21285 Core Logic for SA-110 Microprocessor

Datasheet

September 1998

Order Number: 278115-001

Contents

1 Introduction 1–1

1.1 Features 1–1
1.1.1 Power Management 1–2
1.2 System Applications 1–2

2 Signal Description 2–1

2.1 PCI Signals 2–1
2.2 SA-110 Signals 2–3
2.3 ROM Signals 2–5
2.4 SDRAM Signals 2–5
2.5 Serial Port Signals 2–7
2.6 Miscellaneous Signals 2–7
2.7 X-Bus/Arbiter Signals 2–8
2.7.1 X-Bus Selection 2–9
2.7.2 PCI Arbiter Selection 2–10
2.8 JTAG Signals 2–10
2.9 Pin State During Reset 2–11
2.10 Pin Assignment 2–12
2.11 Pins Listed in Numeric Order 2–13
2.12 Pins Listed in Alphabetic Order 2–17

3 Transactions 3–1

3.1 SA-110 Originated Transactions 3–1
3.1.1 CSR Write 3–1
3.1.2 CSR Read 3–1
3.1.3 SDRAM All-Banks Precharge 3–2
3.1.4 SDRAM Mode Register Set 3–3
3.1.5 SDRAM Write 3–3
3.1.6 SDRAM First Read 3–3
3.1.6.1 Lock 3–3
3.1.7 ROM Read 3–3
3.1.8 ROM Write 3–3
3.1.9 PCI Memory Write 3–4
3.1.10 PCI Memory Read 3–4
3.1.11 PCI I/O Write 3–4
3.1.12 PCI I/O Read 3–5
3.1.13 PCI Configuration Write 3–5
3.1.14 PCI Configuration Read 3–5
3.1.15 PCI Special Cycle 3–5
3.1.16 PCI IACK Read 3–5
3.1.17 X-Bus Write 3–5
3.1.18 X-Bus Read 3–6
3.1.19 Outbound Write Flush 3–6
3.1.20 SA-110 Cache Flush 3–6
3.1.21 Reserved Addresses 3–6
3.2 PCI Target Transactions 3–6

3.2.1 Unsupported PCI Cycles As Target 3–7
3.2.2 Memory Write to SDRAM 3–7
3.2.3 Memory Read, Memory Read Line, Memory Read Multiple to SDRAM 3–9
3.2.4 Type 0 Configuration Write 3–10
3.2.5 Type 0 Configuration Read 3–10
3.2.6 Write to CSR 3–11
3.2.7 Read to CSR 3–11
3.2.8 Write to I_2O Address 3–11
3.2.9 Read to I_2O Address 3–11
3.2.10 Memory Write to ROM 3–12
3.2.11 Memory Read to ROM 3–12
3.3 PCI Master Transactions 3–13
3.3.1 Dual Address Cycles (DAC) Support 3–13
3.3.1.1 For SA–110 Accesses 3–13
3.3.1.2 For DMA Accesses 3–14
3.3.2 Memory Write, Memory Write and Invalidate 3–14
3.3.2.1 From SA-110 3–14
3.3.2.2 From DMA 3–15
3.3.2.3 Selecting PCI Command for Writes 3–15
3.3.3 Memory Read, Memory Read Line, Memory Read Multiple 3–15
3.3.3.1 From SA-110 3–16
3.3.3.2 From DMA 3–16
3.3.3.3 Read Bursting Policy 3–16
3.3.4 I/O Write 3–16
3.3.5 I/O Read 3–17
3.3.6 Configuration Write 3–17
3.3.7 Configuration Read 3–19
3.3.8 Special Cycle 3–19
3.3.9 IACK Read 3–19
3.3.10 PCI Request Operation 3–19
3.3.11 Master Latency Timer 3–20
3.4 PCI Error Summary 3–20
3.4.1 Errors As PCI Target 3–20
3.4.1.1 Address Parity Error 3–20
3.4.1.2 Write Data Parity Error 3–21
3.4.1.3 Read Data Parity Error 3–21
3.4.2 Errors As PCI Master 3–21
3.4.2.1 Master Abort 3–21
3.4.2.2 Write Data Parity Error 3–22
3.4.2.3 Target Abort on Write 3–22
3.4.2.4 Read Data Parity Error 3–22
3.4.2.5 Target Abort on Read 3–22

4 SDRAM and ROM Operation 4–1

4.1 SDRAM Control 4–1
4.1.1 SDRAM Addresses 4–2
4.1.2 Commands 4–3
4.1.3 Parity 4–4
4.2 ROM Control 4–5
4.2.1 Addressing 4–5

4.2.2 Reads 4–7
4.2.3 Writes 4–7
4.2.4 Timing 4–7
4.2.5 Blank ROM Programming 4–8

5 SA-110 Operation 5–1

5.1 SA-110 Control 5–1
5.1.1 Address Map Partitioning 5–1
5.1.2 Byte Enables 5–2
5.1.3 SA-110 Bus Arbiter 5–2
5.2 X-Bus Interface 5–3
5.2.1 Address and Data Bus Generation 5–4
5.2.2 Device Support 5–5
5.2.3 Timing 5–5
5.3 Ordering and Deadlock Avoidance 5–6
5.3.1 Transaction Ordering 5–6
5.3.2 Ordering Rules 5–7
5.3.3 Deadlock Avoidance 5–7

6 Functional Units 6–1

6.1 PCI Bus Arbiter 6–1
6.1.1 Priority Algorithm 6–1
6.1.2 Determining Priority 6–2
6.2 DMA Channels 6–2
6.2.1 DMA Channel Operation 6–3
6.2.2 SDRAM-to-PCI Transfer 6–6
6.2.3 PCI-to-SDRAM Transfer 6–6
6.2.4 Channel Alignment 6–6
6.3 I_2O Message Unit 6–7
6.3.1 I_2O Inbound FIFO Operation 6–9
6.3.2 I_2O Outbound FIFO Operation 6–9
6.3.3 Circulation of MFAs 6–10
6.4 Timers 6–11
6.4.1 Timer 4 Application 6–11
6.5 Serial Port 6–12
6.5.1 Data Handling 6–12
6.5.2 Initialization 6–12
6.5.3 Frame Format 6–12
6.5.4 Baud Rate Generation 6–13
6.5.5 Receive Operation 6–13
6.5.6 Transmit Operation 6–13
6.5.7 Serial Port Interrupts 6–14
6.6 UART Register Definitions 6–14
6.6.1 UARTDR—Offset 160h 6–14
6.6.2 RXSTAT—Offset 164h 6–15
6.6.3 H_UBRLCR—Offset 168h 6–16
6.6.4 M_UBRLCR—Offset 16Ch 6–17
6.6.5 L_UBRLCR—Offset 170h 6–17
6.6.6 UARTCON-Offset 174h 6–18
6.6.7 UARTFLG—Offset 178h 6–18

7 Registers 7–1
7.1 PCI Configuration Space Registers 7–1
7.1.1 Vendor ID Register—Offset 00h 7–2
7.1.2 Device ID Register—Offset 02h 7–2
7.1.3 Command Register—Offset 04h 7–3
7.1.4 Status Register—Offset 06h 7–5
7.1.5 Revision ID Register—Offset 08h 7–6
7.1.6 Class Code Register—Offset 0Ah 7–6
7.1.7 Cache Line Size Register—Offset 0Ch 7–6
7.1.8 Latency Timer Register—Offset 0Dh 7–6
7.1.9 Header Type Register—Offset 0Eh 7–6
7.1.10 BIST Register—Offset 0Fh 7–7
7.1.11 CSR Memory Base Address Register—Offset 10h 7–7
7.1.12 CSR I/O Base Address Register—Offset 14h 7–8
7.1.13 SDRAM Base Address Register—Offset 18h 7–9
7.1.14 CardBus CIS Pointer Register—Offset 28h 7–9
7.1.15 Expansion ROM Base Address Register—Offset 30h 7–10
7.1.16 Capabilities Pointer—Offset 34h 7–10
7.1.17 Subsystem Vendor ID Register—Offset 2Ch 7–11
7.1.18 Subsystem ID Register—Offset 2Eh 7–11
7.1.19 Interrupt Line Register—Offset 3Ch 7–11
7.1.20 Interrupt Pin Register—Offset 3Dh 7–11
7.1.21 Min_Gnt Register—Offset 3Eh 7–11
7.1.22 Max_Lat Register—Offset 3Fh 7–12
7.1.23 Capability Identifier Register—Offset 70h 7–12
7.1.24 Power Management Capabilities (PMC) Register—Offset 72h 7–12
7.1.25 Power Management Control/Status (PMCSR) Register—Offset 74h 7–13
7.1.26 Data—Offset 77h 7–14
7.2 PCI Control and Status Registers 7–14
7.2.1 Outbound Interrupt Status Register—Offset 30h 7–15
7.2.2 Outbound Interrupt Mask Register—Offset 34h 7–16
7.2.3 I_2O Inbound FIFO Register—Offset 40h 7–16
7.2.3.1 Reading I_2O Inbound FIFO 7–16
7.2.3.2 Writing I_2O Inbound FIFO 7–16
7.2.4 I_2O Outbound FIFO Register—Offset 44h 7–17
7.2.4.1 Reading I_2O Outbound FIFO 7–17
7.2.4.2 Writing I_2O Outbound FIFO 7–17
7.2.5 Mailbox n Registers—Offset 50h to 5Ch 7–17
7.2.6 Doorbell Register—Offset 60h 7–18
7.2.7 Doorbell Setup—Offset 64h 7–18
7.2.8 ROM Write Byte Address Register—Offset 68h 7–18
7.3 SA-110 Control and Status Registers 7–19
7.3.1 DMA Channel n Byte Count Register—Offset 80h/A0h 7–21
7.3.2 DMA Channel n PCI Address Register—Offset 84h/A4h 7–22
7.3.3 DMA Channel n SDRAM Address Register—Offset 88h/A8h 7–22
7.3.4 DMA Channel n Descriptor Pointer Register—Offset 8Ch/ACh 7–23
7.3.5 DMA Channel n Control Register—Offset 90h/B0h 7–24
7.3.6 DMA Channel n DAC Address—Offset 94h/B4h 7–25
7.3.7 CSR Base Address Mask Register—Offset F8h 7–26

7.3.8 CSR Base Address Offset Register—Offset FCh 7–27
7.3.9 SDRAM Base Address Mask Register—Offset 100h 7–27
7.3.10 SDRAM Base Address Offset Register—Offset 104h 7–28
7.3.11 Expansion ROM Base Address Mask Register—Offset 108h 7–29
7.3.12 SDRAM Timing Register—Offset 10Ch 7–31
7.3.13 SDRAM Address and Size Registers—Offsets 110h to 11Ch 7–33
7.3.14 I_2O Inbound Free_List Head Pointer Register—Offset 120h 7–34
7.3.15 I_2O Inbound Post_List Tail Pointer Register—Offset 124h 7–34
7.3.16 I_2O Outbound Post_List Head Pointer Register—Offset 128h 7–34
7.3.17 I_2O Outbound Free_List Tail Pointer Register—Offset 12Ch 7–35
7.3.18 I_2O Inbound Free_List Count Register—Offset 130h 7–35
7.3.19 I_2O Outbound Post_List Count Register—Offset 134h 7–35
7.3.20 I_2O Inbound Post_List Count Register—Offset 138h 7–36
7.3.21 SA-110 Control Register—Offset 13Ch 7–36
7.3.22 PCI Address Extension Register—Offset 140h 7–39
7.3.23 Prefetchable Memory Range Register—Offset 144h 7–40
7.3.24 X-Bus Cycle/Arbiter Register—Offset 148h 7–41
7.3.25 X-Bus I/O Strobe Mask Register—Offset 14Ch 7–43
7.3.26 Doorbell PCI Mask Register—Offset 150h 7–44
7.3.27 Doorbell SA-110 Mask Register—Offset 154h 7–44
7.3.28 Interrupt Controller Registers 7–44
7.3.29 IRQStatus/FIQStatus Register—Offset 180h/280h 7–46
7.3.30 IRQRawStatus/FIQRawStatus Register—Offset 184h/284h 7–47
7.3.31 IRQEnable/FIQEnable Register—Offset 188h/288h 7–47
7.3.32 IRQEnableSet/FIQEnableSet Register—Offset 188h/288h 7–47
7.3.33 IRQEnableClear/FIQEnableClear Register—Offset 18Ch/28Ch 7–47
7.3.34 IRQSoft/FIQSoft Register—Offset 190h/290h 7–48
7.3.35 SA–110 DAC Address Registers—Offsets 200h to 204h 7–48
7.3.36 SA-110 DAC Control Registers—Offset 208h 7–49
7.3.37 PCI Address 31 Alias Register—Offset 20Ch 7–49
7.3.38 Timer Control Registers 7–50
7.3.39 TimerNLoad Register—Offsets 300h, 320h, 340h, and 360h 7–50
7.3.40 TimerNValue Register—Offsets 304h, 324h, 344h, and 364h 7–50
7.3.41 TimerNControl Register—Offsets 308h, 328h, 348h, and 368h 7–51
7.3.42 TimerNClear Register—Offsets 30Ch, 32Ch, 34Ch, and 36Ch 7–51

8 JTAG Test Port 8–1

8.1 Test Access Port Controller 8–1
8.2 Boundary-Scan Implementation Exceptions 8–2
8.3 Instruction Register 8–2
8.4 Bypass Register 8–2
8.5 Boundary-Scan Register 8–3

9 Electrical Specifications 9–1

9.1 PCI Electrical Specification Conformance 9–1
9.2 Absolute Maximum Ratings 9–1
9.3 DC Specifications 9–2
9.4 AC Timing Specifications 9–3
9.4.1 PCI Clock Timing Specifications 9–3

9.4.2 PCI Signal Timing Specifications 9–4
9.4.3 PCI Reset Timing Specifications 9–5
9.4.4 JTAG Timing Specifications 9–6
9.5 Memory and SA-110 Interface Timing 9–6
9.5.1 SA-110, 21285, and SDRAM Clock Signal Timing Specifications 9–7
9.5.2 SA-110, 21285, and SDRAM Interface Timing Specifications 9–8

10 Mechanical Specifications 10–1

Support, Products, and Documentation ..... -

Figures

1-1 21285 Host Bridge Application Diagram 1–2
1-2 SA-110 Coprocessor Application Diagram 1–3
1-3 Intelligent Add-In Card Application Diagram 1–3
2-1 Clock Generation 2–8
2-2 21285 PBGA Pin Down View 2–12
3-1 21285 Block Diagram 3–2
3-2 Mapping PCI Address to SDRAM Address 3–8
4-1 SDRAM Configuration 4–1
4-2 ROM Configuration 4–5
5-1 X-Bus Configuration 5–4
5-2 X-Bus Timing 5–6
6-1 Secondary Arbiter Example 6–2
6-2 DMA Descriptor Read 6–3
6-3 SDRAM-to-PCI Transfers 6–7
6-4 I_2O Overview 6–8
6-5 Circulation of MFAs 6–10
6-6 Timer Block Diagram 6–11
7-1 PCI-Generated SDRAM Access 7–29
7-2 Interrupt Source Configuration 7–45
8-1 Test Access Port (TAP) Controller State Transitions 8–1
9-1 PCI Clock Signal AC Parameter Measurement Conditions 9–3
9-2 PCI Signal Timing Measurement Conditions 9–4
9-3 Clock Signal AC Parameter Measurement Conditions 9–7
9-4 Interface Timing Measurement Conditions 9–8
9-5 Input Timing Measurement Conditions 9–9
10-1 256 Plastic Ball Grid Array (PBGA) Package 10–1

Tables

2-1 PCI Signal-Type Abbreviations 2–1
2-2 PCI Bus Interface Signals 2–1
2-3 SA-110 Signals Connected to the 21285 2–3
2-4 SA-110 Signals Not Connected to the 21285 2–4
2-5 ROM Signals 2–5
2-6 SDRAM Signals 2–5
2-7 ma Signals 2–6
2-8 Serial Port Signals 2–7

2-9 Miscellaneous Signals 2–7
2-10 X-Bus Signals 2–9
2-11 PCI Arbiter Signals 2–10
2-12 JTAG Signals 2–10
2-13 Pin State at Reset 2–11
2-14 21285 PBGA Location Pin List 2–13
2-15 21285 Pin Name Pin List 2–17
3-1 SDRAM Address Generation 3–8
3-2 SA-110 Address Generation 3–14
3-3 DMA Address Generation 3–15
3-4 I/O Address Generation 3–16
3-5 Configuration Address Generation 3–17
4-1 Array Sizes 4–2
4-2 SDRAM Addresses 4–3
4-3 SDRAM Commands 4–3
4-4 ROM Addressing 4–6
4-5 ROM Address Generation for SA-110 Accesses 4–6
4-6 ROM Address Generation for PCI Accesses 4–6
5-1 SA-110 4GB Address Mapping 5–1
6-1 DMA Channel Write 6–5
6-2 Final Write 6–5
6-3 Aligning Byte Data 6–5
6-4 FIFO Pointers 6–8
6-5 UART Interrupts 6–14
7-1 Register Access 7–1
7-2 PCI Configuration Mapping 7–2
7-3 PCI Control and Status Registers 7–15
7-4 SA-110 Control and Status Registers 7–19
7-5 Descriptor Block Format 7–23
7-6 PCI Window Sizes 7–26
7-7 SDRAM Window Sizes 7–28
7-8 Expansion ROM Window Sizes 7–30
7-9 I_2O Inbound and Outbound List Sizes 7–39
7-10 Interrupt Controller Registers 7–45
7-11 Interrupt Source Bits 7–45
7-12 Timer Control Registers 7–50
8-1 Test Modes and Features 8–2
9-1 Absolute Maximum Ratings 9–1
9-2 Functional Operating Range 9–1
9-3 DC Parameters 9–2
9-4 PCI Clock Signal AC Parameters 9–3
9-5 PCI Signal Timing Specifications 9–4
9-6 PCI Signal Timing AC Parameters 9–5
9-7 PCI Reset Timing Specifications 9–5
9-8 JTAG Timing Specifications 9–6
9-9 SA-110, 21285, and SDRAM Clock Signal AC Parameters 9–7
9-10 Memory and SA-110 AC Parameters 9–9
10-1 256 Plastic Ball Grid Array (PBGA) Package Dimensions 10–2

Introduction 1

Intel Semiconductor's 21285 is a single-chip interface between an SA-110 microprocessor, synchronous DRAM memory (SDRAM), read-only memory (ROM), and the PCI bus. It is designed to support the following three modes of operation:

- PCI-based SA-110 computer. In this configuration, the SA-110 processor is the host Intelprocessor. It configures all PCI peripherals in the system.
- Attached processor. In this configuration, the SA-110 processor system interfaces directly to the host system PCI bus. The 21285 is configured by the host processor in the system.
- Intelligent add-in card. In this configuration, the SA-110 processor controls a PCI-based subsystem that is designed to be plugged into a host system PCI slot. Normally, a PCI-to-PCI bridge device interfaces the oncard PCI system to the host PCI system. The SA-110 processor typically configures the devices on the add-in card. Alternatively, these devices and the 21285 can be configured by the host processor in the system.

The SA-110 bus must be configured for enhanced mode (SA-110 signal **CONFIG** is high), address pipeline enabled (SA-110 signal **APE** tied high), and asynchronous bus clock (SA-110 signal **SnA** tied low). The SA-110 bus interface and PCI interface clocks are asynchronous to each other (all synchronization is done within the 21285). However, the SA-110 bus clock frequency *must* always be greater than or equal to the PCI bus clock frequency.

1.1 Features

The 21285 has the following features:

- SDRAM interface
- Flash ROM interface
- PCI Revision 2.1 compliant interface (32 bit, 33 MHz)
- DAC support (for Rev_ID 4 or higher)
- Enhanced parity support (for Rev_ID 4 or higher)
- Power Management Support (for Rev_ID 4 or higher)
- DMA controllers
- Interrupt controller
- Programmable timers
- Doorbell registers/mailboxes
- I_2O message unit
- X-Bus (8-bit parallel port)
- Serial port (UART)

- PCI bus arbiter
- Supports both 5-V and 3.3-V signaling environments
- Provides an IEEE standard 1149.1 JTAG interface

1.1.1 Power Management

The 21285 REV_ID 4 or higher provides support to enable applications to be compliant with the *PCI Bus Power Management Interface Specification*. Specifically, the required configuration registers are provided in the 21285 along with an interrupt to the SA-110 processor, indicating that the registers have been written by the host processor. This allows the application to provide firmware (run by the SA-110) to control subsystem power and meet the requirements of the *PCI Bus Power Management Specification*.

1.2 System Applications

Figure 1-1 shows an application diagram of a 21285 based SA-110 system.

Figure 1-1. 21285 Host Bridge Application Diagram

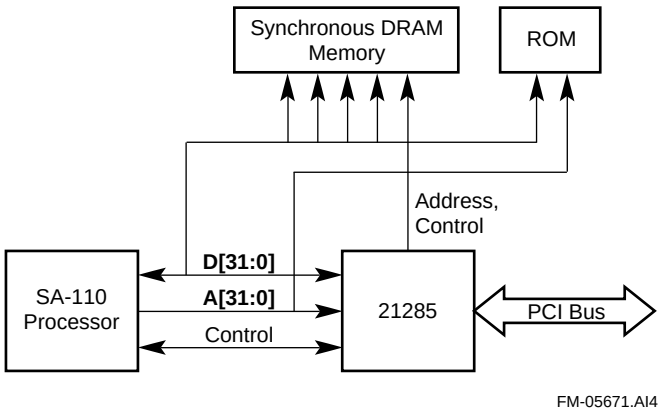


Figure 1-2 shows an application diagram of the SA-110 coprocessor system. In this system:

- SA-110 acts as a coprocessor of accelerator.
- Host CPU configures all PCI devices. The SA-110/21285 appears as a PCI device.
- SA-110 may allow host CPU to have a window to local memory.
- SA-110 can access system memory (as a PCI bus master).

Figure 1-2. SA-110 Coprocessor Application Diagram

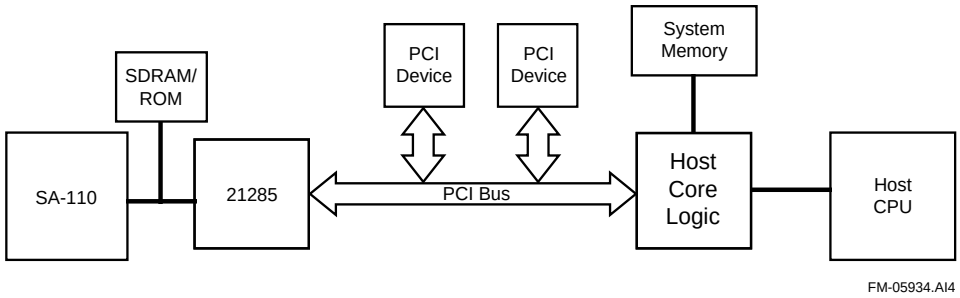
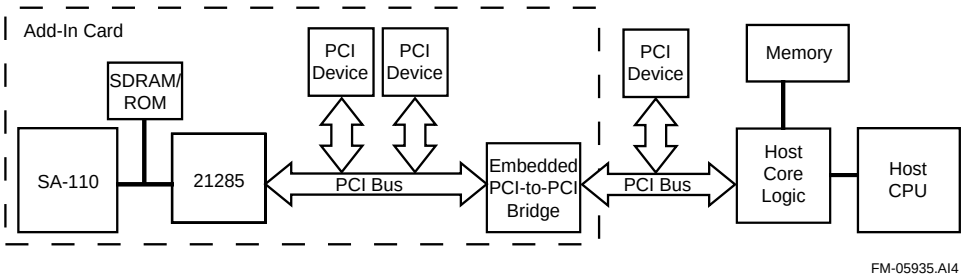


Figure 1-3 shows an application diagram of the intelligent add-in card system. In this system:

- SA-110 is an embedded controller on add-in card.
- Host CPU configures all PCI devices. Subsystem appears as a PCI device.
- PCI devices on add-in card are private to SA-110.

Figure 1-3. Intelligent Add-In Card Application Diagram



Signal Description 2

The 21285 signals are categorized into one of several groups: PCI, SA-110, ROM, SDRAM, serial port, miscellaneous, X-Bus/Arbiter, and JTAG.

Table 2-1 defines the PCI and SA-110 signal-type abbreviations used in the signal tables. These abbreviations use the same conventions and descriptions as found in the *PCI Local Bus Specification, Revision 2.1* and the *Intel SA-110 Microprocessor Technical Reference Manual*.

Table 2-1. PCI Signal-Type Abbreviations

Signal Type	Description
I	Standard input only.
O	Standard output only.
TS	Tristate bidirectional.
STS	Sustained tristate. Active low signal owned and driven by one and only one agent at a time. The agent that drives this pin low must drive it high for at least one clock before letting it float. A new agent cannot start driving this signal any sooner than one clock after the previous owner tristates it. A pull-up is required to sustain the inactive state until another agent drives it, and it must be provided by the central resource (that is, on a PC board).
OD	Standard open drain allows multiple devices to share as a wire-OR. A pull-up is required to sustain the inactive state until another agent drives it, and it must be provided by the central resource.
ICOCZ	Input, CMOS threshold, output CMOS levels, tristateable
IC	Input, CMOS threshold
OCZ	Output, CMOS levels, tristateable

2.1 PCI Signals

The timing of the PCI signals is referenced to the **pci_clk.** Table 2-2 describes the PCI bus interface signals.

Table 2-2. PCI Bus Interface Signals (Sheet 1 of 3)

Signal Name	Type	Description
ad[31:0]	TS	**Address/data**. These signals are multiplexed address and data bus. The 21285 receives addresses as target and drives addresses as master. It receives write data and drives read data as target. It drives write data and receives read data as master.
cbe_l[3:0]	TS	**Command byte enables**. These signals are multiplexed command and byte enable signals. The 21285 receives commands as target and drives commands as master. It receives byte enables as target and drives byte enables as master.
par	TS	**Parity**. This signal carries even parity for **ad** and **cbe_l**. It has the same receive and drive characteristics as the address and data bus, except that it is one PCI cycle later.

Table 2-2. PCI Bus Interface Signals (Sheet 2 of 3)

Signal Name	Type	Description
frame_l	STS	**Frame**. **frame_l** indicates the beginning and duration of an access. The 21285 receives as target and drives as master.
irdy_l	STS	**Initiator ready**. **irdy_l** indicates the master's ability to complete the current data phase of the transaction. The 21285 receives as target and drives as master.
trdy_l	STS	**Target ready**. **trdy_l** indicates the target's ability to complete the current data phase of the transaction. The 21285 drives as target and receives as master.
stop_l	STS	**Stop**. **stop_l** indicates that the target is requesting the master to stop the current transaction. The 21285 drives as target and receives as master.
devsel_l	STS	**Device select**. **devsel_l** indicates that the driving device has decoded its address as the target of the current access. The 21285 drives as target and receives as master.
idsel	I	**Initialization device select**. **idsel** is used as a chip select during configuration read and write transactions.
perr_l	STS	**Parity error**. **perr_l** is used to report data parity errors. The 21285 asserts this when it receives bad data parity as target of a write or master of a read.
serr_l	I, OD	**System error**. **serr_l** as an input can cause an interrupt to the SA-110 if the 21285 PCI central function (**pci_cfn**) is asserted. As an output, it is asserted by the 21285 when a bit in the SA-110 control register is set by the SA-110, or in response to a PCI address parity error, when the 21285 **pci_cfn** is deasserted.
req_l	O	**Request**. If the internal arbiter is not used, **req_l** is the request from the 21285 indicating that it wants to become bus master on the PCI bus. If the internal arbiter is used (see Section 6.1), **req_l** is the grant to the 21285 from the arbiter, and must be connected to **gnt_l** external to the 21285.
gnt_l	I	**Grant**. **gnt_l** from the PCI arbiter (either external or internal), indicates that the 21285 can take control of the PCI bus. If the 21285 does not have any transactions to do on the PCI when **gnt_l** is asserted, it parks the PCI bus.
pci_irq_l	I, OD	**PCI interrupt**. **pci_irq_l** is an output used to interrupt the host processor or the SA-110. It is asserted when there is a doorbell set or there are messages on the I_2O outbound post_list.
pci_rst_l	I, TS	**PCI reset**. **pci_rst_l** is used to bring all PCI devices to a known initial state. When it is asserted, all registers in the 21285 are put into their reset state, all transaction queues are reset, and all PCI-related state sequencers are initialized. When **pci_cfn** is deasserted, **pci_rst_l** is an input, and when asserted, the 21285 asserts **nRESET** to the SA-110. When **pci_cfn** is asserted, **pci_rst_l** is an output controlled by **nRESET** and the SA-110 control register.

Table 2-2. PCI Bus Interface Signals (Sheet 3 of 3)

Signal Name	Type	Description
pci_clk	I	**Clock**. **pci_clk** is the reference for PCI signals and internal operations.
pci_cfn	I	**PCI central function**. When **pci_cfn**[a] is asserted, the 21285 is the PCI central function and: • **nRESET** is an input. • **pci_rst_l** is an output asserted when **nRESET** is asserted. • The 21285 provides bus parking during reset. • **serr_l** is received. When **pci_cfn**<superscript>a is deasserted, the 21285 is not the PCI central function and: • **pci_rst_l** is an input. • **nRESET** is an output asserted when **pci_rst_l** is asserted. • The 21285 does not provide bus parking during reset. • **serr_l** is an output.
Vio	I	**I/O voltage**. **Vio** must be tied to either 5 V or 3.3 V to correspond to the signaling environment of the PCI bus as described in the *PCI Local Bus Specification, Revision 2.1.*

a. The value on this pin may be read via the SA-110 control register.

2.2 SA-110 Signals

The timing of the SA-110 signals is referenced to the **MCLK.** Table 2-3 describes the SA-110 signals that are connected to the 21285. Table 2-4 describes the SA-110 signals that are not connected to the 21285 and must be forced to a 1 or 0 state.

The SA-110 must be configured for little endian mode. The 21285 does not support big endian mode.

Table 2-3. SA-110 Signals Connected to the 21285 (Sheet 1 of 2)

Signal Name	Type	Description
A[31:0]	ICOCZ	Address bus. The address is requested by the SA-110 on bits [31:2] and the two high-byte enables on bits [1:0]. These signals are also driven by the 21285 when accessing the ROM or X-Bus.
D[31:0]	ICOCZ	Data bus. The data bus carries write data from the SA-110; output data from SDRAM, ROM, X-Bus, and PCI (through the 21285); and register data from the 21285.
MAS[1:0]	ICOCZ	**Memory access size**. Indicates the low 2-byte enables during SA-110 bus cycles. These pins are driven by the 21285 during ROM and X-Bus accesses to keep them from floating. The level is undefined.
nMREQ	IC	**Memory request**. Indicates that the SA-110 is doing a bus cycle in the following cycle.
nRW	IC	**Read/write**. Indicates whether an SA-110 bus cycle is a read or a write.
CLF	IC	**Cache line fill**. Indicates that an SA-110 bus cycle is a cache line read or a full cache block evict. The 21285 uses the information to control SDRAM and ROM bursts.

Table 2-3. SA-110 Signals Connected to the 21285 (Sheet 2 of 2)

Signal Name	Type	Description
LOCK	IC	**Locked operation**. Indicates that an SA-110 read is followed by a write. While it is asserted, it prevents rearbitration for the SA-110 bus; that is, no other transaction can happen on that bus except for SA-110 bus cycles.
ABE	OCZ	**Address bus enable**. **ABE** is deasserted by the 21285 to tristate **A[31:0]** so it can access SDRAM and ROM.
DBE	OCZ	**Data bus enable**. **DBE** is deasserted by the 21285 to tristate **D[31:0]** so it can access SDRAM and ROM.
nIRQ	OCZ	**Interrupt request**. **nIRQ** is asserted when one or more of the interrupt sources is active, unmasked, and directed to normal interrupt. It is deasserted when all interrupt sources are removed or masked by writing the IRQEnable/FIQEnable registers.
nFIQ	OCZ	**Fast interrupt request**. **nFIQ** is asserted when one or more of the interrupt sources is active, unmasked, and directed to fast interrupt. It is deasserted when all interrupt sources are removed or masked by writing the IRQEnable/ FIQEnable registers.
nRESET	ICOCZ	**SA-110 reset**. When **pci_cfn** is negated, **nRESET** is an output and is asserted when **pci_rst_l** is asserted. When **pci_cfn** is asserted, **nRESET** is bidirectional. It is normally an input, but it will be driven asserted by the 21285 if the watchdog timer is enabled and expires. This signal requires an external pull-up resistor.
MCLK	OCZ	**21285 to SA-110 clock**. **MCLK** is an internally buffered and inverted version of **osc**, and is controlled by the 21285 to stall the SA-110 during SA-110 bus cycles.

Table 2-4. SA-110 Signals Not Connected to the 21285

Signal Name	Forced Status
nWAIT	1
MSE	1
ABORT	0
APE	1
CONFIG	1
SnA	0

2.3 ROM Signals

Table 2-5 describes signal outputs that are used for accessing the ROM. Refer to Section 4.2.1 for additional ROM address bit connections. While the ROM is being accessed, signal **rom_ce_l** is asserted. Refer to Section 4.2 for a description of the address and data bus signals.

Table 2-5. ROM Signals

Signal Name	Type	Description
A[31]	ICOCZ	**ROM write enable**. This signal allows data to be stored in a flash ROM.
A[30]	ICOCZ	**ROM output enable**. This signal allows the ROM to output data.
A[29:28]	ICOCZ	**Two low address bits of ROM**. Table 4-4 describes the use of these bits.
rom_ce_l	OCZ	**ROM chip enable**. Chip enable is asserted by the 21285 during ROM accesses.

2.4 SDRAM Signals

The timing of the SDRAM signals is referenced to the **sdclk**. The SDRAM data lines may be directly connected to **D[31:0]**, or through transceivers, depending on the number of SDRAM chips. Table 2-6 describes the SDRAM signals.

Note: The term array is used to represent a group of SDRAM chips that share a common chip select.

Table 2-6. SDRAM Signals (Sheet 1 of 2)

Signal Name	Type	Description
ma[12:0]	ICOCZ	**Memory addresses**. **ma[12:0]** provides multiplexed row and column addresses to the SDRAMs. Some of the **ma** pins are used at reset to latch configuration information. Refer to Table 2-7.
ba[1:0]	OCZ	**Bank addresses**. Selects SDRAM bank.
cmd[2:0]	OCZ	**Command lines**. SDRAM row address strobe (RAS), column address strobe (CAS), and write enable (WE).
dqm[3:0]	OCZ	**Masks**. Control write operations for each byte. **dqm[3]** controls **D[31:24]** **dqm[2]** controls **D[23:16]** **dqm[1]** controls **D[15:8]** **dqm[0]** controls **D[7:0]**
cs_l[3:0]	OCZ	**Chip selects**. Selects the four arrays of SDRAM.

Table 2-6. SDRAM Signals (Sheet 2 of 2)

Signal Name	Type	Description
d_wren_l	OCZ	**SDRAM data bus buffer control**. Used as the direction control for an optional data bus buffer between the SDRAM and the 21285/SA-110. Asserted during writes to the SDRAM.
parity[3:0]	ICOCZ	**Byte parity for D[31:0]**. Used for SDRAMs if enabled in the SDRAM timing register. **parity[3]** for **D[31:24]** **parity[2]** for **D[23:16]** **parity[1]** for **D[15:8]** **parity[0]** for **D[7:0]** These signals should not be left floating. If SDRAM parity is not used; connect them to V_{ss} or V_{dd}. If parity is used, then bus sustainers are required.
sdclk[3:0]	OCZ	**SDRAM clocks**. These clocks are internally buffered versions of **osc**. There is one clock for each array of SDRAM chips.

Table 2-7 describes the **ma** signals that are inputs when **nRESET** is asserted, and are latched at the deassertion of **nRESET**. These signals are used to control the configuration of various chip features. The pins have an internal pull-up resistor, therefore, any pins that are not terminated are considered to be a logical 1. A pull-down resistor can be used to supply a logical 0.

Note: If external buffering is used on the **ma** pins, the internal pull-up resistor may be inadequate to force a reliable logic level on **ma[8:2]**. This occurs because buffers such as 74LVT16244A contain internal bus hold circuitry on their input pins that will force an input high or low if it attempts to float. The pull-up resistor internal to the 21285 is unable to supply enough current to overcome the hysterisis in the bus hold circuitry. In this case, a 4.7 K pull-up or pull-down resistor should be fitted externally on each of the **ma[8:2]** pins.

Table 2-7. ma Signals (Sheet 1 of 2)

Signal Name	Type	Description
ma[8]	ICOCZ	Intel reserved test mode. When 1: Normal operation. When 0: Enable Intel reserved test mode.
ma[7]	ICOCZ	X-Bus/Arbiter mode. When 0: X-Bus/Arbiter pins are used for the internal PCI arbiter. When 1: X-Bus/Arbiter pins are used for the X-Bus.
ma[6]	ICOCZ	Blank ROM program mode. When 1: Normal mode. The 21285 does not allow access from the PCI until after the 21285 CSRs have been initialized from the SA-110. When 0: Blank ROM mode. The 21285 allows access from the PCI prior to the normal initialization from the SA-110. See Section 4.2.5 for a description of all the specific differences.

Table 2-7. ma Signals (Sheet 2 of 2)

Signal Name	Type	Description
ma[5:4]	ICOCZ	ROM width. 11 - Byte 01 - Word 10 - Dword 00 - Reserved
ma[3]	ICOCZ	Class code selection. When 1: Class code register reads B40001h. When 0: Class code register reads 0E0001h.
ma[2]	ICOCZ	Intel reserved test mode. When 1: Normal operation. When 0: Enable ohold test mode.

2.5 Serial Port Signals

Table 2-8 describes the serial port signals

Table 2-8. Serial Port Signals

Signal Name	Type	Description
rx	IC	Receive data into the UART
tx	OCZ	Transmit data from the UART

2.6 Miscellaneous Signals

Table 2-9 describes the miscellaneous signals.

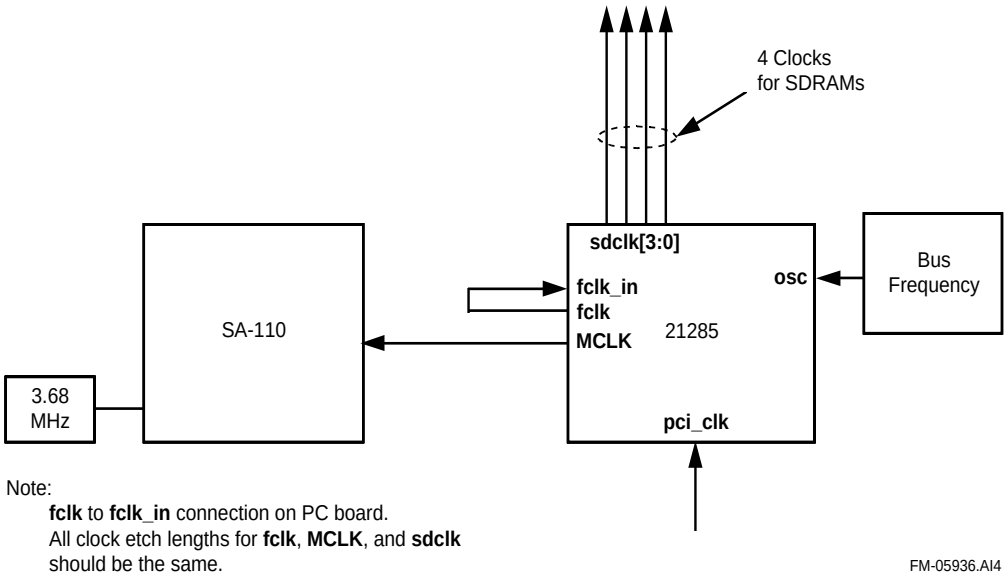
Table 2-9. Miscellaneous Signals (Sheet 1 of 2)

Signal Name	Type	Description
osc	IC	Reference input clock. Supplied by an oscillator and buffered to create MCLK, sdclk, and fclk.
fclk	OCZ	Clock output for 21285. fclk is an internally buffered version of osc and must be connected on the circuit board to fclk_in (Figure 2-1).

Table 2-9. Miscellaneous Signals (Sheet 2 of 2)

Signal Name	Type	Description
fclk_in	IC	Clock input for 21285. fclk_in must be connected on the circuit board to fclk (Figure 2-1). The etch length of the connection should be matched to the etch length of MCLK to the SA-110 and sdclk to the SDRAMs to minimize skew.
irq_in_l[3:0]	IC	Interrupt inputs. These signals are general-purpose inputs that can assert nFIQ or nIRQ, can be masked, or can clock the timers. These signals are level sensitive with programmable priority and are synchronized before being used.
scan_en	IC	Scan enable. This pin is used only in the chip manufacturing test for the internal scan chain. It must be 0 for normal operation.

Figure 2-1. Clock Generation



2.7 X-Bus/Arbiter Signals

The following group of pins can be configured either as a parallel port (X-Bus) or as a PCI arbiter. The configuration is determined by the value latched on **ma[7]** at reset.

2.7.1 X-Bus Selection

X-Bus (**ma[7]**=1) is a general-purpose parallel interface port that allows connection of 8- and 16-bit peripheral chips. The address to the peripherals is supplied by **A[31:0]** (or possibly a buffered version of this signal depending upon the number of devices). The data for the peripherals is transmitted on **D[31:0]**, which is connected through transceivers controlled by **xd_wren_l**.

Table 2-10 describes the X-Bus signals.

Table 2-10. X-Bus Signals

Signal Name	Type	Description
xd_wren_l	OCZ	**X-Bus data buffer control**. Used as the direction control for a data bus buffer between the X-Bus and the 21285/SA-110. Asserted during writes to the X-Bus.
xior_l	OCZ	**X-Bus read strobe**. Asserted by the 21285 to enable the X-Bus peripheral to drive read data.
xiow_l	OCZ	**X-Bus write strobe**. Asserted by the 21285 to enable the X-Bus peripheral to accept write data.
xcs_l[2:0]	ICOCZ	**X-Bus chip selects**. As outputs, **xcs_l[2:0]** are used as chip selects asserted by the 21285 during writes or reads to X-Bus devices. For information on address ranges, see Table 5-1. As inputs, **xcs_l[2:0]** can be used as interrupts providing that they are not used for X-Bus chip selects. The direction is controlled by the SA-110 control register. Signal **xcs_l[1]** is also used as the Power Management event signal **pre_pme_l**. To be used in this mode, it must be enabled as a output in the SA-110 Control Register. This output is asserted by the 21285 when PMCSR bits [15] (**PME_Status**) and [8] (**PME_En**) are both enabled.

2.7.2 PCI Arbiter Selection

The PCI arbiter (**ma[7]**=0) receives requests from five potential bus masters (four external and the 21285), and asserts a grant to the master with the highest priority. See Section 6.1 for a description of the PCI arbiter operation.

Table 2-11 describes the PCI arbiter signals.

Note: When the internal arbiter is selected, the **req_l** pin is the arbiter grant to the 21285 and must be connected to **gnt_l** external to the 21285.

Table 2-11. PCI Arbiter Signals

Signal Name	Type	Description
pci_gnt_l[0]	OCZ	**pci_gnt_l[0]** output
xd_wren_l	OCZ	**pci_gnt_l[1]** output
xior_l	OCZ	**pci_gnt_l[2]** output
xiow_l	OCZ	**pci_gnt_l[3]** output
xcs_l[2:0]	ICOCZ	**pci_req_l[2:0]** inputs
pci_req_l[3]	IC	**pci_req_l[3]** input

2.8 JTAG Signals

JTAG is the IEEE 1149.1 test access port. The JTAG input signals have a weak pull-up resistor internal to the chip, so that any input not terminated will be interpreted as a high.

Table 2-12 describes the JTAG signals.

Table 2-12. JTAG Signals

Signal Name	Type	Description
tck	IC	Test interface reference clock. This clock times all the transfers on the JTAG test interface.
tms	IC	Test interface mode select. tms causes state transitions in the test access port (TAP) controller.
tdi	IC	Test interface data input. tdi is the serial input through which JTAG instructions and test data enter the JTAG interface.
tdo	OCZ	Test interface data output. tdo is the serial output through which test instructions and data from the test logic leave the 21285.
trst_l	IC	Test interface reset. When asserted low, the TAP controller is asynchronously forced to enter a reset state, which in turn asynchronously initializes other test logic. This pin must be driven or held low to achieve normal device operation.

2.9 Pin State During Reset

Table 2-13 defines the state of both the output (O) and bidirectional (TS) pins during reset. SA-110 related pins are controlled by **nRESET** and PCI related pins are controlled by **pci_rst_l**.

Table 2-13. Pin State at Reset (Sheet 1 of 2)

Signal Name	Type	Value (if O)
ad	O if **pci_cfn** is enabled, otherwise TS	00000000h
cbe_l	O if **pci_cfn** is enabled, otherwise TS	0h
par	O if **pci_cfn** is enabled, otherwise TS	0
frame_l	TS	—
irdy_l	TS	—
trdy_l	TS	—
stop_l	TS	—
devsel_l	TS	—
perr_l	TS	—
serr_l	TS	—
req_l	TS	—
pci_irq_l	TS	—
A	TS	—
D	O	Undefined
MAS	TS	—
ABE	O	Asserted (h)
DBE	O	Deasserted (l)
nIRQ	O	Undefined
nFIQ	O	Undefined
MCLK	O	—
ba	TS	—
ma	TS	—
cmd	O	Undefined
dqm	O	Asserted (h)
cs_l	O	Deasserted (h)
d_wren_l	O	Asserted (l)
parity	O	Undefined
sdclk	O	Follows **osc**
tx	O	Asserted (h)
fclk	O	Follows **osc**
rom_ce_l	O	Deasserted (h)
pci_gnt_l[3]	TS	—
xd_wren_l	TS	—

Table 2-13. Pin State at Reset **(Sheet 2 of 2)**

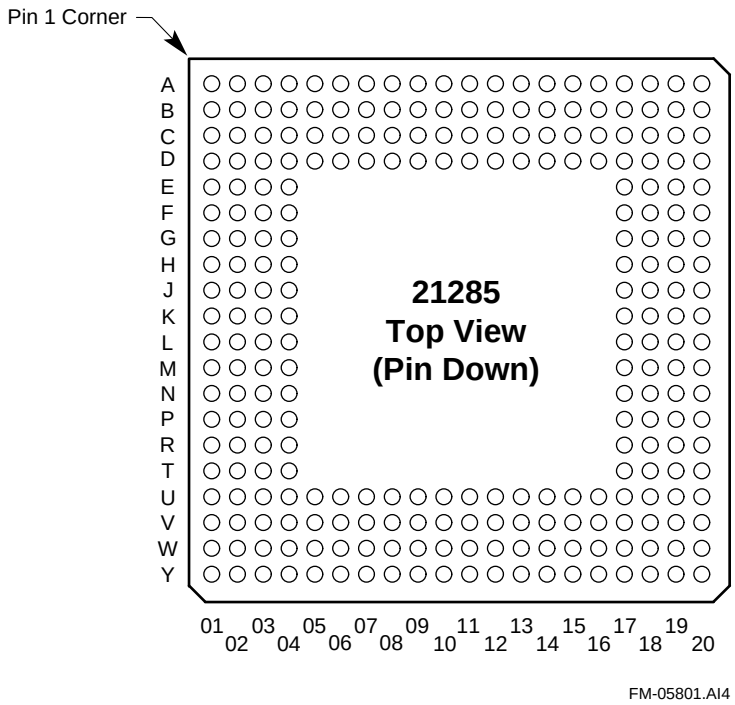
Signal Name	Type	Value (if O)
xior_l	TS	—
xiow_l	TS	—
xcs_l	TS	—
tdo	O	Undefined

2.10 Pin Assignment

This section describes the 21285 pin assignment and lists the pins according to location (numeric) and in alphabetic order.

Figure 2-2 shows the 21285 256-point ball grid array, representing the pins in vertical rows labeled alphabetically, and horizontal rows labeled numerically.
Table 2-14 and Table 2-15 use this location to identify pin assignments.

Figure 2-2. 21285 PBGA Pin Down View



2.11 Pins Listed in Numeric Order

Table 2-14 lists the 21285 pins in order of location (numeric), showing the location code, name, and signal type of each pin.

Figure 2-2 provides the map for identifying the pin location codes, listed in alphabetic order in the PBGA Location column in Table 2-14.

Table 2-1 defines the signal-type abbreviations used in the Type column in Table 2-14.

Table 2-14. 21285 PBGA Location Pin List (Sheet 1 of 4)

PBGA Location	Pin Name	Type	PBGA Location	Pin Name	Type
A1	**Vss**	P	A2	**Vdd**	P
A3	**D[3]**	ICOCZ	A4	**D[6]**	ICOCZ
A5	**dqm[0]**	OCZ	A6	**D[10]**	ICOCZ
A7	**D[12]**	ICOCZ	A8	**D[15]**	ICOCZ
A9	**D[16]**	ICOCZ	A10	**D[19]**	ICOCZ
A11	**D[20]**	ICOCZ	A12	**D[21]**	ICOCZ
A13	**parity[2]**	ICOCZ	A14	**D[24]**	ICOCZ
A15	**D[26]**	ICOCZ	A16	**D[30]**	ICOCZ
A17	**dqm[3]**	OCZ	A18	**nMREQ**	IC
A19	**sdclk[2]**	OCZ	A20	**Vdd**	P
B1	**ba[1]**	OCZ	B2	**Vss**	P
B3	**D[1]**	ICOCZ	B4	**D[4]**	ICOCZ
B5	**D[7]**	ICOCZ	B6	**D[8]**	ICOCZ
B7	**D[11]**	ICOCZ	B8	**D[14]**	ICOCZ
B9	**dqm[1]**	OCZ	B10	**D[18]**	ICOCZ
B11	**Vss**	P	B12	**Vdd**	P
B13	**dqm[2]**	OCZ	B14	**DBE**	OCZ
B15	**D[27]**	ICOCZ	B16	**D[31]**	ICOCZ
B17	**sdclk[1]**	OCZ	B18	**Vdd**	P
B19	**Vss**	P	B20	**Vss**	P
C1	**ma[10]**	ICOCZ	C2	**Vdd**	P
C3	**D[0]**	ICOCZ	C4	**Vss**	P
C5	**D[5]**	ICOCZ	C6	**parity[0]**	ICOCZ
C7	**D[9]**	ICOCZ	C8	**D[13]**	ICOCZ
C9	**Vss**	P	C10	**Vdd**	P
C11	**Vdd**	P	C12	**D[22]**	ICOCZ
C13	**Vss**	P	C14	**D[25]**	ICOCZ
C15	**D[29]**	ICOCZ	C16	**parity[3]**	ICOCZ
C17	**sdclk[0]**	OCZ	C18	**sdclk[3]**	OCZ

Table 2-14. 21285 PBGA Location Pin List **(Sheet 2 of 4)**

PBGA Location	Pin Name	Type	PBGA Location	Pin Name	Type
C19	**MCLK**	OCZ	C20	**fclk_in**	IC
D1	**Vss**	P	D2	**ba[0]**	OCZ
D3	**ma[12]**	ICOCZ	D4	**Vss**	P
D5	**D[2]**	ICOCZ	D6	**Vdd**	P
D7	**Vss**	P	D8	**Vss**	P
D9	**parity[1]**	ICOCZ	D10	**D[17]**	ICOCZ
D11	**Vdd**	P	D12	**D[23]**	ICOCZ
D13	**Vss**	P	D14	**D[28]**	ICOCZ
D15	**Vdd**	P	D16	**Vss**	P
D17	**Vss**	P	D18	**fclk**	OCZ
D19	**osc**	IC	D20	**LOCK**	IC
E1	**ma[7]**	ICOCZ	E2	**ma[8]**	ICOCZ
E3	**ma[9]**	ICOCZ	E4	**ma[11]**	ICOCZ
E17	**Vdd**	P	E18	**CLF**	IC
E19	**nRW**	IC	E20	**MAS[1]**	ICOCZ
F1	**ma[4]**	ICOCZ	F2	**ma[5]**	ICOCZ
F3	**ma[6]**	ICOCZ	F4	**Vdd**	P
F17	**Vdd**	P	F18	**ABE**	OCZ
F19	**A[0]**	ICOCZ	F20	**A[1]**	ICOCZ
G1	**ma[2]**	ICOCZ	G2	**Vss**	P
G3	**ma[3]**	ICOCZ	G4	**Vdd**	P
G17	**MAS[0]**	ICOCZ	G18	**Vss**	P
G19	**A[2]**	ICOCZ	G20	**A[3]**	ICOCZ
H1	**—**	NC	H2	**ma[0]**	ICOCZ
H3	**ma[1]**	ICOCZ	H4	**Vss**	P
H17	**Vss**	P	H18	**A[4]**	ICOCZ
H19	**A[5]**	ICOCZ	H20	**A[6]**	ICOCZ
J1	**irq_in_l[0]**	IC	J2	**irq_in_l[1]**	IC
J3	**irq_in_l[2]**	IC	J4	**irq_in_l[3]**	IC
J17	**A[7]**	ICOCZ	J18	**A[8]**	ICOCZ
J19	**A[9]**	ICOCZ	J20	**A[10]**	ICOCZ
K1	**Vss**	P	K2	**Vss**	P
K3	**Vdd**	P	K4	**Vdd**	P
K17	**A[11]**	ICOCZ	K18	**Vdd**	P
K19	**Vss**	P	K20	**Vdd**	P
L1	**cs_l[3]**	OCZ	L2	**cs_l[2]**	OCZ
L3	**cs_l[1]**	OCZ	L4	**cs_l[0]**	OCZ

Table 2-14. 21285 PBGA Location Pin List (Sheet 3 of 4)

PBGA Location	Pin Name	Type	PBGA Location	Pin Name	Type
L17	**Vdd**	P	L18	**A[13]**	ICOCZ
L19	**A[14]**	ICOCZ	L20	**A[12]**	ICOCZ
M1	**d_wren_l**	OCZ	M2	**cmd[2]**	OCZ
M3	**cmd[1]**	OCZ	M4	**cmd[0]**	OCZ
M17	**A[18]**	ICOCZ	M18	**A[17]**	ICOCZ
M19	**A[16]**	ICOCZ	M20	**A[15]**	ICOCZ
N1	**rom_ce_l**	OCZ	N2	**scan_en**	IC
N3	**tdi**	IC	N4	**Vss**	P
N17	**Vss**	P	N18	**A[20]**	ICOCZ
N19	**Vss**	P	N20	**A[19]**	ICOCZ
P1	**tdo**	OCZ	P2	**tms**	IC
P3	**tck**	IC	P4	**xd_wren_l**	OCZ
P17	**A[26]**	ICOCZ	P18	**A[23]**	ICOCZ
P19	**A[22]**	ICOCZ	P20	**A[21]**	ICOCZ
R1	**trst_l**	IC	R2	**xiow_l**	OCZ
R3	**xcs_l[2]**	ICOCZ	R4	**Vdd**	P
R17	**Vdd**	P	R18	**A[27]**	ICOCZ
R19	**A[25]**	ICOCZ	R20	**A[24]**	ICOCZ
T1	**xior_l**	OCZ	T2	**xcs_l[1]**	ICOCZ
T3	**tx**	OCZ	T4	**pci_req_l[3]**	IC
T17	**nRESET**	ICOCZ	T18	**A[30]**	ICOCZ
T19	**A[29]**	ICOCZ	T20	**A[28]**	ICOCZ
U1	**xcs_l[0]**	ICOCZ	U2	**rx**	IC
U3	**pci_rst_l**	I, TS	U4	**Vss**	P
U5	**ad[27]**	TS	U6	**Vdd**	P
U7	**idsel**	I	U8	**Vss**	P
U9	**Vdd**	P	U10	**Vdd**	P
U11	**stop_l**	TS	U12	**par**	TS
U13	**Vss**	P	U14	**Vss**	P
U15	**Vdd**	P	U16	**ad[4]**	TS
U17	**Vss**	P	U18	**nFIQ**	OCZ
U19	**nIRQ**	OCZ	U20	**Vdd**	P
V1	**pci_gnt[0]**	OCZ	V2	**pci_clk**	I
V3	**req_l**	O	V4	**ad[28]**	TS
V5	**ad[25]**	TS	V6	**Vdd**	P
V7	**ad[21]**	TS	V8	**ad[19]**	TS
V9	**ad[16]**	TS	V10	**trdy_l**	STS

Table 2-14. 21285 PBGA Location Pin List **(Sheet 4 of 4)**

PBGA Location	Pin Name	Type	PBGA Location	Pin Name	Type
V11	**Vdd**	P	V12	**serr_l**	I, OD
V13	**ad[14]**	TS	V14	**ad[11]**	TS
V15	**ad[8]**	TS	V16	**ad[7]**	TS
V17	**ad[3]**	TS	V18	**ad[1]**	TS
V19	**Vio**	I	V20	**A[31]**	ICOCZ
W1	**gnt_l**	I	W2	**Vss**	P
W3	**ad[31]**	TS	W4	**ad[29]**	TS
W5	**ad[24]**	TS	W6	**ad[23]**	TS
W7	**Vss**	P	W8	**ad[18]**	TS
W9	**cbe_l[2]**	TS	W10	**irdy_l**	TS
W11	**Vss**	P	W12	**perr_l**	TS
W13	**ad[15]**	TS	W14	**ad[12]**	TS
W15	**ad[10]**	TS	W16	**pci_irq_l**	OD
W17	**ad[6]**	TS	W18	**Vss**	P
W19	**Vss**	P	W20	**ad[0]**	TS
Y1	**Vdd**	P	Y2	**ad[30]**	TS
Y3	**ad[26]**	TS	Y4	**Vdd**	P
Y5	**cbe_l[3]**	TS	Y6	**ad[22]**	TS
Y7	**ad[20]**	TS	Y8	**ad[17]**	TS
Y9	**frame_l**	STS	Y10	**devsel_l**	STS
Y11	**Vdd**	P	Y12	**pci_cfn**	I
Y13	**cbe_l[1]**	TS	Y14	**ad[13]**	TS
Y15	**Vss**	P	Y16	**ad[9]**	TS
Y17	**cbe_l[0]**	TS	Y18	**ad[5]**	TS
Y19	**ad[2]**	TS	Y20	**Vdd**	P

2.12 Pins Listed in Alphabetic Order

Table 2-15 lists the 21285 pins in alphabetic order, showing the name, location code, and signal type of each pin.

Figure 2-2 provides the map for identifying the pin location codes.

Table 2-1 defines the signal-type abbreviations used in the Type column in Table 2-15.

Table 2-15. 21285 Pin Name Pin List (Sheet 1 of 4)

Pin Name	PBGA Location	Type	Pin Name	PBGA Location	Type
A[0]	F19	ICOCZ	**A[30]**	T18	ICOCZ
A[1]	F20	ICOCZ	**A[31]**	V20	ICOCZ
A[2]	G19	ICOCZ	**ABE**	F18	OCZ
A[3]	G20	ICOCZ	**ad[0]**	W20	TS
A[4]	H18	ICOCZ	**ad[1]**	V18	TS
A[5]	H19	ICOCZ	**ad[2]**	Y19	TS
A[6]	H20	ICOCZ	**ad[3]**	V17	TS
A[7]	J17	ICOCZ	**ad[4]**	U16	TS
A[8]	J18	ICOCZ	**ad[5]**	Y18	TS
A[9]	J19	ICOCZ	**ad[6]**	W17	TS
A[10]	J20	ICOCZ	**ad[7]**	V16	TS
A[11]	K17	ICOCZ	**ad[8]**	V15	TS
A[12]	L20	ICOCZ	**ad[9]**	Y16	TS
A[13]	L18	ICOCZ	**ad[10]**	W15	TS
A[14]	L19	ICOCZ	**ad[11]**	V14	TS
A[15]	M20	ICOCZ	**ad[12]**	W14	TS
A[16]	M19	ICOCZ	**ad[13]**	Y14	TS
A[17]	M18	ICOCZ	**ad[14]**	V13	TS
A[18]	M17	ICOCZ	**ad[15]**	W13	TS
A[19]	N20	ICOCZ	**ad[16]**	V9	TS
A[20]	N18	ICOCZ	**ad[17]**	Y8	TS
A[21]	P20	ICOCZ	**ad[18]**	W8	TS
A[22]	P19	ICOCZ	**ad[19]**	V8	TS
A[23]	P18	ICOCZ	**ad[20]**	Y7	TS
A[24]	R20	ICOCZ	**ad[21]**	V7	TS
A[25]	R19	ICOCZ	**ad[22]**	Y6	TS
A[26]	P17	ICOCZ	**ad[23]**	W6	TS
A[27]	R18	ICOCZ	**ad[24]**	W5	TS
A[28]	T20	ICOCZ	**ad[25]**	V5	TS
A[29]	T19	ICOCZ	**ad[26]**	Y3	TS

Table 2-15. 21285 Pin Name Pin List (Sheet 2 of 4)

Pin Name	PBGA Location	Type	Pin Name	PBGA Location	Type
ad[27]	U5	TS	**D[18]**	B10	ICOCZ
ad[28]	V4	TS	**D[19]**	A10	ICOCZ
ad[29]	W4	TS	**D[20]**	A11	ICOCZ
ad[30]	Y2	TS	**D[21]**	A12	ICOCZ
ad[31]	W3	TS	**D[22]**	C12	ICOCZ
ba[0]	D2	ICOCZ	**D[23]**	D12	ICOCZ
ba[1]	B1	ICOCZ	**D[24]**	A14	ICOCZ
cbe_l[0]	Y17	TS	**D[25]**	C14	ICOCZ
cbe_l[1]	Y13	TS	**D[26]**	A15	ICOCZ
cbe_l[2]	W9	TS	**D[27]**	B15	ICOCZ
cbe_l[3]	Y5	TS	**D[28]**	D14	ICOCZ
CLF	E18	IC	**D[29]**	C15	ICOCZ
cmd[0]	M4	OCZ	**D[30]**	A16	ICOCZ
cmd[1]	M3	OCZ	**D[31]**	B16	ICOCZ
cmd[2]	M2	OCZ	**DBE**	B14	ICOCZ
cs_l[0]	L4	OCZ	**devsel_l**	Y10	TS
cs_l[1]	L3	OCZ	**dqm[0]**	A5	ICOCZ
cs_l[2]	L2	OCZ	**dqm[1]**	B9	ICOCZ
cs_l[3]	L1	OCZ	**dqm[2]**	B13	ICOCZ
D[0]	C3	ICOCZ	**dqm[3]**	A17	ICOCZ
D[1]	B3	ICOCZ	**d_wren_l**	M1	OCZ
D[2]	D5	ICOCZ	**fclk**	D18	OCZ
D[3]	A3	ICOCZ	**fclk_in**	C20	IC
D[4]	B4	ICOCZ	**frame_l**	Y9	TS
D[5]	C5	ICOCZ	**gnt_l**	W1	I
D[6]	A4	ICOCZ	**idsel**	U7	I
D[7]	B5	ICOCZ	**irdy_l**	W10	TS
D[8]	B6	ICOCZ	**irq_in_l[0]**	J1	IC
D[9]	C7	ICOCZ	**irq_in_l[1]**	J2	IC
D[10]	A6	ICOCZ	**irq_in_l[2]**	J3	IC
D[11]	B7	ICOCZ	**irq_in_l[3]**	J4	IC
D[12]	A7	ICOCZ	**LOCK**	D20	IC
D[13]	C8	ICOCZ	**ma[0]**	H2	ICOCZ
D[14]	B8	ICOCZ	**ma[1]**	H3	ICOCZ
D[15]	A8	ICOCZ	**ma[2]**	G1	ICOCZ
D[16]	A9	ICOCZ	**ma[3]**	G3	ICOCZ
D[17]	D10	ICOCZ	**ma[4]**	F1	ICOCZ

Table 2-15. 21285 Pin Name Pin List (Sheet 3 of 4)

Pin Name	PBGA Location	Type	Pin Name	PBGA Location	Type
ma[5]	F2	ICOCZ	**serr_l**	V12	I, OD
ma[6]	F3	ICOCZ	**stop_l**	U11	TS
ma[7]	E1	ICOCZ	**tck**	P3	IC
ma[8]	E2	ICOCZ	**tdi**	N3	IC
ma[9]	E3	ICOCZ	**tdo**	P1	OCZ
ma[10]	C1	ICOCZ	**tms**	P2	IC
ma[11]	E4	ICOCZ	**trdy_l**	V10	TS
ma[12]	D3	ICOCZ	**trst_l**	R1	IC
MAS[0]	G17	IC	**tx**	T3	OCZ
MAS[1]	E20	IC	**Vdd**	A2	P
MCLK	C19	OCZ	**Vdd**	A20	P
nFIQ	U18	OCZ	**Vdd**	B12	P
nIRQ	U19	OCZ	**Vdd**	B18	P
nMREQ	A18	IC	**Vdd**	C2	P
nRESET	T17	ICOCZ	**Vdd**	C10	P
nRW	E19	IC	**Vdd**	C11	P
osc	D19	IC	**Vdd**	D6	P
par	U12	TS	**Vdd**	D11	P
parity[0]	C6	ICOCZ	**Vdd**	D15	P
parity[1]	D9	ICOCZ	**Vdd**	E17	P
parity[2]	A13	ICOCZ	**Vdd**	F4	P
parity[3]	C16	ICOCZ	**Vdd**	F17	P
pci_cfn	Y12	I	**Vdd**	G4	P
pci_clk	V2	I	**Vdd**	K3	P
pci_gnt_l[0]	V1	OCZ	**Vdd**	K4	P
pci_irq_l	W16	OD	**Vdd**	K18	P
pci_req_l[3]	T4	IC	**Vdd**	K20	P
pci_rst_l	U3	I, TS	**Vdd**	L17	P
perr_l	W12	TS	**Vdd**	R4	P
req_l	V3	O	**Vdd**	R17	P
rom_ce_l	N1	OCZ	**Vdd**	U6	P
rx	U2	IC	**Vdd**	U9	P
scan_en	N2	IC	**Vdd**	U10	P
sdclk[0]	C17	OCZ	**Vdd**	U15	P
sdclk[1]	B17	OCZ	**Vdd**	U20	P
sdclk[2]	A19	OCZ	**Vdd**	V6	P
sdclk[3]	C18	OCZ	**Vdd**	V11	P

Table 2-15. 21285 Pin Name Pin List (Sheet 4 of 4)

Pin Name	PBGA Location	Type	Pin Name	PBGA Location	Type
Vdd	Y1	P	**Vss**	K1	P
Vdd	Y4	P	**Vss**	K2	P
Vdd	Y11	P	**Vss**	K19	P
Vdd	Y20	P	**Vss**	N4	P
Vio	V19	I	**Vss**	N17	P
Vss	A1	P	**Vss**	N19	P
Vss	B2	P	**Vss**	U4	P
Vss	B11	P	**Vss**	U8	P
Vss	B19	P	**Vss**	U13	P
Vss	B20	P	**Vss**	U14	P
Vss	C4	P	**Vss**	U17	P
Vss	C9	P	**Vss**	W2	P
Vss	C13	P	**Vss**	W7	P
Vss	D1	P	**Vss**	W11	P
Vss	D4	P	**Vss**	W18	P
Vss	D7	P	**Vss**	W19	P
Vss	D8	P	**Vss**	Y15	P
Vss	D13	P	**xcs_l[0]**	U1	OCZ
Vss	D16	P	**xcs_l[1]**	T2	OCZ
Vss	D17	P	**xcs_l[2]**	R3	OCZ
Vss	G2	P	**xd_wren_l**	P4	OCZ
Vss	G18	P	**xior_l**	T1	OCZ
Vss	H4	P	**xiow_l**	R2	OCZ
Vss	H17	P	**—**	H1	NC

Transactions 3

All 21285 transactions occur between the SA-110, PCI, SDRAM, ROM, and X-Bus. Figure 3-1 shows the three FIFOs used in performing the various transactions.

- Outbound FIFO (256 bytes)
  - SA-110 write addresses and data for PCI
  - SA-110 read addresses for PCI
  - DMA write addresses and data for PCI
  - DMA read addresses for PCI
- Inbound FIFO (256 bytes)
  - PCI write addresses and data for SDRAM and ROM
  - PCI read addresses for SDRAM and ROM
  - DMA write data for SDRAM
  - SA-110 read data from PCI
- PCI Read FIFO (128 bytes)
  - PCI read data from SDRAM and ROM

3.1 SA-110 Originated Transactions

The following sections describe SA-110 originated bus transactions. These transactions are classified by their address spaces (refer to Table 5-1).

The 21285 controls the SA-110 clock, **MCLK,** by stretching the high time to stall the SA-110 during bus cycles (indicated by assertion of **nMREQ** by SA-110). The number of cycles that **MCLK** stays high may be different for each address space and depends upon other 21285 activity at the time **nMREQ** asserts.

3.1.1 CSR Write

The SA-110 write data is written to the selected CSR (the CSR with offset equal to SA-110 address **A[10:2]**). The SA-110 is normally stalled for three cycles, but can be stalled longer. If a nonexistent CSR is selected within the CSR address range, the write data is discarded (no error action is taken).

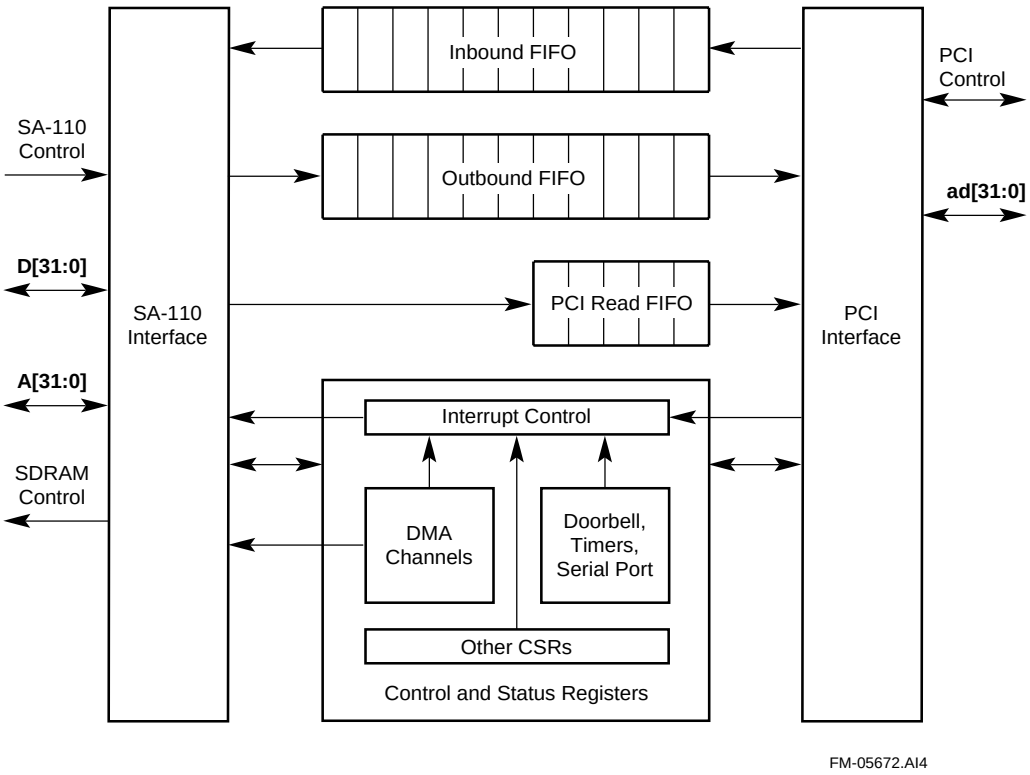
The CSR address space is listed in Tables 7-2, 7-3, and 7-4.

3.1.2 CSR Read

The selected CSR (the CSR with offset equal to SA-110 address **A[10:2]**) read data is driven onto **D[31:0]**. If a nonexistent CSR is selected within the CSR address range, zeros are read (no error action is taken). The SA-110 normally stalls for three cycles.

The CSR address space is listed in Tables 7-2, 7-3, and 7-4.

Figure 3-1. 21285 Block Diagram



3.1.3 SDRAM All-Banks Precharge

An SA-110 read from one of the four SDRAM array mode register regions causes an all-banks precharge to be issued to the associated SDRAM array. An all-banks precharge must be performed to each SDRAM array as part of the power-up initialization sequence, prior to a SDRAM mode register set or any other access to SDRAM.

The SA-110 is stalled while the precharge command is issued to the selected SDRAM array. The value on the low-order SA-110 address lines is ignored and read data should be ignored. During the precharge command, **ba[1:0]** is set to a value of 11 and **ma[12:8]** is set to a value of 11111. This ensures that, for any supported SDRAM type, the precharge command is interpreted as an all-banks precharge.

SDRAM vendors recommend that the **dqm** signals into the SDRAMs be forced high at reset, and held high until an all-banks precharge command is issued to the array. The 21285 complies with this recommendation by forcing the **dqm** signals high at reset, and holding them high until four all-bank precharge commands have been issued (the internal logic that counts these commands increments on all-banks precharge, mode register set, and refresh commands). The programmer should ensure that one all-banks precharge is issued to each array, even if fewer than four arrays are present.

3.1.4 SDRAM Mode Register Set

SDRAM mode register set occurs in response to an SA-110 write to one of four SDRAM array mode regions. SDRAMs require the mode register to be written after power-up and prior to accessing the SDRAM.

The SA-110 is stalled while a mode register set command is issued to the selected SDRAM array. There are four different mode register address regions; one for each array. The value of SA-110 address **A[8:2]** is the data written into the SDRAM mode register via the **ma[6:0]** pins**.** Pins **ba[1:0]** are 00 and **ma[12:7]** are 000000. The write data on **D[31:0]** is ignored. The SA-110 software must calculate the address within the mode register set address range in order to send the correct mode configuration information to each array of SDRAM. The SDRAM mode registers should be configured for a burst size of 4 with linear burst addressing.

3.1.5 SDRAM Write

The SA-110 is stalled while the selected SDRAM bank is addressed. It is unstalled after the data has been latched into the SDRAM from **D[31:0]**.

3.1.6 SDRAM First Read

The SA-110 is stalled while the selected SDRAM bank is addressed. It is unstalled when the first SDRAM data has been driven to the **D[31:0]** bus.

3.1.6.1 Lock

When **LOCK** is asserted by the SA-110, PCI or DMA originated accesses to SDRAM (see Section 3.2.2) will not occur; access waits until **LOCK** deasserts. (The SA-110 asserts **LOCK** during the read part of the SWP instruction, and deasserts it immediately following the write that completes the instruction).

3.1.7 ROM Read

The SA-110 is stalled while the ROM is read. Each Dword may require one, two, or four ROM reads, depending on the ROM width (latched at reset from the **ma** pins and readable in the SA-110 control register). The data is collected and packed into Dwords to be driven onto **D[31:0]**. When the ROM read(s) complete(s), the 21285 unstalls the SA-110. See Section 4.2 for a description of ROM accesses.

3.1.8 ROM Write

The SA-110 is stalled while the ROM is written. The ROM write data must be placed on the proper byte lanes by software running on the SA-110, that is, the data is not aligned in hardware by the 21285. Only one write is done regardless of the ROM width. When the ROM write completes, the 21285 unstalls the SA-110.

3.1.9 PCI Memory Write

The SA-110 address, write data, and byte masks are collected into the Outbound FIFO and written to PCI memory space at a later time. If there is room in the FIFO, the SA-110 stalls for one cycle. If there is no room in the FIFO, the SA-110 stalls until room is created by the unloading of some data to PCI.

Data with ascending addresses is placed in the Outbound FIFO in such a way as to be written to PCI in a burst. In order to maximize the burst size, the 21285 does not attempt to unload the Outbound FIFO until one or more of the following conditions occur:

- An ascending address written from SA-110
- 16 entries have been merged
- An SA-110 read
- Two SA-110 cycles with no bus activity
- The Outbound FIFO is full
- A 4KB address boundary is crossed

See the *Intel Semiconductor SA-110 Microprocessor Technical Reference Manual* for a description of the SA-110 write buffers and pin bus operation with sequential addresses.

3.1.10 PCI Memory Read

The SA-110 address is compared to a PCI prefetch address latch inside the 21285. If it matches, and prefetch data is valid, then the read data is driven on **D[31:0]** and the SA-110 is not stalled. If it does not match, or there is no valid PCI prefetch data, the SA-110 is stalled while the 21285 performs PCI memory read. It becomes unstalled when the read data is returned and driven to the **D[31:0]**. During the wait time, PCI write data in the Inbound FIFO (if any) may be written to the SDRAM or the ROM.

The 21285 may read more than one Dword (referred to as prefetching) from PCI depending on the contents of the prefetchable PCI range register (see Section 7.3.23). The prefetch read data is put into the Inbound FIFO. As each Dword is taken by the SA-110, PCI prefetch address latch is incremented by one Dword. If the SA-110 performs a read to an address that does not match PCI prefetch address latch, then all prefetched PCI read data is discarded. Also, if the SA-110 does not perform any external bus read cycles for two cycles, the prefetched read data is discarded; the two-cycle count is reinitialized each time some of the prefetched read data is consumed by the SA-110.

3.1.11 PCI I/O Write

The SA-110 address, write data, and byte mask are collected into the Outbound FIFO and written to PCI I/O space at a later time. If there is room in the FIFO, the SA-110 stalls for one cycle. If there is no room in the FIFO, the SA-110 stalls until room is created by the unloading of some data to PCI. Unlike a PCI memory write, no burst packing is done.

3.1.12 PCI I/O Read

The SA-110 is stalled while PCI controller performs PCI I/O read. It becomes unstalled when the read data is returned and driven to the **D[31:0]**. During the wait time, PCI write data in the Inbound FIFO (if any) can be written to the SDRAM and/or the ROM. No prefetching on PCI is done.

3.1.13 PCI Configuration Write

The SA-110 address, write data, and byte masks are collected into the Outbound FIFO and written to PCI configuration space at a later time. If there is room in the FIFO, the SA-110 stalls for one cycle. If there is no room in the FIFO, the SA-110 stalls until room is created by the unloading of some data to PCI. Unlike a PCI memory write, no burst packing is done.

Both type 0 and type 1 configuration cycles can be generated (see Table 3-5 and Section 5.1).

3.1.14 PCI Configuration Read

The SA-110 is stalled while the 21285 performs the PCI configuration read. It becomes unstalled when the read data is returned and driven to the **D[31:0]**. During the wait time, PCI write data in the Inbound FIFO (if any) can be written to the SDRAM and the ROM.

Both type 0 and type 1 configuration cycles can be generated (see Table 3-5 and Section 5.1).

If the PCI configuration read ends in a master abort, the read data is replaced by FFFFFFFFh. Typically, this situation occurs when initialization code running on the SA-110 is scanning PCI bus to determine what devices are present.

No prefetching on PCI occurs during a PCI configuration read.

3.1.15 PCI Special Cycle

The SA-110 address, write data, and byte masks are collected into the Outbound FIFO and written as a PCI special cycle at a later time. If there is room in the FIFO, the SA-110 stalls for one cycle. If there is no room in the FIFO, the SA-110 stalls until room is created by the unloading of some data to PCI. Unlike a PCI memory write, no burst packing is done.

3.1.16 PCI IACK Read

The SA-110 is stalled while the 21285 performs the PCI IACK read. It becomes unstalled when the read data is returned and driven to the **D[31:0]**. During the wait time, PCI write data in the Inbound FIFO (if any) can be written to the SDRAM or the ROM.

No prefetching on PCI occurs during a PCI IACK read.

3.1.17 X-Bus Write

The SA-110 is stalled while the 21285 performs the write to the X-Bus device. The SA-110 address and write data are driven to the X-Bus through transceivers. The SA-110 becomes unstalled when the write is complete.

3.1.18 X-Bus Read

The SA-110 is stalled while the 21285 performs the read from the X-Bus device. It becomes unstalled when the read data is returned and driven to the **D[31:0]** (the software must be aware of the width of the device data, and ignore the bytes of the **D** bus that are not driven by the X-Bus device).

3.1.19 Outbound Write Flush

This range of addresses can be used to force all pending translations in the Outbound FIFO to be written to the PCI.

For a read that is within the Outbound Write Flush address range, the SA-110 is stalled while all writes in the Outbound FIFO (if any) are delivered. When the last data (if any) is written to PCI, the SA-110 is unstalled. The read data returned is undefined.

For a write that is within the Outbound Write Flush address range, the SA-110 is stalled for several cycles, and an internal flag is set in the 21285. The write data is discarded. The flag clears when the Outbound FIFO is empty. Any subsequent SA-110 write to PCI is stalled until the flag is cleared.

3.1.20 SA-110 Cache Flush

SA-110 accesses to the SA-110 cache flush region are not stalled. Write data is discarded and read data is undefined. The SA-110 software can use this region to displace data from internal caches by doing reads that evict dirty data to memory.

Refer to the *Intel Semiconductor SA-110 Microprocessor Technical Reference Manual* for details describing the software Dcache algorithm.

3.1.21 Reserved Addresses

If the SA-110 attempts an access to the reserved address space, it stalls for several cycles. Write data can *possibly* be written to some other destination (that is, CSRs PCI memory space, and so on). Read data is undefined.

3.2 PCI Target Transactions

The 21285 signals retry to the PCI master in response to all configuration type 0 reads and writes (if its **idsel** is asserted) until the SA-110 has set the initialize complete bit ([0]) in the SA-110 control register. This gives the SA-110 software the chance to initialize registers, such as the SDRAM base address mask, prior to the host configuration code being able to read them.

The command register memory space enable bit ([1]) must be set for the 21285 to respond to any memory transaction.

The command register I/O space enable bit ([0]) must be set for the 21285 to respond to any I/O transaction.

Note: Some master devices, such as Intel Semiconductor's 21150, 21152, 21153, and 21154 PCI-to-PCI Bridge chips, discard a transaction after 2^{24} retries. This takes approximately 1.9 seconds, which implies that the SA-110 should set the initialize complete bit before that time.

The following sections describe the target response of the 21285 to various PCI cycles.

3.2.1 Unsupported PCI Cycles As Target

The following PCI transactions are not supported by the 21285 as a target:

- I/O write to SDRAM
- I/O read to SDRAM
- I/O write to ROM
- I/O read to ROM
- Type 1 configuration write
- Type 1 configuration read
- Special cycle
- IACK cycle
- Dual-address cycle

The following commands are aliased:

- Memory write and invalidate to memory write
- Memory read line and memory read multiple to memory read for CSR and ROM address space only (that is, not SDRAM space)

3.2.2 Memory Write to SDRAM

PCI memory write to SDRAM occurs if the PCI address matches the SDRAM base address register (at offset 18h) or the CSR base address register (at offset 10h with an offset greater than FFFh), and the PCI command is either a memory write or a memory write and invalidate.

The PCI memory write data is collected in the Inbound FIFO and written to SDRAM at a later time. The 21285 requests the SDRAM at the end of each eight Dword boundary of the PCI burst (or at the end of the burst).

If PCI address bits [1:0] are not 00 (that is, not linear increment mode), and the master attempts to continue the burst past the first Dword, the 21285 signals a target disconnect.

The PCI address is mapped down to local address 0 and then mapped up by the offset (see Figure 3-2).

Figure 3-2. Mapping PCI Address to SDRAM Address

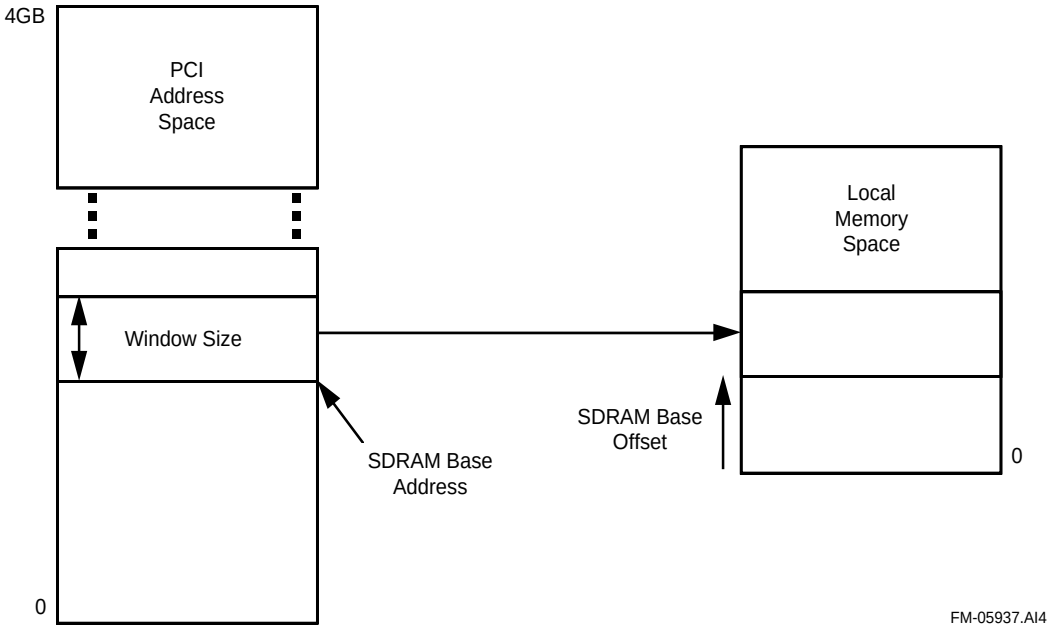


Table 3-1 describes how the SDRAM address is derived from the PCI address.

Note: If the match is to the SDRAM base address register, then the SDRAM base offset is used. If the match is to the CSR base address register, then the CSR base offset is used.

Table 3-1. SDRAM Address Generation

Bits	Description
27:18	If the corresponding bit of the base address mask register is a 0, use the base offset register bit. If the corresponding bit of the base address mask register is a 1, use the PCI address bit.
17:2	PCI address bits [17:2].
1:0	00 indicating an aligned Dword address.

Subsequent SDRAM addresses in the burst are incremented by four.

The SDRAM **dqm** signals are asserted for each Dword according to the corresponding byte enable from PCI. If the byte enable is not asserted, then the corresponding **dqm** signal is not asserted.

If fewer than five Dwords are available in the Inbound FIFO at the start of the write, the 21285 signals retry to the PCI master. If the Inbound FIFO fills during the write, the 21285 signals target disconnect to the PCI master.

3.2.3 Memory Read, Memory Read Line, Memory Read Multiple to SDRAM

PCI memory read from SDRAM occurs if the PCI address matches the SDRAM base address register (at offset 18h) or the CSR base address register (at offset 10h with an offset greater than FFFh), and the command is either a memory read, memory read line, or memory read multiple.

The read is completed as a PCI delayed read. On the first occurrence of the read, the 21285 signals a retry to the PCI master. If the delayed read latch is not full, the 21285 latches the address and command, and places it into the Inbound FIFO.

When the address reaches the head of the FIFO, the 21285 reads the SDRAM.

Table 3-1 describes how the SDRAM address is derived from the PCI address.

The amount of data that is read is a function of the following PCI commands:

- Memory read—One to four Dwords depending on the PCI address.

PCI Address Bits [3:2]	Number of Dwords Read
00	4
01	3
10	2
11	1

- Memory read line—Up to one cache line. The amount of data read is from the PCI address up to the top Dword of the cache line. If the cache line size is not 4 or 16, then the value of 8 is used for the cache line.
- Memory read multiple—Up to 32 Dwords. The amount of data read is from the PCI address up to the top Dword of the 32 Dword aligned block.

Exceptions to the above rules for the memory read line and memory read multiple commands are:

- If the PCI address bits [1:0] are not 00 (that is, not linear increment mode), read only one Dword.
- Reads do not cross a SDRAM page boundary.

When the read data is read from SDRAM into the PCI read FIFO, the 21285 begins decrementing its discard timer. If the PCI bus master has not repeated the read by the time the discard timer reaches zero, the 21285 discards the read data, invalidates the delayed read address, and sets a bit in the SA-110 control register (which interrupts the SA-110 if enabled). The discard timer counts 2^{15} (32768) PCI clocks.

When the master repeats the read command, the 21285 compares the address and checks that the command is a memory read, a memory read line, or a memory read multiple (that is, all memory read command types are aliased for a match). If there is a match, the response is as follows:

- If the read data has not yet been read from SDRAM, the response is retry.
- If the read data has been read from SDRAM, assert **trdy_l** and deliver the data. If the master attempts to continue the burst past the amount of data read from SDRAM, the 21285 target disconnects.

If the delayed read latch is full and a new read to a different address is attempted, that read gets a retry and no change is made in the delayed read latch. In other words, the new read does *not* displace the in-progress read.

A memory read multiple command initiates streaming prefetch from SDRAM so that it can supply read data at the maximum PCI bus data rate. Streaming prefetch works as follows:

- The 21285 reads 32 Dwords when the read is initially queued. This means that the address must be aligned to 32 Dwords.
- When the PCI master repeats the read, the 21285 starts to deliver data. When the PCI master consumes the 17th Dword, the 21285 reads the next 16 Dwords from SDRAM.
- As long as the PCI master continues to consume the 16th Dword of subsequent blocks, the 21285 reads the next 16 Dwords.

The 21285 never stalls **trdy_l** while supplying SDRAM read data.

Streaming prefetch does not start if a PCI write to SDRAM or ROM is queued into the Inbound FIFO between the delayed read being started and the read data being delivered. This allows the write data to be written to SDRAM. In this case, only the first 32 Dwords are read.

Streaming prefetch stops when any of the following events occur:

- The address crosses a SDRAM page boundary.
- The PCI master ends the cycle (by deasserting **frame_l**). In this case, the 21285 discards any remaining prefetched data immediately.
- Other bus activity on the SA-110 bus prevents data from being available in time; the 21285 will never negate **trdy_l** for the burst

3.2.4 Type 0 Configuration Write

PCI configuration write to the CSR occurs when **idsel** is asserted, the PCI command is a configuration write, and the PCI address bits [1:0] are 00. The PCI write data is written to the CSR selected by PCI address bits [7:2]. The PCI byte enables determine which bytes are written. If a nonexistent CSR is selected within the CSR address range, the data is discarded and no error action is taken.

If the PCI master attempts to do a burst longer than one data phase, the 21285 target disconnects.

3.2.5 Type 0 Configuration Read

PCI configuration read to the CSR occurs when **idsel** is asserted, the PCI command is a configuration read, and the PCI address bits [1:0] are 00. The data from the CSR selected by PCI address bits [7:2] is returned on **ad[31:0]**. If a nonexistent CSR is selected within the CSR address range, the data returned is zeros and no error action is taken.

If the PCI master attempts to do a burst longer than one data phase, the 21285 target disconnects

.

3.2.6 Write to CSR

PCI write to a CSR occurs if either of the following conditions are satisfied:

- The PCI address matches the CSR memory base address register (see Section 7.1.11), and the PCI command is either a memory write or memory write and invalidate.
- The PCI address matches the CSR I/O base address register (see Section 7.1.12), and the PCI command is an I/O write.

The data is written to the CSR with offset equal to PCI address bits [7:2]. The I_2O CSR addresses, offsets 40h and 44h, are handled differently (see Section 3.2.8). The PCI byte enables determine which bytes are written. If a nonexistent CSR is selected within the CSR address range, the data is discarded and no error action is taken.

If the PCI master attempts to do a burst longer than one data phase, the 21285 target disconnects.

3.2.7 Read to CSR

PCI read to a CSR occurs if either of the following conditions are satisfied:

- The PCI address matches the CSR memory base address register (see Section 7.1.11), and the PCI command is either memory read, memory read line, or memory read multiple.
- The PCI address matches the CSR I/O base address register (see Section 7.1.12), and the PCI command is an I/O read.

The data from the CSR with offset equal to PCI address [7:2] is returned onto **ad[31:0]**. The I_2O CSR addresses, offsets 40h and 44h, are handled differently (see Section 3.2.9). If a nonexistent CSR is selected within the CSR address range, the data returned is zeros.

If the PCI master attempts to do a burst longer than one data phase, the 21285 target disconnects.

3.2.8 Write to I_2O Address

PCI write to I_2O address space occurs when the PCI address matches the CSR memory base address register (see Section 7.1.11), the PCI command is either memory write or memory write and invalidate, and the register offset is either 40h or 44h. The write data is placed into the Inbound FIFO. When it reaches the head of the FIFO, it is written into SDRAM at the address in the inbound post list tail pointer register (for offset 40h), or the outbound free list tail pointer register (for offset 44h).

If the PCI master attempts to do a burst longer than one data phase, the 21285 target disconnects.

Write data is discarded if the PCI address matches the CSR I/O base address register (see Section 7.1.12), the PCI command is an I/O write, and the register offset is either 40h or 44h.

3.2.9 Read to I_2O Address

PCI read to I_2O address space occurs when the PCI address matches the CSR memory base address register (see Section 7.1.11); the PCI command is either memory read, memory read line, or memory read multiple; and the register offset is either 40h or 44h.

This read is completed as a delayed read. On the first occurrence of the read, the 21285 signals a retry to the PCI master. If the delayed read latch is not full, the 21285 latches the address and command, and places it into the Inbound FIFO.

When the read address reaches the head of the FIFO, the 21285 reads the SDRAM at the address in the inbound free list head pointer register (for offset 40h), or the outbound post list head pointer register (for offset 44h). If the list being read is empty, the value FFFFFFFFh is substituted for the data read from the SDRAM.

When the master repeats the read command, the 21285 compares the address and checks that the command is a memory read, a memory read line, or a memory read multiple (that is, all memory read command types are aliased for a match). If there is a match, the response is as follows:

- If the read data has not yet been read from SDRAM, the response is retry.
- If the read data has been read from SDRAM, assert **trdy_l** and deliver the data. If the PCI master attempts to do a burst longer than one data phase, the 21285 target disconnects.

If the delayed read latch is full and a new read from a different address is attempted, that read gets a retry and no change is made in the delayed read latch. In other words, the new read does *not* displace the in-progress read.

If the PCI address matches the CSR I/O base address register (see Section 7.1.12), the PCI command is an I/O read, and the register offset is either 40h or 44h. The 21285 returns 0 for read data.

For more information about CSR and I_2O registers, see Chapter 7.

3.2.10 Memory Write to ROM

PCI memory write to ROM occurs when the PCI address matches the expansion ROM base address register, bit [0] of the expansion ROM base address register is a 1, and the PCI command is either memory write or memory write and invalidate. The PCI memory write address and data is collected in the Inbound FIFO to be written to ROM at a later time. The 21285 target disconnects after one data phase.

Only a single ROM write cycle is done regardless of the PCI byte enable and the ROM width field settings in the SA-110 control register. In addition, the data to be written must be put into the proper byte lanes by software. See Section 4.2 for a description of the ROM write cycle and the way in which the ROM write address is derived from the PCI address.

3.2.11 Memory Read to ROM

PCI memory read to ROM occurs when the PCI address matches the expansion ROM base address register, bit [0] of the expansion ROM base address register is a 1, and the PCI command is either a memory read, memory read line, or memory read multiple.

The read is completed as a PCI delayed read. On the first occurrence of the read, the 21285 signals a retry to the PCI master. If the delayed read latch is not full, the 21285 latches the address and command, and places it into the Inbound FIFO.

When the address reaches the head of the FIFO, the 21285 reads the ROM. The ROM is read from one to four times, depending on the ROM width setting in the SA-110 control register, and the data is collected and packed. See Section 4.2 for a description of the ROM read cycle and the way in which the ROM read address is derived from the PCI address.

When the master repeats the read command, the response is one of the following:

- If the read data has not yet been read from ROM, the response is retry.
- If the read data has been read from ROM, assert **trdy_l** and deliver the data. If the master attempts to continue the burst beyond one Dword, the 21285 target disconnects.

If the delayed read latch is full and a new read from a different address is attempted, that read gets a retry and no change is made in the delayed read latch. In other words, the new read does *not* displace the in-progress read.

3.3 PCI Master Transactions

The following sections describe the PCI master transactions performed by the 21285. All PCI master transactions performed by the 21285 are caused by either SA-110 bus cycles that fall into the various PCI address ranges (see Section 5.1.1), or by DMA channels. PCI write cycles are caused by SA-110 writes and SDRAM-to-PCI DMAs. PCI read cycles are caused by SA-110 reads and PCI-to-SDRAM DMAs.

The command register bus master bit must be set for the 21285 to perform any of the transactions in this section.

3.3.1 Dual Address Cycles (DAC) Support

Rev_ID 4 or higher of the 21285 can generate DAC cycles as PCI bus master, enabling it to access any address in the full 2^{64} bit PCI memory space address range. DAC cycles can be done for PCI transaction types Memory Write, Memory Write and Invalidate, Memory Read, Memory Read Line, and Memory Read Multiple. For both SA-110 accesses and DMA accesses, the 21285's PCI bus master controller performs a DAC as needed (when all bits in address [63:32] are not 0). The address is sent onto AD low 32 bits first (in the same cycle as FRAME_L assertion) with a command code D, indicating DAC. In the next cycle, the high 32 bits of address are driven on AD with the appropriate command code, such as Memory Read and Memory Write, as shown in the PCI specification

3.3.1.1 For SA–110 Accesses

PCI memory accesses to an address above 4GB are controlled by the contents of two registers which are set up by the SA-110:

- SA–110 DAC Address (Section 7.3.35)
- SA–110 DAC Control (Section 7.3.36)

The software must write the full, high 32 bits of the address, which becomes address [63:32] on the PCI bus. The actual load or store is then done to the PCI memory space address region; the offset into that region supplies the lowest 31 bits ([30:2], [1:0] will be 00b) of the address. Bit [31] of the PCI address is the value in bit [15] of the PCI Address Extension Register. This means that the software can address a 2GB region of PCI memory space without rewriting the registers.

Note that the SA–110 software must ensure that writes to the address registers and the actual loads and stores to PCI DAC space are performed as atomic operations. The following methods can provide this atomicity:

- Have each routine that modifies the DAC address registers save them prior to modifying, and restore them prior to returning.
- Allow only one process to use the address registers. All DAC accesses must be done as calls through that routine, which is run with interrupts disabled.
- Allow multiple processes that can use the address registers. All such processes must run with interrupts disabled during the critical region.
- Have multiple routines coordinate access to the address registers via a semaphore or equivalent method.

3.3.1.2 For DMA Accesses

Each DMA channel has a register which holds the upper 32 bits of the PCI address.

3.3.2 Memory Write, Memory Write and Invalidate

This section describes memory write and memory write and invalidate commands from either the SA-110 or DMA. The following general rules apply to the command transactions:

- If the 21285 receives either a target retry response or a target disconnect response before all of the write data has been delivered, it resumes the transaction at the first opportunity, using the address of the first undelivered Dword.
- If the 21285 receives a master abort, it discards all of the write data from that transaction and sets the status register received master abort bit [29], which, if enabled, interrupts the SA-110.
- If the 21285 receives a target abort, it discards all of the remaining write data from that transaction, if any, and sets the status register received target abort bit [28], which, if enabled, interrupts the SA-110.
- The 21285 can deassert **frame_l** prior to delivering all data due to the master latency timer (see Section 3.3.11). If this occurs, it resumes the memory write at the first opportunity, using the address of the first undelivered Dword.

3.3.2.1 From SA-110

PCI memory write or memory write and invalidate commands are completed when the SA-110 address writes to the PCI memory space. Table 3-2 shows how the PCI address is derived from the SA-110 address.

Table 3-2. SA-110 Address Generation

Bits	Description
31	From the PCI address extension register, PCI memory space upper address bit [15].
30:2	Equal to SA-110 address bits [30:2].
1:0	00 indicating linear increment mode.

The PCI byte enables for each data phase are derived from the SA-110 **A/MAS** signals. The length of the burst is determined by how much sequential data was collected into the Outbound FIFO (see Section 3.1.9).

3.3.2.2 From DMA

PCI memory write and memory write and invalidate commands are completed by a DMA channel programmed for SDRAM-to-PCI transfer. Table 3-3 shows how the PCI address is derived from the DMA address.

Table 3-3. DMA Address Generation

Bits	Description
31:2	From the DMA channel n PCI address register.
1:0	00.

The PCI byte enables are all asserted with the possible exception of the first or last Dword of a DMA transfer. For more information about DMA channels, see Section 6.2.

The length of the burst is normally 8 or 16 Dwords, with the possible exception of the first or last burst of a DMA transfer.

3.3.2.3 Selecting PCI Command for Writes

The 21285 attempts to use a memory write and invalidate command when the following conditions are met. If any condition is not met, the 21285 uses the memory write command.

- Memory write and invalidate enable bit [4] in the command register is a 1.
- Cache line size is set to a value of 4, 8, or 16.
- The address of the first Dword in the burst is aligned to a cache line as defined in the cache line size register.
- The number of Dwords in the burst is an integer multiple of the cache line size.
- There are no unoccupied bytes in the burst.

3.3.3 Memory Read, Memory Read Line, Memory Read Multiple

This section describes memory read, memory read line, and memory read multiple commands from either the SA-110 or DMA. The following general rules apply to the command transactions:

- If the 21285 receives a target retry response, it repeats the same PCI command at the first opportunity.
- If the 21285 receives a master abort, it substitutes FFFFFFFFh for the read data and sets the status register received master abort bit [29], which, if enabled, interrupts the SA-110.
- If the 21285 receives a target abort, it sets the status register received target abort bit [28], which, if enabled, interrupts the SA-110.

3.3.3.1 From SA-110

PCI memory reads are done when the SA-110 performs reads from the memory space. The actual PCI command used is based on whether or not there is a match to the prefetchable PCI range register. For no match, the command is a memory read. For a match, the command type is specified in the prefetchable PCI range register. Table 3-2 shows how the PCI address is derived from the SA-110 address.

For memory read, the PCI byte enables are derived from the SA-110 **A/MAS** signals. For memory read line and memory read multiple, the PCI byte enables assert for all data phases.

For memory read, one Dword is read. For memory read line and memory read multiple, the maximum number of Dwords that can be read is specified in the prefetchable memory range register.

3.3.3.2 From DMA

The PCI command used is specified by the DMA channel PCI read type field in the DMA channel control register. Table 3-3 shows how the PCI address is derived from the DMA address.

The PCI byte enables assert for all data phases.

The maximum number of Dwords that can be read is specified in the DMA channel control register.

3.3.3.3 Read Bursting Policy

The 21285 attempts to read Dwords from the start address upwards, and continues to accept read data until either the target disconnects or the 21285 deasserts **frame_l**. This situation occurs when the 21285 has read all the data that it requires, or the Inbound FIFO becomes full. However, all Dwords beyond the first Dword are optional. If the target disconnects after delivering the first Dword, the 21285 does not resume the read at that time (for a DMA originated read, it resumes at the address of the next unread Dword after the normal channel interburst delay). For an SA-110 read, all Dwords that have been read are eligible to be delivered to the SA-110 as prefetched read data. For a DMA read, all Dwords that have been read are written to SDRAM.

3.3.4 I/O Write

The I/O write is completed when the SA-110 address is in the PCI I/O space. Table 3-4 shows how the PCI address is derived from the I/O address.

Table 3-4. I/O Address Generation

Bits	Description
31:16	From the PCI address extension register, PCI I/O space upper address bits [31:16].
15:0	Equal to SA-110 address bits [15:0].

The PCI byte enables for each data phase are derived from the SA-110 **A/MAS** signals. One Dword is written.

The following general rules apply to the command transactions:

- If the 21285 receives a target retry response, it repeats the I/O write command at the first opportunity.
- If the 21285 receives a master abort, it discards the write data and sets the status register received master abort bit [29], which, if enabled, interrupts the SA-110.
- If the 21285 receives a target abort, it discards the write data and sets the status register received target abort bit [28], which, if enabled, interrupts the SA-110.

3.3.5 I/O Read

I/O read is completed when the SA-110 address is in the PCI I/O space. Table 3-4 shows how the PCI address is derived from the I/O address.

The PCI byte enables for each data phase are derived from the SA-110 **A/MAS** signals. One Dword is read.

The following general rules apply to the command transactions:

- If the 21285 receives a target retry response, it repeats the I/O read command at the first opportunity.
- If the 21285 receives a master abort, it substitutes FFFFFFFFh for the read data and sets the status register received master abort bit [29], which, if enabled, interrupts the SA-110.
- If the 21285 receives a target abort, it substitutes FFFFFFFFh for the read data and sets the status register received target abort bit [28], which, if enabled, interrupts the SA-110.

3.3.6 Configuration Write

Configuration write is completed when the SA-110 address is in the PCI configuration space. Table 3-5 shows how the PCI address is derived from the configuration address.

Table 3-5. Configuration Address Generation (Sheet 1 of 2)

Bits	Description	
	Note: **If SA-110 address bits [23:22] are not equal to 11, or the SA-110 address is in PCI type 1 configuration space, then the PCI address is derived directly from the SA-110 address as shown in the next three row entries**.	
31:24	All bits are 0.	
23:2	Equal to SA-110 address bits [23:2].	
1:0	If the SA-110 address is in PCI type 0 configuration space, it equals 00. If the SA-110 address is in PCI type 1 configuration space, it equals 01.	
	Note: **If SA-110 address bits [23:22] are equal to 11, and the SA-110 address is in PCI type 0 configuration space, then the PCI address is derived directly from the SA-110 address bits [15:11]**.	
31:11	Derived from the following decodes:	
	SA-110 A[15:11]	PCI ad[31:11]
	00000	00000000000000000001

Table 3-5. Configuration Address Generation (Sheet 2 of 2)

Bits	Description	
31:11 (Cont.)	Derived from the following decodes (Cont.):	
	00001	00000000000000000010
	00010	00000000000000000100
	00011	00000000000000001000
	00100	00000000000000010000
	00101	00000000000000100000
	00110	00000000000001000000
	00111	00000000000010000000
	01000	00000000000100000000
	01001	00000000001000000000
	01010	00000000010000000000
	01011	00000000100000000000
	SA-110 A[15:11]	PCI ad[31:11]
	01100	00000001000000000000
	01101	00000010000000000000
	01110	00000100000000000000
	01111	00001000000000000000
	10000	00010000000000000000
	10001	00100000000000000000
	10010	01000000000000000000
	10011	10000000000000000000
	10100	00000000000000000000
	10101 through 11111	00000000000000000000
10:2	Equal to SA-110 address bits [10:2].	
1:0	00.	

It is the responsibility of SA-110 software to generate an address that is meaningful for the PCI configuration cycle. For example, normally **ad[23:11]** bits are used for **idsel** of PCI devices during a type 0 configuration cycle, so only one of those bits is a 1.

The PCI byte enables for each data phase are derived from the SA-110 **A/MAS** signals. One Dword is written.

The following general rules apply to the command transactions:

- If the 21285 receives a target retry response, it repeats the configuration write command at the first opportunity.
- If the 21285 receives a master abort, it discards the write data and sets the status register received master abort bit [29], which, if enabled, interrupts the SA-110.
- If the 21285 receives a target abort, it discards the write data and sets the status register received target abort bit [28], which, if enabled, interrupts the SA-110.

3.3.7 Configuration Read

Configuration read is completed when the SA-110 address is in the PCI configuration space. Table 3-5 shows how the PCI address is derived from the configuration address.

The PCI byte enables for each data phase are derived from the SA-110 **A/MAS** signals. One Dword is read.

The following general rules apply to the command transactions:

- If the 21285 receives a target retry response, it repeats the configuration read command at the first opportunity.
- If the 21285 receives a master abort, it substitutes FFFFFFFFh for the read data and sets the status register received master abort bit [29], which, if enabled, interrupts the SA-110.
- If the 21285 receives a target abort, it substitutes FFFFFFFFh for the read data and sets the status register received target abort bit [28], which, if enabled, interrupts the SA-110.

3.3.8 Special Cycle

Special cycle is completed when the SA-110 address is in the PCI IACK/Special space. The special cycle is caused by an SA-110 write. The PCI address is undefined.

The PCI byte enables for each data phase are derived from the SA-110 **A/MAS** signals. One Dword is written.

Special cycles are broadcast to all PCI agents, so **devsel_l** is not asserted and no errors can be received.

3.3.9 IACK Read

IACK read is completed when the SA-110 address is in the PCI IACK/Special space. An IACK read is caused by an SA-110 read. The PCI address is undefined.

The PCI byte enables for each data phase are derived from the SA-110 **A/MAS** signals. One Dword is read.

The following general rules apply to the command transactions:

- If the 21285 receives a target retry response, it repeats the IACK read at the first opportunity.
- If the 21285 receives a master abort, it substitutes FFFFFFFFh for the read data and sets the status register received master abort bit [29], which, if enabled, interrupts the SA-110.
- If the 21285 receives a target abort, it substitutes FFFFFFFFh for the read data and sets the status register received target abort bit [28], which, if enabled, interrupts the SA-110.

3.3.10 PCI Request Operation

The 21285 asserts **req_l** to act as bus master on the PCI for SA-110 and DMA originated transactions. It deasserts **req_l** for two cycles when it receives a retry or disconnect response from the target. However, if **gnt_l** is asserted, the 21285 can start a PCI transaction regardless of the state of **req_l**.

When the 21285 requests the PCI bus, it performs a PCI transaction when **gnt_l** is received. Once **req_l** is asserted, the 21285 never deasserts it *prior* to receiving **gnt_l** (nor deasserts it *after* receiving **gnt_l** without doing a transaction).

3.3.11 Master Latency Timer

When the 21285 begins a PCI transaction as master, asserting **frame_l,** it begins decrementing its master latency timer. When the timer value reaches zero, the 21285 checks the value of **gnt_l**. If **gnt_l** is deasserted, the 21285 deasserts **frame_l** (if it is still asserted) at the earliest opportunity. This is normally the next data phase for all transactions except for the memory write and invalidate command (MWI). For MWI, it is at the data phase at the top of a cache line.

When the command is MWI and the latency timer expires while the last Dword of a cache line is being delivered, and **frame_l** is still asserted, the master will deliver the entire next cache line before stopping the transaction.

3.4 PCI Error Summary

This section summarizes the various PCI error conditions and the response of the 21285 to these conditions.

3.4.1 Errors As PCI Target

PCI target errors are listed as follows:

- Address parity error
- Write data parity error
- Read data parity error

3.4.1.1 Address Parity Error

An address parity error is detected when **par** driven by the PCI master does not match the expected parity for the address and command. This causes the following actions to occur:

- The status register detected parity error bit [31] is set.
- If the (potentially corrupted) address or command matches any of the 21285 base address registers, then the 21285 claims the cycle and proceeds as though the address was correct.
- If the command register parity error response bit [6] is a 1, command register SERR enable bit [8] is a 1, and **pci_cfn** is a 0, then **serr_l** is asserted for one cycle and the status register signaled system error bit [30] is set.

3.4.1.2 Write Data Parity Error

A write data parity error is detected when the **par** that is received by the 21285 does not match the expected parity for data and byte enables. This causes the following actions to occur:

- The status register detected parity error bit [31] is set.
- If the command register parity error response bit [6] is a 1, then:
  - **perr_l** is asserted.
  - If the data destination is SDRAM, and SDRAM parity is enabled, incorrect parity is written to the SDRAM. The write is completed to the intended destination (that is, CSR, SDRAM, or ROM) despite the detection of a parity error.

3.4.1.3 Read Data Parity Error

A read data parity error is detected when the PCI master asserts **perr_l** in response to read data driven by the 21285. No action is taken by the 21285.

3.4.2 Errors As PCI Master

PCI master errors are listed as follows:

- Master abort
- Write data parity error
- Target abort on write
- Read data parity error
- Target abort on read

3.4.2.1 Master Abort

A master abort occurs when **devsel_l** is not asserted within five cycles after the 21285 asserts **frame_l.** This causes the following actions to occur:

- Status register received master abort bit [29] is set (except if the transaction is a special cycle).
- If the transaction was a write from SA-110 or DMA, all write data for the transaction is discarded.
- If the transaction was an SA-110 read, FFFFFFFFh is inserted for read data.
- If the transaction was a DMA read or write, channel *n* error bit [3] in the channel control register is set, stopping the channel.

3.4.2.2 Write Data Parity Error

A write data parity error occurs when the PCI target asserts **perr_l** in response to write data driven by the 21285. This causes the following actions to occur:

- If the command register parity error response bit [6] is a 1, then:
  - — The status register detected parity error data bit [24] is set.
  - — If the transaction was a DMA, channel *n* error bit [3] in the channel control register is set, stopping the channel.

3.4.2.3 Target Abort on Write

When the PCI target signals a target abort, the following actions occur:

- Status register received target abort bit [28] is set.
- Any remaining write data is discarded.
- If the transaction was a DMA, channel *n* error bit [3] in the channel control register is set, stopping the channel.

3.4.2.4 Read Data Parity Error

A read data parity error is detected when the **par** that is received by the 21285 does not match the expected parity for data and byte enables. This causes the following actions to occur:

- The status register detected parity error bit [31] is set.
- If the command register parity error response bit [6] is set, then the status register data parity error detected bit [24] is set and the 21285 asserts **perr_l**.
- Dwords with bad parity are marked as such in the Inbound FIFO.
  - — For DMA operations, if SDRAM parity is enabled, incorrect parity is written to the SDRAM. The channel *n* error bit [3] in the channel control register is set, stopping the channel.

3.4.2.5 Target Abort on Read

When the PCI target signals a target abort, the following actions occur:

- If the read is a demand read for the SA-110, it substitutes FFFFFFFFh for the read data.
- If the read was prefetched for the SA-110, all prefetching stops. All data prefetched prior to the target abort can be delivered to the SA-110 using a normal prefetch operation.
- If the transaction was a DMA, channel *n* error bit [3] in the channel control register is set, stopping the channel.

SDRAM and ROM Operation 4

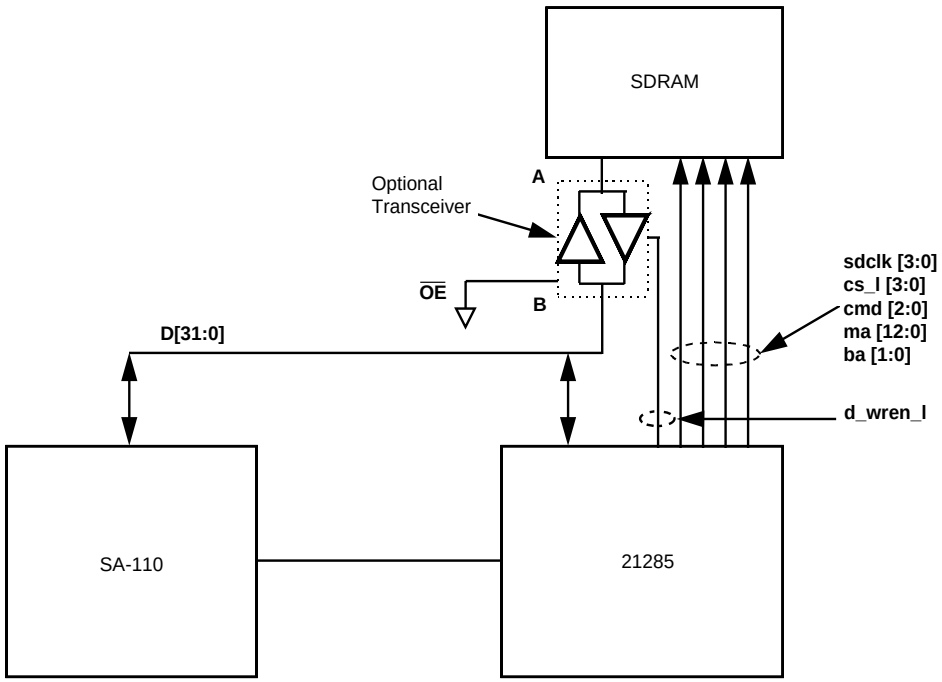
This chapter describes the operation of the SDRAM and ROM.

4.1 SDRAM Control

The SDRAM controller on the 21285 controls from one to four arrays of synchronous DRAMs (SDRAMs). SDRAM supported parts include: 8Mb, 16Mb, and 64Mb.

All SDRAMs share command and address bits, but have separate clock and chip select bits (see Figure 4-1).

Figure 4-1. SDRAM Configuration



Note:
When SA-110 reads or writes SDRAM, the data does not pass through 21285.

FM-05938.AI4

SDRAM operations performed by the 21285 are refresh, read, write, and mode register set. Reads and writes are generated by either the SA-110, PCI bus masters (including I_2O accesses), or DMA channels.

Mode register set is generated in response to an SA-110 write to a specified address space (see Section 5.1.1). The mode register of the SDRAMs must be written before any SDRAM writes or reads can be done. The mode register burst length field (bits [2:0]) must be written to a value of 2h to indicate a burst size of four, and the burst type field (bit [3]) must be written to a value of 0h to indicate sequential burst.

Memory reads and writes are done in bursts of four Dwords. For a read that requires less than four Dwords (for example, a memory read from PCI), the 21285 discards the unused data. For a write that requires less than four Dwords, the 21285 uses the **dqm** pins to inhibit writing to some Dwords. The **dqm** pins also inhibit writing to unoccupied bytes. Table 4-1 lists the available array sizes. The row/column multiplexer mode is the value programmed into the SDRAM address and size register (see Section 7.3.13).

Table 4-1. Array Sizes

SDRAM Type			Address Bits			SDRAMs	Array	Row/Column
Banks	Depth	Width	Bank	Row	Col.	in Array	Size	Multiplexer Mode
8Mb Parts								
2	128K	32	1	9	8	1	1MB	000
16Mb Parts								
2	256K	32	1	10	8	1	2MB	000
2	512K	16	1	11	8	2	4MB	001
2	1M	8	1	11	9	4	8MB	001
2	2M	4	1	11	10	8	16MB	001
64Mb Parts								
2	1M	32	1	12	8	1	8MB	010
4	512K	32	2	11	8	1	8MB	011
2	2M	16	1	13	8	2	16MB	010
4	1M	16	2	12	8	2	16MB	100
2	4M	8	1	13	9	4	32MB	010
4	2M	8	2	12	9	4	32MB	100
2	8M	4	1	13	10	8	64MB	010
4	4M	4	2	12	10	8	64MB	100

4.1.1 SDRAM Addresses

The SDRAM addresses are driven on the **ma[12:0]** bits. In Table 4-2, the row address is always equal to bits [23:21], [18:9] of the address being accessed, however, some SDRAMs do not use all of the addresses. The column address is determined by the address multiplexer mode field of the SDRAM address and size register for the selected array. The array selection is based on bits [27:20] of the address (see Section 7.3.13). The SDRAM bank address is driven on **ba[1:0]** (indicating that it is an **A[20:19]** where **A[20]** may not be used) during both row and column time.

Note: "—" in Table 4-2 indicates that the address is not used by the SDRAM in this configuration. The pin is driven by the 21285 and its value can be either 0 or 1.

Note: "**ap**" in Table 4-2 indicates the autoprecharge bit that is used by the SDRAM during column address time. A low (deasserted), indicates no autoprecharge, which occurs during read and write commands when there is another burst pending to the same page. A high (asserted), indicates autoprecharge, which occurs during read and write commands of the last burst.

Banks are always closed by autoprecharge during read or write, not by using the autoprecharge command.

Table 4-2. SDRAM Addresses

			BA		SDRAM Address ma[12:0]												
Mode	**Size**[a]		**1**	**0**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
000	1	Row	—	19	—	—	—	—	17	16	15	14	13	12	11	10	9
000	1	Col.	—	19	—	—	—	—	**ap**	18	8	7	6	5	4	3	2
000	0	Row	—	19	—	—	—	18	17	16	15	14	13	12	11	10	9
000	0	Col.	—	19	—	—	—	**ap**	—	20	8	7	6	5	4	3	2
001	x	Row	—	19	—	—	21	18	17	16	15	14	13	12	11	10	9
001	x	Col.	—	19	—	—	**ap**	23	22	20	8	7	6	5	4	3	2
010	x	Row	—	19	23	22	21	18	17	16	15	14	13	12	11	10	9
010	x	Col.	—	19	—	—	**ap**	25	24	20	8	7	6	5	4	3	2
011	x	Row	20	19	—	—	21	18	17	16	15	14	13	12	11	10	9
011	x	Col.	20	19	—	—	**ap**	—	—	22	8	7	6	5	4	3	2
100	x	Row	20	19	—	22	21	18	17	16	15	14	13	12	11	10	9
100	x	Col.	20	19	—	—	**ap**	25	24	23	8	7	6	5	4	3	2

a. Size references the least significant bit of the array size field of the SDRAM address and size register (see Section 7.3.13).

4.1.2 Commands

The SDRAM command is driven onto the **cmd[2:0]** bits (RAS, CAS, and WE for the SDRAMs). The command is only valid when chip select (**cs_l**) is asserted (or as described in Section 7.3.12, on the cycle before **cs_l** is asserted). The **cs_l** asserts either during the same cycle as the command, or one cycle later depending on the value in the SDRAM timing register. Table 4-3 lists the SDRAM command usage.

Table 4-3. SDRAM Commands **(Sheet 1 of 2)**

SDRAM Command	cmd[2:0]	Description
Mode register set	000	Mode register set is issued in response to a SA-110 write to mode register address region.
Auto refresh	001	Auto refresh is done at the earliest time after refresh interval timer expiration, that is, after any SDRAM operation in progress finishes. All arrays are refreshed at the same time.
Precharge	010	Precharge is issued in response to a SA-110 read to the mode register address region. It is only used by initialization software to wake up the SDRAMs after power-up.

Table 4-3. SDRAM Commands (Sheet 2 of 2)

SDRAM Command	cmd[2:0]	Description
Bank activate	011	A bank activate command is required before the start of a read or write if the bank is not open. This command can only follow an earlier read or write with autoprecharge or an auto refresh. Row address is driven on **ma[12:0]** and bank address is driven on **ba[1:0]** at the same time.
Write	100	Write command follows an earlier bank activate, or a read or write without autoprecharge. Column address is driven on **ma[12:0]** and bank address is driven on **ba[1:0]** at the same time. Four Dwords from either the SA-110 or 21285, and also the corresponding **dqm[3:0]** from the 21285, are driven on the same cycle and the next three sequential cycles. If fewer than four Dwords are written, the **dqm[3:0]** inhibit writing.
Read	101	Read command follows an earlier bank activate, or a read or write without autoprecharge. Column address is driven on **ma[12:0]** and bank address is driven on **ba[1:0]** at the same time. From one to four Dwords are latched by either the SA-110 or 21285 after CAS latency.

4.1.3 Parity

The SDRAM can be optionally protected by byte parity. Parity is enabled by bit [12] in the SDRAM timing register. When parity is enabled, the 21285 drives even parity for each of the bytes of the **D[31:0]** bus (refer to Table 2-6) for SDRAM writes, and receives and compares parity for SDRAM reads. When parity is disabled, the **parity** pins are tristated and ignored for reads. In this case, the parity pins must not be left floating; connect them to V_{ss} or V_{dd}.

When writing from the PCI or a DMA channel, **parity** is driven at the same time as data.

During a write from the SA-110, the operation depends on bit [13] of the SDRAM timing register.

- If a 0, data is received by the 21285 and flows through an internal parity generator onto **parity**. Therefore, it is necessary to run a parity-enabled memory subsystem slower than a comparable parity-disabled memory to allow for the parity computation.
- If a 1, the SA-110 provides the parity information for the write. The 21285 leaves its parity pins tristate.

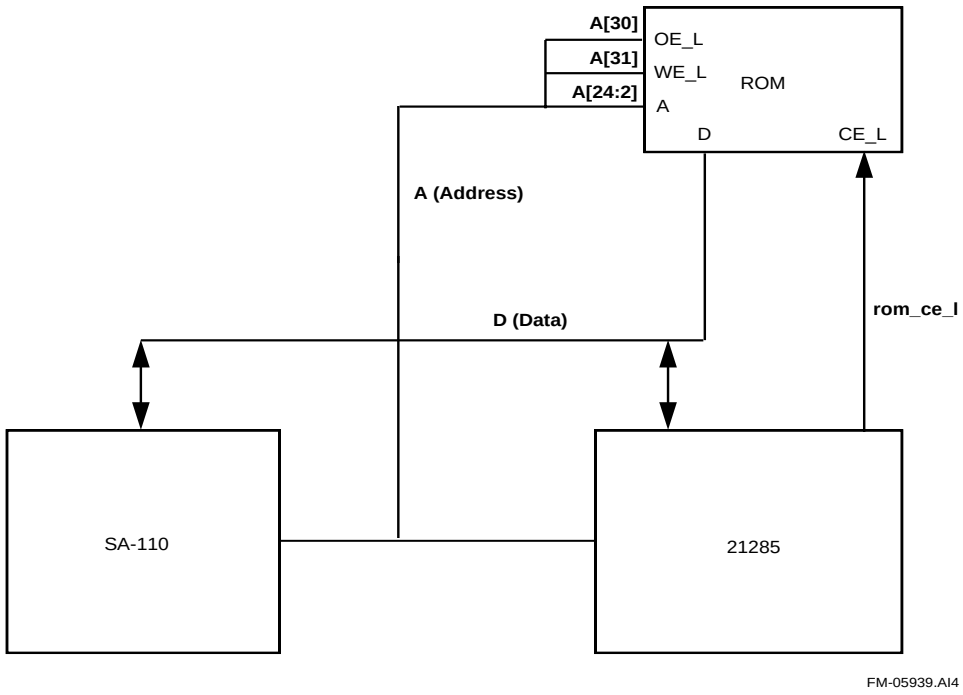
When a SDRAM read has bad parity, the following results:

- SA-110 originated read transaction—data is delivered to the SA-110 as is. Bit [4] sets in the SA-110 control register, which interrupts the SA-110 if enabled.
- PCI originated read transaction—bad parity is delivered to the PCI bus master with the data. Bit [5] sets in the SA-110 control register.
- DMA channel originated read transaction—bad parity is not sent by the 21285 to the PCI target. Bit [6] sets in the SA-110 control register.

4.2 ROM Control

Figure 4-2 shows the ROM configuration. The ROM output enable and write enable are connected to address bits [30:31] respectively. The ROM address is connected to address bits [24:2].

Figure 4-2. ROM Configuration



This section describes the following aspects of the 21285 ROM interface:

- Addressing
- Reads
- Writes
- Timing
- Blank ROM programming

4.2.1 Addressing

The ROM can always be addressed by the SA-110 at 41000000h through 41FFFFFFh as listed in Table 5-1. After reset, the ROM is also aliased at every 16 megabytes throughout memory space, blocking access to SDRAM. This allows the SA-110 to boot from ROM at address 0. After any SA-110 write, the alias address range is disabled.

The ROM address pins should be connected to the SA-110 address (**A**) pins depending on the ROM width as described in Table 4-4.

Table 4-4. ROM Addressing

ROM Width	ROM Address [23:2]	ROM Address [1]	ROM Address [0]
Byte	**A[23:2]**	**A[29]**	**A[28]**
Word (2 bytes)	0, **A[23:3]**	**A[2]**	**A[29]**
Dword (4 bytes)	00, **A[23:4]**	**A[3]**	**A[2]**

During ROM accesses from the SA-110, the 21285 latches the address and drives it back onto **A**. Table 4-5 shows how the ROM address is derived from the SA-110 address.

Table 4-5. ROM Address Generation for SA-110 Accesses

Bits	Description
A[29:28]	Controlled by the 21285 depending upon the ROM width (refer to Sections 4.2.2 and 4.2.3).
A[23:5]	Same value that was driven by the SA-110.
A[4:2]	Initial value as driven by the SA-110. For cache line fills, subsequent values are generated by the 21285.
A[1:0]	Driven, but not defined.

During a ROM access from the PCI, **A** is driven by the 21285. Table 4-6 shows how the ROM address is derived from the PCI address.

Table 4-6. ROM Address Generation for PCI Accesses

Bits	Description
A[29:28]	Controlled by the 21285 depending upon the ROM width (refer to Sections 4.2.2 and 4.2.3).
A[23:20]	If the corresponding bit of the expansion ROM base address mask register is a 0, the address bit is 0. If the corresponding bit of the expansion ROM base address mask register is a 1, use the PCI address.
A[19:6]	PCI address bits [19:6].
A[5]	If the PCI accesses the lower 32 bytes of ROM, this address is the inverse of PCI address bit [5]. Otherwise, it is equal to PCI address bit [5].
A[4:2]	PCI address bits [4:2].

The reason for the conditional inversion of bit 5 is that the SA-110 needs to read a vector from ROM address 0h, and the PCI requires the expansion ROM header to be at ROM address 0h. To accommodate both needs, ROM addresses 20h through 3Fh are swapped with 0h through 1Fh when addressed from the PCI. The SA-110 vector should be placed in ROM address 0h, and the PCI expansion ROM header should be placed in ROM address 20h through 3Fh, as seen from the SA-110.

4.2.2 Reads

During ROM reads, the 21285 asserts **rom_ce_l** and **A[30]**, which should be connected to the ROM output enable (**oe_l**) signal.

For reads from a byte-wide ROM, the 21285 performs the following:

- Sequences **A[29:28]** with values 3,2,1,0
- Latches and packs the ROM data read from **D[7:0]** internally

For reads from a word-wide ROM, the 21285 performs the following:

- Toggles **A[29]** from 1 to 0
- Latches and packs the ROM data read from **D[15:0]** internally

For reads from a Dword-wide ROM, the 21285 latches the data from **D[31:0]**. If the read is from the SA-110, the 21285 drives the packed data back onto **D[31:0]**. If the read is from the PCI, the data is sent back to the PCI.

For cache line fill reads from the SA-110, the 21285 reads eight Dwords from the ROM in wrapped order by sequencing **A[4:2]**.

4.2.3 Writes

The ROM's write mechanism is provided to allow reprogramming of Flash ROM. During ROM writes, the 21285 asserts **rom_ce_l** and **A[31]**, which should be connected to the ROM write enable (**we_l**) signal. The value for the low-order ROM address lines (driven on **A[29:28]**) is provided by the value written to the ROM write byte address register. Only one write is done to the ROM regardless of the ROM width and byte enables.

When writing to the ROM from the PCI, it is important for software to synchronize writes to the ROM byte address register with writes to the ROM itself. This is necessary because writes to the ROM are placed into the Inbound FIFO, while CSR writes are performed immediately. Since a PCI read from the ROM flushes the Inbound FIFO, synchronization can be achieved by using the following algorithm:

1. Read data from ROM address *x* (where *x* is any ROM address).
2. Write the byte address for address *y* to the ROM write byte address register.
3. Write data to ROM address *y*.
4. Repeat steps 1 through 3 for ROM address *y*+1.

4.2.4 Timing

Timing during ROM accesses can be controlled by values in the SA-110 control register. The ROM access time, burst time, and tristate time can be specified.

For SA-110 accesses to the ROM, the SA-110 is stalled until the access is complete. For PCI accesses to the ROM, address bus enable (**ABE**) and data bus enable (**DBE**) are deasserted. The SA-110 will be stalled if it attempts to start an external bus cycle while the PCI ROM access is in progress.

All ROM address, data, and control signals are controlled synchronously to **fclk_in**.

The ROM read timing is as follows:

- At the start of the cycle, the address is driven and **rom_ce_l** is asserted.
- After one **fclk_in** cycle, **rom_oe_l** is asserted (this signal is driven on **A[30]**).
- After a further ROM access time **fclk_in** cycle, the ROM data is latched, **rom_oe_l** negates, and the address changes.
- If another read is required (either to pack a Dword or to fulfill an SA-110 cache line fill), then the new address remains valid for ROM burst time **fclk_in** cycles and **rom_oe_l** is reasserted after one **fclk_in** cycle. This means that **rom_oe_l** is negated for the first **fclk_in** cycle of the access. This is a deliberate policy to provide compatibility with some designs of the ROM emulator.
- When the final read has been completed, there is a delay of ROM tristate time **fclk_in** cycles before another device is enabled onto the **D** bus. This feature allows ROMs with slow data turn-off times to be accommodated.

The ROM write timing is as follows:

- At the start of the cycle, the address and data are driven and **rom_ce_l** is asserted.
- After two **fclk_in** cycles, **rom_we_l** is asserted (this signal is driven on **A[31]**).
- After the address has been valid for a total of 1 + ROM access time **fclk_in** cycles, **rom_we_l** is negated.
- After a further **fclk_in** cycle, **rom_ce_l** negates and the address and data go invalid.
- The ROM tristate time delay is imposed after ROM writes, but serves no useful purpose.

4.2.5 Blank ROM Programming

The 21285 has a mode that allows programming of blank Flash ROMs in place on a circuit board. This mode is enabled if both **ma[6]** and **pci_cfn** are 0 when the 21285 is reset. When this mode is enabled:

- **nRESET** is asserted by the 21285 to keep the SA-110 in reset state. (This is necessary since there may be no code in the ROM yet.)
- The initialize complete bit [0] in the SA-110 control register is set internally by the 21285. This allows the 21285 to complete type 0 configuration accesses.
- The expansion ROM base address mask is reset to 00F00000h (this is the normal default) causing the expansion ROM base address to request 16MB. This is the largest size ROM address space.
- The SDRAM base address mask is reset to 00FC0000h causing the SDRAM base address to request 16MB. This guarantees that there will be a 16MB address region allocated in PCI address space, but not used by the device. Flash programming software can reallocate that space to the ROM when the BIOS has not allocated any address space to the ROM.
- The normal PCI configuration software running on the host processor can load both the expansion ROM base address register and the command register, allowing the 21285 to respond to PCI memory cycles. The host processor can then perform normal ROM writes.

SA-110 Operation 5

This chapter describes the operation of the following:

- SA-110 interface
- X-Bus interface
- Ordering and deadlock avoidance

5.1 SA-110 Control

This section provides descriptions for the following:

- Address map partitioning
- Byte enables
- SA-110 bus arbiter

5.1.1 Address Map Partitioning

Table 5-1 describes the address map partitioning the SA-110's 4GB address space.

Note: SDRAM and ROM address regions are the only regions that the SA-110 can mark as cacheable. All other regions must be noncacheable.

CSR space is larger than required to address all CSRs. Address bits [11:0] are used to select the CSR. Bits [19:12] are ignored.

Reserved space should not be used by SA-110 software. Accidental accesses to reserved space will *not* cause the 21285 bus interface to hang, however, because the reserved address can alias to another valid address, a write can change the state of a SDRAM, a CSR, and so on.

Both Single Address Cycles and Dual Address Cycles (DAC) are performed using PCI memory space. DAC are performed if the upper address bits contained in the SA–110 DAC Address register are not all zero.

Table 5-1. SA-110 4GB Address Mapping (Sheet 1 of 2)

Function	Start Address	End Address	Size
SDRAM	0000 0000h	0FFF FFFFh	256MB
Reserved	1000 0000h	3FFF FFFFh	—
SDRAM array 0 mode register	4000 0000h	4000 3FFFh	16KB
SDRAM array 1 mode register	4000 4000h	4000 7FFFh	16KB
SDRAM array 2 mode register	4000 8000h	4000 BFFFh	16KB
SDRAM array 3 mode register	4000 C000h	4000 FFFFh	16KB
X-Bus XCS0	4001 0000h	4001 0FFFh	4KB

Table 5-1. SA-110 4GB Address Mapping (Sheet 2 of 2)

Function	Start Address	End Address	Size
X-Bus XCS1	4001 1000h	4001 1FFFh	4KB
X-Bus XCS2	4001 2000h	4001 2FFFh	4KB
X-Bus no CS	4001 3000h	4001 3FFFh	4KB
Reserved	4001 4000h	40FF FFFFh	—
ROM	4100 0000h	41FF FFFFh	16MB
CSR space	4200 0000h	420F FFFFh	1MB
Reserved	4210 0000h	4FFF FFFFh	—
SA-110 cache flush	5000 0000h	50FF FFFFh	16MB
Reserved	5100 0000h	77FF FFFFh	—
Outbound write flush	7800 0000h	78FF FFFFh	16MB
PCI IACK/special space	7900 0000h	79FF FFFFh	16MB
PCI type 1 configuration	7A00 0000h	7AFF FFFFh	16MB
PCI type 0 configuration	7B00 0000h	7BFF FFFFh	16MB
PCI I/O space	7C00 0000h	7C00 FFFFh	64KB
Reserved	7C01 0000h	7FFF FFFFh	—
PCI memory space	8000 0000h	FFFF FFFFh	2GB

5.1.2 Byte Enables

During SA-110 bus cycles, the byte enables are driven on **A[1:0]** concatenated with **MAS[1:0]** (SA-110 must be in enhanced mode). The byte enables are used during SDRAM writes and PCI writes and reads, with the exception of memory read line and memory read multiple, which assert all PCI byte enables despite the state of the SA-110 byte enables.

5.1.3 SA-110 Bus Arbiter

The purpose of the SA-110 bus arbiter is to choose which of several operations to perform when more than one is possible at a given time.

The operations that use the SA-110 bus are as follows:

- Refresh SDRAM
- SA-110 originated operations
  - — Read SDRAM
  - — Write SDRAM
  - — Read ROM
  - — Write ROM
  - — Read CSR
  - — Write CSR
  - — Read PCI

  - Post write to PCI
  - Read X-Bus
  - Write X-Bus
- PCI operations (that come through the Inbound FIFO)
  - Read SDRAM (nonprefetch)
  - Read SDRAM (prefetch)
  - Write SDRAM
  - Read ROM
  - Write ROM
- DMA operations
  - Read SDRAM
  - Write SDRAM

The arbiter rules are as follows:

1. SDRAM refresh is always the highest priority.
2. The SA-110 is always the second highest priority, except for the cycle immediately following the completion of an SA-110 transaction, when it is the lowest priority. This rule gives the SA-110 high priority but prevents it from continually blocking lower priority operations.
3. When a DMA and Inbound FIFO operation are both outstanding, priority alternates between DMA and the Inbound FIFO.

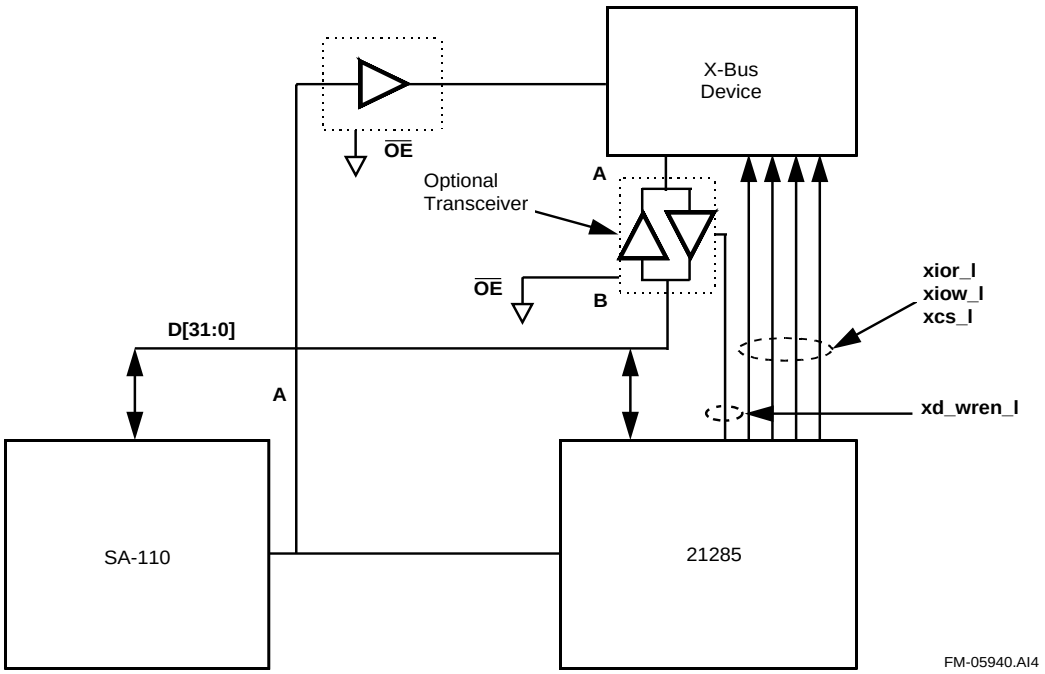
5.2 X-Bus Interface

The X-Bus interface allows low-performance 8-, 16-, and 32-bit peripherals with ISA-bus-like interfaces to be attached to the SA-110 side of the 21285. Typical devices that can be attached include:

- Super I/O controller
- UARTs
- Nonvolatile RAM
- Real-time clock
- I^2C
- IrDA (for wireless infrared communication)
- Input buffer (soft inputs), such as switches
- Output latch (soft outputs), such as LEDs

Figure 5-1 shows the X-Bus configuration. Only programmed I/O from the SA-110 is supported; the X-Bus does not support DMA or PCI accesses.

Figure 5-1. X-Bus Configuration



5.2.1 Address and Data Bus Generation

The X-Bus address and data buses are generated by buffering the address and data buses (**A** and **D**). Low voltage TTL buffers must be used for **D** since the SA-110 buses are not 5-V tolerant.

The X-Bus address bus buffer is unidirectional and is permanently enabled. The X-Bus data bus buffer is bidirectional and is permanently enabled. Its direction is controlled by the **xd_wren_l** pin from the 21285.

A standard 74LVT24S part can be used for the data transceiver. The behavior of **xd_wren_l** has been designed so that during a 21285 write, the transceiver's T/nR pin will be low. Therefore, the transceiver B port should connect to the SA-110/21285 and the A port should connect to the X-Bus devices. This also suits the use of the National Semiconductor LCX family, which has a 5-V tolerant A port and a 3.3-V tolerant B port.

The **xcs_l[2:0]** signals assert based on the SA-110 address space, and can be used as chip selects for three devices. More devices can be attached by using the no CS space and decoding the address externally. Signal **xior_l** asserts for a read and **xiow_l** asserts for a write.

5.2.2 Device Support

The X-Bus can support 8-bit, 16-bit, and 32-bit devices. An 8-bit peripheral device data bus is wired to the low-order byte lane of the buffered **D** bus. A 16-bit peripheral device data bus is wired to the two low-order byte lanes of the buffered **D** bus. The peripheral device address bus is wired so that its least-significant address line is wired to **A[2]** and so on. An 8-bit peripheral will be accessed at address offset 0, 4, 8, C, and so on, in X-Bus space. The X-Bus does not provide byte lane control.

5.2.3 Timing

Timing on the X-Bus is controlled by two CSRs: X-Bus cycle and X-Bus I/O strobe mask. The following two parameters control each of the X-Bus address spaces:

- Length—the total duration of the access.
- Strobe mask—a bit mask where bit 0 represents the state of the strobe for the first clock of the access, and more significant bits represent the state of the strobe for subsequent clock cycles. The strobe is asserted when the bit is set to 0 and negated when the bit is set to 1. Unused bits must be set to 1.

The X-Bus logic is clocked at the **fclk_in** frequency prescaled by 1, 2, 3, or 4. The prescaler is controlled by the X-Bus cycle register.

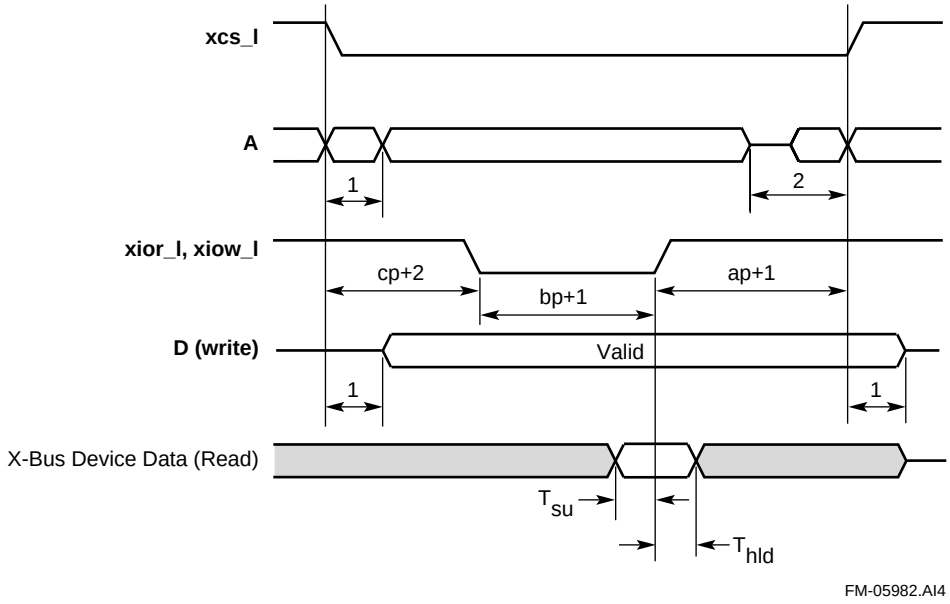
The strobe mask field is shifted out, cycle by cycle, to the **xior_l** or **xiow_l** strobe pin. By setting the length and strobe fields, the user can impose address setup, strobe duration, and address hold.

If the cycle length is set to *x* cycles, the *c* least significant bits of the strobe mask field are set, and the next *b* bits of the strobe mask are clear, then let $a = x - b - c$. For a prescaler of *p*, Figure 5-2 shows the cycle timing.

For example, if the cycle length is set to 7, the strobe mask field is set to F1h, and the prescaler is set to 2, then:

- $b = 3$
- $c = 1$
- $a = 7 - 3 - 1 = 3$
- The strobe will be asserted after four cycles and will remain asserted for seven cycles. The access will continue for an additional seven cycles.

Figure 5-2. X-Bus Timing



On reads, the X-Bus device drives read data to the SA-110, and the 21285 unstalls the SA-110 after the cycle count has expired.

On writes, the SA-110 drives write data to the X-Bus device, and the 21285 unstalls the SA-110 after the cycle count has expired.

5.3 Ordering and Deadlock Avoidance

This section provides a description of transaction ordering, ordering rules, and deadlock avoidance.

5.3.1 Transaction Ordering

Transaction ordering refers to the order, in time, in which posted writes and reads must be delivered to their destination relative to the way in which each was received from the source. These rules are necessary in order to comply with the PCI specification, and for software and bus mastering (DMA) devices to communicate properly.

Complying with the ordering rules is complicated by the presence of FIFOs for posted write data and prefetched read data. The following FIFOs are located on the 21285:

- Outbound FIFO
  - — SA-110 to PCI write data
  - — SDRAM to PCI DMA

- Inbound FIFO
  - — SA-110 read data and prefetch read data from PCI
  - — PCI to SDRAM/ROM write data
  - — PCI to SDRAM DMA
- PCI read FIFO
  - — PCI read data from SDRAM/ROM

5.3.2 Ordering Rules

The following ordering rules must be observed (refer to the rules in the *PCI Local Bus Specification, Revision 2.1*).

1. SA-110 posted writes to PCI must finish on PCI in the order in which they were accepted from SA-110.
2. PCI posted writes to SDRAM must finish in SDRAM in the order in which they were accepted from PCI.
3. An SA-110 read (either I/O, configuration, IACK, or memory) to PCI must force all SA-110 generated writes to PCI to finish before starting the read on PCI.
4. PCI reads from SDRAM must force all PCI writes to SDRAM to finish before starting the read.

Note: Rule 5 is *not* enforced in hardware because the read data is in a different FIFO than the write data. When required, software must write the outbound write flush address to enforce ordering.

5. PCI reads from SDRAM must force all SA-110 posted writes to PCI to finish on PCI before completing the read.
6. An SA-110 read (either I/O, configuration, IACK, or memory) to PCI must force all PCI-generated writes to SDRAM to finish before completing the read. Register reads and writes are not posted and, therefore, are not ordered with respect to PCI/SDRAM accesses.

5.3.3 Deadlock Avoidance

To avoid deadlocks, PCI posted writes to SDRAM and/or ROM must be able to complete while the SA-110 is stalled during an access to the PCI. This can happen for either of the following cases:

- An SA-110 read to PCI
- An SA-110 write to PCI if the Outbound FIFO is full

To break the deadlock, the 21285 must be able to write data to SDRAM and/or ROM while the SA-110 is stalled. To do so, the 21285 deasserts both the address and data bus enables (**ABE** and **DBE**) to the SA-110. Any PCI writes that get posted *after* the SA-110 access has started, as well as those that had been posted prior to it, may need to be done.

Functional Units 6

This chapter contains descriptions of the following functional units:

- PCI bus arbiter
- DMA channels
- I_2O message unit
- Timers
- Serial port

6.1 PCI Bus Arbiter

The 21285 contains a PCI bus arbiter that supports four external masters in addition to the 21285. In order to enable the arbiter, **ma[7]** must be a 0 at reset. The arbiter and X-Bus cannot both be used at the same time since they share I/O pins.

When the PCI arbiter is selected, the **xcs_l** pins are request inputs and the xcs direction bits in the SA-110 control register must not be written to a 1.

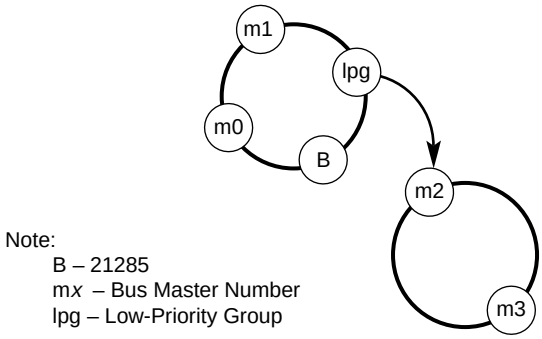
6.1.1 Priority Algorithm

The arbiter supports a programmable two-level rotating priority algorithm. Two groups of masters are assigned, a high-priority group and a low-priority group. The low-priority group as a whole represents one entry in the high-priority group; that is, if the high-priority group consists of *n* masters, then in at least every *n*+1 transactions, the highest priority is assigned to the low-priority group. Priority rotates evenly among the low-priority group. Therefore, members of the high-priority group can be serviced *n* transactions out of *n*+1, while one member of the low-priority group is serviced once every *n*+1 transactions.

Figure 6-1 shows an example of an arbiter where three masters, including the 21285, are in the high-priority group, and two masters are in the low-priority group. Using this example, if all requests are always asserted, the highest priority rotates among the masters in the following fashion (high-priority members are given in *italics*, low-priority members are given in **boldface** type):

B, m0, m1, **m2,** *B, m0, m1,* **m3,** *B m0, m1,* **m2,** *B, m0, m1*, **m3,** and so on.

Figure 6-1. Secondary Arbiter Example



Each bus master, including the 21285, can be configured to be in either the low-priority group or the high-priority group as determined by the value of the corresponding priority bit in the arbiter control register. Each master has a corresponding bit. If the bit is a 1, the master is assigned to the high-priority group; if the bit is a 0, the master is assigned to the low-priority group. If all the masters are assigned to one group, the algorithm defaults to a straight rotating priority among all the masters.

6.1.2 Determining Priority

Priorities are reevaluated every time **frame_l** is asserted, that is, at the start of each new transaction on the PCI bus. From this point until the time that the next transaction starts, the arbiter asserts the grant signal corresponding to the highest priority request that is asserted. If a grant for a particular request is asserted, and a higher priority request subsequently asserts, the arbiter deasserts the asserted grant signal and asserts the grant corresponding to the new higher priority request on the next PCI clock cycle. When priorities are reevaluated, the highest priority is assigned to the next highest priority master relative to the master that initiated the previous transaction. The master that initiated the last transaction now has the lowest priority in the group.

If the arbiter detects that an initiator has failed to assert **frame_l** after 16 cycles of both grant assertion and PCI bus idle condition, the arbiter deasserts the grant. That master does not receive any more grants until it deasserts its request for at least one PCI clock cycle.

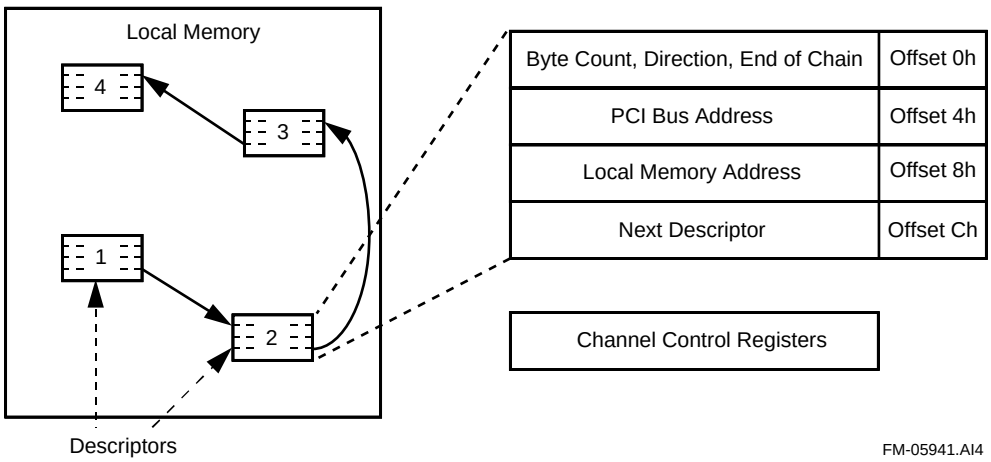
To prevent bus contention, if the PCI bus is idle, the arbiter never asserts one grant signal in the same PCI cycle in which it deasserts another. It deasserts one grant, and then asserts the next grant no earlier than one PCI clock cycle later. If the PCI bus is busy, that is, either **frame_l** or **irdy_l** is asserted, the arbiter can deassert one grant and assert another during the same PCI clock cycle.

6.2 DMA Channels

There are two DMA channels; each of which can move blocks of data from SDRAM to PCI or PCI to SDRAM. The DMA channels read parameters from a list of descriptors in memory, perform the data movement, and stop when the list is exhausted.

Figure 6-2 shows DMA descriptors in local memory. Each descriptor occupies four Dwords and must be naturally aligned. The channels read the descriptors from local memory into working registers.

Figure 6-2. DMA Descriptor Read



6.2.1 DMA Channel Operation

DMA channel operation is as follows:

1. The SA-110 sets up the descriptors in SDRAM. If there is only one operation to do, this step may be omitted. Each descriptor is composed of four Dwords that provide the following information:

- The number of bytes to be transferred and the direction of transfer.
- The PCI bus address of the transfer.
- The SDRAM address of the transfer.
- The address of the next descriptor in SDRAM or the DAC address.

  Bit 31 of the SDRAM address (third word of the descriptor) is called the D4 mode bit; it defines the information contained in the fourth Dword of the descriptor.

  If 0:

  — The fourth Dword of the descriptor is the local memory address of the next descriptor, assuming the end-of-chain bit is not set. If the end-of-chain is set, the fourth Dword of the descriptor is loaded into the Descriptor Register, which will not be used.

  — The Channel *n* DAC register is left unchanged when the descriptor is read.

  — Note that if the first descriptor in the chain does not specify the DAC address, the DAC register must be initialized by software if a nonzero value is required prior to setting the Channel Enable bit.

  If 1:

  — The fourth Dword of the descriptor is the value of the upper 32 bits of the PCI address for the current transfer (that is, pertaining to the descriptor currently being fetched from memory) and is loaded into the Channel *n* DAC register. Note that this is the case

regardless of the value of the end-of-chain bit, and that this value can be 0 if the PCI address of the transfer has 0 for its upper 32 bits.

— The address of the next descriptor (if any) is the address of the current descriptor plus 16.

This gives DMA control software the flexibility to provide PCI addresses anywhere in 64-bit PCI memory space, as well as the ability to locate descriptors anywhere in local memory, except for one limitation: when a descriptor has specified the upper 32-bit PCI address, the *next* descriptor must be located contiguous in memory (that is, an address that is 16 greater than the descriptor just fetched.

When the first descriptor is in the DMA channel registers, the channel DAC Address register must be initialized (if a nonzero value is required,) and the D4 mode bit (bit [31] of the SDRAM Address register) must be written to 0.

2. The SA-110 writes the address of the first descriptor into the DMA channel *n* descriptor pointer register. As an alternative, the SA-110 can write the values of the parameters of the first (or only) descriptor directly into the registers. If there is only one descriptor, the end of chain bit [31] in the DMA channel *n* byte count register must be set, and the value of the DMA channel *n* descriptor pointer register is a don't care; otherwise, it must be the address of the next descriptor.
3. The SA-110 writes the DMA channel *n* control register with other miscellaneous parameters, and sets the channel enable bit. If the descriptor was written into the registers in step 2, the channel initial descriptor in register bit [4] in the DMA channel *n* control register must also be set.
4. If the channel initial descriptor in register bit [4] is clear, the channel reads the descriptor block into the channel control, channel PCI address, channel SDRAM address, and channel descriptor pointer registers.
5. The channel transfers the data until the byte count is exhausted, and then sets the channel transfer done bit [2] in the DMA channel *n* control register.
6. If the end of chain bit [31] in the DMA channel *n* byte count register (which is in bit [31] of the first word of the descriptor) is clear, the channel reads the next descriptor and transfers the data. If it is set, the channel sets the chain done bit [7] in the DMA channel *n* control register and then stops.

The channel initial descriptor in register bit [4] in the DMA channel *n* control register is useful for nonchained transfers. It allows operation of the channel without the need to set up a list in SDRAM.

There is no restriction on byte alignment of the source address or the destination address. DMA reads are always unmasked reads (all byte enables asserted), either from SDRAM or PCI. For PCI-to-SDRAM transfers, the PCI command is memory read, memory read line, or memory read multiple according to the PCI read type field bits [6:5] in the DMA channel *n* control register. After each read, the byte count is decremented by the number of bytes read, and the source address is incremented.

After each write, the destination address is incremented by the number of bytes written. On the initial write, some low-order bytes can be masked based on the low two bits of the destination address as described in Table 6-1. After the first write, the destination address is incremented so that it is Dword aligned.

Table 6-1. DMA Channel Write

Initial Destination[a] Address [1:0]	Initial Byte Enable (Active High)
00	1111
01	1110
10	1100
11	1000

a. Destination is SDRAM for PCI-to-SDRAM transfers, and PCI for SDRAM-to-PCI transfers.

In Table 6-2, on the final write, some of the high-order bytes can be masked depending on the byte count and the initial destination address.

Table 6-2. Final Write

Final Byte Enable (Active High)				
Byte Count DIV 4 Remainder	Initial Destination Address[a] [1:0]			
	00	01	10	11
0	1111	0001	0011	0111
1	0001	0011	0111	1111
2	0011	0111	1111	0001
3	0111	1111	0001	0011

a. Destination is SDRAM for PCI-to-SDRAM transfers, and PCI for SDRAM-to-PCI transfers.

The 21285 may need to realign the data depending on the initial source and destination addresses. The data in each byte lane is rotated right or left by 0, 1, 2, or 3 byte lanes as described in Table 6-3. Data is packed into Dwords before being written to the destination.

Table 6-3. Aligning Byte Data

	Initial Destination[a] Address [1:0]			
Initial Source[b] Address [1:0]	00	01	10	11
00	0	Left 1	Left 2	Left 3
01	Right 1	0	Left 1	Left 2
10	Right 2	Right 1	0	Left 1
11	Right 3	Right 2	Right 1	0

a. Destination is SDRAM for PCI-to-SDRAM transfers, and PCI for SDRAM-to-PCI transfers.
b. Source means PCI for PCI-to-SDRAM transfers, and SDRAM for SDRAM-to-PCI transfers.

6.2.2 SDRAM-to-PCI Transfer

For a SDRAM-to-PCI transfer, the channel reads the SDRAM and places the data into the Outbound FIFO when the following conditions are met:

- There is enough free space in the FIFO (according to the SDRAM read length field bits [18:16] of the channel control register).
- The PCI interburst delay for the channel has elapsed.

The number of Dwords read from SDRAM is specified by the read length field bits [18:16] of the channel control register. At the beginning or end of a transfer, fewer Dwords may be read depending on alignment and byte count.

6.2.3 PCI-to-SDRAM Transfer

For a PCI-to-SDRAM transfer, the channel enqueues a read request to the PCI (in the Outbound FIFO) when the PCI interburst delay for the channel has elapsed.

The 21285 attempts to read the number of Dwords specified by the read length bit [15] of the channel control register (or the number of Dwords remaining in the transfer, if that is smaller). If the target disconnects before that number of Dwords has been read, the 21285 waits for the PCI interburst delay before starting another read. The 21285 also terminates the cycle prior to reading the requested number of Dwords if the Inbound FIFO fills or if the master latency timer expires. In all cases, all Dwords that were read are written into SDRAM.

6.2.4 Channel Alignment

The performance of the DMA channels varies depending on the alignment of the PCI address space relative to the SDRAM space. There are three cases:

- **Fully Aligned**—Bits [3:0] of the initial address in PCI space are equal to bits [3:0] of the initial address in SDRAM space. This means that the addresses are aligned to a 16-byte resolution, which is important because 16 bytes is the burst length of the SDRAMs (burst length of SDRAM is 4, and the **D** bus is 4 bytes wide).
- **Dword Aligned—**Bits [1:0] of the initial address in PCI space are equal to bits [1:0] of the initial address in SDRAM space. This means that the addresses are aligned to a 4-byte (Dword) resolution.
- **Unaligned**—Bits [1:0] of the initial address in PCI space are *not* equal to bits [1:0] of the initial address in SDRAM space.

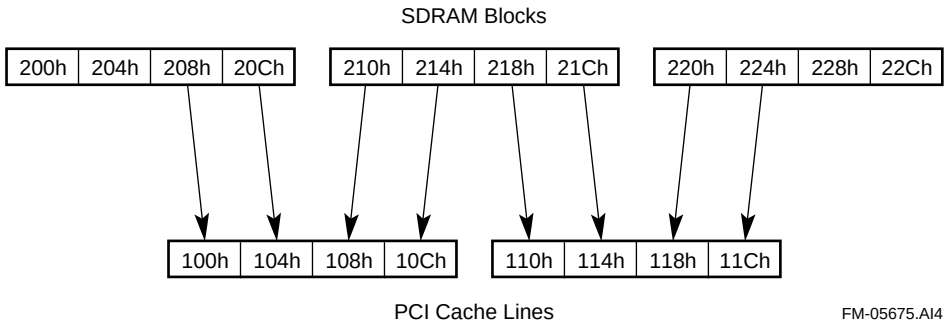
The alignment affects the way that the DMA channels handle the data.

SDRAM-to-PCI Transfers - For all alignment cases, the DMA channels align the PCI address to the value specified by SDRAM read length field bits [18:16] of the channel control register to maximize the use of the PCI memory write and invalidate command. For the first burst at the start of a transfer, the channel can write a partial PCI cache line (depending on the initial address in PCI space) to get the address aligned. After the first burst, subsequent bursts are aligned as stated in one of the three cases described at the beginning of this subsection.

For Dword aligned and unaligned cases, the DMA channels may need to read more data from SDRAM than is transferred to the PCI bus for each burst. Figure 6-3 shows an example of this case. The PCI address is 100h and the SDRAM address is 208h. To write eight Dwords at PCI addresses 100h through 11Ch (a cache line), the channel must read three blocks from SDRAM 200h through 22Ch.

The Dword data from SDRAM 208h transfers to PCI 100h, and so on. The first two and last two Dwords are discarded by the channel, but still consume SDRAM cycles. When the next burst of data is moved, the SDRAM block starting at 220h is read again, and the first two Dwords are discarded.

Figure 6-3. SDRAM-to-PCI Transfers



PCI-to-SDRAM Transfers - For all alignment cases, the DMA channels align the PCI address to the value specified by the PCI read length bit [15] of the channel control register. For the first burst at the start of a transfer, the channel may read a smaller number of Dwords (depending on the initial address in PCI space) to get the address aligned. After the first burst, subsequent bursts are aligned as stated in one of the three cases described at the beginning of this subsection.

For Dword aligned and unaligned cases, the DMA channels may need to write more SDRAM blocks (with appropriate masking via **dqm**), in a similar way to the SDRAM-to-PCI example shown in Figure 6-3.

6.3 I_2O Message Unit

This section describes the operation of the 21285 I_2O message unit. Sections 7.3.14 through 7.3.20 of this specification describe the I_2O registers.

The message unit provides a standardized message-passing mechanism between a host and a local processor (the SA-110 is the local processor). It provides a means for the host to read and write lists over the PCI bus at offsets of 40h and 44h from the first base address.

The message unit supports four logical FIFOs (see Figure 6-4):

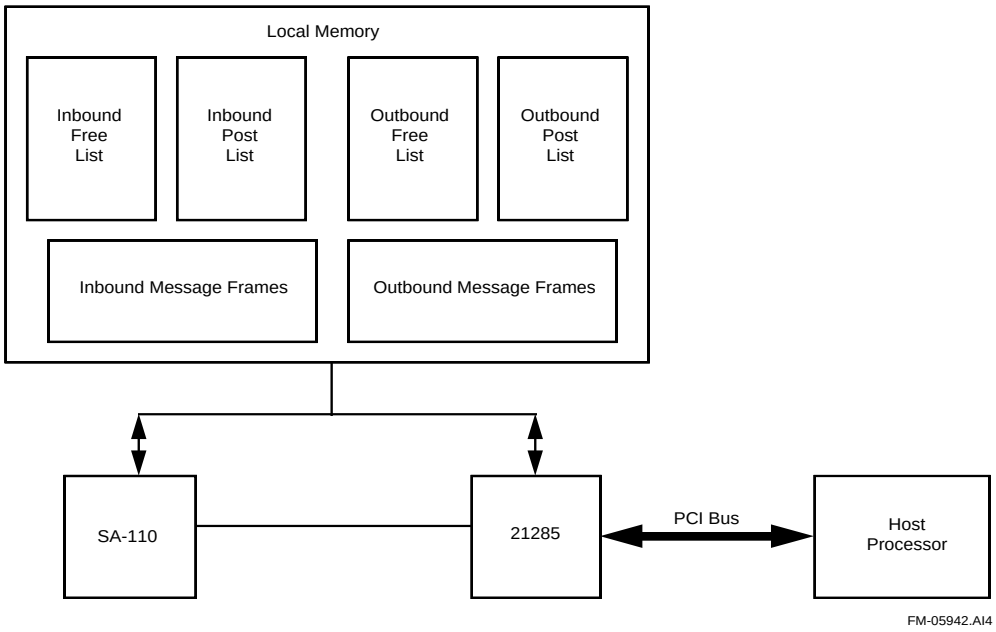
- Inbound free_list FIFO
- Inbound post_list FIFO
- Outbound free_list FIFO
- Outbound post_list FIFO

The FIFOs are used to hold message frame addresses (MFAs). The MFAs are offsets (pointers) to the message frames. The 21285 does not interpret the MFA values other than to recognize the special indicator for an invalid MFA (which is FFFFFFFFh), nor does it access the message frames.

The I_2O Inbound FIFOs are used to manage messages that are I/O requests from the host processor to SA-110. The I_2O Outbound FIFOs are used to manage messages that are replies from SA-110 to the host processor.

The FIFOs are stored in SA-110 SDRAM. Each FIFO has two pointers: a head pointer and a tail pointer. Table 6-4 lists the four pointers that are maintained in the 21285 hardware, and the four pointers that are maintained by SA-110 as variables in software.

Figure 6-4. I_2O Overview



Each FIFO is the same size as determined by the I_2O size field [12:10] in the SA-110 control register.

Table 6-4. FIFO Pointers

21285	**SA-110**
Inbound free_list head	Inbound free_list tail
Inbound post_list tail	Inbound post_list head
Outbound free_list tail	Outbound free_list head
Outbound post_list head	Outbound post_list tail

6.3.1 I_2O Inbound FIFO Operation

The I_2O Inbound FIFO operation is as follows:

Initialization

1. The I_2O inbound FIFOs are initialized by the SA-110. During operation, the host sends messages to the local processor (SA-110). The local processor operates on the messages.
2. The SA-110 allocates memory space for both the inbound free_list and inbound post_list FIFOs, and initializes the inbound pointers (both 21285 registers and software variables) with the address of the first MFA.
3. The SA-110 initializes the inbound free_list FIFO by writing valid MFA values to all entries. For each write to the inbound free_list FIFO, the SA-110 must also do a write to the inbound free_list count register to increment the number of entries.

Host posts an inbound message

1. When it needs to send a request message, the host processor removes an MFA from the head of the inbound free_list (via a read over the PCI bus to the 21285 register offset 40h).
2. The host processor writes the request message to the MFA in SA-110 memory (via writes over the PCI to SA-110 SDRAM).
3. The host processor places the MFA onto the tail of the inbound post_list (via a write over the PCI bus to offset 40h). The 21285 internally increments the inbound post_list count register, which interrupts SA-110 (if not masked by IRQEnable/FIQEnable). The write to offset 40h is ordered with respect to the PCI-to-SDRAM write.

SA-110 accepts inbound message

1. The SA-110 removes the MFA from the head of the inbound post_list. For each read of the inbound post_list, the SA-110 must also do a write to the inbound post_list count register to decrement the number of entries.
2. The SA-110 reads the request message from the MFA and performs the application-specific action based on the message.
3. The SA-110 writes the MFA to the tail of the inbound free_list so that the message frame can be reused at a future time. It also writes to the inbound free_list count register to increment the number of entries.
4. It is possible for the host to post more than one message before the SA-110 accepts any posted messages. The interrupt to the SA-110 remains asserted as long as there is at least one message posted.

6.3.2 I_2O Outbound FIFO Operation

The I_2O Outbound FIFO operation is as follows:

Initialization

1. The I_2O outbound FIFOs are initialized by the SA-110. During operation, the local processor (SA-110) sends messages to the host. The host operates on the messages.
2. The SA-110 allocates memory space for both the outbound free_list and outbound post_list FIFOs, and initializes the outbound pointers (both 21285 registers and software variables) with the address of the first MFA.

3. The host processor initializes the outbound free_list FIFO by writing valid MFAs to all entries.

SA-110 posts an outbound message

1. When it needs to send a reply message, the SA-110 removes an MFA from the head of the outbound free_list.
2. The SA-110 writes the reply message to the MFA in local memory (via SA-110 writes to local memory).
3. The SA-110 places the MFA onto the tail of the outbound post_list. The SA-110 must also do a write to the outbound post_list count register to increment the number of entries. The 21285 asserts **pci_irq_l** when the value in the outbound post_list count register is not zero (if not masked by outbound interrupt mask register).

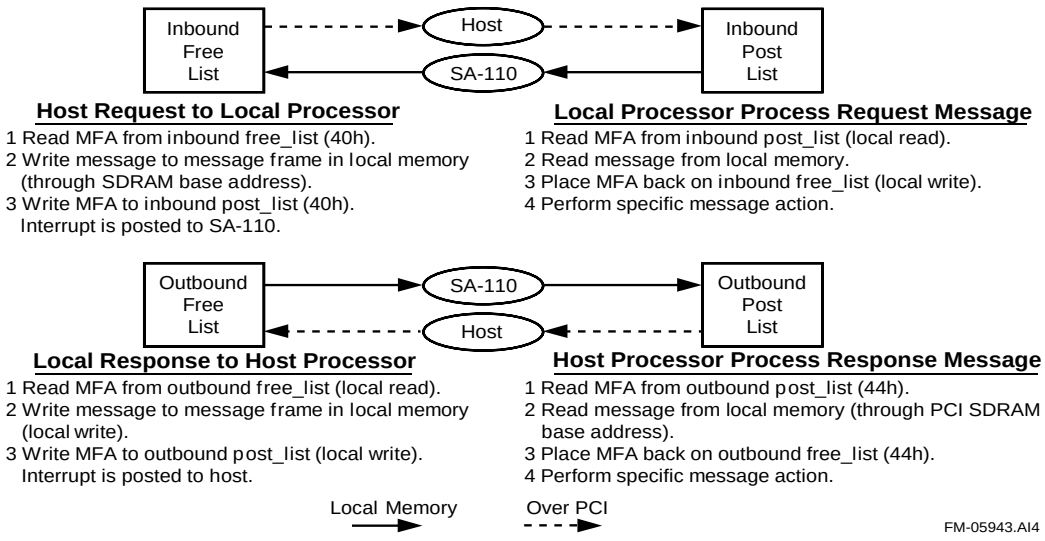
Host accepts outbound message

1. The host processor removes the MFA from the head of the outbound post_list (via a read over the PCI bus to offset 44h). The 21285 internally decrements the value in the outbound post_list count register.
2. The host processor reads the reply message from the MFA and performs the application-specific action based on the message.
3. The host processor writes the MFA to the tail of the outbound free_list (via a write over the PCI bus to offset 44h) so that the message frame may be reused at a future time.

6.3.3 Circulation of MFAs

Figure 6-5 shows the circulation of MFAs from free lists to post lists and back. Initially all inbound MFAs are on the inbound free_list and all outbound MFAs are on the outbound free_list.

Figure 6-5. Circulation of MFAs



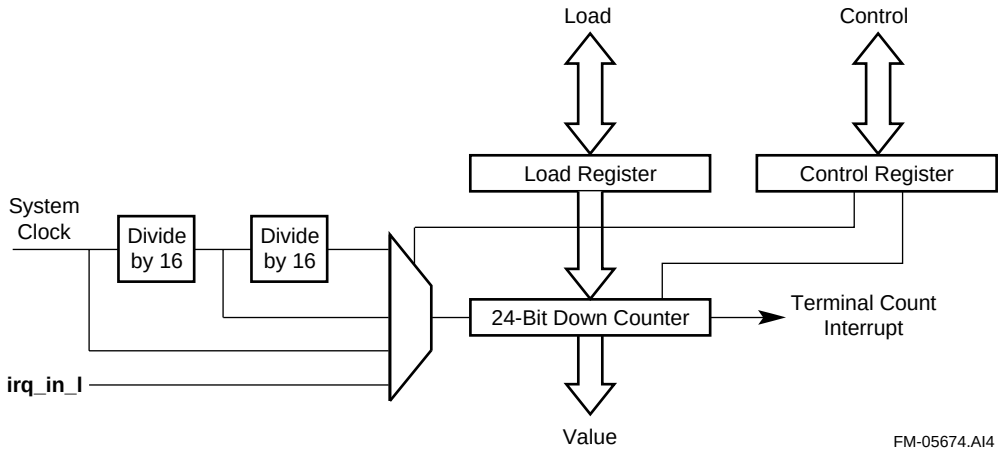
6.4 Timers

The 21285 contains four timers. Each timer is a 24-bit timer that can be preloaded and either free-run, or decremented to zero and then reloaded. Each timer is clocked in one of four ways:

- fclk_in
- **fclk_in** divided by 16
- **fclk_in** divided by 256
- External input

When a timer reaches zero it generates an interrupt. The interrupt can be enabled or disabled in the IRQEnable/FIQEnable registers. The interrupt remains asserted until cleared by a write (any data) to the associated TimerClear register. Figure 6-6 shows a block diagram of the timer function for each timer.

Figure 6-6. Timer Block Diagram



The timer count signal can be an external event connected on the **irq_in_l** pins. Signal **irq_in_l[0]** is connected to timer 1, **irq_in_l[1]** is connected to timer 2, and so on. The assertion edge is detected, synchronized, and used to advance the timer by one count. The edge-detector circuit is reenabled when the signal becomes deasserted.

6.4.1 Timer 4 Application

Timer 4 can be used as a watchdog timer if **pci_cfn** is asserted. If the watchdog enable bit [13] in the SA-110 control register is set, a reset sequence is initiated when timer 4 counts to zero. This reset sequence is as follows: the **nRESET** and **pci_rst_l** pins assert for several cycles, and then **nRESET** is deasserted (**pci_rst_l** is deasserted by SA-110 software clearing bit [9] in the SA-110 control register). When **pci_cfn** is asserted, **nRESET** is normally an input driven by the power-on reset circuitry on the board, but in this case, it must be driven by the 21285. So, to use the watchdog timer, the power-on circuitry must drive **nRESET** with an open-drain driver on the board.

System software can use the watchdog as follows. Set timer 4 for periodic interrupts and disable the interrupt on **nIRQ/nFIQ**. A periodic process (based on one of the other timers) would write to Timer4Load. If that process ever fails to write to Timer4Load within the countdown time, then both the SA-110 and the 21285 reset.

Once the watchdog enable bit is set, it can only be cleared by a chip reset.

6.5 Serial Port

The serial port is a general-purpose, full-duplex, universal asynchronous receivertransmitter (UART), which supports similar functionality to the 16C550 UART. It can operate at baud rates from approximately 225 bps to approximately 200 Kbps; the exact range is dependent upon the **fclk_in** frequency. It supports five to eight bits of data; odd, even, or no parity; one start bit; either one or two stop bits; and can transmit a continuous break signal.

The external pins dedicated to this interface are **tx** and **rx**. Modem control signals (RTS, CTS, DTR, and DSR) are not implemented.

The UART registers are only accessible to the SA-110; these registers cannot be accessed via the PCI bus.

6.5.1 Data Handling

A 16-entry, 8-bit FIFO is used to buffer outgoing data, and a 16-entry, 10-bit FIFO is used to buffer incoming data (two bits per entry are used to store framing and parity error flags for each character received). It is possible to provide single data buffering by disabling all FIFO entries but one.

6.5.2 Initialization

Following reset, the UART is disabled. Operation is initialized by the following:

1. Program the UART control register with the desired mode of operation.
2. Write the H_UBRLCR, L_UBRLCR, and M_UBRLCR registers setting up baud rate, parity, stop bits, word length, and enable FIFO.
3. Set the enable bit in the UARTCON register.

Once programmed, transmission and reception of data begins on the transmit (**tx**) and receive (**rx**) pins.

6.5.3 Frame Format

Nonreturn to zero (NRZ) encoding is used by the UART. Each data frame is between 7 and 12 bits long depending on the size of the data programmed, if parity is enabled, and if a second stop bit is enabled. The frame begins with a start bit that is represented by a high-to-low transition. Next, either 5, 6, 7, or 8 bits of data are transmitted beginning with the least significant bit. An optional parity bit follows, which is set if even parity is enabled and an even number of ones exist within the data byte, or if odd parity is enabled and the data byte contains an odd number of ones. The data frame ends with either one or two stop bits (selected within the control register) that is represented by one or two successive bit periods of a logic one.

6.5.4 Baud Rate Generation

The baud rate is derived by dividing down the 21285 clock (**fclk_in**). The signal **fclk_in** is divided by four and used as the reference clock. That clock is first divided by a programmable number between 1 and 1024, and then by a fixed value of 16. The receive/transmit baud clock is synchronized with the data stream each time a transition is detected on the receive data line. Receive data is sampled halfway through each bit period by counting 8 of the 16 clocks that are produced before the fixed divide by 16 takes place.

6.5.5 Receive Operation

The UART receives incoming data using a serial shifter; latches the frame; and strips it of its start, parity, and stop bits; and then places the data within the receive FIFO. If parity is enabled, the number of data bits (that are one) are counted, as data is extracted from each frame. Parity is then checked by comparing this value to the stripped parity bit. Either odd or even parity is used as specified by the programmer. If a parity error is detected, the parity error bit is set in the FIFO entry corresponding to the data. If a logic zero is detected by the receive logic where a stop bit (or bits) was expected, the framing error bit is set in the FIFO entry corresponding to the data. When the FIFO fills more than halfway, an interrupt is signaled. If the data is not removed soon enough, and the FIFO is completely filled, an overrun bit is set in RXSTAT if the receive logic attempts to place additional data within the FIFO. If the UART is disabled and a one-to-zero transition is detected (a start bit), the receiver status interrupt is signaled. (This dual-purpose interrupt is also signaled if the UART is enabled, the receive FIFO contains valid data, and a 32-bit period has elapsed without the reception of data on **rx**.)

6.5.6 Transmit Operation

The UART transmit logic operates at the same time as the receive logic (full-duplex). Data is taken from the transmit FIFO; start, parity, and stop bits are added to generate a frame; and the value is loaded into a serial shift register. The contents are shifted out onto the **tx** pin and clocked by the baud clock. When the transmit FIFO is emptied more than halfway, an interrupt is signaled. If new data is not supplied soon enough, and the FIFO is completely emptied, the transmit line is forced high (one) to indicate the idle state.

6.5.7 Serial Port Interrupts

Table 6-5 describes how serial port interrupts can be generated.

Table 6-5. UART Interrupts

Name	Source	Condition
RXINT	Rx interrupt	Asserted when the UART is disabled and a low level on the receive line has been detected (start bit). This interrupt is cleared when one of the events that generates it becomes false (when the UART becomes enabled or the receive line goes high). Also asserted when the receive FIFO is enabled and is more than half full, or when the receive FIFO is not empty and there is no data for more than a 32-bit period, or when the receive FIFO is disabled and data is received. The Rx interrupt is cleared when the receive FIFO is less than half full or the holding register is empty.
TXINT	Tx interrupt	Asserted when the transmit FIFO is less than half full, or when the transmit FIFO is disabled and the holding buffer is empty. The Tx interrupt is cleared when the transmit FIFO is more than half full or the holding register is full.

6.6 UART Register Definitions

This section describes the seven UART registers.

6.6.1 UARTDR—Offset 160h

The UART data register (UDR) corresponds to the top-most entry of both the transmit and receive FIFOs. When the UDR is read, the top-most entry of the receive FIFO is accessed.

Dword Bit	Name	R/W	Description
7:0	Data	R/W	Bits [7:0] contain FIFO data. Error data associated with the received character is available in RXSTAT. After the read, the data in the next location of the receive FIFO is automatically transferred up to the UDR. When the UDR is written, the top-most FIFO entry of the 8-bit transmit FIFO is accessed. Write data is placed in the FIFO. After a write, the data is automatically transferred down to the next location of the transmit FIFO. For data sizes other than eight bits, the upper bits of this field are zero extended.
31:8	—	R	Read only as 0.

Note: The received data must be read first (UARTDR) followed by the status error associated with the data (RXSTAT). This read sequence cannot be reversed.

6.6.2 RXSTAT—Offset 164h

Reading from the RXSTAT provides the error status associated with the data received on UARTDR. Flags in this register indicate error conditions, such as overrun, framing, and parity errors, which occurred during the unpacking of a received frame. Each entry in the receive FIFO contains two error bits that correspond to the data stored within the same FIFO entry. The parity error bit is set when parity is enabled (PE = 1) and the parity type programmed using OES does not correspond to the parity check of the incoming serial data stream that is calculated by the receive logic. The parity error bit is set when the expected parity does not match the received parity. The framing error bit is set when the stop bit, within a frame of incoming serial data, is a zero instead of a one.

Dword Bit	Name	R/W	Description
0	Frame error	R	This bit is set if a framing error (FE) occurred.
1	Parity error	R	This bit is set if a parity error (PER) occurred.
2	Overrun error	R	The overrun error (ORE) bit is set if more data is received by the UART when the FIFO is full. (It is cleared by reading the UARTDR register.)
31:3	—	R	Read only as 0.

Note: The received data must be read first (UARTDR) followed by the status error associated with the data (RXSTAT). This read sequence cannot be reversed.

6.6.3 H_UBRLCR—Offset 168h

Writing to this register sets the bit rate and mode for the UART.

Dword Bit	Name	R/W	Description
0	Break	R/W	When the break (BRK) bit is set, the UART first completes generation and transmission of the frame currently being processed, stops fetching data from the transmit FIFO, and forces the transmit pin low. The transmit pin remains low until the BRK bit is cleared or a reset occurs. The break signal does not affect the receive portion of the FIFO, thus normal operation on the receive line continues during the signaling of a break.
1	Parity enable	R/W	The parity enable (PE) bit is used to enable or disable parity checking by the receive data logic. Parity generation by the transmit logic is not affected by the PE bit. When parity is enabled (PE = 1), the odd/even parity select (OES) control bit is decoded to determine which type of parity should be checked, and each piece of data placed within the receive FIFO is checked. If the parity type programmed in the OES bit does not match the parity of the data received, the data is tagged by setting bit [8] of the FIFO location corresponding to the data that provides the parity error.
2	Odd/even select	R/W	The odd/even parity select (OES) bit is used to select whether odd or even parity should be used by the transmit and receive logic. When the OES is 1, even parity is selected; when the OES is 0, odd parity is selected. The MSB in each frame is used as the parity bit. (The transmit logic generates a parity bit by counting the number of ones within the data to be transmitted, and sets the parity bit if the type of parity selected matches the parity of the data. The receive data logic strips the parity bit and counts the number of ones in the received data. If the parity type of the data does not match the parity selected by OES, the parity error status flag is set within the status register, as well as bit [8] of the FIFO location corresponding to the data that produced the parity error.)
3	Stop bit select	R/W	The stop bit select (SBS) bit selects whether one or two stop bits should be used in transmission. When SBS = 0, one stop bit is inserted in the transmit frame for each character, and the receive data logic looks for and strips one stop bit per character.
4	Enable FIFO	R/W	The enable FIFO (EF) bit is used to select whether the whole FIFO should be used, or whether just the top-most entry should be used to buffer both transmit and receive data. When EF = 1, all 16 entries in both the transmit and the receive FIFO are used. When EF = 0, only the top-most entries are used. The programming of this bit also affects generation of the RXINT and TXINT interrupts. When only the top-most entry is enabled, a service request is generated each time a frame is received or transmitted. If the FIFO is filled halfway, 8 of 16 entries contain valid data.
6:5	Data size select	R/W	Specifies the UART data length as follows: 11 = 8 bits 10 = 7 bits 01 = 6 bits 00 = 5 bits When this field is programmed to be less than eight bits, the data is right justified in the FIFO, and the unused bits are zero filled.
31:7	—	R	Read only as 0.

6.6.4 M_UBRLCR—Offset 16Ch

The 12-bit baud rate divisor (BRD) field (top four bits of M_UBRLCR and lower eight bits from L_UBRLCR) is used to select the baud rate of the UART. A total of 4096 different baud rates can be selected, ranging from a minimum of approximately 225 bps (depending on the clock frequency) to a maximum of 200 Kbps. Refer to Section 6.5.4 for a description of baud rate generation. A desired baud rate, given a specific BRD value, or the required BRD value, given a desired baud rate, can be calculated using the following two respective equations, where BRD is the decimal equivalent of the binary value programmed into the bit field:

Note: baud_clk = **fclk_in**/4.

Baud rate = baud_clk / 16 x (BRD + 1)

BRD = **(**baud_clk / 16 x baud rate) - 1

Dword Bit	Name	R/W	Description
3:0	High baud rate divisor	R/W	Writing to these bits set the top four bits of the 12-bit baud rate for the UART.
31:4	—	R	Read only as 0.

6.6.5 L_UBRLCR—Offset 170h

The 12-bit baud rate divisor (BRD) field (top four bits of M_UBRLCR and lower eight bits from L_UBRLCR) is used to select the baud or bit rate of the UART. (For more information on M_UBRLCR and for a complete description of the BRD, see Section 6.6.4).

Dword Bit	Name	R/W	Description
7:0	Low baud rate divisor	R/W	Writing to these bits set the bottom eight bits of the 12-bit baud rate for the UART.
31:8	—	R	Read only as 0.

Note: Internally to the UART, H_UBRLCR, M_UBRLCR, and L_UBRLCR forms a single 19-bit register (UBRLCR), which is updated on a single write strobe generated by an H_UBRLCR write. In order to internally update the contents of M_UBRLCR or L_UBRLCR, an H_UBRLCR write must always be performed at the end.

The three registers must be updated with the following sequence:

- L_UBRLCR write, M_UBRLCR write, and H_UBRLCR write UARTCON—Offset 174h

6.6.6 UARTCON-Offset 174h

This register controls the encoding and protocols.

The UE bit is the only control bit that is reset to a known state to ensure that the UART is disabled following a reset. The reset state of all other control bits is unknown and must be initialized before enabling the UART.

Dword Bit	Name	R/W	Description
0	UART enable	R/W	The UART enable (UE) bit is used to enable and disable all UART operation. When UE = 1, the UART is enabled for serial transmission. It is required that the user first program all other control bits before setting UE. If the UE bit is cleared to zero while the UART is actively transmitting data, transmission is permitted to complete on the current byte of data that is being processed by the receiver or transmitter, and the UART is disabled, keeping all data in the FIFOs intact. UE is the only bit within the UART that is reset to a known state.
1	S	R/W	The SIREN HP SIR protocol enable bit. This bit has no effect if the UART is not enabled.
2	I	R/W	The RTXM IrDa Tx mode bit controls the IrDa encoding strategy. Clearing this bit means that each zero bit transmitted is represented as a pulse of width 3/16th of the bit rate period. Setting this bit means that each zero bit transmitted is represented as a pulse of width 3/16th of the 115000 bps (1.6 µs), regardless of the selected bit rate. Setting this bit uses less power but may reduce the distance for good quality transmissions.
31:3	—	R	Read only as 0.

6.6.7 UARTFLG—Offset 178h

The UARTFLG register provides the status of the FIFO.

Dword Bit	Name	R/W	Description
2:0	Reserved	—	Read as 0.
3	Transmitter busy	R	The transmitter busy flag (TBY) is a read-only bit that is set when the transmitter is actively processing data for transmission, and is cleared when the transmitter is idle or the UART is disabled (UE=0).
4	Receive FIFO status	R	When 1: No characters available. When 0: One or more characters available.
5	Transmit FIFO status	R	When 1: Busy. When 0: Ready to accept a character.
31:6	—	R	Read only as 0.

Registers 7

This chapter describes the following categories of registers:

- PCI configuration space registers
- PCI control and status registers
- SA-110 control and status registers
  - Interrupt controller registers
  - Timer control registers
  - DMA control registers
  - I_2O control registers
  - Miscellaneous registers

7.1 PCI Configuration Space Registers

PCI configuration space registers conform to the *PCI Local Bus Specification, Revision 2.1*. These registers are accessed from the PCI by configuration reads and configuration writes, and are byte writable. They are also accessible from the SA-110 (offset is from address 4200 0000h). Table 7-1 shows the allowable access to each register.

Table 7-1. Register Access

Abbreviation	Definition
R	Read only. Writes have no effect.
R/W	Read/write.
W1C	Read. Write 1 to clear.
W0C	Write zero to clear.
W1S	Read. Write 1 to set.
WO	Write only.

Table 7-2 lists the PCI configuration mapping.

Table 7-2. PCI Configuration Mapping

31 24	23 16	15 8	7 0	Offset
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Latency Timer	Cache Line Size	0Ch
CSR Memory Base Address				10h
CSR I/O Base Address				14h
SDRAM Base Address				18h
Reserved (Unused Base Address)				1Ch
Reserved (Unused Base Address)				20h
Reserved (Unused Base Address)				24h
CardBus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Cap_Ptr	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch
Reserved				4Ch – 69h
PM[a] Capabilities		PM Capability Identifier		70h
Data	Reserved	PM Control and Status		74h

a. Power Management

7.1.1 Vendor ID Register—Offset 00h

Dword Bit	Name	R/W	Description
15:0	Vendor ID	R	Identifies Intel Corporation as the vendor of this device. Internally hardwired to be 1011h.

7.1.2 Device ID Register—Offset 02h

Dword Bit	Name	R/W	Description
31:16	Device ID	R	Identifies this device as the 21285. Internally hardwired to be 1065h.

7.1.3 Command Register—Offset 04h

Dword Bit	Name	R/W	Description (Sheet 1 of 2)
0	I/O space enable	R/W	When 0: The 21285 does not respond to PCI I/O transactions as a target. When 1: The 21285 response to I/O transactions for CSR accesses when the address matches the CSR I/O base address register. Reset value: 0.
1	Memory space enable	R/W	When 0: The 21285 does not respond to PCI memory transactions as a target. When 1: The 21285 response to memory transactions for CSR accesses when the address matches the CSR memory base address register, the SDRAM base address, or the configuration ROM base address. Reset value: 0.
2	Master enable	R/W	When 0: The 21285 does not become a PCI bus master. When 1: The 21285 becomes a PCI bus master in response to SA-110 originated cycles to the PCI or DMA channel accesses. This bit does not affect the 21285 placing SA-110 or DMA originated addresses and/or data into the Outbound FIFO. However, if this bit is 0, address and data remain in the FIFO. Reset value: 0.
3	Special cycle enable	R	The 21285 ignores special cycle transactions as target, so this bit is read only and returns 0.
4	Memory write and invalidate enable	R/W	When 0: The 21285 never uses the memory write and invalidate as master. When 1: The 21285 uses memory write and invalidate if other conditions are met (see Section 3.3.2). Reset value: 0.
5	VGA palette snoop enable	R	Reads as 0 to indicate that the 21285 never snoops VGA palette writes.
6	Parity error response	R/W	Controls the 21285's response when a parity error is detected on the primary interface. When 0: The 21285 does not assert **perr_l** or **serr_l** in response to data and address parity errors, respectively. When 1: The 21285 asserts **perr_l** or **serr_l** (if enabled) in response to parity errors. The 21285 sets status register bit [31] when a parity error is detected regardless of the state of this bit. Reset value: 0.

Dword Bit	Name	R/W	Description (Sheet 2 of 2)
7	Wait cycle control	R	Reads as 0 to indicate that the 21285 does not perform address or data stepping.
8	SERR# enable	R/W	Controls the enable for **serr_l**. When 0: Signal **serr_l** cannot be driven by the 21285. When 1: Signal **serr_l** can be driven low by the 21285 under the conditions described in Section 2.1. Reset value: 0.
9	Fast back-to-back enable	R/W	When 0: The 21285 does not generate fast back-to-back transactions as the master. When 1: The 21285 is enabled to generate fast back-to-back transactions. Reset value: 0.
15:10	Reserved	R	Reserved. Returns 0 when read.

7.1.4 Status Register—Offset 06h

Dword Bit	Name	R/W	Description
19:16	Reserved	R	Reserved. Returns 0 when read.
20	Reserved	R	Reserved. Returns 1 when read. Indicates that the 21285 parts with a REV_ID of 4 or greater supports PCI management.
21	66-MHz capable	R	Reads as 0 to indicate that the 21285 is not capable of 66-MHz operation.
22	UDF supported	R	Reads as 0 to indicate that the 21285 does not support user-defined features.
23	Fast back-to-back capable	R	Reads as 1 to indicate that the 21285 is capable of accepting fast back-to-back transactions as a target.
24	Data parity error detected	W1C	This bit is set to 1 when the command register parity error response bit [6] is set and either of the following are true: • The 21285 is master of a PCI write and **perr_l** is asserted by the target. • The 21285 is master of a PCI read and asserts **perr_l** to indicate that the read data had bad parity. Reset value: 0.
26:25	DEVSEL#timing	R	Reads as 01 to indicate that the 21285 asserts **devsel_l** with medium timing.
27	Signaled target abort	R	Reads as 0 to indicate that the 21285 never signals target abort as a target.
28	Received target abort	W1C	This bit is set to 1 when the 21285 is the master of a transaction that terminates with a target abort. Reset value: 0.
29	Received master abort	W1C	This bit is set to 1 when the 21285 is the master of a transaction (except for special cycles) that terminates with a master abort. Reset value: 0.
30	Signaled system error	W1C	This bit is set to 1 when the 21285 has asserted **serr_l**. This occurs either when the SA-110 control register assert SERR bit is set by the SA-110, or in response to a PCI address parity error. Reset value: 0.
31	Detected parity error	W1C	This bit is set to 1 when the 21285 detects one of the following conditions: • Address parity error • Incorrect write data parity when the 21285 is the target of a write • Incorrect read data parity when the 21285 is the master of a read Reset value: 0.

7.1.5 Revision ID Register—Offset 08h

Dword Bit	Name	R/W	Description
7:0	Revision ID	R	Indicates the revision number of this device. The initial revision reads as 0. Subsequent revisions increment by 1.

7.1.6 Class Code Register—Offset 0Ah

Dword Bit	Name	R/W	Description
31:8	Class code	R	Reads as 0B4001h if **ma[3]** = 1 at reset. Reads as 0E0001h if **ma[3]** = 0 at reset.

7.1.7 Cache Line Size Register—Offset 0Ch

Dword Bit	Name	R/W	Description
7:0	Cache line size	R/W	Indicates the number of Dwords in the host processor's cache line. Legal values are 4, 8, or 16. Other values are treated as 8. Used during memory writes initiated by the 21285 to determine whether to use the memory write or memory write and invalidate command. Reset value: 0.

7.1.8 Latency Timer Register—Offset 0Dh

Dword Bit	Name	R/W	Description
15:8	Latency timer	R/W	Indicates the value of the latency timer, which limits the length of a burst that the 21285 performs as master when its bus grant is removed. Reset value: 0.

7.1.9 Header Type Register—Offset 0Eh

Dword Bit	Name	R/W	Description
23:16	Header type	R	Reads as 0 indicating a header type of zero.

7.1.10 BIST Register—Offset 0Fh

Dword Bit	Name	R/W	Description
31:24	BIST	R/W	The SA-110 can write to this field to indicate to a host that it provides BIST. The 21285 does not interpret or set this field, however, bit [30] (bit [6] of the BIST field that the host processor uses to invoke BIST) can be enabled to interrupt the SA-110. Reset value: 0.

7.1.11 CSR Memory Base Address Register—Offset 10h

Dword Bit	Name	R/W	Description
6:0	Memory address space	R	Reads as 0 or 8 as determined by bit [18] of the CSR base address mask register.
17:7	CSR base address	R/W	Read/write or read-only 0 as determined by bit [18] of the CSR base address mask register.
27:18	CSR base address	R/W	Read/write or read-only 0 as determined by the corresponding bit of the CSR base address mask register.
31:28	CSR base address	R/W	Contains the base address of the CSRs. Reset value: 0.

The read/write capability of the CSR base address register is controlled by the CSR base address mask register with the following values:

- A 1 in the mask register causes the corresponding bit in the base address register to act as a read-only bit.
- A 0 in the mask register causes the corresponding bit in the base address register to act as a read/write bit.

Using this mechanism, the SA-110 can program the mask register in such a way as to have the base address register indicate to the configuration software the amount of PCI address space required. The mask takes precedence over the base address, that is, if the mask is a 1, then the corresponding bit position of the address is not compared in determining if a PCI access hits the base address register.

The address selects either CSRs or SDRAM as listed in the following table.

Address Value	Region Selected
0—7Ch	CSR
80—FFCh	Read-only 0
1000—FFFFFFFCh	SDRAM

For more information about the CSR base address mask register, see Section 7.3.7.

7.1.12 CSR I/O Base Address Register—Offset 14h

Dword Bit	Name	R/W	Description
6:0	CSR address space	R	Reads as 1 to indicate that the CSRs require 128 bytes of I/O address space.
31:7	CSR base address	R/W	Contains the base address of the CSRs. Reset value: 0.

7.1.13 SDRAM Base Address Register—Offset 18h

The read/write capability of the SDRAM base address register is controlled by the SDRAM base address mask register with the following values. For more information about the SDRAM base address mask register, see Section 7.3.9.

- A 1 in the mask register causes the corresponding bit in the base address register to act as a read-only bit.
- A 0 in the mask register causes the corresponding bit in the base address register to act as a read/write bit.

Using this mechanism, the SA-110 can program the mask register in such a way as to have the base address register indicate, to configuration software running on a host processor, the amount of PCI address space required. The mask takes precedence over the base address, that is, if the mask is a 1, then the corresponding bit position of the address is not compared in determining if a PCI access hits the base address register.

Dword Bit	Name	R/W	Description
3:0	Memory space indicator type and prefetchable	R	If bit [31] of the SDRAM base address mask register is 0, this field is read as 8, indicating that the SDRAM must be mapped to PCI memory space, may be located anywhere in 32-bit PCI address space, and is prefetchable. If bit [31] of the SDRAM base address mask register is 1, this field reads as 0.
17:4	—	R	Read only as 0.
27:18	SDRAM base address, Lower	R/W	Read/write or read-only 0 as determined by the corresponding bit in the SDRAM base address mask register. Reset value: 0.
31:28	SDRAM base address, Upper	R/W	Read/write or read-only 0 as determined by bit [31] in the SDRAM base address mask register. Reset value: 0.

7.1.14 CardBus CIS Pointer Register—Offset 28h

The CardBus CIS pointer register is not used. It is read as 0.

7.1.15 Expansion ROM Base Address Register—Offset 30h

The read/write capability of the expansion ROM base address register is controlled by the expansion ROM base address mask registers with the following values.

- A 1 in the mask register causes the corresponding bit in the base address register to act as a read-only bit.
- A 0 in the mask register causes the corresponding bit in the base address register to act as a read/write bit.

Using this mechanism, the SA-110 can program the mask register in such a way as to have the base address register indicate, to configuration software running on a host processor, the amount of PCI address space required. The mask takes precedence over the base address, that is, if the mask is a 1, then the corresponding bit position of the address is not compared in determining if a PCI access hits the base address register.

Dword Bit	Name	R/W	Description
0	Expansion ROM address decode enable	R/W	Read/write or read-only 0 as determined by bit [31] in the expansion ROM base address mask register. Reset value: 0.
19:1	—	R	Read only as 0.
23:20	Expansion ROM base address, Lower	R/W	Read/write or read-only 0 as determined by the corresponding bit in the expansion ROM base address mask register. Reset value: 0.
31:24	Expansion ROM base address, Upper	R/W	Read/write or read-only 0 as determined by bit [31] in the expansion ROM base address mask register. Reset value: 0.

7.1.16 Capabilities Pointer—Offset 34h

Dword Bit	Name	R/W	Description
7:0	Capabilities Pointer	R	Indicates the offset in configuration space of the first new capability; in this case, Power Management. Value: 70h.
31:8		R	Reset value: 0.

7.1.17 Subsystem Vendor ID Register—Offset 2Ch

Dword Bit	Name	R/W	Description
15:0	Subsystem vendor ID	R/W	The 21285 does not interpret or set this field. Reset value: 0.

7.1.18 Subsystem ID Register—Offset 2Eh

Dword Bit	Name	R/W	Description
31:16	Subsystem ID	R/W	The 21285 does not interpret or set this field. Reset value: 0.

7.1.19 Interrupt Line Register—Offset 3Ch

Dword Bit	Name	R/W	Description
7:0	Interrupt line	R/W	The 21285 does not interpret or set this field. Reset value: 0.

7.1.20 Interrupt Pin Register—Offset 3Dh

Dword Bit	Name	R/W	Description
11:8	Interrupt pin	R/W from SA-110, R from PCI	The SA-110 sets this field according to the routing of **pci_irq_l**. Usually, **pci_irq_l** is routed to INTA# and this field is programmed with a value of 1. The 21285 does not interpret or set this field. Reset value: 0.
15:12	—	R	Read only as 0.

7.1.21 Min_Gnt Register—Offset 3Eh

Dword Bit	Name	R/W	Description
23:16	Min_Gnt	R/W from SA-110, R from PCI	The 21285 does not interpret or set this field. It can be written by the SA-110 software to inform the system BIOS of subsystem grant requirements. Reset value: 0.

7.1.22 Max_Lat Register—Offset 3Fh

Dword Bit	Name	R/W	Description
31:24	Max_Lat	R/W from SA-110, R from PCI	The 21285 does not interpret or set this field. It can be written by the SA-110 software to inform the system BIOS of subsystem latency requirements. Reset value: 0.

7.1.23 Capability Identifier Register—Offset 70h

Dword Bit	Name	R/W	Description
7:0	Cap_ID	R	Value of 01h identifies the capability of PCI Power Management.
15:8	Next Item Ptr	R	Value of 0 indicates there is no next item.

7.1.24 Power Management Capabilities (PMC) Register—Offset 72h

The PMC register is used to indicate to system software which power management features are supported. In the 21285, this register is Read/Write from the SA-110 and Read-Only from the PCI. This enables the SA-110 firmware to determine exactly what features of the *PCI Power Management Interface Specification* are supported. The field names from this specification are listed here for reference; however, the 21285 does not interpret or use these bits for any hardware-specific function, other than register reads and writes.

This register is cleared at reset

Dword Bit	R/W	Description (Sheet 1 of 2)
15	R/W from SA-110 R from PCI	PME# can be asserted from D0.
14	R/W from SA-110 R from PCI	PME# can be asserted from D1.
13	R/W from SA-110 R from PCI	PME# can be asserted from D2.
12	R/W from SA-110 R from PCI	PME# can be asserted from $D3_{hot}$.
11	R/W from SA-110 R from PCI	PME# can be asserted from $D3_{cold}$.
10	R/W from SA-110 R from PCI	D2 Power Management state is supported.
9	R/W from SA-110 R from PCI	D1 Power Management state is supported.

Dword Bit	R/W	Description (Sheet 2 of 2)
8:6	R 0	
5	R/W from SA-110 R from PCI	Device-specific initialization (DSI) is required following the transition to D0 state.
4	R/W from SA-110 R from PCI	Auxiliary Power Source. Support for PME# in $D3_{cold}$ state requires auxiliary power supplied by the system by way of proprietary delivery vehicle.
3	R/W from SA-110 R from PCI	PME Clock. Indicates that the function relies on the presence of the PCI clock for PME# operation.
2:0	R/W from SA-110 R from PCI	Version. Indicates the version of the *PCI Bus Power Management Interface Specification* with which the function complies.

7.1.25 Power Management Control/Status (PMCSR) Register—Offset 74h

This register is used by the system software to manage and monitor the PCI function's power management state.

Any write to this register sets a status bit that can be enabled in IRQ/FIQEnable to interrupt the SA-110. A write by the SA-110 to this register clears this status bit.

The field names from the *PCI Power Management Interface Specification* are listed here for reference; however, only the Power State field, **PME_Status**, and **PME_En** are used by the 21285 (other than for register reads and writes).

This register is cleared at reset.

Dword Bit	R/W	Description
15	R/W from SA-110 R from PCI	**PME_Status**. Indicates that the device would assert PME# if **PME_En** is set. If this bit and **PME_En** in this register are both set, the 21285 will assert **Pre_PME_l**.
14:13	R/W from SA-110 R from PCI	Data Scale. Scaling factor used when interpreting the value of the data register.
12:9	R/W	Data Select. Selects the value to be reported through the data register.
8	R/W	**PME_En**. Enables the function to assert PME#. If this bit and PME_Status in this register are both set, the 21285 will assert **Pre_PME_l**.
7:2	R 0	
1:0	R/W	Power State. Determines the power state of the function. When the value in this field is D3 and the PMCSR register is written, a soft reset occurs; that is, the 21285 chip asserts nRESET and resets the internal state as if **PCI_RST** had been asserted.

The sticky-bit operation described in the *PCI Bus Power Management Interface Specification* must be emulated by SA-110 firmware. The following events must take place:

1. The SA-110 firmware wakes up the system and writes PMCSR[15] to 1.

2. The 21285 asserts **Pre_PME** due to the SA-110 setting the **PME_Status** bit (assuming that **PME_En** is set).
3. System software, in response to PME#, writes Power State from D3 to D0. This causes the 21285 to assert nRESET, which clears all registers, including PMCSR, and resets the SA-110.
4. When initializing the 21285, the SA-110 must write **PME_Status** and **PME_En** to 1 to indicate to system software that this device was the one that requested the system to wake up. This implies some state, visible to the SA-110, which is not changed by nRESET assertion. Because the SA-110 can lock out host configuration register reads until initialization software is complete, the firmware is guaranteed time to set the bits before the host can read them.

7.1.26 Data—Offset 77h

This register can be used by the SA-110 software to report data values in response to writes by the host to the Data_Select field of the PMCSR register.

This register is cleared at reset.

Dword Bit	R/W	Description
7:0	R/W from SA-110 R from PCI	Data. This register is used to report the state-dependent data requested by the Data_Select field. The value of this register is scaled by the value reported by the Data_Scale field.

7.2 PCI Control and Status Registers

PCI control and status registers, which are not defined in the PCI specification, are specific to the 21285. These registers are accessed from PCI by memory and/or I/O commands. These registers are also accessible from the SA-110 (offset is from address 4200 0000h).

The mailbox registers are byte writable; all other registers are writable as Dwords.

The outbound interrupt status register and the outbound interrupt mask register (at offsets 30h and 34h respectively) are not accessible to the SA-110. SA-110 accesses to offset 30h will access the expansion ROM base address register. SA-110 accesses to offset 34h are reserved.

I_2O registers are used to implement a message unit. The message unit is described in Section 6.3. The associated SA-110 control and status registers are described in Section 7.3.

The 21285 provides the mailbox and doorbell registers to allow for communication between the SA-110 and the host processor.

Table 7-3 lists the PCI control and status registers.

Table 7-3. PCI Control and Status Registers

31 0	Offset
Reserved	00h to 2Ch
Outbound Interrupt Status	30h
Outbound Interrupt Mask	34h
Reserved	38h to 3Ch
Inbound FIFO (I_2O)	40h
Outbound FIFO (I_2O)	44h
Reserved	48h to 4Ch
Mailbox 0	50h
Mailbox 1	54h
Mailbox 2	58h
Mailbox 3	5Ch
Doorbell	60h
Doorbell Setup	64h
ROM Write Byte Address	68h
Reserved	6Ch to 7Ch

7.2.1 Outbound Interrupt Status Register—Offset 30h

The outbound interrupt status register indicates the reasons why the 21285 is asserting **pci_irq_l**.

Dword Bit	Name	R/W	Description
1:0	—	R	Read only as 0.
2	Doorbell interrupt	R	Reads as 1 to indicate that for at least one bit position, both doorbell PCI mask and doorbell register have that bit set. Reset value: 0.
3	Outbound post list interrupt	R	Reads as 1 to indicate that the outbound post_list count register is not equal to 0; that is, there is at least one MFA on the outbound post_list. Reset value: 0.
31:4	—	R	Read only as 0.

7.2.2 Outbound Interrupt Mask Register—Offset 34h

The outbound interrupt mask register is used to allow the host processor to disable the 21285 from asserting **pci_irq_l**.

Dword Bit	Name	R/W	Description
1:0	—	R	Read only as 0.
2	Doorbell interrupt mask	R/W	When 1: Disables doorbell interrupts. Reset value: 0.
3	Outbound post list interrupt mask	R/W	When 1: Disables outbound post list interrupts. Reset value: 0.
31:4	—	R	Read only as 0.

7.2.3 I_2O Inbound FIFO Register—Offset 40h

The I_2O Inbound FIFO register is used to access the inbound free_list and inbound post_list from PCI memory space. This register appears as a read-only register if accessed by the SA-110.

7.2.3.1 Reading I_2O Inbound FIFO

When the I_2O Inbound FIFO is read from PCI (from CSR memory base address—offset 40h), the 21285 reads the SDRAM using the value in the inbound free_list head pointer register as the address.

- If the inbound free_list count register is not equal to 0, then it:
  a. Returns the data read from SDRAM to the PCI master.
  b. Increments the value in the inbound free_list head pointer register modulo I_2O FIFO size.
  c. Decrements the value in the inbound free_list count register.
- If the inbound free_list count register is 0, then it returns FFFFFFFFh to the PCI master.

7.2.3.2 Writing I_2O Inbound FIFO

When the I_2O Inbound FIFO is written from PCI (from CSR memory base address—offset 40h), the 21285 does the following:

1. Writes SDRAM using the value in the inbound post_list tail pointer register as the address. The data written to SDRAM is the PCI write data.
2. Increments the value in the inbound post_list tail pointer register modulo I_2O FIFO size.
3. Increments the value in the inbound post_list count register that will interrupt the SA-110 if enabled.

7.2.4 I_2O Outbound FIFO Register—Offset 44h

The I_2O Outbound FIFO register is used to access the outbound free_list and outbound post_list from PCI memory space. This register appears as a read-only register if accessed by the SA-110.

7.2.4.1 Reading I_2O Outbound FIFO

When the I_2O Outbound FIFO is read from PCI (from CSR memory base address—offset 44h), the 21285 reads the SDRAM using the value in the outbound post_list head pointer register as the address.

- If the outbound post_list count register is not equal to 0, then it:
  a. Returns the data read from SDRAM to the PCI master.
  b. Increments the value in the outbound post_list head pointer register modulo I_2O FIFO size.
  c. Decrements the value in the outbound post_list count register.
- If the outbound post_list count register is 0, then it returns FFFFFFFFh to the PCI master.

7.2.4.2 Writing I_2O Outbound FIFO

When the I_2O Outbound FIFO is written from PCI (from CSR memory base address—offset 44h), the 21285 does the following:

1. Writes SDRAM using the value in the outbound free_list tail pointer register as the address. The data written to SDRAM is the PCI write data.
2. Increments the value in the outbound free_list tail pointer register modulo I_2O FIFO size.

7.2.5 Mailbox *n* Registers—Offset 50h to 5Ch

The mailbox *n* registers consist of four registers: mailbox 0 through mailbox 3. The mailbox *n* registers can be read and written, with byte resolution, from both the SA-110 and PCI.

Dword Bit	Name	R/W	Description
31:0	Mailbox *n* data	R/W	Passes messages between the SA-110 and the host processor. Usage is application dependent. The messages are not used internally by the 21285 in any way. Reset value: Contents of mailbox *n* are undefined.

7.2.6 Doorbell Register—Offset 60h

Each bit in the doorbell register is write-1-to-set from the SA-110, write-1-to-clear from the PCI bus.

Dword Bit	Name	R/W	Description
31:0	Software interrupt	R/W1S from SA-110 and R/W1C from PCI	Causes a software interrupt from the SA-110 to host processor, or from the host processor to SA-110. Reset value: Contents are undefined.

7.2.7 Doorbell Setup—Offset 64h

Doorbell setup is an alias of the doorbell register to allow for initialization and test. It is a read/write register.

To initialize the doorbells:

1. Write doorbell PCI mask register (see Section 7.3.26) with a 1 in all bit positions used for SA-110-to-PCI notification.
2. Write doorbell SA-110 mask register (see Section 7.3.27) with a 1 in all bit positions used for PCI-to-SA-110 notification.
3. Write doorbell setup register such that all bit positions used for SA-110-to-PCI notification are written with 0, and all bit positions used for PCI-to-SA-110 notification are written with 1. Unused bits can be written to either value.

Dword Bit	Name	R/W	Description
31:0	Read/write data address	R/W	Writes to this address; place data directly into the doorbell register. Reads to this address; read the value in the doorbell register.

7.2.8 ROM Write Byte Address Register—Offset 68h

The ROM write byte address register is used to supply the two low-order bits of the ROM address during write cycles.

Dword Bit	Name	R/W	Description
1:0	ROM address	R/W	Supplies the two low-order bits of the ROM address during write cycles. It must be controlled by software during writes to byte or word wide ROMs, since neither the SA-110 or PCI can put the data on the low-byte lane(s) and simultaneously place the correct byte (word) address on the two low-order address bits. Reset value: Undefined.
31:2	—	R	Read only as 0.

7.3 SA-110 Control and Status Registers

SA-110 control and status registers are accessible only from the SA-110 (offset is from address 4200 0000h). These registers are not byte writable.

Table 7-4 lists the control and status registers.

Table 7-4. SA-110 Control and Status Registers (Sheet 1 of 3)

Register	Offset
DMA Channel 1 byte count	80h
DMA Channel 1 PCI address	84h
DMA Channel 1 SDRAM address	88h
DMA Channel 1 descriptor pointer	8Ch
DMA Channel 1 control	90h
DMA Channel 1 DAC address	94h
Reserved	98h to 9Ch
DMA Channel 2 byte count	A0h
DMA Channel 2 PCI address	A4h
DMA Channel 2 SDRAM address	A8h
DMA Channel 2 descriptor pointer	ACh
DMA Channel 2 control	B0h
DMA Channel 2 DAC address	B4h
Reserved	B8h to ECh
CSR base address mask	F8h
CSR base address offset	FCh
SDRAM base address mask	100h
SDRAM base address offset	104h
Expansion ROM base address mask	108h
SDRAM timing	10Ch
SDRAM address and size (array 0)	110h
SDRAM address and size (array 1)	114h
SDRAM address and size (array 2)	118h
SDRAM address and size (array 3)	11Ch
Inbound free_list head pointer (I_2O)	120h
Inbound post_list tail pointer (I_2O)	124h
Outbound post_list head pointer (I_2O)	128h
Outbound free_list tail pointer (I_2O)	12Ch
Inbound free_list count (I_2O)	130h
Outbound post_list count (I_2O)	134h
Inbound post_list count (I_2O)	138h
SA-110 control	13Ch

Table 7-4. SA-110 Control and Status Registers (Sheet 2 of 3)

Register	Offset
PCI address extension	140h
Prefetchable memory range	144h
X-Bus cycle/Arbiter	148h
X-Bus I/O strobe mask	14Ch
Doorbell PCI mask	150h
Doorbell SA-110 mask	154h
UARTDR	160h
RXSTAT	164h
H_UBRLCR	168h
M_UBRLCR	16Ch
L_UBRLCR	170h
UARTCON	174h
UARTFLG	178h
Reserved	17Ch
IRQStatus	180h
IRQRawStatus	184h
IRQEnable/IRQEnableSet	188h
IRQEnableClear	18Ch
IRQSoft	190h
SA-110 DAC address[a]	200h
SA-110 DAC address[a]	204h
SA-110 DAC control	208h
PCI address 31 alias	20Ch
Reserved	194h to 19Ch
FIQStatus	280h
FIQRawStatus	284h
FIQEnable/FIQEnableSet	288h
FIQEnableClear	28Ch
FIQSoft	290h
Reserved	294h to 2FCh
Timer1Load	300h
Timer1Value	304h
Timer1Control	308h
Timer1Clear	30Ch
Reserved	310h to 31Ch
Timer2Load	320h
Timer2Value	324h
Timer2Control	328h

Table 7-4. SA-110 Control and Status Registers (Sheet 3 of 3)

Register	Offset
Timer2Clear	32Ch
Reserved	330h to 33Ch
Timer3Load	340h
Timer3Value	344h
Timer3Control	348h
Timer3Clear	34Ch
Reserved	350h to 35Ch
Timer4Load	360h
Timer4Value	364h
Timer4Control	368h
Timer4Clear	36Ch
Reserved	370h to 7FCh

a. This register is accessed by two different addresses.

7.3.1 DMA Channel *n* Byte Count Register—Offset 80h/A0h

The DMA channel *n* byte count register (*n* = 1 or 2) contains the following fields.

Dword Bit	Name	R/W	Description
23:0	Byte count	R/W	Indicates the number of bytes to be transferred. It is updated internally after each read as the DMA operation progresses. Reset value: Undefined.
29:24	Channel interburst delay	R/W	Indicates the number of counts of the prescaled value (see Section 7.3.5 bits [9:8]) that the channel will wait before attempting another PCI burst. Reset value: Undefined.
30	Channel transfer direction	R/W	When 0: PCI to SDRAM. When 1: SDRAM to PCI. Reset value: Undefined.
31	End of chain	R/W	When 0: Indicates that more descriptor list entries follow. When 1: Indicates that the current operation is the last in the chain. Reset value: Undefined.

7.3.2 DMA Channel *n* PCI Address Register—Offset 84h/A4h

The DMA channel *n* PCI address register (*n* = 1 or 2) contains the low 32 bits of the DMA transfer's PCI address. It is the address of the source of data for PCI-to-SDRAM transfers, and of the destination of data for SDRAM-to-PCI transfers.

It is loaded with the start address of the transfer and is updated internally after each PCI transaction as the transfer proceeds. The initial value does not need to be Dword aligned. The PCI byte enable is adjusted according to the two low-order bits of the address. After the first PCI access, the updated value is Dword aligned.

Dword Bit	Name	R/W	Description
31:0	PCI address	R/W	Contains the address on the PCI bus for reads (PCI to SDRAM) or writes (SDRAM to PCI). It is updated internally as the DMA operation progresses. Reset value: Undefined.

7.3.3 DMA Channel *n* SDRAM Address Register—Offset 88h/A8h

The DMA channel *n* SDRAM address register (*n* = 1 or 2) contains the SDRAM address of the DMA transfer. It is the address of the source of data for SDRAM-to-PCI transfers, and of the destination of data for PCI-to-SDRAM transfers.

It is loaded with the start address of the transfer and is updated internally after each SDRAM transaction as the transfer proceeds. The initial value does not need to be Dword aligned. The SDRAM **dqm** is adjusted according to the two low-order bits of the address. After the first SDRAM access, the updated value is Dword aligned.

Dword Bit	Name	R/W	Description
27:0	SDRAM address	R/W	Contains the address of the SDRAM for reads (SDRAM to PCI) or writes (PCI to SDRAM). It is updated internally as the DMA operation progresses. Reset value: Undefined.
31	Descriptor/DAC control	R/W	This bit determines if the fourth Dword of the descriptor represents the next descriptor address of the DAC address, which can be 0. 0 = Next descriptor 1 = DAC address When the channel initial descriptor in register bit [4] of the Channel Control Register is set, the software writes 0 to this bit. Reset value: 0
30:28	—		Reserved.

7.3.4 DMA Channel *n* Descriptor Pointer Register—Offset 8Ch/ACh

The DMA channel *n* descriptor pointer register (*n* = 1 or 2) contains the SDRAM address of the next DMA descriptor for this channel. If the end-of-chain bit in the DMA channel *n* byte count register is 0, the DMA channel reads the next descriptor from SDRAM when the current transfer is done.

The descriptor must be aligned, that is, the four low-order bits of the address must be 0.

Dword Bit	Name	R/W	Description
3:0	—	R	Read only as 0.
27:4	Descriptor pointer	R/W	Contains the address of the next descriptor in SDRAM. Reset value: Undefined.
31:28	—	R	Read only as 0.

Table 7-5 lists the format of the descriptor block in memory.

Table 7-5. Descriptor Block Format

Offset from Descriptor Pointer	Description
0h	Byte count
4h	PCI address
8h	SDRAM address
Ch	Next descriptor pointer or DAC Address

7.3.5 DMA Channel *n* Control Register—Offset 90h/B0h

The DMA channel *n* control register (*n* = 1 or 2) contains values that control the DMA channels for the duration of a chain operation.

Dword Bit	Name	R/W	Description (Sheet 1 of 2)
0	Channel enable	R/W	When 0: Channel not active. When written from 0 to 1: Channel fetches the first descriptor block (unless channel initial descriptor in register bit [4] of this register is a 1), and then performs DMA operations until it does a transfer with the end-of-chain bit equal to 1. This bit is cleared internally when channel chain done is set. Reset value: 0.
1	—	R	Read only as 0.
2	Channel transfer done	W1C	Indicates that a transfer has completed, that is, the transfer count from one of the descriptors has reached 0. Reset value: 0.
3	Channel error	W1C	Indicates that the channel detected either a PCI master abort, target abort, or parity error during a PCI transfer, or bad SDRAM parity during a SDRAM read. When set, the channel stops operation regardless of the byte count and/or end-of-chain bits. Reset value: 0.
4	Channel initial descriptor in register	R/W	When 0: Indicates that the channel must read the first descriptor from SDRAM. When 1: Indicates that the SA-110 has written the first descriptor to the channel registers. Channel reads of subsequent descriptors, if any, are not affected by this bit. Note that if this bit is set, the DAC register must be loaded along with the other transfer parameters if a nonzero value is required. Reset value: Undefined.
6:5	Channel PCI read type	R/W	This field is only meaningful if the transfer direction is PCI to SDRAM, that is, reads are from the PCI bus. It defines the command type that should be used during the PCI reads. • 00=Memory read • 01=Memory read line • 10=Memory read multiple • 11=Memory read multiple Reset value: Undefined.

Dword Bit	Name	R/W	Description (Sheet 2 of 2)
7	Channel chain done	W1C	Indicates that a chain has completed either normally or due to an error condition. When this bit is a 1, it can interrupt the SA-110 if enabled in the FIQ/IRQEnable register. Reset value: 0.
9:8	Channel interburst delay prescale	R/W	Indicates the prescale value for the counter that determines the number of SA-110 cycles that the channel waits before attempting another PCI burst. The number of counts of the prescaled value is read in the descriptor. • 00=4 • 01=8 • 10=16 • 11=32 Reset value: Undefined.
14:10	—	R	Read only as 0.
15	PCI read length	R/W	This field defines the number of Dwords that the 21285 attempts to read per burst from the PCI during PCI-to-SDRAM transfers (during the beginning or end of a transfer the number may be less). • 0=8 Dwords • 1=16 Dwords Reset value: Undefined.
18:16	SDRAM read length	R/W	This field defines the number of Dwords read from SDRAM per burst for SDRAM-to-PCI transfers (during the beginning or end of a transfer the number may be less). • 000=1 Dword • 001=2 Dword • 010=4 Dword • 011=8 Dword • 100=16 Dword • 101, 110, 111=Reserved Reset value: Undefined.
31:19	—	R	Read only as 0.

7.3.6 DMA Channel *n* DAC Address—Offset 94h/B4h

The DMA channel *n* DAC Address register (*n* = 1 or 2) holds the upper 32 bits of the 64-bit PCI address. If this register has a value of zero, the PCI address is in the low 4GB of PCI memory space and DAC cycles are not performed on the PCI. If the value is nonzero, then DAC cycles are performed.

This register can be written by software prior to enabling the channel, or from the fourth Dword of the first descriptor (if the first descriptor is fetched from memory). As each subsequent descriptor is fetched from memory, this register is either left unmodified or written from the fourth Dword of the descriptor. In both cases, the decision whether to update this register from the descriptor is determined by a bit in the third Dword of the descriptor. See Section 6.2.1 for more information about DMA channel operation.

The DMA channel *n* DAC Address register is cleared by chip reset, and also when the channel-done bit is set at the completion of a DMA chain.

Dword Bit	Name	R/W	Description
31:0	DAC Address	RW	Provides bits 63:32 of the PCI address. Reset value: 0.

Note: This register does not increment during a DMA transfer. This means that a transfer that crosses a 4GB boundary must be performed with two descriptors.

7.3.7 CSR Base Address Mask Register—Offset F8h

Dword Bit	Name	R/W	Description
17:0	—	R	Read only as 0.
27:18	Mask	R/W	When 1: Causes the corresponding bit in the CSR memory base address register to act as a read-only 0 bit. When 0: Causes the corresponding bit in the CSR memory base address register to act as a read/write bit. Reset value: 0.
31:28	—	R	Read only as 0.

Table 7-6 lists the values that should be programmed into the CSR base address mask register to enable the different window sizes from the PCI. All other values are illegal.

Table 7-6. PCI Window Sizes

Window Size	Value
128B	00000000h
512KB	00040000h
1MB	000C0000h
2MB	001C0000h
4MB	003C0000h
8MB	007C0000h
16MB	00FC0000h
32MB	01FC0000h
64MB	03FC0000h
128MB	07FC0000h
256MB	0FFC0000h

7.3.8 CSR Base Address Offset Register—Offset FCh

Dword Bit	Name	R/W	Description
17:0	—	R	Read only as 0.
27:18	Offset	R/W	Each bit of this register is used as a CSR address if the corresponding bit of the CSR base address mask register is a 0. Reset value: 0.
31:28	—	R	Read only as 0.

7.3.9 SDRAM Base Address Mask Register—Offset 100h

The SDRAM base address mask register must be written by the SA-110 before the initialize complete bit [1] in the SA-110 control register is set.

Dword Bit	Name	R/W	Description
17:0	—	R	Read only as 0.
27:18	Mask	R/W	When 1: Causes the corresponding bit in the SDRAM base address register to act as a read-only 0 bit. When 0: Causes the corresponding bit in the SDRAM base address register to act as a read/write bit. Reset value: 0.
30:28	—	R	Read only as 0.
31	SDRAM window disable	R/W	When 1: Causes bits [31:28] in the SDRAM base address register to act as read-only 0 bits. When 0: Causes bits [31:28] in the SDRAM base address register to act as read/write bits. SA-110 software can indicate to configuration software to allow for no window to SDRAM by making the entire SDRAM base address register read as 0. This is accomplished by setting this bit and the mask field of this register to all 1s. Reset value: 0.

Table 7-7 lists the values that should be programmed into the SDRAM base address mask register to enable the different window sizes from PCI into SDRAM. All other values are illegal.

Table 7-7. SDRAM Window Sizes

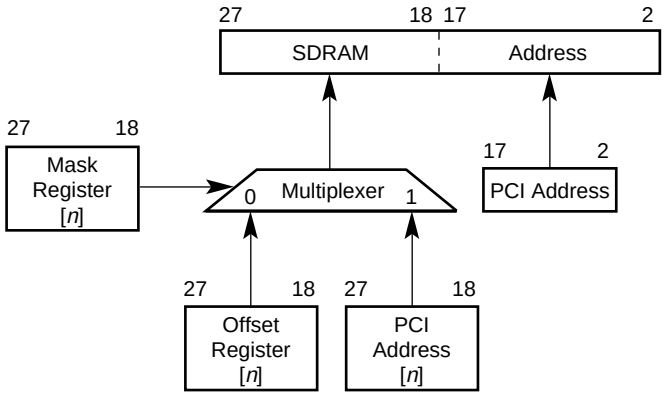
Window Size	Value
256KB	00000000h
512KB	00040000h
1MB	000C0000h
2MB	001C0000h
4MB	003C0000h
8MB	007C0000h
16MB	00FC0000h
32MB	01FC0000h
64MB	03FC0000h
128MB	07FC0000h
256MB	0FFC0000h
No window	8FFC0000h

7.3.10 SDRAM Base Address Offset Register—Offset 104h

The SDRAM base address offset register must be written by the SA-110 before the initialize complete bit [1] in the SA-110 control register is set.

Dword Bit	Name	R/W	Description
17:0	—	R	Read only as 0.
27:18	Offset	R/W	This field contains the upper address bits for PCI-generated SDRAM accesses (see Figure 7-1). Each bit of this register is used as a SDRAM address if the corresponding bit of the SDRAM base address mask register is a 0. Reset value: 0.
31:28	—	R	Read only as 0.

Figure 7-1. PCI-Generated SDRAM Access



7.3.11 Expansion ROM Base Address Mask Register—Offset 108h

The expansion ROM base address mask register must be written by the SA-110 before the initialize complete bit [1] in the SA-110 control register is set.

Dword Bit	Name	R/W	Description
19:0	—	R	Read only as 0.
23:20	Mask	R/W	When 1: Causes the corresponding bit in the expansion ROM base address register to act as a read-only 0 bit. When 0: Causes the corresponding bit in the expansion ROM base address register to act as a read/write bit. Reset value: Fh.
30:24	—	R	Read only as 0.
31	ROM window disable	R/W	When 1: Causes bits [31:24] and bit [0] in the expansion ROM base address register to act as read-only 0 bits. When 0: Causes bits [31:24] and bit [0] in the expansion ROM base address register to act as read/write bits. SA-110 software can indicate to configuration software to allow for no window to ROM by making the entire expansion ROM base address register read as 0. This is accomplished by setting this bit and the mask field of this register to all 1s. Reset value: 0.

Table 7-8 lists the values that should be programmed into the expansion ROM base address mask register to enable the different window sizes into ROM. All other values are illegal.

Table 7-8. Expansion ROM Window Sizes

Window Size	Value
1MB	00000000h
2MB	00100000h
4MB	00300000h
8MB	00700000h
16MB	00F00000h
No window	80F00000h

7.3.12 SDRAM Timing Register—Offset 10Ch

This register controls timing for all SDRAMs. All cycles are referenced to **fclk_in**. Refer to the SDRAM specification for information about the proper values to use in this register.

Dword Bit	Name	R/W	Description (Sheet 1 of 2)
1:0	Row precharge time (T_{rp})	R/W	The minimum number of cycles from autoprecharge internally in the SDRAM to the next row activate or auto-refresh command. For reads, autoprecharge is assumed to start four clock cycles after the read command. If the next command is row activate to a different SDRAM bank, it does not need to wait for $T_{rp.}$ • 00=1 cycle • 01=2 cycles • 10=3 cycles • 11=4 cycles Reset value: 0.
3:2	Last data in to activate or refresh (T_{dal})	R/W	On write cycles, this is the minimum number of cycles from the last (4th) data being written to the next activate or refresh command. T_{dal} will generally be greater than T_{rp} due to the relationship $T_{dal} = T_{rp} + T_{dpl}$ (where T_{dpl} is the time from the last data being written to the start of the autoprecharge). • 00=2 cycles • 01=3 cycles • 10=4 cycles • 11=5 cycles Reset value: 0.
5:4	RAS-to-CAS delay (T_{rcd})	R/W	The number of cycles from a row activate command to the first read or write command. • 00=Reserved • 01=Reserved • 10=2 cycles • 11=3 cycles Reset value: 0.
7:6	CAS latency (T_{cas})	R/W	The number of cycles from a read command to valid data from the SDRAM. This value must be the same as is written to the SDRAM mode register bits [5:4]. (Bit [6] of the SDRAM mode register must be 0.) • 00=Reserved • 01=Reserved • 10=2 cycles • 11=3 cycles Reset value: 0.

Dword Bit	Name	R/W	Description (Sheet 2 of 2)
10:8	Row cycle time (T_{rc})	R/W	The minimum number of cycles from an auto-refresh command to the next row activate command. (If bit [11], command drive time, of this register is a 1, then T_{rc} will be one cycle longer than selected here; that is, the command is asserted on **cmd** at the end of T_{rc} and **cs_l** is asserted one cycle later.) • 000=Reserved • 001=4 cycles • 010=5 cycles • 011=6 cycles • 100=7 cycles • 101=8 cycles • 110=9 cycles • 111=10 cycles Reset value: 0.
11	Command drive time	R/W	Indicates the number of cycles from **ba**, **ma**, and **cmd** lines valid to **cs_l** asserted. This can be used to allow for buffer delay in a large memory system. When 0: Same cycle. When 1: 1 cycle. Reset value: 0.
12	Parity enable	R/W	Controls whether or not parity is enabled for the SDRAMs. When 0: No parity enabled. The pins **parity[3:0]** are tristate. No parity check on reads. When 1: Parity is enabled. The pins **parity[3:0]** are driven on writes; value received from SDRAMs is checked on reads. Reset value: 0.
13	SA-110 Prime	R/W	Indicates that the SA-110 chip will drive parity information when it is performing a write. This bit should not be written to 1 if Parity Enable (bit [12] of this register) is not a 1. Reset value: 0.
15:14	—	R	Read only as 0.
21:16	Refresh interval (T_{ref})	R/W	Indicates the number of cycles, prescaled by 32, between SDRAM refresh operations. A value of 0h disables refresh. The auto-refresh command will be done at the next SA-110 bus arbitration point (see Section 5.1.3) after the refresh interval expires, but the timer reloads and counts immediately; that is, it does not wait for the refresh to occur before reloading. All SDRAMs are refreshed at the same time. Reset value: 0.
31:22	—	R	Read only as 0.

7.3.13 SDRAM Address and Size Registers—Offsets 110h to 11Ch

The SDRAM address and size registers consist of four registers controlling array 0 through array 3. These four registers define each of the four SDRAM arrays' start address, size, and address multiplexing. Software must ensure that the arrays of SDRAM are mapped so there is no overlap of addresses. The arrays *do not* need to all be the same size, however, the start address of each array must be naturally aligned to the size of the array. The arrays *do not* need to form a contiguous address space, but to do so with different size arrays, place the largest array at the lowest address, next largest array above, and so on.

Note: The 21285 does not detect memory accesses to nonexistent memory locations. The SDRAM sequencer may possibly toggle address and command pins, return unpredictable read data on reads, and discard data on writes.

Dword Bit	Name	R/W	Description
2:0	Array size	R/W	Indicates the size of the SDRAM array. This field is decoded to form a mask that is used in comparing the memory address to the array base field when determining if the address is in this array. • 000=0 (array disabled) • 001=1MB (compare bits [27:20]) • 010=2MB (compare bits [27:21]) • 011=4MB (compare bits [27:22]) • 100=8MB (compare bits [27:23]) • 101=16MB (compare bits [27:24]) • 110=32MB (compare bits [27:25]) • 111=64MB (compare bits [27:26]) Reset value: Undefined.
3	—	R	Read only as 0.
6:4	Address multiplex	R/W	Controls the row and column multiplexer and implicitly defines the number of banks in the SDRAM array according to Table 4-2. Reset value: Undefined.
19:7	—	R	Read only as 0.
27:20	Array base	R/W	Indicates the base address of the array. This value is compared to address bits [27:20], masked by the decoded array size field, to determine which array of SDRAMs the address is located in. Reset value: Undefined.
31:28	—	R	Read only as 0.

7.3.14 I_2O Inbound Free_List Head Pointer Register—Offset 120h

The value written into the inbound free_list head pointer register represents the address in SA-110 SDRAM space of the head (next entry to be read by a PCI read to 40h) of the I_2O inbound free_list.

Dword Bit	Name	R/W	Description
1:0	—	R	Read only as 0.
27:2	Inbound free_list entry	R/W	Address of I_2O inbound free_list head. Reset value: Undefined.
31:28	—	R	Read only as 0.

7.3.15 I_2O Inbound Post_List Tail Pointer Register—Offset 124h

The value written into the inbound post_list tail pointer register represents the address in SA-110 SDRAM space of the tail (next entry to be written by a PCI write to 40h) of the I_2O inbound post_list.

Dword Bit	Name	R/W	Description
1:0	—	R	Read only as 0.
27:2	Inbound post_list entry	R/W	Address of I_2O inbound post_list tail. Reset value: Undefined.
31:28	—	R	Read only as 0.

7.3.16 I_2O Outbound Post_List Head Pointer Register—Offset 128h

The value written into the outbound post_list head pointer register represents the address in SA-110 SDRAM space of the head (next entry to be read by a PCI read to 44h) of the I_2O outbound post_list.

Dword Bit	Name	R/W	Description
1:0	—	R	Read only as 0.
27:2	Outbound post_list entry	R/W	Address of I_2O outbound post_list head. Reset value: Undefined.
31:28	—	R	Read only as 0.

7.3.17 I_2O Outbound Free_List Tail Pointer Register—Offset 12Ch

The value written into the outbound free_list tail pointer register represents the address in SA-110 SDRAM space of the tail (next entry to be written by a PCI write to 44h) of the I_2O outbound free_list.

Dword Bit	Name	R/W	Description
1:0	—	R	Read only as 0.
27:2	Outbound free_list entry	R/W	Address of I_2O outbound free_list tail. Reset value: Undefined.
31:28	—	R	Read only as 0.

7.3.18 I_2O Inbound Free_List Count Register—Offset 130h

The contents of the inbound free_list count register contain the number of entries available on the I_2O inbound free_list. When the SA-110 writes to this address, the action is dependent on the value of bit 31 of the data. If 0, the value in the register is incremented by one (disregarding the rest of the write data); if 1, the data is written into the register. When a PCI bus master removes an entry from the free_list (via a read to offset 40h), the value is decremented by one.

Dword Bit	Name	R/W	Description
15:0	Inbound free_list count	R/W	I_2O inbound free_list count. Reset value: 0.
31:16	—	R	Read only as 0.

7.3.19 I_2O Outbound Post_List Count Register—Offset 134h

The contents of the outbound post_list count register contain the number of entries available on the I_2O outbound post_list. When the SA-110 writes to this address, the action is dependent on the value of bit 31 of the data. If 0, the value in the register is incremented by one (disregarding the rest of the write data); if 1, the data is written into the register. When a PCI bus master removes an entry from the post_list (via a read to offset 44h), the value is decremented by one. When the value is not equal to 0, the 21285 asserts **pci_irq_l** (if not masked by the outbound interrupt mask).

Dword Bit	Name	R/W	Description
15:0	Outbound post_list count	R/W	I_2O outbound post_list count. Reset value: 0.
31:16	—	R	Read only as 0.

7.3.20 I_2O Inbound Post_List Count Register—Offset 138h

The contents of the inbound post_list count register contain the number of entries available on the I_2O inbound post_list. When a PCI bus master places an entry onto the post_list (via a write to offset 40h), the value is incremented by one. When the SA-110 writes to this address, the action is dependent on the value of bit 31 of the data. If 0, the value in the register is decremented by one (disregarding the rest of the write data); if 1, the data is written into the register. When the value is not equal to 0, an interrupt is posted to the SA-110 (if not masked by IRQEnable/FIQEnable).

Dword Bit	Name	R/W	Description
15:0	Inbound post_list count	R/W	I_2O inbound post_list count. Reset value: 0.
31:16	—	R	Read only as 0.

7.3.21 SA-110 Control Register—Offset 13Ch

Dword Bit	Name	R/W	Description (Sheet 1 of 3)
0	Initialize complete	R/W	This bit indicates that the SA-110 software has initialized the 21285 CSRs. Specifically, the following CSRs should be loaded before this bit is set. • SDRAM base address mask register • SDRAM base address offset register • Expansion ROM base address mask register When 0: The 21285 returns retry response as the target of PCI configuration cycles, and will not assert **devsel_l** to the PCI, I/O, or memory commands. When 1: The 21285 returns normal response to PCI configuration cycles. This allows the SA-110 to initialize CSRs such as subsystem vendor ID, base address masks, and so on before the host processor's configuration software can read them. Reset value: 0.
1	Assert SERR	R/W	When 1: The 21285 asserts **serr_l** for one cycle if **pci_cfn** is deasserted and command register bit SERR# enable bit [8] is a 1. Reset value: 0.
2	—	R	Read only as 0.
3	Received SERR	W1C	Set when **serr_l** is asserted by an external device and **pci_cfn** is asserted. It can cause an interrupt to the SA-110 if enabled in the IRQEnable/FIQEnable registers (see Section 7.3.31). Reset value: 0.

Dword Bit	Name	R/W	Description (Sheet 2 of 3)
4	SA-110 SDRAM parity error	W1C	Set when an SA-110 read from the SDRAM has bad parity and the SDRAM parity is enabled. It can cause an interrupt to the SA-110 if enabled in the IRQEnable/FIQEnable registers (see Section 7.3.31). Reset value: 0.
5	PCI SDRAM parity error	W1C	Set when a PCI-originated read from the SDRAM has bad parity and the SDRAM parity is enabled. It can cause an interrupt to the SA-110 if enabled in the IRQEnable/FIQEnable registers (see Section 7.3.31). Reset value: 0.
6	DMA SDRAM parity error	W1C	Set when a DMA read from the SDRAM has bad parity and the SDRAM parity is enabled. It can cause an interrupt to the SA-110 if enabled in the IRQEnable/ FIQEnable registers (see Section 7.3.31). Reset value: 0.
7	—	R	Read only as 0.
8	Discard timer expired	W0C	Set when the discard timer counts to 0 and the 21285 has discarded read data (see Section 3.2.3). It can cause an interrupt to the SA-110 if enabled in the IRQEnable/ FIQEnable registers (see Section 7.3.31). To clear this error, software must write a 0 to this bit. Reset value: 0.
9	PCI not reset	R/W	When 1: The 21285 does not assert **pci_rst_l**. When 0: The 21285 asserts **pci_rst_l** if **pci_cfn** is asserted. This bit may be used to assert **pci_rst_l** without resetting the SA-110 or 21285. Software must meet the minimum specification timing when **pci_rst_l** asserts. In addition, assertion of this bit resets the Inbound FIFO, the Outbound FIFO, and the PCI data FIFO. While this bit is set, no SA-110-to-PCI transactions can be enqueued. Reset value: 0.
12:10	I_2O list size	R/W	Indicates how many entries are in the I_2O inbound and outbound lists, and controls which bits of the I_2O head and tail registers count, and which are constant. Table 7-9 lists the entry and control data. Reset value: 0.

Dword Bit	Name	R/W	Description (Sheet 3 of 3)
13	Watchdog enable	W1S	When 1: Timer 4 resets the SA-110 when it is enabled and reaches 0. This bit is a write 1 to set (W1S); once it has been written to a 1, it can only be cleared by a chip reset. Reset value: 0.
15:14	ROM width	R	Indicates the width of the ROM that controls the number of ROM accesses to pack on ROM reads. Reflects the value latched from the **ma[5:4]** lines. • 11=Byte width • 01=Word width • 10=Dword width • Undefined
19:16	ROM access time	R/W	Indicates the number of **fclk_in** cycles to wait from address, chip enable, and output enable driven to the ROM, until ROM data is latched on the first ROM access of a Dword. Exception: a value of 0 means 16 cycles; values of 1 and 2 are illegal. All other values are valid. Reset value: 0.
23:20	ROM burst time	R/W	Indicates the number of **fclk_in** cycles to wait from address incremented to the ROM, until ROM data is latched on the second, third, and fourth ROM accesses of a Dword (if required). Exception: a value of 0 means 16 cycles; values of 1 and 2 are illegal. All other values are valid. Reset value: 0.
27:24	ROM tristate time	R/W	Indicates the number of **fclk_in** cycles to wait from ROM output enable deasserted, until the 21285 allows any other agent, that is; the SA-110, SDRAM, X-Bus, or 21285 to drive **D**. Exception: a value of 0 means 16 cycles. A value of 1 is illegal. Reset value: 0.
30:28	XCS direction	R/W	Controls the direction of **xcs_l[2:0]**. When 0: Input. When 1: Output. ***Note:*** When the PCI Arbiter is selected, these bits must not be written to a 1. Reset value: 0.
31	PCI central function	R	This bit reflects the value on the **pci_cfn** pin.

Table 7-9 lists the size of the I_2O inbound and outbound lists.

Table 7-9. I_2O Inbound and Outbound List Sizes

Value	Number of Entries	Count Bits
000	256	[9:2]
001	512	[10:2]
010	1K	[11:2]
011	2K	[12:2]
100	4K	[13:2]
101	8K	[14:2]
110	16K	[15:2]
111	32K	[16:2]

7.3.22 PCI Address Extension Register—Offset 140h

Bit [15] of this register provides bit [31] of the PCI address during both dual address and single address cycles. Bit [15] is written with SA-110 write data bit [15] during writes to the PCI Address Extension Register address. It is cleared by any write to the SA-110 DAC Address Register at offset 200h: it is set by any write to the SA-110 DAC Address Register at offset 204h. It is also readable at the PCI Address 31 Alias Register (offset 20Ch).

Dword Bit	Name	R/W	Description
14:0	—	R	Read only as 0.
15	PCI memory space upper address bit	R/W	This bit provides PCI address [31] during SA-110-originated PCI memory accesses. Reset value: Undefined.
31:16	PCI I/O space upper address field	R/W	These bits provide PCI address [31:16] during SA-110-originated PCI I/O accesses. Reset value: Undefined.

7.3.23 Prefetchable Memory Range Register—Offset 144h

Dword Bit	Name	R/W	Description
0	Prefetch range enable	R/W	When 0: There is no match to the prefetchable memory range. All SA-110-to-PCI memory space reads use the memory read command. Reset value: 0.
1	Prefetch read type	R/W	Determines the command to use for an SA-110-to-PCI memory space read when there is a match to the prefetchable memory range. When 0: Memory read line command. When 1: Memory read multiple command. Reset value: 0.
4:2	Prefetch length	R/W	Determines the maximum number of Dwords that the 21285 attempts to prefetch when there is a match to the prefetchable memory range. • 000=1 Dword • 001=2 Dwords • 010=4 Dwords • 011=8 Dwords • 100=16 Dwords • 101=32 Dwords • 110=Reserved • 111=Reserved Reset value: 0.
7:5	—	R	Read only as 0.
18:8	Mask	R/W	When 1: Ignore the corresponding bit in the compare for prefetchable PCI memory space. Bits [18:8] mask bits [30:20] of the address respectively. The mask allows for the prefetchable range to be from 1MB (no bits masked) to 2GB (all bits masked). The size of the prefetchable memory range must be naturally aligned. Reset value: 0.
19	—	R	Read only as 0.
30:20	Base address	R/W	The value in this field is compared to SA-110-generated PCI address bits [30:20] (masked by the mask field of this register) to determine if it is in the prefetchable range. If it is in the prefetchable range, the command specified by the prefetch read type field of this register is used on reads. If it is not in the prefetchable range, the memory read command is used. Reset value: 0.
31	—	R	Read only as 0.

7.3.24 X-Bus Cycle/Arbiter Register—Offset 148h

This register is used either to control the parallel port (X-Bus) or the internal PCI Arbiter. The choice is based on the value latched on **ma[7]** at reset.

When X-Bus is selected (**ma[7]**=1), the register contents are as follows.

Dword Bit	Name	R/W	Description
2:0	Device 0 cycle length	R/W	Contains the number of X-Bus cycles for an access to the X-Bus device in **xcs_l[0]** range. Reset value: 0.
5:3	Device 1 cycle length	R/W	Contains the number of X-Bus cycles for an access to the X-Bus device in **xcs_l[1]** range. Reset value: 0.
8:6	Device 2 cycle length	R/W	Contains the number of X-Bus cycles for an access to the X-Bus device in **xcs_l[2]** range. Reset value: 0.
11:9	Device *n* cycle length	R/W	Contains the number of X-Bus cycles for an access to the X-Bus device in no chip select range. Reset value: 0.
13:12	Strobe shift divisor	R/W	**fclk_in** is divided by the value in this field to determine the frequency of the clock that is used to time the cycle length and shift the strobe mask. • 00=1 • 01=2 • 10=3 • 11=4 Reset value: 0.
22:14	—	R	Read only as 0.
23	PCI Arbiter	R	Read only as 0. Allows software to determine that the X-Bus is selected.
27:24	Interrupt input level	R/W	These bits control the assertion level of **irq_in_l[3:0]**. The values are 0 for low assertions and 1 for high assertions. Reset value: 0.
30:28	X-Bus chip select	R/W	These bits control the assertion level of **xcs_l[2:0]**. The values are 0 for low assertions and 1 for high assertions. Reset value: 0.
31	PCI interrupt request	R/W	This bit controls the assertion level of **pci_irq_l** when used as an input pin. The value is 0 for low assertion and 1 for high assertion. Reset value: 0.

When the internal PCI Arbiter is selected (**ma[7]**=0), the register contents are as follows.

Dword Bit	Name	R/W	Description
4:0	Arbiter priority	R/W	Indicates which priority group each master is assigned to. 0=Low priority. 1=High priority. Bit [4] controls the 21285 priority; bits [3:0] controls request [3:0] respectively. Reset value: 0.
13:5	Undefined	R/W	—
22:14	—	R	Read only as 0.
23	PCI Arbiter	R	Read only as 1. Allows software to determine that the PCI Arbiter is selected.
27:24	Interrupt input level	R/W	These bits control the assertion level of **irq_in_l[3:0]**. The values are 0 for low assertions and 1 for high assertions. Reset value: 0.
30:28	X-Bus chip select	R/W	These bits control the assertion level of **xcs_l[2:0]**. The values are 0 for low assertions and 1 for high assertions. Reset value: 0.
31	PCI interrupt request	R/W	This bit controls the assertion level of **pci_irq_l** when used as an input pin. The value is 0 for low assertion and 1 for high assertion. Reset value: 0.

7.3.25 X-Bus I/O Strobe Mask Register—Offset 14Ch

Dword Bit	Name	R/W	Description
7:0	I/O device 0 strobe mask	R/W	Strobe mask shifted out to **xior_l** or **xiow_l** (for read or write respectively) during an access to X-Bus device in **xcs_l[0]** range. When 1: Strobe asserts. When 0: Strobe deasserts. Reset value: 0.
15:8	I/O device 1 strobe mask	R/W	Strobe mask shifted out to **xior_l** or **xiow_l** during an access to X-Bus device in **xcs_l[1]** range. When 1: Strobe asserts. When 0: Strobe deasserts. Reset value: 0.
23:16	I/O device 2 strobe mask	R/W	Strobe mask shifted out to **xior_l** or **xiow_l** during an access to X-Bus device in **xcs_l[2]** range. When 1: Strobe asserts. When 0: Strobe deasserts. Reset value: 0.
31:24	I/O device *n* strobe mask	R/W	Strobe mask shifted out to **xior_l** or **xiow_l** during an access to X-Bus device in range. When 1: Strobe asserts. When 0: Strobe deasserts. Reset value: 0.

7.3.26 Doorbell PCI Mask Register—Offset 150h

The doorbell PCI mask register is used to individually enable each bit of the doorbell register to assert an interrupt to PCI. An internal signal, doorbell PCI interrupt, is asserted if the corresponding bits of doorbell PCI mask and doorbell registers are both set.

Dword Bit	Name	R/W	Description
31:0	Doorbell PCI mask interrupt	R/W	Doorbell PCI interrupt causes the assertion of **pci_irq_l** if enabled in the outbound interrupt mask register. Reset value: 0.

7.3.27 Doorbell SA-110 Mask Register—Offset 154h

The doorbell SA-110 mask register is used to individually enable each bit of the doorbell register to assert an interrupt to SA-110. An internal signal, doorbell SA-110 interrupt, is asserted if the corresponding bits of doorbell SA-110 mask is set and doorbell is clear.

Dword Bit	Name	R/W	Description
31:0	Doorbell SA-110 mask interrupt	R/W	Doorbell SA-110 interrupt causes the assertion of **nIRQ** or **nFIQ** if enabled in the IRQEnable or FIQEnable registers (see Section 7.3.31). Reset value: 0.

7.3.28 Interrupt Controller Registers

There are 27 interrupt sources, and each interrupt source can be used to generate **nIRQ** or **nFIQ** (or neither, or both). The control registers for the **nFIQ** and **nIRQ** control are identical. The interrupt controller does not impose any priority on the interrupts; all enabled interrupts are delivered at equal priority. The Status and RawStatus are read-only, and EnableClear and Soft are write-only. Also, Enable and EnableSet share the same address for reads and writes respectively. Figure 7-2 shows the configuration for each interrupt source.

Figure 7-2. Interrupt Source Configuration

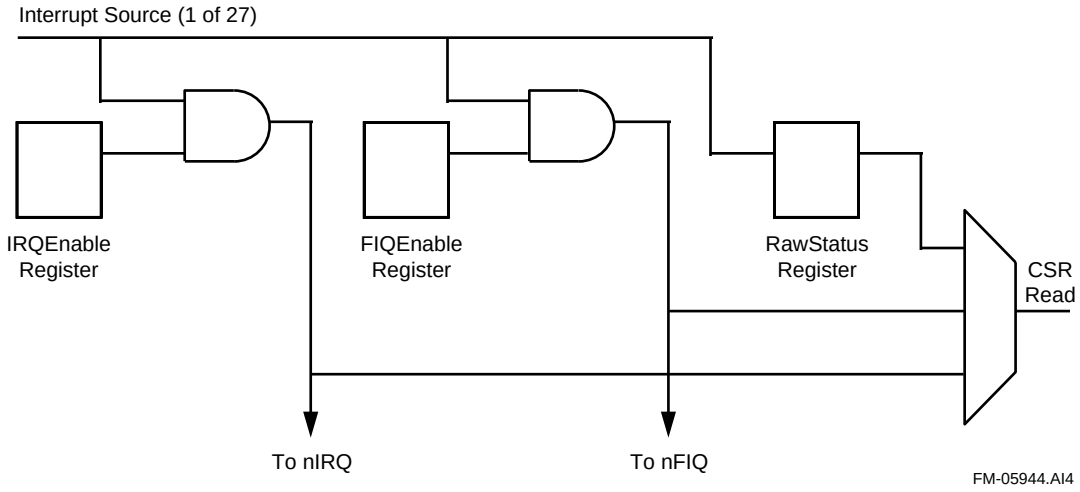


Table 7-10 lists the interrupt controller registers.

Table 7-10. Interrupt Controller Registers

Address	Read	Write
180h	IRQStatus	—
184h	IRQRawStatus	—
188h	IRQEnable	IRQEnableSet
18Ch	—	IRQEnableClear
190h	—	IRQSoft
280h	FIQStatus	—
284h	FIQRawStatus	—
288h	FIQEnable	FIQEnableSet
28Ch	—	FIQEnableClear
290h	—	FIQSoft

Table 7-11 lists the 32 available interrupt source bit positions. Unused bit positions are reserved for future use and read as 0 for read-only registers, and must be 0 for write-only registers. The bit position for each of the interrupt sources is the same for all registers. All interrupts are level-sensitive; the specific mechanism for clearing each interrupt source is also listed in the table.

Table 7-11. Interrupt Source Bits (Sheet 1 of 2)

Bits	Interrupt Source	How Cleared
0	Reserved	—
1	Soft interrupt	Write 0 to IRQSoft/FIQSoft bit 1
2	Console RX	Refer to serial port
3	Console TX	Refer to serial port

Table 7-11. Interrupt Source Bits (Sheet 2 of 2)

Bits	Interrupt Source	How Cleared
4	Timer 1	Write to TIMER1Clear (any data)
5	Timer 2	Write to TIMER2Clear (any data)
6	Timer 3	Write to TIMER3Clear (any data)
7	Timer 4	Write to TIMER4Clear (any data)
8	irq_in_l[0]	Specific to external device
9	irq_in_l[1]	Specific to external device
10	irq_in_l[2]	Specific to external device
11	irq_in_l[3]	Specific to external device
12	xcs_l[0]	Specific to external device
13	xcs_l[1]	Specific to external device
14	xcs_l[2]	Specific to external device
15	Doorbell from host	Write to doorbell SA-110 mask or doorbell registers
16	DMA channel 1	W1C bit in DMA control register
17	DMA channel 2	W1C bit in DMA control register
18	pci_irq_l	Specific to external device
19	PMCSR written by host	Write PMCSR from SA-110
21:20	Reserved	—
22	Start BIST	Write 0 to BIST register bit [6]
23	Received SERR	W1C bit in SA-110 control register
24	SDRAM parity	W1C bits in SA-110 control register
25	I_2O inbound post_list	Retire all inbound posted entries
26	Reserved	—
27	Discard timer expired	W1C bit in SA-110 control register
28	Data parity error detected	W1C bit in PCI status register
29	Received master abort	W1C bit in PCI status register
30	Received target abort	W1C bit in PCI status register
31	Detected parity error	W1C bit in PCI status register

7.3.29 IRQStatus/FIQStatus Register—Offset 180h/280h

This read-only register is a bit-wise AND of RawStatus and IRQEnable/FIQEnable.

Dword Bit	Name	R/W	Description
31:0	IRQStatus/ FIQStatus	R	A 1 in a bit position indicates that the associated interrupt source is both active and enabled. The **nIRQ/nFIQ** output pin is the NOR of all of the bits in this register.

7.3.30 IRQRawStatus/FIQRawStatus Register—Offset 184h/284h

The IRQRawStatus and FIQRawStatus read-only registers always contain identical data.

Dword Bit	Name	R/W	Description
31:0	IRQRawStatus/ FIQRawStatus	R	A 1 in a bit position indicates that the associated interrupt source is active.

7.3.31 IRQEnable/FIQEnable Register—Offset 188h/288h

This read-only register is used to mask the interrupt input sources and defines which active sources generate an interrupt request to the SA-110 on **nIRQ/nFIQ**. The value of this register can only be changed by writing to the EnableSet and EnableClear registers.

Dword Bit	Name	R/W	Description
31:0	IRQEnable/FIQEnable	R	A 1 indicates that the associated interrupt source is enabled and allows an interrupt request. A 0 indicates that the interrupt is disabled. Reset value: 0.

7.3.32 IRQEnableSet/FIQEnableSet Register—Offset 188h/288h

This write-only register is used to set bits in the IRQEnable/FIQEnable register.

Dword Bit	Name	R/W	Description
31:0	IRQEnableSet/ FIQEnableSet	WO	When writing to this location, each data bit that is high causes the corresponding bit in the enable register to be set. Data bits that are low have no effect on the corresponding bit in the enable register.

7.3.33 IRQEnableClear/FIQEnableClear Register—Offset 18Ch/28Ch

This write-only register is used to clear bits in the IRQEnable/FIQEnable register.

Dword Bit	Name	R/W	Description
31:0	IRQEnableClear/ FIQEnableClear	WO	When writing to this location, each data bit that is high causes the corresponding bit in the enable register to be cleared. Data bits that are low have no effect on the corresponding bit in the enable register.

7.3.34 IRQSoft/FIQSoft Register—Offset 190h/290h

This write-only register can be used to generate an IRQ/FIQ under software control.

Dword Bit	Name	R/W	Description
0	—	—	Don't care.
1	RawStatus register bit [1]	WO	This bit generates bit [1] (as either 0 or 1) of the RawStatus register (and its current state can be found by reading that register).
31:2	—	—	Don't care.

7.3.35 SA–110 DAC Address Registers—Offsets 200h to 204h

This register provides bits 63:32 of the PCI address for SA-110 accesses to PCI memory space. It is accessed by two different addresses. For both addresses, this register is written with the SA-110 write data. The two addresses are used to control bit [15] of the PCI Address Extension register (Offset 140h) which supplies bit [31] of the PCI address for all SA-110 accesses to PCI memory space. Writing this register at address 200h will clear bit [15] of the PCI Address Extension register. Writing this register at address 204h will set bit [15] of the PCI Address Extension register.

Dword Bit	Name	R/W	Description
31:10	DAC Address	R/W	Provide bits 63:32 of the PCI address. Reset value: 0

7.3.36 SA-110 DAC Control Registers—Offset 208h

This register provides information used to control prefetching for SA-110 loads from DAC PCI memory space. DACs are performed if the value of all bits in the SA-110 DAC address register is not equal to 0.

Dword Bit	Name	R/W	Description
31:4	—	R	
3	DAC PF	R/W	Specifies if a DAC read should prefetch. When 0: PCI command is Memory Read and 1 Dword is read. When 1: PCI command is specified by DAC Read Command field of this register. Reset value: 0
2	DAC Read Command	R/W	Specifies the PCI command type for DAC read (if PF bit of this register is a 1). 0 = Memory Read Line 1 = Memory Read Multiple Reset value: 0
1:0	DAC Burst Length	R/W	Specifies the number of Dwords to be prefetched for a DAC read (if PF bit of this register is a 1). 00 = 4 01 = 8 10 = 16 11 = 32 Reset value: 0

7.3.37 PCI Address 31 Alias Register—Offset 20Ch

This register provides an alternate way to read the value of bit [15] of the PCI Address Extension Register, which provides the value for bit [31] of the PCI address during all SA-110 accesses to PCI memory space.

Dword Bit	Name	R/W	Description
31:3	—	R	
2	PCI Address 31	R	Bit [15] of the PCI Address Extension Register
1:0	—	R	

7.3.38 Timer Control Registers

The following CSRs control the timers. Table 7-12 lists the addresses of the registers. Timer*N*Value is read-only and Timer*N*Clear is write-only.

Table 7-12. Timer Control Registers

Register	Address
Timer1Load	300h
Timer1Value	304h
Timer1Control	308h
Timer1Clear	30Ch
Timer2Load	320h
Timer2Value	324h
Timer2Control	328h
Timer2Clear	32Ch
Timer3Load	340h
Timer3Value	344h
Timer3Control	348h
Timer3Clear	34Ch
Timer4Load	360h
Timer4Value	364h
Timer4Control	368h
Timer4Clear	36Ch

7.3.39 Timer*N*Load Register—Offsets 300h, 320h, 340h, and 360h

This register holds the initial value of the counter.

Dword Bit	Name	R/W	Description
23:0	Initial counter value	R/W	The act of writing to this register causes the counter to reload with the initial value. Reset value: 0.
31:24	—	R	Read only as 0.

7.3.40 Timer*N*Value Register—Offsets 304h, 324h, 344h, and 364h

This register holds the current value of the counter.

Dword Bit	Name	R/W	Description
31:0	Current counter value	R	Contains the current counter value. Reset value: Undefined.

7.3.41 Timer*N*Control Register—Offsets 308h, 328h, 348h, and 368h

This register controls the timer functions. Timer4 can be used as a watchdog timer. When the watchdog enable bit [13] in the SA-110 control register is set to a 1, Timer4Control cannot be written to. Therefore, software must load Timer4Control prior to setting the watchdog enable bit.

Dword Bit	Name	R/W	Description
1:0	—	R	Read only as 0.
3:2	Select timer pre-scaler	R/W	The decodes are: • 00=**fclk_in** • 01=**fclk_in**/16 • 10=**fclk_in**/256 • 11=External event on **irq_in_l** pin: **irq_in_l[0]** = Timer1 **irq_in_l[1]** = Timer2 **irq_in_l[2]** = Timer3 **irq_in_l[3]** = Timer4 Reset value: 0.
5:4	—	R	Read only as 0.
6	Mode	R/W	When 0: Timer is free running (when the timer reaches 0 it will wrap to FFFFFFh and continue to decrement). When 1: Timer is periodic (when the timer reaches 0 it will reload with the value in Timer*N*Load). Reset value: 0.
7	Enable	R/W	When 1: Timer is enabled. When reenabled, it will resume from the current Timer*N*Value. When 0: Timer is disabled. If a counter is disabled its Timer*N*Value can still be read. Reset value: 0.
31:8	—	R/W	Read as 0. Ignored on write.

7.3.42 Timer*N*Clear Register—Offsets 30Ch, 32Ch, 34Ch, and 36Ch

Any write to the Timer*N*Clear register (any data) clears the associated interrupt. The write has no effect if the associated interrupt was already clear. Timer*N*Clear is a write-only register.

Dword Bit	Name	R/W	Description
31:0	Clear	W	Clears interrupt. Data is ignored.

JTAG Test Port 8

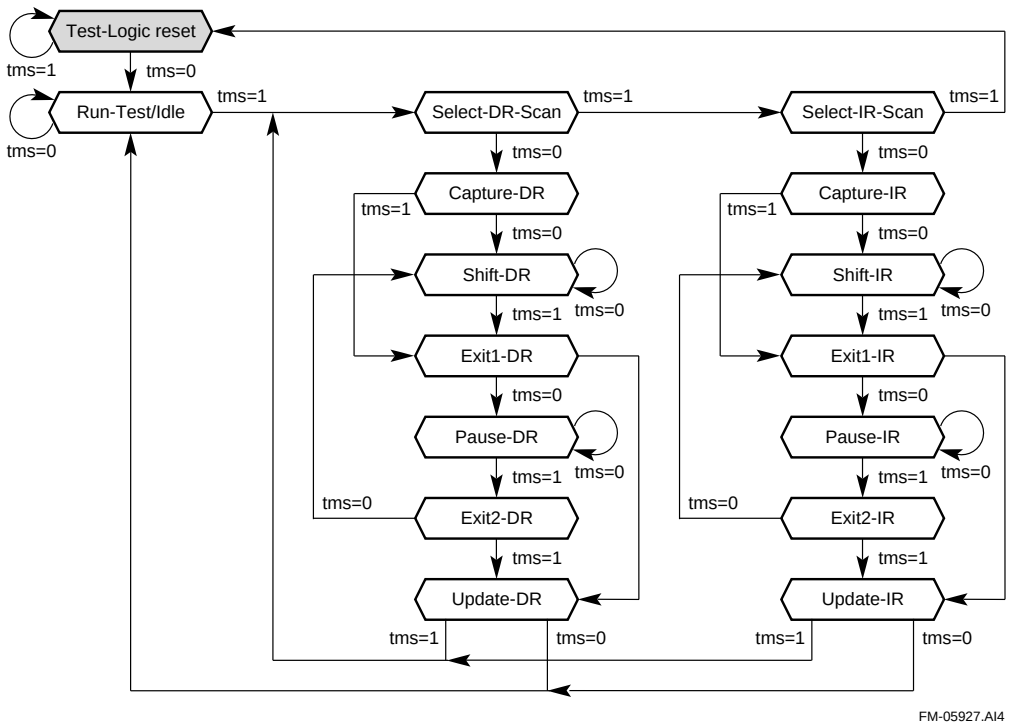
The 21285 contains a serial scan test port that conforms to IEEE standard 1149.1, *IEEE Standard Test Access Port and Boundary-Scan Architecture*.

8.1 Test Access Port Controller

The test access port controller is a finite state machine that interprets IEEE 1149.1 protocols received through the **tms** input.

The state transitions in the controller are caused by the **tms** signal on the rising edge of **tck**. In each state, the controller generates the appropriate clock and control signals that control the operation of the test features. After entry into a state, test-feature operations are initiated on the rising edge of **tck**. Figure 8-1 shows the state transitions.

Figure 8-1. Test Access Port (TAP) Controller State Transitions



8.2 Boundary-Scan Implementation Exceptions

The critical timing of the clocking signals **osc, fclk, sdclk[3:0],** and **MCLK** force the exception of the IEEE 1149.1 standards for these device pins. The following modifications have been implemented:

Note: BC_1, BC_2, and BC_4 are all defined in the IEEE Standard 1149.1, *IEEE Standard Test Access Port and Boundary-Scan Architecture*.

- Signal **osc** is an input, but uses a BC_1 cell to allow the state to be set by the BSR operation.
- Signals **fclk**, **sdclk[3:0]**, and **MCLK** do not use boundary-scan cells. In boundary-scan mode, these pins are logically equivalent to the **osc** pad, which can be controlled in unison from the **osc** BSR cell.

8.3 Instruction Register

The 4-bit instruction register selects the test modes and features. Table 8-1 lists the instruction register encodings that are interpreted as instructions. All other values are illegal. The instructions select and control the operation of the boundary-scan and bypass registers. The instruction register is loaded through the tdi pin, and has a shift-in stage that enables the instruction to be loaded in parallel.

Table 8-1. Test Modes and Features

Instruction Register Contents	Instruction Name (Test Mode or State)	Test Register Selected	Operation
0000	EXTEST	Boundary-scan	External test (the 21285 drives pins using the data in the boundary-scan register).
0001	SAMPLE/PRELOAD	Boundary-scan	Samples I/O.
0010	HIGHZ	Bypass	Tristates all output and I/O pins except the **tdo** pin.
0011	CLAMP	Bypass	Drives pins from the boundary-scan register and selects the bypass register for shifts.
0100	IDCODE	Idcode	Reads the manufacturer's identification number, the design part number, and the design version number.[a]
1111	BYPASS	Bypass	Selects the bypass register for shifts.

a. The manufacturer's identification number is 000001101011, the design part number is 0100000110010100, and the design version number is based on the latest revision.

8.4 Bypass Register

The bypass register is a 1-bit shift register that provides a means for effectively bypassing the JTAG test logic through a single-bit serial connection through the chip from **tdi** to **tdo**. At board-level testing, this helps reduce the overall length of the scan chain.

8.5 Boundary-Scan Register

The boundary-scan register is a single-shift register-based path formed by boundary-scan cells placed at the chip's signal pins. The register is accessed through the JTAG ports **tdi** and **tdo** pins.

A machine-readable boundary-scan index (BSDL.TXT) for each I/O cell in the 21285 design is located at the following URL:

http://ftp.digital.com/pub/Digital/info/semiconductor/literature/dsc-library.html#bridges

The list is ordered from **tdi** to **tdo**, and the index starts at 1 and ends at 360. **tdi** is applied on a rising edge of **tck**. **tdo** is valid on the falling edge.

Notes: The **tx** pin is always configured as an output except when the JTAG HIGHZ state is asserted. The **tdo** pin is enabled per the JTAG specification. All other tristate pins are controlled by the boundary-scan register scan cell indicated in BSDL.TXT, as well as the JTAG HIGHZ instruction.

All input ports are monitored by BC_4 type cells except for **osc**, which is controlled by a BC_1. EXTEST overrides the clock.

All outport ports are controlled by a BC_1 cell.

All bidirectional ports contain a BC_1 for the output path, and a BC_4 for the input path. The enable is controlled by a BC_2 cell.

fclk, **MCLK**, and **sdclk[3:0]** are not monitored or controlled by the boundary-scan register. These signals are indirectly controlled through, and are logically equivalent to the **osc** boundary-scan cell.

tx enable is controlled by the JTAG HIGHZ instruction and not by a boundary-scan register cell. During normal operation, this is an output port only.

tdo is controlled by the TAP controller.

Electrical Specifications 9

This section specifies the following electrical behavior of the 21285:

- PCI electrical conformance
- Absolute maximum ratings
- dc specifications
- ac timing specifications

9.1 PCI Electrical Specification Conformance

The 21285 PCI pins conform to the basic set of PCI electrical specifications in the *PCI Local Bus Specification,Revision 2.1*. See that document for a complete description of the PCI I/O protocol and pin ac specifications.

9.2 Absolute Maximum Ratings

The 21285 is specified to operate at a maximum frequency of 33 MHz at a junction temperature (T(j)) not to exceed 125°C. Table 9-1 lists the absolute maximum ratings for the 21285. Stressing the device beyond the absolute maximum ratings may cause permanent damage. These are stress ratings only. Operating beyond the functional operating range is not recommended, and extended exposure beyond the functional operating range may affect reliability.

Table 9-1. Absolute Maximum Ratings

Parameter	Minimum	Maximum
Junction temperature, T_j	—	125°C
Supply Voltage, V_{dd}	—	3.9 V
Maximum voltage applied to signal pins	—	5.5 V
Maximum power P_{WC}	—	1.77 W @ 33 MHz
Storage temperature range, T_{stg}	-55°C	125°C

Table 9-2 lists the functional operating range.

Table 9-2. Functional Operating Range

Parameter	Minimum	Maximum
Voltage supply, V_{dd}	3.0 V	3.6 V
Operating ambient temperature, T_a	0°C	70°C

9.3 DC Specifications

Table 9-3 defines the dc parameters met by all 21285 PCI signals under normal operating conditions.

Note: In Table 9-3, currents into the chip (chip sinking) are denoted as positive (+) current. Currents from the chip (chip sourcing) are denoted as negative (–) current.

Table 9-3. DC Parameters

Symbol	Parameter	Condition	Minimum	Maximum	Unit
Power Signal					
V_{dd}	Supply voltage	—	3.0	3.6	V
PCI Signals					
V_{il}	Low-level input voltage[a]	—	–0.5	0.3 V_{dd}	V
V_{ih}	High-level input voltage[a]	—	0.5 V_{dd}	V_{dd} + 0.5 V	V
V_{ol}	Low-level output voltage[b]	I_{out} = 1500 µA	—	0.1 V_{dd}	V
V_{ol5V}	Low-level output voltage[c]	I_{out} = 6 mA	—	0.55	V
V_{oh}	High-level output voltage[b]	I_{out} = –500 µA	0.9 V_{dd}	—	V
V_{oh5V}	High-level output voltage[d]	I_{out} = –2 mA	2.4	—	V
I_{il}	Low-level input leakage current[a,e]	0 < V_{in} < V_{dd}	—	±10	µA
C_{in}	Input pin capacitance	—	—	10.0	pF
C_{IDSEL}	**idsel** pin capacitance	—	—	8.0	pF
C_{clk}	**pci_clk** pin capacitance	—	5.0	12.0	pF
Non-PCI Signals					
V_{ihc}	IC input high voltage[d,e]	—	0.8 × V_{dd}	V_{dd} + 0.5	V
V_{ilc}	IC input low voltage[d,e]	—	–0.5	0.2 × V_{dd}	V
I_{ilc}[f]	Input leakage current	0 < V_{in} < V_{dd}	–10	10	µA
V_{ohc}[g]	High-level output voltage[d,e]	I_{oh} = –4 mA	2.4	—	V
V_{olc}[g]	Low-level output voltage[d,e]	I_{ol} = 4 mA	—	0.4	V
V_{ohc}[h]	High-level output voltage[d]	I_{oh} = –12 mA	2.4	—	V
V_{olc}[h]	Low-level output voltage	I_{ol} = 12 mA	—	0.4	V
I_{ts}	Tristate leakage current	—	—	100	µA
C_{in}	Input pin capacitance	—	—	5	pF
C_{io}	Input/output pin capacitance	—	—	12	pF
I_{dd}	Power supply current	—	—	490	mA

a. Guarantees meeting the specification for the 5-V signaling environment.
b. For 3.3-V signaling environment.
c. For 5-V signaling environment.
d. Voltage measured with respect to V_{ss}.
e. IC-CMOS-level inputs (include IC and ICOCZ pin types).
f. Leakage currents for **ma[12:0]** and **ba[1:0]** are -200 µA minimum and 200 µA maximum.
g. Not including clocks.
h. Including clocks (**MCLK**, **fclk**, and **sdclk[3:0]**).

9.4 AC Timing Specifications

The next sections describe the following specifications:

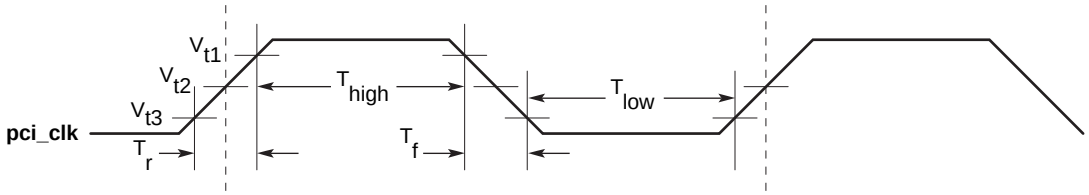
- PCI clock timing
- PCI signal timing
- PCI reset timing
- JTAG timing

9.4.1 PCI Clock Timing Specifications

The ac specifications consist of input requirements and output responses. The input requirements consist of setup and hold times, pulse widths, and high and low times. The output responses are delays from clock to signal. The ac specifications are defined separately for each clock domain (**pci_clk** and **fclk_in**) within the 21285.

Figure 9-1 shows the ac parameter measurements for the **pci_clk** signal, and Table 9-4 specifies **pci_clk** parameter values for clock signal ac timing. See also Figure 9-2 for a further illustration of signal timing.

Figure 9-1. PCI Clock Signal AC Parameter Measurement Conditions



Note:

V_{t1} – 2.0 V for 5-V clocks; 0.5 V_{cc} for 3.3-V clocks

V_{t2} – 1.5 V for 5-V clocks; 0.4 V_{cc} for 3.3-V clocks

V_{t3} – 0.8 V for 5-V clocks; 0.3 V_{cc} for 3.3-V clocks

FM-05688.AI4

Table 9-4. PCI Clock Signal AC Parameters

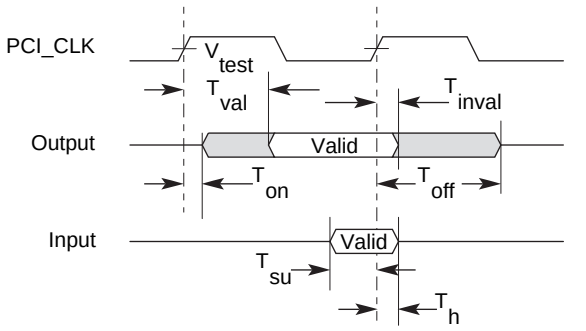
Symbol	Parameter	Minimum	Maximum	Unit
T_{cyc}	**pci_clk** cycle time	30	∞	ns
T_{high}	**pci_clk** high time	11	—	ns
T_{low}	**pci_clk** low time	11	—	ns
	pci_clk slew rate[a]	1	4	V/ns

a. 0.2 V_{cc} to 0.6 V_{cc}

9.4.2 PCI Signal Timing Specifications

Figure 9-2 and Table 9-5 show the PCI signal timing specifications. Table 9-6 correlates the ac parameters with the 21285 signal pins.

Figure 9-2. PCI Signal Timing Measurement Conditions



Note:
V_{test} – 1.5 V for 5-V signals; 0.4 V_{cc} for 3.3-V signals

FM-05676.AI4

Table 9-5. PCI Signal Timing Specifications

Symbol	Parameter	Minimum	Maximum	Unit
T_{val}	**pci_clk** to signal valid delay — bused signals[a,b,c]	2	11	ns
$T_{val(ptp)}$	**pci_clk** to signal valid delay — point-to-point[a,b,c]	2	12	ns
T_{on}	Float to active delay[a,b]	2	—	ns
T_{off}	Active to float delay[a,b]	—	28	ns
T_{su}	Input setup time to **pci_clk**—bused signals[a,b,c]	7	—	ns
$T_{su(ptp)}$	Input setup time to **pci_clk**—point-to-point[a,b,c]	10, 12	—	ns
T_{h}	Input signal hold time from **pci_clk**[a,b]	0	—	ns

a. See Figure 9-2.
b. All PCI interface signals are referenced to **pci_clk**.
c. Point-to-point signals are **req_l** and **gnt_l**. Bused signals are **ad, cbe_l, par, perr_l, serr_l, frame_l, irdy_l, trdy_l, devsel_l, stop,** and **idsel.**

Table 9-6 lists the ac parameters that are applied in Table 9-5.

Table 9-6. PCI Signal Timing AC Parameters

Name	T_{val}	$T_{val\ (ptp)}$	T_{on}, T_{off}	T_{su}, T_h	$T_{su\ (ptp)}$, T_h
ad[31:0]	y	—	y	y	—
cbe_l[3:0]	y	—	y	y	—
par	y	—	y	y	—
frame_l	y	—	y	y	—
irdy_l	y	—	y	y	—
trdy_l	y	—	y	y	—
stop	y	—	y	y	—
devsel_l	y	—	y	y	—
idsel	—	—	—	y	—
perr_l	y	—	y	y	—
serr_l	y	—	—	—	—
req_l	—	y	—	—	—
gnt_l	—	—	—	—	y
pci_irq_l	y	—	—	—	—

9.4.3 PCI Reset Timing Specifications

Table 9-7 shows the reset timing specifications for **pci_rst_l**.

Table 9-7. PCI Reset Timing Specifications

Symbol	Parameter	Minimum	Maximum	Unit
T_{rst}	**pci_rst_l** active time after power stable[a]	1	—	ms
$T_{rst—clk}$	**pci_rst_l** active time after **pci_clk** stable[a]	100	—	µs
$T_{rst—off}$	**pci_rst_l** active-to-output float delay[b]	—	40	ns
	pci_rst_l slew rate[a]	50	—	mV/ns

a. Applies to rising (deasserting) edge only.
b. All PCI output drivers are asynchronously floated when **pci_rst_l** is asserted (except **ad**, **cbe_l**, and **par**, which are driven to 0 if **pci_cfn** is asserted).

9.4.4 JTAG Timing Specifications

Table 9-8 shows the JTAG timing specifications.

Table 9-8. JTAG Timing Specifications

Symbol	Parameter	Minimum	Maximum	Unit
T_{jf}	tck frequency	0	10	MHz
T_{jht}	**tck** high time	45	—	ns
T_{jlt}	**tck** low time	45	—	ns
T_{jrt}	**tck** rise time[a]	—	10	ns
T_{jft}	**tck** fall time[b]	—	10	ns
T_{js}	**tdi**, **tms** setup time to **tck** rising edge	10	—	ns
T_{jh}	**tdi**, **tms** hold time from **tck** rising edge	25	—	ns
T_{jd}	**tdo** valid delay from **tck** falling edge[c]	—	30	ns
T_{jfd}	**tdo** float delay from **tck** falling edge	—	30	ns

a. Measured between 0.8 V and 2.0 V.
b. Measured between 2.0 V and 0.8 V.
c. C_1 = 50 pF.

9.5 Memory and SA-110 Interface Timing

The timing parameters specified in this section are valid for the full range of voltage and temperature variations as stated in Sections 9.2 and 9.3. The ac specifications consist of input requirements and output responses. The input requirements consist of setup and hold times, pulse widths, and high and low times. The output responses are delays from clock to signal. All signal timings are referenced to the rising edge of **fclk_in** unless otherwise stated.

All rise/fall AC parameters are measured from 10% to 90% of V_{dd}. All other AC parameters are measured from 50% V_{dd} of **fclk_in** to 50% V_{dd} of signal specified.

9.5.1 SA-110, 21285, and SDRAM Clock Signal Timing Specifications

Figure 9-3 and Table 9-9 show the SA-110, 21285, and SDRAM clock signal timing specifications.

Figure 9-3. Clock Signal AC Parameter Measurement Conditions

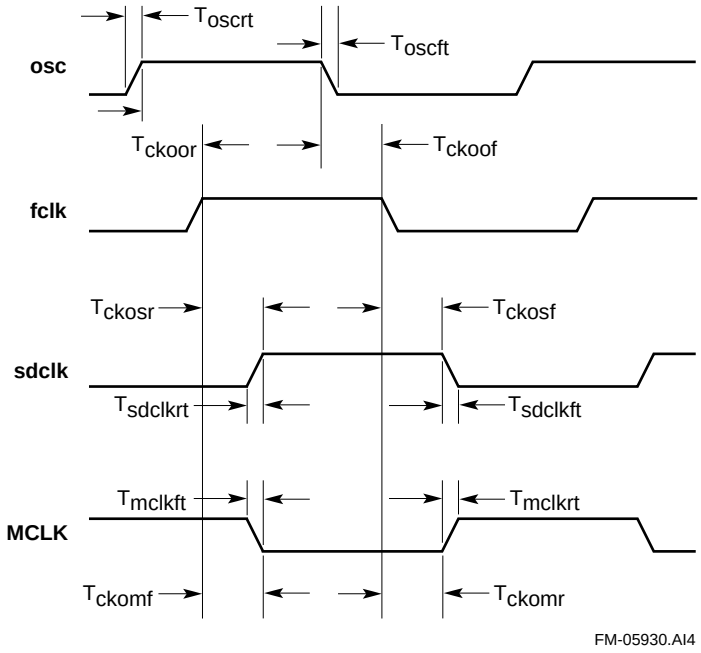


Table 9-9. SA-110, 21285, and SDRAM Clock Signal AC Parameters (Sheet 1 of 2)

Name	Parameter	Minimum	Maximum	Unit
	osc period	15.0	$<T_{cyc}$	ns
	osc high time	6.7	—	ns
	osc low time	6.7	—	ns
T_{ckoor}	**osc** rising **fclk** rising	1.9	4.4	ns
T_{ckoof}	**osc** falling **fclk** falling	1.9	4.0	ns
T_{oscrt}	**osc** rise time	0.5	2.0	ns
T_{oscft}	**osc** fall time	0.5	2.0	ns
T_{ckoi}	**fclk** to **fclk_in** delay[a]	—	—	ns
T_{ckomf}	**fclk** to **MCLK** delay - **fclk** rising	1.6	3.4	ns
T_{ckomr}	**fclk** to **MCLK** delay - **fclk** falling	1.8	3.9	ns
T_{mclkrt}	**MCLK** rise time	0.5	2.0	ns
T_{mclkft}	**MCLK** fall time	0.5	2.0	ns
T_{ckosr}	**fclk** to **sdclk** delay - **fclk** rising[b]	1.6	3.6	ns
T_{ckosf}	**fclk** to **sdclk** delay - **fclk** falling[b]	1.6	3.2	ns

Table 9-9. SA-110, 21285, and SDRAM Clock Signal AC Parameters (Sheet 2 of 2)

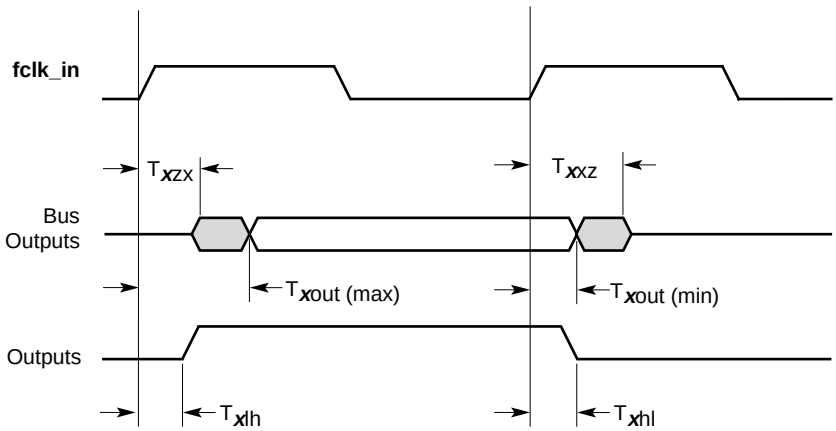
Name	Parameter	Minimum	Maximum	Unit
$T_{sdclkrt}$	**sdclk** rise time	0.5	1.0	ns
$T_{sdclkft}$	**sdclk** fall time	0.5	1.0	ns
T_{skew1}	Skew between rising edges on any two of **sdclk[3:0]**	—	0.5	ns
T_{skew2}	Skew between **MCLK** falling and any of **sdclk[3:0]** rising	—	0.5	ns

a. This delay is not specified. It is imposed by etch delays in the module design. In the module design, the capacitive load and etch length on the following nets should be matched: 21285 **fclk** output to 21285 **fclk_in**; 21285 **MCLK** output to SA-110 MCLK pin; 21285 **sdclk[3:0]** outputs to SDRAM CLK pin.
b. Any of **sdclk[3:0]**.

9.5.2 SA-110, 21285, and SDRAM Interface Timing Specifications

Figures 9-4 and 9-5 with Table 9-10 show the memory and SA-110 interface timing specifications.

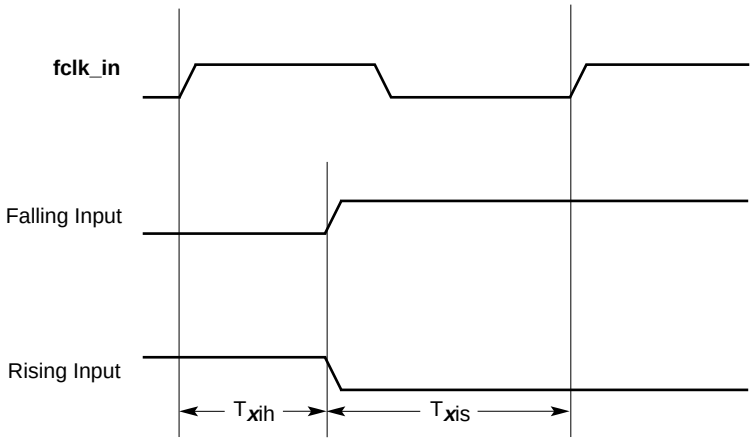
Figure 9-4. Interface Timing Measurement Conditions



Timing Symbol(s)	Corresponding Parameter Value
$T_{x}zx$	T_{dzx}, T_{pzx}, T_{azx}, T_{dqmzx}, T_{mazx}, T_{cmdzx}, T_{cszx}
$T_{x}xz$	T_{dxz}, T_{pxz}, T_{axz}, T_{dqmxz}, T_{maxz}, T_{cmdxz}, T_{cszx}
T_{x}out (max),T_{x}out (min)	T_{dout}, T_{pout}, T_{aout}, T_{dqmout}, T_{maout}, T_{cmdout}, T_{csout}
$T_{x}lh$	T_{xwlh}, T_{dwlh}, T_{rcslh}, T_{rdlh}, T_{wrlh}, T_{xcslh}, T_{abelh}, T_{dbelh}
$T_{x}hl$	T_{xwhl}, T_{dwhl}, T_{rcshl}, T_{rdhl}, T_{wrhl}, T_{abehl}, T_{abehl}, T_{dbehl}

FM-05928.AI4

Figure 9-5. Input Timing Measurement Conditions



Timing Symbol(s)	Corresponding Parameter Value
T_{xih}	T_{dih}, T_{mrih}, T_{loih}, T_{rwih}, T_{clih}, T_{pih} T_{dqmh}, T_{aih}
T_{xis}	T_{dis}, T_{mris}, T_{lois}, T_{rwis}, T_{clis}, T_{pis}, T_{dqms}, T_{ais}

FM-05929.AI4

Table 9-10. Memory and SA-110 AC Parameters **(Sheet 1 of 2)**

Name	Parameter	Minimum	Maximum	Unit
T_{dis}	T_{dis} data-in setup[a]	0.5	—	ns
T_{dih}	T_{dih} data-in hold[a]	1.6	—	ns
T_{pis}	T_{pis} parity-in setup	0.5	—	ns
T_{pih}	T_{pih} parity-in hold	1.6	—	ns
T_{dout}	Data-out delay	4.9	11.4	ns
T_{pouta}	Asynchronous parity-out delay (SA-110 write)	2.9	10.0	ns
T_{pouts}	Synchronous parity-out delay (21285 write)	4.9	11.8	ns
T_{dzx}	Data turn-on time	4.0	11.2	ns
T_{dxz}	Data turn-off time	4.0	11.2	ns
T_{pzx}	Parity turn-on time	4.0	11.1	ns
T_{pxz}	Parity turn-off time	4.0	11.1	ns
T_{ais}	Address input setup	1.7	—	ns
T_{aih}	Address input hold	1.0	—	ns
T_{aout}	Address output delay (includes **rom_oe_l**, **rom_we_l**)	4.4	11.8	ns
T_{azx}	Address turn-on time	4.3	13.0	ns
T_{axz}	Address turn-off time	4.3	13.0	ns
T_{dqma}	**dqm[3:0]** maximum output delay during SA-110 write, timed from SA-110 address in	—	9.1	ns
T_{dqmh}	**dqm[3:0]** minimum output hold during SA-110 write, timed from rising **fclk_in**	4.6	—	ns

Table 9-10. Memory and SA-110 AC Parameters (Sheet 2 of 2)

Name	Parameter	Minimum	Maximum	Unit
T_{dqms}	**dqm[3:0]** output delay during 21285 write	4.5	10.6	ns
T_{maout}	**ma**, **ba** output delay	4.7	10.8	ns
T_{cmdout}	**cmd[2:0]** output delay	5.0	11.5	ns
T_{csout}	**cs_l[3:0]** output delay	5.0	11.4	ns
T_{xwhl}	**xd_wren_l** assertion delay	5.3	11.1	ns
T_{xwlh}	**xd_wren_l** negation delay	5.1	11.7	ns
T_{dwhl}	**d_wren_l** assertion delay	4.5	9.3	ns
T_{dwlh}	**d_wren_l** negation delay	4.6	10.0	ns
T_{rcshl}	**rom_ce_l** assertion delay	4.5	9.6	ns
T_{rcslh}	**rom_ce_l** negation delay	4.6	10.2	ns
T_{rdhl}	**xrd_l** assertion delay	5.3	10.6	ns
T_{rdlh}	**xrd_l** negation delay	5.1	11.1	ns
T_{wrhl}	**xwr_l** assertion delay	5.4	10.8	ns
T_{wrlh}	**xwr_l** negation delay	5.1	11.2	ns
T_{xcshl}	**xcs_l** assertion delay[b]	5.0	11.1	ns
T_{xcslh}	**xcs_l** negation delay[b]	5.0	11.4	ns
T_{abehl}	**ABE** negation delay	5.1	10.3	ns
T_{abelh}	**ABE** assertion delay	4.8	10.4	ns
T_{dbehl}	**DBE** negation delay	4.4	9.1	ns
T_{dbelh}	**DBE** assertion delay	4.5	9.6	ns
T_{mris}	**nMREQ** input setup	1.6	—	ns
T_{mrih}	**nMREQ** input hold	0.9	—	ns
T_{lois}	**LOCK** input setup	0.9	—	ns
T_{loih}	**LOCK** input hold	0.6	—	ns
T_{rwis}	**nRW** input setup	0.5	—	ns
T_{rwih}	**nRW** input hold	1.1	—	ns
T_{clis}	**CLF** input setup	1.0	—	ns
T_{clih}	**CLF** input hold	0.6	—	ns

a. For SA-110 writes to 21285, 21285 reads of SDRAM, and 21285 reads of ROM.
b. For **xcs** signals that are configured as outputs.

Mechanical Specifications 10

The 21285 is contained in an industry-standard 256 plastic ball grid array (PBGA) package, as shown in Figure 10-1.

Figure 10-1. 256 Plastic Ball Grid Array (PBGA) Package

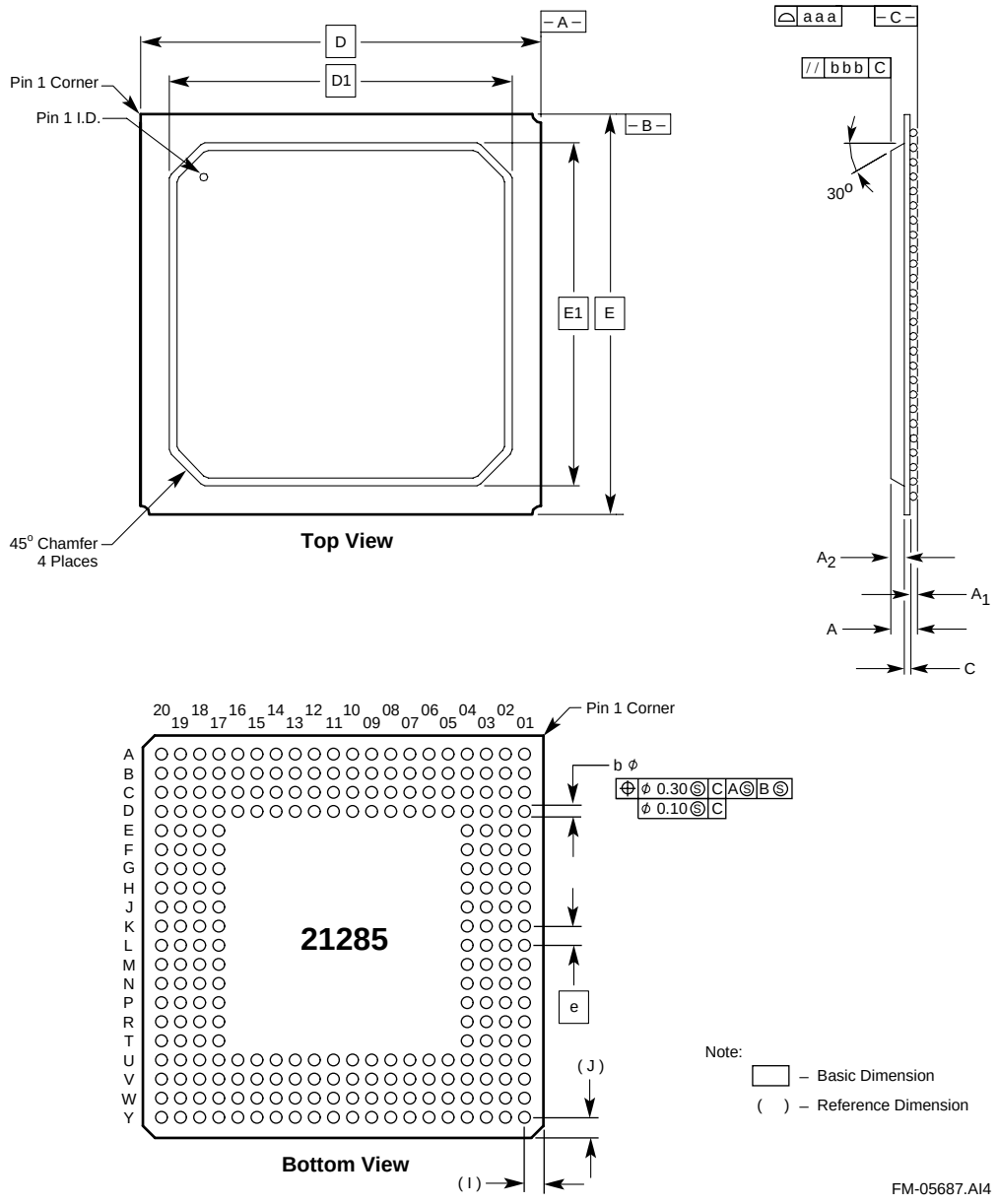


Table 10-1 lists the package dimensions in millimeters

Table 10-1. 256 Plastic Ball Grid Array (PBGA) Package Dimensions

Symbol	Dimension	Minimum Value	Nominal Value	Maximum Value
e	Ball pitch	—	1.27 BSC[a]	—
A	Overall package height	—	2.15	3.50
A_1	Package standoff height	0.50	0.60	0.70
A_2	Encapsulation thickness	—	—	2.50
b	Ball diameter	0.60	0.75	0.90
c	Substrate thickness	0.10	—	2.50
aaa	Coplanarity	—	—	0.20
bbb	Overall package planarity	—	—	0.25
D	Overall package width	26.80	27.00	27.20
D1	Overall encapsulation width	—	24.00	24.70
E	Overall package width	26.80	27.00	27.20
E1	Overall encapsulation width	—	24.00	24.70
I	Location of first row (*x*-direction)	—	1.44 reference[b]	—
J	Location of first row (*y*-direction)	—	1.44 reference[b]	—

a. ANSI Y14.5M-1982 American National Standard Dimensioning and Tolerancing, Section 1.3.2, defines Basic Dimension (BSC) as: A numerical value used to describe the theoretically exact size, profile, orientation, or location of a feature or datum target. It is the basis from which permissible variations are established by tolerances on other dimensions, in notes, or in feature control frames.

b. The value for this measurement is for reference only.

Support, Products, and Documentation

If you need technical support, a *Product Catalog*, or help deciding which documentation best meets your needs, visit the Intel World Wide Web Internet site:
`http://www.intel.com`

Copies of documents that have an ordering number and are referenced in this document, or other Intel literature may be obtained by calling **1-800-332-2717** or by visiting Intel's website for developers at:
`http://developer.intel.com`

You can also contact the Intel Massachusetts Information Line or the Intel Massachusetts Customer Technology Center. Please use the following information lines for support:

For documentation and general information:	
Intel Massachusetts Information Line	
United States:	1–800–332–2717
Outside United States:	1–303-675-2148
Electronic mail address:	techdoc@intel.com

For technical support:	
Intel Massachusetts Customer Technology Center	
Phone (U.S. and international):	1–978–568–7474
Fax:	1–978–568–6698
Electronic mail address:	techsup@intel.com