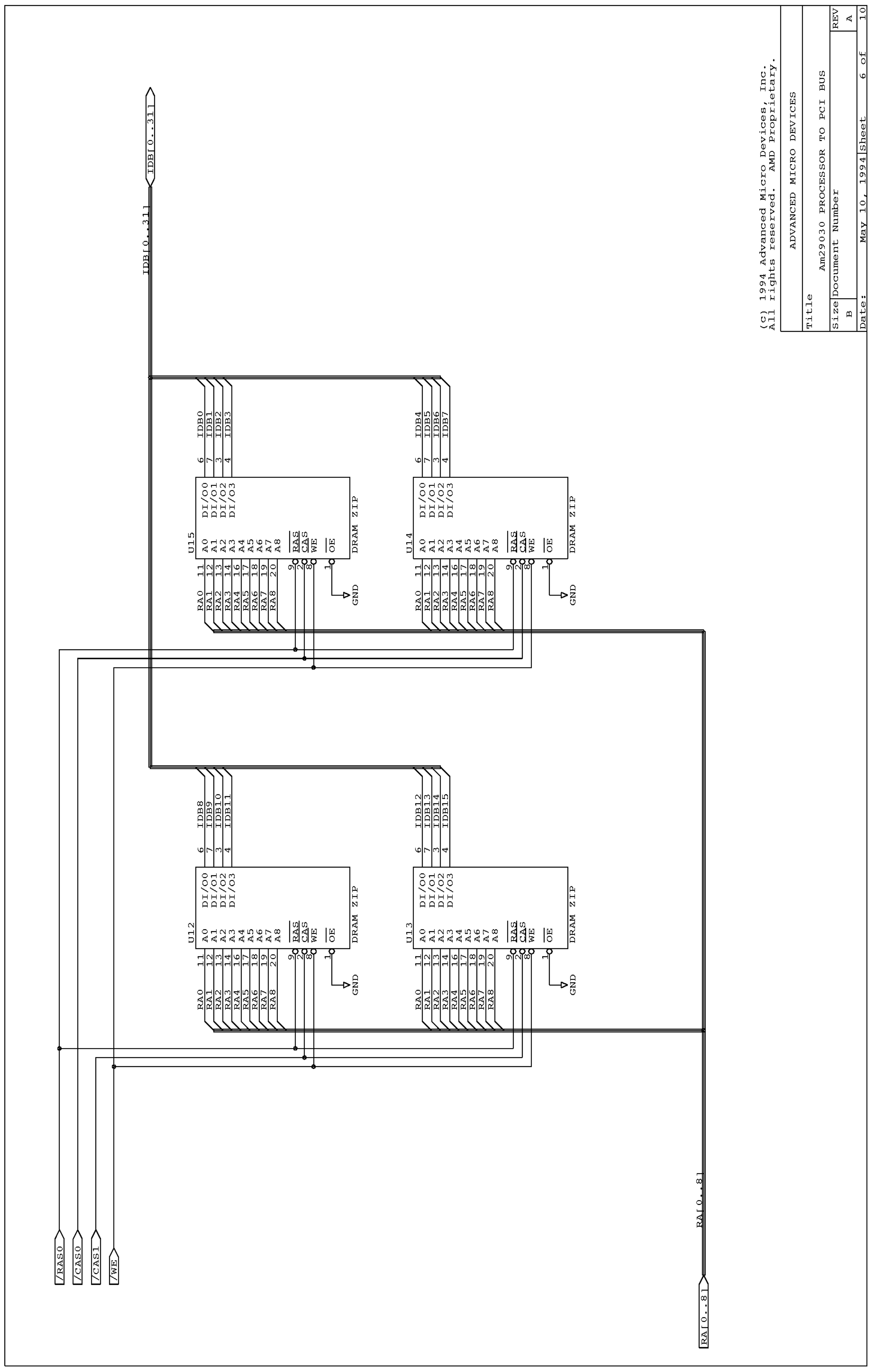

(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR TO PCI BUS
Size
B
Document Number
REV
A
Date: May 10, 1994
Sheet 1 of 10
VCC
GND
C1
10 UF
JP1
1 2 3 4
4 HEADER
R1
100K
R2
100
C13
10 UF
/RESET
S1
SW PUSHBUTTON
C2 .1 UF
C3 .1 UF
C4 .1 UF
C5 .1 UF
C6 .1 UF
C7 .1 UF
C8 .1 UF
C9 .1 UF
C10 .1 UF
C11 .1 UF
C12 .1 UF
C14 .1 UF
C15 .1 UF
C16 .1 UF
C17 .1 UF

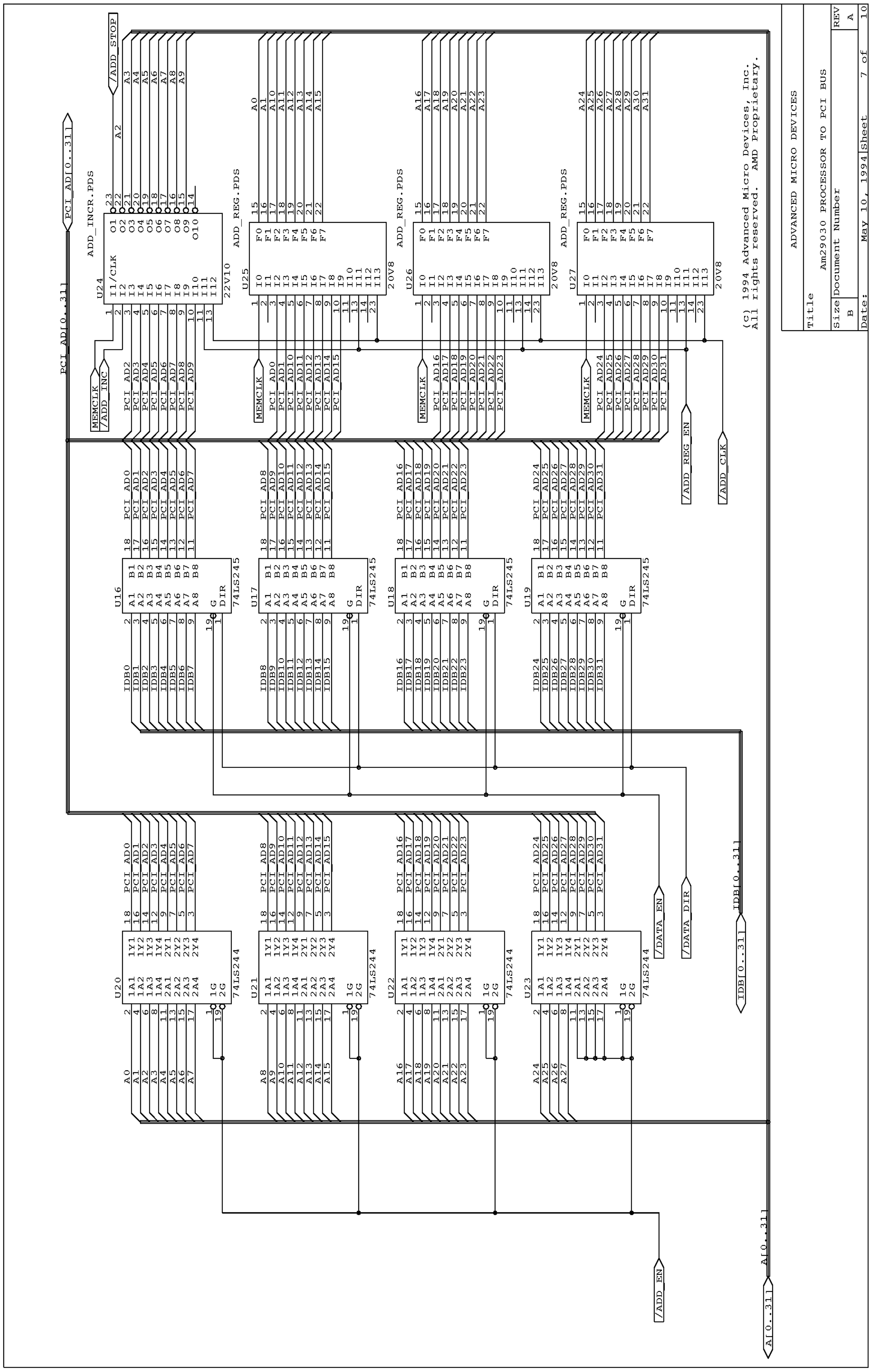

(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR TO PCI BUS
Size B
Document Number
REV A
Date: May 10, 1994
Sheet 2 of 10
29030
U1
IDB[0..31]
A[0..31]
MEMCLK
R5 33
R3 10K
R4 10K
R6 1K
R7 10K
VCC
GND
OSC1
OUT
33 MHZ
/PROC_BGRT
/PROC_RDY
/PROC_ERR
/ROM_CS
/RESET
/INTR0
/PCI_INTR0
/PCI_INTR1
/PCI_INTR2
/PIN145
/PROC_BREQ
/REQ
/WRITE
/BURST
/PROC_IO_MEM
/WE0
/WE1
/WE2
/WE3

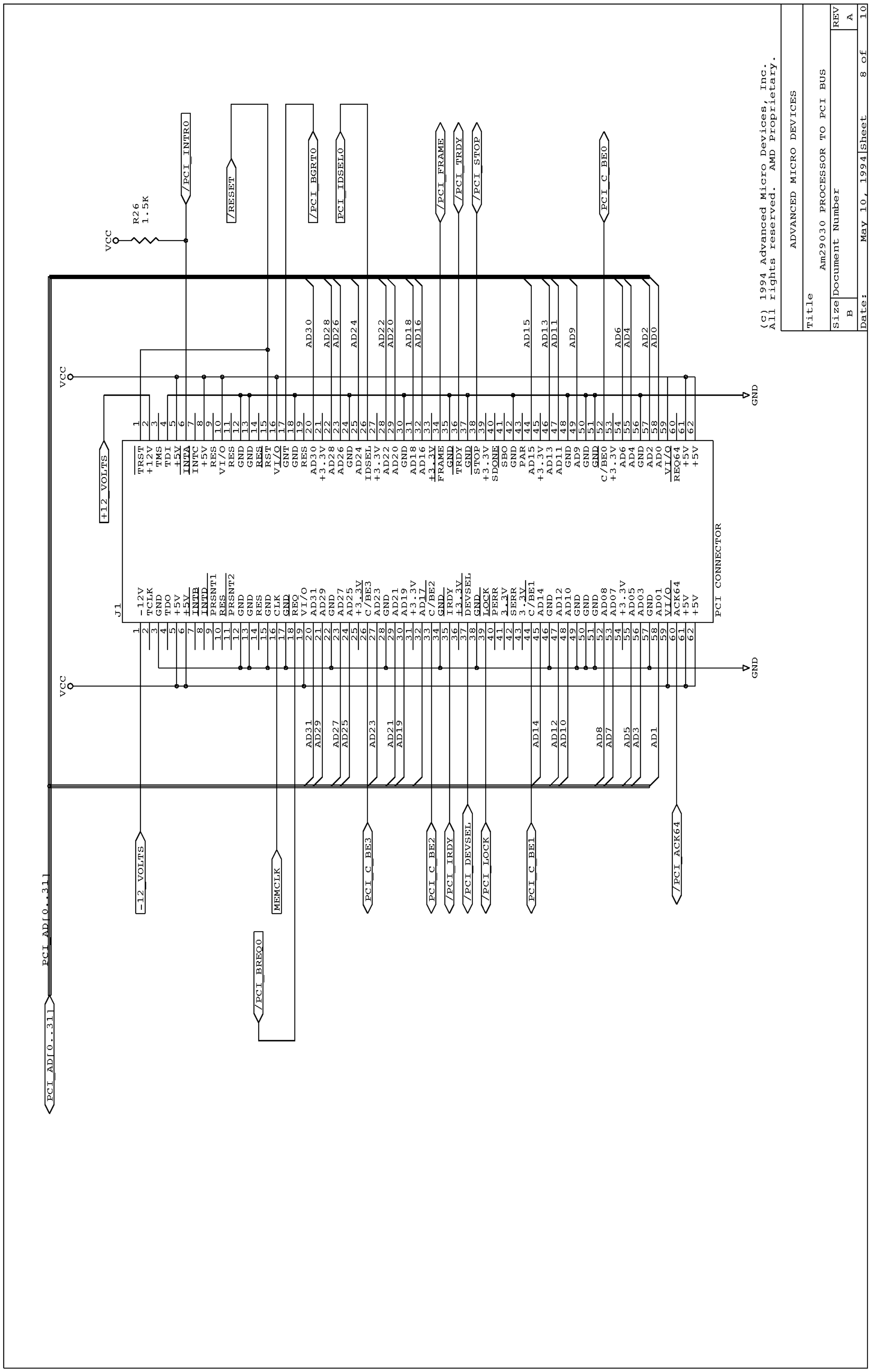

MACH220
U3
BUS CONTROLLER
PROC_BREQ
PCI_BREQ0
PCI_BREQ1
PCI_BREQ2
PROC_CLK
RESET
REQ
BREQ
PROC_IO_MEM
PROC_WRITE
A31
A30
A29
A28
BE0
BE1
BE2
BE3
ADD_STOP
DELAY_IN
PROC_BGRT
PCI_BGRT0
PCI_BGRT1
PCI_BGRT2
PROC_RDY
PROC_ERR
ROM_CS
RAS
CAS0
CAS1
CAS2
CAS3
MUX
PCI_IDSEL0
PCI_IDSEL1
PCI_IDSEL2
PCI_C_BE0
PCI_C_BE1
PCI_C_BE2
PCI_C_BE3
PCI_FRAME
PCI_TRDY
PCI_IRDY
PCI_DEVSEL
PCI_STOP
ADD_EN
ADD_REG_EN
ADD_CLK
ADD_INC
DATA_EN
DATA_DIR
/PROC_BREQ
/PCI_BREQ0
/PCI_BREQ1
/PCI_BREQ2
MEMCLK
/RESET
/REQ
/BURST
/PROC_IO_MEM
/WRITE
/WE0
/WE1
/WE2
/WE3
/ADD_STOP
/PROC_BGRT
/PROC_RDY
/PROC_ERR
/ROM_CS
/MUX
/PCI_BGRT0
/PCI_BGRT1
/PCI_BGRT2
/RAS0
/CAS0
/CAS1
/CAS2
/CAS3
/PCI_FRAME
/PCI_TRDY
/PCI_IRDY
/PCI_DEVSEL
/PCI_STOP
/PCI_LOCK
/PCI_ACK64
/ADD_EN
/ADD_REG_EN
/ADD_CLK
/ADD_INC
/DATA_EN
/DATA_DIR
VCC
R13 4.7K
R14 4.7K
R15 4.7K
R23 1.5K
R24 1.5K
R25 1.5K
R8 33
R9 33
R10 33
R11 33
R12 33
R16 1.5K
R17 1.5K
R18 1.5K
R19 1.5K
R20 1.5K
R21 1.5K
R22 1.5K
U2
IN OUT
DELAY
1
8
99
(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR TO PCI BUS
Size B
Document Number
REV A
Date: May 10, 1994
Sheet 3 of 10

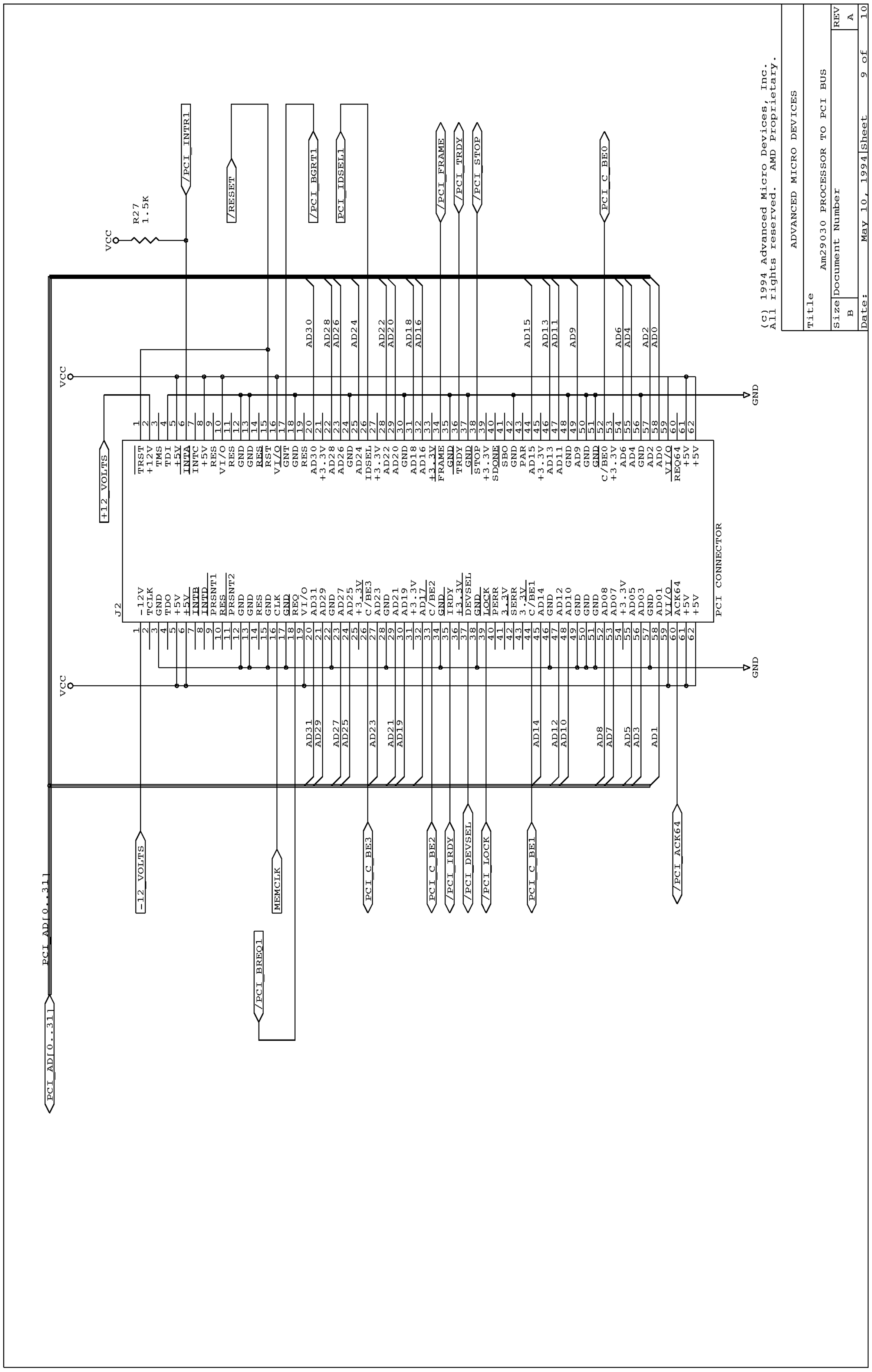

(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR TO PCI BUS
Size B
Document Number
REV A
Date: May 10, 1994
Sheet 4 of 10
RA[0..8]
RA0
RA1
RA2
RA3
RA4
RA5
RA6
RA7
RA8
/WE
U5
U6
U7
74LS157
1A 1B 2A 2B 3A 3B 4A 4B
1Y 2Y 3Y 4Y
A/B
G
A11 A2 A12 A3 A13 A4 A14 A5
A15 A6 A16 A7 A17 A8 A18 A9
A19 A10
VCC
GND
/MUX
/WRITE
IDB[0..31]
IDB24 IDB25 IDB26 IDB27 IDB28 IDB29 IDB30 IDB31
U4
PROM
O0 O1 O2 O3 O4 O5 O6 O7
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A16 A17
CE
OE
VPP
PGM
A18
/ROM_CS
A[0..31]

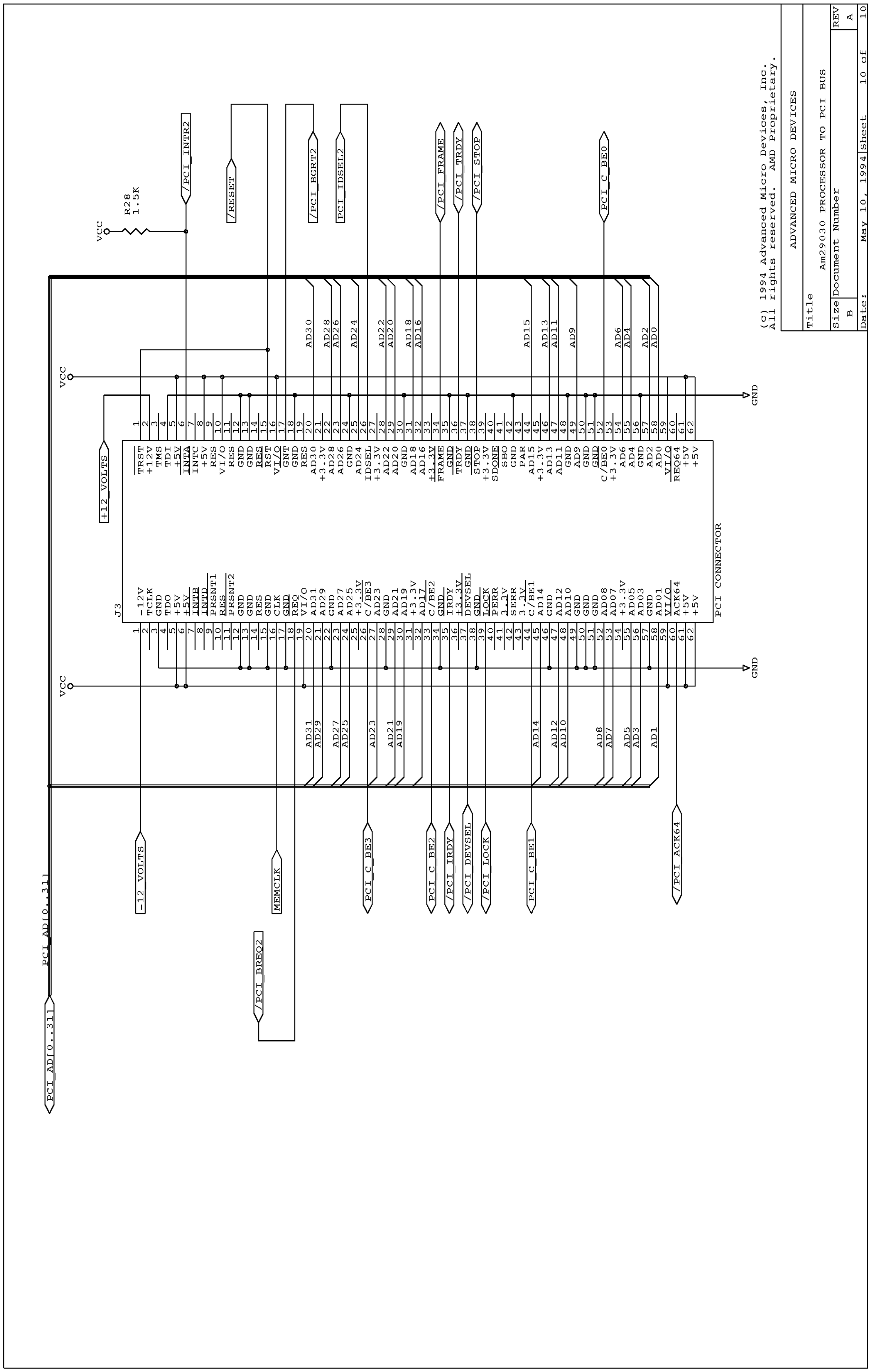

IDB[0..31]
IDB16 IDB17 IDB18 IDB19
IDB20 IDB21 IDB22 IDB23
IDB24 IDB25 IDB26 IDB27
IDB28 IDB29 IDB30 IDB31
U11
U10
U8
U9
DI/O0 DI/O1 DI/O2 DI/O3
A0 A1 A2 A3 A4 A5 A6 A7 A8
RAS CAS WE OE
DRAM ZIP
RA0 RA1 RA2 RA3 RA4 RA5 RA6 RA7 RA8
GND
/RAS0
/CAS2
/CAS3
/WE
RA[0..8]
(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR TO PCI BUS
Size
B
Document Number
REV
A
Date: May 10, 1994
Sheet 5 of 10


(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR TO PCI BUS
Size B
Document Number
REV A
Date: May 10, 1994
Sheet 6 of 10
IDB[0..31]
RA[0..8]
/RAS0
/CAS0
/CAS1
/WE
U15
U14
U12
U13
DRAM ZIP
A0
A1
A2
A3
A4
A5
A6
A7
A8
DI/O0
DI/O1
DI/O2
DI/O3
RAS
CAS
WE
OE
GND
RA0
RA1
RA2
RA3
RA4
RA5
RA6
RA7
RA8
IDB0
IDB1
IDB2
IDB3
IDB4
IDB5
IDB6
IDB7
IDB8
IDB9
IDB10
IDB11
IDB12
IDB13
IDB14
IDB15


ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR TO PCI BUS
Size B
Document Number
REV A
Date: May 10, 1994
Sheet 7 of 10
(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
U24 ADD_INCR.PDS 22V10
U25 ADD_REG.PDS 20V8
U26 ADD_REG.PDS 20V8
U27 ADD_REG.PDS 20V8
U16 74LS245
U17 74LS245
U18 74LS245
U19 74LS245
U20 74LS244
U21 74LS244
U22 74LS244
U23 74LS244
PCI_AD[0..31]
/ADD_STOP
MEMCLK
/ADD_INC
/ADD_REG_EN
/ADD_CLK
/DATA_EN
/DATA_DIR
IDB[0..31]
/ADD_EN
A[0..31]


(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR TO PCI BUS
Size
B
Document Number
REV
A
Date: May 10, 1994
Sheet 8 of 10
J1
PCI CONNECTOR
R26
1.5K
VCC
GND
+12_VOLTS
-12_VOLTS
/PCI_INTR0
/RESET
/PCI_BGRT0
PCI_IDSEL0
/PCI_FRAME
/PCI_TRDY
/PCI_STOP
PCI_C_BE0
PCI_C_BE3
PCI_C_BE2
/PCI_IRDY
/PCI_DEVSEL
/PCI_LOCK
PCI_C_BE1
/PCI_ACK64
MEMCLK
/PCI_BREQ0
PCI_AD[0..31]
TRST
+12V
TMS
TDI
+5V
INTA
INTC
RES
VI/O
GND
RST
GNT
AD30
+3.3V
AD28
AD26
AD24
IDSEL
AD22
AD20
AD18
AD16
FRAME
TRDY
STOP
SDONE
SBO
PAR
AD15
AD13
AD11
AD9
C/BE0
AD6
AD4
AD2
AD0
REQ64
-12V
TCLK
TDO
INTB
INTD
PRSNT1
PRSNT2
CLK
REQ
AD31
AD29
AD27
AD25
C/BE3
AD23
AD21
AD19
AD17
C/BE2
IRDY
DEVSEL
LOCK
PERR
SERR
C/BE1
AD14
AD12
AD10
AD08
AD07
AD05
AD03
AD01
ACK64
AD8
AD7
AD5
AD3
AD1


(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR TO PCI BUS
Size
B
Document Number
REV
A
Date: May 10, 1994
Sheet 9 of 10
PCI CONNECTOR
J2
R27
1.5K
VCC
GND
/PCI_INTR1
/RESET
/PCI_BGRT1
PCI_IDSEL1
/PCI_FRAME
/PCI_TRDY
/PCI_STOP
PCI_C_BE0
+12_VOLTS
-12_VOLTS
MEMCLK
PCI_C_BE3
PCI_C_BE2
/PCI_IRDY
/PCI_DEVSEL
/PCI_LOCK
PCI_C_BE1
/PCI_ACK64
/PCI_BREQ1
PCI_AD[0..31]


(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
Title
ADVANCED MICRO DEVICES
Am29030 PROCESSOR TO PCI BUS
Size B
Document Number
REV A
Date: May 10, 1994
Sheet 10 of 10
J3
PCI CONNECTOR
R28
1.5K
VCC
GND
+12_VOLTS
-12_VOLTS
/PCI_INTR2
/RESET
/PCI_BGRT2
PCI_IDSEL2
/PCI_FRAME
/PCI_TRDY
/PCI_STOP
PCI_C_BE0
PCI_C_BE1
PCI_C_BE2
PCI_C_BE3
/PCI_IRDY
/PCI_DEVSEL
/PCI_LOCK
/PCI_ACK64
/PCI_BREQ2
MEMCLK
PCI_AD[0..31]

Appendix B. PAL Equations

This appendix shows the PAL equations in PALASM® software syntax for: the 030_PCI equations, the ADD_INCR equations, and the ADD_REG equations.

030_PCI PAL EQUATIONS

```
;PALASM Software Design Description

;---------------------------------- Declaration Segment ------------
TITLE    Am29030 PROCESSOR TO PCI CONVERTOR
PATTERN  030_PCI.PDS
REVISION A
AUTHOR   DAVID STOENNER
COMPANY  AMD
DATE     02/21/93

CHIP  Ux  MACH220 DEVICE

;---------------------------------- PIN Declarations ---------------
PIN  ?          MEMCLK
PIN  ?          /RESET
PIN  ?          /REQ
PIN  ?          A31
PIN  ?          A30
PIN  ?          A29
PIN  ?          A28
PIN  ?          /BURST
PIN  ?          /PROC_RDY
PIN  ?          /MEM_ACCESS                          REGISTERED
PIN  ?          /REF_ACCESS                          REGISTERED
PIN  ?          /ROM_CS
PIN  ?          /IDLE                                REGISTERED
PIN  ?          /MEM_RDY
PIN  ?          DELAY_IN
PIN  ?          /WE0
PIN  ?          /WE1
PIN  ?          /WE2
PIN  ?          /WE3
PIN  ?          /WRITE
PIN  ?          /CAS0
PIN  ?          /CAS1
PIN  ?          /CAS2
PIN  ?          /CAS3
PIN  ?          /MUX
PIN  ?          /RAS0
PIN  ?          /PROC_IO_MEM
PIN  ?          /PROC_ERR
PIN  ?          PCI_C_BE0
PIN  ?          PCI_C_BE1
PIN  ?          PCI_C_BE2
PIN  ?          PCI_C_BE3
PIN  ?          /PCI_FRAME
PIN  ?          /PCI_IRDY
PIN  ?          /PCI_TRDY
PIN  ?          /PCI_DEVSEL
PIN  ?          /PCI_DATA_CYCLE
PIN  ?          /DATA_EN
PIN  ?          /DATA_DIR
PIN  ?          /ADD_REG_EN
PIN  ?          /ADD_EN
PIN  ?          /ADD_CLK
PIN  ?          /ADD_INC
PIN  ?          /ADD_STOP
PIN  ?          /PROC_BREQ
PIN  ?          /PROC_BGRT
PIN  ?          /PCI_BREQ0
PIN  ?          /PCI_BREQ1
PIN  ?          /PCI_BREQ2
PIN  ?          /PCI_BGRT0
PIN  ?          /PCI_BGRT1
PIN  ?          /PCI_BGRT2
PIN  ?          PCI_IDSEL0
PIN  ?          PCI_IDSEL1
PIN  ?          PCI_IDSEL2
PIN  ?          /PCI_STOP

; ALL THE NODE DEFINITIONS

NODE ?          CT0
NODE ?          CT1
NODE ?          CT2
node ?          ST1
NODE ?          DEV_SEL_NODE
NODE ?          PCI_FRAME_D
NODE ?          REF_REQ                             REGISTERED
NODE ?          Q0                                  REGISTERED
NODE ?          Q1                                  REGISTERED
NODE ?          Q2                                  REGISTERED
NODE ?          Q3                                  REGISTERED
NODE ?          Q4                                  REGISTERED
NODE ?          Q5                                  REGISTERED
NODE ?          Q6                                  REGISTERED
NODE ?          INTERNAL_WRITE
NODE ?          RESETD
NODE ?          REQ_D

;------------------------------------ Boolean Equation Segment ------
EQUATIONS

; BUS ARBITER AND BUS GRANT STATE MACHINES

RESETD.CLKF = MEMCLK

RESETD := RESET

; REQ_D HAS BEEN ADDED TO FIX AN ARBITRATION PROBLEM OF THE Am29030 PROCESSOR REV
; B,C AND D. TO BE GUARANTEED THAT THE Am29030 PROCESSOR HAS GIVEN UP THE BUS WHEN
; BGRT HAS BEEN REMOVED, REQ MUST BE SAMPLED FALSE FOR 2 CLOCK CYCLES, HENCE THE
; ADDITION OF THE REQ_D TERM AND THE INCLUSION IN THE PCI_BGRTx TERMS. WHEN THIS
; PROBLEM IS FIXED, THE REQ_D MAY BE REMOVED FROM ALL EQUATIONS.

REQ_D.CLKF = MEMCLK

REQ_D := REQ

PROC_BGRT.CLKF = MEMCLK
```

```
PROC_BGRT := /PROC_BGRT * /PCI_BREQ0 * /PCI_BREQ1 * /PCI_BREQ2 * /PCI_FRAME
                * /PCI_IRDY * /RESETD
             + PROC_BGRT * /PCI_BREQ0 * /PCI_BREQ1 * /PCI_BREQ2 * /RESETD

PCI_BGRT0.CLKF = MEMCLK

PCI_BGRT0.TRST = /RESET

PCI_BGRT0 :=   /PCI_BGRT0 * PCI_BREQ0 * /PROC_BGRT * /REQ * /REQ_D
                   * /PCI_BGRT1 * /PCI_BGRT2 * /RESETD
             + PCI_BREQ0 * PCI_BGRT0 * /RESETD
             + PCI_BGRT0 * PCI_FRAME * PCI_FRAME_D * /RESETD

PCI_BGRT1.CLKF = MEMCLK

PCI_BGRT1.TRST = /RESET

PCI_BGRT1 :=   /PCI_BGRT1 * PCI_BREQ1 * /PCI_BREQ0 * /PROC_BGRT * /REQ
                    * /REQ_D * /PCI_BGRT0 * /PCI_BGRT2 * /RESETD
             + PCI_BREQ1 * PCI_BGRT1 * /RESETD
             + PCI_BGRT1 * PCI_FRAME * PCI_FRAME_D * /RESETD

PCI_BGRT2.CLKF = MEMCLK

PCI_BGRT2.TRST = /RESET

PCI_BGRT2 :=   /PCI_BGRT2 * PCI_BREQ2 * /PCI_BREQ1 * /PCI_BREQ0 * /PROC_BGRT
                  * /REQ_D * /REQ * /PCI_BGRT0 * /PCI_BGRT1 * /RESETD
             + PCI_BREQ2 * PCI_BGRT2 * /RESETD
             + PCI_BGRT2 * PCI_FRAME * PCI_FRAME_D * /RESETD

; PCI BUS CONTROL LOGIC

PCI_FRAME.TRST = PROC_BGRT * /RESET

PCI_FRAME =   PROC_BGRT * REQ * A31 * /PCI_FRAME_D
            + PROC_BGRT * A31 * BURST * PCI_FRAME_D

PCI_FRAME_D.CLKF = MEMCLK

PCI_FRAME_D := PCI_FRAME

ADD_CLK = /PROC_BGRT * PCI_FRAME * /PCI_FRAME_D

ADD_EN = PROC_BGRT * PCI_FRAME * /PCI_FRAME_D * /RESET

ADD_REG_EN = PCI_BGRT0 + PCI_BGRT1 + PCI_BGRT2

ADD_INC = /PROC_BGRT * PCI_FRAME * PCI_FRAME_D * PCI_IRDY * PCI_TRDY

CT0.CLKF = MEMCLK

CT0 := PCI_FRAME * /PCI_DEVSEL * /CT0

CT1.CLKF = MEMCLK

CT1 := PCI_FRAME * /PCI_DEVSEL * ( CT0 :+: CT1 )
```

```
CT2.CLKF = MEMCLK

CT2 := PCI_FRAME * /PCI_DEVSEL * ( CT2 :+: ( CT1*CT0 ))

PROC_ERR =   PROC_BGRT * CT2 * CT1 * CT0 * /PCI_DEVSEL
           + PROC_BGRT * PCI_DEVSEL * PCI_STOP
           + PROC_BGRT * PCI_FRAME *  /PCI_DEVSEL * DEV_SEL_NODE

PCI_STOP.TRST = /PROC_BGRT * DEV_SEL_NODE * /RESET

PCI_STOP =    /PCI_STOP * DEV_SEL_NODE * ADD_STOP
            + PCI_STOP * PCI_FRAME * DEV_SEL_NODE

PCI_DEVSEL.TRST = /PROC_BGRT * DEV_SEL_NODE * /RESET

PCI_DEVSEL = VCC

DEV_SEL_NODE.CLKF = MEMCLK

DEV_SEL_NODE :=   /PROC_BGRT * PCI_FRAME * A31 * /A30 * PCI_FRAME_D
                + /PROC_BGRT * DEV_SEL_NODE * PCI_FRAME
                + /PROC_BGRT * DEV_SEL_NODE * /PCI_FRAME * /PCI_TRDY
                + PROC_BGRT * PCI_FRAME * PCI_DEVSEL
                + PROC_BGRT * PCI_FRAME * DEV_SEL_NODE
                + PROC_BGRT * /PCI_FRAME * /PCI_TRDY

PCI_TRDY.TRST = /PROC_BGRT * DEV_SEL_NODE * /RESET

PCI_TRDY = MEM_RDY * /ADD_STOP

PCI_IRDY.TRST = PROC_BGRT * /RESET

PCI_IRDY.CLKF = MEMCLK

PCI_IRDY :=   PCI_FRAME * /PCI_IRDY
            + PCI_IRDY * PCI_FRAME
            + PCI_IRDY * /PCI_FRAME * /PCI_TRDY

PCI_C_BE0.TRST = PROC_BGRT * /RESET

PCI_C_BE0 =   PCI_FRAME * /PCI_FRAME_D * WRITE
            + PCI_FRAME * PCI_FRAME_D * /WE0 * WRITE
            + PCI_FRAME * PCI_FRAME_D * /WRITE

PCI_C_BE1.TRST = PROC_BGRT * /RESET

PCI_C_BE1 =   PCI_FRAME * /PCI_FRAME_D
            + PCI_FRAME * PCI_FRAME_D * /WE1 * WRITE
            + PCI_FRAME * PCI_FRAME_D * /WRITE

PCI_C_BE2.TRST = PROC_BGRT * /RESET

PCI_C_BE2 =   PCI_FRAME * /PCI_FRAME_D * PROC_IO_MEM * A31 * /A30
            + PCI_FRAME * PCI_FRAME_D * /WE2 * WRITE
            + PCI_FRAME * PCI_FRAME_D * /WRITE

PCI_C_BE3.TRST = PROC_BGRT * /RESET
```

```
PCI_C_BE3 =   PCI_FRAME * /PCI_FRAME_D * PROC_IO_MEM * A31 * A30
            + PCI_FRAME * PCI_FRAME_D * /WE3 * WRITE
            + PCI_FRAME * PCI_FRAME_D * /WRITE

PCI_IDSEL0 = PROC_BGRT * REQ * A31 * A30 * /A29 * /A28

PCI_IDSEL1 = PROC_BGRT * REQ * A31 * A30 * /A29 * A28

PCI_IDSEL2 = PROC_BGRT * REQ * A31 * A30 * A29 * /A28

PCI_DATA_CYCLE = PCI_FRAME * PCI_FRAME_D

DATA_EN =   PROC_BGRT * REQ * A31 * PCI_FRAME_D
          + /PROC_BGRT * DEV_SEL_NODE

DATA_DIR =   PROC_BGRT * REQ * WRITE
           + /PROC_BGRT * DEV_SEL_NODE * INTERNAL_WRITE

WRITE.TRST = /PROC_BGRT

WRITE = INTERNAL_WRITE

INTERNAL_WRITE =   PROC_BGRT * WRITE
                 + /PROC_BGRT * PCI_FRAME * PCI_C_BE0 * /PCI_FRAME_D
                 + /PROC_BGRT * PCI_FRAME * INTERNAL_WRITE * PCI_FRAME_D
                 + /PROC_BGRT * PCI_FRAME * PCI_C_BE0 * INTERNAL_WRITE

; MEMORY STATE MACHINES FOR THE RAS CAS GENERATION

MINIMIZE_OFF

IDLE.CLKF = MEMCLK

IDLE :=   /IDLE * /MEM_ACCESS * /REF_ACCESS
        + RESETD
        + /IDLE * PROC_BGRT * MEM_ACCESS * ST1 * MEM_RDY * /BURST
        + /IDLE * /PROC_BGRT * MEM_ACCESS * ST1 * MEM_RDY * /PCI_FRAME
        + /IDLE * REF_ACCESS * /REF_REQ
        + IDLE * PROC_BGRT * /REF_REQ * /( REQ * /A31 * A30 )
        + IDLE * /PROC_BGRT * /REF_REQ * /( PCI_DATA_CYCLE * A31 * /A30 )

MINIMIZE_ON

REF_ACCESS.CLKF = MEMCLK

REF_ACCESS :=   IDLE * REF_REQ * /REF_ACCESS
              + REF_ACCESS * REF_REQ

MEM_ACCESS.CLKF = MEMCLK

MEM_ACCESS :=   IDLE * PROC_BGRT * REQ * /A31 * A30 * /REF_REQ * /MEM_ACCESS
              + IDLE * /PROC_BGRT * PCI_DATA_CYCLE * A31 * /A30
                    * /REF_REQ * /MEM_ACCESS
              + MEM_ACCESS * /ST1
              + MEM_ACCESS * PROC_BGRT * BURST
              + MEM_ACCESS * /PROC_BGRT * PCI_FRAME
              + MEM_ACCESS * /PROC_BGRT * /PCI_FRAME*/PCI_IRDY

ST1.CLKF = MEMCLK
```

```
ST1 :=   MEM_ACCESS

MEM_RDY.CLKF = MEMCLK

MEM_RDY :=    PROC_BGRT * REQ * /A31 * /A30 * /MEM_RDY
            + MEM_ACCESS * /MEM_RDY
            + PROC_BGRT * MEM_ACCESS * MEM_RDY * BURST
            + /PROC_BGRT * MEM_ACCESS * MEM_RDY * PCI_FRAME
            + /PROC_BGRT * MEM_ACCESS * MEM_RDY * /PCI_FRAME * /PCI_IRDY

; ROM_CS ALSO DRIVES THE RDN PIN OF THE Am29030 PROCESSOR AND NEEDS TO BE 1 IF THE
; ROM IS 8 BITS WIDE AND 0 IF IT 16 BITS WIDE DURING A RESET. SO TO DO THIS,
; RESET WOULD BE OR'D WITH THE ROM_CS EQUATION IF 16-BIT MEMORY IS NEEDED.
; IT IS CURRENTLY COMMENTED OUT.

ROM_CS =    PROC_BGRT * REQ * /A31 * /A30
;         + RESET

PROC_RDY =   PROC_BGRT * MEM_RDY
           + PROC_BGRT * PCI_DATA_CYCLE * PCI_TRDY * PCI_IRDY

MINIMIZE_OFF

RAS0 =   MEM_ACCESS
       + REF_ACCESS * /MEMCLK
       + RAS0 * REF_ACCESS

MUX =     MEM_ACCESS * /MEMCLK
        + MEM_ACCESS * MUX

CAS0 =   REF_ACCESS
       + MUX * MEM_ACCESS * ST1 * /INTERNAL_WRITE * MEMCLK * DELAY_IN
       + MUX * MEM_ACCESS * ST1 * /INTERNAL_WRITE * CAS0 * DELAY_IN
       + MUX * MEM_ACCESS * ST1 * /INTERNAL_WRITE * CAS0 * /MEMCLK
       + MUX * MEM_ACCESS * ST1 * PROC_BGRT * INTERNAL_WRITE * WE0 * /MEMCLK
       + MUX * MEM_ACCESS * ST1 * /PROC_BGRT * INTERNAL_WRITE * PCI_C_BE0
         * /MEMCLK * /ADD_STOP

CAS1 =   REF_ACCESS
       + MUX * MEM_ACCESS * ST1 * /INTERNAL_WRITE * MEMCLK * DELAY_IN
       + MUX * MEM_ACCESS * ST1 * /INTERNAL_WRITE * CAS1 * DELAY_IN
       + MUX * MEM_ACCESS * ST1 * /INTERNAL_WRITE * CAS1 * /MEMCLK
       + MUX * MEM_ACCESS * ST1 * PROC_BGRT * INTERNAL_WRITE * WE1 * /MEMCLK
       + MUX * MEM_ACCESS * ST1 * /PROC_BGRT * INTERNAL_WRITE * PCI_C_BE1
         * /MEMCLK * /ADD_STOP

CAS2 =   REF_ACCESS
       + MUX * MEM_ACCESS * ST1 * /INTERNAL_WRITE * MEMCLK * DELAY_IN
       + MUX * MEM_ACCESS * ST1 * /INTERNAL_WRITE * CAS2 * DELAY_IN
       + MUX * MEM_ACCESS * ST1 * /INTERNAL_WRITE * CAS2 * /MEMCLK
       + MUX * MEM_ACCESS * ST1 * PROC_BGRT * INTERNAL_WRITE * WE2 * /MEMCLK
       + MUX * MEM_ACCESS * ST1 * /PROC_BGRT * INTERNAL_WRITE * PCI_C_BE2
         * /MEMCLK * /ADD_STOP

CAS3 =   REF_ACCESS
       + MUX * MEM_ACCESS * ST1 * /INTERNAL_WRITE * MEMCLK * DELAY_IN
       + MUX * MEM_ACCESS * ST1 * /INTERNAL_WRITE * CAS3 * DELAY_IN
       + MUX * MEM_ACCESS * ST1 * /INTERNAL_WRITE * CAS3 * /MEMCLK
       + MUX * MEM_ACCESS * ST1 * PROC_BGRT * INTERNAL_WRITE * WE3 * /MEMCLK
       + MUX * MEM_ACCESS * ST1 * /PROC_BGRT * INTERNAL_WRITE * PCI_C_BE3
         * /MEMCLK * /ADD_STOP

Q0.CLKF = MEMCLK

Q0.T := VCC

Q1.CLKF = MEMCLK

Q1.T :=  Q0

Q2.CLKF = MEMCLK

Q2.T := Q1 * Q0

Q3.CLKF = MEMCLK

Q3.T :=  Q2 * Q1 * Q0

Q4.CLKF = MEMCLK

Q4.T := Q3 * Q2 * Q1 * Q0

Q5.CLKF = MEMCLK

Q5.T := Q4 * Q3 * Q2 * Q1 * Q0

Q6.CLKF = MEMCLK

Q6.T := Q5 * Q4 * Q3 * Q2 * Q1 * Q0

REF_REQ.CLKF = MEMCLK

REF_REQ :=   Q6 * Q5 * Q4 * Q3 * Q2 * Q1 * Q0
           + REF_REQ * /REF_ACCESS

;-------------------------------------------------------------------
```

ADD_INCR PAL EQUATIONS

```
;PALASM Software Design Description

;---------------------------------- Declaration Segment ------------
TITLE    ADDRESS COUNTER AND INCREMENTER FOR THE PCI ADDRESS BUS
PATTERN  ADD_INCR.PDS
REVISION A
AUTHOR   DAVID STOENNER
COMPANY  AMD
DATE     04/02/93

CHIP  U10  PALCE22V10 DEVICE

;---------------------------------- PIN Declarations ---------------
PIN  1          MEMCLK
PIN  2          /INC
PIN  3          A2
PIN  4          A3
PIN  5          A4
PIN  6          A5
PIN  7          A6
PIN  8          A7
PIN  9          A8
PIN  10         A9
PIN  11         /OE
PIN  12         GND
PIN  13         /LOAD
PIN  22         Q2                                    REGISTERED
PIN  21         Q3                                    REGISTERED
PIN  20         Q4                                    REGISTERED
PIN  19         Q5                                    REGISTERED
PIN  18         Q6                                    REGISTERED
PIN  17         Q7                                    REGISTERED
PIN  16         Q8                                    REGISTERED
PIN  15         Q9                                    REGISTERED
PIN  23         /ADD_STOP                             REGISTERED
PIN  24         VCC

;------------------------------------ Boolean Equation Segment ------
EQUATIONS

ADD_STOP := INC*Q9*Q8*Q7*Q6*Q5*Q4*Q3*Q2

Q2 :=   INC*/Q2
      + LOAD*A2

Q3 :=   INC*(Q3 :+: Q2)
      + LOAD*A3

Q4 :=   INC*(Q4 :+: (Q3*Q2))
      + LOAD*A4

Q5 :=   INC*(Q5 :+: (Q4*Q3*Q2))
      + LOAD*A5

Q6 :=   INC*(Q6 :+: (Q5*Q4*Q3*Q2))
      + LOAD*A6

Q7 :=   INC*(Q7 :+: (Q6*Q5*Q4*Q3*Q2))
      + LOAD*A7

Q8 :=   INC*(Q8 :+: (Q7*Q6*Q5*Q4*Q3*Q2))
      + LOAD*A8

Q9 :=   INC*(Q9 :+: (Q8*Q7*Q6*Q5*Q4*Q3*Q2))
      + LOAD*A9

Q2.TRST = OE
Q3.TRST = OE
Q4.TRST = OE
Q5.TRST = OE
Q6.TRST = OE
Q7.TRST = OE
Q8.TRST = OE
Q9.TRST = OE

;--------------------------------------------------------------------
```

ADD_REG PAL EQUATIONS

```
;PALASM Software Design Description

;---------------------------------- Declaration Segment ------------
TITLE    ADDRESS REGISTER FOR THE 32-BIT PCI ADDRESS BUS
PATTERN  ADD_REG.PDS
REVISION A
AUTHOR   DAVID STOENNER
COMPANY  AMD
DATE     04/22/93

CHIP  UXX  PALCE20V8 DEVICE

;---------------------------------- PIN Declarations ---------------
PIN  1          MEMCLK
PIN  3          A0
PIN  4          A1
PIN  5          A2
PIN  6          A3
PIN  7          A4
PIN  8          A5
PIN  9          A6
PIN  10         A7
PIN  12         GND
PIN  13         /OE
PIN  15         Q0                                   REGISTERED
PIN  16         Q1                                   REGISTERED
PIN  17         Q2                                   REGISTERED
PIN  18         Q3                                   REGISTERED
PIN  19         Q4                                   REGISTERED
PIN  20         Q5                                   REGISTERED
PIN  21         Q6                                   REGISTERED
PIN  22         Q7                                   REGISTERED
PIN  23         /LOAD
PIN  24         VCC

;----------------------------------- Boolean Equation Segment ------
EQUATIONS

Q0 :=   /LOAD*Q0
      + LOAD*A0

Q1 :=   /LOAD*Q1
      + LOAD*A1

Q2 :=   /LOAD*Q2
      + LOAD*A2

Q3 :=   /LOAD*Q3
      + LOAD*A3

Q4 :=   /LOAD*Q4
      + LOAD*A4

Q5 :=   /LOAD*Q5
      + LOAD*A5

Q6 :=   /LOAD*Q6
      + LOAD*A6

Q7 :=   /LOAD*Q7
      + LOAD*A7

;-------------------------------------------------------------------
```