1.0 Introduction

1.1 Overview

The VRC4373™ system controller is a software-configurable chip that directly interfaces the VR4300™ CPU and PCI bus without external logic or buffering, and also interfaces memory (SDRAM, EDO, fast-page DRAM, and flash boot ROM) with minimal buffering. From the viewpoint of the VR4300 CPU, the VRC4373 acts as a memory controller, DMA controller, and PCI bridge. From the viewpoint of PCI agents, the VRC4373 acts as either a PCI bus master or a PCI bus target. Alternatively, the VRC4373 may be located on a PCI bus add-on board.

1.2 Features

- **CPU Interface**
  - Direct connection to the 66MHz VR4300 CPU bus
  - 3.3V I/O
  - Support for all VR4300 bus cycles
  - Little-endian or big-endian byte order
- **Memory Interface**
  - Support for boot ROM, base memory, and up to four SIMM™ (DIMM) ranges
  - Programmable address ranges for base memory and SIMM memory
  - 66 MHz memory bus
  - Base memory range: SDRAM and EDO DRAM
  - SIMM memory range: SDRAM, EDO and fast-page DRAM, and flash ROM
  - Several speed grades supported within each memory range
  - Bank-interleaved or non-bank-interleaved SIMM memory ranges
  - On-chip bank-interleaving buffers
  - Open DRAM page maintained within base memory
  - 8-word (32-byte) write FIFO (CPU-to-memory)
  - 2-word (8-byte) prefetch FIFO (memory-to-CPU or memory-to-PCI)
  - On-chip DRAM and SDRAM refresh generation
  - Up to 16 MB of write-protectable boot ROM
  - Boot ROM address and data signals multiplexed on DRAM data signals
  - 3.3V inputs; 5V-tolerant outputs
- **PCI Interface**
  - Master and target capability
  - Host bridge and add-on board modes
  - PCI bus arbiter
  - 4-word (16-byte) bidirectional PCI master FIFO (CPU = PCI bus master)

- 8-word (32-byte) bidirectional PCI target FIFO (memory is PCI bus target)
- 33 MHz PCI bus clock rate
- 133 MB/sec burst transfers
- Interrupt support for add-on board mode
- 3.3V PCI-compliant inputs; 5V-tolerant outputs

❑ **DMA Controller**

- CPU-initiated block transfers between memory and PCI bus
- 8-word (32-byte) bidirectional DMA FIFO
- Two sets of DMA control registers for chained transfers (one set is programmed while data is transferred on the other)
- Bidirectional unaligned transfers
- Transfers at maximum PCI bandwidth of 133 Mb/s

1.3 Terminology

In this document:

❑ *Word* means 4 bytes. This definition of *word* differs from the definition in the *PCI Local Bus Specification*, where a *word* is 2 bytes.

❑ *B* means byte.

❑ *b* means bit.

❑ *Memory* means the local memory attached to the VRC4373 controller.

❑ *SIMM* means Single or Dual In-line Memory Module (SIMM or DIMM), unless explicitly stated otherwise.

❑ *Module* mean a set of chips, as in a SIMM or DIMM.

❑ *SDRAM* means synchronous DRAM

1.4 Reference Documents

The following documents form a part of this data sheet. Unless otherwise specified, the latest version of each document applies.

❑ *"*MIPS® R4300 Preliminary RISC Processor Specification Revision 2.2" (available from the MIPS Group, a division of Silicon Graphics, Inc.)

❑ "PCI Local Bus Specification" Revision 2.1 and "PCI System Design Guide" Revision 1.0 (available from the Peripheral Component Interconnect Special Interest Group)

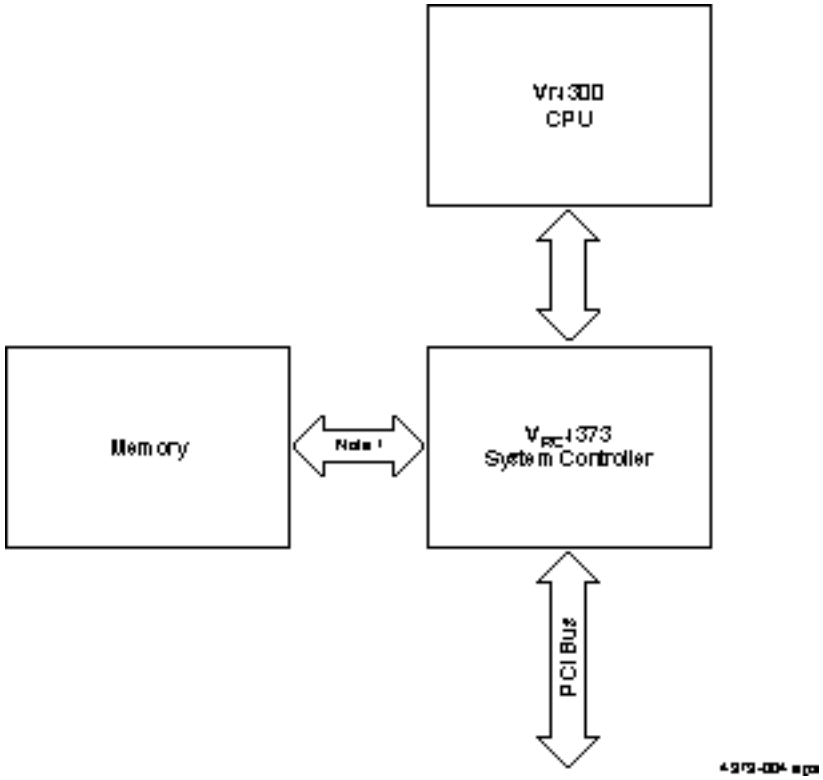
❑ NEC *VR4300™ Microprocessor Data Sheet* (doc. no. U10116EJ3V0DS00)

1.5

System Block Diagram

Figure 1 shows the controller used as a host bridge in a typical system. Alternatively, the controller can be located on a PCI bus add-on board.

Figure 1: System Block Diagram



Note: F244 or F245 buffers may be needed on the MuxAd bus and, for DIMM modules, on certain chip select signals.

2.0 Signal Summary

The controller has 208 signals and 51 power/ground pins, for a total of 251 pins. Table 3 through Table 4 summarize the signal functions.

Table 1: CPU Interface Signals

Signal	I/O	Reset Value	Pullup/ Pulldown	Max. AC Load (pF)	Max. DC Drive (mA)	Description
EOK#	O	High		20	24	External ready; signifies that the controller is capable of accepting a processor request
EValid#	O	High		20	6	External agent valid. Indicates that the controller is driving valid information on the SysAD and SysCmd buses
Int#	O	High		30	6	Interrupt request
MasterClock	O	Toggle		20	24	66 MHz MasterClock to CPU
NMI#	O	High		20	12	Non-maskable interrupt; asserted when a PCI device asserts SERR#
PValid#	I					Processor valid; signifies that the V_R4300 is driving valid information on the SysAD and SysCmd buses
SysAD[31:0]	I/O	Hi-Z		20	6	System address/data bus
SysCmd[4:0]	I/O	Hi-Z		20	6	System command/data ID bus

Table 2: Memory Interface Signals

Signal	I/O	Reset Value	Pullup/ Pulldown	Max. AC Load (pF)	Max. DC Drive (mA)	Description
BOE#	O	High		50	24	Base memory output-enable
BRAS#	O	High		50	24	Base memory row address strobe
BROMCS#	O	High		30	12	Boot ROM chip select
BWE#	O	High		50	24	Base memory write enable
MCASa[3:0]#	O	High		75	24	Column address strobe, even addresses
MCASb[3:0]#	O	High		75	24	Column address strobe, odd addresses
MRAS[3:0]#	O	High		75	24	SIMM memory row address strobes
MDa[31:0]	I/O	High		70	12	Memory data (even), boot ROM address
MDb[31:0]	I/O	High		70	12	Memory data (odd), boot ROM data
MuxAd[9:0]	I/O	Hi-Z	50K down	75	24	Multiplexed row/column address
Mux[10]	I/O	Hi-Z		75	24	Multiplexed row/column address; endian select
MuxAd[13:11]	O	Hi-Z		75	24	Multiplexed row/column address
MWE#	O	High		30	24	Boot ROM and SIMM write enable
SDCAS#	O	High		80	24	SDRAM column address strobe
SDRAS#	O	High		80	24	SDRAM row address strobe
SDCKE[3:0]	O	High		70	24	SDRAM clock enable
SDCLK[3:0]	O	High		50	24	66 MHz SDRAM clock
SDCS[1:0]#	O	High		50	24	SDRAM command select

Table 3: PCI Interface Signals

Signal	I/O	Reset Value	Pullup/ Pulldown	Max. AC Load (pF)	Max. DC Drive (mA)	Description
AD[31:0]	I/O	Hi-Z		110	PCI [a]	PCI A/D[31:0], multiplexed address and data bus
CBE[3:0]#	I/O	Hi-Z		110	PCI	PCI C/BE[3:0]#, bus command and byte-enables
CLK[3:0]	O	Toggle		50	PCI	PCI CLK, 33 MHz
DEVSEL#	I/O	Hi-Z		110	PCI	PCI DEVSEL#, device select
FRAME#	I/O	Hi-Z		110	PCI	PCI FRAME#, cycle frame
GNT[0]#	I/O	High		10	PCI	PCI GNT#, bus grant
GNT[3:1]#	O	High		10	PCI	PCI GNT#, bus grant
IDSEL	I					PCI IDSEL, initialization device select
INTA#	I/O			10	PCI	PCI INTA#, interrupt A
IRDY#	I/O	Hi-Z		110	PCI	PCI IRDY#, initiator ready
LOCK#	I/O	Hi-Z		10	PCI	PCI LOCK#, lock atomic operation
PAR	I/O	Hi-Z		110	PCI	PCI PAR, parity of A/D[31:0] and C/BE[3:0]#
PERR#	I/O	Hi-Z		10	PCI	PCI PERR#, parity error
REQ[0]#	I/O			10	PCI	PCI REQ#, bus request
REQ[3:1]#	I					PCI REQ#, bus request
RST#	I					PCI RST#, reset
SERR#	I/O	Hi-Z		10	PCI	PCI SERR#, system error
STOP#	I/O	Hi-Z		110	PCI	PCI STOP#, stop request from target
TRDY#	I/O	Hi-Z		110	PCI	PCI TRDY#, target ready

a. Compatible with PCI specification

Table 4: Utility Signals

Signal	I/O	Reset Value	Pullup/ Pulldown	Max. AC Load (pF)	Max. DC Drive (mA)	Description
REFCLK	I	Toggle				66 MHz system reference clock

3.0 Register, Resource, and Implementation

3.1 Register Summary

Table 5 summarizes the controller's register set (base address 0F00_0000 in system memory). Accesses above offset 0x1FF return 0 with the data-error bit set on SysCMD[0], update the controller's bus error status register (Section 8.1.1), and cause an interrupt (Int#), if enabled.

Table 5: Register Summary

Offset from Base 0x0F00_0000	Register Name	Size (bytes)	CPU-Bus R/W	PCI bus (R/W)	Reference
0x0	Base memory control register	4	R/W	*Not accessible*	Section 5.6.1 on page 18
0x4	SIMM memory control register 1	4	R/W	*Not accessible*	Section 5.7.1 on page 25
0x8	SIMM memory control register 2	4	R/W	*Not accessible*	Section 5.7.1 on page 25
0xC	SIMM memory control register 3	4	R/W	*Not accessible*	Section 5.7.1 on page 25
0x10	SIMM memory control register 4	4	R/W	*Not accessible*	Section 5.7.1 on page 25
0x14	PCI master address window register 1	4	R/W	*Not accessible*	Section 6.3.1 on page 35
0x18	PCI master address window register 2	4	R/W	*Not accessible*	Section 6.3.1 on page 35
0x1C	PCI target address window register 1	4	R/W	*Not accessible*	Section 6.4.1 on page 37
0x20	PCI target address window register 2	4	R/W	*Not accessible*	Section 6.4.1 on page 37
0x24	PCI master I/O window register	4	R/W	*Not accessible*	Section 6.3.1 on page 35
0x28	PCI configuration data register	4	R/W	*Not accessible*	Section 6.5 on page 38
0x2C	PCI configuration address register	4	R/W	*Not accessible*	Section 6.5 on page 38
0x30	PCI mailbox register 1	4	R/W	R/W	Section 6.11 on page 48
0x34	PCI mailbox register 2	4	R/W	R/W	Section 6.11 on page 48
0x38	DMA control register 1	4	R/W	*Not accessible*	
0x3C	DMA memory address register 1	4	R/W	*Not accessible*	
0x40	DMA PCI address register 1	4	R/W	*Not accessible*	
0x44	DMA control register 2	4	R/W	*Not accessible*	
0x48	DMA memory address register 2	4	R/W	*Not accessible*	
0x4C	DMA PCI address register 2	4	R/W	*Not accessible*	
0x50	Bus error status register	4	R	*Not accessible*	Section 8.1.1 on page 56
0x54	Interrupt control and status register	4	R/W	*Not accessible*	Section 8.1.2 on page 57
0x58	DRAM refresh counter register	4	R/W	*Not accessible*	Section 5.8.1 on page 30
0x5C	Boot ROM write-protect register	4	R/W	*Not accessible*	Section 5.5.1.2 on page 17
0x60	PCI exclusive access register	4	R/W	*Not accessible*	Section 6.12.1 on page 49
0x64	DMA words remaining register	4	R	*Not accessible*	
0x68	DMA current memory address register	4	R	*Not accessible*	
0x6C	DMA current PCI address register	4	R	*Not accessible*	
0x70	PCI retry counter	4	R	*Not accessible*	Section 6.8 on page 47
0x74	PCI enable register	4	R/W	*Not accessible*	Section 6.10 on page 47
0x78	Power-on memory initialization register	4	R/W	*Not accessible*	Section 5.10.1 on page 31
0x7C:0xFF	*Reserved*		*Not accessible*	*Not accessible*	
0x100:0x1FF	PCI configuration space registers (host bridge mode)	1, 2, 4	R/W	*Not accessible*	Section 6.5 on page 38
0x100:0x1FF	PCI configuration space registers (add-on board mode, where the controller is located on a PCI bus board rather than on the mother board)	1, 2, 4	*Not accessible*	R/W	Section 6.7 on page 44

3.2 Resource Accessibility

Table 6 summarizes the accessibility, from the CPU and from PCI bus masters, of the controller's internal registers, memory ranges, and PCI bus resources.

Table 6: Resources Accessible Through The V_{RC}4373 System Controller

Resource	Accessible from CPU	Accessible from PCI Bus	Reference
CPU	—	No	Section 4.0 on page 8
Controller's internal registers (except PCI mailboxes)	Word	No	Section 6.0, Section 5.0, Section 7.0, Section 8.0
Boot ROM	Byte writes Word, half-word, or byte reads	No [a]	Section 5.5 on page 15
Base memory	Any CPU burst [b]	Any PCI burst [c]	Section 5.6 on page 18
SIMM memory	Any CPU burst [a]	Any PCI burst [b]	Section 5.7 on page 25
PCI mailboxes	Word	Word [d]	Section 6.11 on page 48
PCI configuration space registers	Word, half-word, or byte [e]	Word, half-word, or byte in add-on board mode only	Section 6.5 on page 38, Section 6.7 on page 44
PCI memory space	Any CPU burst of 4 words or less [a]	No	*PCI Local Bus Specification*
PCI I/O space	Any CPU burst of 4 words or less [a]	No	*PCI Local Bus Specification*
PCI configuration space	Word, half-word, or byte [e]	Word, half-word, or byte	*PCI Local Bus Specification*

a. Because the boot ROM does not support burst transfers, it cannot be accessed from the PCI bus. The PCI interface issues cache-line reads to the target inside the controller.
b. Alignment and burst length as defined by the V_R4300 CPU.
c. Any size burst length, any alignment. Burst may be disconnected by controller.
d. The controller accepts bursts of words to the PCI mailboxes. However, the controller performs a target disconnect without data after each data transfer.
e. Any size access less than or equal to one word, aligned as defined by the V_R4300 CPU.

3.3 Implementation Summary

To create a system using the V_{RC}4373 system controller:

- Configure the hardware using the information provided throughout this data sheet.
- Power-up and initialize the memory, following the steps in Section 5.10 on page 31.
- Initialize the PCI bus interface, using the configuration register information provided in Section 6.0 on page 34.

4.0 CPU Interface

The controller interfaces directly to the VR4300 CPU, in full compliance with the “MIPS R4300 Preliminary RISC Processor Specification,” Revision 2.2. The connection is via the CPU’s 66 MHz SysAD bus using 3.3-volt I/O. All of the CPU’s SysAD bus operations are supported.

4.1 Endian Configuration

The BE bit in the VR4300 CPU’s configuration register specifies the CPU’s byte ordering at reset. BE=0 configures little-endian order; BE=1 configures big-endian order.

The VRC4373 controller’s CPU interface supports either big- or little-endian byte ordering on the SysAd bus. The order depends on the state of the MuxAd[11] signal at reset, as described in Section 10.0. All of the controller’s other interfaces operate only in little-endian mode. The software implications of this, and some related PCI device examples, are described in Section 11.0.

4.2 Data Rate Control

The controller-to-CPU data rate is determined by the EValidn signal. The CPU-to-controller data rate is programmable in the EP field (bits 27:24) of the CPU’s configuration register. Although the CPU supports both D and Dxx data rates, the controller *only* supports the D data rate.

4.3 Address Decoding

The controller latches the address on the SysAD bus. It then decodes the address and SysCmd signals to determine the transaction type. Ten address ranges can be decoded:

- One range for boot ROM
- One range for the controller’s internal configuration registers
- One range for base memory
- Four ranges for SIMM/DIMM memory
- Two ranges for the PCI master address windows
- One range for the PCI I/O address window

Boot ROM is mapped according to its size, as specified in Table 13 on page 16. The controller’s internal registers are fixed at base address 0x0F00_0000, to allow the CPU to access them during boot, before they have been configured. All other decode ranges are programmable.

4.4 Trace Requirements

All traces between the CPU and the controller must be limited to 3 inches or less. TCLK is not used. See Section 9.0 on page 60 for details on clocking.

5.0 Memory Interface

The CPU accesses memory attached to the controller in the normal way, by addressing the system memory space. For large block transfers, the CPU can also initiate DMA transfers between memory and the PCI (either direction), as described in Section 7.0. PCI bus masters gain access the controller's memory through the PCI target address windows, as described in Section 6.4.

The controller's memory interface has the following internal FIFOs that support transfers between memory and the various sources and destinations:

- 8-word (32-byte) write FIFO (CPU-to-memory)
- 2-word (8-byte) prefetch FIFO (memory-to-CPU or memory-to-PCI)
- 8-word (32-byte) bidirectional DMA FIFO (PCI-to-memory or memory-to-PCI)

5.1 Memory Regions and Devices

The controller connects directly to memory and manages the addresses, data and control signals for the following address ranges.

- *Two boot ROM ranges, standard and fault-recovery*
- *One base memory range (programmable)*
- *Four SIMM memory* ranges *(programmable)*. DIMM modules can also be used.

The following types of memory modules can be used:

- *Flash:* can be used in boot ROM and/or SIMM memory ranges
- *EDO DRAM:* for base or SIMM memory
- *Fast-page DRAM:* for SIMM memory
- *Synchronous DRAM (SDRAM):*
  - 16 Mb for base or SIMM memory (NEC part numbers μPD4516421 and μPD4516821)
  - 64 Mb for SIMM memory (NEC part numbers μPD4564421 and μPD4564821)

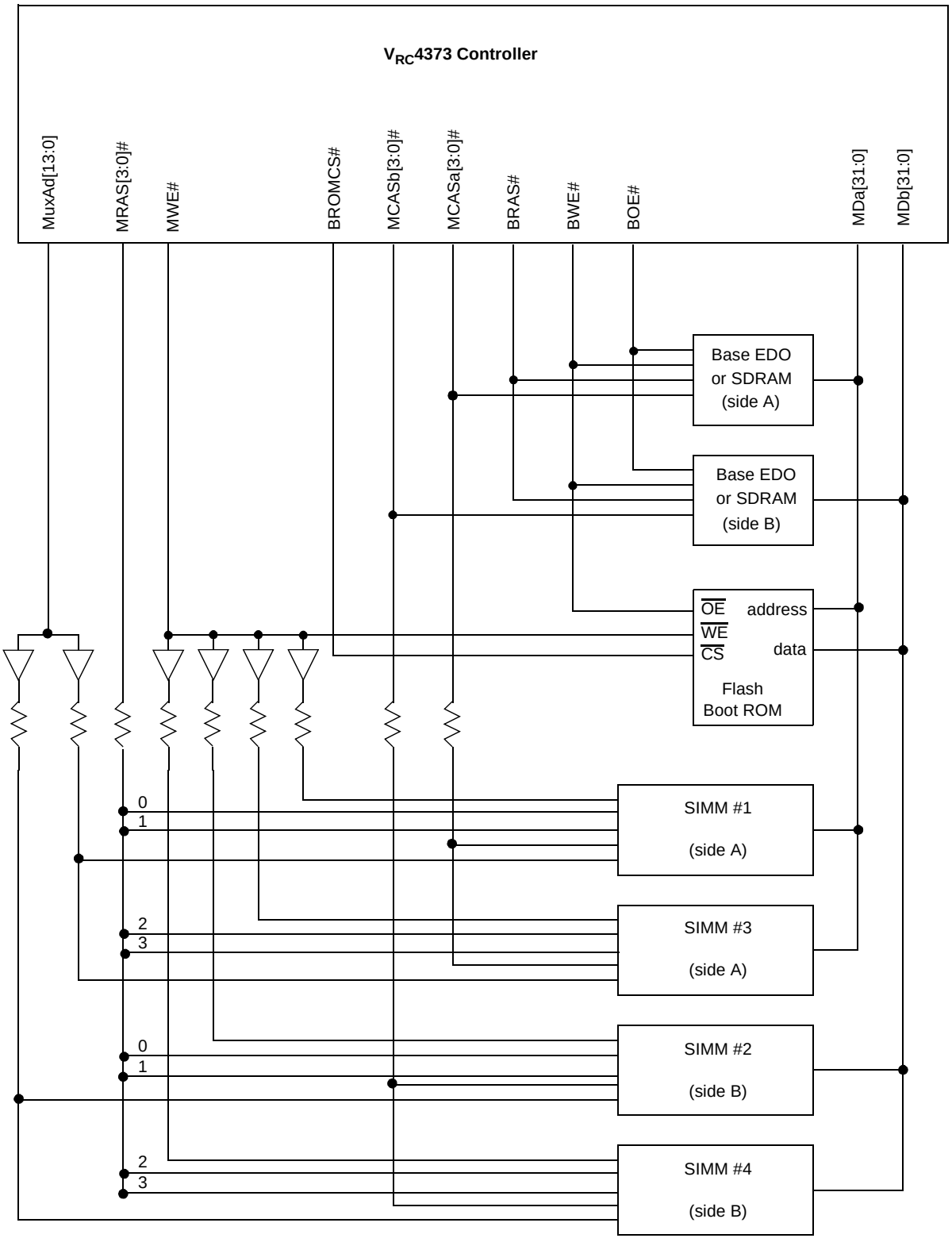
Boot ROM can be populated only with 85-ns flash chips. In addition to its standard boot address range, boot ROM can also be mapped to a fault-recovery range in SIMM Memory slot 4, if that slot is populated with 85-ns flash chips. Prior to accessing boot ROM, software must configure this address range, as described in Section 5.5.

Base memory can be populated with 4Mb EDO or NEC 16 Mb SDRAM chips. If SDRAM is used for base memory, it cannot be bank interleaved. Prior to accessing base memory, software must configure this address range, as described in Section 5.6.

The four SIMM memory ranges can be bank-interleaved, single-sided (SIMM) or double-sided (DIMM), page-mode or non-page mode, and populated with any of the supported memory types. 100-pin DIMMs are the only DIMM package supported. Prior to accessing SIMM memory, software must configure this address range, as described in Section 5.7.

Figure 2 shows a block diagram of controller-to-memory connections for DRAM. Figure 4 on page 23 and Figure 5 on page 29 shows examples of controller-to-memory connections for SDRAM.

Figure 2: Memory Block Diagram



5.2 Address Multiplexing Modes

The controller supports five address multiplexing modes (mux modes) in the base memory and SIMM memory ranges. Table 7 shows these modes and the *row address x column address* configurations they support.

Table 7: Address-Multiplexing Modes

Address Multiplexing Mode	Row-Address X Column-Address Configurations
Mux mode 0:	9x9
Mux mode 1:	10x9, 10x10
Mux mode 2:	11x9, 11x10, 11x11
Mux mode 3:	12x9, 12x10, 12x11, 12x12
Mux mode 4:	14x11 (64 Mb SDRAM only)
Any mux mode:	12x8, 11x8, 10x8

Configuration of the address-multiplexing modes is done in the base memory control register (Section 5.6.1) and SIMM memory control registers (Section 5.7.1). The selection of mode determines which system address bits are output from the controller on the memory-interface MuxAd bus during row and column addressing.

Table 8 shows the MuxAd-to-SysAD for DRAM and flash memory. When DRAM or flash is used for base memory, either the 12x8 or 10x10 configurations may be used.

Table 9 shows the MuxAd-to-SysAD mapping for SDRAM. when SDRAM is used in base memory, mux mode 3 must be used, and only 16Mb SDRAM devices can be used. (64 Mb SDRAM devices can only be used in SIMM memory.)

Table 8: MuxAd-To-SysAd Address Mapping for DRAM and Flash ROM

MuxAd Signals	Bank-Interleaved SysAD Mapping					Non-Bank-Interleaved SysAD Mapping				
	Row	Column				Row	Column			
		Mode 0 (x9)	Mode 1 (x10)	Mode 2 (x11)	Mode 3 (x12)		Mode 0 (x9)	Mode 1 (x10)	Mode 2 (x11)	Mode 3 (x12)
0	11	3	3	3	3	11	3	3	3	3
1	12	4	4	4	4	12	4	4	4	4
2	13	5	5	5	5	13	5	5	5	5
3	14	6	6	6	6	14	6	6	6	6
4	15	7	7	7	7	15	7	7	7	7
5	16	8	8	8	8	16	8	8	8	8
6	17	9	9	9	9	17	9	9	9	9
7	18	10	10	10	10	10	2	2	2	2
8	19	20	21	22	23	18	19	20	21	22
9	20	21	22	23	24	19	20	21	22	23
10	21	22	23	24	25	20	21	22	23	24
11	22	23	24	25	26	21	22	23	24	25

Table 9: MuxAd-To-SysAd Address Mapping for SDRAM

MuxAd Signals	Bank-Interleaved SysAD Mapping			Non-Bank-Interleaved SysAD Mapping		
	Row	Column		Row	Column	
		Mode 3 [a] (16 Mb SDRAM)	Mode 4 [b] (64 Mb SDRAM)		Mode 3 (16 Mb SDRAM)	Mode 4 [a] (64 Mb SDRAM)
0	11	3	3	11	2	2
1	12	4	4	12	3	3
2	13	5	5	13	4	4
3	14	6	6	14	6	6
4	15	7	7	15	7	7
5	16	8	8	16	8	8
6	17	9	9	17	9	9
7	18	10	10	10	5	5
8	19	23	25	18	22	24
9	20	24	26	19	23	25
10	21	*Hardwired to 0*	*Hardwired to 0*	20	*Hardwired to 0*	*Hardwired to 0*
11	22	22	22	21	21	21
12	23	*Not used*	*Hardwired to 0*	22	*Not used*	*Hardwired to 0*
13	24	*Not used*	*Hardwired to 0*	23	*Not used*	*Hardwired to 0*

a. 16 Mb SDRAM can be used in either the base memory or SIMM memory ranges. Base memory must be non-bank interleaved, non-page mode.

b. 64 Mb SDRAM is supported only in the SIMM memory regions, in the bank-interleaved mode. 64 Mb SDRAM cannot be used in the base memory range.

5.3 Memory Performance

The speed of memory accesses is determined by memory type, speed, and bank interleaving. Table 10 lists examples of the number of 66 MHz memory bus clocks required for each transfer of an 8-word (32-byte) CPU instruction cache line fill. The first number in the "SysAD CPU Clocks (66 MHz)" column is for the first word; the remaining numbers are for the subsequent words. Only the most common combinations are shown.

Table 10: Examples of Memory Performance [a]

Memory Type	Bank Interleaved	Page Mode	Page Hit	R/W	Sequential Addresses [b]	Base Memory or SIMM	SysAD CPU Clocks (66MHz)
SDRAM, 10 ns	No	No	No	R	No	Base	9-1-1-1-1-1-1-1
SDRAM, 10 ns	No	No	No	W	No	Base	7-1-1-1-1-1-1-1
SDRAM, 10 ns	No	No	No	R	No	SIMM	11-1-3-1-3-1-3-1
SDRAM, 10 ns	No	No	No	W	No	SIMM	9-4-4-4-4-4-4-4
EDO, 60 ns	No	No	No	R		Base	9-2-2-2-2-2-2-2
EDO, 60 ns	Yes	No	No	R		Base	9-1-1-1-1-1-1-1
EDO, 60 ns	Yes	Yes	No	R		Base	13-1-1-1-1-1-1-1
EDO, 60 ns	Yes	Yes	Yes	R	Yes	Base	3-1-2-1-1-1-1-1 [c]
EDO, 60 ns	Yes	Yes	Yes	R	No	Base	7-1-1-1-1-1-1-1
EDO, 60 ns	No	No	No	W	*N/A*	Base	7-2-2-2-2-2-2-2
EDO, 60 ns	Yes	No	No	W		Base	7-2-2-2 [d]
EDO, 60 ns	Yes	Yes	No	W		Base	11-2-2-2 [d]
EDO, 60 ns	Yes	Yes	Yes	W		Base	5-2-2-2 [d]
EDO, 60 ns	Yes	No	No	R		SIMM	11-1-3-1-3-1-3-1
EDO, 60 ns	No	No	No	R		SIMM	11-4-4-4-4-4-4-4
EDO, 60 ns	Yes	No	No	W		SIMM	9-4-4-4 [d]
EDO, 60 ns	No	No	No	W		SIMM	9-4-4-4-4-4-4-4
Fast-page, 70 ns	Yes	No	No	R		SIMM	11-1-4-1-4-1-4-1
Fast-page, 70 ns	No	No	No	R		SIMM	11-5-5-5-5-5-5-5
Fast-page, 70 ns	Yes	No	No	W		SIMM	10-5-5-5-[d]
Fast-page, 70 ns	No	No	No	W		SIMM	10-5-5-5-5-5-5-5
Flash, 85 ns	Yes	No	No	R		SIMM	13-1-6-1-6-1-6-1
Flash 85 ns	No	No	No	R		SIMM	13-7-7-7-7-7-7-7
Flash 85 ns	Yes	No	No	W		SIMM	11-6-6-6 [d]
Flash 85 ns	No	No	No	W		SIMM	11-6-6-6-6-6-6-6

a. Read performance is calculated by counting the rising edge for TCLK, where the read command is issued by the CPU. Because the CPU issues write data with no wait-states once the write command is issued, the numbers in the table represent the rate at which data is written to memory. The sum of the numbers represents the number of cycles between when the write operation was issued and when the next CPU memory operation can begin.

b. *N/A* means not applicable.

c. After a one- or two-word read starting at offset 0, an instruction-cache line fill that is sequential to the previous one- or two-word read actually has a page-hit wait-state pattern, not sequential.

d. Writes to bank-interleaved memory are performed as 8-byte (double-word) quantities. Thus, the time to write a double-word to bank-interleaved memory only specifies four wait-state numbers.

5.4 Placement, Loading, and Example Delays

Figure 3 shows the physical placement recommendations for mother-board chips. Table 11 shows minimum and maximum AC loadings for the memory interface signals. Table 12 shows example trace delays between memory and the controller.

Figure 3: Memory Placement

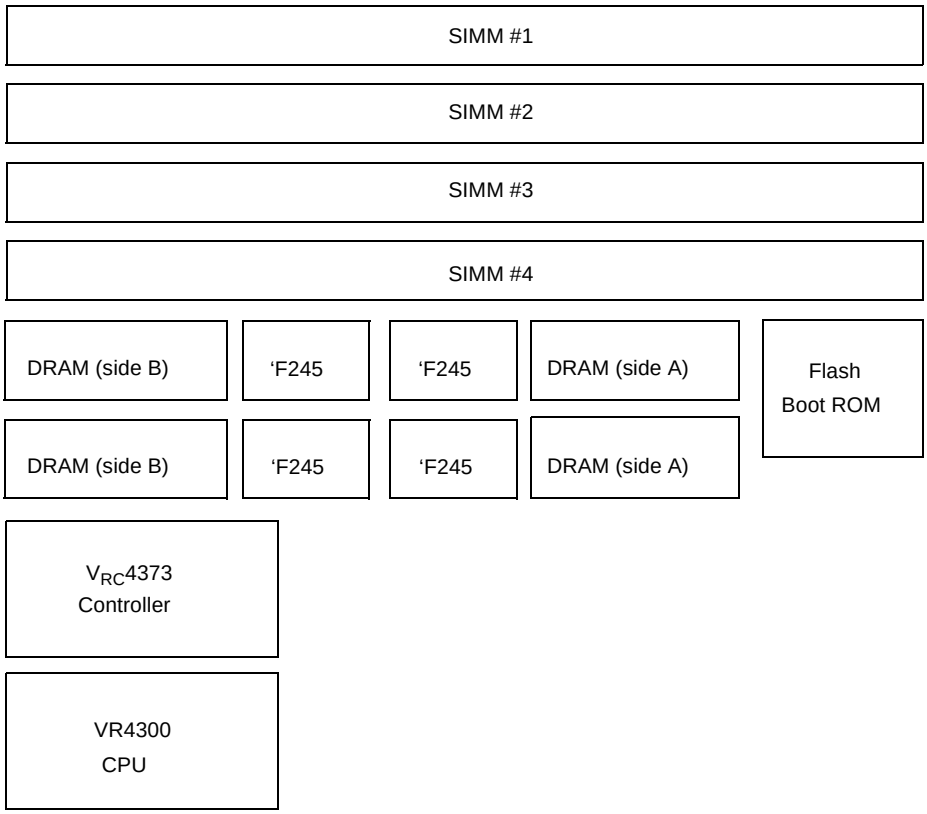


Table 11: Memory-Signal AC Loading

Signal	Min (pF)	Max (pF)	Description
MuxAd[9:0] [a]	10	75	Multiplexed row/column address
MuxAd[13:10]	10	75	Multiplexed row/column address
BRAS#	5	50	Base memory row address strobe
MRAS[3:0]#	15	75	SIMM nemory row address strobes
MCASa[3:0]#	20	75	Column address strobe, even addresses
MCASb[3:0]#	20	75	Column address strobe, odd addresses
MWE#	10	30	Boot ROM and SIMM write enable
BOE#	10	50	Base memory write enable
BWE#	10	50	Base memory output enable
BROMCS#	10	30	Boot ROM chip select
MDa[31:0]	20	70	Memory data (even), boot ROM address
MDb[31:0]	20	70	Memory data (odd), boot ROM data
SDCLK[3:0]	10	50	66 MHz SDRAM clock

Table 11: Memory-Signal AC Loading (Continued)

Signal	Min (pF)	Max (pF)	Description
SDCKE[3:0]	10	70	SDRAM clock enable
SDCS[1:0]#	10	50	SDRAM command select
SDCAS#	20	80	SDRAM column address strobe
SDRAS#	20	80	SDRAM row address strobe

a. MuxAd[9:0] needs a pull down of 50K Ohm externally for the gate-array version.

Table 12: Example SIMM DRAM Delays

Signal	Source	Destination	Delay Subtotal (250ps/inch)	RC Delay (2 x R x C)	Total Delay [a]
MCASa[3:0]#	Controller	SIMM DRAM	1.8 ns	2 x 33 x 100pF = 6.6 ns	8.4 ns
MCASb[3:0]#	Controller	SIMM DRAM	1.8 ns	2 x 33 x 100pF = 6.6 ns	8.4 ns
MuxAd[11:0] [b]	Controller Buffer input Buffer output	Buffer Buffer output (250 pF) SIMM DRAM	0.5 ns 14.0 ns 2.0 ns	 2 x 10 x 250pF = 5.0 ns	21.5 ns
MDa[31:0]	Controller	SIMM DRAM	2.0 ns		2.0 ns
MDb[31:0]	Controller	SIMM DRAM	2.0 ns		2.0 ns
MDa[31:0]	SIMM DRAM	Controller	2.0 ns		2.0 ns
MDb[31:0]	SIMM DRAM	Controller	2.0 ns		2.0 ns
MRAS#	Controller	SIMM DRAM	2.2 ns	2 x 33 x 180pF = 12 ns	14.2 ns
MWE# [b]	Controller Buffer input Buffer output	Buffer Buffer output (130 pF) SIMM DRAM	1.3 ns 11.0 ns 2.0 ns	 2 x 10 x 130pF = 2.6 ns	16.9 ns

a. From controller to SIMM DRAM.
b. To accommodate SIMM loading, F244 or F245 buffers must be used on these signals.

5.5 Boot ROM

Boot ROM can be populated only with 85-ns flash chips, and it must have an access time of 200 ns or less. The controller supports 8-bit boot ROM at locations 0x1F00_0000 through 0x1FFF_FFFF in the system memory space. In addition to this standard boot-address range, boot ROM can also be mapped to a fault-recovery range in SIMM memory slot 4, if that slot is populated with 85-ns flash chips. The boot ROM does not support CPU cache operations, and it is not accessible from the PCI bus.

During boot ROM accesses, MDa[23:0] provide a 24-bit byte address, allowing up to 16 MB of boot ROM to be installed. The size of the boot ROM, and the boot base address, are configured at the rising edge of RST# by the state of MuxAd[0:2], as shown in Table 13.

Writes to Boot ROM must be in byte sizes. Read cycles may be in word, half-word or byte sizes. During read operation, the controller assembles four consecutive byte read cycles into words. The 8-bit boot ROM data is connected to MDb[7:0].

Table 13: boot ROM Size Configuration at Reset

MuxAd[2:0]	Boot ROM Size	Address Range
000	1 MB	0x1FC0_0000 through 0x1FCF_FFFF
001	0.5 MB	0x1FC0_0000 through 0x1FC7_FFFF
010	2 MB	0x1FC0_0000 through 0x1FDF_FFFF
011	4 MB	0x1FC0_0000 through 0x1FFF_FFFF
100	8 MB	0x1F80_0000 through 0x1FFF_FFFF
101 through 111	16 MB	0x1F00_0000 through 0x1FFF_FFFF

The controller asserts the boot ROM chip select (BROMCS#) in the address range 0x1F00_0000 through 0x1FFF_FFFF. When write cycles are performed to the boot ROM space, the controller asserts MWE# in conjunction with BROMCS#. When read cycles are performed, the controller asserts BWE# in conjunction with BROMCS#.

If the CPU attempts to accessboot ROM addresses outside the defined size of the boot ROM, the controller returns 0 with the data error bit set on SysCMD[0]. In addition, the controller's bus error status register (Section 8.1.1) is updated and an interrupt is asserted on the Int# signal, if the interrupt is enabled in the interrupt control and status register (Section 8.1.2).

5.5.1 Boot ROM Write Protection

Boot ROM can be protected in hardware and/or software. Hardware protection is implemented at boot-time. Software protection is implemented by programming the boot ROM write-protect register (Section 5.5.1.2).

5.5.1.1 Hardware Versus Software Protection

Hardware can implement write protection on up to 960 KB of the boot ROM, in blocks of 64 KB at boot time. On the rising edge of reset (RST#), four bits of the MuxAd bus, MuxAd[6:3], are sampled and used to determine the number of 64 KB blocks to be protected. Up to 15 blocks can be protected; all 0000 indicates one 64 KB block, 0001 indicates two blocks, and so on. A value of 1111 indicates that no blocks are to be protected.

For boot ROM sizes less than or equal to 4 MB, the base address of the ROM and the base address for hardware protection are both 0x1FC0_0000. For boot ROM sizes of 8 MB and 16 MB, the base addresses of ROM are 0x1F80_0000 and 0x1F00_0000, respectively, and the base address for hardware protection is 0x1FC0_0000, as shown in Table 13.

Software can also implement write protection by writing to the boot ROM write-protect register. The 6 least significant bits of this register provide additional protection for up to 63 64-KB blocks. The base address for software protection is 0x1FC0_0000 and the protected range consists of the combination of software-implemented protection and hardware-implemented protection. Software can override or re-enable hardware write protection as described in the boot ROM write-protect register's key field (Section 5.5.1.2, immediately below).

5.5.1.2
Boot ROM Write-Protect Register

The boot ROM write protect register stores a word at offset 0x5C. The register is initialized to 0xFFFF_FF3F at reset. Software can override the hardware write protection configured at reset by writing 0xC0DE_99xx to this register, with the least-significant 6 bits containing the desired software write protection value. Care must be taken not to change the protection while data is being written to the boot ROM. To prevent this, read a word from the boot ROM immediately before changing this register.

The register has the following fields:

Bits 5:0	WProt	*Write-Protect Value*. The number of 64 KB blocks, minus 1, to be write-protected (up to 63 blocks). 0 protects 1 block, all 1s disables protection.
Bits 7:6	*Reserved*	Hardwired to 0
Bits 31:8	Key	*Hardware Protection Override Key* 0xC0DE_99 = Override the hardware protection configured at reset. To re-enable hardware protection, software must write a value other than 0xC0DE_99 to this field. After re-enabling of hardware protection, the key field changes to all 1s.

5.5.2
Standard and Fault-Recovery Boot ROM Spaces

The boot ROM may be configured with two distinct address spaces: *standard* and *fault-recovery*. The standard address space is used for normal operation. The fault-recovery space is used when the standard flash ROM is corrupted or requires updating.

5.5.2.1
Standard Space

The *standard* boot ROM address space consists of a hardware-protected boot block and a software-protected address range (Section 5.5.1). Table 14 shows the boot ROM memory map during standard operation for a 1 MB ROM.

Table 14: Example Standard Mode 1MB Boot ROM Memory Map

A[31:28]	Address A[27:0]	Size	Description
1	E00_0000 to FBF_FFFF	28 MB	*Unused*
	FC0_0000 to FC0_FFFF	64 KB	Boot block, hardware-protected against writes
	FC1 0000 to FCF_FFFF	960 KB	Regular flash ROM, software-protected only
	FD0_0000 to FFF_FFFF	3 MB	*Unused*

5.5.2.2
SIMM Slot 4 Fault-Recovery Space

The *fault-recovery* boot ROM address space can be accessed when the standard flash boot ROM is corrupted or requires updating. The controller may be configured to boot from SIMM slot 4 instead of from boot ROM. This is done by driving MuxAd[8] high during reset. When the controller detects this signal high on the rising edge of RST#, the boot ROM address space (0x1F00_0000 through 0x1FFF_FFFF) is mapped into SIMM slot 4. When booting the system in this manner, the controller assumes that SIMM slot 4 is populated with an 85-ns flash SIMM. When booting from SIMM slot 4, the boot ROM will be forced into address 0x1E00_0000, 0x1E80_0000, or 0x1EC0_0000, depending on its size.

Table 15 shows the boot ROM memory map during fault-recover mode operation. Table 16 shows the required values in SIMM memory control register 4 for booting from SIMM 4. In this mode, the SIMM memory control register 4 returns the value 0x1F03_001E when read. The hardware protection for the boot ROM remains in place even when the ROM is remapped for booting from SIMM 4.

Table 15: Example Fault Recovery Mode 1 MB Boot ROM Memory Map

A[31:28]	Address A[27:0]	Size	Description
1	E00_0000 to EBF_FFFF	12 MB	*Unused*
	EC0_0000 to EC0_FFFF	64 KB	Boot block alternate address, hardware-protected against writes
	EC1 0000 to ECF_FFFF	960 KB	Regular flash ROM alternate address, software-protected only
	ED0_0000 to EFF_FFFF	3 MB	*Unused*
	F00_0000 to FFF_FFFF	16 MB	SIMM slot 4, configured for 90 ns flash, double-sided

Table 16: SIMM-Memory Control Register 4 Values for Booting From SIMM 4

Register Field	Value
Memory type	2 = rlash
Number of sides	1 = double-sided
SIMM memory enable	1 = enabled
Address multiplexing mode	1 = 10 x 10 (mux mode 1)
EDO identification mode	0 = normal mode
MDa/MDb bit 31 during EDO ID	0 = read-only
Bank interleaving	0 = non-bank-interleaved
Physical address mask	0x18 = size of 16M bytes
SIMM memory base address [a]	0x78 = base address 0x0f00_0000

a. In boot from SIMM mode, A28 must be 1 to access SIMM4.

5.6 Base Memory

The controller supports up to 16 MB of base memory. This memory, if installed, must be populated with either of the following two types of memory devices:

- 4 Mb EDO (60-ns)
- 16 Mb SDRAM (10-ns) not bank-interleaved

The controller can perform page comparisons on addresses in the base memory range, to allow pages to remain open between accesses to base memory.

5.6.1 Base Memory Control Register

The base memory control register configures base memory. The register is initialized to 0 at reset, and it must not be changed during any other type of access (CPU, DMA, or PCI bus) to base memory. If base memory is enabled, software should perform a read immediately before writing to this register, because write cycles are posted in the controller's write FIFO, and a read cycle will force the controller to write back the FIFO contents before servicing the read.

The base memory control register stores a word at offset 0x0 in the system memory space. It has the following fields:

Bits 1:0	Type	*Memory Type* 0 = EDO 1 = *Invalid* 2 = *Invalid* 3 = SDRAM (16 Mb module only)
Bit 2	*Reserved*	Hardwired to 0
Bit 3	En	*Base-Memory Enable* 1 = Enables base memory 0 = Disables base memory
Bits 5:4	MuxMode	*Address-Multiplexing Mode* 00 = Mux mode 0 01 = Mux mode 1 10 = Mux mode 2 11 = Mux mode 3 See Table 8 and Table 9 for descriptions of these modes. Only mode 3 is allowed for SDRAM.
Bit 6	In	*Bank-Interleaving* 1 = Enable bank interleaving. When bank interleaving is enabled, even-word addresses are accessed on the MDa bus, and odd-word addresses are accessed on the MDb bus. Bank interleaving can only be used when base memory is populated with EDO DRAM; it cannot be used with SDRAM. 0 = Disable bank-interleaving. When bank-interleaving is disabled, all base memory accesses are performed on the MDa bus alone.
Bit 7	PM	*Page Mode* 1 = Enable page mode. When enabled, accesses to base memory leave a memory page open (MRAS# asserted) at the end of the cycle. Page mode can only be used when base memory is populated with EDO DRAM; it cannot be used with SDRAM. 0 = Disable page mode. Accessing the same memory with address bit 28 set to 0 will cause the controller to close the page at the end of the cycle. When disabled, accesses to base memory close the memory page (MRAS# negated) at the end of the cycle.
Bits 13:8	R*eserved*	Hardwired to 0
Bits 15:14	Mask	*Physical-Address Mask*. This 2-bit mask determines the size of base memory by masking off address bits from the address comparison beginning, with bit 22. Thus, bits 23:22 of the physical address may be masked, providing an address space between 4 MB (no bits masked) and 16 MB (2 bits masked). Masks must be a pattern of left justified 1s or 0s. A 1 in the mask field indicates that the corresponding address

		bit is not masked. Address bit 28 is not used in the address comparison; it is only used when page mode is enabled. Addresses with address bit 28 cleared to 0 or set to 1 alias to one another.
Bits 21:16	*Reserved*	Hardwired to 0
Bits 27:22	BaseAdd	*Base Memory Base Address*. This 6-bit field is compared with bits 27:22 of the physical address. A match indicates that the access is to base memory. Base memory must not be overlapped with any other resource in the system.
Bits 31:28	*Reserved*	Hardwired to 0

5.6.2 Base Memory Page Mode

The controller can maintain an open memory page (MRAS# negated) within the base memory range. Page mode is enabled by the PM bit (bit 7) in the base memory control register. Page mode can only be used when base memory is populated with EDO DRAM; it cannot be used with SDRAM.

When enabled, page mode becomes active during accesses to base memory in which address bit 28 set to 1. If a page is currently open from the previous access, the controller performs an address comparison to determine if the current access is within the same page. If so, access is performed using the EDO page-hit timing (MCAS# only). If the access is not within the same page (a page miss), an MRAS# precharge is performed and a normal memory access occurs. The controller then holds the page open at the end of the cycle.

When the base memory is accessed with address bit 28 cleared to 0, page mode is not used. If a page is currently being held open, and the current access is a hit, a normal EDO page-hit cycle is performed. At the end of the cycle, the page is closed. If the access is a miss, an MRAS# precharge occurs, and the page is closed at the end of the cycle.

Because address bit 28 is not used to decode base-memory addresses, base memory is alias to two ranges 256 MB apart. No other system resources may be placed in this address range, whether or not page mode is enabled.

For bank-interleaved base memory, the page size is 512 words, or 2 KB. For non-bank-interleaved base memory, the page size is half that (i.e. 1 KB).

5.6.3 Base Memory Prefetching

When page mode is enabled for base memory (bit 7 set in the base memory control register), the controller automatically prefetches two words from base memory into a 2-word prefetch FIFO. That is, after each read, the controller prefetches two additional words into its internal prefetch FIFO. If the processor subsequently attempts a read from an address immediately following (sequential to) the address of the last read cycle, the first two words will supplied from the prefetch FIFO. Table 17 shows the words that will be placed in the prefetch FIFO following various types of base memory read cycles.

Table 17: Prefetch FIFO Contents Versus Read Sizes and Alignment

Start Address (Word Address)	One-Word Access		Two-Word Access		Four-Word Access		Eight-Word Access	
	Word(s) to CPU	Prefetched Word(s)	Word(s) to CPU	Prefetched Word(s)	Word(s) to CPU	Prefetched Word(s)	Words to CPU	Prefetched Word(s)
0	0	1	0, 1	2, 3	0, 1, 2, 3	4, 5	0, 7	8, 9
1	1	2, 3	*Not used* [a]	*Not used* [a]	*Not used* [a]	*Not used* [a]	*Not used* [a]	*Not used* [a]
2	2	3	2, 3	4, 5	2, 3, 0, 1	4, 5	*Not used* [a]	*Not used* [a]
3	3	4, 5	*Not used* [a]	*Not used* [a]	*Not used* [a]	*Not used* [a]	*Not used* [a]	*Not used* [a]
4	4	5	4, 5	6, 7	4, 5, 6, 7	8, 9	*Not used* [a]	*Not used* [a]
5	5	6, 7	*Not used* [a]	*Not used* [a]	*Not used* [a]	*Not used* [a]	*Not used* [a]	*Not used* [a]

a. Access is illegal because of the alignment

The controller compares the current SysAd address with the previous address to determine the sequential nature of the access. If sequential, the data will be available to the CPU three SysAd clocks prior than in the nonsequential case. Prefetched words are retained in the prefetch FIFO if accesses to resources other than base memory are performed between base memory accesses. Writes to base memory invalidate the prefetched words and force a sequential miss.

5.6.4 SDRAM in Base Memory

The base memory space can be populated with either 16 Mb SDRAM or 4Mb EDO modules. This section describes only the SDRAM option.

5.6.4.1 SDRAM Device Configurations

The controller supports the following 16 Mb NEC SDRAM parts and configurations in base memory. No other manufacturers' SDRAM parts are supported:

- 1M x 16 (16Mb) chips, 2 banks of 2K rows, 256 columns (NEC # µPD4516821)
- 2M x 8 (16Mb) chips, 2 banks of 2K rows, 512 columns (NEC # µPD4516421)

Table 18 shows some of the SDRAM configurations supported for base memory. Table 19 shows SDRAM configurations that are not supported in base memory (but may be supported in SIMM memory).

Table 18: Base-Memory SDRAM Configurations Supported

Memory Size	SysAd Address Bits Required	Bank A	Bank B	Sides	Bank Interleaved	SysAd[25:21] Mask Bits	NEC Part Number
4 MB	21:0	2, 1M x 16	*Empty*	Single	No	11	µPD4516821G5-A10
8 MB	22:0	2, 1M x 16	2, 1M x 16	Single	No	10	µPD4516821G5-A10
16 MB	23:0	4, 2M x 8	4, 2M x 8	Single	No	00	µPD4516421G5-A10

Table 19: Base-Memory SDRAM Configuration Not Supported

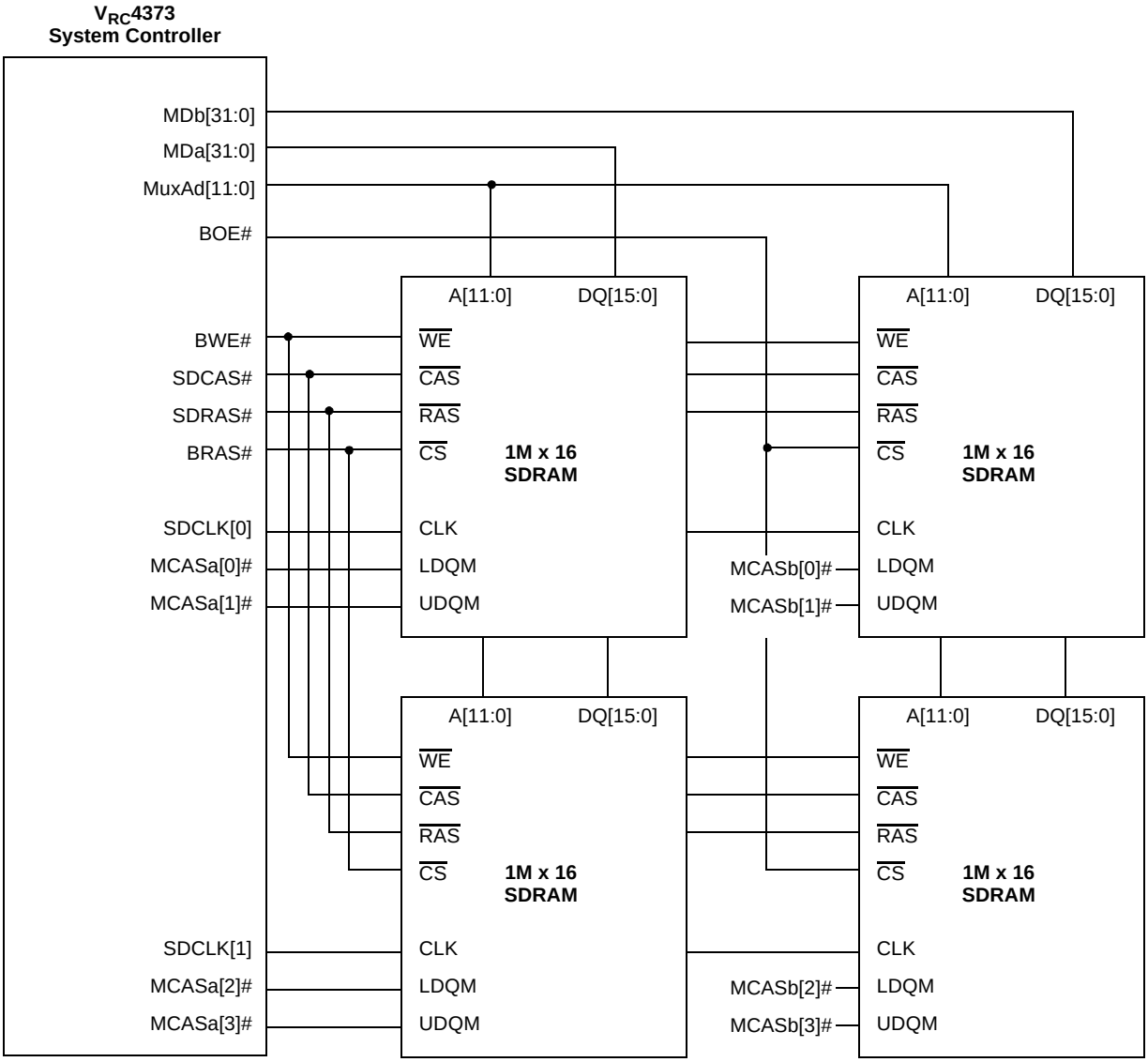
Memory Size	SysAd Address Bits Required	Bank A	Bank B	Sides	Bank-Interleaved	SysAd[25:21] Mask Bits	NEC Part Number
8 MB	22:0	4, 2M x 8	*empty*	Single	No	11	Not supported in base mem
16 MB	23:0	8, 4M x 4	*empty*	Single	No	00	Not supported in base mem
32 MB	24:0	8, 4M x 4	8, 4M x 4	Single	No	00	Not supported in base mem

5.6.4.2
SDRAM Signal Connections

Figure 4 shows how the NEC 16Mb SDRAMs are connected for base memory. Bank A uses BRAS# for chip selects, MCASa[3:0]# for the data I/O masks (DQMs), and MDa[31:0] for data. Bank B uses BOE# for chip selects, MCASb[3:0]# for the DQMs, and MDb[31:0] for data.

Both banks share the same SDRAS#, SDCAS#, and BWE# signals. Both banks must be programmed as *non-bank interleaved, non-page mode*. To do this, clear bits 6 and 7 in the base memory control register (an other values of bits 6 or 7 have unpredictable results). When programmed in this way, the two banks of base memory behave as two halves of the address range, with the highest unmasked address bit controlling bank selection.

Figure 4: SDRAM Connections for Base Memory



5.6.4.3
Bank Interleaving Versus SDRAM Burst Type and Banks

The terms *interleaved* and *bank* have multiple meanings in the context of memory design using SDRAM chips.

- *Bank Interleaving (applied to memory modules):* memory modules (whether SDRAM or DRAM) are said to be *bank-interleaved* (also called double-bank) or *noninterleaved* (also called single-bank). This concept relates to *performance* (in bank-interleaved systems, two words, one from each bank, can be transferred in a single clock cycle) as well as to the *word order* in which data is read into and written out of the modules (even-word addresses are accessed on bank A, and odd-word addresses are accessed on bank B). The system board can be designed

to support bank interleaving by connecting the MDa bus to one group of memory modules and the MDb bus to another. Software can then configure bank interleaving, although in base memory (as opposed to SIMM memory), bank interleaving cannot be implemented if SDRAM chips are present.

- *Banks (applied to memory modules and SDRAMs in different ways):* The banks referenced with respect to memory modules differ from banks inside an SDRAM. For modules, their hardwiring to the MDa or MDb bus identifies their bank. For SDRAMs, MuxAd[11] serves as the bank select for all chips on a module.
- *Burst Type (applies to SDRAM chips):* The burst type of a single SDRAM chip is programmed in the chip's mode register to be either *interleaved* or *sequential*. This concept relates only to the *word order* in which data is read into and written out of the SDRAM chip. The concept does not relate to the number of words transferred in a given clock cycle. The burst type for all SDRAM chips attached to the controller is configured during the Memory Initialization procedure (Section 5.10).

5.6.4.4
SDRAM Word Ordering

Table 20 shows the word address order for an 8-word instruction cache line fill from SDRAM. This order is determined by the SDRAM chips' burst type, which is programmed during the Memory Initialization procedure described in Section 5.10. The controller programs the burst type and word order the same for all SDRAM chips connected to it (both in the base memory and SIMM memory ranges). The term "interleaved" in this table refers to the SDRAM burst type, not the bank interleaving mode. If the burst type is *interleaved* (right-most column), A[2] selects the interleaved word and A[4:3] specifies the starting address. Burst length depends only on the access type performed by the CPU or PCI bus master; all CPU and burst lengths are supported.

Table 20: SDRAM Word Order for Instruction Cache Line Fill

Word Address A[4:2]	SDRAM Chip Burst Type [a]	
	Sequential [b]	Interleaved [c]
000	Not supported	0-1-2-3-4-5-6-7
001	Not supported	1-0-3-2-5-4-7-6
010	Not supported	2-3-0-1-6-7-4-5
011	Not supported	3-2-1-0-7-6-5-4
100	Not supported	4-5-6-7-0-1-2-3
101	Not supported	5-4-7-6-1-0-3-2
110	Not supported	6-7-4-5-2-3-0-1
111	Not supported	7-0-1-2-3-4-5-6

a. The controller assumes that all SDRAMs are initialized to the *interleaved* burst type, using a burst length of 8 words. (The *interleaved* burst type used by SDRAMs is different than the *bank interleaving* used by SIMM or DIMM modules; see Section 5.6.4.3.)

b. The controller does not support *sequential* burst type for SDRAMs. It assumes that all SDRAMs are initialized to the *interleaved* burst type, using a burst length of 8 words.

c. In the *interleaved* burst type for SDRAM chips, if the SIMM or DIMM modules carrying the chips are *bank interleaved*, address bit A[2] selects the bank, and bits A[4:2] specify the word within the bank. (The *interleaved* burst type used by SDRAMs is different than the *bank interleaving* used by SIMM or DIMM modules; see Section 5.6.4.3.)

The controller initialization any SDRAM chips in the 8-word burst length mode. Because of this, the controller always reads 8 words from SDRAM, regardless of the data width requested. However, the controller only returns the requested number of

words to the CPU or PCI bus master; any remaining words are either stored in the controller's internal base memory prefetch FIFO (Section 5.6.3) or discarded. In the case of writes, the CPU or PCI bus master can write any number of bytes, and the controller automatically issues a burst stop or precharge termination command to the SDRAM in order to store only the correct number of bytes in the SDRAM.

5.7 SIMM Memory

The controller supports four programmable address ranges for independently controlled 72-pin SIMM or DIMM modules (in this document, "SIMM" and "DIMM" are used as synonyms). SIMM slots 1 and 3 are connected to the MDa bus (bank A), and SIMM slots 2 and 4 are connected to the MDb bus (bank B). The four SIMM slots may be configured with SDRAM, EDO DRAM, fast-page DRAM, or flash memory modules. To accommodate SIMM loading, F244 or F245 buffers must be used on the controller's MuxAd signals.

5.7.1 SIMM Memory Control Registers

The configuration of each SIMM is controlled by its own SIMM memory control register. The control register for SIMM slot 1 is called SIMM memory control register 1, and so on for SIMM slots 2, 3 and 4. The registers are initialized to 0 at reset. They must not be changed during any other type of access (CPU, DMA, or PCI bus) to the SIMM memory space. If SIMM memory is enabled, software should perform a read immediately before writing to this register, because write cycles are posted in the controller's write FIFO, and a read cycle will force the controller to write back the FIFO contents before servicing the read cycle.

The controller can be configured to force the system to boot from SIMM slot 4 instead of from boot ROM. For details, and for the values in SIMM memory control register 4, see Section 5.5.2.

The four SIMM memory control registers are each 4 bytes wide, at offsets 0x4, 0x8, 0x0C, and 0x10. Each has the following fields:

Bits 1:0	Type	*Memory Type* 0 = EDO DRAM 1 = Fast-page DRAM 2 = Flash 3 = SDRAM (16 Mb or 64 Mb)
Bit 2	SD	*Number Of Sides* 1 = Two-sided. See Section 5.7.3.2 on page 28. 0 = Single-sided
Bit 3	En	*SIMM Memory Enable* 1 = Enables SIMM memory 0 = Disables SIMM memory

Bits 5:4	MuxMode	*Address Multiplexing Mode 0, 1, 2, or 3* 00 = Mux mode 0 01 = Mux mode 1 10 = Mux mode 2 11 = Mux mode 3 See Table 8 and Table 9 for descriptions of these modes. Mux mode 4 is configured by clearing bits 5:4 to 0 (as in mux mode 0), and setting bit 12.
Bit 6	ID	*EDO Identification Mode* 1 = Places the controller into the EDO identification mode, which is a special boot sequence. This bit is used in conjunction with bit 7.
Bit 7	D31	*Value Of MDa or MDb Bit 31 During EDO Identification Mode* (read-only). The state of bit 31 on the MDa or MDb bus, at the time bit 6 of this register (the ID bit) is set to 1 (in other words, while the EDO identification sequence is being performed). At all other times, the value of this bit is 0. This bit is used only in conjunction with bit 6 (the ID bit).
Bit 8	In	*Bank Interleaving* 1 = Enable. When bank interleaving is enabled, even-word addresses are accessed on the MDa bus, and odd-word addresses are accessed on the MDb bus. When this bit is set in SIMM memory control register 1, SIMM 2 is assumed to be bank-interleaved with SIMM 1. When the bit is set in SIMM memory control register 3, SIMM 4 is assumed to be bank interleaved with SIMM 3. This bit is unused in SIMM memory control registers 2 and 4, which are hardwired to 0. See Section 5.7.2. 0 = Disable. When bank interleaving is disabled, both even- and odd-word accesses occur on the MD bus to which the accessed SIMM is connected.
Bit 11:9	*Reserved*	Hardwired to 0
Bit 12	MuxMode4	*Address-Multiplexing Mode 4* 1 = Mux mode 4 (used for 64 Mb SDRAM only). 0 = Mux mode 0, 1, 2 or 3, as determined by bits 5:4, above. See Table 9 for a description of mux mode 4.
Bits 17:13	Mask	*Physical Address Mask*. This 5-bit mask determines the size of SIMM memory by masking off address bits from the address comparison beginning with bit 21. Thus, bits 25:21 of the physical address may be masked, providing an address space between 2 MB (no bits masked) and 64 MB (5 bits masked). Masks must be a pattern of left justified 1s or 0s. A 1 in the mask field indicates that the corresponding address bit is not masked.

Bits 20:18	*Reserved*	Hardwired to 0
Bits 27:21	SimmAdd	*SIMM Memory Base Address*. This 7-bit field (when appended with bits 31:28, which are always 0) is compared with the most-significant 11 bits of the physical address. A match indicates that the access is to the corresponding SIMM. Bit 28 is not used in the address compare.
Bits 31:28	*Reserved*	Hardwired to 0

5.7.2 Bank Interleaving

SIMMs may be bank interleaved to increase performance. SIMM memory control registers 1 and 3 control bank interleaving, allowing SIMM 1 to be bank interleaved with SIMM 2, and SIMM 3 to be bank interleaved with SIMM 4. When the bank-interleaving control bit (bit 8) is set in SIMM memory control register 1, *all parameters* of SIMM 2 are automatically set equal to those of SIMM 1. The address range for the entire bank-interleaved block must be programmed into SIMM memory control register 1.

For example, bank interleaving two 8-MB SIMMs requires programming a 16 MB range in the SIMM memory control register 1; SIMM memory control register 2 is not used and need not be programmed. The modules used in SIMM 1 and SIMM 2 must be equivalent. SIMM 3 may be bank interleaved with SIMM 4 in a similar manner.

When SIMMs are bank interleaved, even-word addresses correspond to the SIMM attached to MDa (bank A), and odd-word address correspond to the SIMM attached to MDb (bank B). When SIMMs are non-bank interleaved, both even- and odd-word accesses occur on the MD bus to which the accessed SIMM is connected.

5.7.3 SDRAM in SIMM Memory

The SIMM Memory address ranges can be populated with flash, SDRAM, EDO, or fast-page SIMM or DIMM modules. This section describes only the SDRAM option. The term *interleaved* has two different meanings in the context of SDRAM memory. See Section 5.6.4.3 on page 23 for details.

5.7.3.1 SDRAM Device Configurations

The controller supports the following 16 Mb and 64 Mb NEC SDRAM parts and configurations in SIMM memory. No other manufacturers' SDRAM parts are supported:

- 1M x 16 (16Mb) chips, 2 banks of 2k rows, 256 columns (NEC # μPD4516821)
- 2M x 8 (16Mb) chips, 2 banks of 2k rows, 512 columns (NEC # μPD4516421)
- 4M x 4 (16Mb) chips, 2 banks of 2k rows, 1K columns (NEC part)
- 4M x 16 (64Mb) chips, 2 banks of 8k rows, 512 columns (NEC # μPD4564821)
- 8M x 8 (64Mb) chips, 2 banks of 8k rows, 1K columns (NEC # μPD4564421)

Table 21 shows some of the SDRAM configurations supported for base memory.

Table 21: SIMM-Memory SDRAM Configuration Examples

Memory Size	SysAd Address Bits Required	Bank A	Bank B	Sides	Bank Interleaved [a]	SysAd[25:21] Mask Bits	NEC Part Number
4 MB	21:0	2, 1M x 16	*Empty*	Single	No	11110	µPD4516821G5-A10
8 MB	22:0	2, 1M x 16	*Empty*	Double	No	11100	µPD4516821G5-A10
8 MB	22:0	2, 1M x 16	2, 1M x 16	Single	Yes	11100	µPD4516821G5-A10
8 MB	22:0	4, 2M x 8	*Empty*	Single	No	11100	µPD4516421G5-A10
16 MB	23:0	2, 1M x 16	2, 1M x 16	Double	Yes	11000	µPD4516821G5-A10
16 MB	23:0	4, 2M x 8	*Empty*	Double	No	11000	µPD4516421G5-A10
16 MB	23:0	4, 2M x 8	4, 2M x 8	Single	Yes	11000	µPD4516421G5-A10
16 MB	23:0	8, 4M x 4	*Empty*	Single	No	11000	NEC part
32 MB	24:0	4, 2M x 8	4, 2M x 8	Double	Yes	10000	µPD4516421G5-A10
32 MB	24:0	8, 4M x 4	8, 4M x 4	Single	Yes	10000	NEC part

a. For bank-interleaved configurations, SIMM 1 and SIMM 3 are connected to the MCASa and MDa buses and are considered bank A; SIMM 2 and SIMM 4 are connected to the MCASb and MDb buses and are considered bank B.

5.7.3.2 Double-Sided (DIMM) Modules

If the *side* bit (bit 2) in the SIMM memory control registers 1 and 3 is set (indicating double-sided modules), the controller maps the lower half of the address range to the front side and the upper half to the back side. If the side bit is cleared (indicating single-sided modules), the controller maps the entire address range to the front side.

100-pin DIMMs are the only DIMM package supported. Each DIMM has four chip-select signals: S0, S1, S2, and S3. In *bank-interleaved* (two-bank) configurations, each module uses all four of its chip-select inputs: its front-side inputs (S0 and S2) and its back-side inputs (S1 and S3). In *non-interleaved* (single-bank) configurations, each module uses only its two front-side chip-select inputs (S0 and S2).

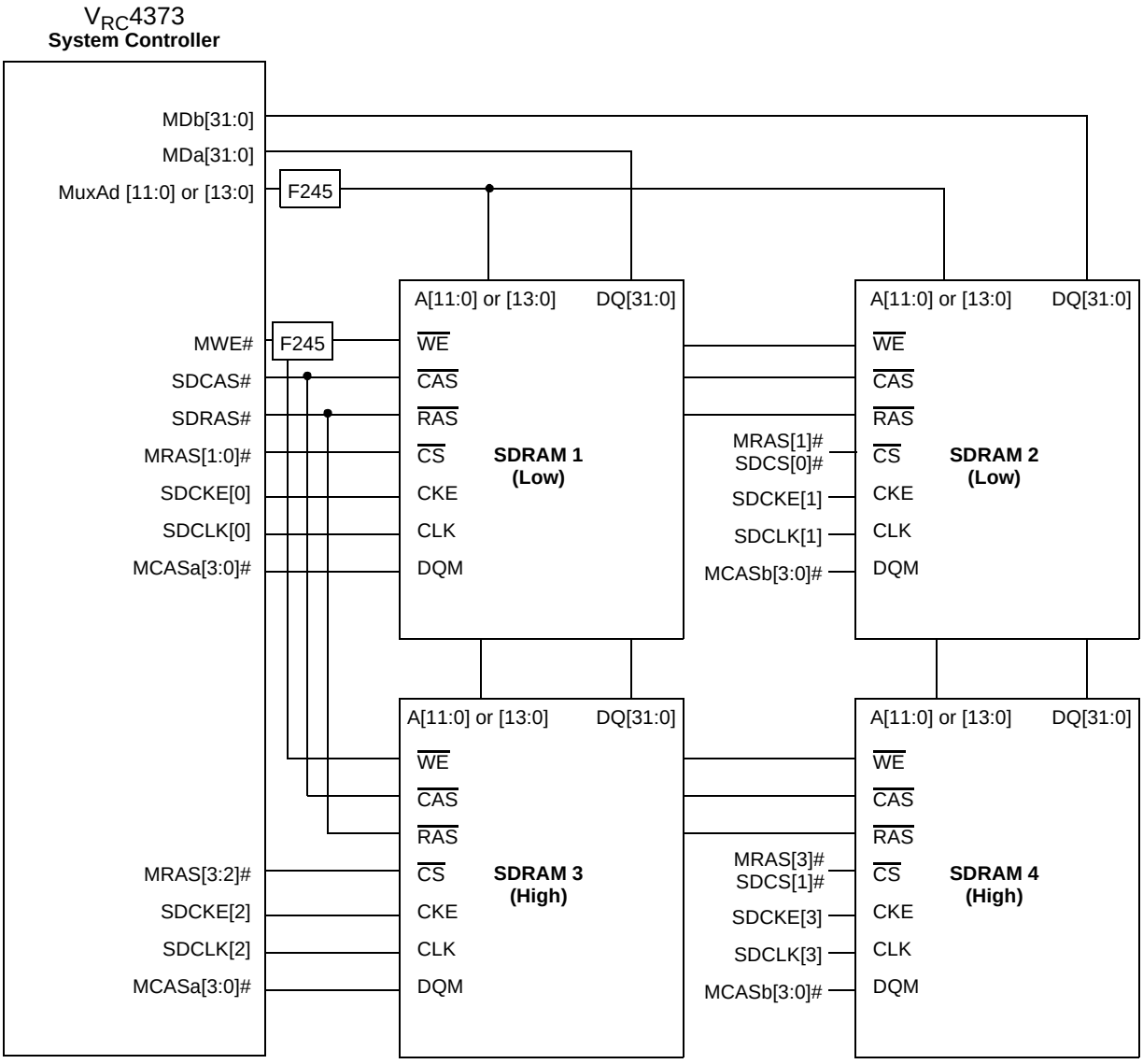
In bank-interleaved configurations, each of the controller signals used as chip-selects can be connected to a maximum of four chip-select inputs on a *single SDRAM chip* (a DIMM module typical carries between 1 and 16 such chips).

5.7.3.3 SDRAM Signal Connections

Figure 5 shows how SDRAMs SIMM or DIMM modules are connected in the SIMM memory range. This example shows six controller signals used as chip-selects: MRAS[3:0] and SDCS[1:0]. Normally eight controller signals would be used as chip-selects, as shown in Table 22 (two signals for each side of each module), but two of the signals in Figure 5 (MRAS[1] and MRAS[3]) serve two modules each, due to the limited number of signals available on the controller. MRAS[1] and MRAS[3 may need buffering. The available chip selects from the controller can be connected in any way to the modules; software need not be concerned about how these connections are made.

If the SIMM memory control registers 1 and 3 are programmed to implement *bank interleaving* (bit 8 set), SIMMs 1 and 3 contain even words (MuxAd[2] = 0), and SIMMs 2 and 4 contain odd words (MuxAd[2] = 1). Alternatively, if the SIMM memory control registers 1 and 3 are programmed to implement *non-bank-interleaved* memory, SIMM 1 and SIMM 3 contain both even and odd words, and SIMM 2 and SIMM 4 are assumed to be absent (and are not accessed).

Figure 5: SDRAM Connections for SIMM Memory



5.7.3.4
SDRAM Loads and Signals

Table 22 shows the number of device loads and signals used by various single-sided (SIMM) and double-sided (DIMM) configurations. In this table, one device load equals 8 pF.

Table 22: Loads and Signals for a Single SIMM or DIMM Module

Signal to SIMM or DIMM Module	Number of Loads [a]								Number of Signals	
	x4		x8		x16		x32			
	1 Side	2 Sides	1 Side	2 Sides	1 Side	2 Sides	1 Side	2 Sides	1 Side	2 Sides
$\overline{RAS}$	8	*N/S* [b]	4	8	2	4	1	2	1	1
$\overline{CAS}$	8	*N/S*	4	8	2	4	1	2	1	1
$\overline{WE}$	8	*N/S*	4	8	2	4	4	2	1	1
DQM	2	*N/S*	1	2	1	2	1	2	4	4
$\overline{CS}$	4	*N/S*	2	2	1	1	1	1	2 [c]	4 [c]
Data	1	*N/S*	1	2	1	2	1	2	16	16
Address	8	*N/S*	4	8	2	4	1	2	12 or 14 [d]	12 or 14 [d]
CLK	4	*N/S*	4	4	2	2	1	1	2 [e]	2 [e]
CKE	4	*N/S*	4	4	2	2	1	1	2 [f]	2 [f]

a. One device load equals 8 pF.
b. *N/S* = not supported.
c. DIMM modules have four chip-select signals (S0, S1, S2, S3). In single-bank configurations, only two are used (S0 and S2). In bank-interleaved configurations, all four are used; S0 and S2 on the front sides, S1 and S3 on the back sides. Each controller signal used for chip-select can connect to a maximum of four DIMMs. See Figure 5 for connections that use 6 chip-selects in a bank-interleaved configuration. MRAS[1] and MRAS[3] may need buffering if they serve two DIMMs, as shown in the figure.
d. 12 address bits for 16 Mb SDRAM chips, 14 address bits for 32 Mb SDRAM chips.
e. Each SIMM or DIMM module has two clock signals (CLK0 and CLK1). There can be a maximum of 8 SDRAM devices per DIMM module, in either single- or double-bank configurations. Each clock signal connects to four SDRAMs through two resistors.
f. Each SIMM or DIMM module has two clock enable signals (CKE0 and CKE1). There can be a maximum of eight SDRAM devices per DIMM module, in either single- or double-bank configurations. Each clock enable signal connects to four SDRAMs.

5.7.3.5
SDRAM Word Ordering

The word-address ordering for cache-line fills from SDRAM in the SIMM memory range is the same as the word-ordering in the base memory range. See the description in Section 5.6.4.4 on page 24.

5.8
DRAM Refresh

The controller supports $\overline{CAS}$-before-$\overline{RAS}$ (CBR) DRAM refreshing to all DRAM address ranges. The refresh clock is derived from the system clock; its rate is determined by a programmable 12-bit counter in the DRAM refresh counter register, described below.

5.8.1
DRAM Refresh Counter Register

The DRAM refresh counter register stores a word at offset 0x58. The register is initialized to 0 at reset. It has the following fields:

Bits 11:0 Cntr — *Refresh Counter Value*
The refresh counter counts down from this value, at the system clock rate. The refresh pulse is generated upon reaching 0. The reset value is 0. A value of 0x400 is equivalent to approximately 15 ms at a 66 MHz clock rate. DRAM refreshing in enabled and

disabled in the power-on memory initialization register (Section 5.10.1).

Bits 31:12 *Reserved* Hardwired to 0

5.8.2 Refresh Mechanism

The refresh logic requests access to DRAM from the internal bus-arbitration logic each time the counter reaches 0. The refresh logic can accumulate up to a maximum of 8 refresh requests while it is waiting for the bus. Once the refresh logic owns the bus, all accumulated refreshes are performed to base memory and any installed SIMMs, and no other accesses (CPU, DMA, or PCI) are allowed. Refreshes are staggered by one clock (that is, there is at least one bus clock between transitions on any pair of MRAS# signals).

Refresh automatically closes all open DRAM pages and clears the base memory prefetch FIFO. Refresh is disabled whenever the ID bit is set in any of the SIMM memory control registers, though refreshes will accumulate normally even when refresh is disabled. Accumulated refreshes are performed as soon as refreshing is re-enabled.

5.9 CPU-to-Memory Write FIFO

The controller has an 8-word CPU-to-memory write FIFO. (PCI writes to memory are buffered in the PCI target FIFO, described in Section 6.4.2.) This FIFO accepts writes at the maximum CPU speed. A single address is held for the buffered write cycle, allowing the buffering of a single write transaction. That transaction may be a word, double-word, 4-word data-cache writeback. When a word is placed in the FIFO by the CPU, the controller attempts to write the FIFO's contents to memory as quickly as possible. If the next CPU read or write cycle is addressed to memory, the controller negates EOK#, thus causing the next CPU transaction (read or write) to stall until the controller empties its FIFO. If the next CPU transaction (read or write) is addressed to a PCI bus target, the controller asserts EOK#, thus allowing the CPU transaction to complete. If, upon completion of such a CPU transaction to a PCI bus target, the controller's FIFO is still not empty, the controller will again negates EOK# to stall the next CPU write until the contents of the FIFO are written back to memory.

5.10 Memory Initialization

Memory must be initialized by software at power-on, before memory is accessed. The following sections describe the configuration register and sequence used for this initialization.

5.10.1 Power-On Memory Initialization Register

The power-on memory initialization register configures SDRAM in both the base memory and SIMM memory address ranges. The register is initialized to 0 at reset, and it must be configured before memory is accessed after power-on. The register stores a word at offset 0x78 and has the following fields:

Bit 0 Mode *SDRAM Mode Set*
1 = Enable writing to the mode registers on all SDRAM chips. When this bit is set, the controller automatically provides the data that configures all SDRAM chips on all SIMMs or DIMMs for a burst length of *8 words*, burst type of *interleaved*, and $\overline{\text{CAS}}$

		latency of *3 cycles*. These are the only SDRAM modes the controller supports. 0 = Disable writing to mode registers on SDRAMs
Bit 1	Precharge	*SDRAM Precharge* 1 = *Precharge*. Setting this bit causes the controller to issue two sequential precharge commands to any SDRAMs in the base memory and SIMM memory ranges. Do not set this bit during normal operation of the system. It should be set only during the power-on process. 0 = *No Precharge*. This bit is cleared automatically by the controller at the end of the two precharge commands.
Bit 2	Refresh	*Refresh Enable (SDRAM and DRAM)* 1 = *Refresh*. Setting this bit causes the controller to issue eight sequential automatic refresh ($\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$) commands to any SDRAMs in the base memory and SIMM memory ranges. This is required during SDRAM initialization. The refresh commands are issued only if the SDRAM memory type has been programmed in the base memory or SIMM memory ranges. This bit also enables refresh of DRAMs. 0 = *No Refresh*. (Default value at power-on or reset.) This bit is cleared automatically by the controller at the end of the eight automatic refresh commands.
Bits 31:3	R*eserved*	Hardwired to 0

5.10.2

Power-On Initialization Sequence

Follow this sequence to configure memory at power-on:

1. Program the base memory control register (see Section 5.6).
2. Program the four SIMM memory control registers, if these address ranges are used (see Section 5.7.1).
3. If SDRAM is installed in any address range, set the precharge bit (bit 1) in the power-on memory initialization register.
4. Wait for 8 CPU clocks, if SDRAM devices are used. (This is required to finish the SDRAM precharge sequence initiated in step 3.)
5. Set the refresh bit (bit 2) in the power-on memory initialization register. This enables refresh for all SDRAM and DRAM, and (if SDRAM is installed) it initiates eight sequential SDRAM refresh cycles.
6. Wait for step 5 to complete. If SDRAM is installed, this is approximately 60 CPU clocks (eight SDRAM refresh cycles).
7. If SDRAM is installed in any address range, set the mode bit (bit 0) in the power-on memory initialization register. This configures all SDRAM chips for a burst length of 8 words, a burst type of interleaved, and $\overline{\text{CAS}}$ latency of three cycles.
8. Wait for 9 CPU clocks, step 7 to complete.
9. Program the DRAM refresh counter register (Section 5.8.1).

At this point, memory is ready to use. All other configuration registers in the controller should then be programmed before commencing normal operation.

6.0 PCI Bus Interface

The controller's PCI bus interface complies with the *PCI Local Bus Specification, Revision 2.1*. Complete master and target capabilities are supported. No external logic or buffering is necessary. The interface implements 3.3V PCI-compliant pads (5V tolerant) using the NEC CMOS-9 process technology. All PCI interface electrical characteristics (loading, drive, impedance, capacitance, and so forth) comply fully with the PCI specification.

The PCI bus interface contains two separate data paths, one for CPU access and one for DMA. Each path has its own data pipeline and FIFO, and each one operates independently of the other. The FIFOs in this interface include:

- 4-word (16-byte) bidirectional PCI master FIFO (CPU is PCI bus master)
- 8-word (32-byte) bidirectional PCI target FIFO (memory is PCI bus target)
- 8-word (32-byte) bidirectional DMA FIFO (PCI-to-memory or memory-to-PCI)

6.1 PCI Bus Timing

The PC bus operates at 33 MHz and supports full burst transfers; no wait states are required with adequately fast memory. Peak PCI bus bandwidth is 133 MB/sec. The PCI bus is synchronized to the SysAD bus, with the SysAD bus clock running at two times the PCI clock.

6.2 PCI Commands Supported

Table 23 summarizes the PCI command codes supported, and not supported, by the controller as master and target.

Table 23: PCI Commands

CBEn[3:0]	Command	As a master	As a target
0000	Interrupt acknowledge	No	*Ignored*
0001	Special cycle	No	*Ignored*
0010	I/O read	Yes, via PCI master I/O window (see Section 6.3)	Yes; must be in add-on board mode and hit PCI I/O base address range.
0011	I/O write	Yes, via PCI master I/O window (see Section 6.3)	Yes; must be in add-on board mode and hit PCI I/O base address range.
010x	*Reserved*	—	*Ignored*
0110	Memory read	Yes, via PCI master address windows (see Section 6.3)	Yes; must hit a PCI target address window (see Section 6.4)
0111	Memory write	Yes, via PCI master address windows (see Section 6.3)	Yes; must hit a PCI target address window (see Section 6.4)
100x	*Reserved*	—	*Ignored*
1010	Configuration read	Yes, via PCI configuration registers (see Section 6.5)	Yes, via PCI configuration registers (see Section 6.5)
1011	Configuration write	Yes, via PCI configuration registers (see Section 6.5)	Yes, via PCI configuration registers (see Section 6.5)
1100	Memory read multiple	No	Aliased to memory read

Table 23: PCI Commands (Continued)

CBEn[3:0]	Command	As a master	As a target
1101	Dual address cycle	No	*Ignored*
1110	Memory read line	No	Aliased to memory read
1111	Memory write and invalidate	No	Aliased to memory erite

6.3 PCI Master Transactions (CPU-to-PCI Bus)

The controller supports bidirectional data transfers between the CPU and PCI bus targets by becoming a PCI bus master. The CPU obtains access to PCI bus resources (summarized in Table 6 on page 7) by accessing a local physical address that corresponds to one of three PCI address windows:

- *PCI master address window 1*
- *PCI master address window 2*
- *PCI master I/O window*

These registers are at offsets 14, 18, and 24, respectively (see Table 5). They are configured through the PCI master address window registers, described below.

6.3.1 PCI Master Window Registers

The three PCI master window registers described above all have the same structure. They are initialized to 0 at reset. A register must not be changed while a write is posted to the PCI bus. There must be at least two CPU clocks between writing to such a register and performing a PCI access through the window mapped by the register.

Each of the three PCI master window registers contain the following fields:

Bits 7:0	PCIAdd	*PCI Address.* This 8-bit field replaces the most-significant 8 bits of the address defined in the LAdd field when the address is transmitted to the PCI bus. Bits masked by the mask field (bits 19:13) are directly transferred from the CPU's SysAD bus (rather than from the PCIAdd field).
Bits 11:8	*Reserved*	Hardwired to 0
Bits 12	E	*Enable* 1 = Enable access to the PCI bus through the address window specified in this register. 0 = Disable access
Bits 19:13	Mask	*Physical Address Mask*. This mask is used to determine the size of the PCI window. It masks 7 address bits from the address comparison, beginning with bit 24. Thus, bits 30-24 may be masked, providing an address block size between 16 MB (no bits masked) and 2 GB (7 bits masked). A 0 in a mask bit indicates that the corresponding address bit is masked.
Bits 23:20	*Reserved*	Hardwired to 0

Bits 31:24 LAdd	*Local Base Address*. This 8-bit field is compared with the most-significant 8 bits of the physical CPU address, conditioned on the mask field. A match indicates that the access is to the PCI bus. LAdd should not be programmed to overlap PCI space with local resources (memory, registers or boot ROM) or PCI target windows; this will result in improper operation.

6.3.2
PCI Master Transaction Details

Transfers between the CPU and PCI bus are buffered through a 4-word bidirectional *PCI master FIFO*. This FIFO stores data and latches the address and byte enables for one CPU-to-PCI read or write transaction. When the CPU accesses an address in the window defined by the LAdd fields (bits 31:24) of either of the two PCI master address window registers or the PCI master I/O window, the data is transferred to and from the PCI bus through the FIFO. The FIFO improves performance and provides a mechanism for resolving deadlocks between the PCI and SysAD buses.

The FIFO size of 4 words allows the CPU to perform all possible write transactions. All CPU read transactions are supported, except instruction cache line fills (8-word burst transfers).

For data cache line fills from the PCI bus, the controller reads 4 words from the PCI bus, beginning with the first word in the line (word address = 0), and returns them to the CPU in the correct sub-block order. For example, a data cache line fill from address 2 is read from the PCI bus as 4 words, beginning at address 0 (0, 1, 2, 3) and returned to the CPU beginning at address 2 (2, 3, 0, 1). The controller does not support the PCI cache line wrap mode.

During CPU-to-PCI bus transactions, the FIFO accepts write data at the CPU rate. If the CPU is performing a data-cache writeback, a burst of 4 words occurs. The address and byte-enables for the cycle are first latched in the FIFO. Then the data words are placed in the FIFO, and the PCI bus is requested. If the CPU attempts another PCI write before the FIFO is empty, the controller stalls the CPU write. If the PCI bus has not yet been acquired before the FIFO is filled, the controller indicates to the CPU that further write (and read) cycles will be stalled by negating the EOK# signal to the processor. Write cycles to resources other than the PCI bus are allowed to complete after the controller decodes the address.

The controller uses the PCI block-transfer protocol if the CPU read is also more than one word. During block reads, the CPU is stalled by the controller until a word has been placed in the FIFO from the PCI bus. For CPU accesses to the PCI bus of less than one word, the controller reads or writes the correct number of bytes.

Until the controller is granted the PCI bus, another PCI master may have ownership of the PCI bus and may request access to the controller as a PCI target. A PCI target FIFO in the controller allows such an access to occur without causing deadlock, as described in the next section.

6.4 PCI Target Transactions (PCI-to-Memory)

The controller supports bidirectional data transfers between a PCI bus master and the controller's memory, as target. The PCI bus master obtains access to the controller's memory by accessing a local physical address that corresponds to one of two PCI address windows:

- *PCI target address window 1*
- *PCI target address window 2*

These registers are at offsets 1C and 20, respectively (see Table 5). They are configured through the PCI master address window registers, described immediately below.

6.4.1 PCI Target Window Registers

The two PCI target address window registers described above are initialized to 0 at reset. There must be at least two CPU clocks between writing to such a register and performing a PCI access through the window mapped by the register.

Each of the two PCI target address window registers contain the following fields:

Bits 10:0	LAdd	*Local Address*. This 11-bit field replaces the most-significant 11 bits of the PCI address defined in the PCIAdd field (bits 31:24), when the address is transmitted to the memory. Bits masked by the mask field are directly transferred from the PCI bus, rather than from the LAdd field.
Bits 11	*Reserved*	Hardwired to 0
Bits 12	E	*Enable* 1 = Enable the PCI bus to access local resources through address window specified in this register. 0 = Disable access.
Bits 19:13	Mask	*PCI Address Mask*. This mask is used to determine the size of the local window. It will mask 7 address bits from the address comparison, beginning with bit 21. Thus, bits 27-21 may be masked, providing an address block size between 2 MB (no bits masked) and 256 MB (7 bits masked). A 0 in a mask bit indicates that the corresponding address bit is masked.
Bits 20	*Reserved*	Hardwired to 0
Bits 31:21	PCIAdd	*PCI Address*. This 11-bit field is compared with the most-significant 11 bits of the PCI address, conditioned on the Mask field. A match indicates that the access is to the controller. Care must be taken not to overlap the two PCI target windows.

6.4.2 PCI-to-CPU Transactions Details

When the controller sees an address on the PCI bus that falls within one of its two PCI target address window ranges, it responds by requesting access to its attached memory. This prevents the CPU from obtaining access to the memory. The controller supports full-speed burst read and write cycles from by a PCI master. Accesses are

performed through an 8-word bidirectional *PCI target FIFO*. This FIFO stores data and latches the address and byte-enables for one PCI-to-CPU read or write transaction. PCI target transfers are performed with higher priority than PCI master transfers. Thus, if both CPU requests and PCI target requests for memory are present simultaneously, the PCI-target transfer occurs first.

During PCI target read cycles, the controller always accesses memory in 4-word reads, using the data cache miss protocol. These words are placed in the target FIFO and sent to the PCI bus at maximum speed. If the PCI read address is not aligned to a cache-line boundary, the controller stores only the required words in the target FIFO. When the PCI word address is 2 or 3, the controller transfers the word(s) and then does a target disconnect; the controller always disconnects if there are less than 2 words left in the FIFO for a PCI read cycle. The controller uses the CPU's sub-block ordering for PCI target read cycles. Table 24 shows the read order for various access quantities.

Table 24: PCI-Target Read Order and Buffering

PCI Word Address	Words Placed in PCI Target FIFO
0	0 1 2 3
1	1 2 3
2	2 3
3	3

If the controller detects bad parity on a PCI target *address cycle*, the controller reports the error in the PCI header, generates an interrupt on INTA# (if enabled), and performs the access (that is, ignores the parity error). If the controller detects bad parity on a PCI target *data cycle*, the controller reports the error in the PCI header, generates an interrupt on INTA# (if enabled), and performs the write.

6.5 PCI Configuration Space

The controller provides a PCI configuration space, as described in the "PCI Local Bus Specification," Section 6. This space supports bus master configuration cycles of PCI devices using a mechanism similar to configuration mechanism #1 ("PCI Local Bus Specification," Section 3.7.4.1).

Two 1-word registers are provided for software to perform configuration cycles:

- *PCI configuration data register*
- *PCI configuration address register*

These registers are at offsets 28 and 2C, respectively (see Table 5), and are cleared to 0 at reset. To perform a configuration cycle, the CPU first writes an address to the PCI configuration address register, and then writes the transaction data to the PCI configuration data register. The access to the data register causes the cycle to begin. Byte enables, read/write state, and the full 32-bit address is passed through to the PCI bus, without mapping. The CPU is stalled during read cycles until the PCI configuration cycle completes. This mechanism precludes the CPU from performing bursts to configuration space.

The PCI configuration address register format contains a register number field that is used to address specific PCI bus targets. Figure 3-20 of the "PCI Local Bus Specifica-

tion" shows the format. Each configurable target on the PCI bus maintains a set of registers, called the PCI configuration space registers, that consist of header registers and device-dependent registers, as defined in Section 6.1 of the "PCI Local Bus Specification."

The controller implements two sets of the PCI configuration space registers, depending on its mode of operation:

- *Host Bridge Mode:* In this mode, the controller is located on the mother board and acts as the PCI host bridge for the system. The PCI configuration space registers for this mode are described in Section 6.6.
- *Add-on Board Mode:* In this mode, the controller is located on a PCI board, rather than on the mother board. The PCI configuration space registers for this mode are described in Section 6.7.

6.6 PCI Configuration Registers (Host Bridge Mode)

The CPU uses this configuration space during system boot to configure the controller. In the host bridge mode, the CPU accesses the controller directly through the controller's registers in the CPU's memory space (Table 5); the PCI configuration address register and PCI configuration data register are not used.

Table 25 shows the controller's PCI configuration space registers for this mode. The alternating pattern of shading and no shading of rows in this table defines 4-byte word boundaries; some registers can be accessed on byte or 2-byte boundaries. The sections that follow define the fields in each register.

After changing any of these registers, at least two CPU clocks must elapse before the start of a PCI access, either by the CPU, DMA, or an external master.

Table 25: PCI Configuration Space Registers (Host Bridge Mode) [a]

Offset from Base 0F00_0000	Size (bytes)	Register Name	Symbol	CPU R/W	Reset Value	Description	Reference
0x100	2	Vendor ID	VID	R	0x1033	Vendor ID for NEC	Section 6.6.1 on page 40
0x102	2	Device ID	DID	R	0x005B	VRC4373 controller's device ID, assigned by NEC	Section 6.6.1 on page 40
0x104	2	Command	PCICMD	R/W	0x0	Provides coarse control of PCI interface	Section 6.6.2 on page 40
0x106	2	Status	PCISTS	R/WC [b]	0x0280	Status for PCI events	Section 6.6.2 on page 40
0x108	1	Revision ID	RID	R	0x0	Device revision	Section 6.6.3 on page 42
0x109	3	Class code	CLASS	R	0x06_0000	Device type	Section 6.6.3 on page 42
0x10C	1	Cache-line size	CLSIZ	R	0x04	System cache line size (words)	Section 6.6.4 on page 42
0x10D	1	Latency timer	MLTIM	R/W	0x0	Value of latency timer for this master, in PCI clocks	Section 6.6.4 on page 42
0x10E	2	*Reserved*		R	0x0		
0x110	4	Mailbox base address	MBADD	R/W	0x0	Base address for both mailboxes.	Section 6.6.5 on page 42
0x13B - 0x114		*Reserved*		R	0x0		
0x13C	1	Interrupt line	INTLIN	R	0x0	PCI interrupt signal	Section 6.6.6 on page 43
0x13D	1	Interrupt pin	INTPIN	R	0x0	PCI interrupt pin	Section 6.6.6 on page 43
0x13F - 0x13E	2	*Reserved*		R	0x0		

Table 25: PCI Configuration Space Registers (Host Bridge Mode) [a] (Continued)

Offset from Base 0F00_0000	Size (bytes)	Register Name	Symbol	CPU R/W	Reset Value	Description	Reference
0x140	1	*Reserved*		R	0x0		
0x141	1	Retry value	RTYVAL	R/W	0x0	Number of PCI bus retries the controller performs before giving up	Section 6.6.7 on page 43
0x142	2	PCI arbiter priority control and take away grant	PAPC	R/W	0x0	Priority scheme used in granting access to PCI bus	Section 6.6.7 on page 43
0x1FF - 0x144		*Reserved*		R	0x0		

a. The alternating pattern of shading and no shading of rows in this table defines 4-byte word boundaries.

b. Writing to this register is special. The bits can only be set by the controller hardware, and they are cleared by writing a 1 to them; writing a 0 leaves them unaffected. For example. writing 0x8000 clears the most significant bit.

6.6.1 Vendor and Device ID

The vendor and device ID registers are read-only. Together, they constitute a word at offset 0x100.

6.6.1.1 Vendor ID

The 2-byte vendor ID register is read-only and can be accessed at offset 0x100:

Bits 15:0 VID — *Vendor ID*. Hardwired to 0x1033 for NEC.

6.6.1.2 Device ID

The 2-byte vendor ID register is read-only and can be accessed at offset 0x102:

Bits 31:16 DID — *Device ID*. Hardwired to 0x0021 for the VRC4373 controller.

6.6.2 Command and Status

The command and status registers, plus reserved fields, constitute a word at offset 0x104.

6.6.2.1 Command

The 2-byte command register is read/write and can be accessed at offset 0x104:

Bit 0 IOEN — *I/O Space Enable*. Cleared to 0 at reset. Software must set it to 1 to enable access to the PCI interrupt register status in the add-on board mode.

Bit 1 MEMEN — *Memory Space Enable*. Cleared to 0 at reset. Software must set it to 1 to allow the controller to respond to memory space accesses.

Bit 2 BMAS — *Bus Master Enable*. Cleared to 0 at reset. Software must set it to 1 to allow the controller to generate PCI accesses.

Bit 3 SPC — *Special Cycles Enable*. Hardwired to 0. The controller ignores special cycles.

Bit 4 MWI — *Memory Write And Invalidate Enable*. Hardwired to 0. The controller does not generate MWI commands.

Bit 5	VGA	Hardwired to 0. The VRC4373 controller is not a VGA device.
Bit 6	PER	*Parity Error (PERR#) Enable* 1 = Enable 0 = Disable
Bit 7	WAIT_CTL	Wait cycle control. Hardwired to 0. The controller never does address stepping.
Bit 8	SERR_EN	*System Error (SERR#) Enable* 1 = Enable 0 = Disable
Bit 9	FBBE	*Fast Back-to-Back Enable*. Hardwired to 0. The controller never generates back-to-back transactions.

The 1-byte location 0x105 is reserved:

Bits 15:10	*Reserved*	Hardwired to 0

6.6.2.2
Status

The 2-byte status register can be accessed at offset 0x106. It uses a special read/write protocol: the bits can be set only by the controller hardware, but they can be cleared by writing a 1 to them; writing a 0 leaves them unaffected. For example. writing 0x8000 clears the most significant bit.

Bits 20:16	*Reserved*	Hardwired to 0
Bit 21	66MHz	*66 MHz Capable*. Hardwired to 0. The controller is a 33 MHz device.
Bit 22	UDF	*User-Definable Configuration (UDF) Support*. Hardwired to 0. The controller doesn't support UDF.
Bit 23	FBBC	*Fast Back-to-Back Capable*. Hardwired to 1. The controller will accept fast back-to-back accesses.
Bit 24	DPR	*Data Parity Reported* 1 = Enable 0 = Disable
Bits 26:25	DEVSEL	*Device Select (DEVSEL) Timing*. Hardwired to 01 (medium response).
Bit 27	STA	*Signaled Target Abort*. Set to 1 if the controller signals a target abort. Otherwise, cleared to 0.
Bit 28	RTA	*Received Target Abort*. Set to 1 whenever the master receives a target abort. Otherwise, cleared to 0.
Bit 29	RMA	*Received Master Abort*. Set whenever the master receives a master abort. Otherwise, cleared to 0.
Bit 30	SSE	*Signaled System Error* 1 = Generate a bus error interrupt 0 = No bus error

Bit 31	DPE	*Detected Parity Error*. Set when the controller detects a parity error. Otherwise, cleared to 0.

6.6.3 Revision ID and Class Code

The revision ID and class code registers are read-only. Together, they constitute a word at offset 0x108. They have the following fields:

6.6.3.1 Revision ID

The 1-byte revision ID register is read-only and can be accessed at offset 0x108:

Bits 7:0	RID	*Revision ID*. Hardwired to 0, indicating a gate array.

6.6.3.2 Class Code

The 3-byte revision ID register is read-only and can be accessed at offset 0x109:

Bits 15:8	Prog	*Programming Interface*. Hardwired to 0.
Bits 23:16	SubCl	*Subclass*. Hardwired to 0.
Bits 31:24	BaseCl	*Base Class*. Hardwired to 0x06 to indicate a bridge device.

6.6.4 Cache Line Size and Latency Timer

The cache line size and latency timer registers are both 1-byte wide, followed by two reserved bytes. Together, these locations constitute a word at offset 0x10C.

6.6.4.1 Cache Line Size

The 1-byte cache-line size register is read-only and can be accessed at offset 0x10C:

Bits 7:0	CLSIZ	*Cache Line Size*. Hardwired to 0x04, indicating four 32-bit words in a cache line

6.6.4.2 Latency Timer

The 1-byte latency timer register is read/write and can be accessed at offset 0x10D:

Bits 10:8	MLTIM	*Master Latency Time* (low 3 bits). Hardwired to 0.
Bits 15:11	MLTIM	*Master Latency Time*. See the “PCI Local Bus Specification,” Section 3.4.4.1 and 6.2.4.

The high 2 bytes in the word starting at offset 0x10E are reserved:

Bits 23:16	*Reserved*	Hardwired to 0
Bits 31:24	*Reserved*	Hardwired to 0

6.6.5 Mailbox Base Addresses

The 1-word, read/write mailbox base addresses register is accessed at offset 0x110 in the PCI configuration space header. This register must not be changed while an external agent is accessing one of the PCI mailboxes.

Bits 10:0	*Reserved*	Hardwired to 0, to indicate that the controller’s PCI mailbox registers are located in a 32-bit memory space on a 4 KB boundary and are not prefetchable.

Bits 11	MBNUM	*Mailbox Number*. Selects the mailbox number: 1 = PCI mailbox register 2 0 = PCI mailbox register 1
Bits 31:12	MBADD	*Mailbox Base Address*. Used to map the controller's two mailboxes into the PCI memory space, on a 4 KB boundary.

6.6.6 Interrupt Line and Interrupt Pin

The interrupt line and interrupt pin registers, together, constitute a word at offset 0x13C.

6.6.6.1 Interrupt Line

The 1-byte interrupt line register is read/write and can be accessed at offset 0x13C:

Bits 7:0	INTLIN	*PCI Interrupt Line Register*. This field should be written by power-on self-test software to indicate which system interrupt controller input is connected to the INTA#I signal.

6.6.6.2 Interrupt Pin

The 1-byte interrupt pin register is read-only and can be accessed at offset 0x13D:

Bits 15:8	INTPIN	*PCI Interrupt Pin Register*. Reset to 1, indicating that INTA# is the controller's PCI interrupt signal.

The two high bytes in the word starting at offset 0x140 are reserved:

Bits 31:16	*Reserved*	Hardwired to 0

6.6.7 Retry Value and PCI Arbiter Priority Control

The retry value and PCI arbiter priority control registers are read/write. Together, these locations constitute the word at offset 0x140. They have the following fields:

Bits 7:0	*Reserved*	Hardwired to 0

6.6.7.1 Retry Value

The 1-byte retry value register can be accessed at offset 0x141:

Bits 15:8	RTYVAL	*Retry Value*. The number of retries the controller should attempt before giving up on a PCI transaction. The actual retry count is readable in the PCI retry counter (Section 6.8).
Bits 23:16	*Reserved*	Hardwired to 0

6.6.7.2 PCI Arbiter Priority Control

The 2-byte PCI arbiter priority control register can be accessed at offset 0x142:

Bits 25:24 PAPC — *PCI Arbiter Priority Control*

00 = *Rotating Fair*. (This is the reset value.) In this scheme, the priority of each requestor changes, in round-robin fashion, after every request to give every request a fair chance to acquire the bus. The rotation sequence begins with the controller's internal request, followed by requestors 0,1,2,3 and back to an internal request. If any of the requestors is not active, the next requestor in the sequence becomes the highest priority. After a requestor has been granted the bus, it retains the bus, dependent on the TKYGNT bit.

01 = *Rotating Alternate 0*. In this scheme, the arbiter treats REQ0# in a special way. REQ0# is granted the bus every other transaction, if asserted. The rotation sequence is 0, i, 0, 1, 0, 2, 0, 3, 0, i ..., where i is the controller's internal request. After a requestor is granted the bus, it retains the bus, dependent on the TKYGNT bit.

10 = *Rotating Alternate 1*. This scheme is identical to the Rotating Alternate 0 scheme, except that the controller's internal request, rather than REQ0#, has the advantage. The rotation sequence is i, 0, i, 1, i, 2, i, 3, i, 0 ... After a requestor is granted the bus, it retains the bus, dependent on the TKYGNT bit.

Bit 25 TKYGNT — *Take Away Grant*

0 = When REQx# is granted, it remains granted until the REQx# is negated. This is the reset value.

1 = When REQx# is granted, the bus loses GNTx# if a higher-priority requestor requests. A rotating priority scheme is used, so all requests are at a higher priority.

The high byte in the word starting at offset 0x140 is reserved:

Bits 31:27 *Reserved* — Hardwired to 0

6.7 PCI Configuration Registers (Add-On Board Mode)

Table 26 shows the controller's PCI configuration space registers when the controller is used in the add-on board mode, that is, when the controller is located on a PCI bus add-on board, rather than on the system mother board. Compare Table 26 with Table 25 on page 39. Five additional registers are defined in the add-on board mode, as described in the sections immediately following Table 26.

Table 26: PCI Configuration Space Registers (Add-On Board Mode) [a]

Offset from Base 0F00_0000	Size (bytes)	Register Name	Symbol	CPU Bus R/W [b]	PCI bus R/W	Reset Value	Description	Reference
0x100	2	Vendor ID	VID	R	R	0x1033	Vendor ID for NEC	Section 6.6.1 on page 40
0x102	2	Device ID	DID	R	R	0x005B	VRC4373 controller's device ID, assigned by NEC	Section 6.6.1 on page 40
0x104	2	Command	PCICMD	R/W [c]	R/W [c]	0x0	Provides coarse control of PCI interface	Section 6.6.4 on page 42
0x106	2	Status	PCISTS	R/WC [d]	R/WC [d]	0x0280	Status for PCI events	Section 6.6.2 on page 40
0x108	1	Revision ID	RID	R	R	0x0	Device revision	Section 6.6.3 on page 42
0x109	3	Class code	CLASS	R	R	0x06_0000	Device type	Section 6.6.3 on page 42
0x10C	1	Cache-line sze	CLSIZ	—	R	0x04	System cache-line size (words)	Section 6.6.4 on page 42
0x10D	1	Latency timer	MLTIM	R/W[c]	R/W[c]	0x0	Value of latency timer for this master, in PCI clocks	Section 6.6.4 on page 42
0x10E	2	*Reserved*		—	R	0x0		
0x110	4	Mailbox base address	MBADD	R/W	R/W	0x0	Base address for both mailboxes	Section 6.6.5 on page 42
0x114	4	Base address register 1	BAR1	—	R/W	0x0	Base address register 1, for target memory	Section on page 44
0x118	4	Base address register 2	BAR2	—	R/W	0x0	Base address register 2, for target memory	Section on page 44
0x11C	4	Base address register 3	BAR3	—	R/W	0x0	Base address register 3, for add-on board interrupt Register	Section 6.7.2 on page 46
0x120	4	Base address register 4	BAR4	—	R/W	0x0	Base address register 4, for add-on board interrupt Register	Section 6.7.3 on page 46
0x138 - 0x124		*Reserved*		—	—	0x0		
0x13C	1	Interrupt line	INTLIN	—	R	0x0	PCI interrupt signal	Section 6.6.6 on page 43
0x13D	1	Interrupt pin	INTPIN	—	R	0x0	PCI interrupt pin	Section 6.6.6 on page 43
0x13F - 0x13E	2	*Reserved*		—	R	0x0		
0x140	1	*Reserved*		—	R	0x0		
0x141	1	Retry value	RTYVAL	—	R	0x0	Number of PCI bus retries the controller performs before giving up	Section 6.6.7 on page 43
0x142	2	PCI arbiter priority control and take away grant	PAPC	—	R	0x0	Priority scheme used in granting access to PCI bus	Section 6.6.7 on page 43
0x1FF - 0x144		*Reserved*		—	R	0x0		

a. The alternating pattern of shading and no shading of rows in this table defines 4-byte word boundaries.
b. — means not used.
c. Writable by CPU if bit 1 of the PCI enable register is set to 1. Writable by PCI host if this bit = 0. See Section 6.10 on page 47.
d. Writing to this register is special. The bits can only be set by the controller hardware, and they are cleared by writing a 1 to them; writing a 0 leaves them unaffected. For example, writing 0x8000 clears the most significant bit.

6.7.1 BAR1 and BAR2 Registers

The BAR1 and BAR2 registers are shown in Table 26. In the add-on board mode, after the controller's CONFIG_DONE bit in the PCI enable register is set to 1, a PCI master

can program the BAR1 and BAR2 registers, and the controller will use them for the PCI target address window address ranges.

Both registers are initialized to 0 at reset.

Bits 20:0	Pref	*Prefetchable*. Hardwired to 0x8, indicating prefetchable, relocatable memory (see PCI Specification, Section 6.2.5.1). This field is not used in the target address window address comparison.
Bits 31:21	Base	*PCI Base Address*. This field is compared with the most-significant 11 bits of the PCI address, after masking bits 27-21 of this field with the corresponding PCI target address window register mask (bits 19-13 of PCI target address window register 1 for BAR1, or PCI target address window register 2 for BAR2). The memory range can vary between 2 MB (no bits masked) and 256 MB (all bits masked). A match indicates the access is to the controller. If the address is all 0s, this register is treated as disabled and memory is not allocated.

6.7.2 BAR3 Register

The BAR3 register is shown in Table 26. It contains the I/O address of the add-on board interrupt register.

Bits 1:0	Space	*Space Indicator*. Hardwired to 01, indicating that the address is to I/O space.
Bits 31:2	IOAddr	*I/O Address*. The I/O address of the add-on board interrupt register (see Section 6.7.4).

6.7.3 BAR4 Register

The BAR4 register is shown in Table 26. It contains the memory-mapped address of the add-on boardinterrupt register.

Bits 3:0	Space	*Space Indicator*. Hardwired to 0, indicating that the address is to memory space.
Bits 31:4	MAddr	*Memory-Mapped Address*. The mapped-memory address for the BAR3 I/O address.

6.7.4 Add-on Board Interrupt Register

The add-on board interrupt register is located at the address specified in the BAR3 Register (see Section 6.7.2). The register specifies the state of the INTA# signal. It contains only a single bit:

Bit 0	PCI_INT	*PCI Interrupt Pending* 1 = pending PCI interrupt on INTA# 0 = no pending PCI interrupt on INTA#
Bits 31:1	*Reserved*	Hardwired to 0

When the CPU reads the SET_PCI_INT bit (bit 3) of the PCI enable register (Section 6.10), the controller returns the value of bit 0 in the add-on board interrupt register, which is the state of the INTA# signal. A PCI master causes the controller to negate

INTA# by writing any value (1 or 0) to the PCI_INT bit (bit 0) of the add-on board interrupt register.

6.8 PCI Retry Counter

The PCI retry counter is a read-only word at offset 0x70. It has only one status field:

Bits 4:0	RTRYCNT	*Retry Count*. The number of PCI bus transactions that the controller has retried. The maximum value for retries is set in the retry value register (Section 6.6.7.1 on page 43).
Bits 31:5	*Reserved*	Hardwired to 0

6.9 PCI Arbiter

The controller has a PCI bus arbiter that arbitrates access by the controller and four other PCI bus masters. Four request/grant signal pairs are provides for the other masters; the controller has a fifth, internal request/grant function for its own requests. The arbiter implements three priority schemes, which are programmable in the PCI arbiter priority control register (PAPC), described in Section 6.6.7.2 on page 44.

6.10 PCI Enable Register

The PCI enable register is accessed at base-address offset 0x74h, as shown in Table 5. It is used to enable the PCI arbitrator, enable the add-on board mode, specify the completion of controller configuration, and setting and clearing interrupts. The register contents are initialized to 0 at reset.

Bit 0	ARB_ENABLE	*Enable Arbitrator*. Enables the built-in PCI arbiter.
Bit 1	ADD_ON_BOARD	*Enable add-on board Mode*. In this mode, the VRC4373 controller is located on a PCI add-on board, rather than on the mother board.
Bit 2	CONFIG_DONE	*PCI Configuration Done*. Software should set this to 1 after configuring the controller's other PCI registers. When set to 1, the controller responds normally to PCI operations. When cleared to 0, the controller responds to PCI target cycles with retry.
Bit 3	SET_PCI_INT	*Assert PCI Interrupt*. Used only in add-on board mode. When set to 1 by the CPU, the controller sets bit 0 of the add-on board interrupt register (Section 6.7.4) to 1 and asserts the PCI interrupt signal (INTA#). When the CPU reads SET_PCI_INT, the controller returns the value of bit 0 in the add-on board interrupt register, which is the state of INTA#l. The controller automatically clears SET_PCI_INT to 0 one clock after software sets it to 1, so there is no need for software to clear it. A PCI master causes the controller to negate INTA# by writing any value (1 or 0) to bit 0 of the add-on board interrupt register.
Bit 4	RST_NMI	*Negate NMI#*. Used only in the add-on board mode. When set to 1, the controller negates its NMI# signal.

		The controller asserts NMI# whenever it detects SERR# (PCI system error) asserted. The NMI# service routine can read this bit to determine its state, or set it to 1, which clears the interrupt. The controller automatically clears the bit to 0 one clock after software sets it to 1.
Bit 5 - 31	Reserved	Hardwired to 0

6.11 PCI Mailbox Registers

The controller has two PCI mailbox registers for passing messages between the CPU and PCI bus masters:

- *PCI mailbox register 1*
- *PCI mailbox register 2*

Both registers are 1-word wide and may be read and written by either the CPU or a PCI bus master. From the CPU side, the addresses of the PCI mailbox registers are at offsets 30 and 34, respectively (see Table 5). From the PCI bus side, the addresses are software configurable, as described in Section 6.5 and Section 6.6. The PCI mailbox registers are mapped into PCI memory space and respond only to PCI memory cycles.

Both PCI mailbox registers are cleared to 0 at reset. The registers respond as soon as the memory space enable (MEMEN) bit is set in the PCI command register (Section 6.6.2); there is no enable function specific to these registers. If the mailbox base address register (offset 0x110 in Table 25) is not initialized before the MEMEN bit (bit 1) is set in the command register (offset 0x104 in Table 25), the base addresses for the two PCI mailbox registers will be mapped to offsets 0x0 and 0x800, respectively, and may collide with other PCI devices.

When a PCI mailbox register is accessed from the PCI bus (either read or write), it causes a mailbox interrupt bit (MB1 or MB2) to be set in the controller's interrupt control and status register (Section 8.1.2 on page 57). The interrupt is automatically cleared when the CPU reads or writes the corresponding PCI mailbox register.

6.12 Exclusive Access to PCI bus Resources

As shown in Table 25 and Table 26, the controller provides a mechanism for obtaining exclusive (locked) access to PCI targets, as defined in the *PCI Local Bus Specification*, Section 3.6. As a master on the PCI bus, the controller can lock a specific target on the PCI bus, using the LOCK# signal.

To request exclusive access to a target, software sets bit 0 of the PCI exclusive access register (described immediately below) to 1. When this bit is set, the next PCI access uses the exclusive protocol, if possible, allowing the addressed resource to become locked to the requester, via the controller. To release the target, software clears bit 0 prior to the last exclusive access; the current access remains exclusive until completed, at which time the target resource is released.

When the PCI bus target is locked, transactions are allowed only between the controller and the locked target. Transactions that do not complete are retried until they successfully complete. If the retry limit set in the retry value register (Section 6.6.7.1 on page 43) is reached, the controller sets bit 2 of the PCI exclusive access register to 1.

If the controller receives an abort during a locked transaction, it sets bit 3 of the PCI exclusive access register to 1.

The controller can also perform exclusive accesses as a target. To configure this, software sets bit 1 of the PCI exclusive access register. When the controller senses that it is the target of a locked PCI bus cycle, it enters the locked mode. While in the locked mode, no other accesses to the controller, either from the PCI bus or from the CPU bus, are allowed until the master negates both FRAME# and LOCK#. However, refresh cycles are permitted to the DRAM system even while the memory is locked.

6.12.1 PCI Exclusive Access Register

The exclusive access register stores a read/write word at offset 0x60. It has the following fields:

Bit 0	EAREQ	*Exclusive Access Request* 1 = Exclusive access request. In response, the controller asserts LOCK#, if conditions on the PCI bus permit (see Section 3.6 of the *PCI Local Bus Specification* for details). 0 = Release target; the controller negates LOCK# after completing the current access.
Bit 1	UNLOCK	*Controller Is Not a Locked Target:* 1 = Disable controller as target of exclusive access. 0 = Enable controller as target of exclusive access.
Bit 2	RTRYREACHED	*Retry Limit Reached* 1 = Retry limit has been reached. The limit is set in the retry value register (Section 6.6.7.1 on page 43), and the retry count can be read in the PCI retry counter register (Section 6.8 on page 47). 0 = Retry limit has not been reached.
Bit 3	ABORT	*Abort Received*. 1 = Either a master-abort or target-abort has been received while the controller was asserting LOCK#. These aborts are described in Figures 3-4 and 3-10 of the *PCI Local Bus Specification*. 0 = No abort has been received.
Bits 31:4	*Reserved*	Hardwired to 0

7.0 DMA Transfers

The controller supports CPU-initiated DMA transfers between memory and the PCI bus. These transfers can be unaligned reads or writes at the full PCI rate of 133 MB/s. Two sets of CPU-programmed registers configure DMA transfers; one set of registers can be read or written while the other set is controlling a transfer. An 8-word (32-byte) bidirectional FIFO, called the DMA FIFO, temporarily stores PCI-to-memory or memory-to-PCI transfers inside the controller.

To initiate a DMA transfer, the CPU configures the controller's DMA registers (Section 7.3) with the memory address, PCI bus address, read-write transfer direction, boundary crossing points, end-of-transfer interrupt enable, and transfer enable. Once configured, the controller arbitrates for the memory and PCI bus and performs the transfer independently of the CPU.

PCI bus masters cannot initiate DMA transfers. Instead, such masters gain access to the controller's memory through PCI target address windows, described in Section 6.4.

7.1 Types of DMA Operations

DMA transfers can be from the PCI bus to memory (called a PCI read), or from memory to the PCI bus (called a PCI write). The direction is set in the R/W bit (bit 29) of the DMA control registers.

7.1.1 PCI-to-Memory Transfers (PCI Read)

For a PCI bus read (from the PCI bus to memory), the controller begins by requesting access to the PCI bus. When granted, the controller reads words from the PCI bus and stores them in its 8-word DMA FIFO. When the FIFO contains 4 words (half-full), the controller requests access to memory, which is granted as soon as any current CPU memory operation completes. Then, the controller begins emptying data from the FIFO to memory at the fastest rate supported by memory.

If the controller's DMA FIFO becomes full during a transfer, the controller releases the resource responsible for filling the FIFO until the FIFO is emptied to 4 words (half-empty). Then the controller reacquires the resource and continues filling the FIFO.

7.1.2 Memory-to-PCI Transfers (PCI Write)

For a PCI bus write (from memory to the PCI bus), the controller begins by requesting access to memory. When granted, the controller reads the first 8 words into its DMA FIFO at the fastest rate supported by memory.

If the addressed memory is in the *base memory range* and is *bank interleaved*, the controller requests access to the PCI bus after receiving 1 word from memory into its DMA FIFO. For any other memory configuration, the controller accumulates 4 words in its FIFO before requesting the PCI bus. The controller attempt to empty the FIFO as quickly as the PCI target can accept the data. Meanwhile, the controller attempts to keep its FIFO full. If the FIFO becomes full, the controller releases memory until the FIFO reaches half full (4 words), at which time it again accesses memory and begins filling the FIFO.

If the FIFO becomes empty, the controller issues a disconnect command to the PCI bus. If there is more data to transferred in the same DMA operation, the controller continues filling its FIFO from memory and accesses the PCI bus when either 1 word or 4 words have been accumulated in the FIFO, depending on the memory type as

described above. When the correct number of words has been read from memory, the controller stops filling its FIFO but continues emptying the FIFO until the last transfer completes.

7.1.3 Transfer Completion

When a DMA transfer completes, the controller interrupts the CPU (if INT# interrupts are enabled), with bit 1 or 2 set in the controller's interrupt control and status register (Section 8.1.2). The controller then checks the other set of DMA control registers to determine if another DMA transfer is pending. If another is pending, the controller allows a one pending CPU-to-memory and one pending CPU-to-PCI transaction to complete before beginning the next DMA transfer.

If the controller receives a PCI master-abort or target-abort termination, the controller resets the DMA channel, indicates the error type by setting bits 1:0 of the bus error status register (Section 8.1.1), and interrupts the CPU (if Int# interrupts are enabled). If a DMA bus error occurs, the controller interrupts the CPU (if Int# interrupts are enabled), with bit 5 set in the interrupt control and status register (Section 8.1.2). If the other DMA channel is enabled to begin a transfer (bit 28 set in the other DMA control register), the controller begins the pending transfer.

7.2 CPU Access During DMA Transfers

After a DMA transfer starts, the CPU cannot access memory until the DMA reaches a *boundary crossing point* in the memory address space. The boundary crossing point is programmed by the CPU, in increments of 8-words, in the DMA control registers. The controller allows the CPU to perform one memory transaction at each boundary crossing point. Thus, a CPU memory-read stalls between boundary crossing points, but a CPU memory write will be buffered in the CPU-to-memory write FIFO. When the write FIFO contains a posted write, all other CPU-to-memory transactions stall (EOK# negated) until a boundary crossing point is reached or the DMA transaction completes.

If the CPU attempts to read an address mapped to the PCI bus during a DMA transfer, the CPU read stalls until the DMA transfer completes. If the CPU attempts to write to an address mapped to the PCI bus address during a DMA transfer, the CPU write is posted in the PCI master FIFO until the DMA transfer completes. When the PCI master FIFO contains a posted write, all other CPU transactions stall (EOK# de-asserted) until the FIFO is emptied.

7.3 DMA Registers

The controller has two sets of DMA configuration registers, each of which controls a DMA transfer. One set of registers may be read or written while the other set is controlling a DMA transfer. The configuration registers are:

- *DMA control registers 1 and 2*
- *DMA memory address registers 1 and 2*
- *DMA PCI address registers 1 and 2*

In addition to these configuration registers, the controller also has the following DMA status registers:

- *DMA words remaining egister*
- *DMA current memory address register*
- *DMA current PCI address register*

The registers at located at offsets 0x38 through 0x4C, and offsets 0x64 through 0x6C, as shown in Table 5 on page 6. The following sections describe the contents of these registers.

7.3.1 DMA Control Registers 1 and 2

These two registers are used by the CPU to configure DMA transfers. One register can be read or written while the other is controlling a DMA transfer. When a DMA transfer has started, changing bits in the DMA control registers, except the DRST and SU bits (bits 24 and 27), have no effect.

The registers are 4 bytes wide, at offsets 0x38 and 0x44. They are initialized to 0 at reset and contain the following fields:

Bits 19:0 BlkSize — *Block Size*. The number of bytes (up to 1 MB) to be transferred. 0 = 1 MB.

Bits 23:20 BoundPnt — *Boundary Crossing Point*. The address boundary, in 8-word (32-byte) increments, at which CPU memory transactions may be performed during a DMA transaction. When the current DMA memory address matches a boundary, as defined in this field, the controller allows the CPU to perform one memory transaction. Boundaries are defined in table below.

BoundPnt Field	Byte Address Boundary
0000	None
0001	32
0010	64
0011	128
0100	256
0101	512
0110	1K
0111	2K
1000	4K
1001	8K
1010	16K
1011	32K
1100	64K
1101	128K
1110	256K
1111	512K

Bit 24 DRST — *DMA Reset*
1 = Terminates an in-progress DMA transfer and resets the DMA logic, after completion of the current bus cycle. This bit take precedence over all other bits in the DMA command register. The value written to the other bits of this register when DRST is 1 is irrelevant: this bit takes precedence.
0 = The controller clears this bit automatically after the DMA channel has been reset.

Bit 25	MIO	*PCI Memory or I/O* 1 = Transaction to or from PCI memory space 0 = Transaction to or from PCI I/O space
Bit 26	INC	*Increment PCI Address* 1 = PCI address incremented as the DMA transfer is performed. 0 = Restart from original starting address for any condition that causes the DMA to restart a PCI burst. The starting address is programmed in the DMA memory address register or the DMA PCI address register.
Bit 27	SU	*Suspend DMA* 1 = Suspend the current DMA transfer after completion of the current PCI cycle. All register values are preserved. 0 = Restart the suspended DMA transfer. This bit may be set and cleared without consideration of the other bits in the DMA control registers, except DRST (bit 24) That is, when the DMA transfer has started, changing bits other than SU and DRST has no effect.
Bit 28	GO	*Begin Transfer* 1 = Start the DMA transfer as soon as the PCI and memory buses are available. 0 = The controller automatically clears this bit after the transfer completes. Software-clearing this bit has no effect; the DMA transfer will continue. This bit must not be set if the bus master enable bit (bit 2) in the PCI command register (Section 6.6.2.1) has not been previously set.
Bit 29	R/W	*PCI Read/Write Direction* 1 = Read data from the PCI bus and write it to local memory. 0 = Read data from local memory and write it to PCI bus.
Bit 30	IE	*Interrupt Enable* 1 = When the DMA transfer completes, the controller interrupts the CPU (if Int# interrupts are enabled), with bit 1 or 2 set in the controller's interrupt vontrol and status register (see Section 8.1.2). 0 = The controller does not interrupt the CPU on completion of the DMA transfer.
Bit 31	BZ	*Busy* (read only) 1 = The DMA transfer controlled by this register is currently in process. This bit may be polled. 0 = No DMA transfer controlled by this register is in process.

7.3.2 DMA Memory Address Registers 1 and 2

These registers are programmed by the CPU with the starting memory address for the transfer. The registers are at offsets 0x3C and 0x48. They are initialized to 0x0 at reset and contains the following fields:

Bits 31:0	Local Address	*Memory Starting Address*. The starting address to be used when accessing the controller's memory. This field remains static throughout the DMA transfer. The current memory address being accessed can be read from the DMA current memory address register (Section 7.3.5).

7.3.3 DMA PCI Address Registers 1 and 2

These registers are programmed by the CPU with the starting PCI bus address for the transfer. The registers are at offsets 0x40 and 0x4C. They are initialized to 0x0 at reset and contains the following fields:

Bits 31:0	LocalAddr	*PCI bus Starting Address*. The starting address to be used when accessing the PCI bus. This field remains static throughout the DMA transfer. The current PCI bus address being accessed can be read from the DMA current PCI address register (Section 7.3.6).

7.3.4 DMA Words Remaining Register

This register can be read by the CPU to determine the number of words remaining in the current DMA transfer. The register is at offset 0x64. It is initialized to 0x0 at reset and contains the following fields:

Bits 31:0	WordCnt	*Words Remaining* (read-only) The number of words remaining to be transferred in the current DMA operation.

7.3.5 DMA Current Memory Address Register

This register can be read by the CPU to determine the memory address currently being accessed in a DMA transfer. The register is at offset 0x68. It is initialized to 0x0 at reset and contains the following fields:

Bits 31:0	CrntAddr	*Current Memory Address* (read-only) The current memory address of the DMA operation.

7.3.6 DMA Current PCI Address Register

This register can be read by the CPU to determine the PCI address currently being accessed in a DMA transfer. The register is at offset 0x6C. It is initialized to 0x0 at reset and contains the following fields:

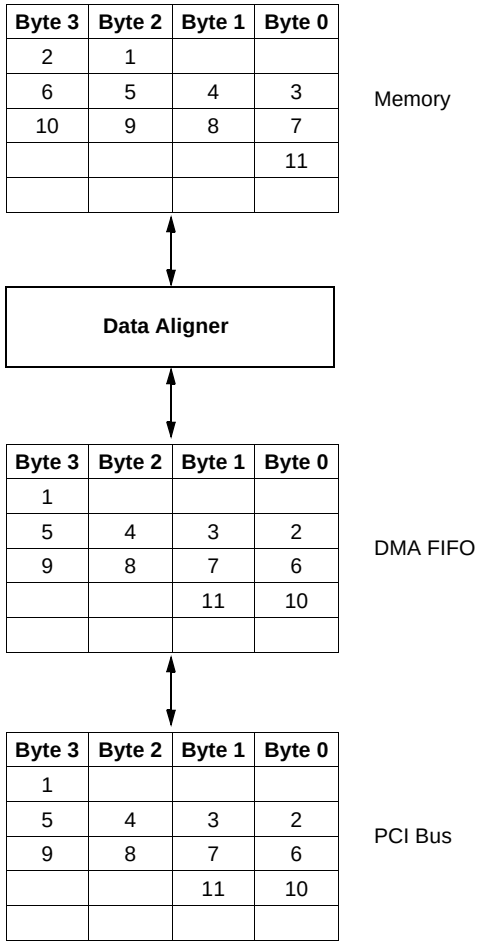
Bits 31:0	CrntAddr	*Current PCI Address* (read-only) The current PCI address of the DMA operation.

7.4 Data Aligner

The controller automatically handles unaligned bidirectional transfers between the PCI bus and memory. The aligner permits the controller to use high-speed burst protocols for transfers, even when both the source and destination addresses are not aligned on word-address boundaries or with each other.

The align function is performed between the DMA FIFO and memory, or vice versa. The aligner shifts byte data into the DMA FIFO or memory, in the alignment required by, or supplied by the PCI bus. Figure 6 shows the operation of the aligner for a DMA transfer from byte address 0002 in memory to byte address 0003 on the PCI bus (or in the opposite direction).

Figure 6: DMA Transfer Alignment Example



8.0 Interrupts

The controller supports maskable interrupts using the Int# input to the CPU, and non-maskable interrupts using the NMI# input to the CPU.

8.1 Maskable Interrupts (Int#)

The controller can be enabled to interrupt the CPU when the following types of memory or PCI bus errors occur:

- *Illegal Address Errors:* Memory accesses by the CPU to physical addresses outside of one of the five memory ranges, one of the three PCI windows, or one of the controller's internal registers.
- *Target Abort, Master Abort, and Retry Limit Errors:* PCI bus accesses by the CPU that result in target abort, master abort, or more retries than specified by retry value register.

The controller reports errors to the CPU by asserting the INT# signal, if enabled by bit 0 in the interrupt control and status register (Section 8.1.2). The CPU's interrupt service routine can then read the bus error status register (Section 8.1.1) to determine the type of error. The Int# signal is a level-sensitive output to the CPU and may not be shared with other interrupt generators. The controller does not prioritize the various interrupt sources.

During CPU reads, any of the detected errors cause the controller to return the correct number of data words, but with the bus error bit set in SysCmd[0] for those words that are returned after the word that caused the bus error.

For DMA accesses to controller memory that miss the configured memory ranges, the bus error status register contain the error information, just as for errors during CPU accesses. DMA accesses to the PCI bus that result in target abort, master abort, or more retries than specified, set only the error type field in the bus error status register but not the error address field. Bus errors generated by the DMA cause the DBE interrupt to be generated, if enabled.

External PCI accesses that hit either target window, but miss all internal controller resources will set the ET code to 0 and the error address. The error address will be the translated address. External PCI accesses can never set an ET code other than 00. This bus error sets the PBE interrupt, if enabled.

8.1.1 Bus Error Status Register

This read-only register should be read by the CPU, when the CPU detects a bus error interrupt from the controller. It contains the cause of the error. The contents remain constant after the error, until read by the CPU. The register is at offset 0x50. It is initialized to 0x0 at reset and contains the following fields:

Bits 1:0	ET	*Error Type* 00 = Illegal address 01 = Target abort received 10 = Master abort signaled 11 = Retry limit reached. The value specified in the retry value register (Section 6.6.7.1) has been reached.

Bits 31:2	EA	*Error Address*. The most significant 30 bits of the local (controller side) physical address that caused the error. This field is valid for CPU and DMA accesses to the controller's memory. It is not valid for DMA accesses to the PCI bus memory space.

8.1.2
Interrupt Control and Status Register

The interrupt control and status register is read-only in its lower byte, read-write in its middle two bytes, and write-only in the high byte. The low byte should be read by the CPU, along with the bus error status register, when the CPU detects an Int# interrupt from the controller. The contents of the interrupt control and status register remain constant after the error, until read by the CPU.

The register is at offset 0x50. It is initialized to 0 at reset and contains the following fields:

Bit 0	CBE	*CPU Bus Error* (read only) 1 = CPU bus error 0 = No such error
Bit 1	DMA1	*DMA Channel 1 Complete* (read only) 1 = Transfer specified in DMA control register 1 is complete 0 = Transfer 1 is not complete
Bit 2	DMA2	*DMA Channel 2 Complete* (read only) 1 = Transfer specified in DMA control register 2 is complete 0 = Transfer 2 is not complete
Bit 3	MB1	*PCI Mailbox 1 Accessed* (read only) 1 = Mailbox 1 accessed from the PCI bus 0 = Mailbox 1 not accessed
Bit 4	MB2	*PCI Mailbox 2 Accessed* (read only) 1 = Mailbox 2 accessed from the PCI bus 0 = Mailbox 2 not accessed
Bit 5	DBE	*DMA Bus Error* (read only) 1 = A bus error occurred during a DMA transfer 0 = No such error
Bit 6	PBE	*PCI Bus Error* (read only) 1 = A bus error occurred during a PCI target access 0 = No such error
Bit 7	PAR	*PCI Parity Error* (read only) 1 = Parity error. The error can be on (a) address parity during a target cycle, (b) data parity during a target write cycle, or (c) data parity during a master read cycle. 0 = No such error

Bit 8	CBEmsk	*CPU Bus Error Enable* (read/write) 1 = Enable CPU bus error interrupts 0 = Disable such interrupts
Bit 9	DMA1msk	*DMA Channel 1 Complete Enable* (read/write) 1 = Enable DMA channel 1 complete interrupts 0 = Disable such interrupts
Bit 10	DMA2msk	*DMA Channel 2 Complete Enable* (read/write) 1 = Enable DMA channel 2 complete interrupts 0 = Disable such interrupts
Bit 11	MB1msk	*PCI Mailbox 1 Accessed Enable* (read/write) 1 = Enable PCI mailbox 1 accessed interrupts 0 = Disable such interrupts
Bit 12	MB2msk	*PCI Mailbox 2 Accessed Enable* (read/write) 1 = Enable PCI mailbox 2 accessed interrupts 0 = Disable such interrupts
Bit 13	DBEmsk	*DMA Bus Error Enable* (read/write) 1 = Enable DMA bus error interrupts 0 = Disable such interrupts
Bit 14	PBEmsk	*PCI Bus Error Enable* (read/write) 1 = Enable PCI bus error interrupts 0 = Disable such interrupts
Bit 15	PARmsk	*PCI Parity Error Enable* (read/write) 1 = Enable PCI parity error interrupts. 0 = Disable such interrupts
Bits 23:16	*Reserved*	Hardwired to 0
Bits 24	CBEclr	*CPU Bus Error Clear* (write-only) 1 = Clear the CPU bus error interrupt (always returns 0 when read)
Bits 25	DMA1clr	*DMA Channel 1 Complete Clear* (write-only) 1 = Clear the DMA channel 1 complete interrupt Always returns 0 when read
Bits 26	DMA2clr	*DMA Channel 2 Complete Clear* (write-only) 1 = Clear the DMA channel 2 complete interrupt Always returns 0 when read
Bits 27	MB1clr	*PCI Mailbox 1 Accessed Clear* (write-only) 1 = Clear the PCI mailbox 1 accessed interrupt Always returns 0 when read
Bits 28	MB2clr	*PCI Mailbox 2 Accessed Clear* (write-only) 1 = Clear the PCI mailbox 2 accessed interrupt Always returns 0 when read
Bits 29	DBEclr	*DMA Bus Error Clear* (write-only) 1 = Clear the DMA bus error interrupt (always returns 0 when read)

Bits 30	PBEclr	*PCI Bus Error Clear* (write-only) 1 = Clear the PCI bus error interrupt (always returns 0 when read)
Bit 31	PARclr	*PCI Parity Error Clear* (write-only) 1 = Clear the PCI parity error interrupt (allways returns 0 when read)

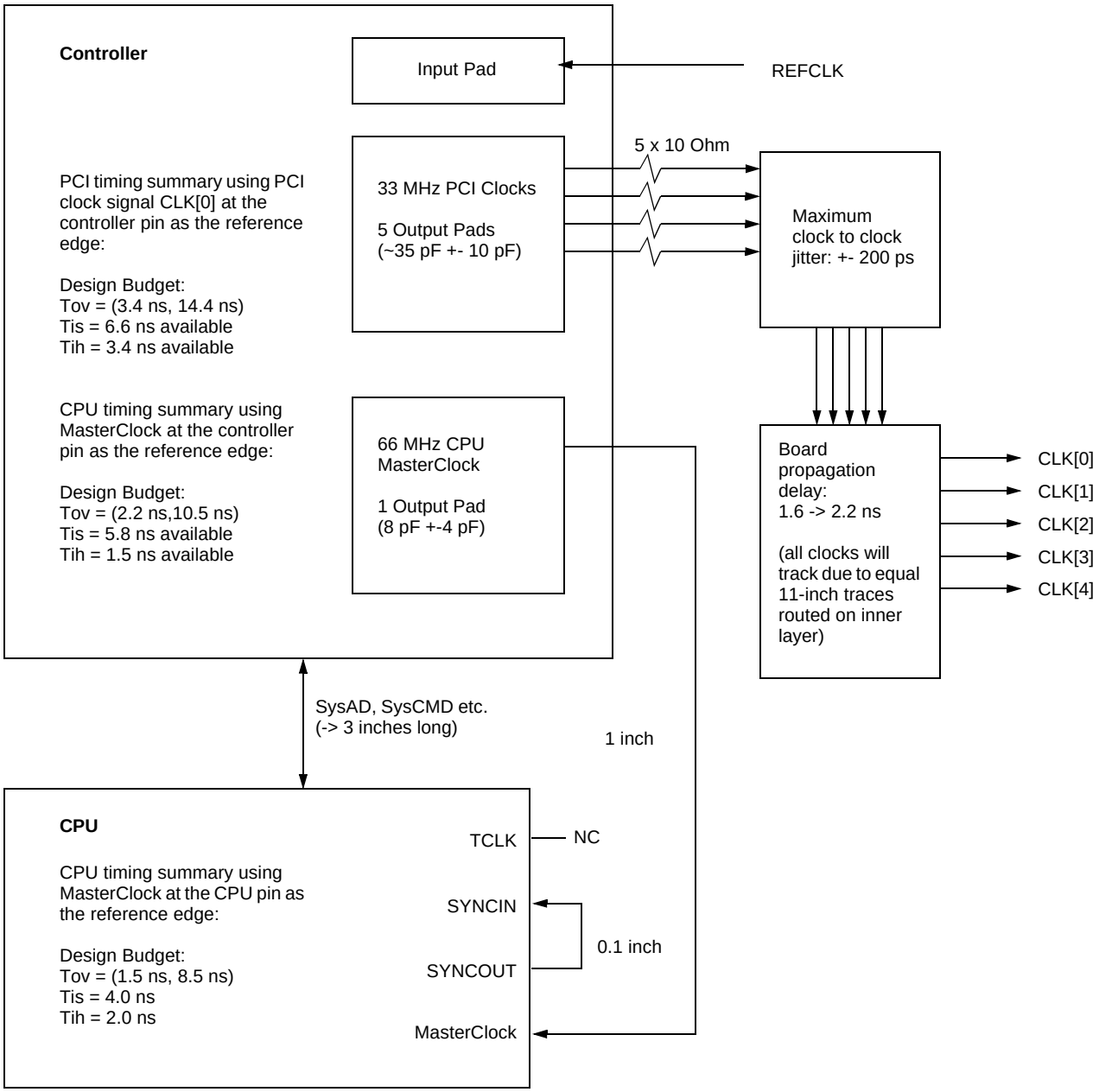
8.2 Non-Maskable Interrupts (NMI#)

The controller asserts NMI# only when a PCI device asserts SERR#.

9.0 Clocking

The controller receives a 66 MHz oscillator reference clock (REFCLK) and distributes the 66 MHz MasterClock to the CPU. The controller also generates and distributes four copies of the 33 MHz PCI clock (CLK[3:0]). Figure 7 shows the controller's clock connections with the system.

Figure 7: Clock Connections



10.0 Reset Configuration Signals

The rising edge of PCI bus reset (RST#) serves as the controller's reset. Table 27 lists the configuration signals that the controller samples for one REFCLK edge while RST# is active.

Table 27: Reset Configuration Signals

MuxAd signals	Function	Description
MuxAd[2:0]	Boot ROM size	Table 13 on page 16
MuxAd[6:3]	Boot ROM write protect	Section 5.5.1.1 on page 16
MuxAd[7]	Test enable 1 = enable 0 = disable	Section 13.0 on page 97
MuxAd[8]	Flash boot enable 1 = enable 0 = disable	Section 5.5.2.2 on page 17
MuxAd[10]	Endian byte order 1 = big endian 0 = little endian	Section 4.1 on page 8

11.0 Endian Mode Software Issues

11.1 Overview

The native endian mode for MIPS processors, like Motorola and IBM 370 processors, is *big endian*. However, the native mode for Intel (which developed the PCI standard) and VAX processors is *little endian*. For PCI-compatibility reasons, most PCI peripheral chips, including the VRC4373 controller, operate natively in *little-endian* mode.

While the VRC4373 controller is natively little-endian, it supports either big- or little-endian mode on the CPU interface. The state of the MuxAd[11] signal at reset determines this endian mode (Section 10.0). However, there are important considerations when using the controller in a mixed-endian design. The most important aspect of the endian issue is which byte lanes of the SysAD bus are activated for a particular address.

If the big-endian mode is implemented for the CPU interface, the controller swaps bytes within words and halfwords that are coming in and going out on the SysAD bus. All of the controller's other interfaces operate in little-endian mode. There are a number of implications associated with this:

- Data in memory is always ordered in little-endian mode, even with a big-endian CPU interface.
- Little-endian bit-fields and other data structures that span two or more bytes (such as bit-fields within registers or FIFOs) are fragmented when the CPU interface is big-endian. The contents of these data structures are byte-swizzled, so that the bits are arranged [7:0], [15:8], [23:16], [31:24], rather than [31:0].
- Big-endian devices on the PCI bus must be byte-swapped external to the controller.

The sections below view the endian issue from a programmer's perspective. They describe how to implement mixed-endian designs and how to make code endian-independent.

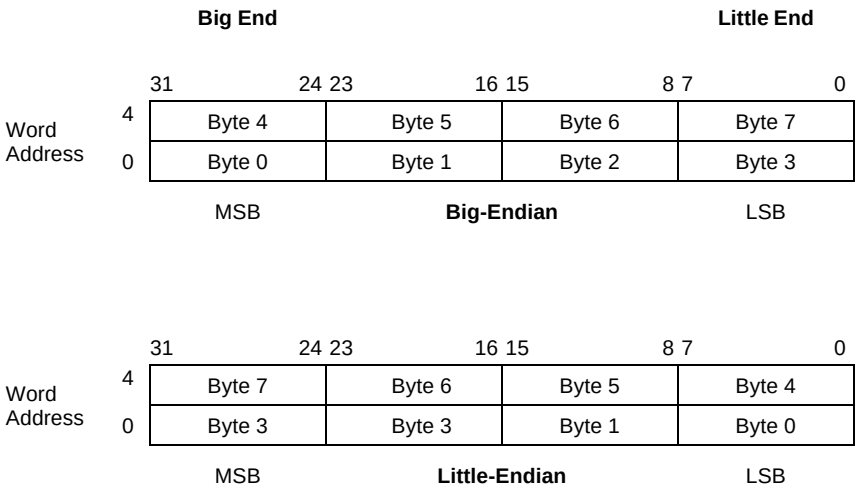
11.2 Endian Modes

The endian mode of a device refers to its word-addressing method and byte order:

- *Big-endian* devices address data items at the *big end* (most significant bit number). The most-significant byte (MSB) in an addressed data item is at the *lowest* address.
- *Little-endian* devices address data items at the *little end* (least significant bit number). The most significant byte (MSB) in an addressed data item is at the *highest* address.

Figure 8 shows the bit and byte order of the two endian modes, as it applies to bytes within word-sized data items. The *bit order* within bytes is the same for both modes. The big (most-significant) bit is on the left side, and the little (least significant) bit is on the right side. Only the *bit order* of sub-items is reversed within a larger addressable data item (halfword, word, doubleword, quadword) when crossing between the two endian modes. The sub-items' *order of significance* within the larger data item remains the same. For example, the least significant half word (LSHW) in a word is always to the right and the most-significant halfword (MSHW) is to the left.

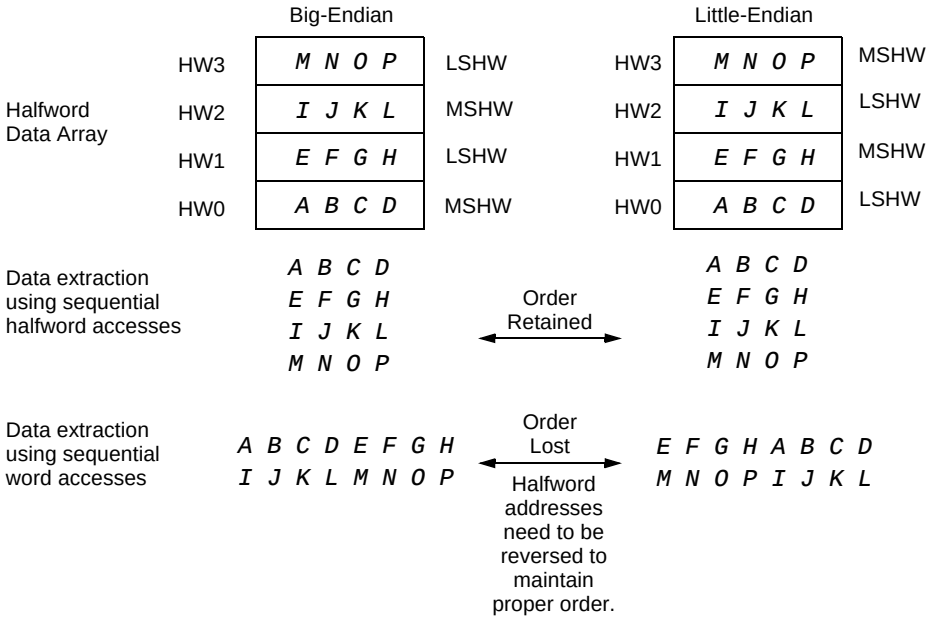
Figure 8: Bit and Byte Order of Endian Modes



MSB = Most Significant Byte
LSB = Least Significant Byte

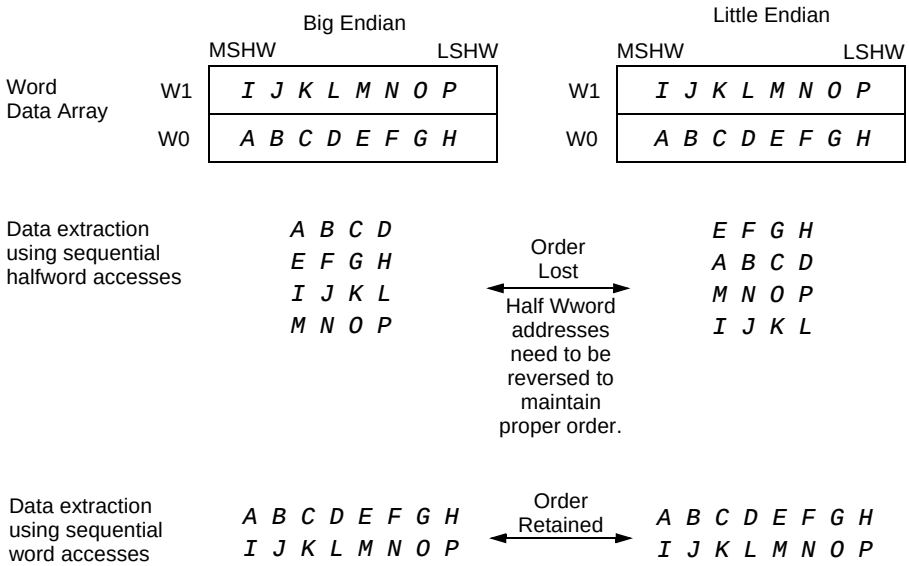
If the access type matches the data item type, no swapping of data sub-items is necessary. Thus, when making half-word accesses into a data array consisting of halfword data (Figure 9), no byte-swapping takes place. In this case, data item *bit order* is retained between the two endian modes. The code that sequentially accesses the half-word data array would be identical, regardless of the endianness of its CPU. The code would be endian-independent.

Figure 9: Halfword Data-Array Example



However, when making half-word accesses into a data array consisting of word data (Figure 10), access to the *more-significant* halfword requires the address corresponding to the *less significant* half word (and vice versa). Such code is not endian-independent. A supergroup access (for example, accessing two half words simultaneously as a word from a half-word data array) causes the same problem. Such problems also arise when a halfword access is made into a 32-bit I/O register, whereas a word access into a 32-bit register creates no problem.

Figure 10: Word Data-Array Example



11.3 LAN Controller Example

The AMD AM79C791 LAN controller is one example of how a PCI bus device that is natively little-endian adapts to mixed-endian environments. This LAN controller provides limited support for big-endian system interfaces. Its designers assumed only two data types: a 32-bit word corresponding to the width of I/O registers, and an 8-bit byte corresponding to the width of the Ethernet DMA FIFO.

11.3.1 DMA Accesses From Ethernet FIFO

Ethernet data packets consist of bytes. To maximize bus bandwidth, these bytes are transferred via 32-bit word DMA accesses into memory. This access-data mismatch corresponds to the supergroup scenario shown at the bottom of Figure 9. The mismatch means that a byte-swap must be performed to allow the little-endian LAN controller to access the big-endian memory. The LAN controller provides its own internal hardware for this byte-swap.

11.3.2 Word Accesses Into I/O Registers

The LAN controller's designers assumed that the 32-bit internal I/O registers would be accessed by 32-bit word transfers. In that case, the access type and data type match, and no swapping of bytes or halfwords is needed because order of significance is the same for both endian modes. For such word transfers, the I/O register model is endian-independent, and the LAN controller's designers did not provide internal swapping hardware for non-word accesses into the I/O registers.

Word accesses offer the advantage that the register address values documented in the AM79C971 Technical Manual can be used without change (although offsets for individual register fields such as the PCI Latency Timer must be ignored). The position of individual register fields as well as byte position within these fields would also remain the same as documented in the Technical Manual.

11.3.3 Byte or Halfword Accesses Into I/O Registers

Word accesses can cause some inconvenience (for example, shadow registers) when modifying only one or two fields within a 32-bit PCI register. In this case, byte or half-word access to the 32-bit register may be simpler. This type of transfer is analogous to the halfword access into a data array consisting of word data types, shown in Figure 10. Such accesses are mismatched to the defined data type and must be *cross-addressed* to get the byte or halfword of interest. The AM79C791 LAN controller does not provide big-endian hardware support to deal with byte or half-word transfers into the I/O registers. Code written to perform byte or half-word accesses into the 32-bit I/O registers will not be endian-independent.

The I/O register field addresses documented in the AM79C971 Technical Manual are based on a register model derived from a little-endian perspective. The number order of these addresses progresses from right (least significant) to left. However, a big-endian system will respond to all addresses as if the number order progresses from left (most significant) to right. To access the desired byte or half word, the address order documented in the technical manual must be reversed.

The fields of the PCI status register and PCI command register are two examples of frequently used I/O register fields. The address offsets documented in the technical manual are 0x06 and 0x04, respectively. The PCI command register field is located in the less-significant halfword of the 32-bit I/O register that is also located at offset 0x04. The PCI Command Register field shares the same offset with its 32-bit register because of the little-endian number order. In a big-endian system, the more-significant halfword (i.e. PCI Status Register field) would share the same offset value with its 32-bit register. So, if the offset 0x04 is used to access the PCI Command Register field, a big-endian system would actually access the PCI Status Register field. To access the proper halfword, the offsets must be exchanged between the two 16-bit register fields. In other words there must be a reversal (or swapping) of number order, relative to the information documented in the Technical Manual.

These special addressing considerations are completely independent of the operand pointers associated with the CPU register used as source or destination. The source or destination within the CPU's register file can be at any location, size, or alignment without altering the transfer results. A common error is to byte-swap CPU register data when transferring a halfword to or from a 32-bit register. The order of significance is the same for both endian modes, so no byte-swap is needed. This is purely an addressing problem.

Table 28 and Table 29 show how the offsets in the AM79C971 Technical Manual are swapped with the other offsets to produce the proper cross-addressed offset required by big-endian systems. The determining factors for the swap are the values of the two least significant bits of the offsets. According to the AM79C971 Technical Manual, the PCI Command Register field has the offset 0x04. Table 29 shows that the offset 0x06 is needed to access the PCI Command Register field. The two least significant bits of 0x04 are b00, which convert to b10 to give the result of 0x06h.

Table 28: Cross-Addressing for Byte Accesses Into a 32-bit I/O Register

Least Significant Bits of Offset From AM79C971 Technical Manual	Least Significant Bits of Offset Required by Big-Endian System
b0 0	b1 1
b0 1	b1 0
b1 0	b0 1
b1 1	b0 0

Table 29: Cross-Addressing for Half-Word Accesses into a 32-bit I/O Register

Least Significant Bits of Offset From AM79C971 Technical Manual	Least Significant Bits of Offset Required by Big-Endian System
b0 0	b1 0
b1 0	b0 0

11.4 GUI Controller Example

The Cirrus Logic CL-GD5465 GUI controller is another example of a PCI bus device that offers some mixed-endian support. The designers of this GUI controller assumed three data types: 32-bit word, 16-bit half word, and 8-bit byte. Unlike the LAN controller which could make certain assumptions as to data type (for I/O register or DMA FIFO accesses), the GUI hardware cannot determine what data type will be used during any particular data transfer; any data type might be involved in any I/O register or RDRAM access.

The data type must be known for a given bus transfer so that the appropriate byte or halfword swap can be performed. The data types may change from bus cycle to the next; one software task may be operating in parallel with and independently of another software task. One of the easiest methods to accommodate such an environment, without semaphores and such, is to provide address apertures into the memory space.

The aperture scheme calls for GUI hardware resources to be mirrored into three address ranges. Depending on which address range is selected, a specific data type and data swap is used. Chapter 13 of the *CL-GD5465 Technical Manual* gives details of these three apertures.

11.4.1 Word Accesses Into the I/O Registers

The GUI controller's internal 32-bit I/O registers can be accessed with 32-bit word transfers. In this case, the access type and data type match; no swapping of bytes or halfwords is required because the order of significance is the same for both endian modes. With such word transfers, the I/O register model is endian-independent, so the first address aperture described in the *CL-GD5465 Technical Manual* is used.

Word accesses have the advantage that the register address values documented in the technical manual can be used without change (although offsets for individual register fields such as the PC latency timer must be ignored). The position of individual register fields as well as byte position within these fields also remains the same as shown in the technical manual.

11.4.2 Byte or Half-Word Accesses Into I/O Registers

As in the LAN controller example, byte or half-word access may be simpler than word accesses when modifying only one or two fields within a 32-bit I/O register. This type of transfer is analogous to the half-word access into a data array consisting of word data types, shown in Figure 10. Such accesses are mismatched to the defined data type and must be swapped to get the byte or half word of interest. Code written to perform byte or halfword accesses into the 32-bit word I/O registers will not be endian-independent.

There are two methods to perform byte or half-word accesses into the GUI controller. The first method is the use of the apertures for half-word swap (second aperture) and byte-swap (third aperture). This method has the advantage that the little-endian addresses documented in the technical manual are the same as those used by big-endian code, except for the addition of the offset required to select the appropriate aperture. (As of this printing, the second aperture remains unverified and has generated some confusion resulting from poor documentation or improper implementation.)

The second method of performing byte or halfword accesses is to *cross-address* the transfer. Care must be taken, however, when referencing the *CL-GD5465 Technical Manual*. The I/O register field addresses documented in the technical manual are based on a little-endian register model. The number order of these addresses progress from right (least significant) to left. However, big-endian systems respond to addresses as if the number order progresses from left (most-significant) to right. To access the desired byte or half word, the address order documented in the technical manual must be reversed.

11.4.3 Accesses Into RDRAM

The CL-GD5465 GUI controller's internal pixel and video engines constrain the Rambus® DRAM (RDRAM) to be little-endian. Here again, big-endian systems have a few problems accessing data subgroups, such as a single-byte access into a 32-bit data type. Sub-item accesses are also a factor for RDRAM and the cross-addressing and address apertures solutions are the same as those described in Section 11.4.2. Supergroup access are also encountered with RDRAM. This situation is mentioned in Section 11.2 and shown in Figure 10. A specific GUI-oriented example of this would be an 8-bit data type, such as a pixel, which is transferred four at a time to maximize PCI bus bandwidth.

There are two methods for dealing with supergroup transfers. First is the address-aperture method, used in the sub-item scenario of Section 11.4.2. The third aperture, byte-swap, is used to provide the proper data swap for the four 8-bit pixel case. The second aperture, half-word swap, is used to transfer such things as two 16-bit pixels simultaneously.

The second aperture method requires that the data order in the CPU register be swapped prior to an RDRAM write access, or immediately after an RDRAM read access. To continue with the previous four-pixel transfer example, the byte number-order of the four pixels in the CPU register would be reversed. Now the pixel number-order increases, starting from the right side of the register (first pixel originally on left, now on right). Then, the four pixels are written into the RDRAM with a standard 32-bit word transfer (first aperture). The case of two 16-bit pixels requires the two half words to be swapped, but not the order of the two bytes inside the half words. This second method is probably more time-consuming and is not recommended.

12.0 Timing Diagrams

This section shows timing diagrams for the controller's various operations on the memory and PCI buses. The following notations are used:

A or An Address or sequential address number. Each address is also marked by a vertical dashed line.

D or Dn Data or sequential data-item number

SDRAM Synchronous DRAM

12.1 Memory Initialization and CPU Accesses to Local Memory

Figure 12 through Figure 16 show the timing for CPU accesses to the controller's local memory.

- CPU Word Write Cycles to Power-On Memory Initialization Registers (Figure 12) (This diagram shows the procedure described in Section 5.10 on page 31)
- CPU Byte Write/Word Read Cycles to/from Flash Boot ROM (Figure 12)
- CPU Word Write/Word Read Cycles to/from EDO Base Memory (Figure 13)
- CPU Word Write/Word Read Cycles to/from SDRAM Base Memory (Figure 14)
- CPU Word Write/Word Read Cycles to/from Fast-Page SIMM Memory (Figure 15)
- *C*PU Word Write/Word Read Cycles to/from SDRAM SIMM Memory (Figure 16)

Figure 11: CPU Word Write Cycles to Power-On Memory Initialization Registers

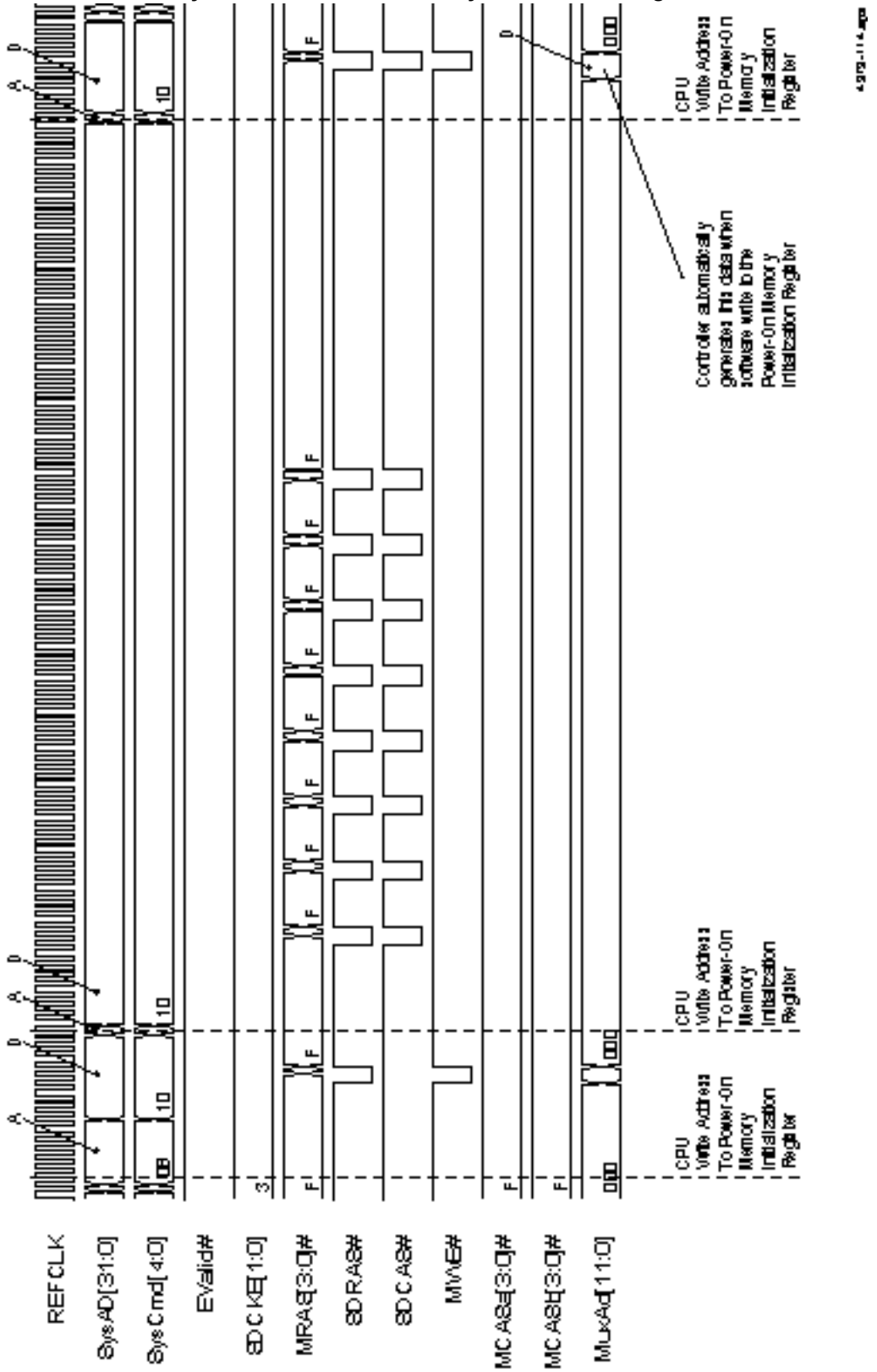


Figure 12: CPU Byte Write/Word Read Cycles to/from Flash Boot ROM

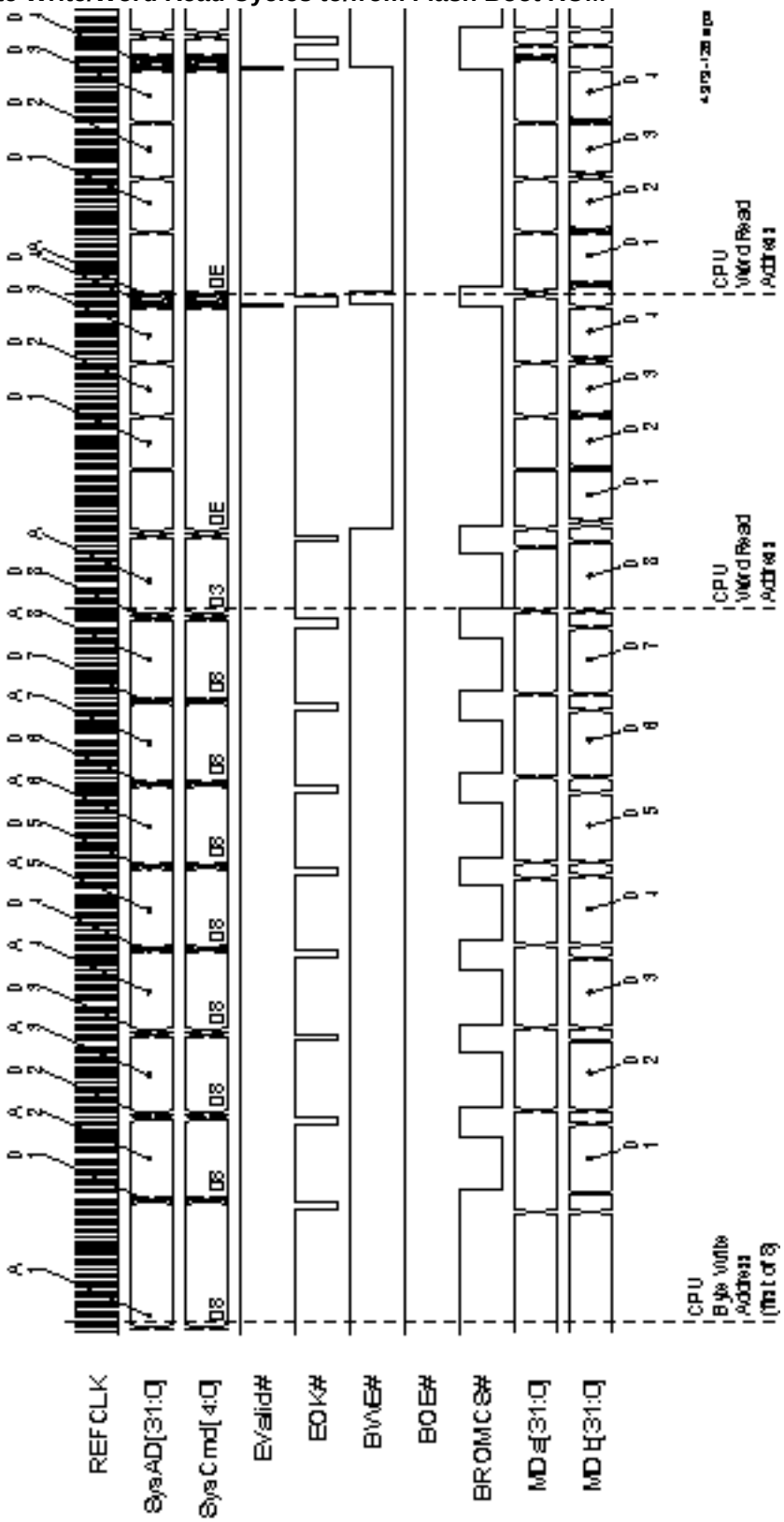


Figure 13: CPU Word Write/Word Read Cycles to/from EDO Base Memory

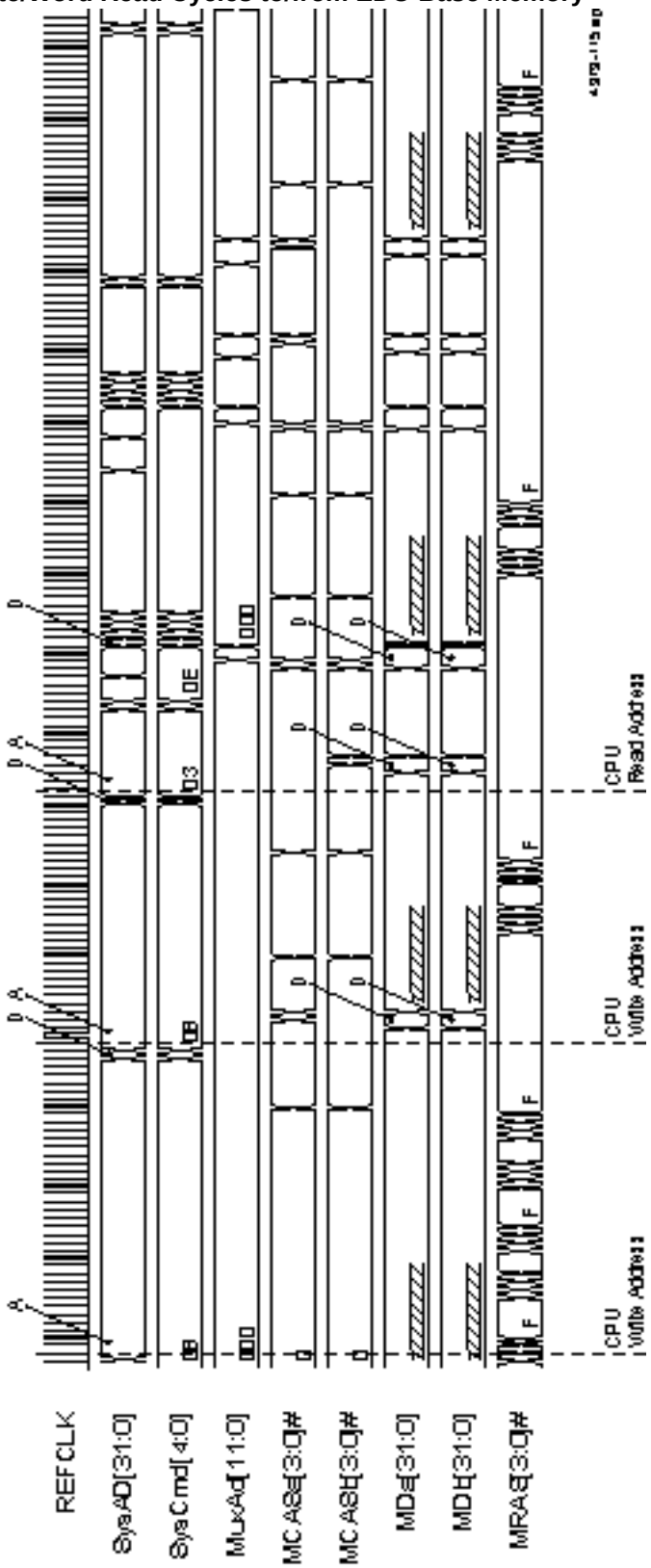


Figure 14: CPU Word Write/Word Read Cycles to/from SDRAM Base Memory

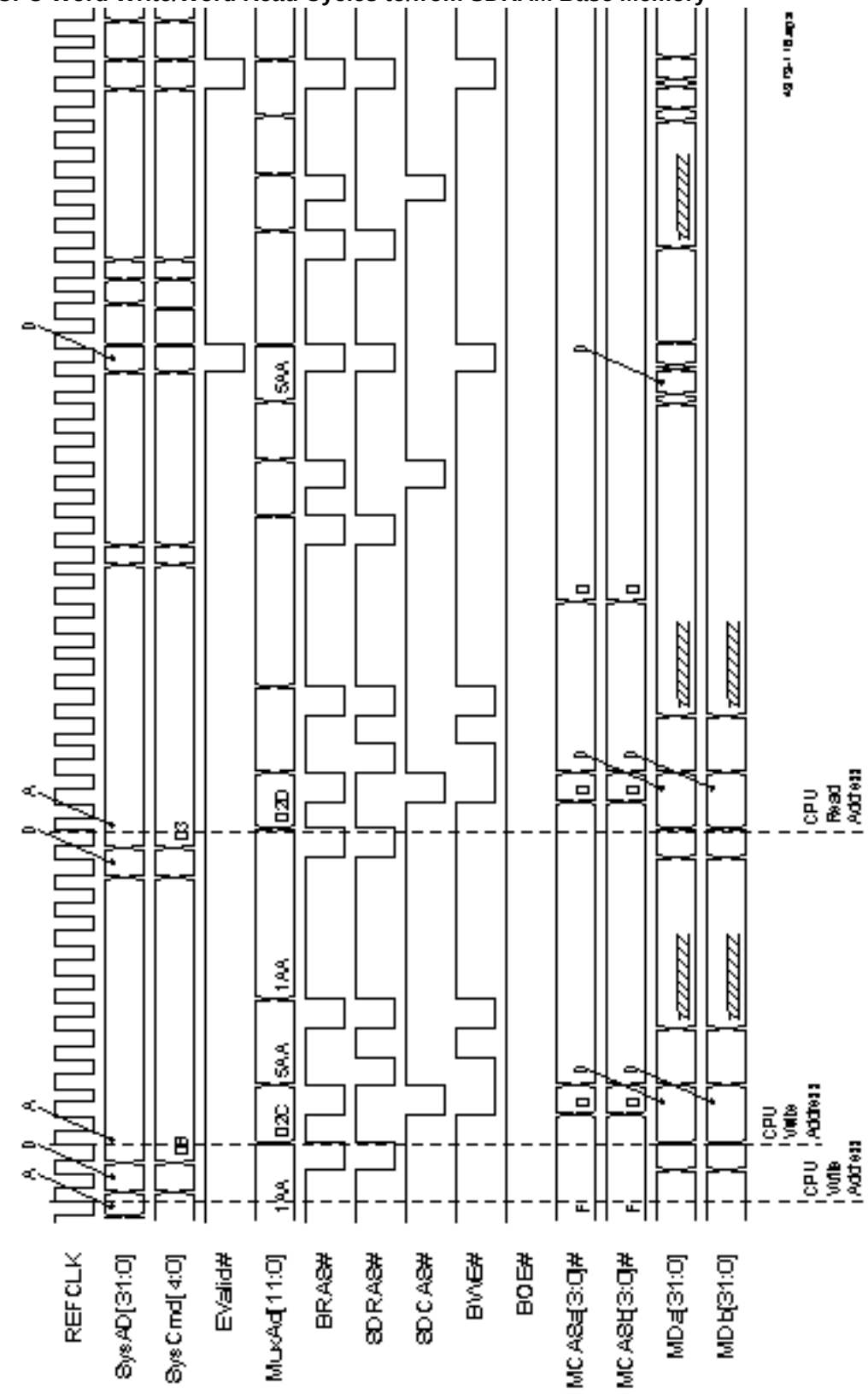


Figure 15: CPU Word Write/Word Read Cycles to/from Fast-Page SIMM Memory

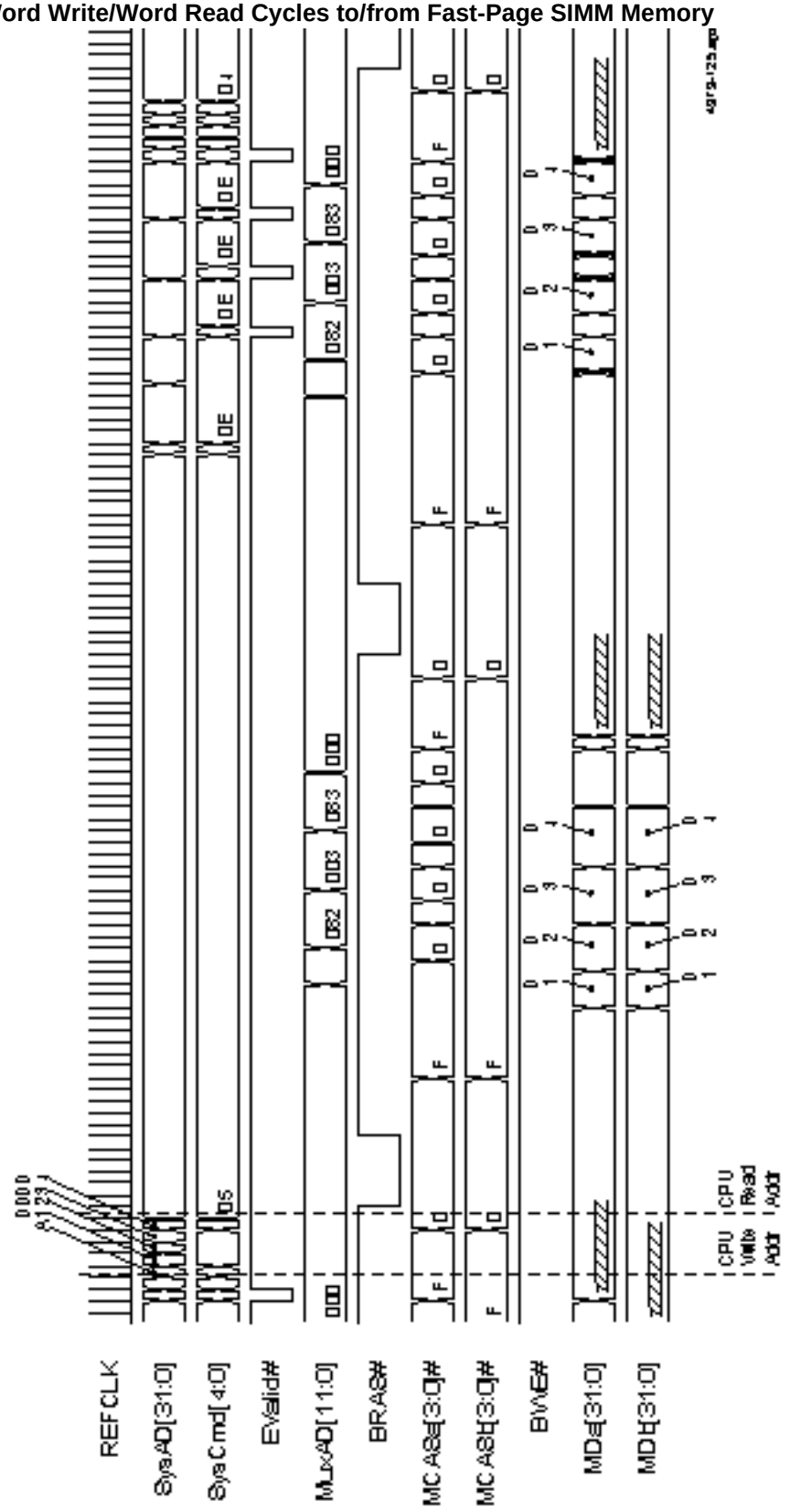
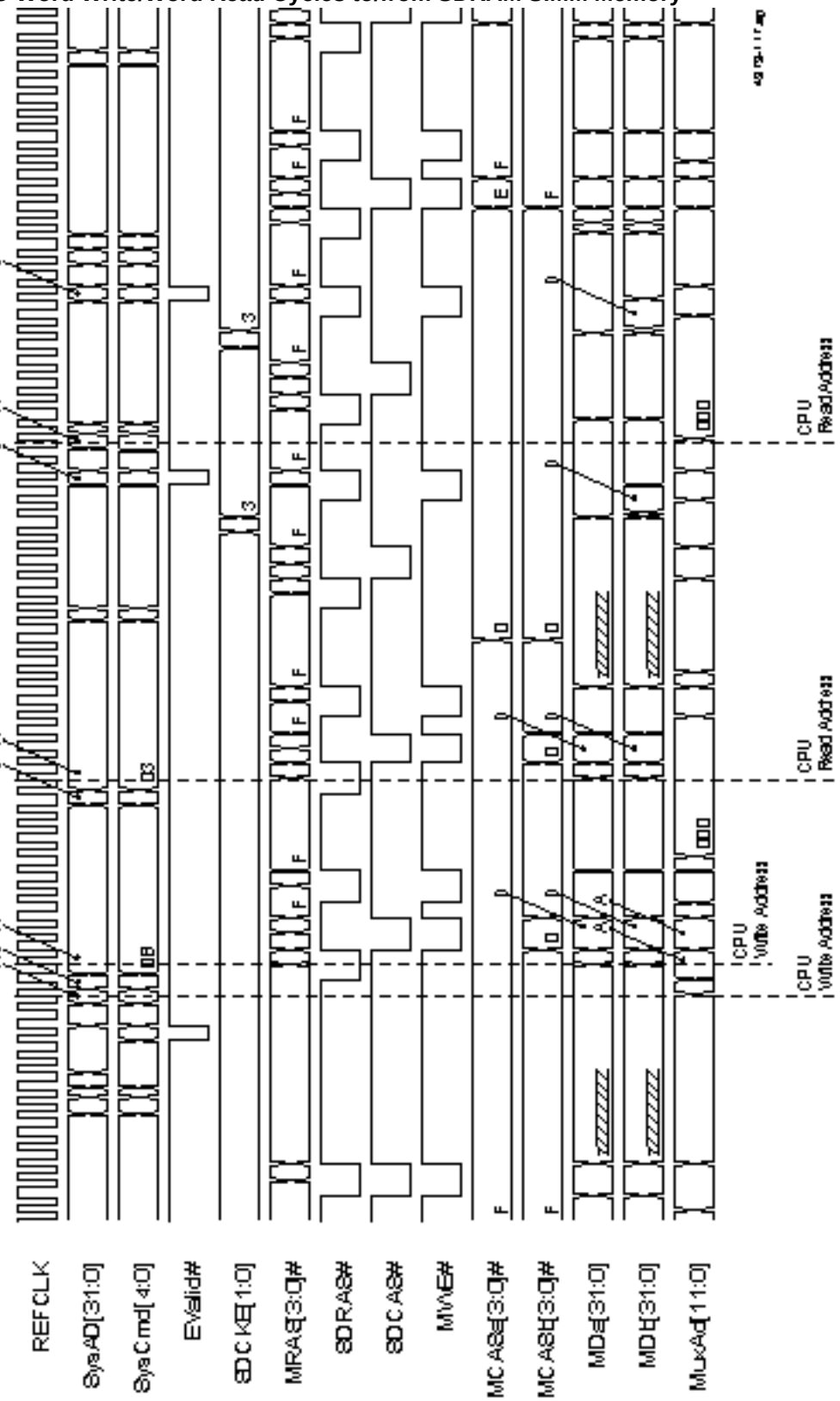


Figure 16: CPU Word Write/Word Read Cycles to/from SDRAM SIMM Memory



12.2
PCI Bus Accesses

Figure 17 through Figure 36 show the timing for various transactions on the PCI bus by the CPU and PCI bus masters.

- **CPU Configuration of PCI Space**
  - CPU Read Cycles from PCI Configuration Space (Figure 17)
  - CPU Write and Read Cycles to/from PCI Configuration Space (Figure 18)
- **CPU Accesses to PCI Memory**
  - CPU Byte Write and Read Cycles to/from PCI Memory (Figure 19)
  - CPU Halfword Write and Read Cycles to/from PCI Memory (Figure 20)
  - CPU Tri-Byte Write and Read Cycles to/from PCI Memory (Figure 21)
  - CPU Word Write Cycles to PCI Memory (Figure 22)
  - CPU Word Write and Read Cycles to/from PCI Memory (Figure 23)
  - CPU Back-to-Back Word Write Cycles to PCI Memory (Figure 24)
  - CPU 4-Word Block Write Cycles to PCI Memory (Figure 25)
  - CPU Back-to-Back 4-Word Block Write Cycles to PCI Memory (Figure 26)
- **CPU Accesses to PCI I/O**
  - CPU 4-Word Block Write Cycles to PCI I/O (Figure 27)
  - CPU Back-to-Back 4-Word Block Write Cycles to PCI I/O (Figure 28)
  - CPU Back-to-Back 4-Word Block Read Cycles from PCI I/O (Figure 29)
- **Controller as PCI Target**
  - PCI Master Word Write Cycles to Controller Memory as PCI Target, with Retry (Figure 30)
  - PCI Master Word Read Cycles from Controller Memory as PCI Target, with Retry (Figure 31)
  - PCI Master 8-Word Write Cycles to Controller Memory as PCI Target (Figure 32)
  - PCI Master 8-Word Read Cycles from Controller Memory as PCI Target (Figure 33)
  - PCI Master 16-Word Write Cycles to Controller Memory as PCI Target (Figure 34)
  - PCI Master 16-Word Read Cycles from Controller Memory as PCI Target (Figure 35)
- **DMA Transfers**
  - DMA Transfer from Controller Memory to PCI Target (Figure 36)

As in the preceding timing diagrams, the following notations are used:

A or An Address or sequential address number. Each address is also marked by a vertical dashed line.

D or Dn Data or sequential data item number

Figure 17: CPU Read Cycle from PCI Configuration Space

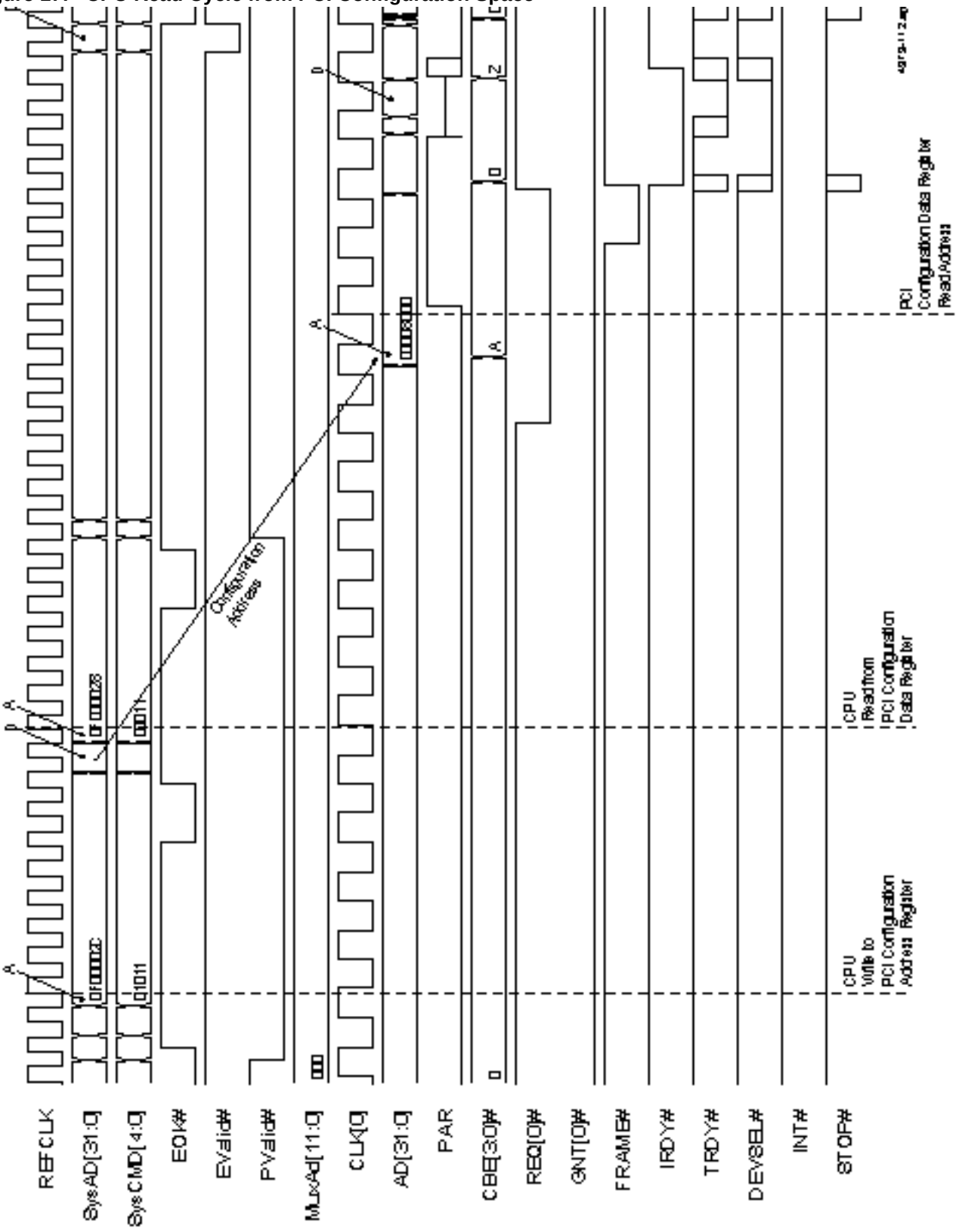


Figure 18: CPU Write and Read Cycles to/from PCI Configuration Space

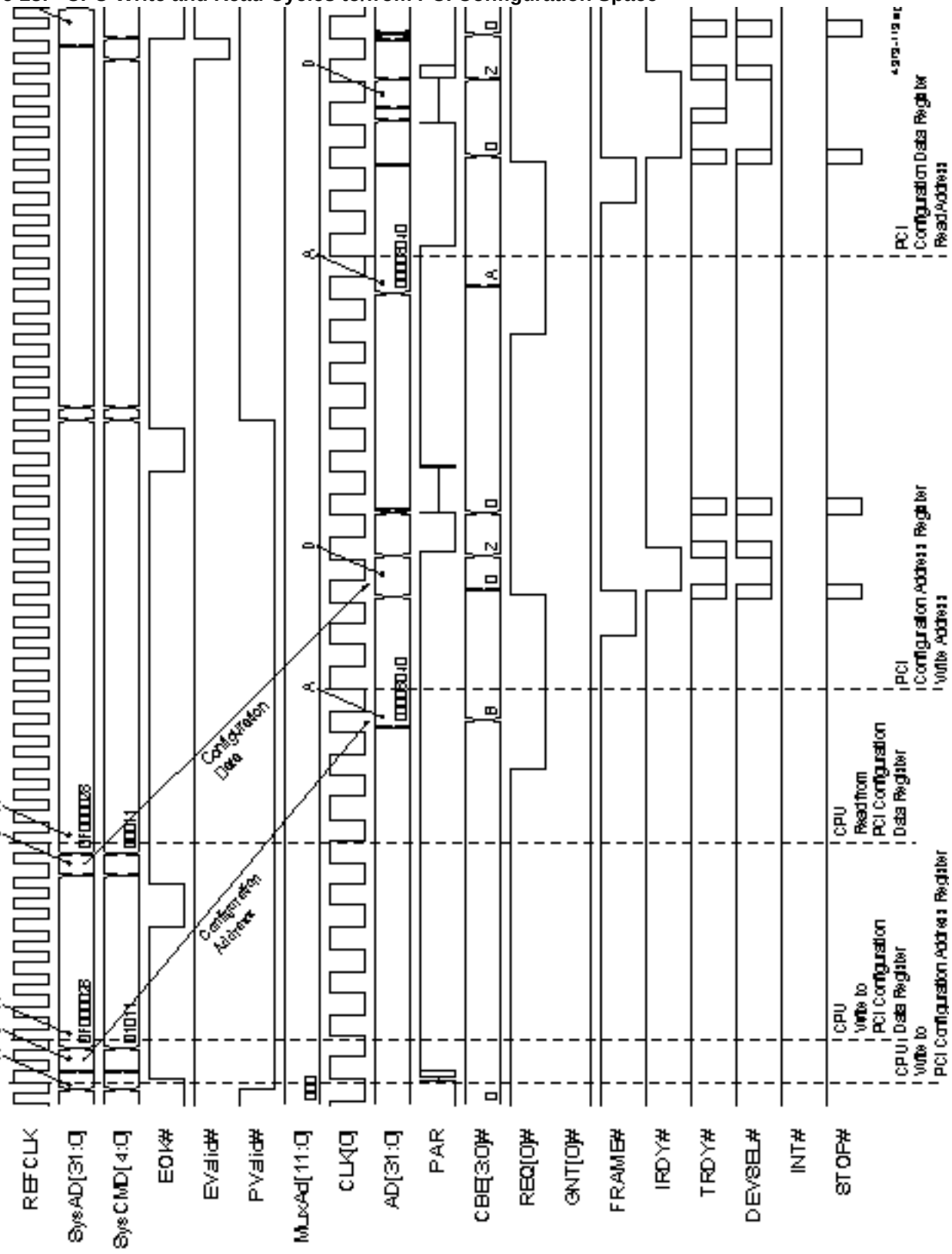


Figure 19: CPU Byte Write and Read Cycles to/from PCI Memory

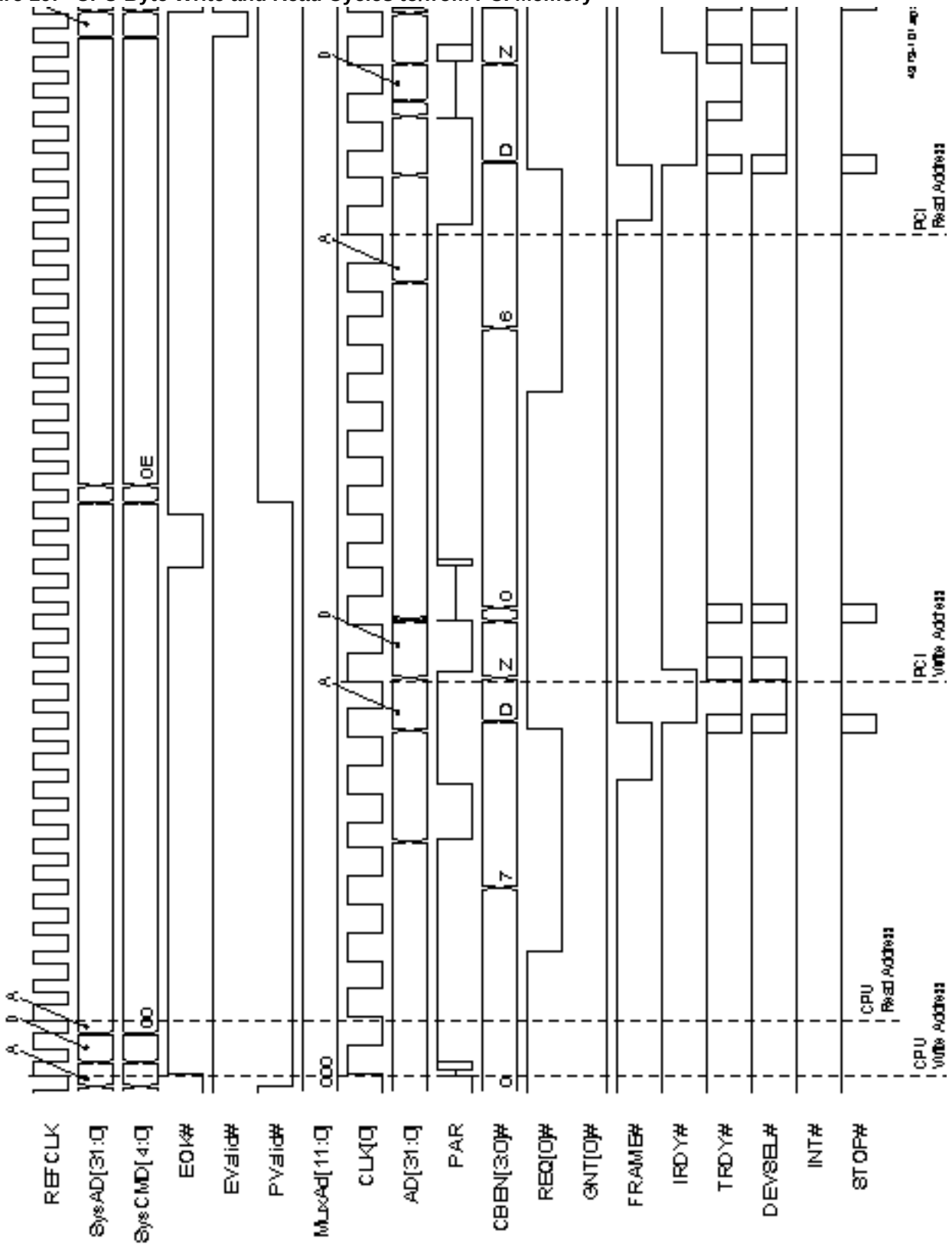


Figure 20: CPU Half-Word Write and Read Cycles to/from PCI Memory

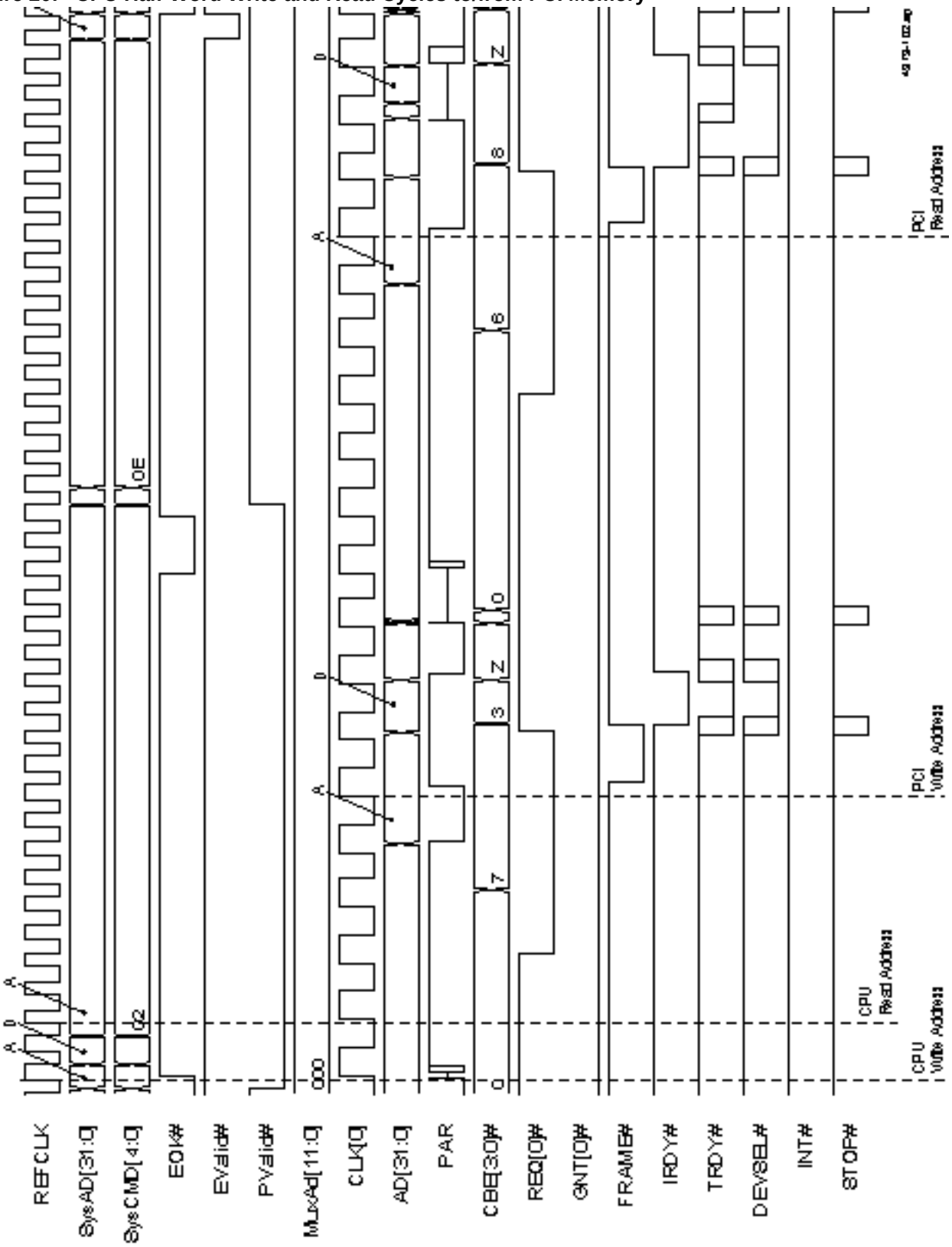


Figure 21: CPU Tri-Byte Write and Read Cycles to/from PCI Memory

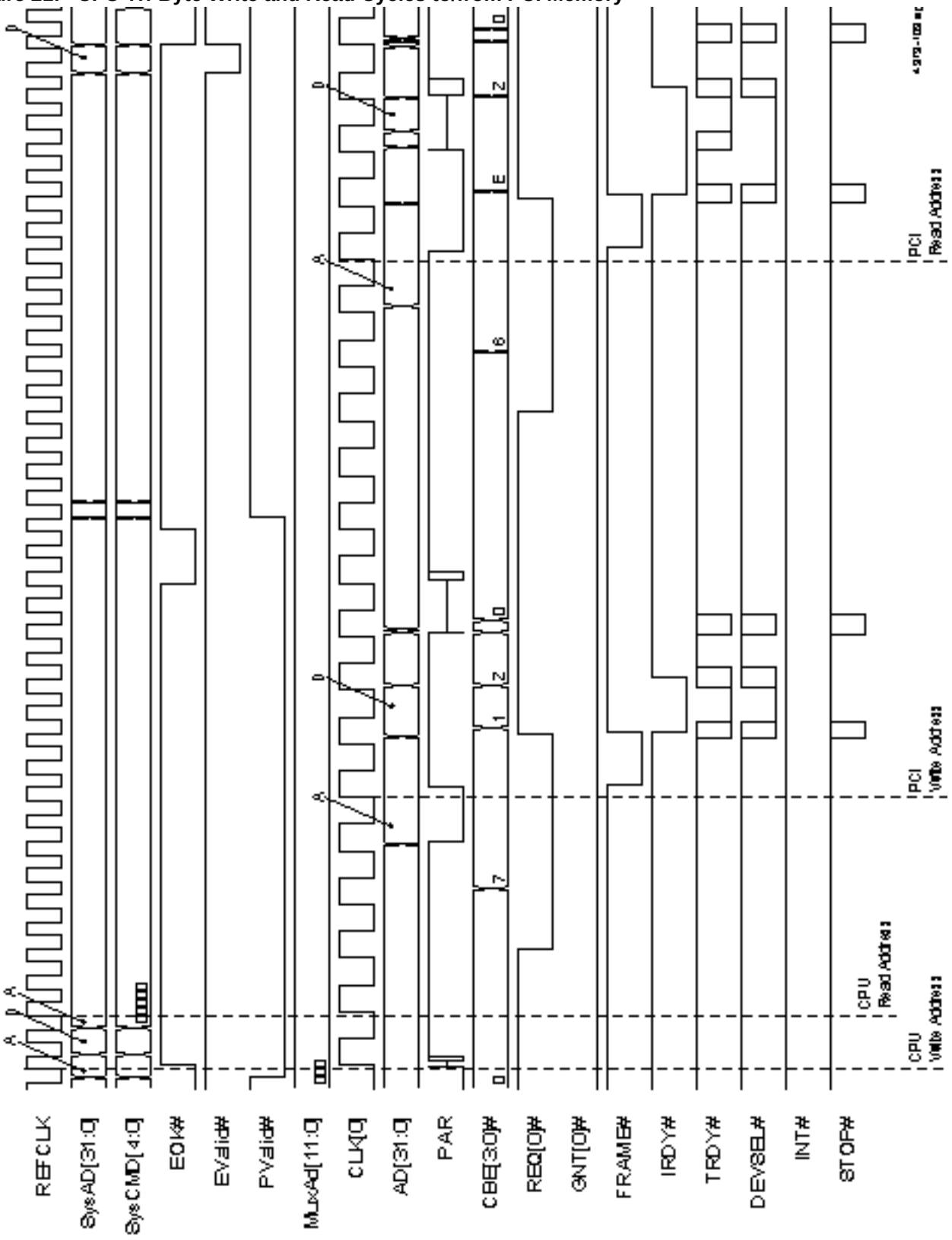


Figure 22: CPU Word Write Cycles to PCI Memory

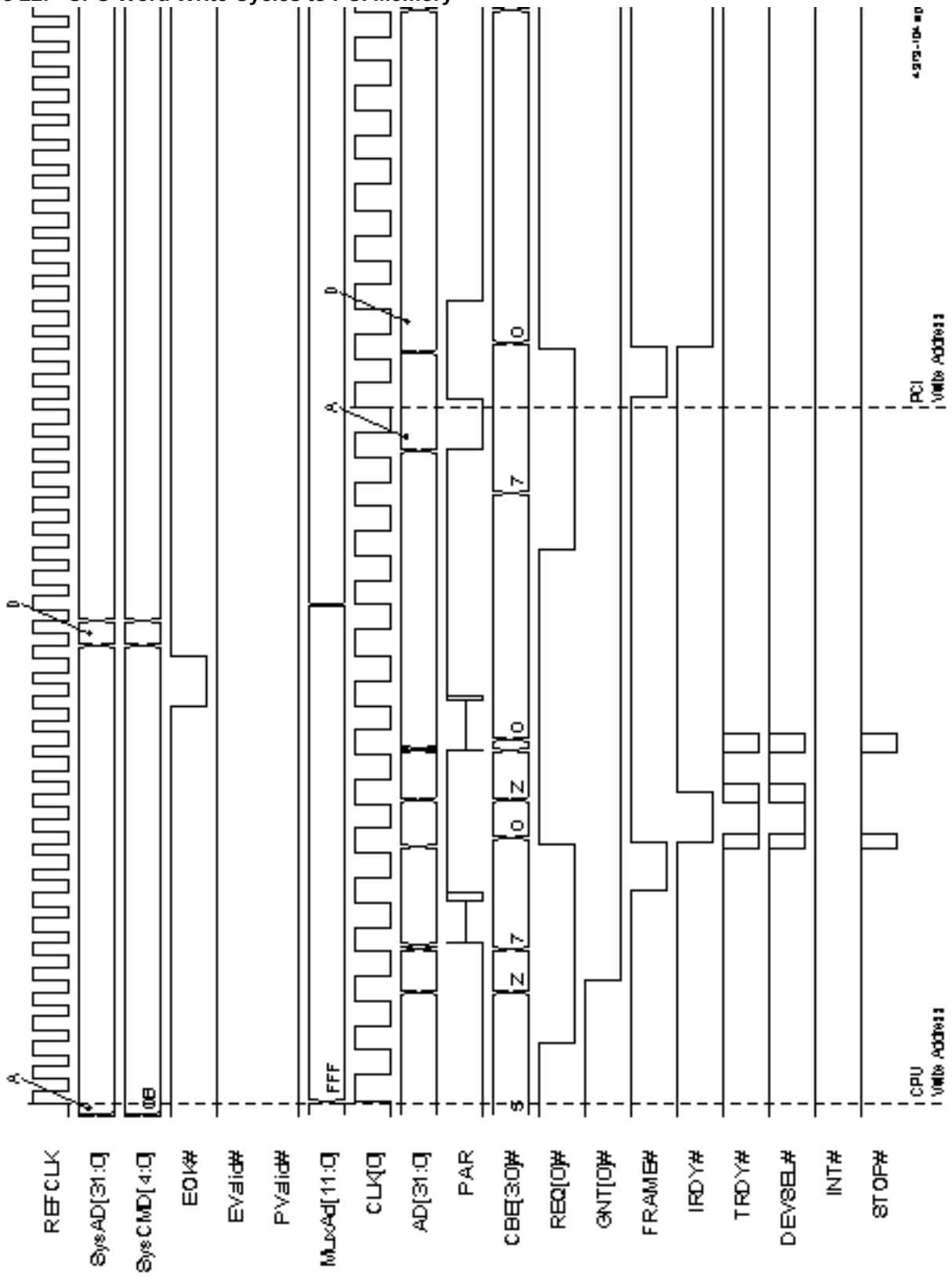


Figure 23: CPU Word Write and Read Cycles to/from PCI Memory

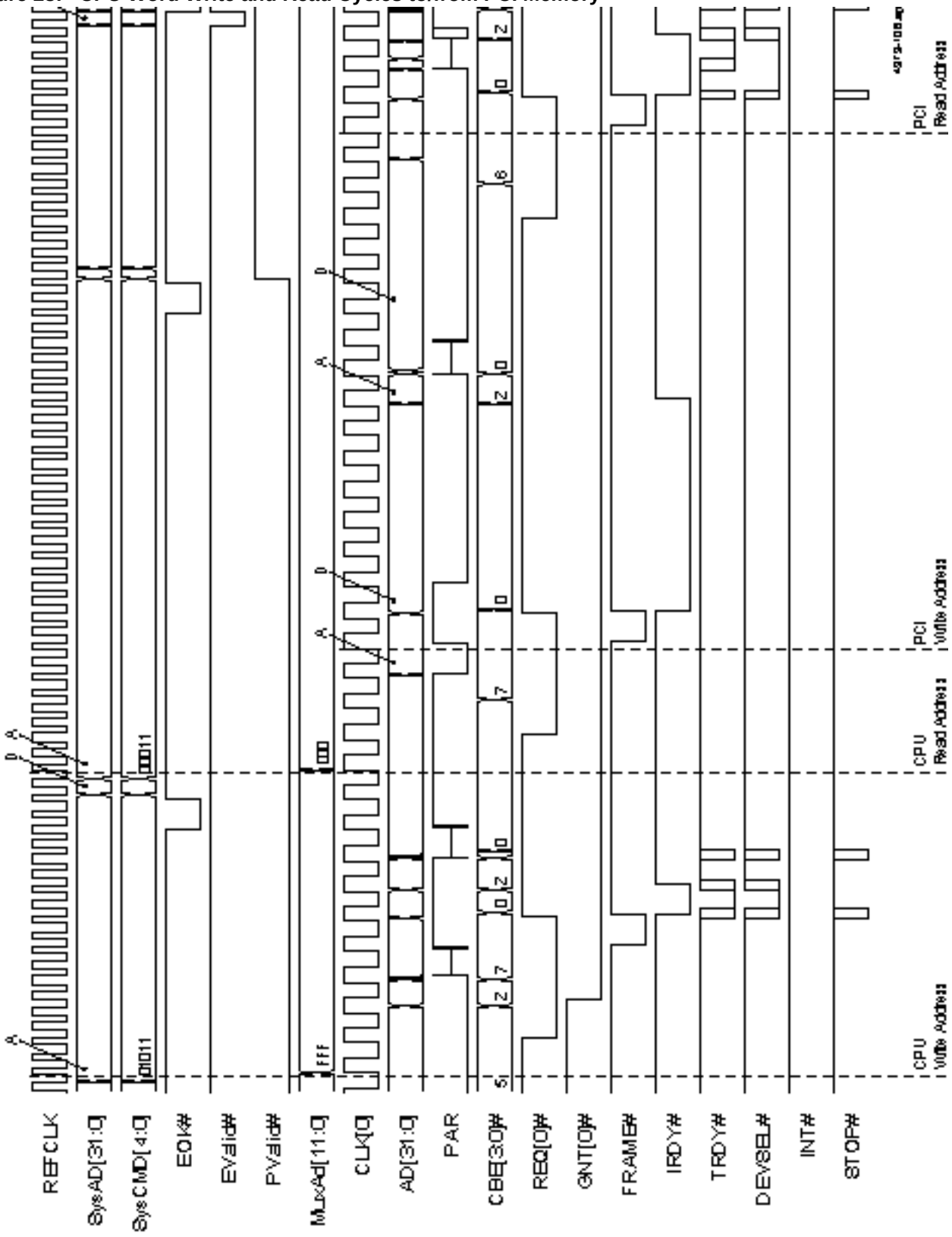


Figure 24: CPU Back-to-Back Word Write Cycles to PCI Memory

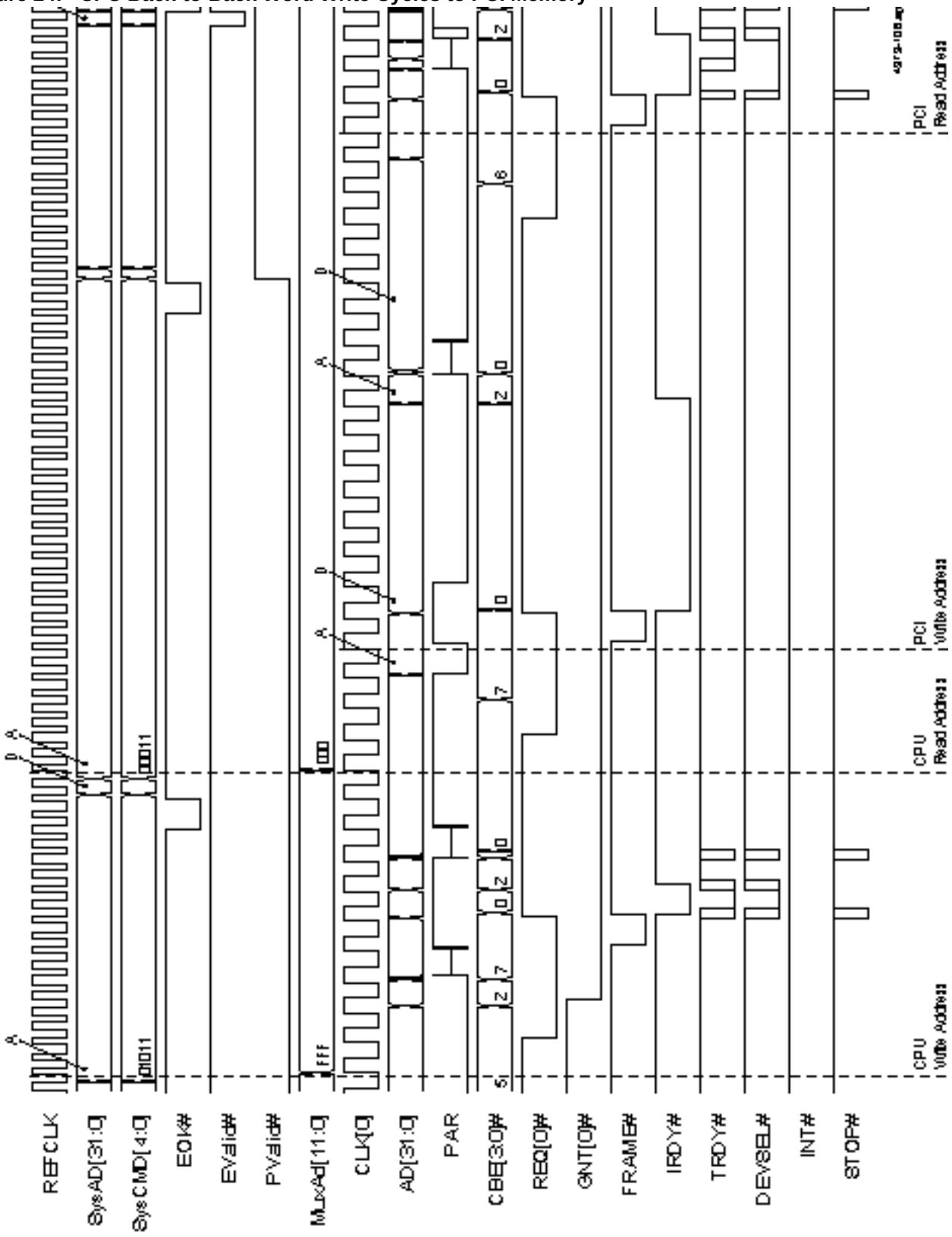


Figure 25: CPU 4-Word Block Write Cycles to PCI Memory

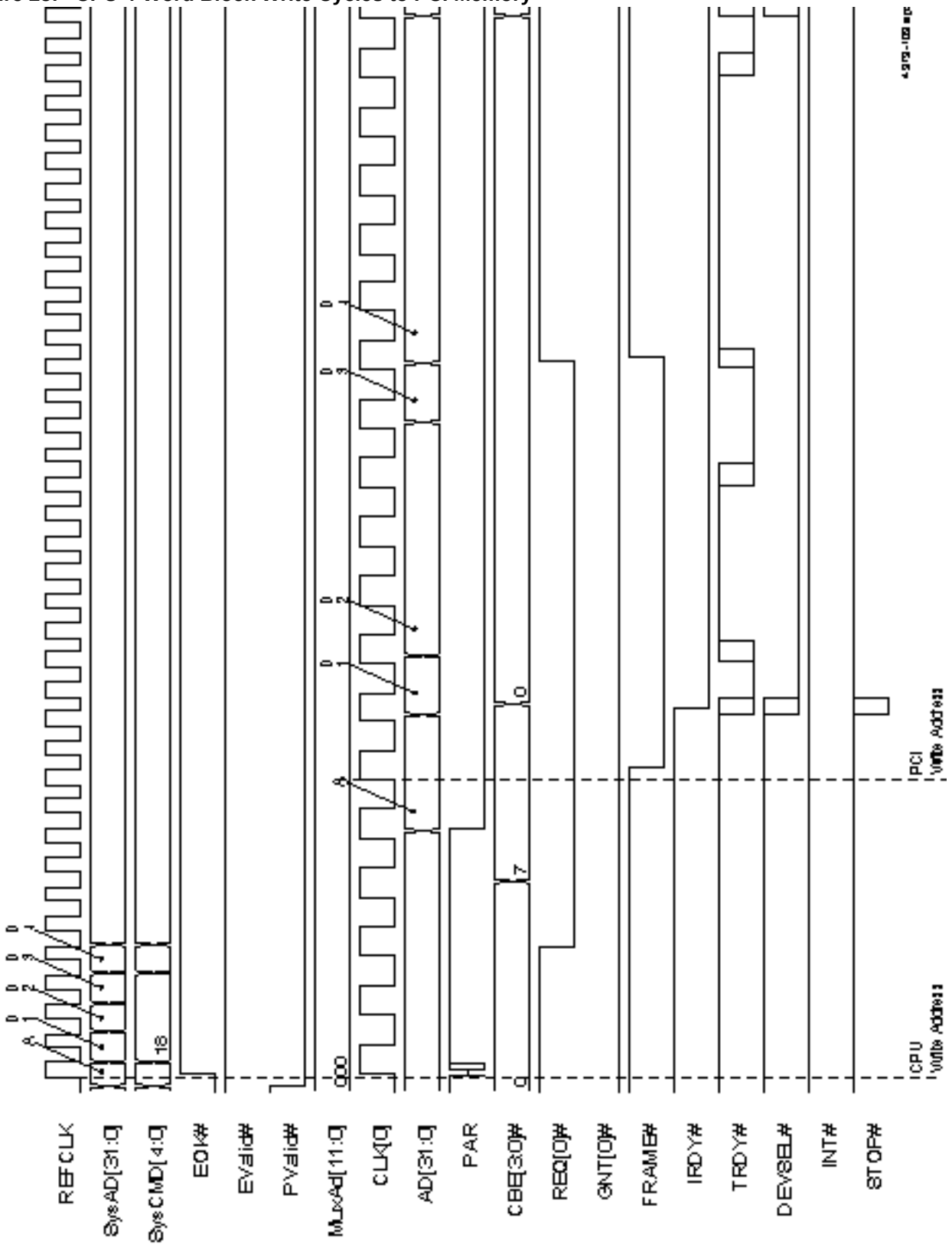


Figure 26: CPU Back-to-Back 4-Word Block Write Cycles to PCI Memory

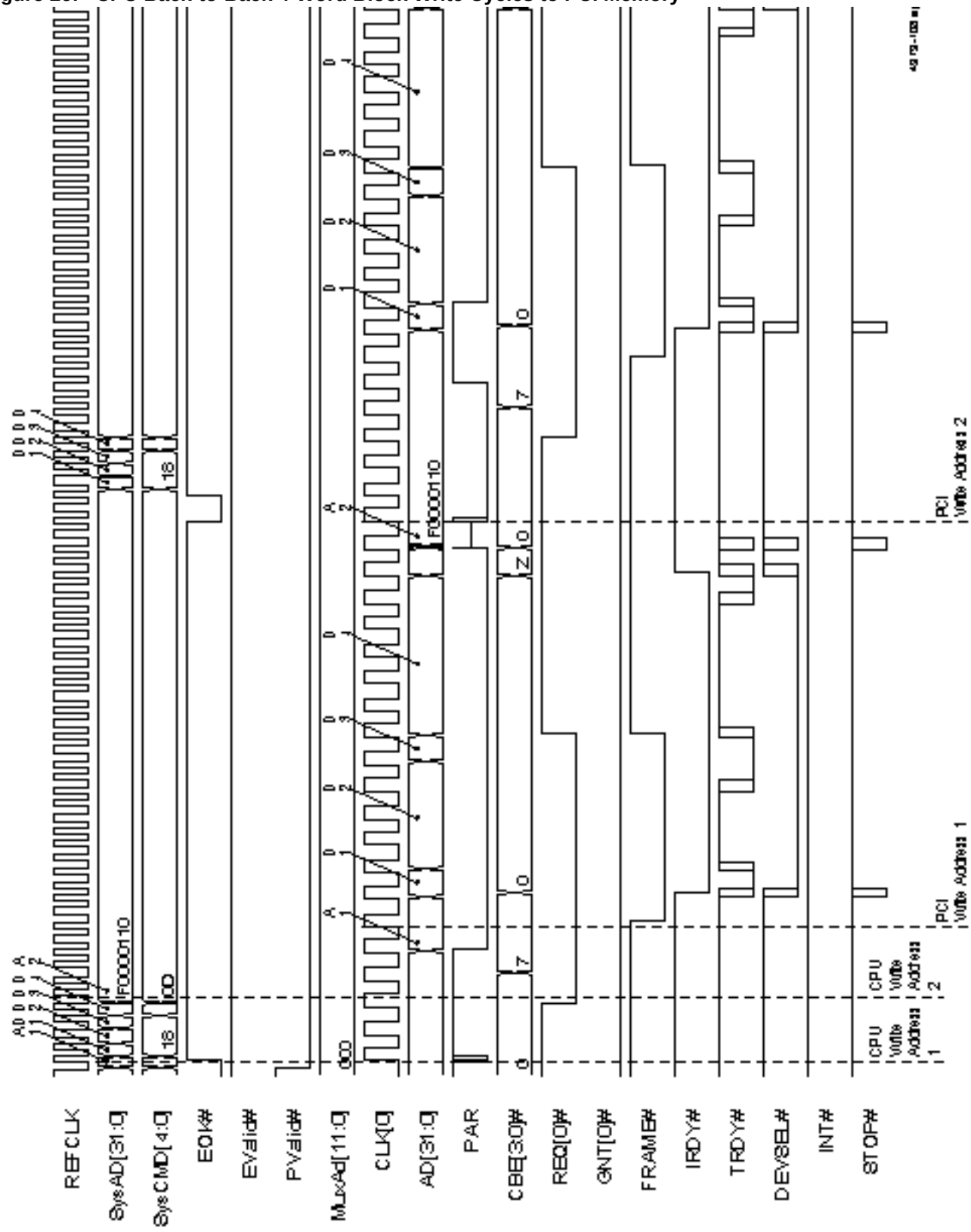


Figure 27: CPU 4-Word Block Write Cycles to PCI I/O

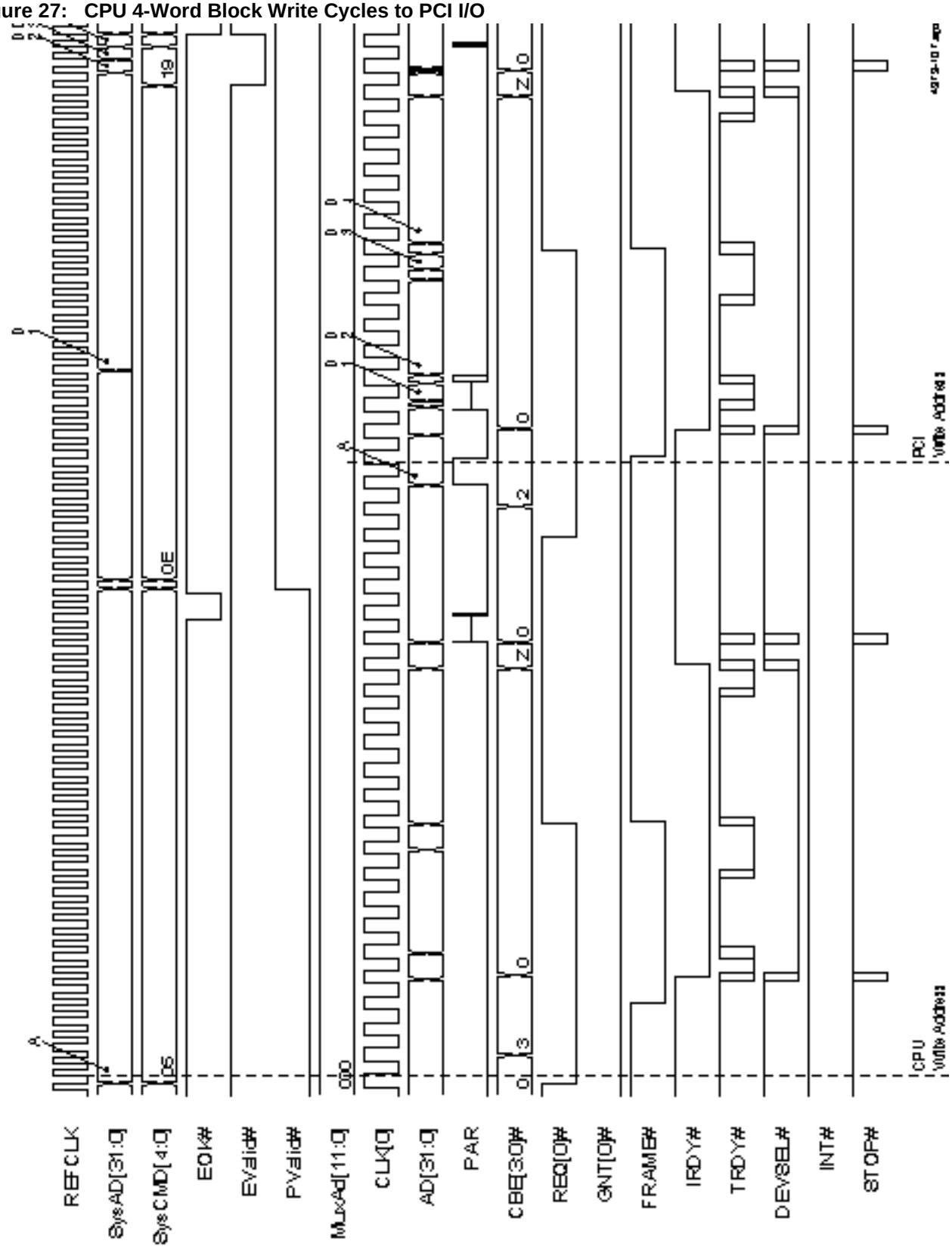


Figure 28: CPU Back-to-Back 4-Word Block Write Cycles to PCI I/O

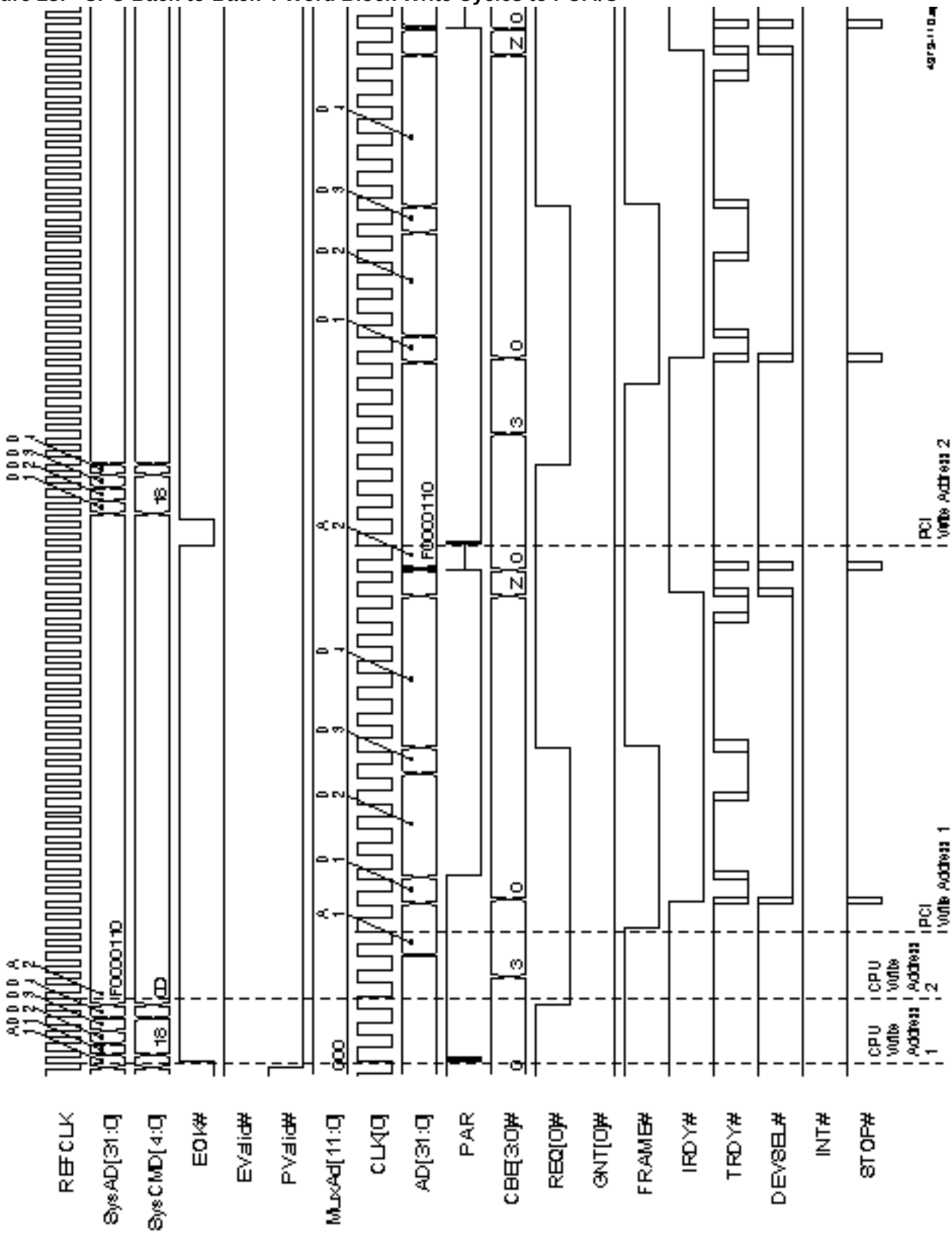


Figure 29: CPU Back-to-Back 4-Word Block Read Cycles from PCI I/O

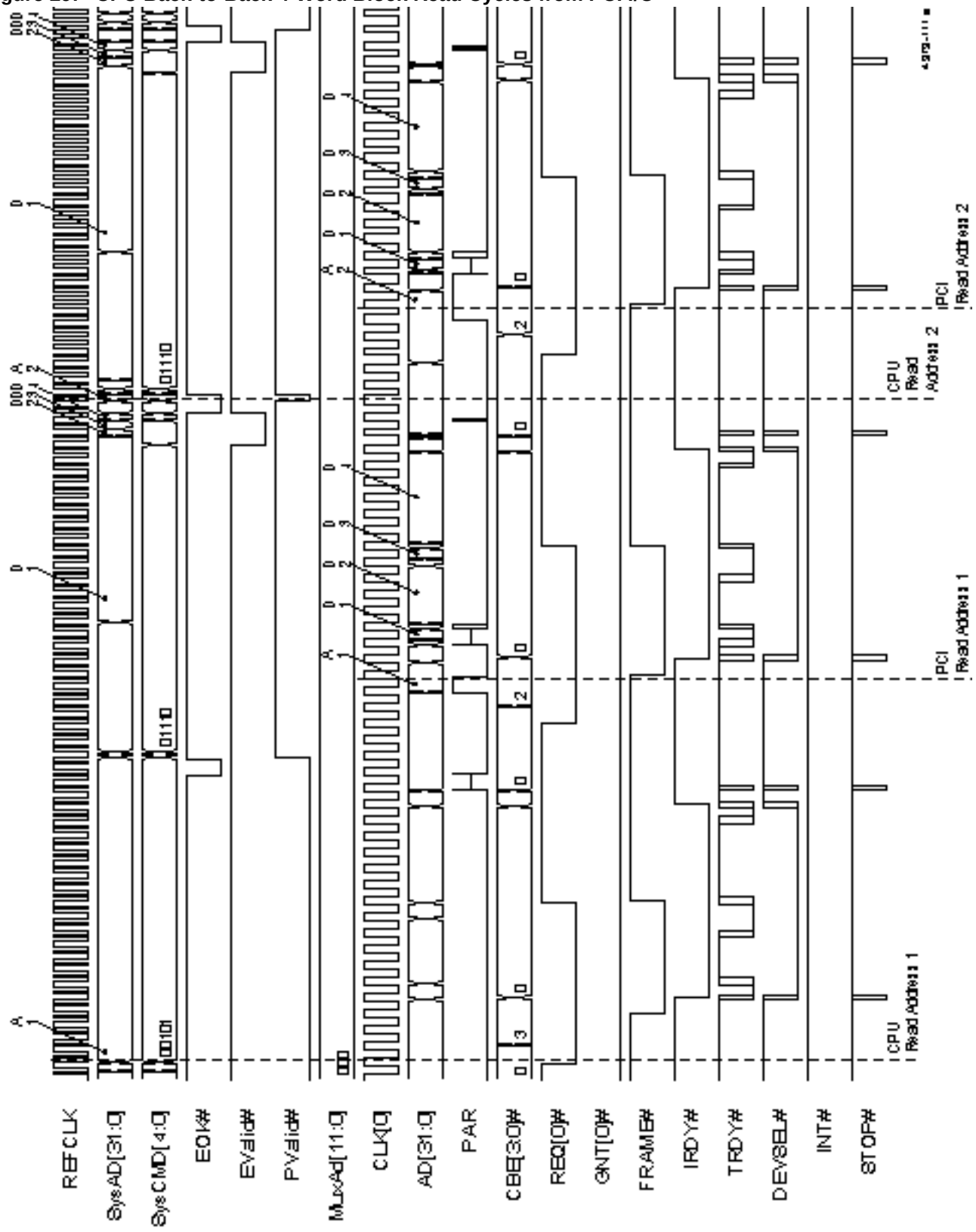


Figure 30: PCI Master Word Write Cycles to Controller Memory as PCI Target, with Retry

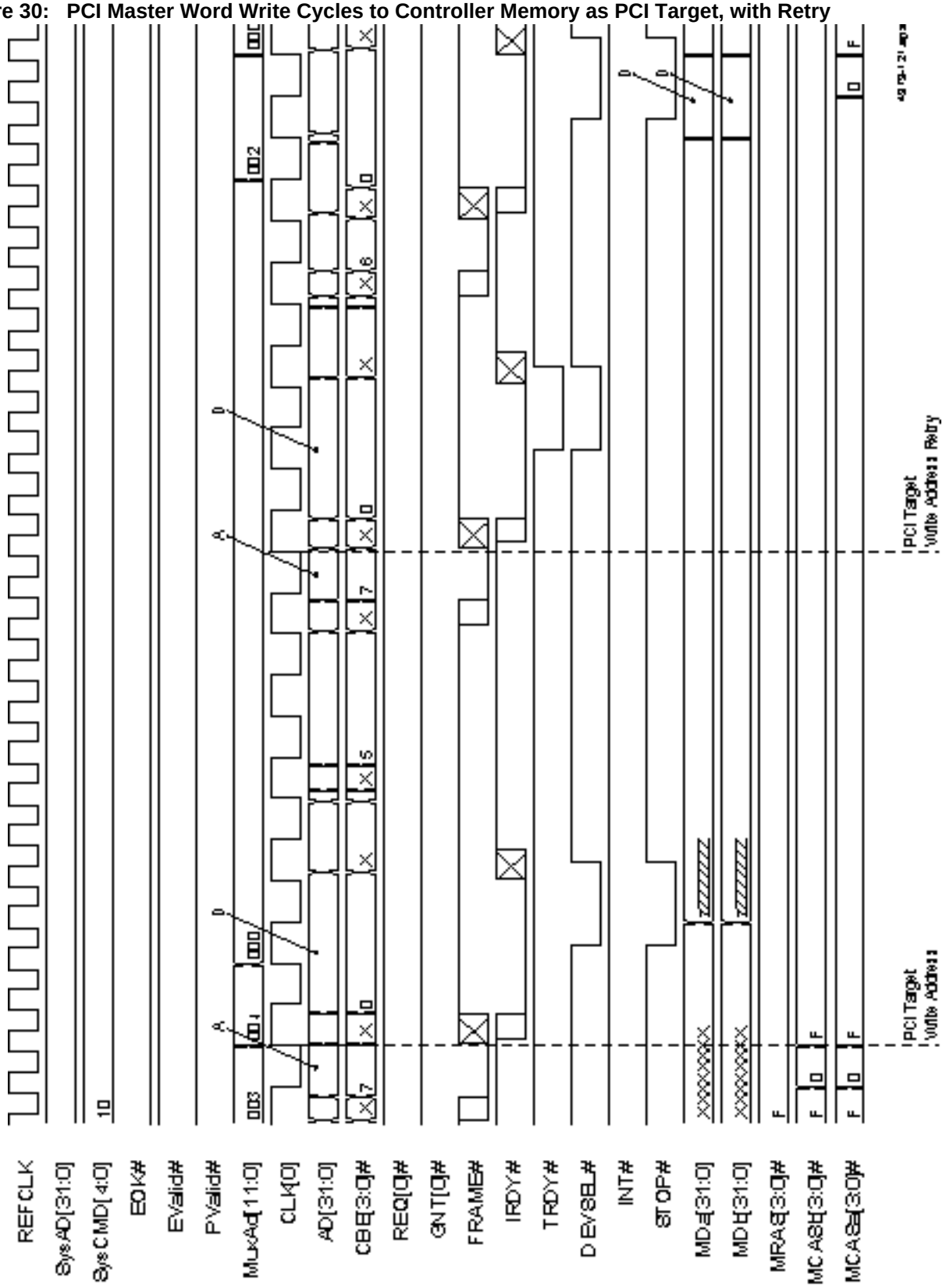


Figure 31: PCI Master Word Read Cycles from Controller Memory as PCI Target, with Retry

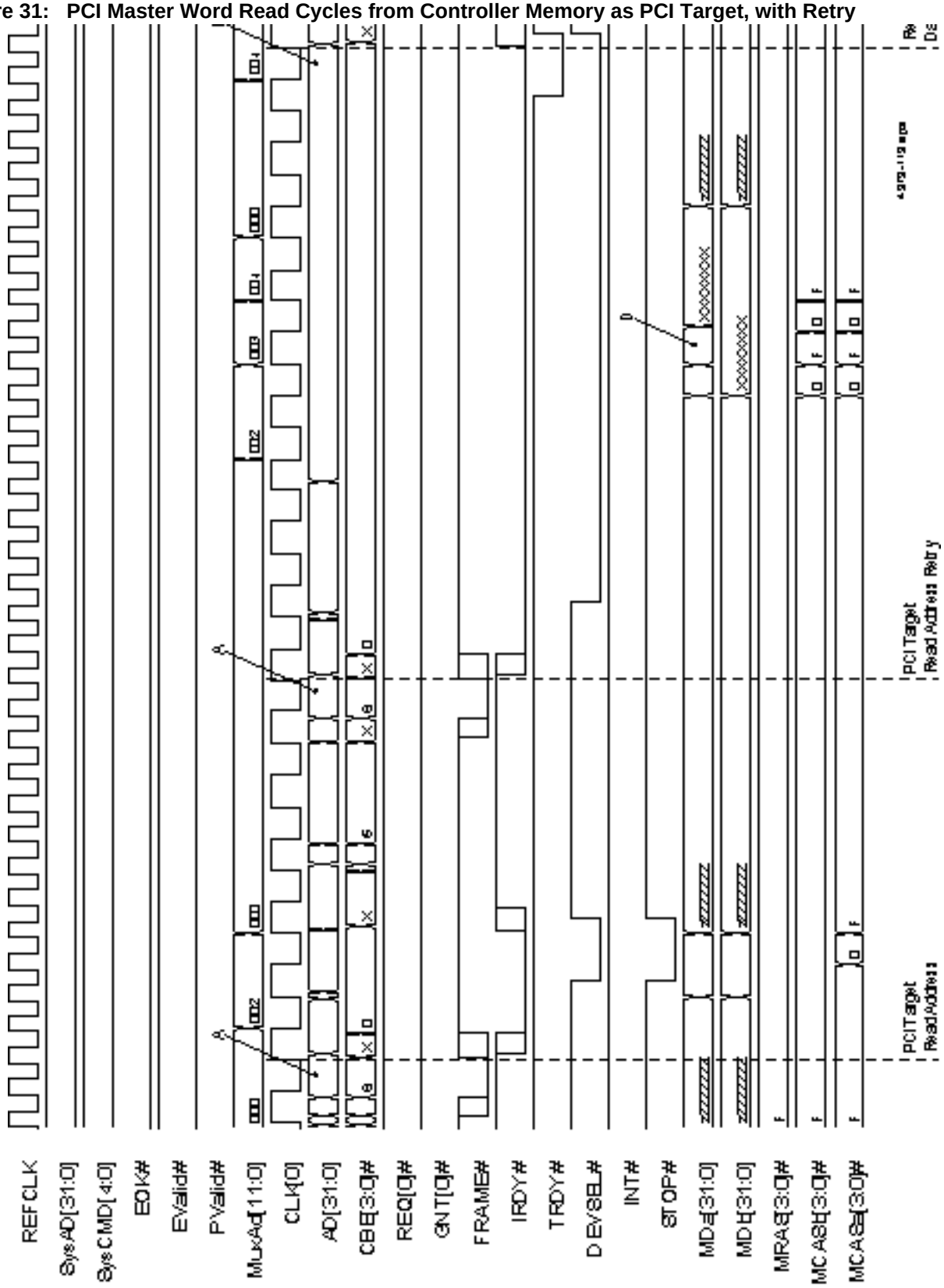


Figure 32: PCI Master 8-Word Write Cycles to Controller Memory as PCI Target

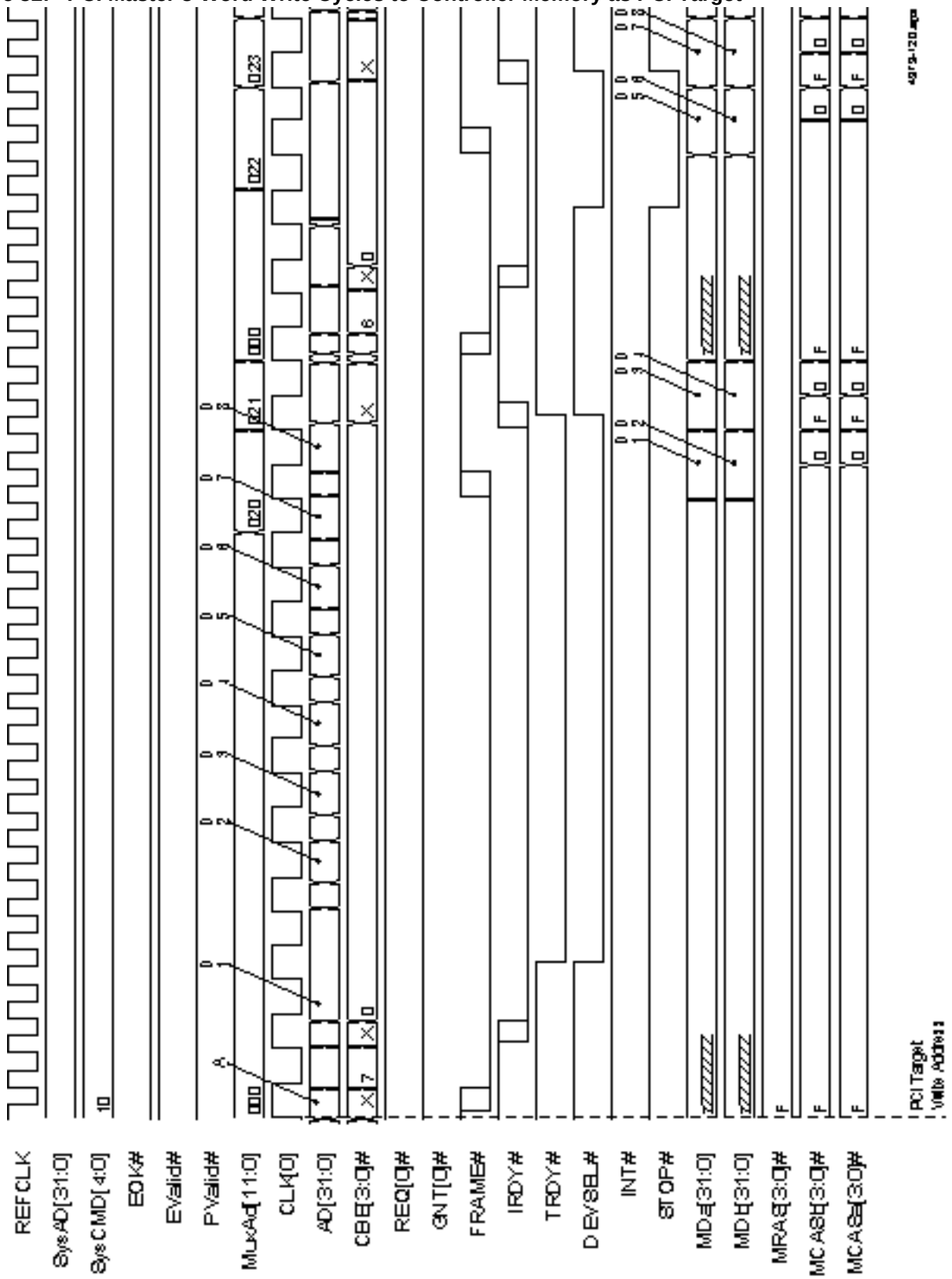


Figure 33: PCI Master 8-Word Read Cycles from Controller Memory as PCI Target

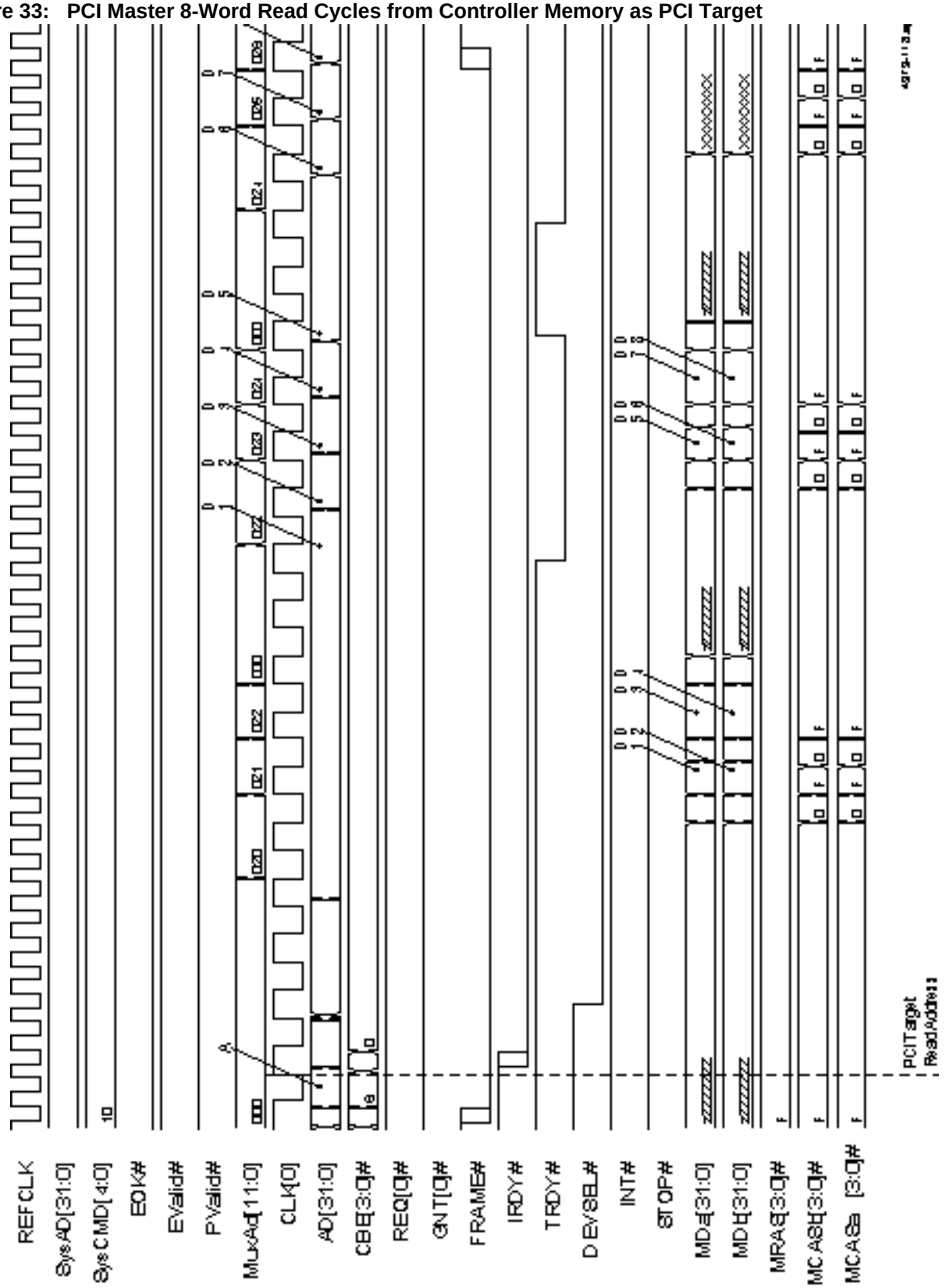


Figure 34: PCI Master 16-Word Write Cycles to Controller Memory as PCI Target

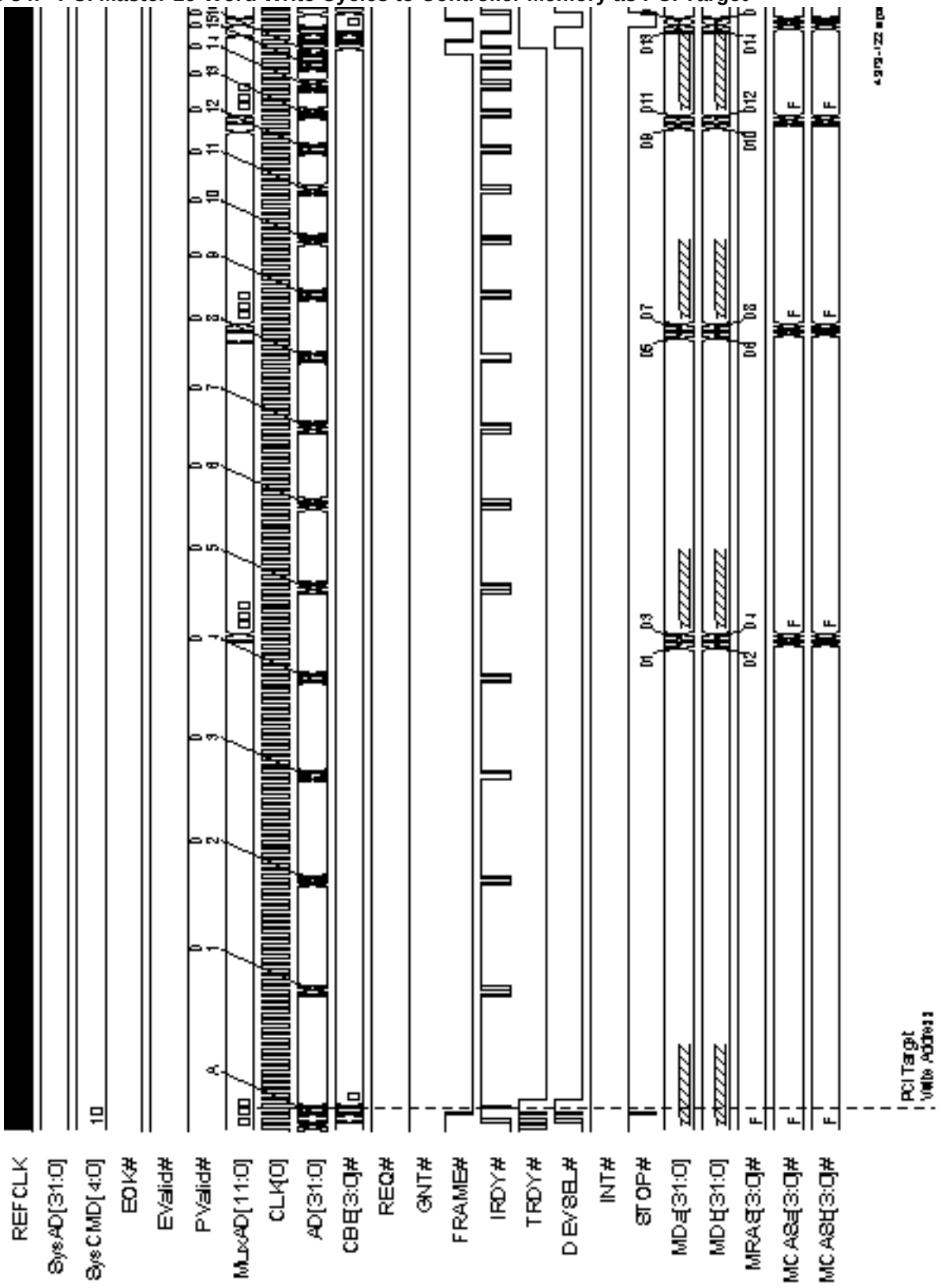


Figure 35: PCI Master 16-Word Read Cycles from Controller Memory as PCI Target

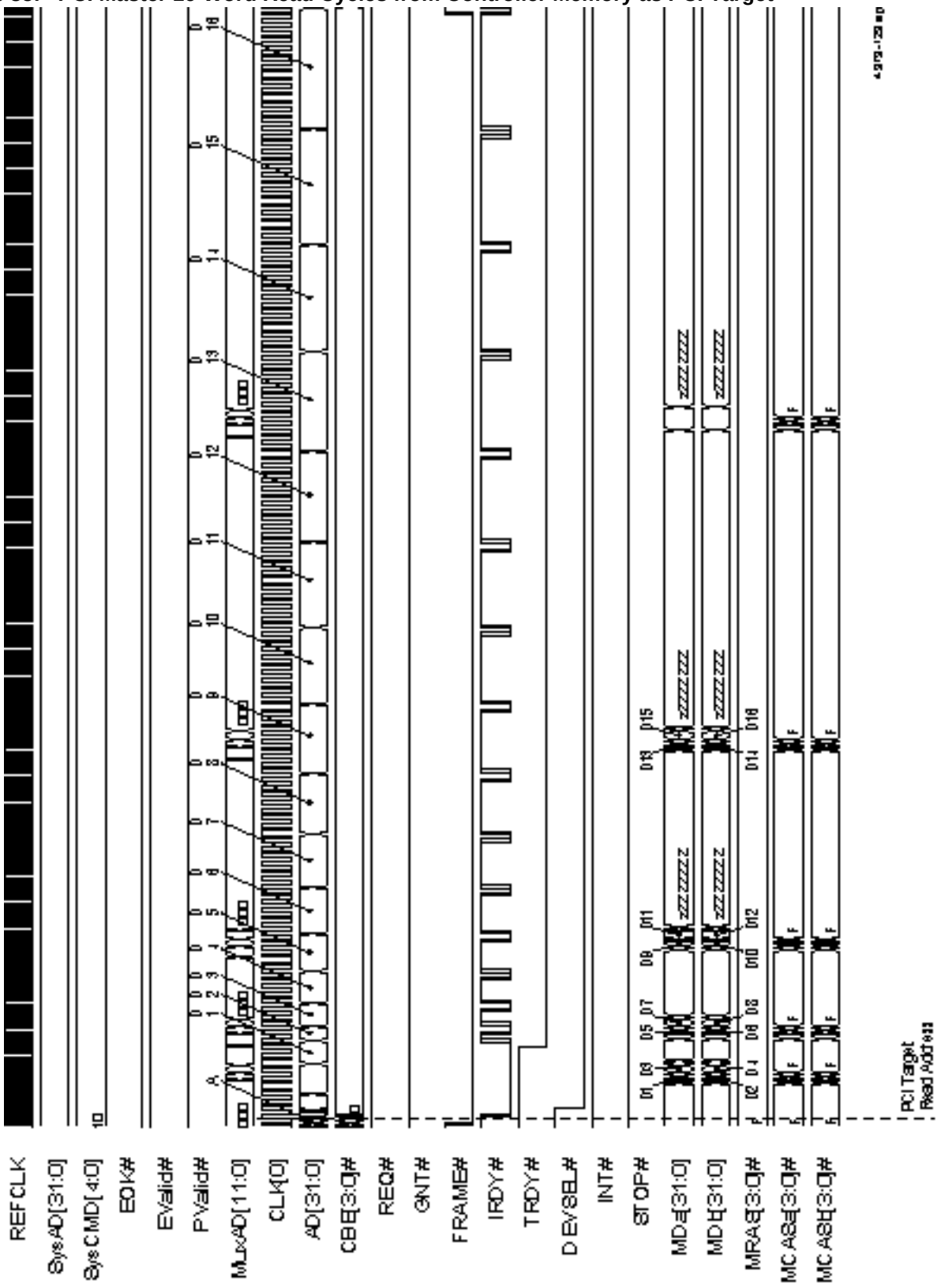
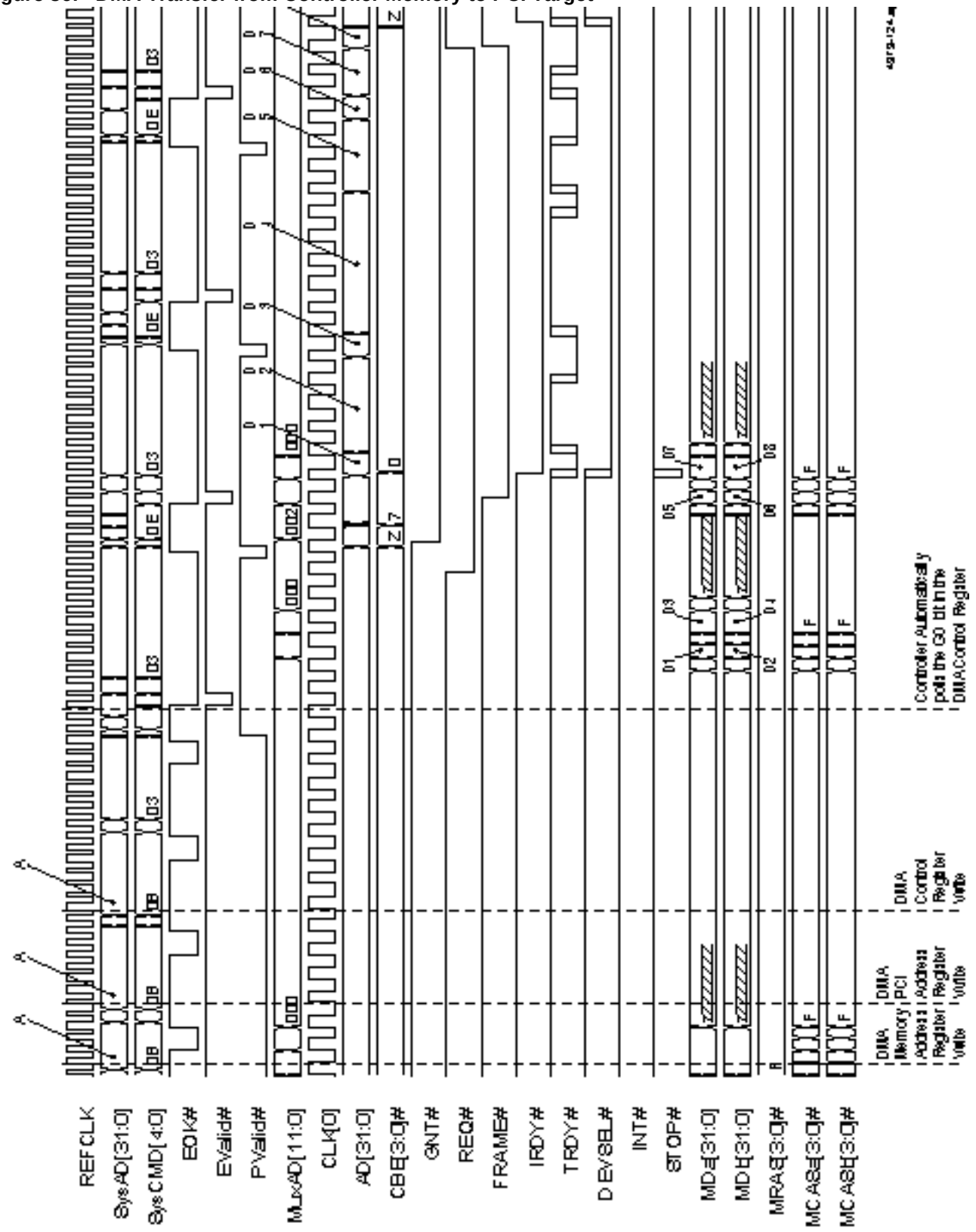


Figure 36: DMA Transfer from Controller Memory to PCI Target



13.0 Testing

The controller may be configured at power-on reset to force all signals into a high-impedance state to facilitate board-level, in-circuit testing. To do this, drive the MuxAd[7] signal high during reset.

14.0 Electrical Specifications

14.1

Absolute Maximum Ratings

Table 30: Maximum Ratings

Parameter	Maximum Rating
Storage temperature	-55°C to 125°C
Operating ambient temperature	0°C to 70°C
DC supply voltage with respect to GND	-0.5V to 4.6V
DC voltage on input pins with respect to GND	-0.5 V to 5.5V
Voltage discharged between any two pins through a 1Kohm from 100 pF	2000V
Maximum power dissipation	1 W

14.2

Operating Conditions

Table 31: Operating Conditions

Symbol	Parameter	Min.	Typical	Max.	Units
V_{DD}	All power pins	3.0	3.3	3.6	V
I_{DSS}	Static current consumption		1.0	200	µA
T_C	Case temperature	-40		+85	°C
T_j	Junction temperature	-40		+125	°C

14.3

DC Specifications

Table 32: DC Specifications for all CMOS-9 PADs

Symbol	Parameter	Min.	Typical	Max.	Units	Notes
V_{IL}	Input low voltage	0		0.8	V	
V_{IN}	Input high voltage	2.0		V_{DD}	V	
V_{oL}	Output low voltage			0.4	V	
V_{OH}	Output high voltage	2.4			V	
I_I	Input leakage current with no pull-down resistor			±10	µA	$V_I = V_{DD}$ or GND (max)
I_I	Input leakage current with 50K internal pull down	28	79	141	µA	$V_I = V_{DD}$
I_{OS}	Output short-circuit current			-250	mA	$V_{IN} = V_{IL}$ to V_{IN}
C_{IN}	Input capacitance		10	20	pF	
C_{OUT}	Output capacitance		10	20	pF	
$C_{I/O}$	I/O capacitance		10	20	pF	

14.4

AC Specifications

14.4.1

PCI CLK[3:0] Outputs

The PCI CLK[3;0] outputs must meet TTL input levels at the destinations. The following figure and table summarize the clock destination requirements, per the “PCI Local Bus Specification.” See Section 9.0 for details about clock distribution and routing.

Figure 37: PCI CLK Input Waveform at Destinations

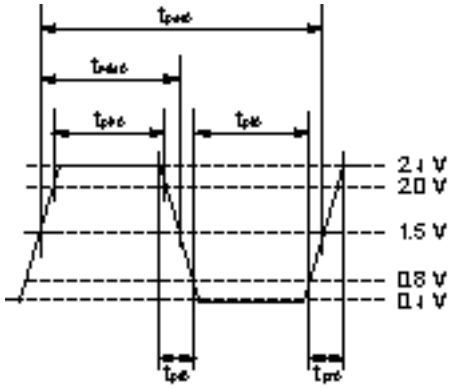


Figure 38: CLK[3:0] Versus MasterClock Output Skew at Controller Pins

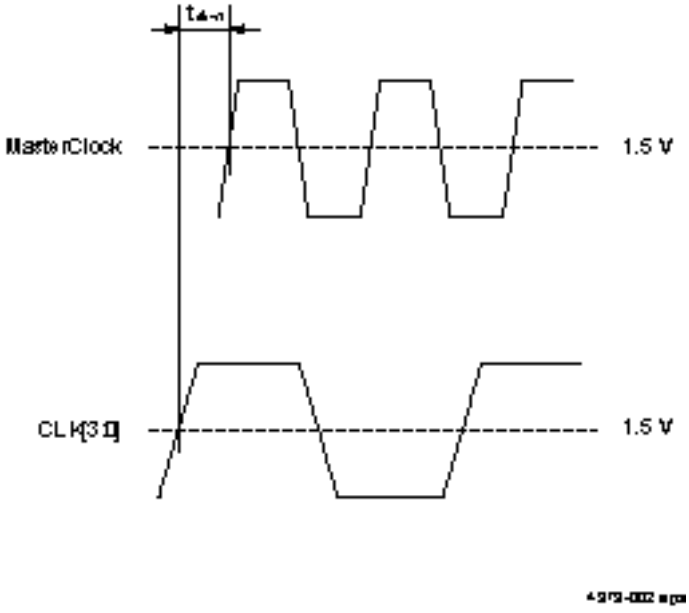


Table 33: CLK[3:0] and MasterClock Destination Timing Requirements

Symbol	Parameter	Min.	Max.	Units	Notes
Tper0	CLK[3;0] period	30		ns	
Tmid0	CLK[3;0] mid time	0.4 Tper0	0.6 Tper0	ns	
Tph0	CLK[3;0] high time	11		ns	
Tpl0	CLK[3;0] low time	11		ns	
Tpr0	CLK[3;0] rise time	1	8	ns	
Tpf0	CLK[3;0] fall time	1	8	ns	
Tjit0	Jitter CLK[n] to CLK[m] at controller output signals	0	0	ps	Loading on all clocks is exactly identical.
Tskw0	Skew CLK[n] to CLK[m] at destinations	0	1	ns	This includes +-200ps of system jitter and 800ps of system skew.
Tskw1	Skew CLK[3;0] at controller to MasterClock at controller	0	1		CLK[3;0] outputs must always be even or ahead of the MasterClock output when both are measured at controller signals.

14.4.2

PCI Inputs, Outputs, and Input/Outputs

The controller can be used in both 3.3- and 5-volt PCI systems. To accommodate both voltages, consideration must be given to switching and signaling levels, per the “PCI Local Bus Specification.” Table 34 reproduces Table 4-7 of the “PCI Local Bus Specification,” with substitutions made for V_{CC} minimums and maximum for the controller. Timing measurements on the controller PCI pads are taken with respect to a V_{TEST} voltage of 1.5 volts. Setup and hold input times are measured from the 1.5-volt level on the rising edge of CLK[3:0] to the 1.5-volt level on the signal. Likewise, minimum and maximum output valid times are measured from the 1.5-volt level on the rising edge of CLK[0] at the controller to the 1.5-volt level on the signal at the controller.

Table 34: Timing Reference Voltages

Symbol	5 V PCI Signalling Requirement	3.3 V PCI Signalling Requirement			Controller Universal PCI Timing Reference Values
		PCI Specification	Vcc=3.0	Vcc=3.6	
Vth	2.4 V	0.6 Vcc	1.8 V	2.16 V	2.4 V
Vtl	0.4 V	0.2 Vcc	0.6 V	0.72 V	0.4 V
Vtest	1.5 V	0.4 Vcc	1.2 V	1.44 V	1.5 V
Vstep1	N/A	0.285 Vcc	.86 V	1.03 V	0.285Vcc
Vstep2	N/A	0.615 Vcc	1.85 V	2.15 V	0.615Vcc
Vmax	2.0 V	0.4 Vcc	1.2 V	1.4 V	2.0 V

Table 35: PCI Signal Timing Design Budget [a]

Symbol	Parameter	Min.[1]	Max.[1]	Units
Tval	CLK[0] to signal valid delay: all but REQ[3:0]# and GNT[3:0]#	3.4[b]	14.4	ns
Tval(ptp)	CLK[0] to signal valid delay: REQ[3:0]# and GNT[3:0]#	3.4[b]	14.4	ns
Ton	Float to active delay	3.4[b]		ns
Toff	Active to float delay		31.4	ns
Tsu	Input setup to CLK[0]: all but REQ[3:0]# and GNT[3:0]#	6.6		ns
Tsu(ptp1)	Input setup to CLK[0]: GNT[3:0]#	6.6		ns
Th	Input hold time from CLK[0]	3.4		ns
Trst-off	Reset active to output float delay (all output drivers)		43.4	ns

Table 35: PCI Signal Timing Design Budget [a] (Continued)

Symbol	Parameter	Min.[1]	Max.[1]	Units
Trst	Reset active time after power stable	1		ms
Trst-clk	Reset active time after CLK stable	100		us
Ttst	Test active to test output state (all output drivers)		43.4	ns

a. The PCI timing here is slightly different from the PCI specification because the controller is the source of the PCI clock. A controller ASIC meeting this timing budget will be PCI compliant in a system that uses the PCI clocking scheme described in Section 9.0. All timing is measured with respect to CLK[0] output signal at the controller.

b. This minimum has been made 1ns too generous to provide 1 ns hold time to all PCI devices.

14.4.3

CPU Interface

All CPU interface signals are measured at the controller with respect to the 1.5 volt level of the rising edge of the MasterClock output signal at the controller. Table 36 shows the CPU timing budget for the controller's CPU interface. The controller drives the MasterClock output to valid CPU MasterClock input levels, as shown in Figure 39. For other CPU interface signals, the controller meets standard TTL input and output level requirements.

Figure 39: MasterClock Input Waveform at CPU

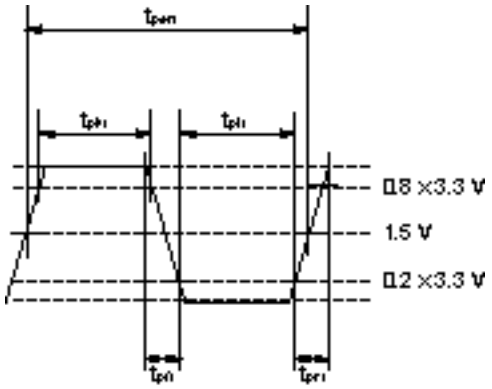


Table 36: CPU Interface Signal Timing

Symbol	Parameter	Min.	Max.	Units
Tper1	MasterClock period	15	50	ns
Tph1	MasterClock high time	4		ns
Tpl1	MasterClock low time	4		ns
Tpr1	MasterClock rise time	1	3	ns
Tpf1	MasterClock fall time	1	3	ns
Tov0	MasterClock at the controller to output valid at the controller	2.2	10.5	ns
Trst-off	Reset active to output float delay (all output drivers)		43.4	ns
Ttst	Test enable active to output float delay (all output drivers)		43.4	ns
Tis0	Input setup at the controller to MasterClock at the controller	5.8		
Tih0	Input hold at the controller to MasterClock at the controller	1.5		

14.4.4
Memory Interface

Table 37 shows the timing budget for the controller's memory interface.

Table 37: DRAM Timing Specifications

Parameter	Symbol	70ns Fast Page-Mode		60 ns EDO		28F016XD - 85		Units
		MIN	MAX	MIN	MAX	MIN	MAX	
Cycle time	tRC(R)	130		110		95		ns
Cycle time	tRC(W)	130		110		75		ns
Read-modify-write cycle time	tRMW	185		160				ns
Access from RAS	tRAC		70		60		85	ns
Access from CAS	tCAC		20		15		35	ns
Access from Column	tAA		35		30		65	ns
RAS to CAS delay	tRCD(R)	20	50	20	45	15	50	ns
RAS to CAS delay	tRCD(W)	20	50	20	45	15	15	ns
RAS to COL delay	tRAD	15	35	15	30	15	15	ns
RAS precharge	tRP	50		40		10		ns
RAS pulse width	tRAS(R)	70		60		86	INF.	
RAS pulse width	tRAS(W)	70		60		65	INF.	ns
RAS hold time	tRSH(R)	20		15		30		ns
RAS hold time	tRSH(W)	20		15		50		ns
CAS pulse width	tCAS(R)	20		15		35	INF.	ns
CAS pulse width	tCAS(W)	20		15		50	INF.	ns
CAS hold time	tCSH(R)	70		50		85		ns
CAS hold time	tCSH(W)	70		50		65		ns
ROW setup time	tASR	0		0		0		ns
ROW hold time	tRAH	10		10		15		ns
COL setup time	tASC	0		0		0		ns
COL hold time	tCAH	15		10		20		ns
COL to RAS lead time	tRAL	35		30		15		ns
ColAddress Hold from RAS	tAR	55		50		35		ns
Read cmd setup time	tRCS	0		0		5		ns
Read cmd hold time (RAS)	tRRH	10		10		0		ns
Read cmd hold time (CAS)	tRCH	0		0		0		ns
WE setup time	tWCS	0		0		0		ns
WE hold time	tWCH	15		10		15		ns
WE hold time (RAS)	tWCR	55		50		30		ns
WE pulse width	tWP	15		10		15		ns
WE to RAS lead time	tRWL	20		15		50		ns
WE to CAS lead time	tCWL	20		15		50		ns
DATA setup time	tDS	0		0		0		ns
DATA hold time	tDH	15		10		15		ns
DATA hold from RAS	tDHR	55		50		30		ns
Output Turn-off	tOFF	0	20	0	15		30	ns
RAS to WE delay	tRWD	100		90		115		ns
CAS to WE delay	tCWD	50		50		65		ns
COL to WE delay	tAWD	65		60		100		ns
CAS to RAS precharge	tCRP	10		10		10		ns
CAS precharge time	tCP	10		10		15		ns
Transition time	tT	3	50	3	50	2	4	ns

Table 37: DRAM Timing Specifications (Continued)

Parameter	Symbol	70ns Fast Page-Mode		60 ns EDO		28F016XD - 85		Units
		MIN	MAX	MIN	MAX	MIN	MAX	
Refresh Period	tREF		16		16		INF.	ms
Precharge to CAS active	tRPC	10		10		10		ns
CAS setup time	tCSR	10		10		10		ns
CAS hold time	tCHR	20		20		10		ns
WE to RAS precharge time	tWRO	10		10				ns
WE to RAS hold time	tWRH	10		10		10		ns
CAS to low-Z	tCLZ	0		0		0		ns
RAS hold ref'd to OE	tROH	20		15		35		ns
Access from OE	tOEA		20		15		35	ns
OE to data delay time	tOED	20		15		30		ns
Output turnoff (OE)	tOEZ	0	20	3	15		30	ns
OE command hold time	tOEH	20		15		15		ns
Page mode cycle time	tPC(R)	45		40		65		ns
Page mode cycle time	tPC(W)	45		40		65		ns
Page mode RMW cycle time	tPRMW	100		85				ns
Access from CAS precharge	tCPA		40		35		70	ns
Page mode RAS width	tRASP(R)	70		60		85	INF.	ns
Page mode RAS width	tRASP(W)	70		60		65	INF.	ns
Page mode RAS hold from CAS	tRHCP	45		35				ns
Page mode CAS precharge to WE	tCPWD	70		60				ns
EDO cycle time	tHCP			25				ns
Output hold time from CAS	tDOH			3	15			ns
Output hold time from WE	tWEZ			3	15			ns

15.0 Pinout

The controller is packaged in an industry-standard 304-pin PQFP package. Table 38 shows the pin assignments. Figure 40 on page 106 shows the package diagram.

Table 38: Pin Assignments

Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	GND	77	V_{DD}	153	GND	229	V_{DD}
2	GND	78	V_{DD}	154	GND	230	V_{DD}
3	MDa[19]	79	MDb[14]	155	PValid#	231	AD[16]
4	MDa[20]	80	MDb[15]	156	SysAd[31]	232	AD[15]
5	MDa[21]	81	MDb[16]	157	SysAd[30]	233	CBE[2]#
6	MDa[22]	82	MDb[17]	158	EOK#	234	FRAME#
7	MDa[23]	83	MDb[18]	159	SysAd[29]	235	IRDY#
8	MDa[24]	84	MDb[19]	160	GND	236	TRDY#
9	V_{DD}	85	MDb[20]	161	SysAd[28]	237	DEVSEL#
10	GND	86	GND	162	SysAd[27]	238	STOP#
11	GND	87	V_{DD}	163	SysAd[26]	239	GND
12	MDa[25]	88	MDb[21]	164	SysAd[25]	240	PAR
13	MDa[26]	89	MDb[22]	165	GND	241	CBE[1]#
14	MDa[27]	90	MDb[23]	166	SysAd[24]	242	AD[14]
15	MDa[28]	91	MDb[24]	167	SysCmd[0]	243	AD[13]
16	MDa[29]	92	MDb[25]	168	SysCmd[1]	244	AD[12]
17	MDa[30]	93	MDb[26]	169	SysCmd[2]	245	AD[11]
18	MDa[31]	94	MDb[27]	170	EValid#	246	AD[10]
19	V_{DD}	95	GND	171	V_{DD}	247	GND
20	GND	96	GND	172	GND	248	GND
21	MCASa[0]#	97	MDb[28]	173	SysCmd[3]	249	AD[9]
22	MCASa[1]#	98	MDb[29]	174	SysCmd[4]	250	AD[8]
23	MCASa[2]#	99	MDb[30]	175	SysAd[23]	251	AD[7]
24	MCASa[3]#	100	MDb[31]	176	SysAd[22]	252	AD[6]
25	V_{DD}	101	GND	177	SysAd[21]	253	AD[5]
26	GND	102	BOE#	178	SysAd[20]	254	CBE[0]#
27	MuxAd[0]	103	BROMCS#	179	SysAd[19]	255	AD[4]
28	MuxAd[1]	104	BRAS#	180	GND	256	GND
29	MuxAd[2]	105	MRAS[0]#	181	SysAd[18]	257	AD[3]
30	MuxAd[3]	106	V_{DD}	182	SysAd[17]	258	AD[2]
31	V_{DD}	107	GND	183	GND	259	AD[1]
32	GND	108	MRAS[1]#	184	MasterClock	260	AD[0]
33	MuxAd[4]	109	MRAS[2]#	185	RST#	261	REQ[0]#
34	MuxAd[5]	110	MRAS[3]#	186	GND	262	REQ[1]#
35	MuxAd[6]	111	MWE#	187	REFCLK	263	REQ[2]#
36	MuxAd[7]	112	BWE#	188	Int#	264	REQ[3]#
37	V_{DD}	113	V_{DD}	189	V_{DD}	265	V_{DD}
38	V_{DD}	114	V_{DD}	190	V_{DD}	266	V_{DD}
39	GND	115	GND	191	GND	267	GND
40	GND	116	GND	192	GND	268	GND
41	MuxAd[8]	117	SDCS[0]	193	No Connect	269	GNT[0]#
42	MuxAd[9]	118	SDCS[1]	194	MuxAd[12]	270	GNT[1]#

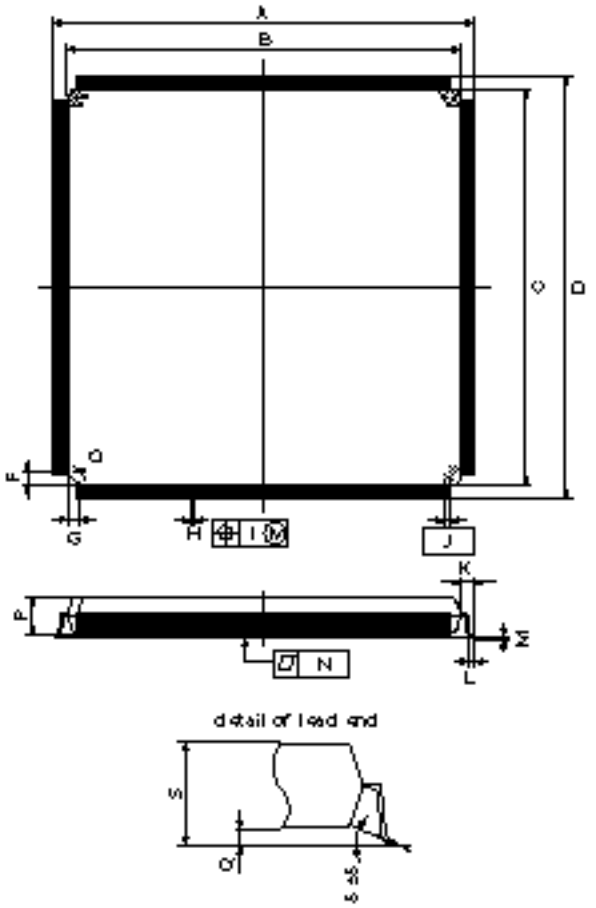
Table 38: Pin Assignments (Continued)

Pin	Name	Pin	Name	Pin	Name	Pin	Name
43	MuxAd[10]	119	SDRAS#	195	NMI	271	GNT[2]#
44	MuxAd[11]	120	SDCAS#	196	MuxAd[13]	272	GND
45	V_{DD}	121	GND	197	No Connection	273	GNT[3]#
46	GND	122	SDCLK[0]	198	GND	274	LOCK#
47	MCASb[0]#	123	GND	199	No Connection	275	PERR#
48	MCASb[1]#	124	SDCLK[1]	200	GND	276	SERR#
49	MCASb[2]#	125	GND	201	CLK[0]	277	IDSEL
50	MCASb[3]#	126	SDCLK[2]	202	CLK[1]	278	INTA#
51	V_{DD}	127	GND	203	GND	279	GND
52	GND	128	SDCLK[3]	204	CLK[2]	280	MDa[0]
53	SDCKE[0]	129	GND	205	CLK[3]	281	MDa[1]
54	SDCKE[1]	130	SysAd[16]	206	GND	282	MDa[2]
55	SDCKE[2]	131	SysAd[15]	207	AD[31]	283	MDa[3]
56	SDCKE[3]	132	SysAd[14]	208	AD[30]	284	MDa[4]
57	V_{DD}	133	GND	209	V_{DD}	285	GND
58	GND	134	GND	210	GND	286	GND
59	MDb[0]	135	SysAd[13]	211	AD[29]	287	MDa[5]
60	MDb[1]	136	SysAd[12]	212	AD[28]	288	MDa[6]
61	MDb[2]	137	SysAd[11]	213	AD[27]	289	MDa[7]
62	MDb[3]	138	SysAd[10]	214	AD[26]	290	MDa[8]
63	MDb[4]	139	SysAd[9]	215	AD[25]	291	MDa[9]
64	MDb[5]	140	SysAd[8]	216	GND	292	MDa[10]
65	MDb[6]	141	SysAd[7]	217	AD[24]	293	MDa[11]
66	V_{DD}	142	V_{DD}	218	AD[23]	294	V_{DD}
67	GND	143	GND	219	CBE[3]#	295	GND
68	MDb[7]	144	SysAd[6]	220	AD[22]	296	MDa[12]
69	MDb[8]	145	SysAd[5]	221	AD[21]	297	MDa[13]
70	MDb[9]	146	SysAd[4]	222	GND	298	MDa[14]
71	MDb[10]	147	SysAd[3]	223	AD[20]	299	MDa[15]
72	MDb[11]	148	SysAd[2]	224	AD[19]	300	MDa[16]
73	MDb[12]	149	SysAd[1]	225	AD[18]	301	MDa[17]
74	MDb[13]	150	SysAd[0]	226	AD[17]	302	MDa[18]
75	GND	151	V_{DD}	227	GND	303	V_{DD}
76	GND	152	V_{DD}	228	GND	304	V_{DD}

Note: The V_{DD} and Grounds are required.

16.0 Package

Figure 40: 304-Pin QFP



NOTE
Each lead centerline is located within 0.08 mm (0.003inch) of its true position (T.P.) at maximum material condition.

P304GL-50NMU, PMU

ITEM	MILLIMETERS	INCHES
A	42.6±0.3	1.677±0.012
B	40.0±0.2	1.575±0.008
C	40.0±0.2	1.575±0.008
D	42.6±0.3	1.677±0.012
F	1.25	0.049
G	1.25	0.049
H	0.20±0.10	0.008±0.004
I	0.08	0.003
J	0.5 (T.P.)	0.020 (T.P.)
K	1.3±0.2	0.051±0.008
L	0.5±0.2	$0.020^{+0.008}_{-0.009}$
M	0.125±0.05	0.005±0.002
N	0.10	0.004
P	3.7	0.146
Q	0.4±0.1	$0.016^{+0.004}_{-0.005}$
S	4.3 MAX.	0.170 MAX.

NEC

Some of the information contained in this document may vary from country to country. Before using any NEC product in your application, please contact a representative from the NEC office in your country to obtain a list of authorized representatives and distributors who can verify the following:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 800-366-9782
Fax: 800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.l.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, the Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 253-8311
Fax: 250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC do Brasil S.A.
Sao Paulo-SP, Brasil
Tel: 011-889-1680
Fax: 011-889-1689

VRC4373 and VR4300 are either trademarks or registered trademarks of NEC Corporation in Japan and/or other countries. MIPS is a registered trademark of MIPS Group, a division of Silicon Graphics, Inc.

NEC