Bridge ASIC Specification

Revision 4.0

for Rev D Si

Do not copy or distribute this document

Steven Miller

Ching Hu

Preface

This document specifies the functionality and operation of the Bridge ASIC. In the course of developing this specification, many people provided valuable assistance to us. We would like to take this opportunity to thank them for the help that contributes to the creation of this specification.

The members of the Bridge Team: Anan Nagarajan, Tony Radi, Richard Frias, Scott Asakawa, Shyamoli Banerjee and Kishore Gottimukkala.

The members of the *Godzilla* Team: Dave Olson, Kevin Meier, James Tornes, Ross Werner, and Eric Linstadt for their critical review.

The members of the *Lego* Team: Ron Kolb, Chuck Narad, and John Burger for their insight on large system I/O requirements.

CHAPTER 1

Overview

1.1 Part Name

Part Name: Bridge

SGI Part Number:

Vendor: LSI Logic

Vendor Part Number:

Process Design Technology: 500K CMOS 2-Layer Metal

Package: 320 BGA

Die Size: 14.00 x 14.00 (l500685C)

Estimate Gate Count: 230K

1.2 System Architecture Overview

The *Bridge* ASIC is a controller which provides bus protocol conversion between the *Crosstalk* interconnect and either the industry standard PCI or SGI GIO buses. The *Bridge* ASIC provides additions features like address mapping, interrupt control, and prefetching.

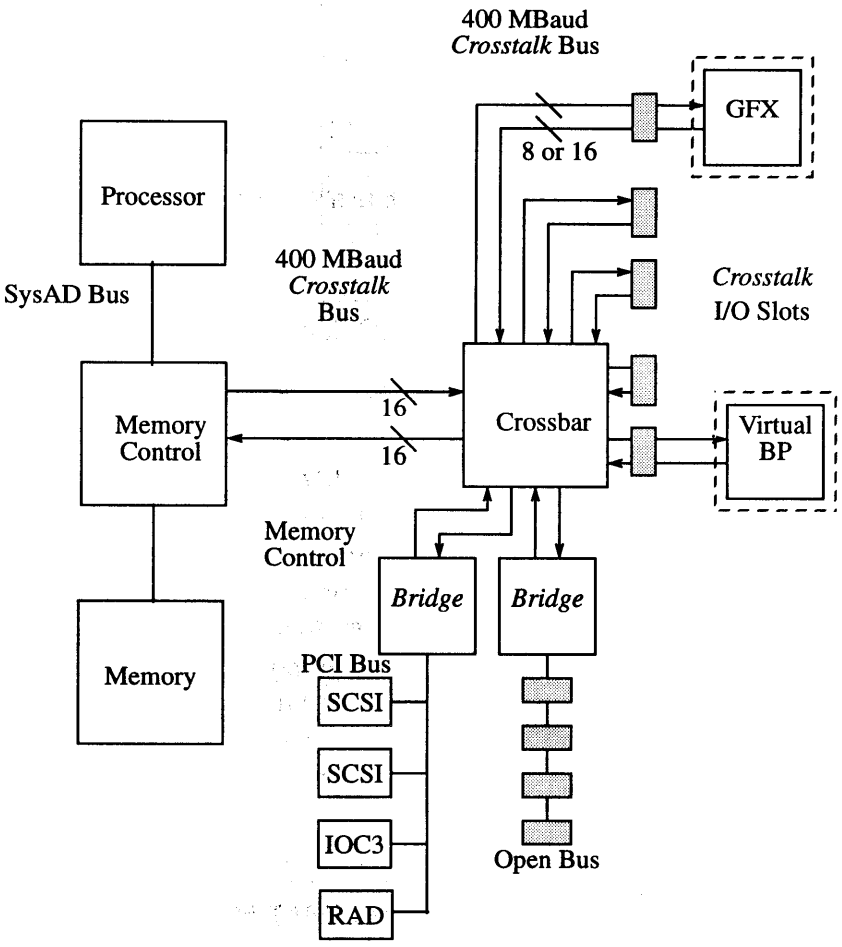


Figure 1 *Godzilla* Block Diagram

The *Bridge* ASIC is a member of the Godzilla architecture family. Figure 1 contains a block diagram for a typical system using the *Godzilla* architecture. For a complete description of the *Godzilla* architecture please refer to the *Godzilla* ***Architecture Specification***. The *Bridge* ASIC can be used to support core devices on a PCI bus, support expansion with option card slots on the industry standard PCI bus, or support legacy options with the GIO bus. A system may contain many *Bridge* ASICs to perform required functions. For this reason the *Bridge* ASIC is the most cost sensitive of the core *Godzilla* ASICs.

Shown in Figure 2 is a block diagram of the *Bridge* ASIC. The *Bridge* also contains a JTAG controller for boundary scan capability.

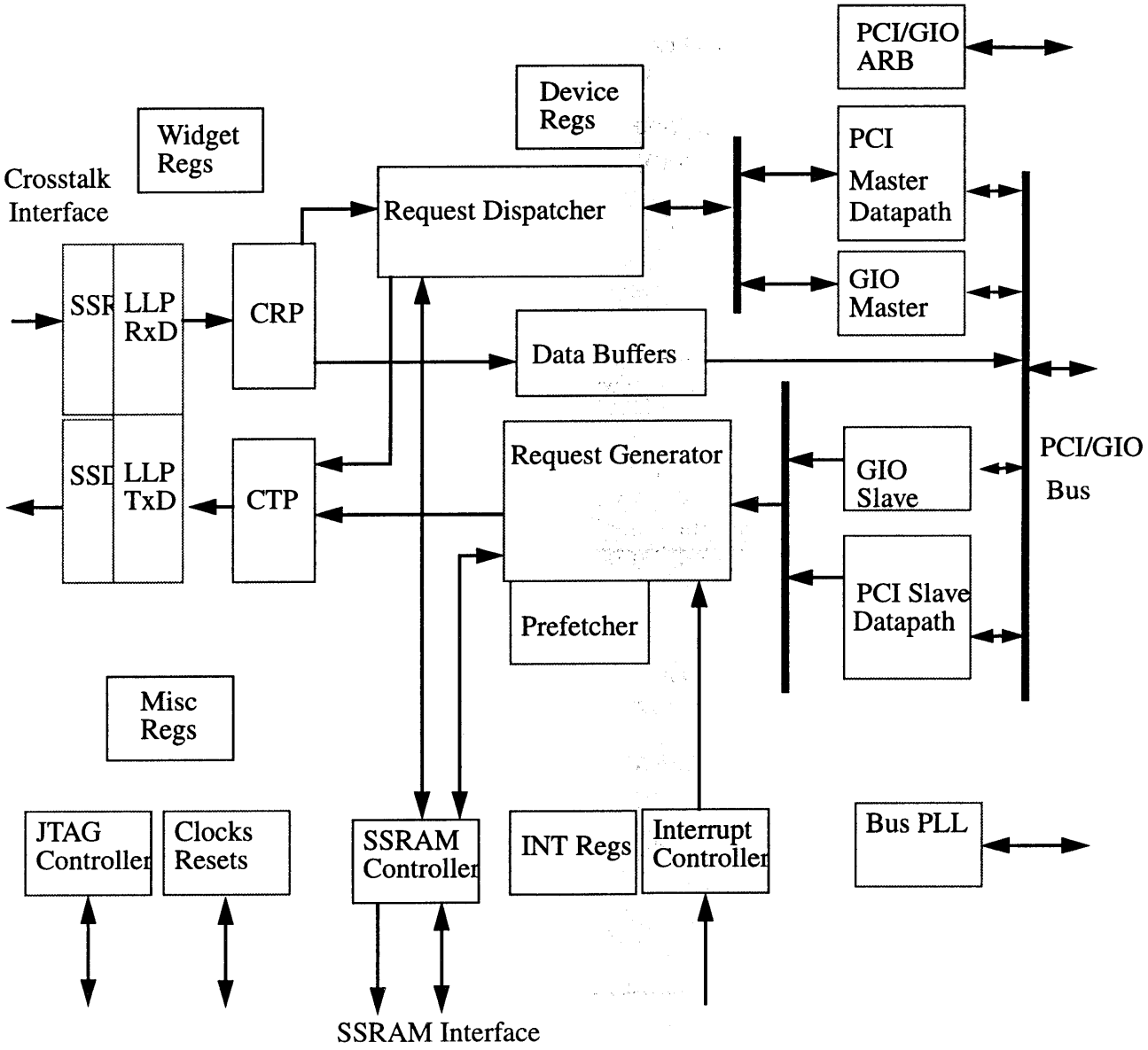


Figure 2 *Bridge* ASIC Block Diagram

1.3 *Crosstalk* Receive and Transmit Processors

The *Bridge* ASIC communicates to the system through an 8-bit *Crosstalk* interconnect link. An 8-bit *Crosstalk* interconnect link consists of 2 unidirectional source synchronous 8-bit data paths. The *Crosstalk* receive and transmit processors provide link control and buffering of packets sent and received across the link. The Crosstalk Receive Processor (CRP) receives packets and transfers them to either the data buffers or the request dispatcher. The Crosstalk Transmit Processor (CTP) gathers responses and bus generated requests and transmits them to other widgets.

1.4 PCI Bus (PCI Master & Slave)

The PCI Local Bus is a 32-bit or 64-bit bus with multiplexed address and data lines. The synchronous bus can operate at a speed up to 33 MHz in burst mode which provides a very high host/memory to peripheral transfer rate. The bus is processor independent and devices are configurable by the host. PCI devices can act as a bus master to transfer data to and from the host or they can access other local PCI devices or devices on another side of the Crosstalk Bus. Supporting PCI Bus allows SGI to have a large number of third party high performance, low cost peripheral devices to choose from to provide most functionalties and best I/O throughput while the I/O bus can last for several generations without a need to change. The *Bridge* supports eight PCI devices. (Due to PCI loading restrictions, the *Bridge* can support only four add-in cards.)

1.5 GIO Bus

GIO Bus is very similar to PCI Bus. They both have multiplexed address/data buses, device dedicated bus request/grant lines, and both support burst transfer mode. The *Bridge* supports only 64-bit pipelined GIO Bus. The *Bridge* functions as both master and slave to the GIO Bus.

1.6 Request Dispatcher

The request dispatcher decodes and distributes all incoming requests to the different functional units. It is also responsible for returning the response from those requests, providing any address translation, and error checking. *Crosstalk* to PCI/GIO bus translation mechanism consists of

generating a PCI/GIO bus address from the *Crosstalk* address. This translation occurs using fixed regions defined in widget slot space and auxiliary I/O space.

1.7 Request Generator and Prefetcher

The request generator is responsible for crosstalk request packet generation, address translation, and response buffer management. The request generator translation mechanism uses two mapping techniques, a direct map scheme for a portion of system memory and a page mapped scheme for the rest. The direct map scheme uses internal registers and predefined areas to perform mapping. The page map scheme uses a Page Mapping Unit (PMU) to perform address translation on a page basis.

To speed up the read access performed by the PCI/GIO devices to data residing across the *Crosstalk* Bus, the Prefetcher circuit will decide whether to prefetch more data following current address. This enables faster read response time when sequential reads are performed by the PCI/GIO I/O devices.

1.8 Data Buffers

Data Buffers provide speed and data buffering between *Crosstalk* Interconnect and PCI/GIO circuit. It off loads read/write packets from high speed, non-stallable *Crosstalk* Bus to the lower performance, two-way handshaking PCI/GIO buses. When the command is from PCI/GIO to *Crosstalk*, address and data are first gathered at Data Buffers before they are read by the PCI/GIO PMU.

1.9 Interrupts

Eight external active low interrupt pins from PCI/GIO Bus are connected to the *Bridge*. The assignment of the pins is not predefined. All interrupt drivers should be open drain or open collector so that multiple devices can drive the same *Bridge* pin or one device can have multiple interrupt pins to report different cases. Any time there is a change on an interrupt pin, *Bridge* reports that to the host. (Reporting the low to high transition of the active low pins can be individually disabled.) The *Bridge* will also

send interrupts to the host when some abnormal conditions occur on the Crosstalk Bus or PCI/GIO Bus.

The reporting mechanism of interrupts by the *Bridge* is to send a double word write packet (with two low significant bytes defined) to the host. Each interrupt condition is individually enabled or disabled by the host. The *Bridge* reports all the interrupt activities to the host while they are enabled. The *Bridge* does not maintain local interrupt priority and no interrupt condition can prevent the reporting of other interrupts.

1.10 SSRAM/FLASH Controller

The SSRAM/FLASH controller supports local SSRAM used by the PMU for storage of address translation entries and FLASH for boot support. The controller supports SSRAM memory configurations of 32K x 16/18, 64K x 16/18 and 256Kx 16/18. In the maximum configuration the SSRAM can hold enough entries to map a 1GByte area of system space. The SSRAM is not accessible from the PCI/GIO bus. The Flash supports configurations up to 2M x 16.

1.11 Related Documents

- *Crosstalk* Bus Specification, Revision 1.4,
  by Steven Miller, and James Tornes
- PCI Local Bus Specification, Revision 2.1
- GIO Bus Specification, Version 2.1,
  by Silicon Graphics
- Link Level Protocol Specification
  by Mike Galles
- *Godzilla* Architecture Specification
  by Karim Abdalla, Steven Miller, James Tornes, Ross Werner, and Daniel Yau
- *HEART* Specification
  by Rick Jeng, Karim Abdalla, Daniel Yau, and James Tornes
- *Crossbow* Specification, Revision 11.0
  by Pravin Shah, Rob Martin
- Speed Racer Product Specification, Revision 2.0

by Ross Werner

1.12 Terms

It is assumed the reader has read the Crosstalk Bus specification and has thorough knowledge of the Crosstalk Bus protocol, packet formats, and electrical characteristics. The following terms are discussed in the Crosstalk Bus specification and used throughout this document:

WIDGET Any ASIC connected to the Crossbar ASIC with *Crosstalk* interconnect links.

TARGET The destination widget of a Crosstalk packet.

INITIATOR The source widget of a packet.

LINK The physical connection from the Bridge ASIC to any Crossbar.

CROSSTALK PACKET

The minimum unit of data transfer on the Crosstalk bus. Data transfer on the Crosstalk Bus occurs only in several fixed data packet sizes ranging from a double word value (8 bytes), to a full cache line (128 bytes). plus header.

CROSSBAR Central routing agent for packets.

1.13 Signal Names

Signal names that end in **_N** denote signals that are active low. The term "assert" means to drive to an active state, "de-assert" means to drive to an inactive state. Buses are denoted by **BUSNAME(MSB:LSB).**

1.14 Programmers Notes

Throughout the text in the following sections are programmers notes shown in italics to call attention to specific functions or frequently asked questions.

CHAPTER 2

Pin Descriptions

2.1 Pin Descriptions

2.1.1 *Crosstalk* Interface Pins

Following are the Crosstalk Bus pins interface between Crossbow and Bridge.

Names	I/O	Types	Description
C_CLK C_CLK_N	I	Diff STL	***Crossbar Generated Clock*** used when packets are transferred from the crossbar to the *Bridge* widget.
C_CD(9:0)	I	STL	***Crossbar*** *Crosstalk* ***Data <9:0>*** Micro packet data from crossbar to *Bridge*.{synchronous to C_CLK}
C_DRDY_N	I	STL	***Crossbar Data Ready*** indicating the start valid micro packet on link. {synchronous to C_CLK}
D_CLK D_CLK_N	O	Diff STL	***Bridge Generated Clock*** used when packets are transferred from the Bridge to the crossbar.

Table 1 **Crosstalk Interface Pins**

Names	I/O	Types	Description
D_CD(9:0)	O	STL	***Bridge** Crosstalk **Data <9:0>*** Micro packet data from Bridge to the crossbar.{synchronous to D_CLK}
D_DRDY_N	O	STL	***Bridge Data Ready*** indicating the start valid micro packet on link. {synchronous to D_CLK}
X_VREF	I	STL	*Crosstalk **STL Voltage Reference***

Table 1 Crosstalk Interface Pins

2.1.2 PCI/GIO Interface Pins

Bridge supports both PCI Bus protocol and GIO Bus protocol. Most the following pins used in PCI Bus are also used in GIO Bus.

Names	I/O	Types	Description
PCI_C_BE(7:0)_N	I/O	PCI I/O	***PCI Bus Command/Byte_Enables*** indicates the PCI transfer command and then byte enables during transfer.
PCI_PAR	I/O	PCI I/O	***Parity*** across PC_AD(31:0) and PCI_C_BE(3:0)_N
PCI_PAR64	I/O	PCI I/O	***Parity*** across PC_AD(63:32) and PCI_C_BE(7:4)_N
PCI_FRAME_N	I/O	PCI I/O	***Frame*** is driven by current PCI bus master to indicate the duration of a cycle
PCI_IRDY_N	I/O	PCI I/O	***Initiator Ready*** indicates the initiator is able to complete the current data phase. This signal is driven to high before it is tri-stated.
PCI_TRDY_N	I/O	PCI I/O	***Target Ready*** indicates the target is able to complete the current data phase. This signal is driven to high before it is tri-stated.

Table 2 PCI/GIO Bus Pins

Names	I/O	Types	Description
PCI_STOP_N	I/O	PCI I/O	***Stop*** indicates the current target is requesting the master to stop the current transaction. This signal is driven to high before it is tri-stated.
PCI_DEVSEL_N	I/O	PCI I/O	***Device Select***, when asserted, indicates the driving device has decoded the address as the target. This signal is driven to high before it is tri-stated.
PCI_PERR_N	I/O	PCI I/O	***Parity Error*** reports only data parity error on the bus. This signal is driven to high before it is tri-stated.
PCI_SERR_N	I/O	PCI I/O	***System Error*** is for reporting address parity error, and other system errors.
PCI_REQ64_N	I/O	PCI I/O	***Request 64-bit transfer*** indicates the master desires to transfer data using 64 bits. This signal is driven to high before it is tri-stated.
PCI_ACK64_N	I/O	PCI I/O	***Acknowledge 64-bit transfer*** indicates the targeted device can transfer data in 64 bits. This signal is driven to high before it is tri-stated.
GIO_AS_N	I/O	CMOS	***Address Strobe***. GIO_AS_N is driven by GIO master on to indicate transfer in progress.
GIO_READ	I/O	CMOS	***GIO Bus READ*** signal to indicates read or write of the cycle and the activity of the cycle.
GIO_MEMDLY	O	CMOS	The flopped version of ***GIO_MASDLY*** in pipelined GIO mode. Always driven by Bridge.
GIO_GRXDLY	I	CMOS	***Graphics Delay*** from pipelined GIO device.
BUS_CLKIN	I	PCI I/O	***Bus clock*** input
BUS_AD(63:0)	I/O	PCI I/O	***Bus Address/Data*** bus
BUS_REQ(7:0)_N	I	PCI I/O	***Request*** indicates to the arbiter that this agent desires use of the bus. Every device has its own request signal.

Table 2 PCI/GIO Bus Pins

Names	I/O	Types	Description
BUS_GNT(7:0)_N	O	PCI I/O	***Grant*** from the Bridge indicates to the agent that access to the bus has been granted.
BUS_INT(7:0)_N	I	PCI I/O	***Interrupt*** Request signals from the devices.
BUS_RSTN(3:0)	O	PCI I/O	***PCI Bus reset*** signals.

Table 2 PCI/GIO Bus Pins

2.1.3 SSRAM/FLASH Pins

Below are the interface pins for the local SSRAM/FLASH.

Names	I/O	Types	Description
SSRAM_CLK	O	CMOS	***66MHz clock*** out for SSRAM. This clock is generated internally from the PCI clock
SSRAM_A(20:0)	O	CMOS	***SSRAM/FLASH Address*** Lines 0 through 20
SSRAM_OE_N	O	CMOS	***SSRAM Output Enable***
SSRAM_WE_N	O	CMOS	***SSRAM/FLASH Write Enable***
SSRAM_ADSP_N	O	CMOS	***SSRAM Address Status***
SSRAM_ADV_N	O	CMOS	***SSRAM Address Advance***
FLASH_OE_N	O	CMOS	***Flash Output Enable***
FLASH_CS(1:0)_N	O	CMOS	***FLASH Chip Enables***
SSRAM_D(15:0)	I/O	CMOS	***SSRAM/FLASH Data*** bits 0 through 15
SSRAM_DP(1:0)	I/O	CMOS	***SSRAM Data Parity*** Bits. Bit 1 corresponds to data 15:8, Bit 0 corresponds to data 7:0

Table 3 SSRAM/FLASH Pins

2.1.4 Miscellaneous Pins

Here are some pins used to define the operating mode of the Bridge and by PLL's.

Names	I/O	Types	Description
PCI_GION	I	CMOS	***PCI/GIO*** is a hard wired strapped pin. Bridge operates in PCI mode when it is 1 or in GIO mode when it is 0.
FLASH_SELECT	I	CMOS	***Flash Select*** is a hard wired strapped pin. Indicates which FLASH chip select should operate from address 0xc0_0000. 0 = CS0 ; 1 = CS1
PLL_RST_N	I	CMOS	PLL reset in pin, discharges the vco.
LLP_RST_N	I	CMOS	Cold power on reset to LLP
SIM_RST_N	I	CMOS	Simulation reset
QUICK_BOOT_N	I	CMOS	Simulation quick boot to LLP
NIC_IO	I/O	CMOS	Number In a Can data pin
LLP_CLK LLP_CLK_N	I	PECL	400 MHz PECL clock input for LLP
CLK_50	O	CMOS	50 MHz clock output derived from 400MHz
CLK_33	O	CMOS	33 MHz clock output derived from 400MHz
CLK_25	O	CMOS	25 MHz clock output derived from 400MHz
CLK_20	O	CMOS	20 MHz clock output derived from 400MHz
BUS_PLL_LP2	O	ANALOG	PCI Bus PLL VCO input
BUS_PLL_VSS	I	ANALOG	PCI Bus PLL ground
BUS_PLL_VDD	I	ANALOG	PCI Bus PLL power
BUS_PLL_AGND	I	ANALOG	PCI Bus PLL analog ground
VDD	I		ASIC power pins

Table 4 **Misc Pins**

Names	I/O	Types	Description
VSS	I		ASIC ground pins

Table 4 Misc Pins

2.1.5 Test Pins

Here are the Bridge test pins for ATPG and JTAG.

Names	I/O	Types	Description
TP(2:0)	I	CMOS	Test mode 000: Normal operation with PLL 001: ATPG SCAN 010: Normal operation without PLL 011: JTAG SCAN 100: PLL test mode 1 101: PLL test mode 2 110: PLL test mode 3 111: PLL test mode 4
JTDI	I	CMOS	JTAG Test Data Input
JTDO	O	CMOS	JTAG Test Data Output
JTMS	I	CMOS	JTAG test mode
JTCK	I	CMOS	JTAG Test Clock input
ENTEI	I	CMOS	Test Input
TRST_N	I	CMOS	Test Reset
SCAN_OUT2	O	CMOS	Secondary Scan Output
PNAND	O	CMOS	Nand Tree Output
TESTOUT(3:0)	O	CMOS	Test Output (See Test Pin Control Register Section 3.2.15)

Table 5 Test Pins

CHAPTER 3

Bridge Internal Registers

This section contains the register definitions for the various sections of the *Bridge* ASIC. All reserved bits within valid registers are driven as 0. Writing to undefined bits in a valid register has no effect. Any access to undefined address areas generates an address error. All the register descriptions contain a reset value field. An X in this field indicate a undefined state after reset.

3.1 Internal Register Address Map

The internal registers in the *Bridge* reside in a *Crosstalk* address region known as "widget space". Every widget has a 16MByte widget space located in system address space based on widget id number. The *Bridge* ASIC can occupy any widget id number from 0x2 through 0xf. The addresses in the Table 6 are *Crosstalk* addresses.

All registers in the *Bridge* are 32-bits or less in size and are aligned to a double-word boundary. The registers are located on data bits 31:0 of the double word. The registers can be accessed by *Crosstalk* double-word packet type with 4 data enables only (32-bit load/store only). Access by other packet type results in an address error exception.

Registers	Double Word Addresses	Data Enables	Remarks
Widget Configuration			
Bridge Identification Register	0x0000_0000	0x0F	read-only
Bridge Status Register	0x0000_0008	0x0F	read-only
Bridge Error Upper Address Register	0x0000_0010	0x0F	read-only
Bridge Error Lower Address Register	0x0000_0018	0x0F	read-only
Bridge Control Register	0x0000_0020	0x0F	read/write
Bridge Request Time-out Value Register	0x0000_0028	0x0F	read/write
Bridge Interrupt Destination Upper Address Register	0x0000_0030	0x0F	read/write
Bridge Interrupt Destination Lower Address Register	0x0000_0038	0x0F	read/write
Bridge Error Command Word Register	0x0000_0040	0x0F	read-only
Bridge LLP Configuration Register	0x0000_0048	0x0F	read/write
Bridge Target Flush Register	0x0000_0050	0x0F	read-only
Aux Error Command Word Register	0x0000_0058	0x0F	read-only
Response Buffer Upper Address Register	0x0000_0060	0x0F	read-only
Response Buffer Lower Address Register	0x0000_0068	0x0F	read-only
Test Pin Control Register	0x0000_0070	0x0F	read/write
PMU & Map			
Direct Map Register	0x0000_0080	0x0F	read/write
SSRAM			
SSRAM Parity Error Register	0x0000_0090	0x0F	read-only
Arbitration			

Table 6 *Bridge* **Register Address Map**

Registers	Double Word Addresses	Data Enables	Remarks
Arbitration Priority Register	0x0000_00A0	0x0F	read/write
Number In A Can			
NIC Register	0x0000_00B0	0x0F	read/write
PCI/GIO			
PCI/GIO Time-out Register	0x0000_00C0	0x0F	read/write
PCI Type 1 Configuration Register	0x0000_00C8	0x0F	read/write
PCI/GIO Error Upper Address Register	0x0000_00D0	0x0F	read only
PCI/GIO Error Lower Address Register	0x0000_00D8	0x0F	read only
Interrupt			
Bridge Interrupt Status Register	0x0000_0100	0x0F	read only
Bridge Interrupt Enable Register	0x0000_0108	0x0F	read/write
Reset Interrupt Status Register	0x0000_0110	0x0F	write only
Interrupt Mode Register	0x0000_0118	0x0F	read/write
Interrupt Device Register	0x0000_0120	0x0F	read/write
Host Error Field Register	0x0000_0128	0x0F	read/write
Interrupt 0 Host Address Register	0x0000_0130	0x0F	read/write
Interrupt 1Host Address Register	0x0000_0138	0x0F	read/write
Interrupt 2 Host Address Register	0x0000_0140	0x0F	read/write
Interrupt 3 Host Address Register	0x0000_0148	0x0F	read/write
Interrupt 4 Host Address Register	0x0000_0150	0x0F	read/write
Interrupt 5 Host Address Register	0x0000_0158	0x0F	read/write
Interrupt 6 Host Address Register	0x0000_0160	0x0F	read/write
Interrupt 7 Host Address Register	0x0000_0168	0x0F	read/write

Table 6 *Bridge* **Register Address Map**

Registers	Double Word Addresses	Data Enables	Remarks
Device			
Device 0 Register	0x0000_0200	0x0F	read/write
Device 1 Register	0x0000_0208	0x0F	read/write
Device 2 Register	0x0000_0210	0x0F	read/write
Device 3 Register	0x0000_0218	0x0F	read/write
Device 4 Register	0x0000_0220	0x0F	read/write
Device 5 Register	0x0000_0228	0x0F	read/write
Device 6 Register	0x0000_0230	0x0F	read/write
Device 7 Register	0x0000_0238	0x0F	read/write
Device 0 Write Request Buffer Register	0x0000_0240	0x0F	read only
Device 1 Write Request Buffer Register	0x0000_0248	0x0F	read only
Device 2 Write Request Buffer Register	0x0000_0250	0x0F	read only
Device 3 Write Request Buffer Register	0x0000_0258	0x0F	read only
Device 4 Write Request Buffer Register	0x0000_0260	0x0F	read only
Device 5 Write Request Buffer Register	0x0000_0268	0x0F	read only
Device 6 Write Request Buffer Register	0x0000_0270	0x0F	read only
Device 7 Write Request Buffer Register	0x0000_0278	0x0F	read only
Even Device Response Buffer Register	0x0000_0280	0x0F	read/write
Odd Device Response Buffer Register	0x0000_0288	0x0F	read/write
Read Response Buffer Status Register	0x0000_0290	0x0F	read only
Read Response Buffer Clear Register	0x0000_0298	0x0F	write only

Table 6 *Bridge* **Register Address Map**

3.2 *Crosstalk* Registers

Crosstalk registers are those registers which are associated with the *Crosstalk* interconnect. Listed below are the *Crosstalk* registers:

- *Bridge* Identification Register
- *Bridge* Status Register
- *Bridge* Error Upper Address Register
- *Bridge* Error Lower Address Register
- *Bridge* Control Register
- *Bridge* Request Time-Out Value Register
- *Bridge* Interrupt Destination Upper Address Register
- *Bridge* Interrupt Destination Lower Address Register
- *Bridge* Error Command Word Register
- *Bridge* Target Flush Register

3.2.1 *Bridge* Identification Register

The *Bridge* identification register is a read only register used by the host CPU during configuration to determine the type of the widget. The format is the same as defined in IEEE 1149.1 JTAG Device Identification Register.

Name	Bits	Value	Description
REV_NUM	31:28	Current Rev	Revision Number current revision of the widget starting at 1. Marking is alpha, 1=A , 2=B , 3=C...
PART_NUM	27:12	0xC002	Part Number "Widget Type"
MFG_NUM	11:1	00000110110	Manufacturer Identity
	0	1	Always read as 1

Table 7 *Bridge* **Identification Register**

3.2.2 *Bridge* Status Register

The status register is a read register which holds status information of the *Bridge*.

Name	Bits	Reset Value	Definition
LLP_REC_CNT	31:24	0	LLP receive retry counter
LLP_TX_CNT	23:16	0	LLP transmit retry counter
	15:7		Reserved
FLASH_SELECT	6	X	A hard wired strapped pin. Indicates which FLASH chip select should operate from address 0xc0_0000. 0 = cs0 ; 1 = cs1
PCI_GIO_N	5	X	This read only bit tells host what mode the Bridge operates. PCI (=1) or GIO (=0).
PENDING	4:0	0	Pending Requests indicates the number of current outstanding read requests.

Table 8 *Bridge* **Status Register**

3.2.3 *Bridge* Error Upper Address

The *Bridge* error upper address register is a read only register which contains the upper 16-bits of the address when any error occurs. Subsequent errors are not logged until the *Bridge* error is cleared. The last logged value is held until the group is cleared and enabled.

Name	Bits	Reset Value	Description
	31:16		Reserved
UPP_ADDR	15:0	X	Address Bits 47:32

Table 9 *Bridge* **Error Upper Address Register**

3.2.4 *Bridge* Error Lower Address

The *Bridge* error lower address register is a read only register which contains the lower 32-bits of the address when any error occurs. Subsequent

errors are not logged until the *Bridge* error is cleared. The last logged value is held until the group is cleared and enabled.

Name	Bits	Reset Value	Description
LOW_ADDR	31:0	X	Address Bits 31:0

Table 10 *Bridge* **Error Lower Address Register**

3.2.5 *Bridge* Control Register

The control register is a read/write register which holds control information for the *Bridge*.

Name	Bits	Reset Value	Definition
FLASH_WR_EN	31	0	Flash Write Enable when 1 enables writing to the Flash device. When 0 an error occurs on the write and the write is not completed.
EN_CLK50	30	1	50 MHz clock out enable 1=> enabled. This clock is generated from the 400 MHz ECL clock and driven out and is asynchronous to any internal clocks.
EN_CLK40	29	1	40 MHz clock out enable 1=> enabled. This clock is generated from the 400 MHz ECL clock and driven out and is asynchronous to any internal clocks.
EN_CLK33	28	1	33 MHz clock out enable 1=> enabled. This clock is generated from the 400 MHz ECL clock and driven out and is asynchronous to any internal clocks.
RST_PIN_N(3:0)	27:24	0xF	Software programmable reset pins. A "1" provides a high level on the rst_pin_n().

Table 11 *Bridge* **Control Register**

Name	Bits	Reset Value	Definition
IO_SWAP	23	0	IO Enable Swapping 1 => enabled. When enabled the byte swapper exchanges data per Appendix A for access above widget space to the PCI IO space.
MEM_SWAP	22	0	Memory Enable Swapping 1 => enabled. When enabled the byte swapper exchanges data per Appendix A for access above widget space to the PCI memory or GIO space.
PAGE_SIZE	21	0	Page Size 0 = > 4K 1 => 16K.
SS_PAR_BAD	20	0	Force Bad SSRAM Parity 0 = Normal Parity 1= Bad Parity
SS_PAR_EN	19	0	SSRAM Parity Enable 0 = Parity disabled 1= Parity enabled
SSRAM_SIZE	18:17	00	Local SSRAM Size Bits 00 = 0K 01 = 32K x 16/18 10 = 64K x 16/18 11 = 256K x 16/18
F_BAD_PKT	16	0	Force Bad LLP Micro-Packet Enable a "1" on this bit enables generation of a bad LLP micro packet each time a new micro-packet is sent. The retry will generate a valid packet.
LLP_XBAR_CRD	15:12	3	LLP Crossbar Credit This four bit value is used to determine the maximum number of packet buffers the crossbar has.
CLR_RLLP_CNT	11	0	Clear Receive LLP Retry Counter this bit when written with 1 clears the LLP receive retry counter. Always read as 0.
CLR_TLLP_CNT	10	0	Clear Transmit LLP Retry Counter this bit when written with 1 clears the LLP transmit counter. Always read as 0.
SYS_END	9	1	System Endianess 1 = Big, 0 = Little

Table 11 *Bridge* **Control Register**

Name	Bits	Reset Value	Definition
MAX_TRANS	8:4	0x1F	Transnum Number of outstanding transactions allowed.
WIDGET_ID	3:0	0xF	Widget ID number.

Table 11 *Bridge* **Control Register**

3.2.6 *Bridge* Request Time-out Value Register

This register contains the reload value for the response timer. The request timer counts every 960 nS (32 PCI clocks)

Name	Bits	Reset Value	Description
	31:20		Reserved
TIME_OUT	19:0	0xFFFFF	Reload Value for the response time-out counter

Table 12 *Bridge* **Request Time-out Value Register**

3.2.7 *Bridge* Interrupt Destination Upper Address Register

The *Bridge* interrupt destination upper address register is a read/write register containing the upper 16-bits of address of the host to which the interrupt is targeted. In addition the target ID is also contained in this register.

Name	Bits	Reset Value	Description
	31:20		Reserved
TARGET_ID	19:16	X	Target ID Number for error interrupt host
UPP_ADDR	15:0	X	Address Bits 47:32

Table 13 *Bridge* **Interrupt Destination Upper Address Register**

3.2.8 *Bridge* Interrupt Destination Lower Address Register

The *Bridge* interrupt destination lower address register is a read/write register which contains the lower 32-bits of the address of the host to which the interrupt is targeted.

Name	Bits	Reset Value	Description
LOW_ADDR	31:0	X	Address Bits 31:0

Table 14 ***Bridge*** **Interrupt Destination Lower Address Register**

3.2.9 *Bridge* Error Command Word Register

The *Bridge* error command word is a read register that holds the command word of a *Crosstalk* packet when errors occur. Errors are indicated with error bits in the interrupt status register. Subsequent errors are not logged until the interrupt is cleared.

Name	Bits	Reset Value	Definition
DIDN	31:28	X	Destination ID Number
SIDN	27:24	X	Source ID Number
PACTYP	23:20	X	Packet Type
TNUM	19:15	X	Transaction number
COHERENT	14	X	Coherent Transaction
DS	13:12	X	Data Size
GBR	11	X	Guaranteed Bandwidth Ring enable
VBPM	10	X	Virtual Backplane Message
ERROR	9	X	Error Occurred
BARRIER	8	X	Barrier
	7:0		Reserved

Table 15 *Bridge* **Error Command Word Register**

3.2.10 *Bridge* LLP Configuration Register

This register contains the configuration information for the LLP modules used in crosstalk interconnect.

Name	Bits	Reset Value	Description
	31:26		Reserved
LLP_MAXRETRY	25:16	0x3FF	LLP Max retry count.
LLP_NULLTIMEOUT	15:10	0x06	Null Time-out value
LLP_MAXBURST	9:0	0x010	LLP Max burst count.

Table 16 *Bridge* **LLP Configuration Register**

3.2.11 *Bridge* Target Flush Register

When read, this register will return a 0x00 after all previous transfers to the *Bridge* have completed.

3.2.12 Aux Error Command Word Register

The Aux error command word is a read-only register that holds the command word of a *Crosstalk* packet when request fifo overflow or unexpected response errors occur. Errors are indicated with error bits in the interrupt status register. Subsequent errors are not logged until this interrupt is cleared.

Name	Bits	Reset Value	Definition
DIDN	31:28	X	Destination ID Number
SIDN	27:24	X	Source ID Number
PACTYP	23:20	X	Packet Type
TNUM	19:15	X	Transaction number
COHERENT	14	X	Coherent Transaction

Table 17 **Aux Error Command Word Register**

Name	Bits	Reset Value	Definition
DS	13:12	X	Data Size
GBR	11	X	Guaranteed Bandwidth Ring enable
VBPM	10	X	Virtual Backplane Message
ERROR	9	X	Error Occurred
BARRIER	8	X	Barrier
	7:0		Reserved

Table 17 Aux Error Command Word Register

3.2.13 Response Buffer Error Upper Address

The response buffer error upper address register is a read only register which contains the upper 16-bits of the address when error associated with response buffer entries occur. Subsequent errors are not logged until the interrupt is cleared.

Name	Bits	Reset Value	Description
	31:23		Reserved
DEV_NUM	22:20		Device Number
BUFF_NUM	19:16		Buffer Number
UPP_ADDR	15:0		Address Bits 47:32

Table 18 Response Buffer Error Upper Address Register

3.2.14 Response Buffer Error Lower Address

The response buffer error lower address register is a read only register which contains the lower 32-bits of the address when error associated with response buffer entries occur. Subsequent errors are not logged until the interrupt is cleared.

Name	Bits	Reset Value	Description
LOW_ADDR	31:0		Address Bits 31:0

Table 19 Response Buffer Error Lower Address Register

3.2.15 Test Pin Control Register

This register selects the output function and value to the four test pins on the bridge.

Name	Bits	Reset Value	Description
	31:12	0	Reserved
TDATA_OUT	11:8	0	Test Data Out to Pins
SEL_TPIN_3	7:6	0	Select function for test pin 3 11 => TDATA_OUT (3) 10 => 01 => llp_grp interrupt 00 = > RetryTimeout
SEL_TPIN_2	5:4	0	Select function for test pin 2 11 => TDATA_OUT (2) 10 => 01 => req_dsp_grp interrupt 00 = > BufFull from LLP8
SEL_TPIN_1	3:2	0	Select function for test pin 1 11 => TDATA_OUT (1) 10 => pci_grp interrupt 01 =>resp_buf_grp interrupt 00 = > core_clk
SEL_TPIN_0	1:0	0	Select function for test pin 0 11 => TDATA_OUT (0) 10 => ssram_grp interrupt 01 => crp_grp interrupt 00 = > internal pci_clk

Table 20 Test Pin Control Register

3.3 Mapping Registers

3.3.1 *Bridge* Direct Mapping Register

This register is used to relocate a 2 GByte region for PCI/GIO to *Crosstalk* transfers.

Name	Bits	Reset Value	Description
	31:24		Reserved
DIR_W_ID	23:20	0	Direct space target widget ID
	19		Reserved
DIR_RMF_64	18	0	Direct Mapper Remote Map Field 64 controls the concatenation of the RMF during a 64-bit address direct map operation. A '0' in this bit generates the following for the RMF: PCI_ADDR55:48 => RMF7:0 0x00 => RMF15:8 A '1' in this bit generates the following: PCI_ADDR55:40 => RMF15:0
DIR_ADD512	17	1	Direct add 512MB offset only when DIR_OFF = 0x000
DIR_OFF	16:0	0x000	Direct Map Offset

Table 21 *Bridge* **Auxiliary Space and Direct Mapping Register**

3.4 SSRAM Registers

3.4.1 SSRAM Parity Error Register

This read only register holds the address in local memory where the parity error occurred, the initiator of the transaction, and the byte in error.

Name	Bits	Reset Value	Description
	31:29		Reserved
SIZE_FAULT	28	X	SSRAM access was outside valid device address limit.
PCI_DEV	27:25	X	PCI device requesting translation
ACC_SRC	24	X	Accessing Source 0 *Crosstalk* Interconnect 1 PMU Descriptor Fetch
PAR_BYTE	23:16	X	Parity Error in Byte 7 through 0 Error = 1 Bit 23 == Data 63:56 .... Bit 16 == Data 7:0
PAR_ADD	15:0	X	16-Bits of SSRAM Address 18:3

Table 22 SSRAM Parity Error Register

3.5 Arbitration Register

This register defines the priority and bus time out timing in PCI/GIO bus arbitration.

Name	Bits	Reset Value	Description
REQ_WAIT_TICK	17:16	2	These 2 bits define the time period used by the request wait circuit used in the arbiter 00 : 240 ns 01 : 480 ns 10 : 960 ns 11 : Reserved
REQ_WAIT_EN	15:8	0	These bits enable the arbiter to delay issuing bus grants to the devices for 4 ticks. The duration of the tick is defined by REQ_WAIT_TICK. REQ_WAIT_EN[7:0] correspond to BUS_REQ_N[7:0].
	7		Reserved
FREEZE_GNT	6	0	When this bit is set by host, no grant will be issued to the PCI/GIO devices and Bridge internal circuit. All activities on the bus are turned off.
EN_BRIDGE_HI	5:3	0 0 1	Three bits to enable Bridge to request the bus in the high priority ring.
EN_BRIDGE_LO	2:0	0 0 0	Three bits to enable Bridge to request the bus in the low priority ring.

Table 23 ARB_PRIORITY Register

3.6 NIC Register

This register contains the number in a can support.

Name	Bits	Reset Value	Description
	31:20		Reserved
NIC_BMP	19:10	0	Bus Master pulse duration in uS
NIC_OFFSET	9:2	0	Sampling offset relative to the deassertion of the Bus Master strobe.
NIC_DATA_VLD	1	0	Sample Data Valid
NIC_DATA	0	X	Sample Data

Table 24 NIC Register

3.7 PCI/GIO

3.7.1 Time-out Register

This register determine bus time out timing for GIO and retry time-out and retry count for PCI bus.

Name	Bits	Reset Value	Description
PCI_RETRY_HLD	20:16	1	The PCI retry hold is the amount of time the *Bridge* waits after a retry operation to request the bus. The unit of time is the 16 PCI clocks or 480nS for a 33MHz bus. If this value is 0, Bridge will always wait at least 2 PCI clocks before retrying the operation.

Table 25 PCI/GIO Time-out Register

Name	Bits	Reset Value	Description
	15:13	0	Reserved
GIO_TIMEOUT	12	0	If a GIO slave is unresponsive to a GIO master command, this bit enables *Bridge* to issue a fake MEMDLY when 31us (1024 GIO clock) is reached in all but GIO master read to Crosstalk cases. The fake MEMDLY in Crosstalk read is generated after packet time-out.
	11:10	0	Reserved
PCI_RETRY_CNT	9:0	0	The PCI retry count determines the number of retry operations the *Bridge* will attempt as a master before taking an error exception. If this value is 0, *Bridge* will always retry the command until it completes.

Table 25 PCI/GIO Time-out Register

3.7.2 PCI Type 1 Configuration Register

This register is use during accesses to the PCI type 1 configuration space. The bits in this register are used to supplement the address during the configuration cycle to select the correct secondary bus and device.

Name	Bits	Reset Value	Description
	31:24		Reserved
BUS_NUM	23:16	0	Bus Number is an encoded value used to select 1 of 256 buses in the system.
DEV_NUM	15:11	0	Device Number is an encoded value used to select 1 of 32 devices on a given bus.
	10:0	0	Reserved

Table 26 PCI Type 1 Configuration Register

3.7.3 PCI/GIO Error Upper Address Register

This register holds the value of the upper address on the PCI/GIO Bus when an error occurs.

Name	Bits	Reset Value	Description
	31:21	X	Reserved
PCI_DEV_MASTER	20	X	This bit when one indicates that the bridge was not PCI master when the error occurred.
PCI_VDEV	19	X	This bit indicates the value of the Virtual Request.
PCI_DEV_NUM	18:16	X	This is the device number of the PCI master when the error occurred.
PCI_UADDR_ERR	15:0	X	This is the address of the current PCI/GIO bus command, PCI address 47:32.

Table 27 **PCI/GIO Error Upper Address Register**

3.7.4 PCI/GIO Error Lower Address Register

This register holds the value of the lower address on the PCI/GIO Bus when an error occurs.

Name	Bits	Reset Value	Description
PCI_LADDR_ERR	31:0	X	This is the address of the current PCI/GIO bus command, PCI address 31:0.

Table 28 **PCI/GIO Error Lower Address Register**

3.8 Interrupt Registers

3.8.1 HOST_ERR_FIELD

This register tells which bit location in the host's Interrupt Status register to set or reset when any error condition happens.

Name	Bits	Reset Value	Description
BRIDGE_ERR_FLD	7:0	X	Bit location of *Bridge* error in host interrupt register.

Table 29 HOST_ERR_FIELD register

3.8.2 INT_STATUS Register

This is the current interrupt status register which maintains the current status of all the interrupting devices. This register is read only and all the bits are active high. A high bit at INT_STATE means the corresponding INT_N pin has been asserted (low).

Name	Bits	Reset Value	Description
MULTI_ERR	31	0	This bit indicates that multiple errors occurred before the first was cleared. This bit is set for any enabled interrupt bit.
PMU_ESIZE_FAULT	30	0	This bit indicates that there was an access to the SSRAM beyond the device limits.
UNEXPECTED_RESP	29	0	This bit indicates that an unexpected response arrived.
BAD_XRESP_PACKET	28	0	Framing error, the data size in command word of a response did not match actual data size sent.

Table 30 Bridge INT_STATUS Register

Name	Bits	Reset Value	Description
BAD_XREQ_PACKET	27	0	Framing error, the data size in command word of a request did not match actual data size sent.
RESP_XTALK_ERROR	26	0	Arriving response packet had either the command word error bit set or the invalid sideband set during a data phase.
REQ_XTALK_ERROR	25	0	Arriving request packet had either the command word error bit set or the invalid sideband set during a data phase.
INVALID_ADDRESS	24	0	Request packet contains an invalid address for the bridge widget.
UNSUPPORTED_XOP	23	0	Request operation is not supported by bridge, packet format ok.
XREQ_FIFO_OFLOW	22	0	Request packet overflow: A request packet arrived with request fifo full. This indicates a credit count problem.
LLP_REC_SNERROR	21	0	LLP Receiver Sequence Number Error indicates that a retry was required on the receiver section of the LLP due to a incorrect micropacket sequence number. This error is based on LLP micropackets not *Crosstalk* packets.
LLP_REC_CBERROR	20	0	LLP Receiver Check Bit Error indicates that a retry was required on the receiver section of the LLP due to a error in the check bits of micropacket. This error is based on LLP micropackets not *Crosstalk* packets.
LLP_RCTY	19	0	LLP Receiver Retry Count Interrupt is generates when the receiver retry counter rolls from ff to 0.
LLP_TX_RETRY	18	0	LLP Transmitter Retry indicates that a retry was required on the transmitter side of the LLP. This error is based on LLP micropackets not *Crosstalk* packets.

Table 30 Bridge INT_STATUS Register

Name	Bits	Reset Value	Description
LLP_TCTY	17	0	LLP Transmitter Retry Count Interrupt is generates when the transmitter retry counter rolls from ff to 0.
SSRAM_PERR	16	0	Status of SSRAM parity error interrupt
PCI_ABORT	15	0	Status of PCI slave abort condition interrupt
PCI_PARITY	14	0	Indicates the bridge detected a parity error.
PCI_SERR	13	0	Status PCI Address/CMD parity error interrupt
PCI_PERR	12	0	Status of PCI/GIO parity error interrupt
PCI_MASTER_TOUT	11	0	Indicates a device select timeout on the PCI bus. or a GIO timeout.
PCI_RETRY_CNT	10	0	PCI retry operation count exhausted.
XREAD_REQ_TOUT	9	0	PCI to Crosstalk read request timeout.
GIO_BENABLE_ERR	8	0	GIO non-contiguous byte enable in crosstalk packet error.
INT_STATE	7:0	0	Status of INT_N[7:0]. A 1 means INT_N is low.

Table 30 Bridge INT_STATUS Register

3.8.3 INT_ENABLE Register

This register enables the reporting of interrupt to the host. Each bit in this register corresponds to the same bit in INT_STATUS register. All bits are zero after reset.

Name	Bits	Reset Value	Description
EN_PMU_ESIZE_FAULT	30	0	Enables PMU External Size Fault
EN_UNEXPECTED_RESP	29	0	Enables unexpected response interrupt.
EN_BAD_XRESP_PACKET	28	0	Enables bad Crosstalk packet interrupt.

Table 31 Bridge INT_ENABLE Register

Name	Bits	Reset Value	Description
EN_BAD_XREQ_PACKET	27	0	Enables bad Crosstalk packet interrupt.
EN_RESP_XTALK_ERROR	26	0	Enables Crosstalk error interrupt.
EN_REQ_XTALK_ERROR	25	0	Enables Crosstalk error interrupt.
EN_INVALID_ADDRESS	24	0	Enables invalid address error interrupt.
EN_UNSUPPORTED_XOP	23	0	Enables unsupported operation error.
EN_XREQ_FIFO_OFLOW	22	0	Enables Crosstalk request fifo overflow interrupt.
EN_LLP_REC_SNERROR	21	0	Enables LLP Receiver Sequence Number Error interrupt.
EN_LLP_REC_CBERROR	20	0	Enables LLP Receiver Check Bit Error interrupt.
EN_LLP_RCTY	19	0	Enables Receiver Retry Count interrupt.
EN_LLP_TX_RETRY	18	0	Enables LLP Transmitter Retry interrupt.
EN_LLP_TCTY	17	0	Enables Crosstalk LLP Transmitter Retry Count interrupt.
EN_SSRAM_PERR	16	0	Enables SSRAM parity error interrupt.
EN_PCI_ABORT	15	0	Enables PCI master/slave abort condition interrupt
EN_PCI_PARITY	14	0	Enables PCI parity error interrupt.
EN_PCI_SERR	13	0	Enables PCI Address/CMD parity error interrupt
EN_PCI_PERR	12	0	Enables PCI/GIO parity error interrupt
EN_PCI_MASTER_TOUT	11	0	Enables PCI master timeout interrupt.
EN_PCI_RETRY_CNT	10	0	Enables PCI retry count interrupt.
EN_XREAD_REQ_TOUT	9	0	Enables Crosstalk read request timeout interrupt.

Table 31 Bridge INT_ENABLE Register

Name	Bits	Reset Value	Description
GIO_BENABLE_ERR	8	0	Enables GIO non-contiguous byte enable interrupt.
EN_INT_STATE	7:0	0	Enables INT_N[7:0].

Table 31 Bridge INT_ENABLE Register

3.8.4 RESET_INT_STATUS Register

This write only register used by the host to clear the INT_STATUS register bits not associated with interrupt pins. Clearing the interrupt group re-arms the interrupt group.

Name	Bits	Reset Value	Writing a one clears the following:
MULTI_CLR	6	0	MULTI_ERR
CRP_GRP_CLR	5	0	UNEXPECTED_RESP, XREQ_FIFO_OFLOW
RESP_BUF_GRP_CLR	4	0	BAD_XRESP_PACKET, RESP_XTALK_ERROR, XREAD_REQ_TOUT
REQ_DSP_GRP_CLR	3	0	UNSUPPORTED_XOP, BAD_XREQ_PACKET, REQ_XTALK_ERROR, INVALID_ADDRESS
LLP_GRP_CLR	2	0	LLP_REC_SNERROR, LLP_REC_CBERROR, LLP_RCTY, LLP_TX_RETRY, LLP_TCTY
SSRAM_GRP_CLR	1	0	SSRAM_PERR PMU_ESIZE_FAULT

Table 32 Bridge INT_RESET Register

Name	Bits	Reset Value	Writing a one clears the following:
PCI_GRP_CLR	0	0	PCI_ABORT, PCI_PARITY, PCI_SERR, PCI_PERR, PCI_MASTER_TOUT, PCI_RETRY_CNT, GIO_BENABLE_ERR

Table 32 Bridge INT_RESET Register

3.8.5 INT_MODE Register

This register defines the interrupting mode of the INT_N pins.

Name	Bits	Reset Value	Description
EN_CLR_PKT	7:0	0	EN_CLR_PKT[i]=1 enables Bridge to send an interrupt clear packet with data[8]=0 to host interrupt status register when INT_N[i] just deasserts. CLR_PKT[7:0] correspond to INT_N[7:0].

Table 33 INT_MODE register

3.8.6 INT_DEV Register

This register associates interrupt pins with devices thus allowing buffer management (flushing) when a device interrupt occurs.

Name	Bits	Reset Value	Description
	31:24	0	Reserved
INT7_DEV	23:21	0	Contains the binary number of the device associated with interrupt 7 (INT7).

Table 34 INT_DEV register

Name	Bits	Reset Value	Description
INT6_DEV	20:18	0	Contains the binary number of the device associated with interrupt 6 (INT6).
INT5_DEV	17:15	0	Contains the binary number of the device associated with interrupt 5 (INT5).
INT4_DEV	14:12	0	Contains the binary number of the device associated with interrupt 4 (INT4).
INT3_DEV	11:9	0	Contains the binary number of the device associated with interrupt 3 (INT3).
INT2_DEV	8:6	0	Contains the binary number of the device associated with interrupt 2 (INT2).
INT1_DEV	5:3	0	Contains the binary number of the device associated with interrupt 1 (INT1).
INT0_DEV	2:0	0	Contains the binary number of the device associated with interrupt 0 (INT0).

Table 34 **INT_DEV register**

3.8.7 Interrupt (x) Host Register

This register allow different host address to be assigned to each interrupt pin and the bit in the host.

Name	Bits	Reset Value	Description
	31:18	0	Reserved
INT_ADDR	17:8	0	Contains the new section of the host address field for interrupt (x). These bits replace *Crosstalk* addresses bits 47:38 from the Interrupt Destination Upper Address Register.
INT_FLD	7:0	0	Bit location of INT_N_(x) in Host interrupt register

Table 35 **Interrupt (x) Host register**

3.9 Device Registers

There are two registers for each of the eight devices on the PCI/GIO bus, one register for read response buffer allocation and the other for general control and mapping.

3.9.1 Device (x)

Name	Bits	Reset Value	Description
	31:29		Reserved
EN_ERROR_LOCK	28	1	Enable the Error Lock feature to the pci device. This bit will lockout the device from the arbiter and hold current pending requests until the error is cleared.
EN_PAGE_CHK	27	1	Enable Prefetcher Page Cross Checking. This bit when 0 forces the prefetcher to stop at a 4k or 16k page boundary.
FORCE_PCI_PAR	26	0	Force a PCI parity error on both pci parity lines.
EN_VIRTUAL	25	0	Enable Virtual Device bit for 64-bit address master.
PMU_WRT_GEN	24	0	Page Mapped Write Gather Enable, A 1 in this bit enables write gather for device accesses in page mapped space.
DIR_WRT_GEN	23	0	Direct Mapped Write Gather Enable, A 1 in this bit enables write gather for device accesses in direct mapped space.
DEV_SIZE	22	0	Device Size 0 = 32-bit, 1 = 64-bit. This bit is programmed to select the device bus size. For use in the device windows area.

Table 36 Device (x) Registers

Name	Bits	Reset Value	Description
REAL_TIME	21	0	Real Time Device Enable 1 => enabled. This bit is programmed to select which arbitration ring the device operates in.
SWAP_PMU	20	0	Enable Swapping in PMU Map Mode 1 => enabled. When enabled the byte swapper exchanges data per Appendix A during transfers through page mapped space.
SWAP_DIRECT	19	0	Enable Swapping in Direct Map Mode 1 => enabled. When enabled the byte swapper exchanges data per Appendix A during transfers through direct mapped space.
PREFETCH	18	0	Prefetcher Enable bit provides two functions. The first is to start/continue the prefetcher operation. The second is read buffer management for the PCI/GIO bus. Normally a read buffer is invalidated after the current cycle is completed, either FRAME_N negated (PCI) or byte count exhausted (GIO). The Prefetcher Enable bit allows buffers to maintain validity until any byte in the last word of the cache line accessed. The buffer may be invalidated if the buffer management logic requires the resource for some other reason. This bit allows multiple bus transaction to a single buffer. For a complete invalidation rule see PCI/GIO Data Buffer Section. This bit is used during transfers through 32-bit direct mapped space.
PRECISE	17	0	Precise Transaction bit when clear enables all read operations to fetch a full cache line regardless of size (byte count on GIO or initial byte field on PCI) or starting address. This bit is used during 32-bit address transfers through direct mapped space.

Table 36 Device (x) Registers

Name	Bits	Reset Value	Description
COHERENT	16	0	Coherent Transaction 1 => coherent. This bit is passed to the CIU enabling the Coherent Transaction bit to be set in the *Crosstalk* packet. This bit is used during transfers through direct mapped space.
BARRIER	15	0	Barrier Transaction 1 => barrier. This bit is passed to the CIU enabling the Barrier Transaction bit to be set in the *Crosstalk* packet. This bit is used during transfers through direct mapped space.
GBR	14	0	GBR Enable 1 => enabled. This bit causes the GBR bit in *Crosstalk* packets to be set when generated from this device. This bit is used for both direct and page mapped operation.
DEV_SWAP	13	0	Device Enable Swapping 1 => enabled. When enabled the byte swapper exchanges data per Appendix A for transfers initiated by a *Crosstalk* widget.
DEV_IO_MEM	12	1	Enable Device Memory or I/O Space When in PCI bus mode, a 1 enables memory space for the device window, a 0 enables I/O space. In GIO mode this bit is ignored.
DEV_OFF	11:0	See Chapter 3	Device Offset Address Bits 31:20. These bits are used to map the device window to the PCI/GIO bus.

Table 36 Device (x) Registers

3.9.2 Device (x) Write Request Buffer Flush

When read, this register will return a 0x00 after the write buffer associated with the device has been flushed.

3.9.3 Even Device Read Response Buffer Register

This register is use to allocate the read response buffers for the even numbered devices. (0,2,4,6)

Name	Bits	Reset Value	Description
BUFF_14_EN	31	0	Enable buffer 14
BUFF_14_VDEV	30	0	Virtual device select for buffer 14.
BUFF_14_PDEV	29:28	0	Upper two bits of device number for buffer 14.
BUFF_12_EN	27	0	Enable buffer 12
BUFF_12_VDEV	26	0	Virtual device select for buffer 12.
BUFF_12_PDEV	25:24	0	Upper two bits of device number for buffer 12.
BUFF_10_EN	23	0	Enable buffer 10
BUFF_10_VDEV	22	0	Virtual device select for buffer 10.
BUFF_10_PDEV	21:20	0	Upper two bits of device number for buffer 10.
BUFF_8_EN	19	0	Enable buffer 8
BUFF_8_VDEV	18	0	Virtual device select for buffer 8.
BUFF_8_PDEV	17:16	0	Upper two bits of device number for buffer 8.
BUFF_6_EN	15	0	Enable buffer 6
BUFF_6_VDEV	14	0	Virtual device select for buffer 6.
BUFF_6_PDEV	13:12	0	Upper two bits of device number for buffer 6.
BUFF_4_EN	11	0	Enable buffer 4
BUFF_4_VDEV	10	0	Virtual device select for buffer 4.
BUFF_4_PDEV	9:8	0	Upper two bits of device number for buffer 4.
BUFF_2_EN	7	0	Enable buffer 2
BUFF_2_VDEV	6	0	Virtual device select for buffer 2.
BUFF_2_PDEV	5:4	0	Upper two bits of device number for buffer 2.

Table 37 Even Device Read Response Buffer Register

Name	Bits	Reset Value	Description
BUFF_0_EN	3	0	Enable buffer 0
BUFF_0_VDEV	2	0	Virtual device select for buffer 0.
BUFF_0_PDEV	1:0	0	Upper two bits of device number for buffer 0.

Table 37 Even Device Read Response Buffer Register

3.9.4 Odd Device Read Response Buffer Register

This register is use to allocate the read response buffers for the odd numbered devices. (1,3,5,7))

Name	Bits	Reset Value	Description
BUFF_15_EN	31	0	Enable buffer 15
BUFF_15_VDEV	30	0	Virtual device select for buffer 15.
BUFF_15_PDEV	29:28	0	Upper two bits of device number for buffer 15.
BUFF_13_EN	27	0	Enable buffer 13
BUFF_13_VDEV	26	0	Virtual device select for buffer 13.
BUFF_13_PDEV	25:24	0	Upper two bits of device number for buffer 13.
BUFF_11_EN	23	0	Enable buffer 11
BUFF_11_VDEV	22	0	Virtual device select for buffer 11.
BUFF_11_PDEV	21:20	0	Upper two bits of device number for buffer 11.
BUFF_9_EN	19	0	Enable buffer 9
BUFF_9_VDEV	18	0	Virtual device select for buffer 9.
BUFF_9_PDEV	17:16	0	Upper two bits of device number for buffer 9.
BUFF_7_EN	15	0	Enable buffer 7
BUFF_7_VDEV	14	0	Virtual device select for buffer 7.
BUFF_7_PDEV	13:12	0	Upper two bits of device number for buffer 7.

Table 38 Odd Device Read Response Buffer Register

Name	Bits	Reset Value	Description
BUFF_5_EN	11	0	Enable buffer 5
BUFF_5_VDEV	10	0	Virtual device select for buffer 5.
BUFF_5_PDEV	9:8	0	Upper two bits of device number for buffer 5.
BUFF_3_EN	7	0	Enable buffer 3
BUFF_3_VDEV	6	0	Virtual device select for buffer 3.
BUFF_3_PDEV	5:4	0	Upper two bits of device number for buffer 3.
BUFF_1_EN	3	0	Enable buffer 1
BUFF_1_VDEV	2	0	Virtual device select for buffer 1.
BUFF_1_PDEV	1:0	0	Upper two bits of device number for buffer 1.

Table 38 Odd Device Read Response Buffer Register

3.9.5 Read Response Buffer Status Register

This read only register contains the current response buffer status.

Name	Bits	Reset Value	Description
RRB_VALID	31:16	0	Read Response Buffer VALID bits indicate that the corresponding buffer currently has data ready for a PCI device.
RRB_INUSE	15:0	0	Read Response Buffer INUSE bits indicate that the corresponding buffer currently has an outstanding Crosstalk request.

Table 39 Read Response Buffer Status Register

3.9.6 Read Response Buffer Clear Register

A write to this register clears the current contents of the buffer..

Name	Bits	Reset Value	Description
	31:16	0	Reserved
RRB_CLEAR	15:0	0	A write of the corresponding Read Response Buffer CLEAR bits will clear the valid bit. This operation can only be used when the buffer has been disabled and is still valid.

Table 40 Read Response Buffer Clear Register

CHAPTER 4

Address Mapping

The *Bridge* ASIC performs two types of address mapping, *Crosstalk* interconnect to PCI or GIO bus (PCI/GIO) {this direction of flow is referred to as *pio mode*} and PCI/GIO bus to *Crosstalk* interconnect {this direction of flow is referred to as *dma mode*}. The *Crosstalk* interconnect specification defines a single 48-bit address space {256 Tera Bytes} per widget of which the *Bridge* uses the lower 8 Giga Bytes. The GIO bus supported on the *Bridge* ASIC has single address space of 4 Giga bytes in size. The PCI bus specification defines multiple address spaces and sizes which are supported in the *Bridge* and are defined later in this chapter.

The pio mode bus translation mechanism consists of generating a 32-bit PCI/GIO bus address from the *Crosstalk* 48-bit address. This translation occurs using per device adjustable and global fixed regions.

The dma mode translation mechanism uses three mapping techniques, 32-bit address and 64-bit address direct map schemes for a portion of system memory and a page based PMU using 32-bit addresses.

4.1 Bus Address Maps

4.1.1 PCI Address Map

The PCI bus specification defines 3 address spaces; the memory space, the I/O space, and the configuration space. PIO mode operations are al-

lowed to all three spaces, only memory space may have dma mode operations. The PCI memory space can operate with either 32-bits or 64-bits address {32-bit address in both pio and dma mode, 64-bit address in dma mode only}. The PCI bus specification also requires that access below 4 GBytes must be performed with a 32-bit address cycle. The PCI I/O address space is always 32-bits in size.

The PCI bus specification defines a type 0 and a type 1 configuration cycle. The *Bridge* supports both type 0 and type 1 configuration cycles used for device initilization. The PCI configuration space is limited to 64 words per logical device, 8 logical devices per physical device, 10 physical devices per bus for a total of 20Kbytes. Type 0 space is mapped for each device, type 1 space uses a sliding window adjustable with a register. *SGI defines a word as a 32-bit quantity and a double word as a 64-bit quantity, the PCI specification defines a word as a 16-bit quantity and a double word as a 32-bit quantity. This document uses the SGI definitions of words and double words.*

The *Bridge* defines usage of the PCI memory space as follows:

PCI Address	Address Size	Definition
0x0000_0000 - 0x3fff_ffff	32-bits	This is the address region where PCI devices reside. This area can be accessed from the *Crosstalk* interconnect.
0x4000_0000 - 0x7fff_ffff	32-bits	Access to this area will generate an operation to system space on the *Crosstalk* interconnect using the PMU.
0x8000_0000 - 0xffff_ffff	32-bits	Access to this area will generate an operation to system space on the *Crosstalk* interconnect using the 32-bit direct mapped hardware.
0x0000_0001_0000_0000 - 0x0fff_ffff_ ffff_ffff	64-bits	Unused, the bridge will not respond.
0x1000_0000_0000_0000 - 0xffff_ffff_ ffff_ffff	64-bits	Access to this area will generate an operation to system space on the *Crosstalk* interconnect using the 64-bit direct mapped hardware.

Table 41 *Bridge* **PCI Memory Space Usage**

The *Bridge* defines the entire I/O space for PCI devices on the bus and allows access from the *Crosstalk* interconnect. The *Bridge* allows the

Crosstalk interconnect access to the PCI configuration space for initilization. For a complete definition of the operation of the PCI bus please refer to the PCI Local Bus Specification 2.1.

4.1.2 GIO Address Map

The GIO bus specification defines a single address space, but multiple sizes, the *Bridge* supports only the 32-bit address size mode.

4.1.3 *Crosstalk* System Address Map

The *Crosstalk* interconnect comprises a single 48-bit address space for each widget. The *Bridge* ASIC is capable of mapping PCI/GIO transaction to any widget, anywhere in the target widget's 48-bit address.

4.2 *Crosstalk* to PCI/GIO Bus Mapping

The *Bridge* ASIC performs address translation from *Crosstalk* system addresses to PCI/GIO bus addresses. This translation is done in sections based on the device registers and through fixed regions which map the entire pio mode accessible areas.

4.2.1 *Crosstalk* View of PCI

The PCI bus is comprised of 3 address spaces, the memory space, the i/o space and the configuration space. Access to these sections are selected by the command field at the start of a PCI bus transaction.

The PCI bus uses a Single Address Cycle to perform a 32-bit address mode access and a Dual Address Cycle to perform a 64-bit address mode access. PCI requires all masters capable of generating a 64-bit address to generate a Single Address Cycle while accessing the first 4 GBytes (32-bits) of address space. **The *Bridge* mapping hardware only supports the translation of *Crosstalk* addresses {pio mode} to Single Address Cycle (32-bit address), dual address cycles will not be generated when Bridge is the PCI bus master.**

The *Bridge* maps the upper 3 GBytes of the 32-bit PCI memory space to *Crosstalk* system {dma mode transfers} space leaving only 1 GByte for PCI devices {pio mode}. The *Bridge* will not allow loopback mapping.

The *Crosstalk* interconnect specification defines a 16 MByte section for each widget called Widget Space. (See Figure 3) Widget space is used to

access internal registers, SSRAM, FLASH, and PCI/GIO devices. Widget space in the *Bridge* ASIC is divided into 10 sections, six 1 MByte sections, three 2 MByte sections, and one 4MByte section. (Figure 3) The first 2 MByte section is used for *Bridge* ASIC internal registers, internal address translation entry ram, external map SSRAM, PCI interrupt acknowledge cycle generation, PCI special cycle generation and PCI configuration space. The 4 MByte section is used to access up to 4 MByte of external flash PROM at the boot address of 0x0000_00c0_0000 which is translated form the processor boot address of 00_1fc0_0000 by the memory controller. The flash area is split into two sections of 2MBytes each, selected by FLASH_CS1_N and FLASH_CS0_N. An external pin (FLASH_SEL) can be used to swap which device is at the base address. The remaining 8 sections are used to access 8 PCI devices. The six 1 MByte sections are used for PCI/GIO devices 2 through 7. The two 2 MByte sections are used for PCI/GIO devices 0 and 1.

The *Bridge* ASIC also supports mapping a larger address then the 16MByte widget space. The larger mapping consists of an eight Gigabyte space starting at address 0x0000 0000 0000 with 16Mbytes of widget space, then the lower 1 GByte of pci memory space alias twice, once for 32-bit pci device targets (0000_4000_0000 - 0000_7fff_ffff), and once for 64-bit device targets(0000_8000_0000 - 0000_bfff_ffff). Starting at 4GB, the entire 4 GByte of I/O space is accessible. Unused areas are reserved.

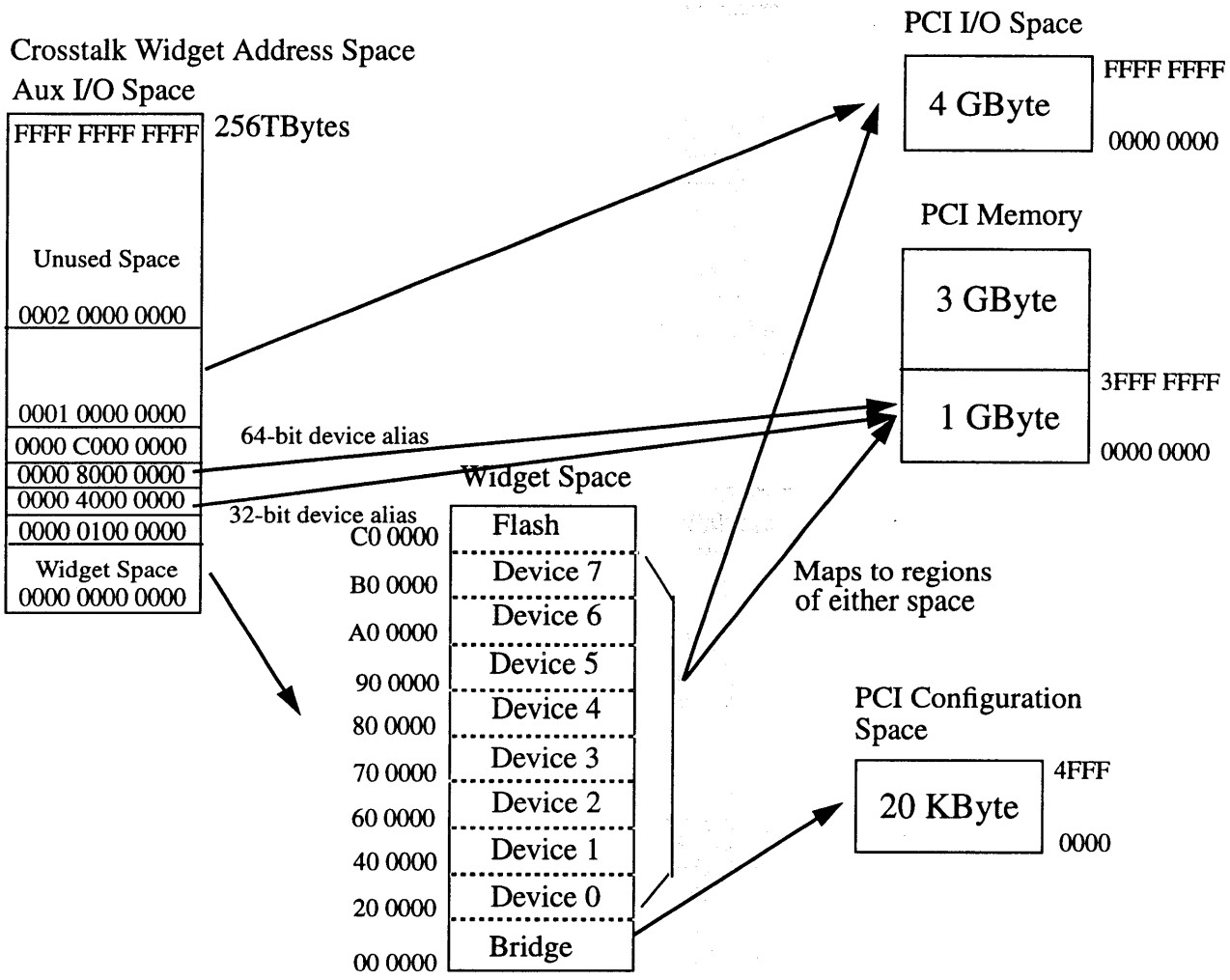


Figure 3 Crosstalk View of the PCI Bus

4.2.2 *Crosstalk* View of GIO

When mapping *Crosstalk* addresses to GIO addresses, only the 32-bit address mode of GIO is used. Mapping for GIO is the same as PCI with the following exceptions: There are no I/O or configuration spaces.

4.2.3 *Crosstalk* Mapping Registers

The *Crosstalk* mapping is controlled by the eight Device Registers. The *Bridge* uses *Crosstalk* addresses to determine if the access is to widget space, or other areas mapped through the *Bridge*. As shown in Figure 3,

widget space contains sections which are mapped to the address spaces on the PCI/GIO buses.

In widget space, the Device Register bit DEV_IO_MEM is used to enable either memory or I/O space. When in GIO mode the *Bridge* ignores DEV_IO_MEM bit. The DEV_OFF field is used to generate the upper bits of the 32-bit address. When mapping into memory space the upper 2 bits (A31:30) of the DEV_OFF field are ignored by the mapping hardware and driven to 0 on the bus. The DEV_SWAP bit controls if the data corresponding to this transaction (PIO type) should be byte swapped. Refer to Appendix A for a complete definition of how byte swapping operates.

Device #	DEV_IO_MEM	DEV_OFF bits	PCI Address Bits
0	1 mem space	9:1	29:21
1	1 mem space	9:1	29:21
2	1 mem space	9:0	29:20
3	1 mem space	9:0	29:20
4	1 mem space	9:0	29:20
5	1 mem space	9:0	29:20
6	1 mem space	9:0	29:20
7	1 mem space	9:0	29:20
0	0 i/o space	11:1	31:21
1	0 i/o space	11:1	31:21
2	0 i/o space	11:0	31:20
3	0 i/o space	11:0	31:20
4	0 i/o space	11:0	31:20
5	0 i/o space	11:0	31:20
6	0 i/o space	11:0	31:20

Table 42 Device Offset Index DEV_OFF

Device #	DEV_IO_MEM	DEV_OFF bits	PCI Address Bits
7	0 i/o space	11:0	31:20

Table 42 Device Offset Index DEV_OFF

The MEM_SWAP bit is used to control byte swapping for accesses to PCI memory or GIO spaces not mapped through the device windows (widget space). The IO_SWAP bit is used to control byte swapping for accesses to PCI IO space not mapped through the device windows.

4.2.4 *Crosstalk* Widget Space Address Map

Crosstalk Address (Widget Space)	Description
0x00_0000 -> 0x00_FFFF	Local Registers See Chapter 2
0x01_0000 -> 0x01_03FF	Internal Address Translation Entry RAM
0x02_0000 -> 0x02_0FFF	PCI Device 0 Configuration Space
0x02_1000 -> 0x02_1FFF	PCI Device 1 Configuration Space
0x02_2000 -> 0x02_2FFF	PCI Device 2 Configuration Space
0x02_3000 -> 0x02_3FFF	PCI Device 3 Configuration Space
0x02_4000 -> 0x02_4FFF	PCI Device 4 Configuration Space
0x02_5000 -> 0x02_5FFF	PCI Device 5 Configuration Space
0x02_6000 -> 0x02_6FFF	PCI Device 6 Configuration Space
0x02_7000 -> 0x02_7FFF	PCI Device 7 Configuration Space
0x02_8000 -> 0x02_8FFF	PCI Type 1 Configuration Space
0x03_0000 -> 0x03_0007	PCI Interrupt Acknowledge Cycle
0x08_0000 -> 0x0F_FFFF	External Sync SSRAM
0x10_0000 -> 0x1F_FFFF	Reserved

Table 43 Widget Space Address Map

Crosstalk Address (Widget Space)	Description
0x20_0000 -> 0x3F_FFFF	PCI/GIO Device 0 Space
0x40_0000 -> 0x5F_FFFF	PCI/GIO Device 1 Space
0x60_0000 -> 0x6F_FFFF	PCI/GIO Device 2 Space
0x70_0000 -> 0x7F_FFFF	PCI/GIO Device 3 Space
0x80_0000 -> 0x8F_FFFF	PCI/GIO Device 4 Space
0x90_0000 -> 0x9F_FFFF	PCI/GIO Device 5 Space
0xA0_0000 -> 0xAF_FFFF	PCI/GIO Device 6 Space
0xB0_0000 -> 0xBF_FFFF	PCI/GIO Device 7 Space
0xC0_0000 -> 0xFF_FFFF	External Flash PROMS 1,0

Table 43 Widget Space Address Map

Table 43 contains the mapping for the *Bridge* widget space. Table 44 contains the offsets for the PCI configuration space for a device. The addition of a type 1 configuration space is included to support devices behind a PCI to PCI bridge. For field definition and programming information consult the ***PCI Bus Specification***.

Offset	Description D31:D24 D23:D16 D15:D8 D7:D0
0x0	Device ID | Vendor ID
0x4	Status | Command
0x8	Class Code | Revision ID
0xC	BIST | Header Type | Latency Timer | Cache Line Size
0x10 -> 0x24	Base Address Registers
0x28	Cardbus DIS Pointer
0x2C	Subsystem ID | Subsystem Vendor ID

Table 44 PCI Configuration Space

Offset	Description D31:D24 D23:D16 D15:D8 D7:D0
0x30	Expansion ROM Base Address
0x34 -> 0x38	Reserved
0x3C	Max_Lat | Min_Gnt | Int Pin | Int Line
0x40 -> 0x100	Device Specific

Table 44 PCI Configuration Space

4.3 PCI/GIO Bus to *Crosstalk* Mapping

The *Bridge* ASIC supports two mechanisms to translate PCI/GIO bus addresses into *Crosstalk* addresses {dma mode}, a direct mapped scheme and a page mapped scheme. The direct mapped scheme maps an area of the PCI/GIO bus, the size of which is dependent on the address mode of the bus cycle, to a portion of the *Crosstalk* system space. Direct mapping generates a *Crosstalk* interconnect address directly from the PCI/GIO bus address. Since some modes of the PCI/GIO bus do not generate enough address bits to complete a 48-bit *Crosstalk* address, the extra bits are generated from a register. When this register is used the mapping is referred to as 32-bit direct mapping, when the register is not used the mapping is referred to as 64-bit direct mapping.

The page mapping scheme maps up to 64K 4KByte or 16KByte pages, anywhere in *Crosstalk* system space for a total of 1GByte of mapped space. Both the page mapped and direct schemes provide coherency, prefetching, and precise attributes to enhance *Bridge* performance. Each of these attributes are selectable for each page in the page mapping area or per device in the direct map area. Section 4.3.4 provides a explanation of the attribute functionality.

4.3.1 PCI View of *Crosstalk*

A PCI bus device can only access the *Crosstalk* interconnect address space through the PCI memory address space. Figure 4 shows the mapping of both 32-bit and 64-bit PCI memory spaces to *Crosstalk* system address space.The first 1 GByte of PCI memory space is defined as local space and the *Bridge* does not respond to any addresses in this space. One

exception is addresses 0x3fff_0000 thru 0x3fff_ffff, which the bridge does respond to and causes a flush of all buffers associated with the device accessing this address. The next 1 GByte area is used for the page mapping scheme define later. The top 2 GBytes in the 32-bit address space is defined as 32-bit direct mapped space. On PCI, addresses generated by a 64-bit address space cycle start above the 32-bit address space (4 GBytes). Within the PCI 64-bit address space the *Bridge* maps 256 TBytes through using the upper address bits as attributes for the transaction.

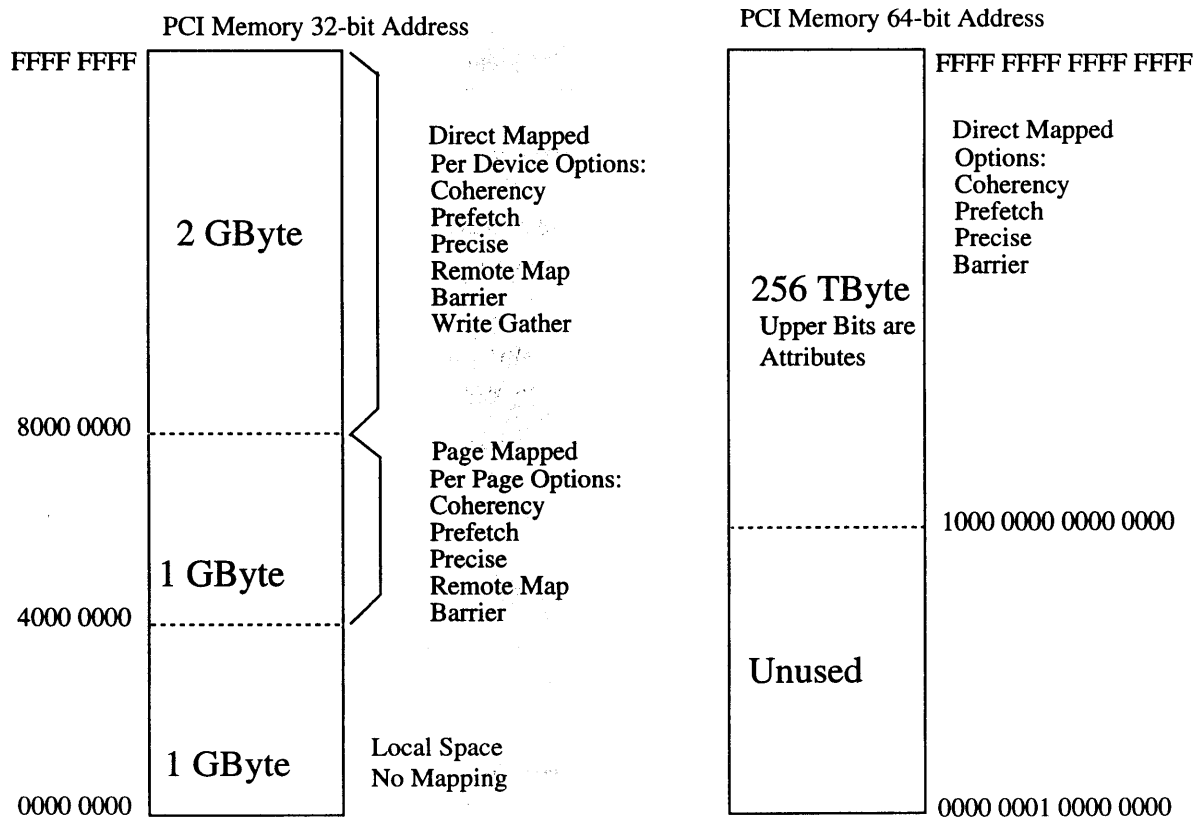


Figure 4 32-bit Address PCI View of Crosstalk

Programmers Note: Some combinations of attributes are not allowed. Combinations to check for are:

A read must be either precise (precise=1 and prefetch =0), prefetch (precise=0 and prefetch =1), or non-precise (precise=0 and prefetch =0). The combination of (precise=1 and prefetch =1) is not allowed. In 64-bit

address mode, do not assert either precise or prefetch on a write. The other attributes can be freely mixed.

4.3.2 GIO View of *Crosstalk*

The 32-bit address space of GIO operates exactly like the 32-bit memory space of PCI.

4.3.3 PCI/GIO Direct Mapping

The *Bridge* supports 2 versions of direct mapping: a 32-bit direct mapped space and a 64-bit direct mapped space. Both versions provide a quick and easy method for PCI/GIO devices to map system resources. As shown in Figure 4, direct mapping remaps a section of the PCI/GIO memory space to a section of the *Crosstalk* system space.

In 32-bit direct mapped mode, access to the upper 2 GBytes of PCI/GIO address space is mapped to *Crosstalk* system space based on the value of DIR_OFF in the direct mapping register. The target widget id is also obtained from this register. Table 45 shows the mapping values for the different values of DIR_OFF. The DIR_OFF register is shared by all devices, transfer attributes are generated from the device registers on a per device basis and the remote map field is always set to 0. It should be noted that all translations are simple bit masking, an additional mapping can be made if the value of DIR_OFF is 0x0 and the DIR_ADD512 bit is set. The *Crosstalk* address generated when DIR_OFF equals 0 and DIR_ADD512 equals 1 is offset 512 MBytes to allow for MIPS kseg spaces. (Refer to the MIPS Processor Specification Version 4 for a detailed description of kseg spaces.) The reason for the DIR_OFF function is in an ISD system, physical memory resides from 0x2000_0000 and 0xa000_0000. To efficiently map this region an offset was required.

DIR_OFF Value	*Crosstalk* Address
0x1FFFF	0xFFFF_8000_0000 - 0xFFFF_FFFF_FFFF
0x1FFFE	0xFFFF_0000_0000 - 0xFFFF_7FFF_FFFF
...	...
0x00002	0x0001_0000_0000 - 0x0001_7FFF_FFFF
0x00001	0x0000_8000_0000 - 0x0000_FFFF_FFFF

Table 45 32-bit Direct Mapped Offset Address

DIR_OFF Value	*Crosstalk* Address
0x00000	0x0000_0000_0000 - 0x0000_7FFF_FFFF

Table 45 32-bit Direct Mapped Offset Address

Access to addresses above the 4 GByte level on PCI require an address larger than 32-bits. It is these cycles that provide access to 64-bit direct mapped mode. Mapping of a 64-bit address cycle is done directly to the 48-bit *Crosstalk* address with no change. The *Bridge* version of GIO does not support address larger then 32-bits.

In 32-bit address direct map mode all the mapping attributes from the device registers are used, in 64-bit address direct map mode only the swap, gbr, and coherent bits are obtained from the device registers. The upper 16 address bits are used to generate the attribute fields.

PCI Address Bits	Attribute
63:60	Target ID
59	Prefetch
58	Precise
57	Virtual Request
56	Barrier
55:48	Remote Map Field

Table 46 64-bit Address Direct Map Attributes

Note: The *Bridge* does not support access to widget 0 (the crossbar) from 64-bit space, this will generate an address above 4 GByte to initiate a dual address cycle.

The remote map field above can be concatenated with either the address bits 40:47 or 0 to form a 16-bit remote address field in the *Crosstalk* packet. The direct mapping attributes are selectable per device from the

device registers and some attributes are also generated from upper address bits during a 64-bit address cycle. Table 47 describes the attributes.

Name	Description
PREFETCH	Prefetcher Enable bit provides two functions. The first is to start/continue the prefetcher operation. The second is read buffer management for the PCI/GIO bus. Normally a cache line read buffer is invalidated after the current cycle is completed, either FRAME_N negated (PCI) or byte count exhausted (GIO). The Prefetcher Enable bit allows buffers to maintain validity until any byte in the last word of the cache line is accessed. This bit allows multiple PCI/GIO bus transactions to a single buffer. For a complete invalidation rules see PCI/GIO Data Buffer Section. 64-bit address mode selectable.
BARRIER	Enable Barrier bit is passed to the CIU enabling the Barrier Transaction bit to be set in the *Crosstalk* packet. 64-bit address mode selectable.
SWAP	Device Enable Swapping 1 => enabled. When enabled the byte swapper exchanges data per Appendix A.
PRECISE	Precise Transaction bit forces read operations to fetch exactly the bytes requested not a full cache line. 64-bit address mode selectable.
COHERENT	Coherent Transaction 1 => coherent. This bit is passed to the CIU causing the Coherent Transaction bit to be set in the *Crosstalk* packet. 64-bit address mode selectable.
Virtual Request	Virtual Request bit is used in buffer management to separate the read prefetch buffers into two groups or "streams".
GBR	GBR Enable 1 => enabled. This bit is passed to the CIU causing the GBR bit to be set in *Crosstalk* packet. Refer to the *Crosstalk* Specification for a complete description of the GBR functionality. The GBR bit is selected on a per device basis from the device registers and is used for both page and direct spaces.

Table 47 Direct Mapping Attributes Per Device

Programmers note: Attribute selection table for the different address spaces.

Attribute	How Attribute Is Specified
Prefetch	Prefetch bit in the device register.
Swap	Swap_direct bit in the device register
Coherent	Coherent bit in the device register
Virtual Request	Not available in this space.
GBR	GBR bit in device register.
Target ID	Dir_w_id field of the direct map register. There is only one field for all devices which will target main memory.

Table 48 Address Attributes In 32 Bit Address Direct Space

Attribute	How Attribute Is Specified
Prefetch	Bit 59 of the address.
Swap	Swap_direct bit in the device register
Coherent	Coherent bit in the device register
Virtual Request	Bit 57 of the address.
GBR	GBR bit in the device register.
Target ID	Address bits (63:60).

Table 49 Address Attributes In 64 Bit Address Direct Space

Attribute	How Attribute Is Specified
Prefetch	Prefetch bit in the page map entry.
Swap	Swap_pmu bit in the device register
Coherent	Coherent bit in page map entry.

Table 50 Address Attributes In Page Mapped Space

Attribute	How Attribute Is Specified
Virtual Request	Not available in this space.
GBR	GBR bit in the device register.
Target ID	Target id field in the page map entry.

Table 50 Address Attributes In Page Mapped Space

4.3.4 PCI/GIO Page Mapping

The PCI/GIO PMU (Page Mapping Unit) is the mechanism which performs the page mapping of GIO/PCI bus addresses to *Crosstalk* system addresses. The PMU translates a 1 GByte section of the 32-bit memory address space into 48-bits of *Crosstalk* interconnect system space. The PMU can only be accessed by a 32-bit address operation. In addition the PMU also provides attributes for *Crosstalk* transfers, and *Bridge* data buffer management on a per page basis.

Page mapping is accomplished with a single level RAM map. The PCI address is used as an index into a RAM which provides an address translation entry (ATE). The contents of the ATE is defined in Table 52. The ATE provides the translated address, remote map field and attributes.

The *Bridge* supports an internal RAM map plus an optional external RAM map. The internal RAM contains 128 entries starting from the bottom of the 1GByte page mapped space. The external map also starts from the bottom of the 1GByte page mapped space but the first 128 pages are mapped from the internal RAM so the first 128 external pages are not used. The external RAM map contains 16K, 32K, or 64K ATEs depending on the external RAM size defined in the configuration register. Also pages mapped with the internal map are ALWAYS WRITE GATHER DISABLED, the PMU_WRT_GEN bit only effects accesses through the external map.

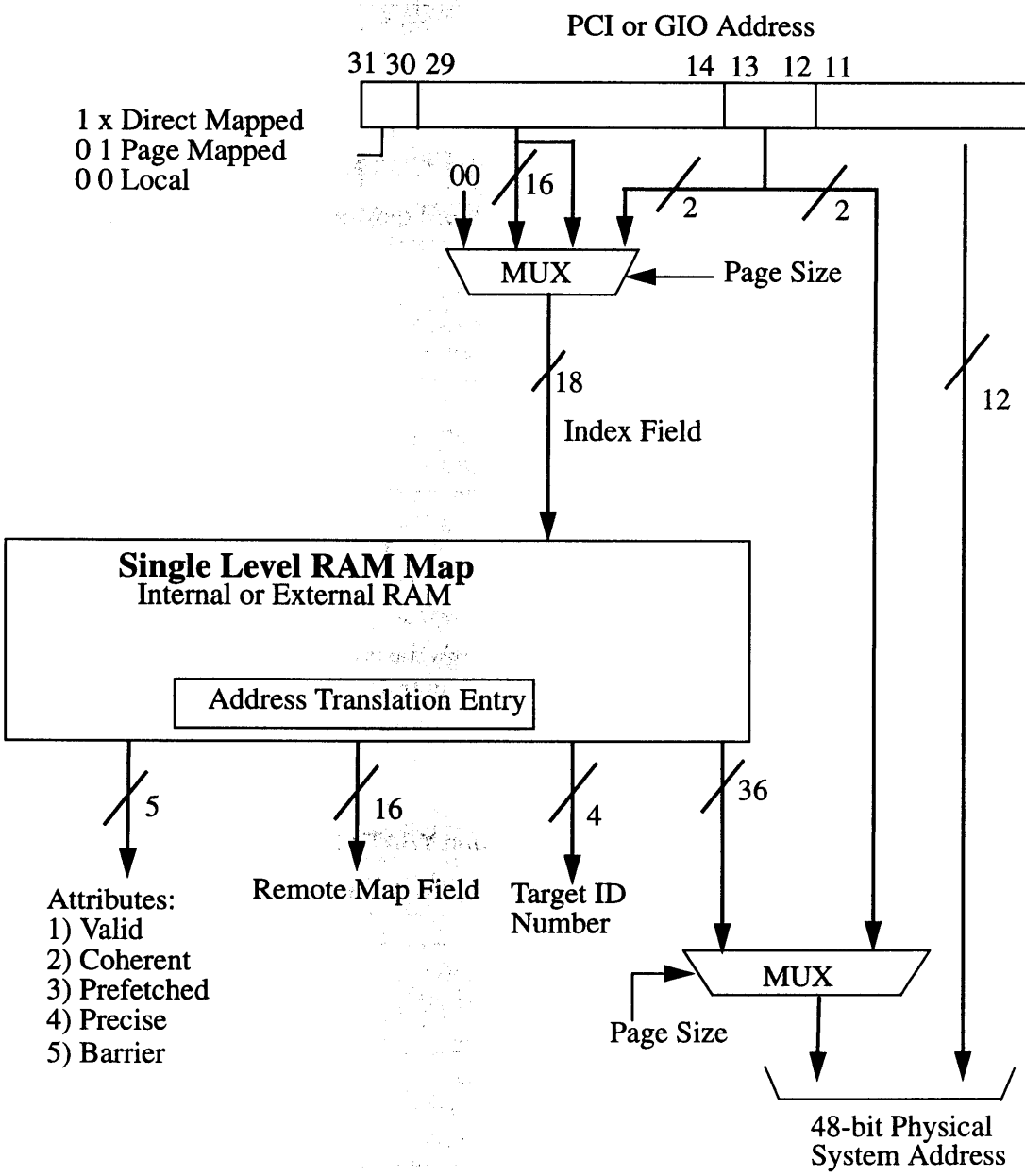


Figure 5 PCI/GIO PMU

Programmers note: The internal map is written a double words only but due to internal design issues it is read back as a single words only. See the software notes section for address ranges for word locations. The external map must be read and written as double words.

Figure 5 contains the translation path for a PCI/GIO address into a *Crosstalk* interconnect system address. Bits 31:30 of the PCI/GIO address are used to select PMU operation. Either 12 or 14 of the lowest address bits are passed through depending on the size of the page selected in the PMU external table pointer register. The remaining address bits (either A29:14 or A29:12) comprise an index (index field) used to select an address translation entry.

The first 128 entries are selected from the internal RAM, the remainder are selected from the external RAM. The external RAM size vs. page space is shown in the table below.

Memory Size	With 4K Page Size Number of Pages / Region Size	With 16K Page Size Number of Pages / Region Size
128KBytes	16K Pages => 64MBytes	16K Pages => 256MBytes
512KBytes	64K Pages => 256MBytes	64K Pages => 1024MBytes

Table 51 Page Size Mapping Regions

Table 52 contains a description of the 8-byte (64-bit) address translation entry. The GBR bit is controlled from the device registers.

Bits	Name	Description
63-48	Remote Map Field	Remote Map Field is a 16-bit field used by the Virtual Backplane for routing information between systems.
47-12	Upper Address bits	Upper Address bits are used generate a 48-bit address.
11-8	Target ID Field	Target ID Field is passed to the CIU for packet routing.
4	Barrier	Barrier Transaction 1 => barrier. This bit is passed to the CIU enabling the Barrier Transaction bit to be set in the *Crosstalk* packet.

Table 52 PMU Address Translation Entry

Bits	Name	Description
3	Prefetcher Enable	Prefetcher Enable bit provides two functions. The first is to start/continue the prefetcher operation. The second is read buffer management for the PCI/GIO bus. Normally a cache line read buffer is invalidated after the current cycle is completed, either FRAME_N negated (PCI) or byte count exhausted (GIO). The Prefetcher Enable bit allows buffers to maintain validity until any byte in the last word of the cache line is accessed. This bit allows multiple PCI/GIO bus transactions to a single buffer. For a complete invalidation rules see PCI/GIO Data Buffer section.
2	Precise Transaction	Precise Transaction bit forces read operations to fetch exactly the bytes requested not a full cache line.
1	Coherent Transaction	Coherent Transaction bit is passed to the CIU enabling the Coherent Transaction bit to be set in the *Crosstalk* packet.
0	Entry Valid	Entry Valid bit indicates that the address translation entry contains valid mapping information.

Table 52 PMU Address Translation Entry

4.3.5 Packetization of PCI/GIO Operations

PCI/GIO non-write buffered write transactions are buffered and split-up into optimal *Crosstalk* packet transfers. PCI read transactions can not be quite as clearly defined since PCI does not indicate if an access is a single word or a precise transfer. The *Bridge* ASIC uses the attributes in the device registers in direct space and the attributes in the page address translation entry for page mapped space, to select the transfer size for the *Crosstalk* packet. The *Bridge* will also make use of the GIO byte count and the PCI command ***Read Line*** and ***Read Multiple*** are used like pci read operation.

4.3.6 PCI/GIO Address Translation Flow Chart

Figure 6, Figure 7, and Figure 8 contain flow charts of PCI/GIO to *Crosstalk* address translation.

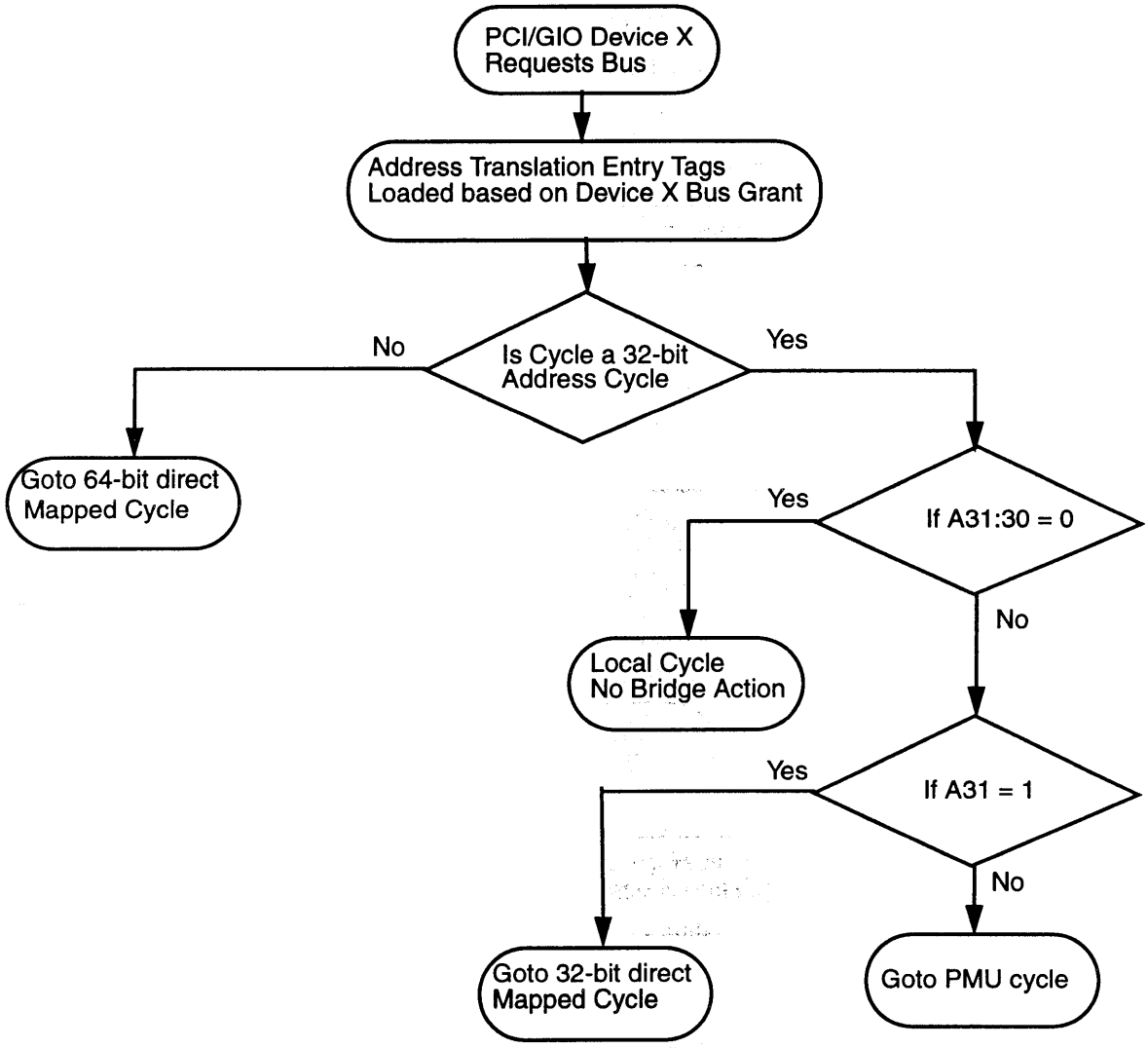


Figure 6 PCI/GIO Address Translation Flow Chart 1

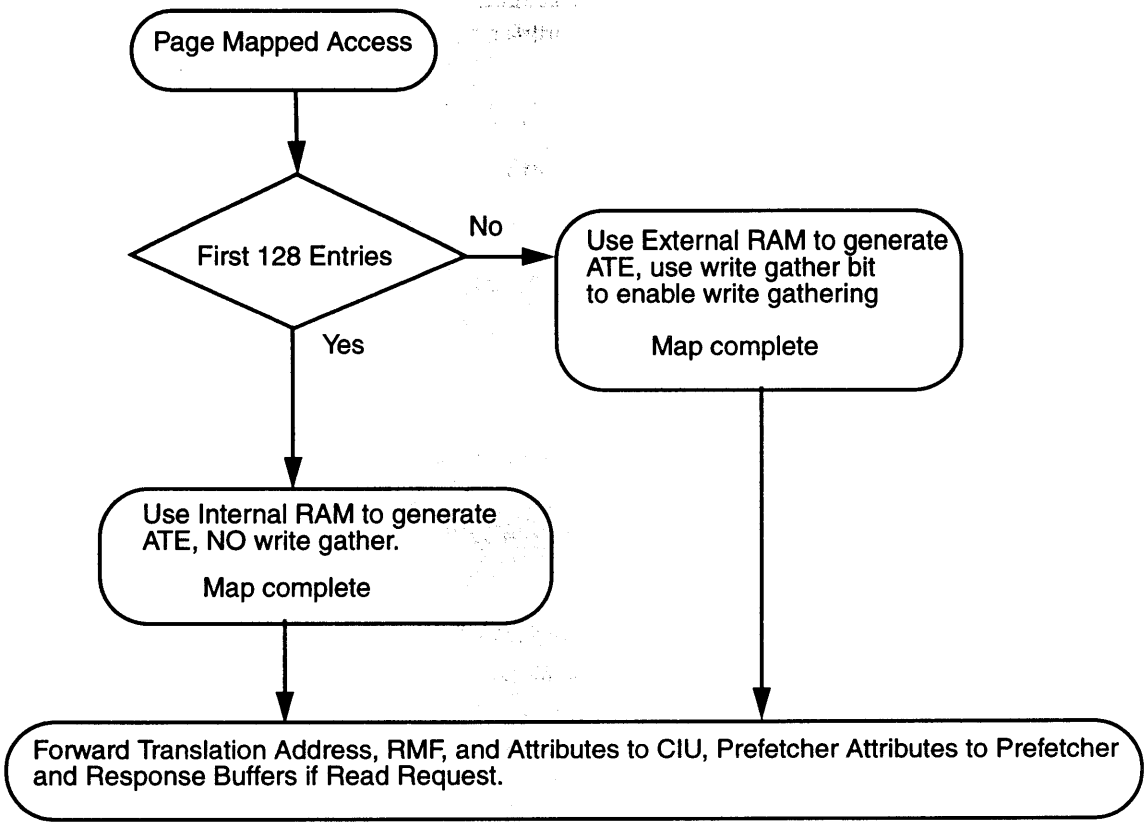


Figure 7 PMU Address Translation Flow Chart

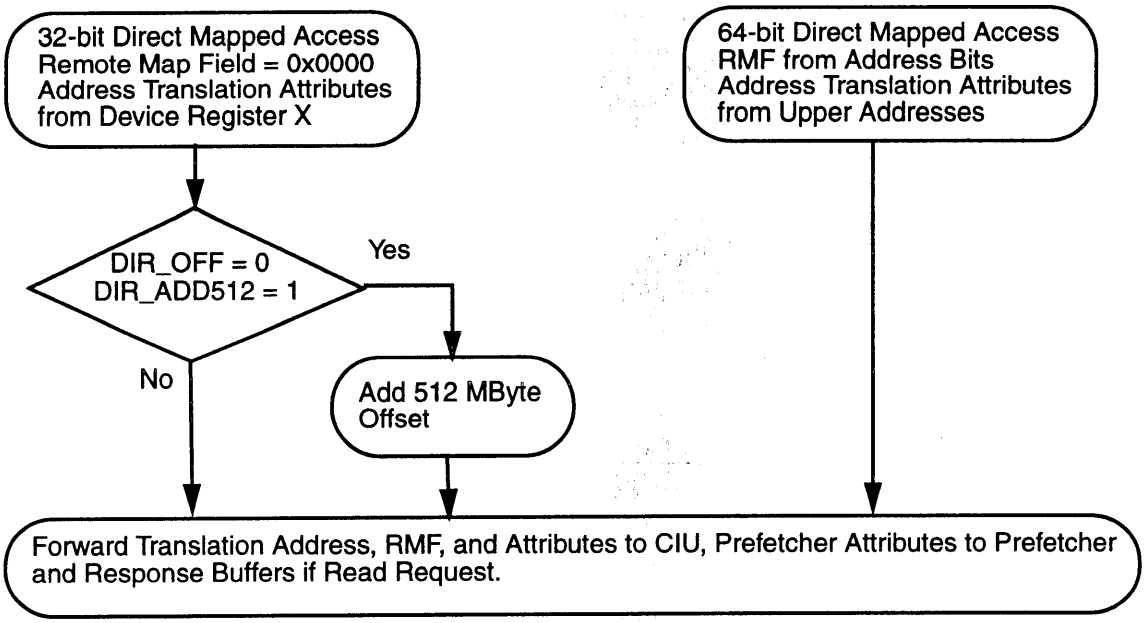


Figure 8 Direct Address Translation Flow Chart

CHAPTER 5

PCI/GIO Data Buffers

5.1 Data Buffer Overview

The *Bridge* ASIC has contains special buffering for two different data paths. The one path is used for operations generated from a widget as a master (pio mode). An example of these operations are such things as load/store operations from the processor. The other path is for operations initiated by a bus master from the PCI/GIO bus (dma mode). An example of these operation would be a SCSI DMA.

The buffering in the *Bridge* ASIC allows peak performance on both the PCI/GIO buses and the *Crosstalk* Interconnect when configured correctly. The next few sections will define the operational flushing policies, programming and use of the buffers in the ASIC.

5.2 Widget Master Buffers

The widget master buffers consist of a request fifo (in the request dispatcher), a request ping-pong (in the PCI master), and a response ping-pong buffer (in the request dispatcher). These buffers allow smooth flow of request operations through the bridge.

The request fifo holds initial requests until the request dispatcher can route the request to the proper section of the *Bridge* ASIC. This fifo can

hold three cache line or quarter cache line write requests or up to fifteen, read or double word write requests. **The Crossbow should be programmed for 3 bridge buffer locations.** The Bridge will automaticly handle the packing and proper credit management of smaller packets. If the operation is to the PCI/GIO bus, the ping-pong buffer provides an additional two operation buffer. If the operation requires a response, the PCI/GIO bus and the SSRAM/FLASH share the ping-pong response buffer.

5.3 PCI Master Buffers

There are two types of PCI master buffers used in the *Bridge* ASIC, read response buffers and write buffers. Read response buffers are used to hold the data returned from a read request made by the PCI master or the prefetcher. Write buffers are used to hold the data from a PCI master write operation until it can be sent on the *Crosstalk* interconnect.

5.3.1 Read Response Buffers

The *Bridge* ASIC contains sixteen read response buffers, split into two groups, the even device group and the odd device group. The read response buffers are hard allocated to a PCI master device by using the even/odd device response buffer registers. The PCI device number is based on which physical bus request/grant pair is used. The *Bridge* ASIC PCI bus supports eight bus request/grant pairs. To allocate a buffer to a given device, the buffer must have it's enable bit and the 2 most significant bits of the device number set. The LSB of the device number is implied by the use of even or odd read response register. The reason for the even/odd buffer is to reduce the compare logic required. This does limit the maximum number of buffers which can be used with any single bus request/grant pair to eight. In addition, each device can have 2 virtual request/grant pairs within a single bus request/grant pair, more on this later.

The *Bridge* ASIC provides three kinds of PCI to *Crosstalk* request mapping which effect read response buffer operations (see Chapter 3), precise, non-precise, and prefetched. All read operations require at least one buffer to be allocated to that bus request/grant pair. Those devices which can use multiple buffers, a buffer must be ready to be assigned to the current transaction. If all the buffers for a given device are in use, then the transaction is retried and no information is stored. Since all PCI operations which are retried with no data transfer will be repeated there is no data loss problems.

Programmers Note: If you do not assign a buffer to a device or a virtual device and that device performs a read, that operation will be retried forever and no error is signaled.

5.3.1.1 Precise Operations

A precise PCI read operation causes the *Bridge* ASIC to issue a retry on the PCI bus and generate a double word *Crosstalk* operation requesting only those bytes selected by the PCI byte enables. The PCI device will continue to request the bus and perform the read operation. The *Bridge* ASIC will continue to issue retries until the response arrives and then the next read operation will provide the requested data. The *Bridge* ASIC compares exact address and byte enables in this mode. After the read is complete the buffer is ready for another transaction.

Read response buffers filled by a precise operation are flushed based on the following rules:

- When a write from the same master and an address match occurs.
- When an interrupt occurs from a interrupt pin assigned to the device associated with the buffer.
- Single access to the buffer. (normal completion)
- PCI master write access to 0x3fff_0000 thru 0x3fff_ffff
- PIO flush (see Section 5.3.1.4)

5.3.1.2 Non-Precise Operations

A non-precise PCI read operation behaves like the precise transaction except that the *Bridge* ASIC generates a *Crosstalk* cache line read operation. During the following read operations only the cache line address bits are use for the compare, (bits 6 through 2 are don't cares). Accessing the buffer at any address within the cache line provides a response and will continue (bursting) until either the end of the cache line is reached or the PCI cycle is terminated by the master. At this point the buffer ready for reuse.

Read response buffers filled by a non-precise operation are flushed based on the following rules:

- When a write from the same master and an address match occurs.
- When an interrupt occurs from a interrupt pin assigned to the device associated with the buffer.
- Single access to the anywhere in buffer. (normal completion)
- PCI master write access to 0x3fff_0000 thru 0x3fff_ffff
- PIO flush (see Section 5.3.1.4)

5.3.1.3 Prefetch Operations

A prefetched PCI read operation starts much like the previous operation, a retry is issued on the PCI bus, a *Crosstalk* cache line request is issued, and the prefetcher is enabled for this transaction. The prefetcher will store the next cache line address and look for addition free buffers currently assigned to the requesting device. If additional assigned buffers are free then the prefetcher will launch incremental cache line read requests until all the buffers are in use. The prefetch can be enabled to stop at either a 4K or 16K page boundary. The prefetcher will continue to search for free buffers and posting read request until either a page crossing or a flush condition is reached. The prefetcher can only increment linearly on physical addresses. Prefetched reads also effect when a buffer is available for the next transaction. Both the precise and non-precise operation only allow a single bus tenure to access the buffer then the buffer free for use in the next transaction. The prefetched buffer will allow multiple bus tenures to access the buffer until the last (most significant word) single/double word is accessed. (32-bit/64-bit bus). With the last access the buffer is ready for another transaction.

Read response buffers filled by a prefetch operation are flushed based on the following rules:

- When any non-sequential read is performed by the PCI master (breaking stream).
- When a write from the same master occurs.
- When an interrupt occurs from a interrupt pin assigned to the device associated with the buffer.
- Access to the last word of data in the buffer. (normal completion)
- PCI master write access to 0x3fff_0000 thru 0x3fff_ffff
- PIO flush (see Section 5.3.1.4)

5.3.1.4 PIO (Processor) Flush

The read response buffers can be cleared with a PIO by setting the corresponding enable bit in the even/odd read response buffer registers to zero, then checking the read response buffer status register. If RRB_INUSE bit is set then, you must wail until the RRB_VALID bit is set and the RRB_INUSE bit is clear. A PIO to the corresponding RRB_CLEAR bit clears the buffer. If both the RRB_VALID and RRB_INUSE are clear then the buffer is cleared. This is also the mechanism used to reassign buffers "on the fly" to other devices. When a buffer is clear and disabled it can be reassigned to another device.

5.3.1.5 Virtual Device

From the above rules of operation for the read response buffers, a single PCI master device could have either a single prefetch stream, or multiple random precise/non-precise requests equalling the number of buffers allocated. This works well for some devices, but others like scsi controllers may have a large data stream which would want to use the prefetch feature and an occasional dma descriptor read to an unrelated address. Using the above rules the data stream is flushed on every descriptor read which will negativity impact performance. It is for cases like these that the virtual request feature was added. To use this feature the PCI master must be able to generate 64-bit PCI addresses. The *Bridge* uses PCI address bit 57 to differentiate between the virtual read streams. The even/odd read response buffer registers have a bit for each buffer to select virtual buffer. This bit is compared against address bit 57 in selecting or clearing the buffers. Write streams are also differentiated with the same bit. If a single address cycle operation is used, the operation is treated as if the virtual device bit was set to "0".

5.3.2 Write Request Buffers

Unlike the read response buffers, the seven write buffers are dynamically allocated by the PCI slave logic. This allows the maximum performance for the minimum amount of buffer ram. The *Bridge* ASIC supports a dual ring arbitration scheme, allowing PCI devices to be either real-time or not. Any non-real time device must leave 2 write buffers free at all times. If a write occurs and only two buffers are free, then the write is retried until more buffers are free. Any real time device can not use more than five buffers at a time, and must leave at least one free if it already has write buffers in use.

These rules apply to all cache line aligned transfers. Non-aligned transfers must be broke up into quarter cache transfers (on the *Crosstalk* interconnect) or write gathered. In large contiguous transfers, only the starting and ending transfers might be effected, hence little performance impact. PCI devices which are not able to burst an entire 128 byte cache might want to use write gathering mode on the write buffers. By setting the write gather bits in the device (x) registers, when a write occurs the data is gathered into larger units to be sent on *Crosstalk*. A PCI device can only have a single write gather buffer in use at one time. No more than four can be active at any one time. Flushing of the write buffers is done when the following occurs:

- A read from the device corresponding to the gather buffer.

- A non-contiguous write.
- An interrupt from the interrupt pins associated with the device.
- A PIO access to the write request buffer flush register.
- PCI master write access to 0x3fff_0000 thru 0x3fff_ffff

CHAPTER 6

Error Cases

The *Bridge* ASIC can generate errors in many different sections and log different amounts of data about the error in question. This section attempts to describe the different types of errors which can occur and the status information which can be obtained.

When any interruptible event occurs, the interrupt status bit corresponding to the event is set in the INT_STATUS register if the matching enable bit in the INT_ENABLE register is set. Some interrupts may log certain information within various error registers. Interrupts which share error register are considered a group. If additional interrupts in the same group occur before they are cleared then the MULTI_ERR bit in the INT_STATUS register is set and the error registers associated with that group will not latch the new error state. INT_STATUS register bits are cleared by writing the RESET_INT_STATUS register.

6.1 Incoming *Crosstalk* Packets

Incoming *Crosstalk* Packets are requests from other widgets or responses to requests generated by the *Bridge* ASIC. These errors are generated from *Crosstalk* operation or packet format.

6.1.1 Response Packets

The Hardware Error bit in the receive buffer is not visible via software but is included to describe the error operation. When an error occurs the bit is set but processing of these error cases are dependent on the PCI device accessing the buffer. If the stream is changed and the buffers flushed without accessing the data the errors are never indicated.

Packet & Error Type	Logged Status
Write Response Packet Since the *Bridge* ASIC does not generate write with response requests, receiving a write response is an error.	UNEXPECTED_RESP bit in the interrupt status register is set. In addition the command word is stored in the Bridge Aux Error Command Word register. (CRP_GROUP)
Read Response Packet Invalid Buffer (TNUM). The *Bridge* only uses tnums 0 through 15 for read requests therefore receiving a tnum greater than 15 is an error.	UNEXPECTED_RESP bit in the interrupt status register is set. In addition the command word is stored in the Bridge Aux Error Command Word register. (CRP_GROUP)
Read Response Packet Buffer not Enabled. When a read request is made, the *Bridge* enables a response buffer corresponding to the tnum in the request. After the request is received the enable is cleared, this prevents the buffer from being overwritten by misrouted packets and generates this error case.	UNEXPECTED_RESP bit in the interrupt status register is set. In addition the command word is stored in the Bridge Aux Error Command Word register. (CRP_GROUP)
Read Response Packet Data Size / Packet Size Mismatch or Unsupported Data Size. An arriving response has the data size encoded in the command word of the response. If the head and tail sideband bits define a data segment which differs from the command word then a data size mismatch occurs.	Hardware error bit and the data size error bit in Receive Buffer number (TNUM) is set. If the buffer is accessed by a PCI device then the BAD_XRESP_PACKET bit in the interrupt status register is set. In addition the lower 48-bits of the PCI address, buffer number, and the PCI device number is stored in the Response Buffer Address Registers. (RESP_BUF_GROUP)

Table 53 Response Packet Error Conditions

Packet & Error Type	Logged Status
Read Response Packet Command Word Error bit set. This bit is set by the source generating the response packet. An example would be a memory read with a unrecoverable error.	Hardware error bit in Receive Buffer number (TNUM) is set. If the buffer is accessed by a PCI device then the RESP_XTALK_ERROR bit in the interrupt status register is set. In addition the lower 48-bits of the PCI address, buffer number, and the PCI device number is stored in the Response Buffer Address Registers. (RESP_BUF_GROUP)
Read Response Packet Sideband Invalid bit set. This bit is set by the source generating the response packet or a transfer agent. An example would be a memory read with a unrecoverable error after the command word been sent.	Hardware Error bit in Receive Buffer number (TNUM) is set. If the buffer is accessed by a PCI device then the RESP_XTALK_ERROR bit in the interrupt status register is set. In addition the lower 48-bits of the PCI address, buffer number, and the PCI device number is stored in the Response Buffer Address Registers. (RESP_BUF_GROUP)

Table 53 Response Packet Error Conditions

6.1.2 Request Packets

Packet & Error Type	Logged Status
Invalid Packet Type: **-Fetch and Op Packet** **-Store and Op Packet** **-Special Request Packet** **-Special Response Packet** **-Reserved Entries** These are packet operations not supported by the bridge.	The UNSUPPORTED_XOP bit in the interrupt status register is set. In addition the 48-bits of the *Crosstalk* address is stored in the Bridge Error Address Registers and the command word is stored in the Bridge Error Command Word register. (REQ_DSP_GROUP)

Table 54 Request Packet Error Conditions

Packet & Error Type	Logged Status
Request Packet Data size / Packet size mismatch. An arriving packet has the data size encoded in the command word of the request. If the head and tail sideband bits define a data segment which differs from the command word then a data size mismatch occurs.	The BAD_XREQ_PACKET bit in the interrupt status register is set. In addition the 48-bits of the *Crosstalk* address is stored in the Bridge Error Address Registers and the command word is stored in the Bridge Error Command Word register.
Request Packet Unsupported Data size. An arriving packet has the data size encoded in the command word of the request which is not supported for the requested operation	The UNSUPPORTED_XOP bit in the interrupt status register is set. In addition the 48-bits of the *Crosstalk* address is stored in the Bridge Error Address Registers and the command word is stored in the Bridge Error Command Word register. (REQ_DSP_GROUP)
Request Packet Command Word Error bit set or Sideband Invalid bit set. This condition on a write request packet indicates that the write data in invalid and the write is not performed. These bits are not checked on read packets.	The REQ_XTALK_ERROR bit in the interrupt status register is set. In addition the 48-bits of the *Crosstalk* address is stored in the Bridge Error Address Registers and the command word is stored in the Bridge Error Command Word register. (REQ_DSP_GROUP)
Request Packet Invalid Address. Indicates that the request packet contains an address not supported by the bridge,	The INVALID_ADDRESS bit in the interrupt status register is set. In addition the 48-bits of the *Crosstalk* address is stored in the Bridge Error Address Registers and the command word is stored in the Bridge Error Command Word register. (REQ_DSP_GROUP)
Request Packet arrives when request fifo full. This can occur if the credit counters in the crossbow or heart are not programmed correctly.	The XREQ_FIFO_OFLOW bit in the interrupt status register is set. (CRP_GROUP)

Table 54 Request Packet Error Conditions

6.1.3 Receive Link Errors

Packet & Error Type	Logged Status
LLP asserts receive check bit error.	LLP_REC_CBERROR bit in the interrupt status register is set. (LLP_GROUP)
LLP asserts receive sequence number error	LLP_REC_SNERROR bit in the interrupt status register is set. (LLP_GROUP)
Receiver retry counter increments from FF -> 00	LLP_RCTY bit in the interrupt status register is set. (LLP_GROUP)

Table 55 Receive Link Errors

6.2 Outgoing *Crosstalk* Packets

Outgoing Packets are requests to other widgets or responses from requests generated by other widgets. These error are generated from *Crosstalk* operation of packet format.

6.2.1 Transmit Link Errors

Packet & Error Type	Logged Status
LLP asserts transmit retry	LLP_TX_RETRY bit in the interrupt status register is set. (LLP_GROUP)
Transmit retry counter increments from FF -> 00	LLP_ TCTY bit in the interrupt status register is set. (LLP_GROUP)
Transmitter max retry occurs	Fatal Error condition LLP shuts down, requires link reset from *Crossbow* ASIC to restart. Can be routed to test pin 3

Table 56 Receive Link Errors

6.3 SSRAM Parity Errors

During an access to the SSRAM, if a parity error occurs, the SSRAM_PERR bit in the INT_STATUS register is set. In addition, 8 bits corresponding to the byte(s) in error and a bit indicating the accessing source, (Mapper/Crosstalk), is logged in the ssram parity error register. In pio mode the double word aligned ssram address of the error, stored in bits 15:0. In pmu map mode, 3 bits of PCI device generating the map access is stored in the SSRAM Parity Error register.

If this error occurred during a *Crosstalk* request then the error bit in the command word of the response packet is set. If this error occurred from a request generator page mapping operation, the request is flushed, and used buffers are deallocated and write data is flush.

6.4 PCI Errors

6.4.1 *Bridge* as PCI Master Errors

Error Type	Logged Status
PCI Target Abort. This condition is caused by the target PCI device asserting target abort.	PCI_ABORT bit in the interrupt status register is set. The PCI address and device number is stored in the PCI Error upper & lower address registers. (PCI_GROUP)
Bridge Detected parity error (read data).	PCI_PARITY bit in the interrupt status register is set. The PCI address and device number is stored in the PCI Error upper & lower address registers. (PCI_GROUP)
PERR parity error (write data).	PCI_PERR bit in the interrupt status register is set. The PCI address and device number is stored in the PCI Error upper & lower address registers. (PCI_GROUP)
SERR system error.	PCI_SERR bit in the interrupt status register is set. The PCI address and device number is stored in the PCI Error upper & lower address registers. (PCI_GROUP)

Table 57 PCI Master Errors

Error Type	Logged Status
Master time-out. This error occurs when no PCI device drives device select.	PCI_MASTER_TOUT bit in the interrupt status register is set. The PCI address and device number is stored in the PCI Error upper & lower address registers. (PCI_GROUP)
Master retry count exhausted. This condition is cause by the target issuing too many retries.	PCI_RETRY_CNT bit in the interrupt status register is set. The PCI address and device number is stored in the PCI Error upper & lower address registers. (PCI_GROUP)

Table 57 PCI Master Errors

6.4.2 *Bridge* as PCI Slave Errors

Error Type	Logged Status
Bridge Detected parity error (write data).	PCI_PARITY bit in the interrupt status register is set. The PCI address and device number is stored in the PCI Error upper & lower address registers. (PCI_GROUP)
PERR parity error (read data)	PCI_PERR bit in the interrupt status register is set. The PCI address and device number is stored in the PCI Error upper & lower address registers. (PCI_GROUP)
SERR system error	PCI_SERR bit in the interrupt status register is set. The PCI address and device number is stored in the PCI Error upper & lower address registers. (PCI_GROUP)
***Crosstalk* Read Request from PCI device time-out.** This error indicates that a Crosstalk operation did not complete is the time allotted.	XREAD_REQ_TOUT bit in the interrupt status register is set. In addition the lower 48-bits of the PCI address, buffer number, and the PCI device number is stored in the Bridge Error Address Registers. (RESP_BUF_GROUP)

Table 58 PCI Slave Errors

6.5 GIO Errors

The only GIO errors occur when the *Bridge* is a GIO master.

Error Type	Logged Status
Master time-out. Either the Bridge or a GIO master timeout.	PCI_MASTER_TOUT bit in the interrupt status register is set. The GIO address and device number is stored in the PCI Error upper & lower address registers. (PCI_GROUP)
Non-contiguous byte enables in crosstalk packet	GIO_BENABLE_ERR bit in the interrupt status register is set. The GIO address is stored in the PCI Error upper & lower addres registers. (PCI_GROUP)

Table 59 GIO Master Errors

CHAPTER 7

Crosstalk Interface Unit

7.1 CIU Overview

The *Crosstalk* Interface Unit (CIU) is the section, of the *Bridge*, that interfaces the *Crosstalk* interconnect to the rest of the functions in the *Bridge* ASIC. The CIU consists of six parts; the LLP section, the *Crosstalk* receive processor (CRP), the *Crosstalk* transmit processor (CTP), the request generator (REQ_GEN), the prefetcher (PFETCH), and the request dispatcher section (REQ_DSP). The LLP provides low level transport mechanism for packet transmission. the CRP and CTP interface internal modules to the LLP and provide the clock boundary crossing. The REQ_DSP dispatches incoming request throughout the chip. The REQ_GEN and PFETCH generate crosstalk request from the bus masters. The CIU operates on the LLP in 8-bit mode only as defined in the *Crosstalk* System Interconnect Specification. Figure 9 contains a block diagram of the CIU.

The CIU controls two unique data flow paths, the *Bridge* ASIC initiating a transaction or the *Crosstalk* interconnect initiating a transaction. *Crosstalk* transactions can be broken down into two phases, a request phase and a response phase. The CIU uses different buffers depending on the operation and phase.

The *Bridge* ASIC has two clock regimes asynchronous to each other, one associated with the *Crosstalk* interconnect and one associated with the PCI/GIO bus. The CIU operates in both regimes.

Note: The term buffer in this section does not preclude operation of that device as a fifo to enhance performance.

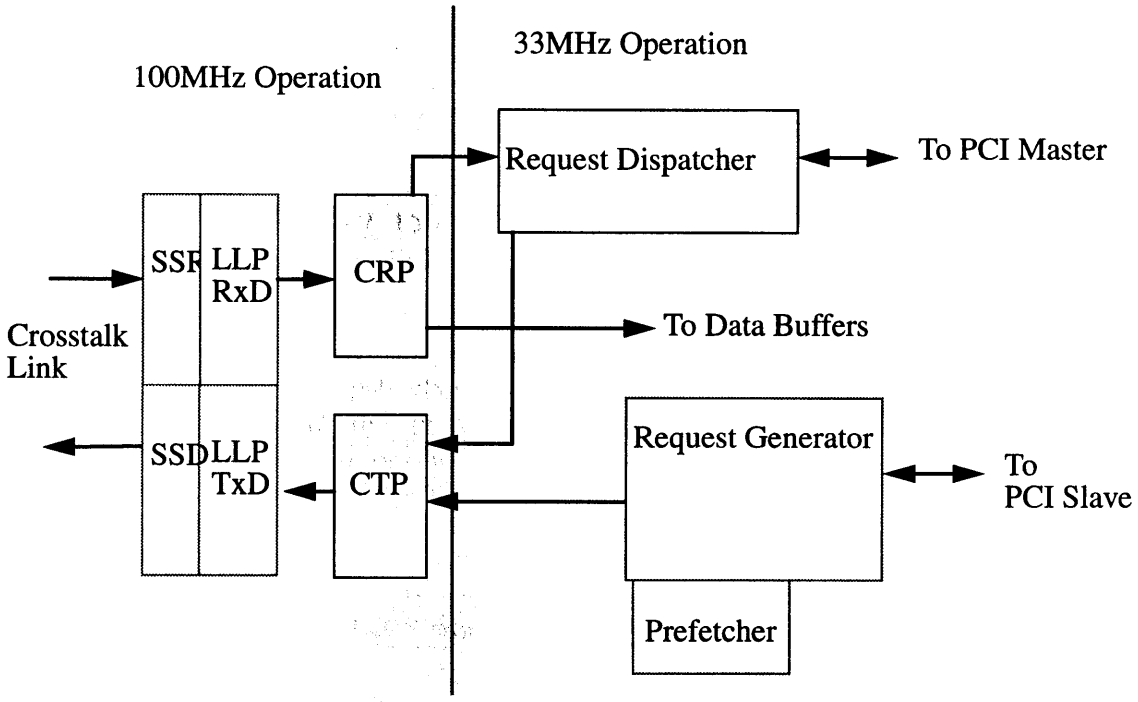


Figure 9 CIU Block Diagram

7.2 *Crosstalk* Data Ordering

The data transferred by a *Crosstalk* packet are arranged in multiples of double-words. The double-word read/write packets and quarter cache-line write packets have DE (Data Enable) bits to select the valid byte lanes. There is no restriction as of whether the enabled byte lanes have to be adjacent or not.

Hence, the data (byte) ordering is based on 64-bit double-words. The *Crosstalk* Specification defines a term, Data Section (DS), which is endian independent. The ordering of DS is shown in the figure below, along with the conventional endian-dependent byte ordering.

As shown in the figure, the DS is endian independent. DS0 represents bits (63:56) in a double-word; DS1 represents bits (55:48) in a double-word; and so on, regardless the endian modes. Also shown in the figure is the endian dependent term: Byte. Note that, for the big endian mode, DS0 is Byte0; DS1 is Byte1; and so on. On the other hand, for the little endian mode, DS0 is Byte 7; DS1 is Byte6, and so on.

The *Crosstalk* Specification also specifies a 32-bit word to contain DE (Data Enable) information for the double-word read/write packets and the quarter cache-line write packets. The formats of the DE bits for each of these types of packets are shown in the following figure.

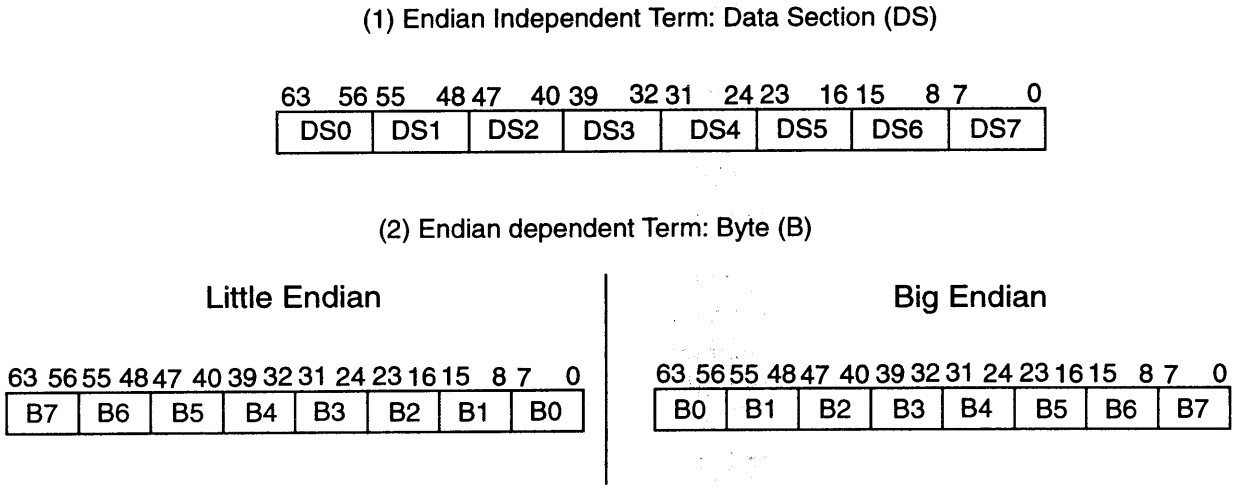


Figure 10 Crosstalk Double Word Data Ordering

The DE0 bit enables DS0; DE1 bit enables DS1; and so on.

For a quarter cache-line write packet, DE8 enables DS0 of the second double-word, which is also known as DS8. Similarly DE9 enables DS9, which is the DS1 of the second double-word. By the same token, DE16 enables the DS0 of the third double-word; DE24 enables the DS0 of the fourth double-word. These two Data Sections are also known as DS16 and DS24 of the quarter cache-line data block, respectively.

7.3 *Bridge* Initiated Transfers

The *Bridge* can initiate transfers from the following sources: the PCI bus, the GIO bus, the Request generator, and the Interrupt controller. All of these transfers are routed into the *Bridge* request buffer from different sections of the chip. Request transfers which require a response must first obtain a read response buffer.

As the request enters the *Bridge* request buffer, the request generator notifies the CTP that a request packet is ready to send. The CTP checks the current availability of buffer space in the crossbar and when a buffer is available for that packet type and priority level sends the packet.

When the *Bridge* generates a Crosstalk request, the tnum in that request is generates using the following rules:

- Read Requests, the tnum is the same as the read response buffer assigned, valid value are: 0x0-0xf
- PCI Write requests, the tnum matches the pci device number plus 0x10, valid value are: 0x10-0x17
- Interrupt Controller Write Requests, the tnum matches the pci device number plus 0x18 for the interrupt pins or 0x1f for error interrupts, valid value are: 0x18-0x1f

If the request required a response, the transaction number is generated from the response buffer number. The response buffers are responsible for maintaining the time-out bit corresponding with the *Bridge* request time-out counter to time-out over due requests. When a time-out occurs, the error information held in the response buffers and is transferred to the error registers and control section when the buffer is accessed by the PCI device. This mechanism provides a time-out value for a non-responsive widget.

When the response arrives it is routed through a pipeline and then to the response holding registers. The *Bridge* allows for sixteen outstanding read requests, with responses from those requests arriving back to back. This requires that the read response buffers be able to be written at the link rate.

7.4 *Crosstalk* Initiated Transfers

Crosstalk initiated transfers can access the following targets: PCI/GIO bus, Local SSRAM, FLASH PROM, some internal buffers, and internal registers. All *Crosstalk* requests are placed in the *Crosstalk* request buffer as they arrive. This buffer allows the request dispatcher access to the first 128-bits of the packet and the data enables. With this information the decoder can determine the destination within the *Bridge* for the packet.

If a *Crosstalk* request transaction required a response then the response packet is place in the *Crosstalk* response buffer. Again the link control handles sending the response. As in all *Crosstalk* widgets, the *Bridge* will always send *Crosstalk* responses ahead of *Bridge* requests.

7.5 *Crosstalk* Reset and Initilization

The *Bridge* ASIC has three reset pins; LLP_RST_N, PLL_RST_N, SIM_RST_N. Only two reset pins, PLL_RST_N and LLP_RST_N are required for proper operation. SIM_RST_N is for proper simulation operation and should be pulled up. PLL reset is used to hold the plls in reset until power and the 400MHz input clock is stable. PLL_RST_N the first reset to be deasserted. LLP reset is fed to the cold reset of the LLP. LLP_RST_N is deasserted after enough time for the PLL to settle. The core logic reset is driven from the link reset out of the LLP section. This line is deaserted some time after LLP_RST_N is deasserted and link connection has been determined.

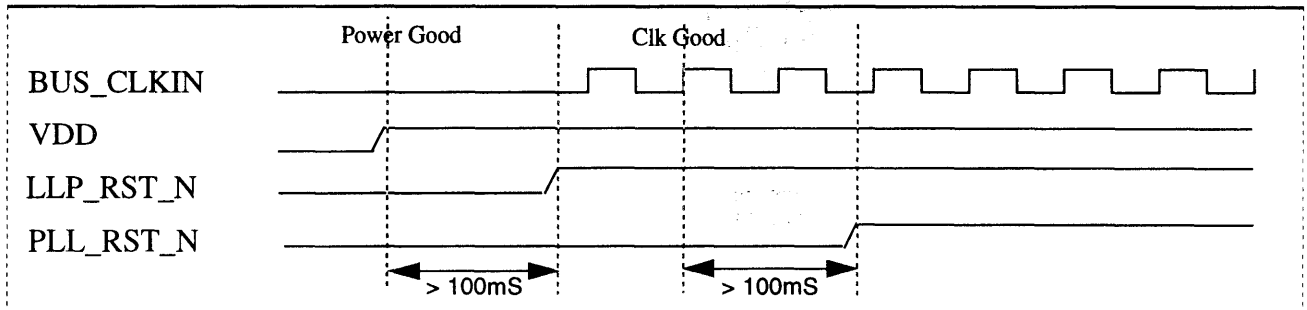


Figure 11 Reset Sequence

CHAPTER 8

PCI Interface Unit and It's Operation

PCI Interface Unit interfaces between PCI Bus and PCI/GIO Data Buffer Unit. A PCI bus state machine can behave as both a master or a slave of the bus. To be a PCI bus master, it has to get the bus grant from the bus arbiter which also resides in the Bridge ASIC chip. At the other side of the unit is the Data Buffer Unit which accepts commands/data from PCI state machine, packs them into a packet and sends them to the Crosstalk Interface Unit (CIU). It also receives command packets from CIU and generates the corresponding PCI commands to the PCI bus devices. First, a little description of PCI Bus.

8.1 PCI Bus Operation

The PCI Local Bus is a 32-bit or 64-bit bus with multiplexed address and data lines. The bus protocal is very similar to GIO-32 or GIO-64 bus. The Bridge chip, when operated in PCI mode, supports both 32-bit and 64-bit data buses and 32-bit address mode as a master and both 32-bit and 64-bit address modes as a slave. Following is a brief description of PCI commands, addresses, bus operations, and arbitration schemes. Refer to PCI Local Bus Specification Revision 2.1 for detailed information.

8.1.1 PCI Commands

PCI bus has shared command and ByteEnable bus, C_BE_N[7:0]. During the first cycle of the transfer, the lower four bits define the type of transfer. After the first cycle the bus represents the ByteEnable of the corresponding byte on the AD[63:0] bus. C_BE_N is active low when it represents Byte Enables. A '0' means the byte is asserted. The definitions of the commands are in **Table 60**.

C_BE_N[3:0]	Command Types	Comments
0000	Interrupt Acknowledge	Bridge Generated Only
0001	Special Cycle	Not used
0010	I/O Read	
0011	I/O Write	
0100	Reserved	
0101	Reserved	
0110	Memory Read	
0111	Memory Write	
1000	Reserved	
1001	Reserved	
1010	Configuration Read	Bridge Generated Only
1011	Configuration Write	Bridge Generated Only
1100	Memory Read Multiple	Same as READ
1101	Dual Address Cycle	This allows 64-bit address.
1110	Memory Read Line	Same as READ
1111	Memory Write and Invalidate	Same as WRITE

Table 60 **PCI Commands**

PCI bus has three physical address spaces, the memory and I/O spaces and configuration space. In the I/O address space, all 32 AD[31:0] are used to provide a full byte address. This allows a byte addressable device

to claim the cycle without waiting one extra cycle for ByteEnables. The device then checks ByteEnables to finish the operation. In the memory address space, AD[31:3] are used to address at the double word boundary in 64-bit mode and AD[31:2] are used to address at word boundary in 32-bit mode. Bridge will only support linear incrementing of address (when AD[1:0]="00") in a memory comand. If AD[1:0]="10" (cache line wrap mode where critical word/double word first) or AD[1:0]="X1", Bridge disconnects after one data phase.

8.1.2 PCI Basic Operations

The basic PCI operation is defined by C_BE_N[7:0[(command and byte enables), FRAME_N(beginning of command and data cycles), IRDY_N (initiator/master ready), TRDY_N (target/slave ready) and DEVSEL_N (target acknowledge of address decode). AD[63:0] has address and data.

8.1.2.1 Write Transaction

In a normal write cycle, the master/initiator asserts FRAME_N with address on AD and command on C_BE_N[3:0] in the first cycle. From second cycle, IRDY_N indicates the data is ready and C_BE_N[7:0] contains the byte enable information from the master. When TRDY_N from the target is asserted, the cycle is finished. If more data is to be sent by the initiator, the master can keep IRDY_N active and put more data on the AD bus. The slave device uses TRDY_N to receive more data. The write operation is shown in Figure 12. Also shown in the figure are PAR (even number of 1's of AD[31:0], C_BE_N[3:0] and PAR, one clock after AD valided by IRDY_N), SERR_N (address and command parity error one clock after PAR) and PERR_N (data parity error one clock after PAR). In 64-bit mode, PAR64 is the even parity bit to protect AD[63:32] and C_BE_N[7:4].

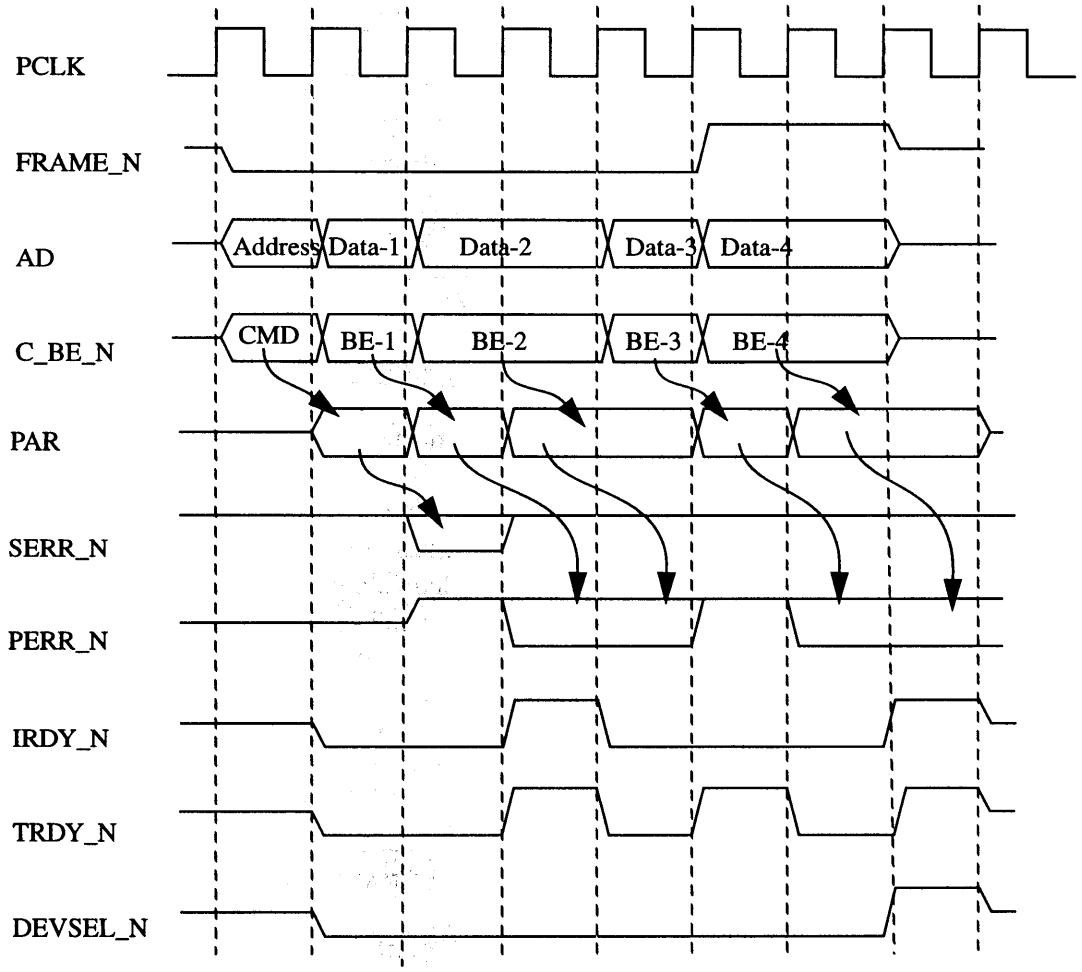


Figure 12 PCI Write (32-bit address)

8.1.2.2 Read Transation

A PCI read cycle is similar to a write cycle except that data and PAR are driven by the target. After the initiator drives the AD bus with address in the first cycle enabled by FRAME_N, it tri-states the AD bus from the next cycle. The target will drive the bus one cycle later. A turn around cycle is here to prevent bus contention. C_BE_N bits are always driven by the master in a read cycle. PCI spec allows C_BE_N's to change every data cycle, even though it may not be too useful. As a master, Bridge will

pass the byte enable information from the read packet to the target. It will ignore the C_BE_N's and return all data as a slave in a read cycle.

For both read and write operations, FRAME_N should be deasserted when the last IRDY_N will be or is being asserted. This allows other bus master to start the operation early.

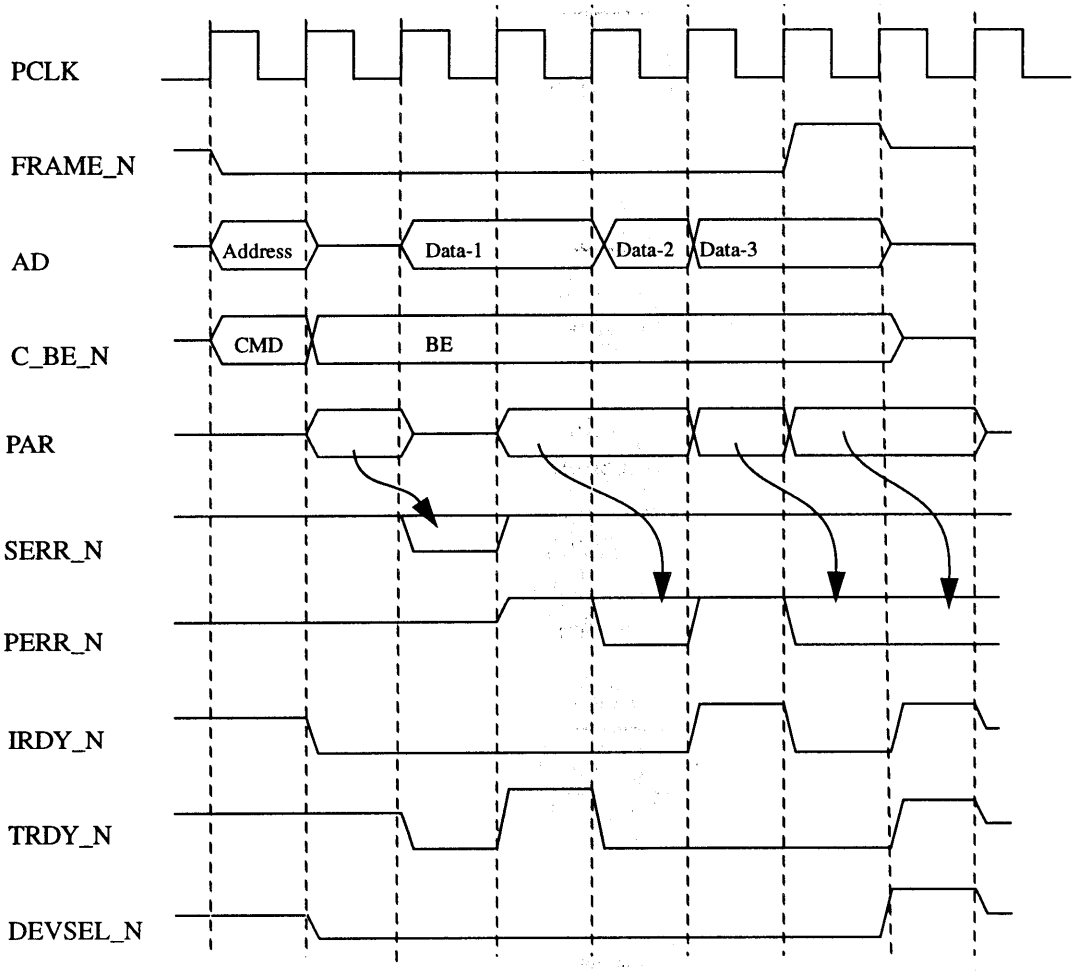


Figure 13 PCI Read (32-bit address)

8.1.3 Termination

The PCI operation can be terminated by both the master (initiator) or the target (slave).

8.1.3.1 Master Termination

The master terminates the PCI cycle by deasserting FRAME_N and asserting IRDY_N. This signals the target that it's the final data phase of the cycle. The cause for it can be either 1)Completion of the cycle or 2)Timeout when GNT_N from the arbitration circuit is deasserted or its internal latency timer has expired. A third kind of master termination is Master Abort. This is due no response from the target after a certain clocks. The master will terminate with a master abort (it finishes the cycle when TRDY_N is still deasserted) and issues an interrupt to the host.

8.1.3.2 Target Termination

Normally, a cycle is always finished by the master. A target has to use a separate signal, STOP_N, to finish the cycle early. The target termaination can be either 1)Disconnect or 2)Retry. Disconnect is because the target may not respond to the request in fast enough time yet the data just transferred has been received. A retry from the target tells the master that it is busy, can not receive any data or it will take a long time to fetch data. It asks the master to retry the same command some time later. Retry means the target has not received any data, the master has to save the data being sent when it sees a retry. Bridge chip will signal retry to a PCI master when it tries to read data from host or a remote target. A third kind of target termination is Target Abort. It happens when DEVSEL_N is deasserted at the same time STOP_N is asserted. This indicates the target requires the transaction to be terminated and doesn't want the transaction tried again. A target abort interrupt will be issued.

8.1.4 PCI Arbitration

Each PCI master has a dedicated REQ_N to the centralized arbiter and each receives its own GNT_N from the arbiter. When a PCI device wants to be a master of the bus, it asserted its REQ_N. When it sees its GNT_N is asserted and both FRAME_N and IRDY_N are deasserted, the device can drive the bus and start its own transaction. The following figure is borrowed from the PCI spec to show how two PCI agents request the bus and get access to the bus. The bus request is on a per transaction basis. Device does not need to relinquish the bus even when its GNT_N is not driven active after it asserts FRAME_N. It has to release the bus when GNT_N is high and the device has owned the bus longer than the time defined in the Master Latency Timer.

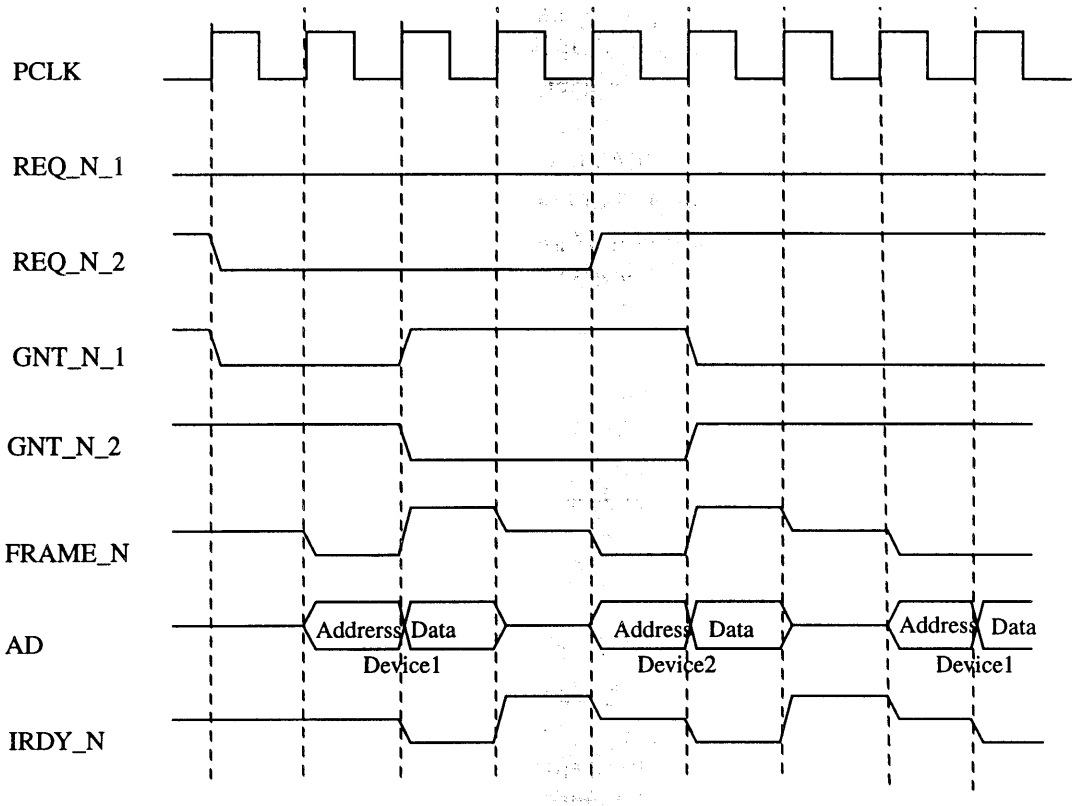


Figure 14 PCI Arbitration

8.2 Unsupported PCI Features

The PCI Bus is defined to support a lot of system configurations so that it may be used as CPU/memory local bus or it can interface to a remote I/O peripheral. Consequently, not all the features are needed in the Bridge. Following are the list of the features not supported in this chip.

1. Cache suppert: Bridge does not snoop anything. No SDONE and SBO_N pins.
2. No cache line toggle mode on memory command address. The address is always linear incrementing and the 2 lsb's of the address should always be "00".
3. Memory Read Line command and Memory Read Multiple command: Bridge can only interpret these commands as slave. It does not generate these commands as a master.

4. Memory Write and Invalidate command: No cache line size register implemented. A Memory Write and Invalidate command will be treated as a normal Memory Write command.
5. Built-in Self Test: Bridge needs external patterns to test it.

8.3 *Bridge* PCI Operations

Behaving as a PCI bus master, Bridge allows the host and other widgets on the Crosstalk Bus to read (memory read, I/O read and configuration read) or write (memory write, I/O write and configuration write) any devices on the PCI Bus. As a slave, Bridge will only allow PCI devices to do memory read or memory write widgets behind Crosstalk bus. All other commands will not be responded by not asserting DEVSEL_N.

8.3.1 Crosstalk Writes PCI

When a write packet is sent from Crosstalk to PCI, it goes through the Crosstalk Map circuit to generate 32-bit PCI address and stores the new command in the write buffer of the PCI/GIO Data Buffer. Once the PCI state machine detects the outstanding write command at the buffer, it requests for the bus, waits for the grant signal from the arbiter and sends the data to the destination. The data at the Data Buffer is 64-bit wide with correct target endianness. The PCI control circuit checks the size of the target device to send correct data size. If write response packet is requested, a write response packet will be sent to the Crosstalk Interface Unit at the end of the write.

If the target "retry" the command, Bridge will release the bus back to the arbiter. The commands in the Data Buffer will stay the same until the first command is completed. When the number of retry of the packet is more than the number specified by PCI_RETRY_CNT[3:0] in PCI/GIO Timeout Register, the packet is discarded and a PCI_RETRY_CNT interrupt is issued. This retry timeout mechanism applies to both write and read.

8.3.2 Crosstalk Reads PCI

The Crosstalk read request packet goes through the same path as the write request packet. The order of write and read requests are maintained in the PCI/CIO Data Buffer. A read command is sent to the PCI bus when PCI state machine has the bus. The read back data will be stored in the read response buffer. When all the data are gathered, a read response packet is sent to CIU. Bus size conversion is done at the PCI data input

side if the targeted device is 32-bit wide. If Crosstalk is big endian, bit[63:32] is the low addressed word and bit[31:0] is the high addressed word. If Crosstalk is little endian, bit[31:0] is the low addressed word and bit[63:32] is the high addressed word. Byte swapping to match PCI device with the Crosstalk bus is performed when the data is stored into the Data Buffer.

8.3.3 Interrupt Acknowledge

Since Bridge knows which pin the interrupt occurs and it tells host which pin interrupts , host does not need to poll an interrupt controller for the identification of the interrupting device. If an interrupt controller is used in the system, host writes the interrupt acknowledge register to generate an Interrupt Acknowledge command to the PCI Bus. A read to the register will return the 8-bit data read from the interrupt controller.

8.3.4 PCI Writes Crosstalk

When a PCI device owns the bus, Bridge behaves as a slave to pass data from the PCI bus to the Crosstalk Bus in a write case. To maximize the throughput at the host, write gathering feature is supported with PCI writes. It allows the PCI devices not to transfer data in 128 bytes quantity when they want to send a large block of data to the host. With FRAME_N active on the bus, the Bridge PCI slave state machine follows the following rules to respond to a write command:

1. If address does not match Bridge's direct or pmu external space or command is not memory read/write (or read multiple, read line, or write invalidate), do nothing by not driving any signals
2. If address match and device is a real time device, then accept data by driving TRDY_N if there is free data buffer or "retry" the device if not data buffer is available (this scenario should not happen). (All cases follow have address in Bridge address space.)
3. (Non-real time device) If write gather is not enabled, accept data if more than 2 data buffers are free or retry the device. (Each device in its device register has two bits to enable write gathering in direct mapped space and page mapped space.)
4. (Non-real time device) If write gather is enabled, the slave fsm checks if there is a partially filled data buffer assigned to the device. When the incoming address and byte enable bits match that of the expected address and byte enables, new data is accepted and appended to the end of previous transfer in the same data buffer. If address or byte enables do not match, Bridge retry the device and

mark the buffer "done" and data buffer circuit will send it to CIU. Each device can have at most one partially filled write buffer. If there is no partially filled buffer, the fsm check whether more than 2 free data buffers are available to receive new data. Retry the master if not enough buffers or accept data with 3 or more free buffers.

The data stored in the PCI_RW_Req Data Buffer are in 64-bit size. When the PCI device sends data in 32-bit size to the host, 32-bit word to 64-bit double word collection is done here. If Crosstalk is big endian, low addressed word is stored at bit[63:32] and high addressed word is sent to bit[31:0]. If Crosstalk is little endian, low addressed word is sent to bit[31:0] and high addressed word is sent to bit[63:32].

Each data buffer is 128 bytes. The PCI slave fsm will stop the master (called disconnect) and/or mark the buffer "done" to flush the buffer following these rules:

1. Always disconnect the master when next data will reach the end of the buffer. Then mark done to the buffer.
2. Always disconnect the master when any byte enable bits of current quarter cache line is deasserted. (This limitation is due to each data buffer has only one 32-bit DE register.) The bits representing the bytes between the quarter cache line address and the starting address are deasserted at the beginning of the data transfer.
3. For a partially filled buffer where write gathering is enabled for the device, the buffer will be flushed if any one of the following four conditions happen (besides the end of buffer and byte enable limitations):
    1. New address does not match expected address
    2. The device does a read
    3. The device generates an interrupt
    4. Host flushes the device write buffers

8.3.5 PCI Reads Crosstalk

A PCI read to the Crosstalk is the most complex part of the circuit and it affects the system performance the most. Multiple PCI read response buffers function as prefetch cache are provided in the PCI/GIO Data Buffer to maintain a high throughput for PCI reads. When a PCI master reads data behind the Crosstalk bus, the PCI state machine checks with the PCI read response buffer to see if the data is already residing in the buffer (by previous prefetch) by checking the address and the valid bit in the buffer tag. If the data is ready, the state machine passes the data from the buffer to the bus until the bus master terminates the operation or when

the data in the buffer are all sent out. If the PCI state machine found no data in the buffer, it "retry" the master and sends a read request command through PCI_RW_Req buffer to PMU and CIU. It will then mark the Request_In_Progress bit in the buffer tag. The third case is after the comparison, the address is correct, the Valid bit is off and the Request_In_Progress bit is on, that means a read command had been sent out moments early, PCI state machine still issues "retry" to the PCI device but it generates no new read request command.

8.4 *Bridge* PCI Arbitration

The arbitration circuit for PCI is identical to that of GIO. The arbiter supports two priority rings, high priority (real time) and low priority. Each ring has a round-robin priority scheme with the last device owning the bus with the lowest priority. The REAL_TIME bit in the device register defines which ring the device is in. Devices in the high priority ring (REAL_TIME=1) always get the bus before the lower priority devices. If a low priority device currently owns the bus and a high priority device requests for the bus, the grant to the low priority one will be deasserted and the bus will be granted to the high priority device. The device will release the bus when its Master Latency Timer expires. However, a device in high priority ring will not be removed from the bus by another high priority device until it has finished a certain number of cycles. To prevent a high priority device from getting on and off the bus all the time and starving the lower priority devices when it gets "retry" in PCI mode, host can program REQ_WAIT_EN in ARB_PRIORITY register to force the arbiter to wait for 4 ticks of 8/16/32 PCI clocks time (defined by REQ_WAIT_TICK[1:0]) before it issues a new grant to the same device.

Following figure shows the priority rings arrangement of the PCI devices and the Bridge.

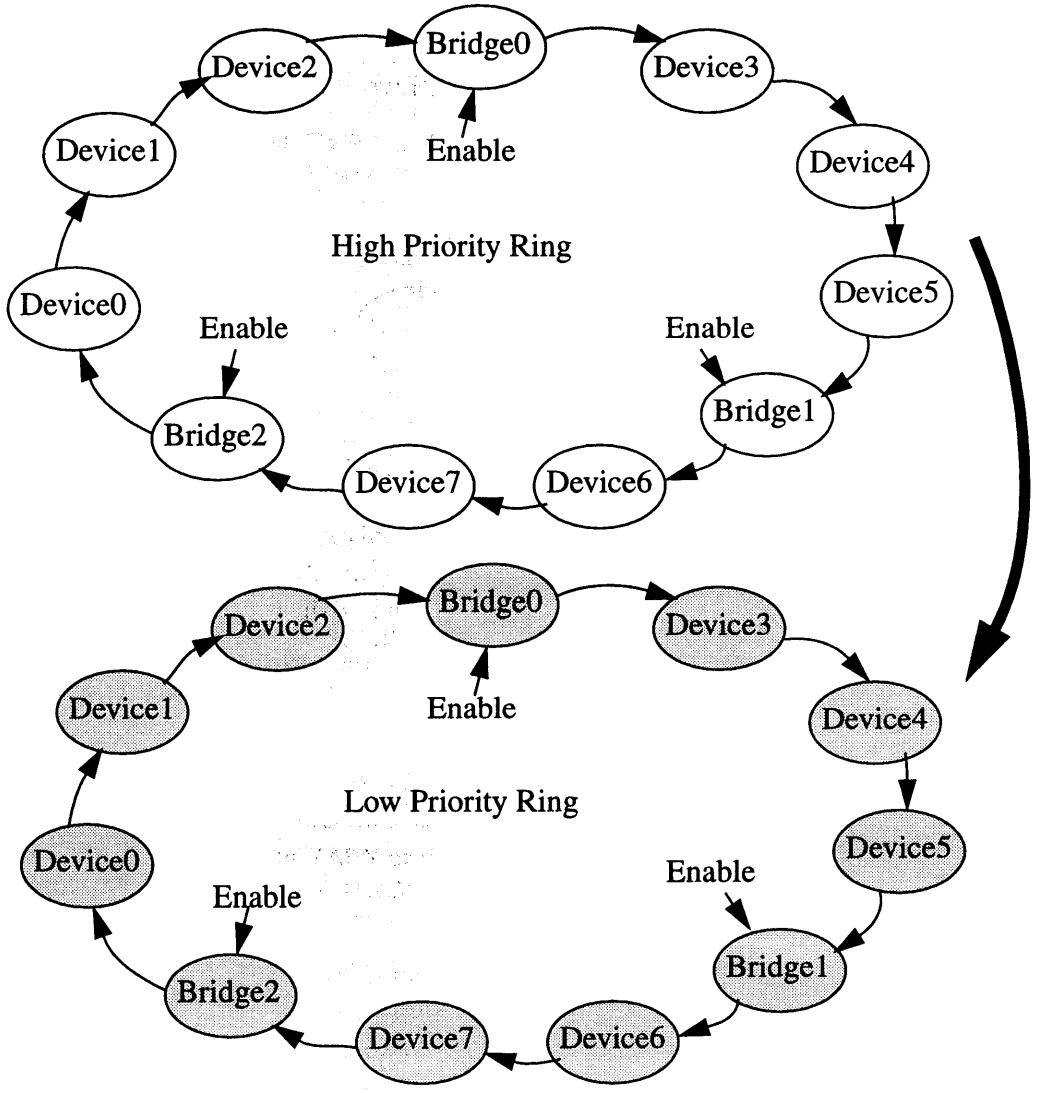


Figure 15 Arbitration Priority Rings

Eight PCI devices and the Bridge can all request to be the master of the bus. The devices request the bus with their own REQ_N pins. PCI state machine also generates request to the arbiter to request the bus. The arbiter follows the mode in ARB_PRIORITY register and the relative position of the devices on the ring to assign bus mastership.

Here is an example of bus arbitration. If no devices are assigned to high priority ring and Device2 currently has the highest priority, then the priority order is Device2, Bridge (if EN_BRIDGE_LO[0]=1), Device3, Device4, Device5, Bridge (if EN_BRIDGE_LO[1]=1), Device6, Device7, Bridge(if EN_BRIDGE_LO[2]=1), Device0, Device1. So when REQ_N_4 is asserted and internal Bridge requests for the bus at the same time, Device4 will get the bus if EN_BRIDGE_LO[0]=0 or Bridge gets the bus if EN_BRIDGE_LO[0]=1.

To guarantee each device has a fair share of the bus bandwidth, the devices are assigned the bus in a round robin manner. The device being granted the bus will be assigned the lowest priority after it owns the bus. In this case, if Device4 is granted the bus now (and no matter what device has the highest priority previously), Device5 will have the highest priority next and Device4 will have the lowest priority. Multiple Bridge requesting points in the priority rings provides flexible bus master time for the internal Bridge circuit. With all three bits of EN_BRIDGE set to 1's, Bridge will be offered more bus grants by the bus arbiter.

Things get a little complex when both priority rings are used. In the examples below, assuming Device0, Device 2, Device4 and Bridge1 are set in the high priority ring (Device1, Device3, Device5, Device6 and Device7 are automatically assigned to the low priority ring), and Bridge0 is also set in the low priority ring, the arbiter will grant the bus as two examples shown below: Example 1) If Device3 has the bus now and Device4 has the highest priority in the high priority ring, Device5 will have the highest priority in the low priority ring next. When Device2 and Device5 request for the bus, arbiter will stop the grant to Device3 immediately and issue bus grant to Device2. Once Device2 owns the bus, the next highest priority device in the high ring is Bridge1 and it is still Device5 has the highest priority in the lower ring. Example 2) If Device0 is the bus master now (it does not matter which one has highest priority previously and Device2 will have the highest priority next) and Device5 has the highest priority on the lower ring, the priority order is Device2, Device4, Bridge, Device0, Device5, Device6, Device7, Device1 and Device3. Device0 will be able to finish its cycle until other high priority devices request the bus and the timer reaches end count.

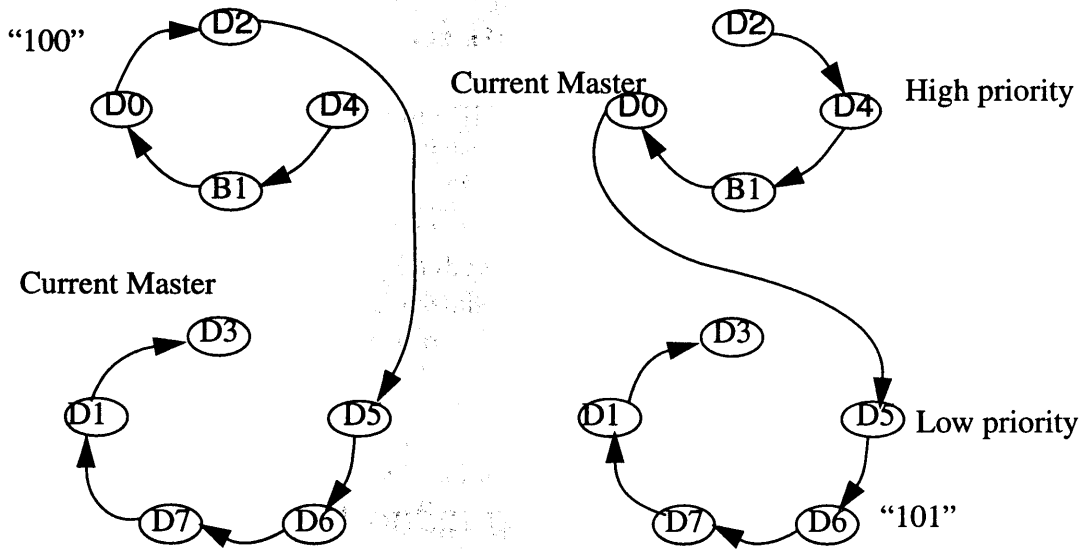


Figure 16 Arbitration Examples

8.5 *Bridge* PCI Interrupt

Same as in the GIO bus configuration, Bridge accepts eight active-low INT_N pins. There is no restrictions on which pin should be assigned to which device. The connection of INT_N pins with PCI devices are system implementation dependent. Bridge will just report to host what is happening on the interrupt pins. In PCI mode, Bridge will also report the status of Crosstalk bus Address/Command parity error, Crosstalk bus data parity error, Crosstalk bus time out, PCI address/command parity error, PCI data parity error, PCI master/salve abort. Refer to the Interrupt chaper for a detailed explanation of interrupt registers and operation.

8.6 PCI Configuration Space ID Select

The bridge address lines 24:31 and 32:39 as the ID selects during the configuration cycles. The table below shows the configuration space to address line correspondence. Note: The bridge duplicates the selects on the upper word lines as well. Either of the lines will work, the duplication is done to minimize loading across different system configurations.

Device Number	Lower Word Address Line	Upper Word Address Line
0	24	32
1	25	33
2	26	34
3	27	35
4	28	36
5	29	37
6	30	38
7	31	39

Table 61 **Configuration ID Select Lines**

CHAPTER 9

GIO Bus Interface Unit

The GIO Bus Interface Unit interfaces between GIO Bus and PCI/GIO Data Buffers. The GIO Bus state machine can behave as both a master or a slave of the bus. In GIO Bus master mode, GIU requires a bus grant from the bus arbiter which also resides in the Bridge ASIC chip. At the other side of the unit is the Data Buffer Unit which accepts commands/ data from GIO state machine, packs them into a packet and sends them to the Crosstalk Interface Unit (CIU). The GIU also receives command packets from CIU and generates the corresponding GIO commands to the GIO bus devices.

9.1 Bridge GIO Bus Operations

The Bridge ASIC, when operating in GIO mode, supports only pipelined GIO64 version of the GIO specification. No external GIO Bus pipeline registers are needed between a pipelined GIO device and the Bridge ASIC because Bridge has them included then internal to the chip. The outside GIO devices just need to follow the GIO Bus protocol to communicate with Bridge ASIC. Refer to GIO Bus Specification for detailed bus operation.

Behaving as a GIO bus master, the Bridge ASIC allows the host and other widgets on the Crosstalk Interconnect to read or write any devices on the GIO Bus. Since the GIO Bus has only one address space, all addresses on

the GIO Bus are memory mapped. This is one of the differences between GIO and PCI Buses. As a slave, Bridge responds to other GIO Bus masters to access data in the Crosstalk address space where the data may be in main memory or at another PCI/GIO device behind a Crosstalk widget.

9.1.1 Unsupported GIO Bus Functions

Since Bridge is used as an I/O bus controller instead of a memory controller, some GIO functions of previous system are not supported here in the Bridge:

- DMA: Bridge does not support any master DMA functions
- GIO-32: all versions, GIO-64 non-pipelined
- Subblock ordering: no decrementing addresses
- No Parity Support
- Big Endian Only
- No Preemption
- 32-bit Addressing only
- Transfer larger that double word from crosstalk are not allowed
- Single GFXDLY only

9.1.2 Crosstalk Initiated Writes

When a write packet is sent from the Crosstalk Interconnect to the GIO bus, it goes through the Request Generator generating a 32-bit GIO address and stores the new command in the write buffer of the PCI/GIO Data Buffer. Once the GIO state machine detects the outstanding write command at the buffer, it requests for the bus and sends the data to the destination. The data at the Data Buffer is 64-bit wide with correct target endianness. The GIO control circuit checks the size of the target device to send correct data size. If write response packet is requested, a write response packet will be sent to CIU at the end of the write.

Other widgets on the Crosstalk Interconnect which initiate data transfers should guarantee that the targeted device can finish the data transfer before it acquires the bus, or a deadlock may occur. Since a GIO device should always respond to a PIO command as a slave, the command initiator does not need to check the status of the targeted device before it starts a PIO transfer. (Writing double-word data to a 32-bit device is not considered a PIO command.)

9.1.3 Crosstalk Initiated Reads

The Crosstalk read request packet goes through the same path as the write request packet. The order of write and read requests are maintained in the PCI/GIO Data Buffer. A read command is sent to the GIO bus when GIO state machine has the bus. The read response data will be stored in the read response buffer. When all the data for the request is gathered, a read response packet is sent to the Crosstalk Interface Unit.

Widgets initiating GIO read commands should guarantee that no deadlock will occur when it reads multiple data cycles.

9.1.4 GIO Initiated Writes

When a GIO device is bus master, Bridge behaves as a slave to fetch data from other Crosstalk Widgets. The GIO state machine first decodes the command and the address on the bus. When it detects the address is within its range, it deasserts MEMDLY to acknowledge that it can receive data in the next cycle or two cycles later. It will then send the address to the Bridge PMU to do the address mapping parallel with the receiving of the data. The data is stored in the GIO Write/Read Data Buffer before it is sent to the Crosstalk Interface Unit together with the new address from the PMU circuit.

9.1.5 GIO Initiated Reads

A GIO read to the Crosstalk Interconnect is the most complex part of the circuit and it affects the system performance the most. Since GIO Bus has no "retry" protocol, the requesting device will hold the bus until its request is fulfilled, and a read to a Crosstalk widget can take some time to finish. Besides the long latency for the read, bus idling time should be considered in calculating the bus bandwidth. To maintain the bus at a high throughput for GIO reads, multiple GIO read response buffers function as prefetch cache are provided in the PCI/GIO Data Buffer. When a GIO master reads data behind the Crosstalk bus, the GIO state machine checks with the GIO read response buffer to see if the data is already residing in the buffer (by previous prefetch) by checking the address and the valid bit in the buffer tag. If the data is ready, the state machine sends data to the bus until the bus master terminates the operation or when the data in the buffer are all sent out. If the GIO state machine found no data in the buffer, it sends either a read command or a prefetch command to Bridge PMU and Crosstalk Interface Unit. It will then mark the Request_In_Progress bit in the buffer tag. If the address is correct, the Valid bit is off and the Request_In_Progress bit is on, that means a read

command is in flight. The GIO state machine will just wait for the data to come back.

9.2 *Bridge* GIO Arbitration

The arbitration circuit for GIO is identical to that of PCI. Refer to PCI Arbitration in PCI Chapter for detailed information. Note that the arbitration priority in Bridge is different from previous GIO Bus arbiter. It follows a two-ring order according to device number and its priority assignment. The devices can be programmed to a fixed (ring) order or the new bus master will be assigned the lowest priority to allow other devices to own a fair portion of the bus.

9.3 *Bridge* GIO Interrupt

Same as in the PCI bus configuration, Bridge accepts eight active-low INT_N pins. There is no restrictions on which pin should be assigned to which device. The connection of INT_N pins with GIO devices are system implementation dependent. Bridge will just report to host the activity from the interrupt pins.

CHAPTER 10

SSRAM / Flash PROM Control

The *Bridge* supports synchronous static random access memory (SSRAM), and Flash PROM. Local SSRAM is used to support PMU descriptor for PCI/GIO bus devices. The SSRAM control section supports memory configurations from 64KBytes to 512KBytes. Configuring SSRAM is done in the widget control register. Only the *Crosstalk* Interface and the Request Generator PMU can access this memory. The same interface also supports a flash PROM which is large enough to include boot and driver software. The flash PROM can be written from the *Bridge* ASIC.

10.1 SSRAM

10.1.1 SSRAM Size

The SSRAM is located at address 0x0000_0008_0000 in widget space. Table 62 shows the possible configuration and sizes supported by the *Bridge* ASIC.

Memory Size	SRAM	Parity	Technology
0			
32K x 16	(1) 32K x 16	NO	1MBit
32K x 18	(1) 32K x 18	YES	1MBit
64K x 16	(1) 64K x 16	NO	1MBit
64K x 18	(1) 64K x 18	YES	1MBit
256K x 16	(1) 256K x 16	NO	4MBit
256K x 18	(1) 256K x 18	YES	4MBit

Table 62 SSRAM Configurations

10.1.2 SSRAM Control Bits

SSRAM options are selected by local SSRAM size, parity enable, and force bad parity control bits in the *Bridge* control register. The local SSRAM size bits determines the SSRAM configuration.

The parity enable bit is used to enable checking of data during read operations from SSRAM. Parity is always generated by the SSRAM control section on writes. Bad parity can be generated by setting the Force Bad Parity bit. If a parity error occurs, the 16 bits of SSRAM address, byte(s) in error bits, the source section and the pci device requesting mapping access are stored in the SSRAM parity error register. If multiple parity errors occur, the SSRAM parity error register stores the first error and holds that value until cleared. The write does not change the contents of the register but rearms the storing mechanism. An interrupt is also generated by a parity error. The parity error generates a widget "other error" class

interrupt to the host device. See the Error Cases Section under SSRAM errors.

10.1.3 SSRAM Operation

The SSRAM operation section defines the timing used to access the SSRAM. The SSRAM supported by the *Bridge* must have registered output and a burst counter.

The SSRAM is accessed from the *Crosstalk* interconnect by a 64-bit write or a 64/32-bit read. Writes must be a complete double word to allow programming the entire address translation entry. Any other accesses to the SSRAM from the *Crosstalk* interconnect will generate an error. The PMU will access the SSRAM with a 64-bit read only.

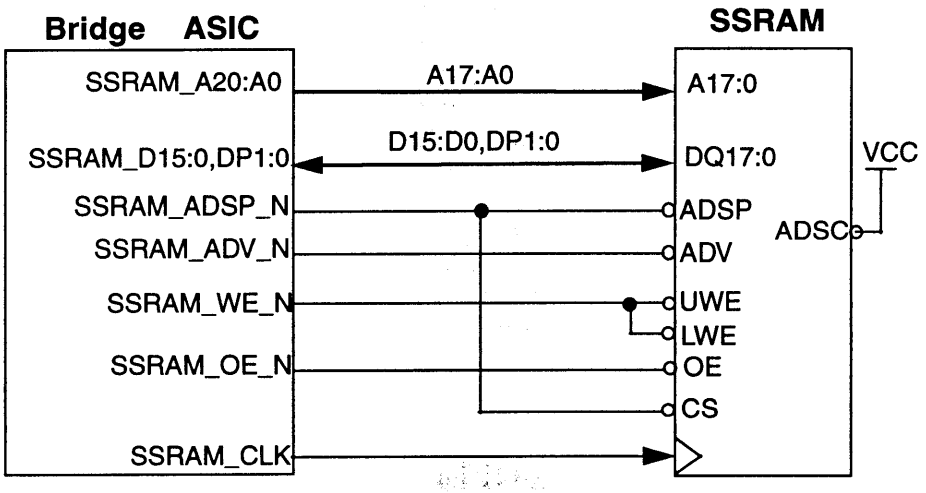


Figure 17 SSRAM Implementation

10.1.4 SSRAM Read Cycle

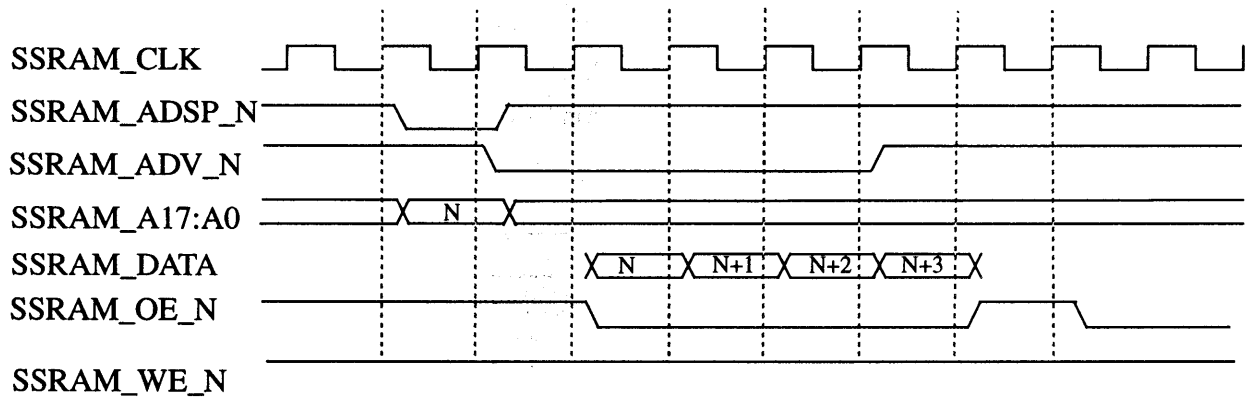


Figure 18 SSRAM Read Timing

10.1.5 SSRAM Write Cycle

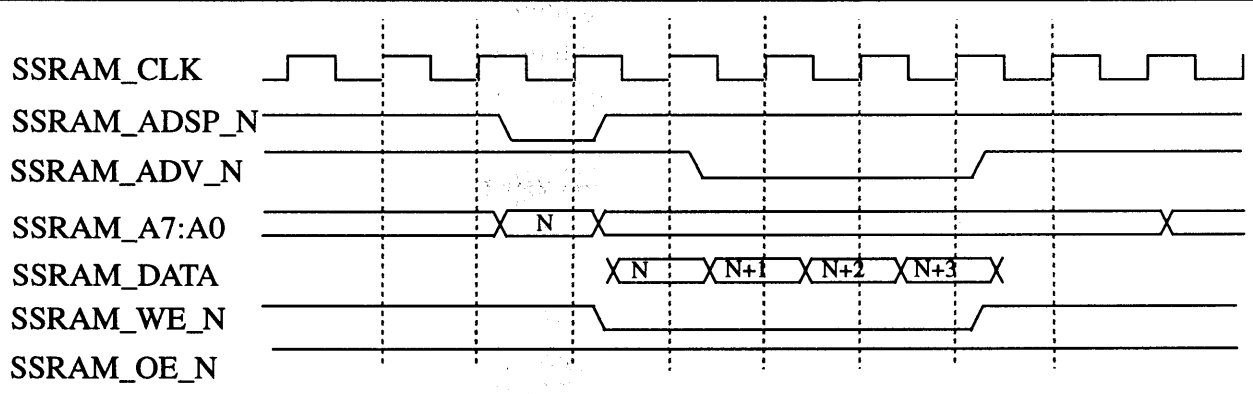


Figure 19 SSRAM Write Timing

10.1.6 SSRAM Requirements

The requirements of SSRAM are:

- 15 ns Clock Time/ 66MHz
- Registered Inputs (Address,Data,Control)
- Registered Outputs (Data)
- Burst Counter

- Fully Static
- 3.3V I/O compatible
- Moto MCM67C518 (32Kx18) 9nS
- Moto MCM67C618 (64Kx18) 9nS

10.2 Flash PROM

10.2.1 Flash PROM Operation

The *Bridge* supports connecting a FLASH PROM to the SSRAM port. The target flash device is a sector erasable part, either 256K x16 or 512K x 16. Flash devices are protected via a checksum method and therefore parity circuits on the *Bridge* are disabled during flash PROM access. The *Bridge* allows only *Crosstalk* operations to access the flash, reads can be any valid *Crosstalk* read request; writes are 16-bit only. The restriction on write size is due to the various programming methods used by different flash vendors. Also to write the flash the FLASH_WR_EN bit in the control register must be set.

The *Bridge* supports two flash devices, 2MBytes each, with different chip selects. A pin on the *Bridge* forces swapping of the 2 sections. This allows the use of both a flash device and a PROM socket to support field flash programming errors.

Shown in Figure 20 is the connection of flash PROM to the SSRAM interface for ID PROM option.

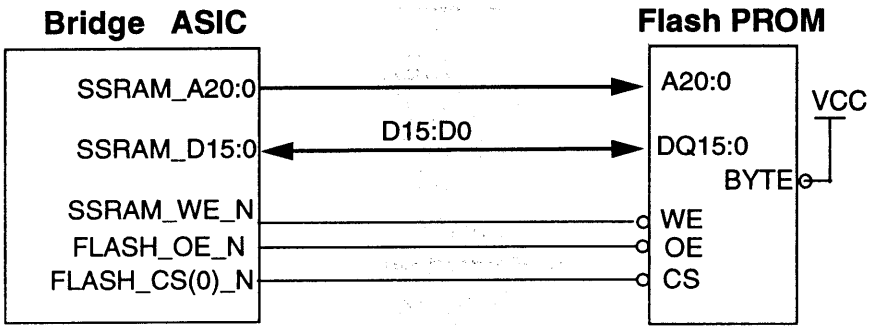


Figure 20 Flash PROM Implementation

10.2.2 Flash PROM Read Cycle

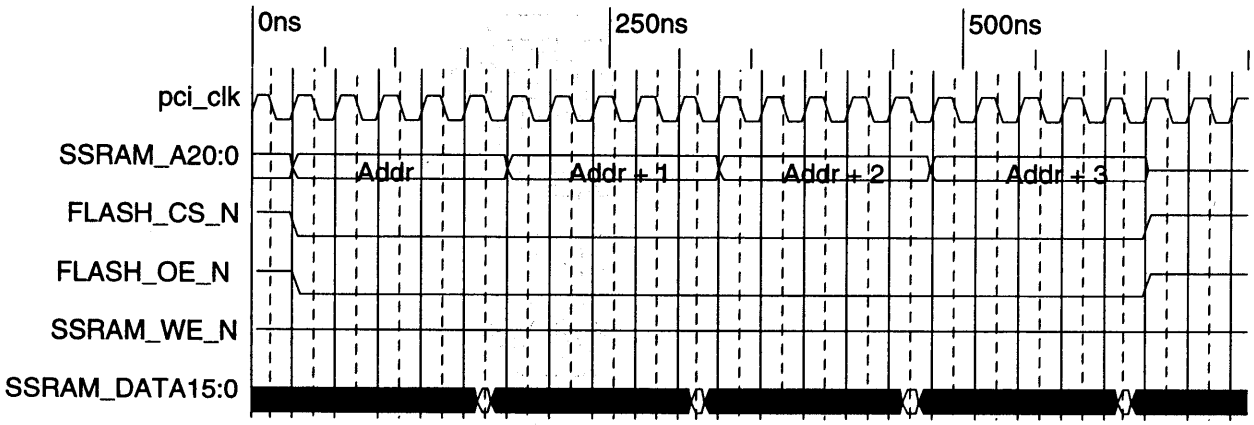


Figure 21 FLASH Read Timing

10.2.3 Flash PROM Write Cycle

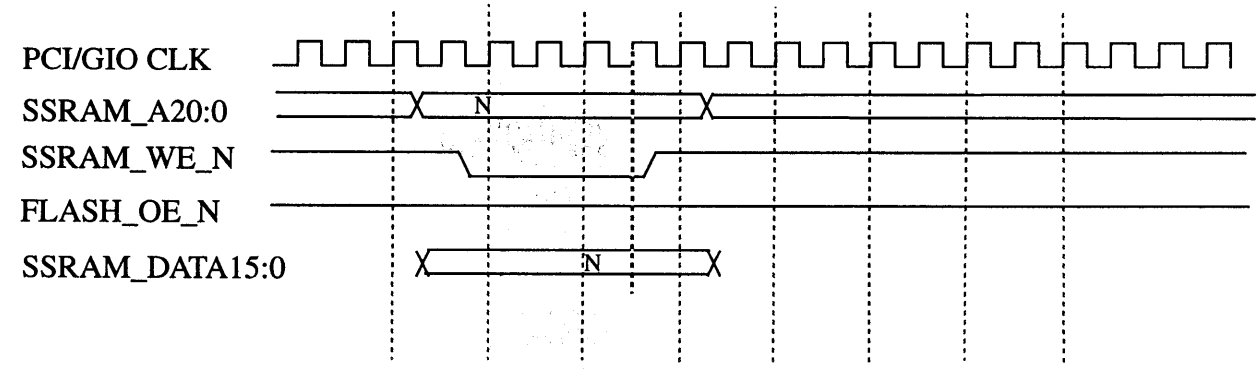


Figure 22 EEPROM Write Timing

10.2.4 Flash Prom Requirements

- AMD 29F400 - 120nS
- Intel 28F400 - 120nS

CHAPTER 11

Interrupts

11.1 Bridge Interrupt Introduction

Bridge can either accept interrupt requests from the GIO/PCI devices as an interrupt collector or it can initiate an interrupt when some predefined error condition occurs. Interrupts are therefore broken into two classes, interrupts from the interrupt pins and error interrupts. An interrupt is a crosstalk double word write packet using the address and widget ID defined in the interrupt destination register. The data in this write packet consists of an eight bit field, selecting the level of interrupt at the host and a single bit indicating set or clear of that interrupt.

The interrupt pins status is always reflected as the inverted values of the pin even when the interrupt for that pin is masked. Enabling the interrupt (in the interrupt enable register) allows for the assertion interrupt to be sent, enabling the clear packet (in the interrupt mode register) allows the negation interrupt to be sent. Both enables must be set to allow generation of the clear packets. For each interrupt pin, the interrupt level and an address overlay is taken for the interrupt (x) host register. This allows each interrupt a unique level and address with in the same target ID.

The interrupt pins can also influence the buffer flush management for a device. When a device finishes transferring data to or from the Bridge, there may be partially filled data in the write gathering buffer or unused prefetched data in the read buffer. The device can initiate an interrupt or do a memory write to address 0x3FFF_XXXX to flush the PCI_RW_Req

buffer and invalidate all the data in the PCI_R_Resp buffers which are assigned to the device. If the device uses an interrupt pin to flush/invalidate data buffers, the interrupt packet will be sent after the data. The interrupt pins must be assigned to each device using the interrupt device register. Multiple pins can be assigned to a single device.

Programmers Note: The device number is determined by which bus request/bus grant pair is used. This usually but not always corresponds to the configuration space used. Also the device number is encoded in the tnum of the crosstalk packet.

Error interrupts have different behaviors from the pin interrupts in the following ways: error interrupts can only generate set interrupt operations; always use the interrupt destination address without any overlays; use the same level for all error from the host error field register. Error interrupt are grouped based on the logging registers used. Only the first error is logged, any additional interrupt in the same group causes the multi-error bit to be set in the interrupt status register. This bit is shared with all error groups. Error interrupt are cleared by group, but can be enabled individually.

11.2 Interrupt Operations

The Bridge only supports level triggered interrupts shown in Figure 23. The interrupt status value follows the reverse of INT_N pins. The status bit goes to one when the corresponding INT_N pin goes low and it goes to zero when the INT_N pin goes high. No host clear is needed to clear the status bits and Bridge interrupt circuit ignores the host clear if it is ever issued.

Interrupt pins are clocked by the PCI/GIO Bus clock at the PCI/GIO interface side. The signal change needs to last for at least one clock cycle before the interrupt is reported to host. When more than one interrupt sources from PCI/GIO devices are OR'ed on the system board before it drives an interrupt pin, the board should provide mechanism for the host to read some register so that the host can know the real source of the interrupt.

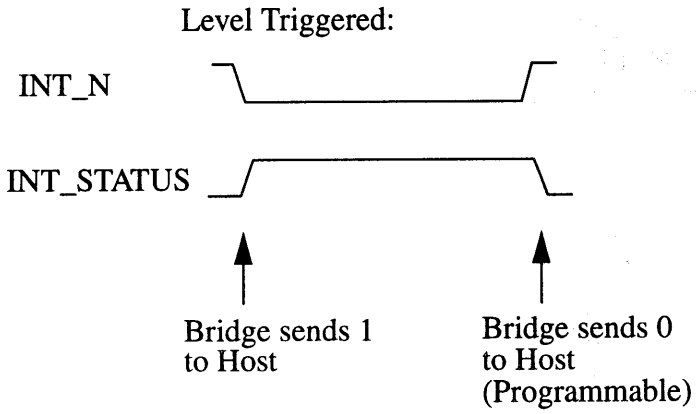


Figure 23 Level Triggered Interrupts

CHAPTER 12

Pin Locations and AC Parameters

12.1 Package Ball Assignments

Pin Name	BALL	Pin Name	BALL
VDD	D06	BUS_AD_63	AB05
PCI_C_BE_N_7	C05	BUS_AD_62	AA06
PCI_C_BE_N_6	B04	BUS_AD_61	AC03
PCI_C_BE_N_5	D07	BUS_AD_60	AB04
PCI_C_BE_N_4	B05	BUS_AD_59	AA05
PCI_C_BE_N_3	A04	VDD2 ****	AA04
PCI_C_BE_N_2	D08	BUS_AD_58	AB02
PCI_C_BE_N_1	C07	BUS_AD_57	AA03
PCI_C_BE_N_0	B06	BUS_AD_56	AB01
VDD2 ****	D09	VDD	Y04

Table 63 **Package Ball Assignments**

Pin Name	BALL	Pin Name	BALL
PCI_DEVSEL_N	C08	BUS_AD_55	Y03
PCI_ACK64_N	B07	BUS_AD_54	AA01
PCI_REQ64_N	A06	BUS_AD_53	Y02
PCI_PERR_N	C09	BUS_AD_52	W04
PCI_PAR64	B08	BUS_AD_51	W03
VDD	D10	BUS_AD_50	W02
PCI_PAR	A07	VDD ****	V04
PCI_IRDY_N	C10	BUS_AD_49	V03
PCI_FRAME_N	B09	BUS_AD_48	W01
PCI_TRDY_N	B10	BUS_AD_47	V02
PCI_STOP_N	A09	BUS_AD_46	U04
PCI_SERR_N	D11	BUS_AD_45	U03
VDD ****	C11	BUS_AD_44	V01
SSRAM_A_20	B11	VDD	T04
SSRAM_A_19	A10	BUS_AD_43	U02
SSRAM_A_18	C12	BUS_AD_42	T03
SSRAM_A_17	D12	BUS_AD_41	T02
SSRAM_A_16	B12	BUS_AD_40	R04
SSRAM_A_15	A12	BUS_AD_39	T01
SSRAM_A_14	B13	BUS_AD_38	R03
SSRAM_A_13	C13	VDD ****	P04
VDD2	D13	BUS_AD_37	R02
SSRAM_A_12	A13	BUS_AD_36	R01
SSRAM_A_11	B14	BUS_AD_35	P03

Table 63 Package Ball Assignments

Pin Name	BALL	Pin Name	BALL
SSRAM_A_10	A15	BUS_AD_34	P02
SSRAM_A_9	C14	BUS_AD_33	N04
SSRAM_A_8	D14	BUS_AD_32	N02
SSRAM_A_7	B15	VDD2	M03
SSRAM_A_6	A16	BUS_AD_31	M02
SSRAM_A_5	C15	BUS_AD_30	M04
VDD2	D15	BUS_AD_29	L02
SSRAM_A_4	B16	BUS_AD_28	L03
SSRAM_A_3	A18	BUS_AD_27	K01
SSRAM_A_2	C16	BUS_AD_26	K02
SSRAM_A_1	D16	VDD ****	L04
SSRAM_A_0	B17	BUS_AD_25	K03
FLASH_SELECT	A19	BUS_AD_24	J01
FLASH_CS_N_1	B18	BUS_AD_23	K04
FLASH_CS_N_0	C17	BUS_AD_22	J02
VDD2	D17	BUS_AD_21	J03
SSRAM_ADV_N	A20	BUS_AD_20	H02
SSRAM_ADSP_N	B19	VDD	J04
SSRAM_CLK	D18	BUS_AD_19	G01
FLASH_OE_N	A21	BUS_AD_18	H03
SSRAM_OE_N	C19	BUS_AD_17	H04
SSRAM_WE_N	A22	BUS_AD_16	G02
VDD2	D19	BUS_AD_15	F01
BUS_CLKIN	B21	BUS_AD_14	G03

Table 63 Package Ball Assignments

Pin Name	BALL	Pin Name	BALL
BUS_PLL_VDD	C20	VDD ****	G04
BUS_PLL_AGND	C21	BUS_AD_13	F02
BUS_PLL_LP2	D20	BUS_AD_12	F03
BUS_PLL_VSS	D21	BUS_AD_11	F04
PLL_RST_N	C22	BUS_AD_10	E02
SSRAM_D_0	B23	BUS_AD_9	D02
SSRAM_D_1	C23	BUS_AD_8	E03
SSRAM_D_2	D22	VDD	E04
SSRAM_D_3	C24	BUS_AD_7	C01
VDD2	E21	BUS_AD_6	D03
SSRAM_D_4	D23	BUS_AD_5	C02
SSRAM_D_5	E22	VDD ****	D04
SSRAM_D_6	D24	BUS_AD_4	C03
SSRAM_D_7	E23	BUS_AD_3	D05
SSRAM_D_8	F21	BUS_AD_2	B02
SSRAM_D_9	F22	BUS_AD_1	C04
SSRAM_D_10	F23	BUS_AD_0	B03
SSRAM_D_11	F24	VDD	AA15
VDD2	G21	GIO_READ	AD15
SSRAM_D_12	G22	GIO_MEMDLY	AC14
SSRAM_D_13	G23	GIO_GRXDLY	AB14
SSRAM_D_14	G24	GIO_AS_N	AA14
SSRAM_D_15	H23	BUS_INT_N_7	AD13
SSRAM_DP_0	J24	BUS_INT_N_6	AC13

Table 63 Package Ball Assignments

Pin Name	BALL	Pin Name	BALL
SSRAM_DP_1	H22	BUS_INT_N_5	AB13
VDD	H21	VDD	AA13
C_CD_9	J23	BUS_INT_N_4	AD12
C_CD_7	K24	BUS_INT_N_3	AD10
VDD2	J21	BUS_INT_N_2	AC12
C_CD_5	K23	BUS_INT_N_1	AA12
C_CD_3	L23	BUS_INT_N_0	AC11
VDD2	K21	BUS_REQ_N_7	AD09
C_CD_1	M24	VDD ****	AB12
C_DRDY_N	N24	BUS_REQ_N_6	AB11
VDD2	L21	BUS_REQ_N_5	AA11
C_CLK_N	M23	BUS_REQ_N_4	AC10
C_CLK	N23	BUS_REQ_N_3	AD07
VDD2	M21	BUS_REQ_N_2	AC09
C_CD_0	P23	BUS_REQ_N_1	AC08
C_CD_2	R24	BUS_REQ_N_0	AB10
VDD2	N21	VDD	AA10
C_CD_4	R23	BUS_GNT_N_7	AD06
C_CD_6	T24	BUS_GNT_N_6	AC07
VDD2	R22	BUS_GNT_N_5	AB09
C_CD_8	T23	BUS_GNT_N_4	AC06
VDD	R21	BUS_GNT_N_3	AB08
X_VREF	P21	VDD	AA09
D_CD_9	U23	BUS_GNT_N_2	AD04

Table 63 Package Ball Assignments

Pin Name	BALL	Pin Name	BALL
D_CD_7	V24	BUS_GNT_N_1	AC05
D_CD_5	V23	BUS_GNT_N_0	AA08
D_CD_3	U22	BUS_RSTN_3	AB07
D_CD_1	W24	BUS_RSTN_2	AD03
VDD2	U21	BUS_RSTN_1	AB06
LLP_CLK	W23	VDD	AA07
LLP_CLK_N	Y23	BUS_RSTN_0	AC04
D_CLK	W22	CLK_50	AC19
D_CLK_N	Y22	CLK_25	AD19
D_DRDY_N	AA24	TRST_N	AA18
D_CD_0	AA23	JTDI	AB18
D_CD_2	AB24	JTCK	AC18
D_CD_4	AA22	ENTEI	AD18
D_CD_6	AB23	VDD	AA17
D_CD_8	AB22	NIC_IO	AB17
VDD	AA21	CLK_20	AC17
SIM_RST_N	AC23	PCI_GION	AC16
QUICK_BOOT_N	AB21	CLK_33	AB16
LLP_RST_N	AA20	TESTOUT_3	AA16
PNAND	AC22	TESTOUT_2	AD16
JTMS	AC21	TESTOUT_1	AC15
SCAN_OUT2	AB20	TESTOUT_0	AB15
TP2	AD22	TP1	AB19
JTDO	AC20	TP0	AD21

Table 63 Package Ball Assignments

Pin Name	BALL	Pin Name	BALL
VDD	AA19	VSS	P01
VSS	A02	VSS	Y24
VSS	B24	VSS	AD17
VSS	L24	VSS	A23
VSS	U24	VSS	H24
VSS	AD01	VSS	P22
VSS	A03	VSS	AA02
VSS	C06	VSS	AD20
VSS	M01	VSS	A24
VSS	V21	VSS	J22
VSS	AD02	VSS	P24
VSS	A05	VSS	AB03
VSS	C18	VSS	AD23
VSS	M22	VSS	B01
VSS	V22	VSS	K22
VSS	AD05	VSS	T21
VSS	A08	VSS	AC01
VSS	D01	VSS	AD24
VSS	N01	VSS	B20
VSS	W21	VSS	L01
VSS	AD08	VSS	T22
VSS	A11	VSS	AC02
VSS	E01	VSS	B22
VSS	N03	VSS	L22

Table 63 Package Ball Assignments

Pin Name	BALL	Pin Name	BALL
VSS	Y01	VSS	U01
VSS	AD11	VSS	AC24
VSS	A14	VSS	AD14
VSS	E24	VSS	A17
VSS	N22	VSS	H01
VSS	Y21		

Table 63 Package Ball Assignments

Note: The pins marked with a * should be connected to 3.3V for rev 1.0 parts, and 5.0V for rev 2.0 parts.***

12.2 AC Parameters

Signal Name	Reference Clock	Clock to Output Min/Max nS	Setup nS	Hold nS	Loading	Buffer Type
SSRAM_A	SSRAM_CLK	2.70 / 7.30			70 pF	bt6rpf
SSRAM_D SSRAM_DP	SSRAM_CLK	1.20 / 4.80	2.75	.50	70 pF	bd6crpf
SSRAM_OE_N SSRAM_WE_N SSRAM_ADSP_N SSRAM_ADV_N	SSRAM_CLK	1.20 / 4.80			70 pF	bt6rpf
FLASH_CS_N FLASH_OE_N	SSRAM_CLK	2.70 / 7.30			70 pF	bt6rpf

Table 64 AC Parameters

Signal Name	Reference Clock	Clock to Output Min/Max nS	Setup nS	Hold nS	Loading	Buffer Type
PCI_C_BE_N PCI_PAR PCI_PAR64 PCI_FRAME_N PCI_IRDY_N PCI_TRDY_N PCI_STOP_N PCI_DEVSEL_N PCI_PERR_N PCI_SERR_N PCI_ACK64_N PCI_REQ64_N	BUS_CLKIN	2.0 / 11.0	2.0	0	50pF	bdepcif
BUS_AD	BUS_CLKIN	2.0 / 11.0	2.0	0	50pF	bdepcif
BUS_REQ BUS_INT	BUS_CLKIN		2.0	0	50pF	bdepcif
BUS_GNT BUS_RSTN	BUS_CLKIN	2.0 / 11.0			50pF	bdepcif
GIO_AS_N GIO_READ	BUS_CLKIN	2.0 / 12.0			50pF	bd6crpf
GIO_MEMDLY	BUS_CLKIN	2.0 / 12.0			50pF	bt6rpf
GIO_GRXDLY	BUS_CLKIN		2.0	0	50pF	ibuf
PCI_GION FLASH_SELECT	BUS_CLKIN		2.0	0	50pF	ibuf
NIC_IO	BUS_CLKIN		2.0	0	50 pF	bd4codf
CLK_33 CLK_40 CLK_50	No Phase relationship, derived from llp clock				50pF	bt6rpf

Table 64 AC Parameters

APPENDIX A

Byte Swapping

When the width of a bus becomes more than one byte wide, it is possible to transfer more than one byte in a single cycle. When performing this transfer a numbering scheme is required to indicate the address of each byte of that transfer. Both possible numbering schemes exists as standard addressing methods known as ***Big Endian*** and ***Little Endian*** addressing. Big Endian addressing numbers the lowest addressed bytes on the most significant bits of the bus during a transfer. Little Endian numbers the lowest addressed bytes on the least significant bits of the bus during a transfer.

If devices of different addressing schemes are required to communicate, the position of bytes during a transfer will need to be swapped. The *Bridge* ASIC supports the PCI bus which uses Little Endian address mode and the GIO bus which can use either addressing mode. The system can be of either addressing mode as well and therefore swapping may be desirable.

The *Bridge* ASIC provides byte swapping between the *Crosstalk* Interconnect and the PCI/GIO bus. Byte swapping can be enabled per device for *Crosstalk* to PCI/GIO accesses with the device registers. Byte swapping can be enabled for PCI/GIO to *Crosstalk* in both mapping modes. The direct mapping mode supports byte swapping on a per device basis using the device registers.

As stated above for bytes, if the width of the bus is multiples of the data size the same addressing problem occurs. Half-words on a word wide

bus, or words on a double word bus. The *Bridge* ASIC **does not** support half-word, or word swapping. The following tables depict byte, half-word, word, and double word transfers with and without **byte** swapping enabled. Correct alignments for word and half-word transfer must be made by the CPU. The following Tables provide an exhaustive list of the possibilities.

A.1 Big Endian System

A.1.1 Swapping

T5 SysAD Bus								T5	PCI	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	Addr	Addr	31:24	23:16	15:8	7:0
A								0x0	0x0				A
	A							0x1	0x0			A	
		A						0x2	0x0		A		
			A					0x3	0x0	A			
				A				0x4	0x4				A
					A			0x5	0x4			A	
						A		0x6	0x4		A		
							A	0x7	0x4	A			

Table 65 Byte Data Big Endian SysAD to PCI-32 with Byte Swap

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
A	B							0x0	0x0			B	A

Table 66 Half Word Data Big Endian SysAD to PCI-32 with Byte Swap

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
	A	B						0x1	0x0		B	A	
		A	B					0x2	0x0	B	A		
			A	B				0x3	0x0	A			
									0x4				B
				A	B			0x4	0x4			B	A
					A	B		0x5	0x4		B	A	
						A	B	0x6	0x4	B	A		

Table 66 **Half Word Data Big Endian SysAD to PCI-32 with Byte Swap**

**Note: For transfers that cause multiple cycles, the lower address transfer is done first.

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
A	B	C	D					0x0	0x0	D	C	B	A
	A	B	C	D				0x1	0x0	C	B	A	
									0x4				D
		A	B	C	D			0x2	0x0	B	A		
									0x4			D	C
			A	B	C	D		0x3	0x0	A			
									0x4		D	C	B
				A	B	C	D	0x4	0x4	D	C	B	A

Table 67 **Word Data Big Endian SysAD to PCI-32 with Byte Swap**

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
A	B	C	D	E	F	G	H	0x0	0x0	D	C	B	A
								0	0x4	H	G	F	E

Table 68 **Double Word Data Big Endian SysAD to PCI-32 with Byte Swap**

**Note: For transfers that cause multiple cycles, the lower address transfer is done first.

T5 SysAD Bus								T5	PCI	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	Addr	Addr	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
A								0x0	0x0								A
	A							0x1	0x0							A	
		A						0x2	0x0						A		
			A					0x3	0x0					A			
				A				0x4	0x0				A				
					A			0x5	0x0			A					
						A		0x6	0x0		A						
							A	0x7	0x0	A							

Table 69 **Byte Data Big Endian SysAD to PCI-64 with Byte Swap**

T5 SysAD Bus								T5 Addr	PCI Addr	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
A	B							0x0	0x0							B	A
	A	B						0x1	0x0						B	A	
		A	B					0x2	0x0					B	A		
			A	B				0x3	0x0				B	A			
				A	B			0x4	0x0			B	A				
					A	B		0x5	0x0		B	A					
						A	B	0x6	0x0	B	A						

Table 70 **Half Word Data Big Endian SysAD to PCI-64 with Byte Swap**

T5 SysAD Bus								T5 Addr	PCI Addr	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
A	B	C	D					0x0	0x0					D	C	B	A
	A	B	C	D				0x1	0x0				D	C	B	A	
		A	B	C	D			0x2	0x0			D	C	B	A		
			A	B	C	D		0x3	0x0		D	C	B	A			
				A	B	C	D	0x4	0x0	D	C	B	A				

Table 71 **Word Data Big Endian SysAD to PCI-64 with Byte Swap**

T5 SysAD Bus								T5 Addr	PCI Addr	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
A	B	C	D	E	F	G	H	0x0	0x0	H	G	F	E	D	C	B	A

Table 72 **Double Word Data Big Endian SysAD to PCI-64 with Byte Swap**

A.1.2 No Swapping

T5 SysAD Bus								T5	PCI	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	Addr	Addr	31:24	23:16	15:8	7:0
A								0x0	0x0	A			
	A							0x1	0x0		A		
		A						0x2	0x0			A	
			A					0x3	0x0				A
				A				0x4	0x4	A			
					A			0x5	0x4		A		
						A		0x6	0x4			A	
							A	0x7	0x4				A

Table 73 **Byte Data Big Endian SysAD to PCI-32 No Swap**

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
A	B							0x0	0x0	A	B		
	A	B						0x1	0x0		A	B	
		A	B					0x2	0x0			A	B
			A	B				0x3	0x0				A
									0x4	B			
				A	B			0x4	0x4	A	B		
					A	B		0x5	0x4		A	B	
						A	B	0x6	0x4			A	B

Table 74 **Half Word Data Big Endian SysAD to PCI-32 No Swap**

**Note: For transfers that cause multiple cycles, the lower address transfer is done first.

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
A	B	C	D					0x0	0x0	A	B	C	D
	A	B	C	D				0x1	0x0		A	B	C
									0x4	D			
		A	B	C	D			0x2	0x0			A	B
									0x4	C	D		
			A	B	C	D		0x3	0x0				A
									0x4	B	C	D	
				A	B	C	D	0x4	0x4	A	B	C	D

Table 75 **Word Data Big Endian SysAD to PCI-32 No Swap**

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
A	B	C	D	E	F	G	H	0x0	0x0	A	B	C	D
								0	0x4	E	F	G	H

Table 76 **Double Word Data Big Endian SysAD to PCI-32 No Swap**

**Note: For transfers that cause multiple cycles, the lower address transfer is done first.

T5 SysAD Bus								T5	PCI	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	Addr	Addr	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
A								0x0	0x0	A							
	A							0x1	0x0		A						
		A						0x2	0x0			A					
			A					0x3	0x0				A				
				A				0x4	0x0					A			
					A			0x5	0x0						A		
						A		0x6	0x0							A	
							A	0x7	0x0								A

Table 77 **Byte Data Big Endian SysAD to PCI-64 No Swap**

T5 SysAD Bus								T5 Addr	PCI Addr	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
A	B							0x0	0x0	A	B						
	A	B						0x1	0x0		A	B					
		A	B					0x2	0x0			A	B				
			A	B				0x3	0x0				A	B			
				A	B			0x4	0x0					A	B		
					A	B		0x5	0x0						A	B	
						A	B	0x6	0x0							A	B

Table 78 Half Word Data Big Endian SysAD to PCI-64 No Swap

T5 SysAD Bus								T5 Addr	PCI Addr	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
A	B	C	D					0x0	0x0	A	B	C	D				
	A	B	C	D				0x1	0x0		A	B	C	D			
		A	B	C	D			0x2	0x0			A	B	C	D		
			A	B	C	D		0x3	0x0				A	B	C	D	
				A	B	C	D	0x4	0x0					A	B	C	D

Table 79 Word Data Big Endian SysAD to PCI-64 No Swap

T5 SysAD Bus								T5 Addr	PCI Addr	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
A	B	C	D	E	F	G	H	0x0	0x0	A	B	C	D	E	F	G	H

Table 80 Double Word Data Big Endian SysAD to PCI-64 No Swap

A.2 Little Endian System

A.2.1 Byte Swapping

T5 SysAD Bus								T5	PCI	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	Addr	Addr	31:24	23:16	15:8	7:0
							A	0x0	0x0	A			
						A		0x1	0x0		A		
					A			0x2	0x0			A	
				A				0x3	0x0				A
			A					0x4	0x4	A			
		A						0x5	0x4		A		
	A							0x6	0x4			A	
A								0x7	0x4				A

Table 81 Byte Data Little Endian SysAD to PCI-32 with Byte Swap

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
						B	A	0x0	0x0	A	B		
					B	A		0x1	0x0		A	B	
				B	A			0x2	0x0			A	B
			B	A				0x3	0x0				A
									0x4	B			
		B	A					0x4	0x4	A	B		
	B	A						0x5	0x4		A	B	
B	A							0x6	0x4			A	B

Table 82 **Half Word Data Little Endian SysAD to PCI-32 with Byte Swap**

**Note: For transfers that cause multiple cycles, the lower address transfer is done first.

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
				D	C	B	A	0x0	0x0	A	B	C	D
			D	C	B	A		0x1	0x0		A	B	C
									0x4	D			
		D	C	B	A			0x2	0x0			A	B
									0x4	C	D		
	D	C	B	A				0x3	0x0				A
									0x4	B	C	D	
D	C	B	A					0x4	0x4	A	B	C	D

Table 83 **Word Data Little Endian SysAD to PCI-32 with Byte Swap**

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
H	G	F	E	D	C	B	A	0x0	0x0	A	B	C	D
								0	0x4	E	F	G	H

Table 84 **Double Word Data Little Endian SysAD to PCI-32 with Byte Swap**

**Note: For transfers that cause multiple cycles, the lower address transfer is done first.

T5 SysAD Bus								T5	PCI	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	Addr	Addr	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
							A	0x0	0x0	A							
						A		0x1	0x0		A						
					A			0x2	0x0			A					
				A				0x3	0x0				A				
			A					0x4	0x0					A			
		A						0x5	0x0						A		
	A							0x6	0x0							A	
A								0x7	0x0								A

Table 85 **Byte Data Little Endian SysAD to PCI-64 with Byte Swap**

T5 SysAD Bus								T5 Addr	PCI Addr	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
						B	A	0x0	0x0	A	B						
					B	A		0x1	0x0		A	B					
				B	A			0x2	0x0			A	B				
			B	A				0x3	0x0				A	B			
		B	A					0x4	0x0					A	B		
	B	A						0x5	0x0						A	B	
B	A							0x6	0x0							A	B

Table 86 **Half Word Data Little Endian SysAD to PCI-64 with Byte Swap**

T5 SysAD Bus								T5 Addr	PCI Addr	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
				D	C	B	A	0x0	0x0	A	B	C	D				
			D	C	B	A		0x1	0x0		A	B	C	D			
		D	C	B	A			0x2	0x0			A	B	C	D		
	D	C	B	A				0x3	0x0				A	B	C	D	
D	C	B	A					0x4	0x0					A	B	C	D

Table 87 **Word Data Little Endian SysAD to PCI-64 with Byte Swap**

T5 SysAD Bus								T5 Addr	PCI Addr	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
H	G	F	E	D	C	B	A	0x0	0x0	A	B	C	D	E	F	G	H

Table 88 **Double Word Data Little Endian SysAD to PCI-64 with Byte Swap**

A.2.2 No Swapping

T5 SysAD Bus								T5	PCI	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	Addr	Addr	31:24	23:16	15:8	7:0
							A	0x0	0x0				A
						A		0x1	0x0			A	
					A			0x2	0x0		A		
				A				0x3	0x0	A			
			A					0x4	0x4				A
		A						0x5	0x4			A	
	A							0x6	0x4		A		
A								0x7	0x4	A			

Table 89 **Byte Data Little Endian SysAD to PCI-32 No Swap**

**Note: For transfers that cause multiple cycles, the lower address transfer is done first.

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
						B	A	0x0	0x0			B	A

Table 90 **Half Word Data Little Endian SysAD to PCI-32 No Swap**

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
					B	A		0x1	0x0		B	A	
				B	A			0x2	0x0	B	A		
			B	A				0x3	0x0	A			
									0x4				B
		B	A					0x4	0x4			B	A
	B	A						0x5	0x4		B	A	
B	A							0x6	0x4	B	A		

Table 90 Half Word Data Little Endian SysAD to PCI-32 No Swap

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
				D	C	B	A	0x0	0x0	D	C	B	A
			D	C	B	A		0x1	0x0	C	B	A	
									0x4				D
		D	C	B	A			0x2	0x0	B	A		
									0x4			D	C
	D	C	B	A				0x3	0x0	A			
									0x4		D	C	B
D	C	B	A					0x4	0x4	D	C	B	A

Table 91 Word Data Little Endian SysAD to PCI-32 No Swap

**Note: For transfers that cause multiple cycles, the lower address transfer is done first.

T5 SysAD Bus								T5 Addr	PCI Addr	32-bit PCI Bus			
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			31:24	23:16	15:8	7:0
H	G	F	E	D	C	B	A	0x0	0x0	D	C	B	A
								0	0x4	H	G	F	E

Table 92 **Double Word Data Little Endian SysAD to PCI-32 No Swap**

T5 SysAD Bus								T5	PCI	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	Addr	Addr	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
							A	0x0	0x0								A
						A		0x1	0x0							A	
					A			0x2	0x0						A		
				A				0x3	0x0					A			
			A					0x4	0x0				A				
		A						0x5	0x0			A					
	A							0x6	0x0		A						
A								0x7	0x0	A							

Table 93 **Byte Data Little Endian SysAD to PCI-64 No Swap**

T5 SysAD Bus								T5 Addr	PCI Addr	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
						B	A	0x0	0x0							B	A
					B	A		0x1	0x0						B	A	
				B	A			0x2	0x0					B	A		
			B	A				0x3	0x0				B	A			
		B	A					0x4	0x0			B	A				
	B	A						0x5	0x0		B	A					
B	A							0x6	0x0	B	A						

Table 94 Half Word Data Little Endian SysAD to PCI-64 No Swap

T5 SysAD Bus								T5 Addr	PCI Addr	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
				D	C	B	A	0x0	0x0					D	C	B	A
			D	C	B	A		0x1	0x0				D	C	B	A	
		D	C	B	A			0x2	0x0			D	C	B	A		
	D	C	B	A				0x3	0x0		D	C	B	A			
D	C	B	A					0x4	0x0	D	C	B	A				

Table 95 Word Data Little Endian SysAD to PCI-64 No Swap

T5 SysAD Bus								T5 Addr	PCI Addr	64-bit PCI Bus							
63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0			63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
H	G	F	E	D	C	B	A	0x0	0x0	H	G	F	E	D	C	B	A

Table 96 Double Word Data Little Endian SysAD to PCI-64 No Swap

APPENDIX B

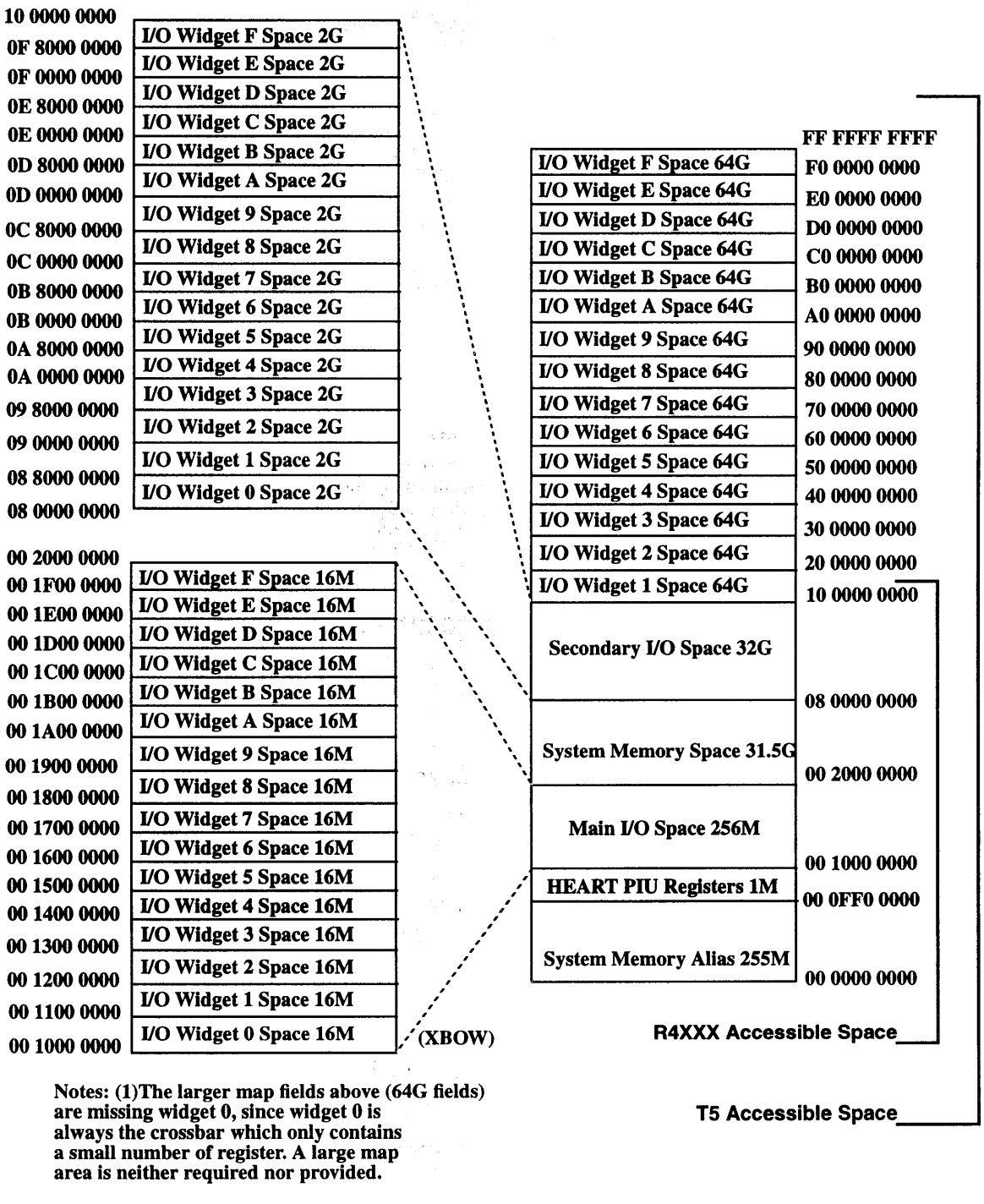
Software Notes and Restrictions

B.1 Addressing in a Godzilla System

Godzilla is the code name for the chip set based on Heart, Bridge, and Crossbow. ISD will produce it's next generation product using these chips. Some helpful tables on address are included here to aid the programmer when accessing PCI through a Bridge.

Although widgets have a 256 Tera Byte addressing space available, not all widgets use this space, if fact, the Bridge only uses 8 Giga Bytes. Since the Heart supports different processors with unique address limitations the Heart contains a small, medium and large window to each widget. The small window is 16 Mega Bytes, the medium is 2 Giga Bytes and the large window is 64 Giga Bytes. Figure 24 contains the processor address maps for the different ranges. Table 97contains the starting widget space address locations in the *Godzilla* system space for widget id 0x2 through 0xf for each of the three windows. For access to each window and processor type please refer to the Heart Specification.

The following tables assume IRIX as Big Endian.



Notes: (1)The larger map fields above (64G fields) are missing widget 0, since widget 0 is always the crossbar which only contains a small number of register. A large map area is neither required nor provided.

Figure 24 Heart & R10000 Address Map to Bridge

Widget ID Number	Small Window	Medium Window	Large Window
0x2	0x0000_1200_0000	0x0009_0000_0000	0x0020_0000_0000
0x3	0x0000_1300_0000	0x0009_8000_0000	0x0030_0000_0000
0x4	0x0000_1400_0000	0x000A_0000_0000	0x0040_0000_0000
0x5	0x0000_1500_0000	0x000A_8000_0000	0x0050_0000_0000
0x6	0x0000_1600_0000	0x000B_0000_0000	0x0060_0000_0000
0x7	0x0000_1700_0000	0x000B_8000_0000	0x0070_0000_0000
0x8	0x0000_1800_0000	0x000C_0000_0000	0x0080_0000_0000
0x9	0x0000_1900_0000	0x000C_8000_0000	0x0090_0000_0000
0xA	0x0000_1A00_0000	0x000D_0000_0000	0x00A0_0000_0000
0xB	0x0000_1B00_0000	0x000D_8000_0000	0x00B0_0000_0000
0xC	0x0000_1C00_0000	0x000E_0000_0000	0x00C0_0000_0000
0xD	0x0000_1D00_0000	0x000E_8000_0000	0x00D0_0000_0000
0xE	0x0000_1E00_0000	0x000F_0000_0000	0x00E0_0000_0000
0xF	0x0000_1F00_0000	0x000F_8000_0000	0x00F0_0000_0000

Table 97 Widget Base Address

Description	Processor Offset from Base above	Cycle type and size
Widget Configuration		
Bridge Identification Register	0x0000_0004	read-only Word
Bridge Status Register	0x0000_000C	read-only Word
Bridge Error Upper Address Register	0x0000_0014	read-only Word
Bridge Error Lower Address Register	0x0000_001C	read-only Word

Table 98 Address Map From Base Address

Description	Processor Offset from Base above	Cycle type and size
Bridge Control Register	0x0000_0024	read/write Word
Bridge Request Time-out Value Register	0x0000_002C	read/write Word
Bridge Interrupt Destination Upper Address Register	0x0000_0034	read/write Word
Bridge Interrupt Destination Lower Address Register	0x0000_003C	read/write Word
Bridge Error Command Word Register	0x0000_0044	read-only Word
Bridge LLP Configuration Register	0x0000_004C	read/write Word
Bridge Target Flush Register	0x0000_0054	read-only Word
Aux Error Command Word Register	0x0000_005C	read-only Word
Response Buffer Upper Address Register	0x0000_0064	read-only Word
Response Buffer Lower Address Register	0x0000_006C	read-only Word
Test Pin Control Register	0x0000_0074	read/write Word
PMU & Map		
Direct Map Register	0x0000_0084	read/write Word
SSRAM		
SSRAM Parity Error Register	0x0000_0094	read-only Word
Arbitration		
Arbitration Priority Register	0x0000_00A4	read/write Word
Number In A Can		
NIC Register	0x0000_00B4	read/write Word
PCI/GIO		
PCI/GIO Time-out Register	0x0000_00C4	read/write Word
PCI Type 1 Configuration Register	0x0000_00CC	read/write Word

Table 98 Address Map From Base Address

Description	Processor Offset from Base above	Cycle type and size
PCI/GIO Error Upper Address Register	0x0000_00D4	read only Word
PCI/GIO Error Lower Address Register	0x0000_00DC	read only Word
Interrupt		
Bridge Interrupt Status Register	0x0000_0104	read only Word
Bridge Interrupt Enable Register	0x0000_010C	read/write Word
Reset Interrupt Status Register	0x0000_0114	write only Word
Interrupt Mode Register	0x0000_011C	read/write Word
Interrupt Device Register	0x0000_0124	read/write Word
Host Error Field Register	0x0000_012C	read/write Word
Interrupt 0 Host Address Register	0x0000_0134	read/write Word
Interrupt 1Host Address Register	0x0000_013C	read/write Word
Interrupt 2 Host Address Register	0x0000_0144	read/write Word
Interrupt 3 Host Address Register	0x0000_014C	read/write Word
Interrupt 4 Host Address Register	0x0000_0154	read/write Word
Interrupt 5 Host Address Register	0x0000_015C	read/write Word
Interrupt 6 Host Address Register	0x0000_0164	read/write Word
Interrupt 7 Host Address Register	0x0000_016C	read/write Word
Device		
Device 0 Register	0x0000_0204	read/write Word
Device 1 Register	0x0000_020C	read/write Word
Device 2 Register	0x0000_0214	read/write Word
Device 3 Register	0x0000_021C	read/write Word
Device 4 Register	0x0000_0224	read/write Word

Table 98 Address Map From Base Address

Description	Processor Offset from Base above	Cycle type and size
Device 5 Register	0x0000_022C	read/write Word
Device 6 Register	0x0000_0234	read/write Word
Device 7 Register	0x0000_023C	read/write Word
Device 0 Write Request Buffer Register	0x0000_0244	read only Word
Device 1 Write Request Buffer Register	0x0000_024C	read only Word
Device 2 Write Request Buffer Register	0x0000_0254	read only Word
Device 3 Write Request Buffer Register	0x0000_025C	read only Word
Device 4 Write Request Buffer Register	0x0000_0264	read only Word
Device 5 Write Request Buffer Register	0x0000_026C	read only Word
Device 6 Write Request Buffer Register	0x0000_0274	read only Word
Device 7 Write Request Buffer Register	0x0000_027C	read only Word
Even Device Response Buffer Register	0x0000_0284	read/write Word
Odd Device Response Buffer Register	0x0000_028C	read/write Word
Read Response Buffer Status Register	0x0000_0294	read only Word
Read Response Buffer Clear Register	0x0000_029C	write only Word
Internal Rams		
Internal ATE Ram	0x0001_0000	write Dbl Word
Internal ATE Ram data word 63:32	0x0001_0004	read Word
Internal ATE Ram data word 31:0	0x0001_1004	read Word
PCI Configuration Spaces		
PCI Device 0 Configuration Space	0x0002_0000	read/write Word
PCI Device 1 Configuration Space	0x0002_1000	read/write Word
PCI Device 2 Configuration Space	0x0002_2000	read/write Word

Table 98 Address Map From Base Address

Description	Processor Offset from Base above	Cycle type and size
PCI Device 3 Configuration Space	0x0002_3000	read/write Word
PCI Device 4 Configuration Space	0x0002_4000	read/write Word
PCI Device 5 Configuration Space	0x0002_5000	read/write Word
PCI Device 6 Configuration Space	0x0002_6000	read/write Word
PCI Device 7 Configuration Space	0x0002_7000	read/write Word
PCI Type 1 Configuration Space	0x0002_8000	read/write Word
PCI unique cycles		
PCI Interrupt Acknowledge Cycle	0x0003_0000	read Word
External SSRAM		
External SSRAM	0x0008_0000	read/write Dbl Word
PCI/GIO Device 0 Window	0x0020_0000	read/write all sizes
PCI/GIO Device 1 Window	0x0040_0000	read/write all sizes
PCI/GIO Device 2 Window	0x0060_0000	read/write all sizes
PCI/GIO Device 3 Window	0x0070_0000	read/write all sizes
PCI/GIO Device 4 Window	0x0080_0000	read/write all sizes
PCI/GIO Device 5 Window	0x0090_0000	read/write all sizes
PCI/GIO Device 6 Window	0x00A0_0000	read/write all sizes
PCI/GIO Device 7 Window	0x00B0_0000	read/write all sizes
Flash /EEPROM		
Flash Prom 0	0x00C0_0000	read all sizes write Half Word
Flash Prom 1	0x00E0_0000	read all sizes write Half Word

Table 98 Address Map From Base Address

12.3 Current Anomalies in Rev1 Si

- Symptom: A precise operation through mapped space generates a cache line (non-precise) read on the crosstalk bus. Workaround: Do not operate on addresses with side-effects in dma-mode.
- Symptom: When mapping through external page map ram, parity errors which occur during this operation, the external ram address is not correctly logged in the parity error register. Workaround: re-read map from crosstalk side (pio mode) and look for error.
- Symptom: When frame_n is high and master fsm sees stopn=0, irdy_n is driven high by bridge for 2 cycles before it is tristated, It should only be driven high for 1 cycle before it is tristated. Workaround: none, we don't expect the 3rd bullet to be a real problem except the tristate dead cycle could be missing on irdy_n.
- Symptom: After an address parity error happens, Bridge slave fsm issues stop_n on the next request. The cause is stop_n was asserted but never negated from the addr parity error. Workaround: none at this time.
- Symptom: Read response time-out occurs causing multiple crosstalk read requests to be launched because requesting pci device has bus when error occurs. Workaround: ground one of the unused bus requests and program the arbitration register to add delay between arbitration of at least 960nS.
- Symptom: Prefetcher does not store the mapped address when using page mapped mode with either internal or external maps. Workaround: None, do not use prefetching in mapped space.
- Symptom: Clear interrupt writes from int pins are lost during heavy write traffic from pci bus. Workaround: Lower the traffic load by changing the arbitration register to add inter-device delays.
- Symptom: Assertion of multi-error when the following errors occur: unexpected response Workaround: Clear multi error when exception occurs.
- Symptom: Can not generate request fifo overflow error condition Workaround: None
- Symptom: Partial cache line writes (from a pci master) that require multiple qcl xtalk packets from a single write buffer may be overwritten by subsequent writes. This is caused when the write buffer is freed after the first qcl packet is sent rather than the last. Workaround: Only send words or full cache lines, or add large delays between master tenures.

- Symptom: Byte swapping in pmu space can break if other bridge ops occur intermixed with the pmu space traffic. Workaround: Set dirmap and pmu swapping to the same value or don't use pmu space.
- Symptom: Response packets arriving out-of-order can cause corruption of dataout to PCI device. This also has the side effect of retransmitting read requests. This failure is more pronounced when prefetcher is turned on. Workaround: Avoid sending out-of-order response packets, if possible. Do not use prefetcher. Program only one channel per device.
- Symptom: A bad micro-packet during a read response with the tail bit set (when it should be clear) will cause the response buffer to be marked bad and generate a bad xtalk response error interrupt if enabled. In GIO mode, this error will hang the bus, in PCI mode this error forces a re-read of the pci address after the buffer is flushed. Workaround: Keep retries on xtalk to a minimum.

12.4 Current Anomalies in Rev 2 Si

- Symptom: When mapping through external page map ram, parity errors which occur during this operation, the external ram address is not correctly logged in the parity error register. Workaround: reread map from crosstalk side (pio mode) and look for error.
- Symptom: Arriving requests which do not contain the correct number of micro-packets as defined in the command word cause the request dispatcher to get out of sync. The request dispatcher detects the error and issues an interrupt but a read of the interrupt status register might not be possible due to the error. Workaround: reset the bridge after a error interrupt and read time-out of the interrupt status register. Note: The crossbow will not allow incorrectly sized packets to be sent to the bridge therefore this is a very rare error condition.
- Symptom: The bridge does not assert REQ64* during reset to indicate that the bridge is capable of 64-bit data transfers. Workaround: Additional hardware can be added external to the bridge to drive this signal during reset.
- Symptom: If during a prefetch op where a device has only a single response buffer allocated, and the entire write buffer queue backs up and stops. Then if the read stream is changed, the prior read is not flush and the new address is loaded on the old data. This will

cases the delivery of the old data with the new address. Workaround: Disable write gather.

- Symptom: In GIO mode the internal control expects a 64-bit address rather than a 32-bit one. Workaround: GIO requires fib 3 on rev 2 bridges.
- Symptom: Enabling write gather for any device with a dual address cycle or a external mapped cycle will cause other devices to enable write gathering when using dual address or mapped cycles. The bridge does not check the state of the write gather enable bit in this case for those devices. Workaround: Do not use write gather with dac or external page mapped cycles.
- Symptom: A precise operation through external mapped space generates a cache line (non-precise) read on the crosstalk bus. Workaround: Do not operate on addresses with side-effects in dma-mode.
- Symptom: A bad micro-packet during a read response with the tail bit set (when it should be clear) will cause the response buffer to be marked bad and generate a bad xtalk response error interrupt if enabled. In GIO mode, this error will hang the bus, in PCI mode this error forces a re-read of the pci address after the buffer is flushed. Workaround: Keep retries on xtalk to a minimum.
- Symptom: Prefetcher starts operating incorrectly. This happens due to corruption of the address storage ram in the prefetcher when a non-real time pci request is pulled and a real-time one is put in it's place. Workaround: Use only a single arbitration ring on pci bus. GBR and RR can still be uniquely used per device.
- Symptom: GIO burst writes over 256 bytes long, starting on a non-cache line boundry may experiance qcl drop of data in teh middle of the transfer. This is caused when the write buffer is freed after the first qcl packet is sent rather than the last. Workaround: None
- Symptom: When write gather is on, it prevents correct buffer flushing of the read buffers. This will result in possiable data corruption when a device has both write gather and prefetcher on.Workaround: Do not use write gather mode.
- Symptom: When the request que for a pci device (8 entries) fills up and the following happens; the request starts and then is held off by a pio response which is a cache line in size. If during the response operation, another request from the same device can overwrite the cw/address of the pending request that was suspended for the read response. Workaround: Do not allow cache line pio ops to pci or flash.

- Symptom: If a prefetch read stream is flush by an interrupt while the stream is currently active and the interrupt pin is held low for multiple pci clocks, the re-request of the flushed op can de sent multiple times. This is caused by the pci fsm issuing the operation and the interrupt controller marking the op flushed. This has a tiny performance penalty in that the op is fetch twice if it occurs. In a rev B Si, if a write que causes a stall during the same time then the write stall data corruption bug from above can occur. Workaround: Do not flush a currently active dma stream.

12.5 Current Anomalies in Rev C (3) Si

- Symptom: When mapping through external page map ram, parity errors which occur during this operation, the external ram address is not correctly logged in the parity error register. Workaround: re-read map from crosstalk side (pio mode) and look for error.
- Symptom: Arriving requests which do not contain the correct number of micro-packets as defined in the command word cause the request dispatcher to get out of sync. The request dispatcher detects the error and issues an interrupt but a read of the interrupt status register might not be possible due to the error. Workaround: reset the bridge after a error interrupt and read time-out of the interrupt status register. Note: The crossbow will not allow incorrectly sized packets to be sent to the bridge therefore this is a very rare error condition.
- Symptom: The bridge does not assert REQ64* during reset to indicate that the bridge is capable of 64-bit data transfers. Workaround: Additional hardware can be added external to the bridge to drive this signal during reset.
- Symptom: If a prefetch read stream is flush by an interrupt while the stream is currently active and the interrupt pin is held low for multiple pci clocks, the re-request of the flushed op can be sent multiple times. This is caused by the pci fsm issuing the operation and the interrupt controller marking the op flushed. This has a tiny performance penalty in that the op is fetch twice if it occurs. Workaround: None needed
- Symptom: PIOs cause any partial write buffer for, device 0 only, to prematurely flush. There is no correctness issues just a were slight performance penality. Workaround: None needed
- Symptom: Control or LLP register writes can cause a hang if followed by any other internal operation other than a read of the regis-

ter which was written. Workaround: Control or LLP register writes must be followed by a read to the same register before any other internal bridge operations can occur.

- Symptom: Response retries can cause lock-up on outgoing requests. Workaround: Enable transmit interrupt on sender, then rewrite the bridge control register on error. This clears the counter which causes the lock-up.
- Symptom: If a PCI device retries a PIO op forever, the bridge should detect this and generate an interrupt but it does not. On reads the retry count is never incremented, on word writes it is never cleared, and can cause data to be dropped. Workaround: Program the retry count to 0x0 always.
- Symptom: GIO master timeout is broken, the bridge will accept up to 4 gio reads where a response is not generated, the 5th read will hang the bridge. The timeout will generate an interrupt if so enabled but locks up the PIO response buffers. Workaround: Either avoid all probes or reset the bridge after every 4 probes.
- Symptom: If interrupt lines are asserted when the outgoing write request pipe is stalled to crosstalk, the interrupt can be missed and a bogus crosstalk packet can be sent. This bug is can only be generated in GIO mode. Workaround: Do not use the interrupt lines if there is any chance of the outgoing request que backing up.
- Symptom: Bridge always kicks off pci bus masters after they are parked for 16 pci clocks but does not drive the ad, par, and cmd lines. Workaround: pullup those lines if pci bus masters are used.
- Symptom: The bridge does not pass the correct address to the pci bus when mapping to pci i/o space at widget address 0x1_0000_0000. Workaround: use banked space to access pci io/o space.
- Symptom: The bridge pio swapping bits mem_swap and io_swap in the control register which control pio swapping to the large memory and i/o windows do not enable the swapper. Workaround: use banked space access if pio swapping is required to memory of io spaces.

12.6 Current Anomalies in Rev D (4) Si

- Symptom: When mapping through external page map ram, parity errors which occur during this operation, the external ram address

is not correctly logged in the parity error register. Workaround: re-read map from crosstalk side (pio mode) and look for error.

- Symptom: Arriving requests which do not contain the correct number of micro-packets as defined in the command word cause the request dispatcher to get out of sync. The request dispatcher detects the error and issues an interrupt but a read of the interrupt status register might not be possible due to the error. Workaround: reset the bridge after a error interrupt and read time-out of the interrupt status register. Note: The crossbow will not allow incorrectly sized packets to be sent to the bridge therefore this is a very rare error condition.
- Symptom: The bridge does not assert REQ64* during reset to indicate that the bridge is capable of 64-bit data transfers. Workaround: Additional hardware can be added external to the bridge to drive this signal during reset.
- Symptom: If a prefetch read stream is flush by an interrupt while the stream is currently active and the interrupt pin is held low for multiple pci clocks, the re-request of the flushed op can be sent multiple times. This is caused by the pci fsm issuing the operation and the interrupt controller marking the op flushed. This has a tiny performance penalty in that the op is fetch twice if it occurs. Workaround: None needed
- Symptom: PIOs cause any partial write buffer for, device 0 only, to prematurely flush. There is no correctness issues just a were slight performance penalty. Workaround: None needed
- Symptom: Control or LLP register writes can cause a hang if followed by any other internal operation other than a read of the register which was written. Workaround: Control or LLP register writes must be followed by a read to the same register before any other internal bridge operations can occur.
- Symptom: GIO master timeout is broken, the bridge will accept up to 4 gio reads where a response is not generated, the 5th read will hang the bridge. The timeout will generate an interrupt if so enabled but locks up the PIO response buffers. Workaround: Either avoid all probes or reset the bridge after every 4 probes.
- Symptom: Access to external ssram while dma access requires an ate access to external ssram can cause an invalid ate to be returned for the access to ssram. Workaround: Assure that all devices which might request an ate translation using external ssram be disabled during external ssram map updates.

12.7 Future Feature Requests

- Error Interrupts: allow disabled error conditions to be observable in some register.
- Credit count: make the count readable through a register.
- Add write w/ response and graphics flow control.

APPENDIX C

Revision Log

C.1 Additions since Rev1.0

4/24/95 Added:

- Aux Error Command Word Register
- Response Buffer Error Upper Address Register
- Response Buffer Error Lower Address Register
- Modified Reset Interrupt Status Register

C.2 Additions since Rev1.2

2/7/96 Added:

- Clarifications

C.3 Additions since Rev1.3

6/20/96 Added:

- Revision 2.0 Clarifications and Bug fixes

C.4 Additions since Rev 2.0

10/7/96 Added:

- Revision 3.0 Clarifications and Bug fixes

C.5 Additions since Rev 3.0

3/27/97 Added:

- Revision 4.0 Clarifications and Bug fixes