Xbridge ASIC Specification

Revision 0.5

for Rev A Si

Do not copy or distribute this document

Steven Miller

Preface

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

XBridge is a combination of existing components (the Bridge and the Crossbow). I would like to thank the designers of the Bridge, Anan Nagarajan and the designers of the Crossbow, Rob Martin without whom this ASIC would not be possible.

I would like to thank John Keen for his review of this document.

CHAPTER 1 Overview

1.1 Part Name

Part Name: XBridge

SGI Part Number: 099-0182-001

Vendor: IBM

Vendor Part Number: 21L06674

Process Design Technology: SA-12 CMOS 5-Layer Metal

Package: 1088 CCGA 42mm w/ 87 mil columns

Die Size: 10.1mm x 10.1mm

Estimate Gate Count: 1,000 K

1.2 System Architecture Overview

The *XBridge* ASIC is a controller which provides an integration of the Crossbow and Bridge ASIC functionality. To that end, this document is the combination of the Crossbow and Bridge ASIC specifications. The first sections are based on the Crossbow and the latter sections are the PCI Bridge.

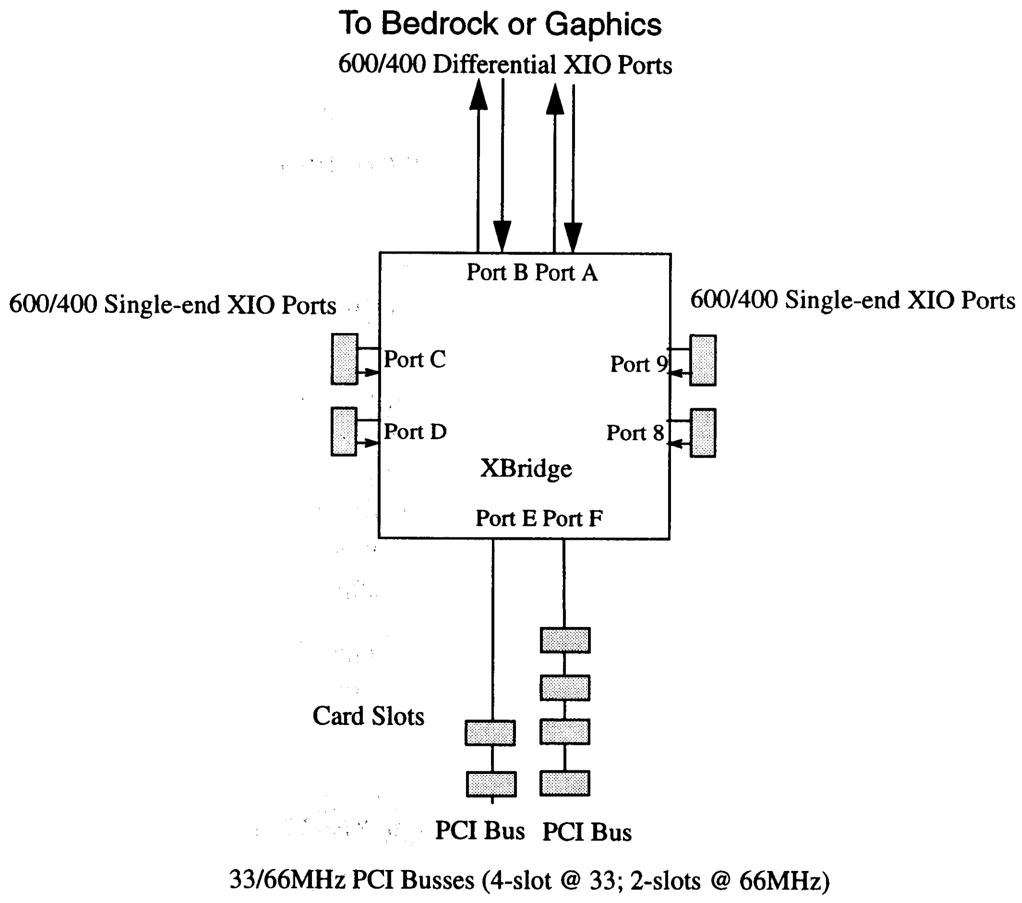

To Bedrock or Gaphics
600/400 Differential XIO Ports
Port B Port A
600/400 Single-end XIO Ports
600/400 Single-end XIO Ports
Port C
Port D
Port 9
Port 8
XBridge
Port E Port F
Card Slots
PCI Bus
PCI Bus
33/66MHz PCI Busses (4-slot @ 33; 2-slots @ 66MHz)

Figure 1 XBridge System Block Diagram

The *XBridge* ASIC is a member of the Godzilla architecture family. Figure 1 contains a block diagram for a typical I/O system using XBridge. For a complete description of the *Godzilla* architecture please refer to the *Godzilla* ***Architecture Specification***. The *XBridge* ASIC can be used to support I/O devices on either XIO or expansion with option card slots on the industry standard PCI bus.

An *XBridge* can operate in two modes, the first (XBridge mode) is as shown in Figure 1, a combination of pci bridges and crossbow. The second mode, (Super Bridge mode) which allows the XBridge to function as

2 Bridge widgets each with a single pci bus. In this mode, four of the crosstalk ports and the internal crossbar are not used (See Figure 2). No communication between the pci busses is allowed internally. The basic operation of XBridge in this mode is like having two Bridge ASICs.

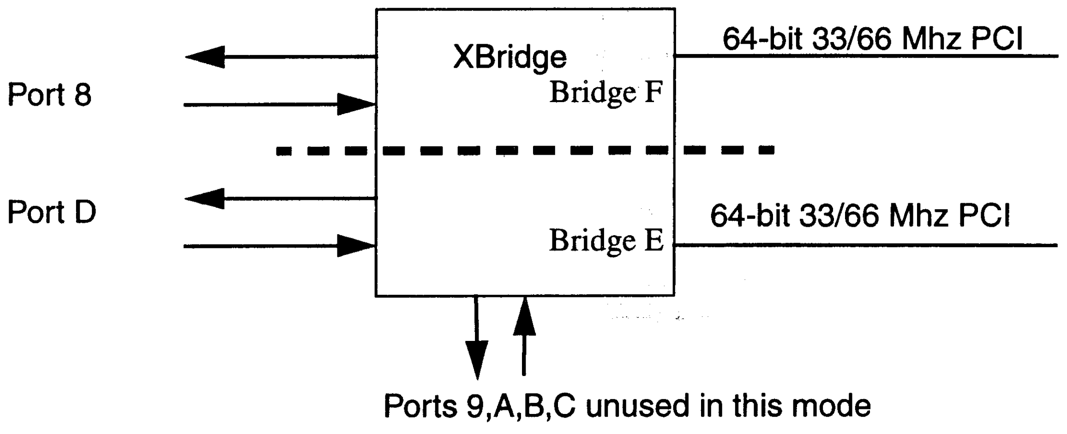


Figure 2 Super Bridge Mode Block Diagram

Shown in Figure 3 is a block diagram of the *XBridge* ASIC. The *XBridge* also contains a JTAG controller for boundary scan capability. The term XIO port is used in the systems environment to indicate a Crosstalk port. This document interchanges these terms throughout.

Figure 3 contains six major modules, the Source Link, the Destination Link, the Crossbar, Widget 0, which are all part of the Crossbow subsection. The Bridge subsection and the LLP.

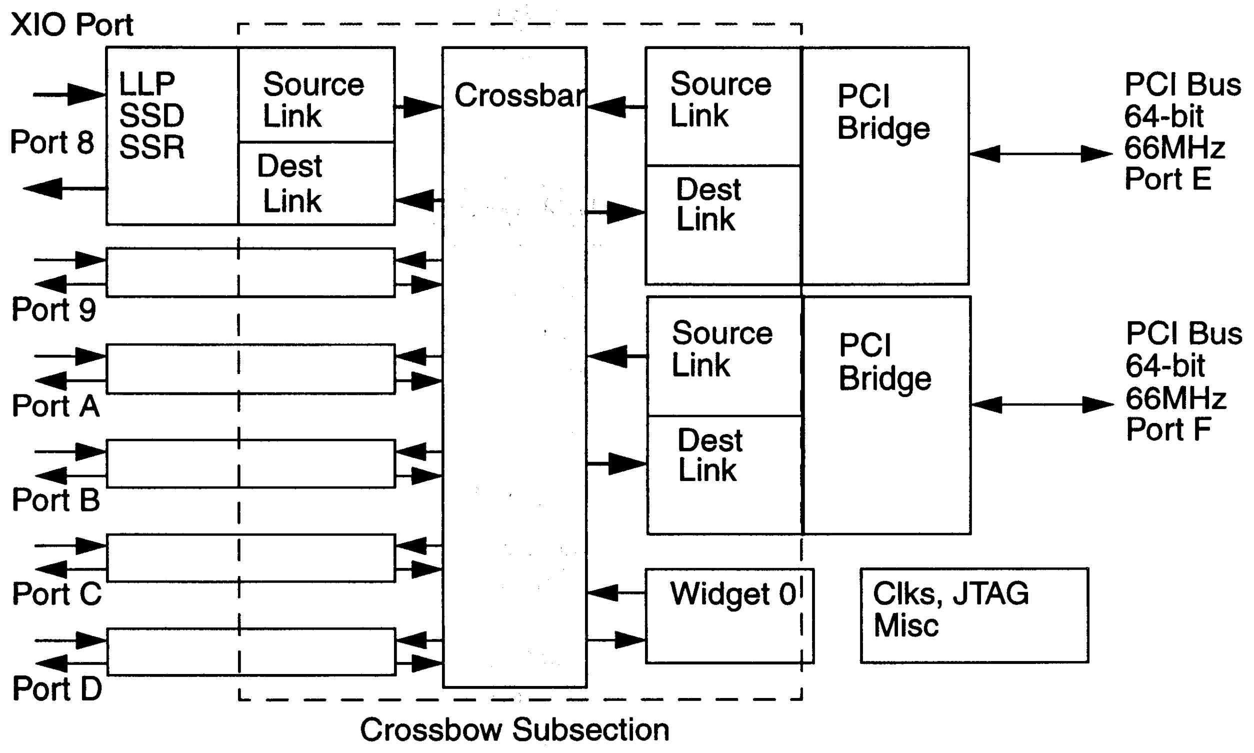


Figure 3 XBridge Block Diagram

1.3 Crossbow Subsection

The Crossbar subsection consists of the Source Link, the Destination Link (Dest Link), the Crossbar, and Widget 0. Detailed operation and diagrams can be found in the Crossbow Subsection Architectural Description Chapter.

1.3.1 Source Link

The source link controller (Source Link) handles all incoming traffic between the physical link interface (LLP) or PCI Bridge and the crossbar. The LLP handles the all Link Protocol and passes complete micropackets to the source link. The PCI Bridge also passes complete micropackets to the source link.

The source link's packet receive control logic scans the sideband data in the micropacket for "crosstalk packet start" code. If this is received the control logic will begin filling one of 4 input packet buffers. The input packet buffer serves two purposes, it provides a place to park a crosstalk packet when the packet destination is busy and it provides for rate matching between the data stream coming from a LLP port operating in 8-bit mode and the crossbar. The Crossbow architecture is such that the crossbar transfers data at 800 Mbytes/sec; however, the LLP's provide data at either 400 Mbytes/sec or 800 Mbytes/sec depending on whether they are associated with 8 bit or 16 bit links respectively. The PCI Bridge always provides data at the 800 Mbytes/sec rate.

The packet receive logic will also write pertinent information from the command word portions of the packet and place it in the crossbar request queue which is located in the request manager section of the source link. The information written into the crossbar request queue defines the Crosstalk packet's destination, priority, and type (request or response). It is the request manager's job to determine which packets are eligible for arbitration and from among the packets that are eligible for arbitration select the packet which has the highest priority and arbitrate for a connection to that packet's destination crossbar port. While the crosstalk packet is being received and put into one of the input packet buffers, the request manager will check the status of the destination port (port_status in the destination link) and the priority of the packets in the queue to determine which of the packets in the input packet buffer has the highest priority. Details of the request manager selection algorithm are left for the architectural description portions of this document.

If the packet which has just entered the queue has the highest priority of all packets currently in the queue, it will advance to the front of the queue and enter the crossbar connection arbitration phase. If there are higher priority crossbar connection requests already in the queue, it will wait until they are serviced.

During the crossbar arbitration phase the request manager sends a crossbar connection request (port_req) to the destination link controller (Dest Link) associated with the Crosstalk packet's destination. The request manager then alerts the packet dispatch control (in the Source Link) that a crossbar connection arbitration is in progress.

When the Crosstalk packet wins arbitration a port_grant signal will be sent back from the destination link controller to the requesting source. The dispatch controller will begin transferring the packet out of the input packet buffer and into the crossbar. The request manager will then retire the entry from the request queue. As the dispatch controller is transferring the packet it monitors the destination's buffer full flag (Buffull). This flag indicates to the source that the destination can currently accept no more data.

When the transfer of the packet nears completion the dispatch controller will release control of the destination port by asserting port_release. This frees the crossbar connection arbiter to start a new arbitration phase and establish a new crossbar connection.

Once the dispatch controller has finished transferring the packet to the destination link, a "credit" is returned to the initiator widget. Credits are returned via the return sideband data stream. The credit indicates an input packet buffer slot has been freed.

1.3.2 Destination Link

The central connection through which all link controllers pass data is the crossbar switch. The crossbar switch consists of 9 ports, 6 of which are used for external XIO ports, 2 for PCI interfaces, and the last for widget 0. Each crossbar connection consists of one 68 bit source port and one 68 bit destination port. The destination ports utilizes a 68 bit wide by 8 mux to establish a connection from the source ports. The 68 bits consist of 64 bits of data, 3 bits of sideband information and 1 bit of control (bit indicates valid data). The destination link controller's crossbar connection arbiter controls the mux. All data is registered in and registered out of the crossbar, hence it is a "one hop" interconnect.

1.3.3 Widget 0

All access to internal registers in the crossbow is via widget 0. Widgets wishing to modify crossbow registers should direct their request packets to the widget 0 destination. Widget 0 behaves much the same as any set of link controllers. Source link controllers wishing to connect to widget 0 send a crossbar connection request to widget 0. The widget 0 crossbar connection arbiter will send an acknowledgment and then receive the packet. After widget 0 has received the packet it will perform the necessary operations on the Crossbow registers. If a response is required, widget 0 will then form a response packet and transfer it back to the initiating widget via the crossbar.

1.3.4 Crossbar Interconnect

The central connection through which all link controllers pass data is the crossbar switch. The destination link controller's crossbar connection arbiter controls the mux. All data is registered in and registered out of the crossbar, hence it is a “one hop” interconnect.

1.4 PCI Bridge Subsection

The figure below contains a block diagram of the PCI bridge sub section in the XBridge ASIC. The major subsections are the Crosstalk Transmit and Receive Processors (CTP & CRP), the Request Dispatcher, the PCI Master and Slave, the Interrupt Controller, the Data Buffers and the Request Generator /Prefetcher.

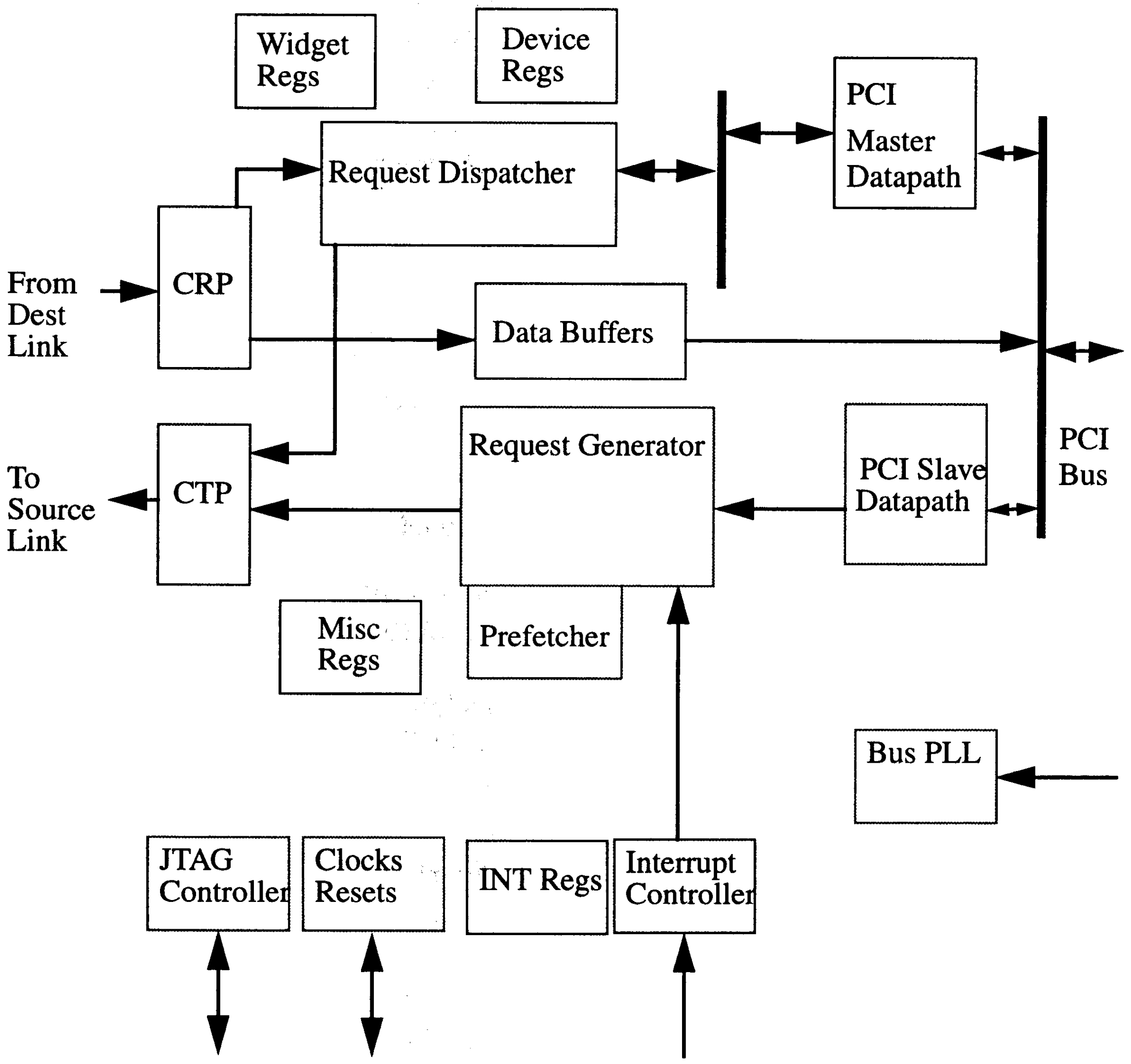


Figure 4 *XBridge* Bridge Subsection Block Diagram

1.4.1 *Crosstalk* Receive and Transmit Processors

The PCI Bridge subsection communicates to the XBridge core through the *Crosstalk* Receive Processor (CRP) and *Crosstalk* Transmit Processor (CTP) CTP blocks. The *Crosstalk* receive and transmit processors provide link control and buffering of packets sent and received from the Crossbar (XBridge mode) or XIO port (Super Bridge mode). The Crosstalk Receive Processor (CRP) receives packets and transfers them to either the data buffers (response packets) or the request dispatcher (request packets). The Crosstalk Transmit Processor (CTP) gathers responses and bus-generated requests and transmits them to other widgets.

1.4.2 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 66 MHz in burst mode which provides a very high host/memory to peripheral transfer rate. For more detailed information please refer to the PCI Local Bus Specification revision 2.1. The bus is processor independent in that there are no processor specific operations which might exclude general purpose processors. 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 via the Crosstalk Interconnect. 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 functionality and best I/O throughput while the I/O bus (PCI) can last for several generations without a need to change. The *XBridge* supports eight PCI devices. (Due to PCI loading restrictions, the *XBridge* can support only four add-in cards, loading will be less when operating at 66MHz.)

1.4.3 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. The *Crosstalk* to PCI bus translation mechanism consists of generating a PCI bus address from the *Crosstalk* address. This translation occurs using fixed regions defined in widget slot space and auxiliary I/O space. For detailed description of the address maps and use please refer to the PCI Bridge Address Mapping Chapter.

1.4.4 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 devices to data residing across the *Crosstalk* Interconnect, the Prefetcher circuit will uses attributes from the address to decide whether to prefetch more data following current address. This enables faster read response time when sequential reads are performed by the PCI I/O devices.

1.4.5 Data Buffers

Data Buffers provide data buffering between *Crosstalk* Interconnect and PCI circuit. It off loads read/write packets from high speed, non-stallable *Crosstalk* Bus to the lower performance, two-way handshaking PCI buses.

1.4.6 Interrupts

Eight external active low interrupt pins from each PCI Bus are connected to the *XBridge*. 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 *XBridge* pin or one device can have multiple interrupt pins to report different cases. Any time there is a change on an interrupt pin, *XBridge* reports that to the host via a double word write packet. (Reporting the low to high transition of the active low pins can be individually disabled.) Each PCI subsection in *XBridge* will also send interrupts to the host when some abnormal conditions occur on the Crosstalk Bus or PCI Bus. These condition can be found in the PCI Bridge Subsection Error Cases Chapter.

Each interrupt condition is individually enabled or disabled by the host. The *XBridge* reports all the interrupt activities to the host while they are enabled. The *XBridge* does not maintain local interrupt priority and no interrupt condition can prevent the reporting of other interrupts.

1.5 Changes in Bridge

1) Remove GIO mode

2) Remove External SSRAM, reflect 0xc0_0000 thru 0xff_ffff on pci bus address 0x0 to 0x3f_ffff.

3) Support 66MHz PCI operation

4) fix the following:

Symptom: Arriving requests which do not contain the correct number of micro-packets as defined in the command word this will 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.

Symptom: The bridge does not assert REQ64* during reset to indicate that the bridge is capable of 64-bit data transfers.

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.

5) Error Interrupts: allow disabled error conditions to be observable in some register.

error view register 0x170

error multiple register 0x178

6) Provide a single NIC pin operated from Widget 0 in XBridge mode, and from Bridge at Port F in Super Bridge Mode.

7) Credit count: make the count readable through a register.

bits 15:8 in status register for tx and tx credit counts

8) Increase the size of the internal page map to either 512 entries and also removing the RMF field from page map (RMF always 0).

9) Add a swap per ATE entry.

10) Clock out 66 and 33 only, no enables.

11) Add view to read response buffer addresses

buffer 0 low 0x300

buffer 0 hi 0x308

. . . .

buffer 15 low 0x3f0

buffer 15 hi 0x3f8

12) Remove the following registers

ssram parity error 0x90

13) remove pending and max trans fields from control register.

14) Add 8 misc inputs for each bridge viewable through bridge

status reg bits 7:0

15) Add 4 more outputs to test outputs on each bridge.

16) Add swap attribute in 64-bit PCI slave mode.

17) remap the rest of pci mapped space to direct space.

18) Add Interrupt generation via PIO in two modes: first, PIO always generates interrupt; second, interrupt pin active and PIO required to generate interrupt.

Will not Fix:

1) Symptom: If a prefetch read stream is flushed 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 fetched twice if it occurs.

2) 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.

1.6 Changes in Crossbow

1.) Source ID field added to the widget 0 status register. Bits 9:6 of the Crossbow Widget 0 Status register now contain a field which indicates the port that the read request originated from. So a widget may determine which port it is connected to by simply reading the widget 0 status register.

2.) Warm Reset Fences added:

Nine 8 bit registers have been added to widget 0. Each register is associated with a xbow port or widget 0. The bits in each register determine which source ports are the port will accept warm resets. These registers are reset only by power on reset.

Field	Bits	Reset *	Access	Description
Reserved	31:8			
WarmReset from F Ok	7	1	R/W	0:Warm resets are blocked if they originate from this block. 1:Warm reset from this block will cause Port 8 to receive a warm reset.
WarmReset from E Ok	6	1	R/W	
WarmReset from D Ok	5	1	R/W	
WarmReset from C Ok	4	1	R/W	
WarmReset from B Ok	3	1	R/W	
WarmReset from A Ok	2	1	R/W	
WarmReset from 9 Ok	1	1	R/W	
WarmReset from 8 Ok	0	1	R/W	

Table 1 Port 8 Reset Fence Register

Register	Address
XBow widget0 reset fence	0x00_0000_0078
XBow Link 8 reset fence	0x00_0000_0080
XBow Link 9 reset fence	0x00_0000_0088
XBow Link A reset fence	0x00_0000_0090
XBow Link B reset fence	0x00_0000_0098
XBow Link C reset fence	0x00_0000_00A0
XBow Link D reset fence	0x00_0000_00A8
XBow Link E reset fence	0x00_0000_00B0
XBow Link F reset fence	0x00_0000_00B8

Table 2 Register Map Changes

3.) General Purpose Lock register added:

An 8 bit register was added as a general lock register. Presumably it will be used to determine who owns the NIC/microlan resource. Upon power on reset the register is cleared to all zeroes. A write to location 0x00_0000_00C0 will occur only if the current value is all zeroes. A write to location 0x00_0000_C8 will clear the register. The register may be read from either location. It is presumed that on power up a host widget will read its source id from the widget0 status register, write this value into the lock register, and then read to see if it obtain the lock. If would then release the lock by writing the lock registers clear address.

4.) Watchdog timers added to Widget0:

Originally widget 0 was not designed to recovered from a crashed host. It was assumed that if the host widget was not responding eventually a reset would be coming. In a dual host situation, a widget 0 that is hung waiting for a destination to respond or a source to release a connection would be a bad thing indeed. If a source does not release the widget 0 port after 8 time-out intervals as set per the Crossbow Packet Time-out register, the arbiter will time-out, set bit 3 in the widget 0 status register to indicate a source time-out occurred. Widget 0 will then complete the transaction and grant the next requestor access to the widget 0 port. Similarly, if a destination port never grants access to a widget 0 request, bit 1 in the widget 0 status register will be set to indicate a destination time-out error occurred and the transaction will complete. This feature is not enabled unless explicitly enabled by setting bit 4 in the widget 0 control register.

1.7 Related Documents

- *Crosstalk* Interconnect Specification, Revision 1.4,
  by Steven Miller, and James Tornes
- Bridge ASIC Specification, Revision 3.0,
  by Steve Miller and Ching Hu
- PCI Local Bus Specification, Revision 2.1
- 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 13.0
  by Pravin Shah, Rob Martin
- Speed Racer Product Specification, Revision 2.0
  by Ross Werner

1.8 Terms

It is assumed the reader has read the Crosstalk Interconnect specification and has thorough knowledge of the Crosstalk Interconnect protocol, packet formats, and electrical characteristics. The following terms are discussed in the Crosstalk Interconnect 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 PACKE*T	The minimum unit of data transfer on the Crosstalk Interconnect. Data transfer on the Crosstalk Interconnect 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. Examples are Crossbow and XBridge.

1.9 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).**

1.10 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.

1.11 Reading This Document

This document like the XBridge itself is a merge of the Crossbow and Bridge ASICs. Every attempt has been made to update the Crossbow and Bridge sections to reflect any changes. Where conflicts exists, please refer to the Register and XBridge Architecture Sections for resolution.

CHAPTER 2

Pin Descriptions

2.1 Pin Descriptions

For more detailed information on Crosstalk Signals please refer to the Crosstalk Interconnect Specification, and PCI Signals please refer to the PCI Local Bus Specification. In addition to signal name and description, the direction and buffer type is listed in the pin table. Xbridge has 3 types of I/O pins, 3.3V compliant, 2.5V compliant and STL. For more information on STL please refer to the Crosstalk Designer's Guide. The term "Crossbar Generated" signals is a generic term used to indicate signals generated by a ASIC which operates as a Crossbar. For the 6 Crosstalk Ports, XBridge operates as a Crossbar in XBridge mode.

2.1.1 *Crosstalk* Interface Pins

Xbridge supports 6 Crosstalk ports indexed 8 through D. The first 2 characters (X8, X9) indicate which port.

Names	I/O	Types	Description
P8_O_CLK_H P8_O_CLK_L	O	Diff STL	***Crossbar Generated Clock*** used when packets are transferred from the crossbar to widget.

Table 3 **Crosstalk Interface Pins**

Names	I/O	Types	Description
P8_O_CD(19:0)	O	STL	***Crossbar** Crosstalk **Data <19:0>*** Micro packet data from crossbar to widget.{synchronous to O_CLK}
P8_O_DRDY_L	O	STL	***Crossbar Data Ready*** indicating the start valid micro packet on link. {synchronous to O_CLK}
P8_I_CLK_H P8_I_CLK_L	I	Diff STL	***Widget Generated Clock*** used when packets are transferred from the widget to the crossbar.
P8_I_CD(19:0)	I	STL	***Widget** Crosstalk **Data <19:0>*** Micro packet data from Bridge to the crossbar.{synchronous to I_CLK}
P8_I_DRDY_L	I	STL	***Widget Data Ready*** indicating the start valid micro packet on link. {synchronous to I_CLK}
P8_VREF(1,2)	I	STL	*Crosstalk **STL Voltage Reference***
P8_LINK ALIVE	O	2.5V CMOS	*Crosstalk **Link Alive*** indicates that the link has completed reset negotiation correctly.
P8_W_PRESENT_N	I	2.5V CMOS	*Crosstalk **Widget Present***
P8_W_SPEED	I	2.5V CMOS	*Crosstalk **Widget XIO Speed 1= 600 0=400***
P9_O_CLK_H P9_O_CLK_L	O	Diff STL	***Crossbar Generated Clock*** used when packets are transferred from the crossbar to widget.
P9_O_CD(19:0)	O	STL	***Crossbar** Crosstalk **Data <19:0>*** Micro packet data from crossbar to widget.{synchronous to O_CLK}
P9_O_DRDY_L	O	STL	***Crossbar Data Ready*** indicating the start valid micro packet on link. {synchronous to O_CLK}
P9_I_CLK_H P9_I_CLK_L	I	Diff STL	***Widget Generated Clock*** used when packets are transferred from the widget to the crossbar.

Table 3 **Crosstalk Interface Pins**

Names	I/O	Types	Description
P9_I_CD(19:0)	I	STL	***Widget*** *Crosstalk* ***Data <19:0>*** Micro packet data from Bridge to the crossbar.{synchronous to I_CLK}
P9_I_DRDY_L	I	STL	***Widget Data Ready*** indicating the start valid micro packet on link. {synchronous to I_CLK}
P9_VREF(1,2)	I	STL	*Crosstalk* ***STL Voltage Reference***
P9_LINK ALIVE	O	2.5V CMOS	*Crosstalk* ***Link Alive*** indicates that the link has completed reset negotiation correctly.
P9_W_PRESENT_N	I	2.5V CMOS	*Crosstalk* ***Widget Present***
P9_W_SPEED	I	2.5V CMOS	*Crosstalk* ***Widget XIO Speed 1= 600 0=400***
PA_O_CLK_H PA_O_CLK_L	O	Diff STI	***Crossbar Generated Clock*** used when packets are transferred from the crossbar to widget.
PA_O_CD(19:0)_H PA_O_CD(19:0)_L	O	Diff STI	***Crossbar*** *Crosstalk* ***Data <19:0>*** Micro packet data from crossbar to widget.{synchronous to O_CLK}
PA_O_DRDY_H PA_O_DRDY_L	O	Diff STI	***Crossbar Data Ready*** indicating the start valid micro packet on link. {synchronous to O_CLK}
PA_I_CLK_H PA_I_CLK_L	I	Diff STI	***Widget Generated Clock*** used when packets are transferred from the widget to the crossbar.
PA_I_CD(19:0)_H PA_I_CD(19:0)_L	I	Diff STI	***Widget*** *Crosstalk* ***Data <19:0>*** Micro packet data from Bridge to the crossbar.{synchronous to I_CLK}
PA_I_DRDY_H PA_I_DRDY_L	I	Diff STI	***Widget Data Ready*** indicating the start valid micro packet on link. {synchronous to I_CLK}
PA_LINK ALIVE	O	2.5V CMOS	*Crosstalk* ***Link Alive*** indicates that the link has completed reset negotiation correctly.
PA_W_PRESENT_N	I	2.5V CMOS	*Crosstalk* ***Widget Present***
PA_W_SPEED	I	2.5V CMOS	*Crosstalk* ***Widget XIO Speed 1= 600 0=400***

Table 3 Crosstalk Interface Pins

Names	I/O	Types	Description
PB_O_CLK_H PB_O_CLK_L	O	Diff STI	***Crossbar Generated Clock*** used when packets are transferred from the crossbar to widget.
PB_O_CD(19:0)_H PB_O_CD(19:0)_L	O	Diff STI	***Crossbar** Crosstalk **Data <19:0>*** Micro packet data from crossbar to widget.{synchronous to O_CLK}
PB_O_DRDY_H PB_O_DRDY_L	O	Diff STI	***Crossbar Data Ready*** indicating the start valid micro packet on link. {synchronous to O_CLK}
PB_I_CLK_H PB_I_CLK_L	I	Diff STI	***Widget Generated Clock*** used when packets are transferred from the widget to the crossbar.
PB_I_CD(19:0)_H PB_I_CD(19:0)_L	I	Diff STI	***Widget** Crosstalk **Data <19:0>*** Micro packet data from Bridge to the crossbar.{synchronous to I_CLK}
PB_I_DRDY_H PB_I_DRDY_L	I	Diff STI	***Widget Data Ready*** indicating the start valid micro packet on link. {synchronous to I_CLK}
PB_LINK ALIVE	O	2.5V CMOS	*Crosstalk **Link Alive*** indicates that the link has completed reset negotiation correctly.
PB_W_PRESENT_N	I	2.5V CMOS	*Crosstalk **Widget Present***
PB_W_SPEED	I	2.5V CMOS	*Crosstalk **Widget XIO Speed 1= 600 0=400***
PC_O_CLK_H PC_O_CLK_L	O	Diff STL	***Crossbar Generated Clock*** used when packets are transferred from the crossbar to widget.
PC_O_CD(19:0)	O	STL	***Crossbar** Crosstalk **Data <19:0>*** Micro packet data from crossbar to widget.{synchronous to O_CLK}
PC_O_DRDY_L	O	STL	***Crossbar Data Ready*** indicating the start valid micro packet on link. {synchronous to O_CLK}
PC_I_CLK_H PC_I_CLK_L	I	Diff STL	***Widget Generated Clock*** used when packets are transferred from the widget to the crossbar.

Table 3 Crosstalk Interface Pins

Names	I/O	Types	Description
PC_I_CD(19:0)	I	STL	***Widget** Crosstalk **Data <19:0>*** Micro packet data from Bridge to the crossbar.{synchronous to I_CLK}
PC_I_DRDY_L	I	STL	***Widget Data Ready*** indicating the start valid micro packet on link. {synchronous to I_CLK}
PC_VREF(1,2)	I	STL	*Crosstalk **STL Voltage Reference***
PC_LINK ALIVE	O	2.5V CMOS	*Crosstalk **Link Alive*** indicates that the link has completed reset negotiation correctly.
PC_W_PRESENT_N	I	2.5V CMOS	*Crosstalk **Widget Present***
PC_W_SPEED	I	2.5V CMOS	*Crosstalk **Widget XIO Speed 1= 600 0=400***
PD_O_CLK_H PD_O_CLK_L	O	Diff STL	***Crossbar Generated Clock*** used when packets are transferred from the crossbar to widget.
PD_O_CD(19:0)	O	STL	***Crossbar** Crosstalk **Data <19:0>*** Micro packet data from crossbar to widget.{synchronous to O_CLK}
PD_O_DRDY_L	O	STL	***Crossbar Data Ready*** indicating the start valid micro packet on link. {synchronous to O_CLK}
PD_I_CLK_H PD_I_CLK_L	I	Diff STL	***Widget Generated Clock*** used when packets are transferred from the widget to the crossbar.
PD_I_CD(19:0)	I	STL	***Widget** Crosstalk **Data <19:0>*** Micro packet data from Bridge to the crossbar.{synchronous to I_CLK}
PD_I_DRDY_L	I	STL	***Widget Data Ready*** indicating the start valid micro packet on link. {synchronous to I_CLK}
PD_VREF(1,2)	I	STL	*Crosstalk **STL Voltage Reference***
PD_LINK ALIVE	O	2.5V CMOS	*Crosstalk **Link Alive*** indicates that the link has completed reset negotiation correctly.
PD_W_PRESENT_N	I	2.5V CMOS	*Crosstalk **Widget Present***

Table 3 **Crosstalk Interface Pins**

Names	I/O	Types	Description
PD_W_SPEED	I	2.5V CMOS	*Crosstalk **Widget XIO Speed 1= 600 0=400***

Table 3 **Crosstalk Interface Pins**

2.1.2 PCI Interface Pins

XBridge supports PCI Bus protocol with 2 PCI busses, one on port E and the other on port F.

Names	I/O	Types	Description
PE_PCI_C_BE(7:0)_N	I/O	PCI 3.3V I/O	***PCI Bus Command/Byte_Enables*** indicates the PCI transfer command and then byte enables during transfer.
PE_PCI_PAR	I/O	PCI 3.3V I/O	***Parity*** across PC_AD(31:0) and PCI_C_BE(3:0)_N
PE_PCI_PAR64	I/O	PCI 3.3V I/O	***Parity*** across PC_AD(63:32) and PCI_C_BE(7:4)_N
PE_PCI_FRAME_N	I/O	PCI 3.3V I/O	***Frame*** is driven by current PCI bus master to indicate the duration of a cycle
PE_PCI_IRDY_N	I/O	PCI 3.3V 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.
PE_PCI_TRDY_N	I/O	PCI 3.3V 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.
PE_PCI_STOP_N	I/O	PCI 3.3V 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.
PE_PCI_DEVSEL_N	I/O	PCI 3.3V 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.

Table 4 **PCI Bus Pins**

Names	I/O	Types	Description
PE_PCI_PERR_N	I/O	PCI 3.3V I/O	***Parity Error*** reports only data parity error on the bus. This signal is driven to high before it is tri-stated.
PE_PCI_SERR_N	I/O	PCI 3.3V I/O	***System Error*** is for reporting address parity error, and other system errors.
PE_PCI_REQ64_N	I/O	PCI 3.3V 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.
PE_PCI_ACK64_N	I/O	PCI 3.3V 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.
PE_PCI_CLK	I	PCI 3.3V I/O	***Bus clock*** input
PE_PCI_AD(63:0)	I/O	PCI 3.3V I/O	***Bus Address/Data*** bus
PE_PCI_REQ(7:0)_N	I	PCI 3.3V I/O	***Request*** indicates to the arbiter that this agent desires use of the bus. Every device has its own request signal.
PE_PCI_GNT(7:0)_N	O	PCI 3.3V I/O	***Grant*** from the XBridge indicates to the agent that access to the bus has been granted.
PE_PCI_INT(7:0)_N	I	PCI 3.3V I/O	***Interrupt*** Request signals from the devices.
PE_PCI_RSTN(3:0)	O	PCI 3.3V I/O	***PCI Bus reset*** signals.
PE_PCI_MIN(7:0)	I	PCI 3.3V I/O	***PCI Bus Misc Input*** Can be read from widget status register
PE_PCI_MOUT(7:0)	O	PCI 3.3V I/O	***PCI Bus Misc Output*** (See Test Pin Control Register Section)
PF_PCI_C_BE(7:0)_N	I/O	PCI 3.3V I/O	***PCI Bus Command/Byte_Enables*** indicates the PCI transfer command and then byte enables during transfer.
PF_PCI_PAR	I/O	PCI 3.3V I/O	***Parity*** across PC_AD(31:0) and PCI_C_BE(3:0)_N

Table 4 PCI Bus Pins

Names	I/O	Types	Description
PF_PCI_PAR64	I/O	PCI 3.3V I/O	***Parity*** across PC_AD(63:32) and PCI_C_BE(7:4)_N
PF_PCI_FRAME_N	I/O	PCI 3.3V I/O	***Frame*** is driven by current PCI bus master to indicate the duration of a cycle
PF_PCI_IRDY_N	I/O	PCI 3.3V 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.
PF_PCI_TRDY_N	I/O	PCI 3.3V 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.
PF_PCI_STOP_N	I/O	PCI 3.3V 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.
PF_PCI_DEVSEL_N	I/O	PCI 3.3V 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.
PF_PCI_PERR_N	I/O	PCI 3.3V I/O	***Parity Error*** reports only data parity error on the bus. This signal is driven to high before it is tri-stated.
PF_PCI_SERR_N	I/O	PCI 3.3V I/O	***System Error*** is for reporting address parity error, and other system errors.
PF_PCI_REQ64_N	I/O	PCI 3.3V 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.
PF_PCI_ACK64_N	I/O	PCI 3.3V 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.
PF_PCI_CLK	I	PCI 3.3V I/O	***Bus clock*** input
PF_PCI_AD(63:0)	I/O	PCI 3.3V I/O	***Bus Address/Data*** bus

Table 4 PCI Bus Pins

Names	I/O	Types	Description
PF_PCI_REQ(7:0)_N	I	PCI 3.3V I/O	***Request*** indicates to the arbiter that this agent desires use of the bus. Every device has its own request signal.
PF_PCI_GNT(7:0)_N	O	PCI 3.3V I/O	***Grant*** from the Bridge indicates to the agent that access to the bus has been granted.
PF_PCI_INT(7:0)_N	I	PCI 3.3V I/O	***Interrupt*** Request signals from the devices.
PF_PCI_RSTN(3:0)	O	PCI 3.3V I/O	***PCI Bus reset*** signals.
PF_PCI_MIN(7:0)	I	PCI 3.3V I/O	***PCI Bus Misc Input*** Can be read from widget status register
PF_PCI_MOUT(7:0)	O	PCI 3.3V I/O	***PCI Bus Misc Output*** (See Test Pin Control Register Section)

Table 4 **PCI Bus Pins**

2.1.3 PCI PLL Pins

Here are some pins used to control the Phase Locked Loops (PLL's)

Names	I/O	Types	Description
PE_SPEED_SEL(1:0)	I	2.5V CMOS	***Port E PCI Clock Speed Select*** these pins select the operational speed for PCI. 00 = 33MHz, 01 = 66MHz, 11 = 100MHz
PF_SPEED_SEL(1:0)	I	2.5V CMOS	***Port F PCI Clock Speed Select*** these pins select the operational speed for PCI. 00 = 33MHz, 01 = 66MHz, 11 = 100MHz
PE_PLL_RST_N	I	3.3V CMOS	***Port E PLL reset*** in pin, discharges the vco.
PF_PLL_RST_N	I	3.3V CMOS	***Port F PLL reset*** in pin, discharges the vco.
PE_PLL_TUNE(5:0)	I	2.5V CMOS	***Port E PCI PLL Tune Bits*** use to tune PLL between 33 and 100 MHz

Table 5 **PCI PLL Pins**

Names	I/O	Types	Description
PF_PLL_TUNE(5:0)	I	2.5V CMOS	***Port F PCI PLL Tune Bits*** use to tune PLL between 33 and 100 MHz
PE_PLL_VDDA	I	2.5V CMOS	PLL VDDA Test Pin
PF_PLL_VDDA	I	2.5V CMOS	PLL VDDA Test Pin
PE_PLL_LOCK	O	2.5V CMOS	PLL Lock Pin
PF_PLL_LOCK	O	2.5V CMOS	PLL Lock Pin
PE_PLL_TESTOUT	O	2.5V CMOS	PLL Testout Pin
PF_PLL_TESTOUT	O	2.5V CMOS	PLL Testout Pin
PE_PLL_TESTIN	I	2.5V CMOS	PLL Testin Pin
PF_PLL_TESTIN	I	2.5V CMOS	PLL Testin Pin

Table 5 **PCI PLL Pins**

2.1.4 Miscellaneous Pins

Here are some pins used to control the define the operating modes of the XBridge .

Names	I/O	Types	Description
PWR_ON_RST_N	I	3.3V CMOS	***Power on Reset*** Cold power on reset to LLP
SIM_RST_N	I	2.5V CMOS	***Simulation Reset*** MUST be tied high.
IXBOW_INTR	O	3.3V CMOS	***Internal Crossbow Link Interrupt***
NIC_IO	I/O	3.3V CMOS	***Number In a Can*** data pin
LLP_CLK_600_H LLP_CLK_600_L	I	PECL	***LLP Clock*** 600 MHz PECL clock input for LLP
LLP_CLK_400_H LLP_CLK_400_L	I	PECL	***LLP Clock*** 400 MHz PECL clock input for LLP
CLK_100_OUT	O	3.3V CMOS	100 MHz clock output derived from 400MHz

Table 6 **Misc Pins**

Names	I/O	Types	Description
CLK_66_OUT	O	3.3V CMOS	66 MHz clock output derived from 400MHz
CLK_33_OUT	O	3.3V CMOS	33 MHz clock output derived from 400MHz
BRIDGE_MODE	I	2.5V CMOS	***Bridge Mode Select*** 1= Super Bridge Mode, 0= XBridge Mode
THERM_RES1,2	I		Thermal Monitor Pins
VDD	I		ASIC power pins
VSS	I		ASIC ground pins

Table 6 Misc Pins

2.1.5 Test Pins

Here are the Bridge test pins for ATPG and JTAG.

Names	I/O	Types	Description
TMS	I	2.5V CMOS	***JTAG Test Mode***
TCK	I	2.5V CMOS	***JTAG Test Clock*** input
TRST_L	I	2.5V CMOS	***JTAG Reset***
TDI	I	2.5V CMOS	***JTAG Data In***
TDO	I	2.5V CMOS	***JTAG Data Out***
LSSD_A	I	2.5V CMOS	IBM Test Pin
LSSD_B	I	2.5V CMOS	IBM Test Pin
LSSD_B2	I	2.5V CMOS	IBM Test Pin
LSSD_C	I	2.5V CMOS	IBM Test Pin
LSSD_ARRAY_C	I	2.5V CMOS	IBM Test Pin
LSSD_JTAG_C	I	2.5V CMOS	IBM Test Pin
IOTEST	I	2.5V CMOS	IBM Test Pin

Table 7 Test Pins

Names	I/O	Types	Description
TESTMODE	I	2.5V CMOS	IBM Test Pin
SCAN_GATE	I	2.5V CMOS	IBM Test Pin
RI	I	2.5V CMOS	IBM Test Pin
DI1	I	2.5V CMOS	IBM Test Pin
DI2	I	2.5V CMOS	IBM Test Pin
LT	I	2.5V CMOS	IBM Test Pin
RE	I	2.5V CMOS	IBM Test Pin
BIST_C	I	2.5V CMOS	IBM Test Pin
TESTM3	I	2.5V CMOS	IBM Test Pin
ABSTRSTL	I	2.5V CMOS	IBM Test Pin
SCAN_OUT13,14	I	2.5V CMOS	IBM Test Pin
PSRO_0,1,2,3	I	2.5V CMOS	IBM Test Pin

Table 7 **Test Pins**

CHAPTER 3 XBridge Architecture

This chapter will describe the differences between XBridge and a combination of Crossbow and Bridge. Operational differences of XBridge and Super Bridge modes will be described as well.

3.1 Core Differences

The following section describes if differences in the XBridge internal Crossbow from a Revision 2.0 Crossbow ASIC and a XBridge internal PCI Bridge from a Revision D Bridge ASIC.

3.1.1 Internal Crossbow Changes

There are no changes to the basic Crossbow RTL from the revision 2.0 source tree. New LLPs in XBridge (Revision 2.5) supports a full asynchronous interface allowing the ports to independently run at either 400 or 600 MBaud. Two of the six XIO ports are differential signaling to aid in cable interconnects.

3.1.2 Internal PCI Bridge Changes

The Bridge section received a modest amount of RTL changes as well as some functional goals changes. The entire list is described in the previous chapter, this section will hit the major items effecting programming and operation.

3.1.2.1 Page Map Space

The external page map ram option has been removed and the internal page map ram has grown to 512 entries. This allows for 8M of mapped area using 16K pages. The Address translation entries has changed, the remote map field has been removed and a per page swap bit (bit 5) has been added.

3.1.2.2 Register Changes

A number of bits have changed in the control and status registers, please refer to the PCI Bridge Register section for those bit changes. Several new register have been added to provide additional features.

In the interrupt area the error interrupt view and multiple interrupt registers have been added. Also force interrupt locations for the 8 external interrupt pins have been added. A PIO write to the corresponding location will either force always, or force when active, an interrupt packet.

Buffer Address Match registers allow visibility into the read response address and status information.

3.1.2.3 Pin additions

Eight general purpose input pins and an additional 4 output pins have been added to aid in support both general workstation functions and PCI hot plug-ability.

3.1.2.4 66MHz PCI Support

The XBridge will support 66MHz PCI options and faster speeds as technology allows.

3.2 XBridge Mode

When operating in XBridge mode the XBridge operates much like a Crossbow with 2 Bridge ASICs connected to ports 0xe and 0xf. In addition to the changes outlined above, the PCI Bridge has the following differences:

The LLP Config and NIC register are not used, changing the value in these registers has no effect on operation. The flash module is accessed through a space provided in the PCI Bridge connected to port 0xf.

The NIC Pin is accessed through the Internal Crossbow widget 0 NIC register.

3.3 Super Bridge Mode

In Super Bridge Mode the entire Internal Crossbow is disabled and no access is provided. Both of the PCI Bridges are connected to external ports. The current plan is to use XD and X8 ports but this may changed based on physical design requirements.

The Bridges will power up as 0xe and 0xf but can be changed. The port which powers up as 0xf will have access to the NIC (through the NIC register). The other Bridge NIC register is unused.

Both the Bridge LLP config registers are used to configure there respective LLPs.

CHAPTER 4

Crossbow Subsection Programmers Interface

4.1 Overview

The Crossbow subsection, although not a physical widget, still has a register set accessed as logical widget 0. There are 2 kinds of registers found in the Crossbow subsection: those which are general to the entire subsection and those which are associated with a link. The per link register set consists of the following:

- Link(x) Input Buffer Flush
- Link(x) Control
- Link(x) Status (Read Only)
- Link(x) Arbitration Upper
- Link(x) Arbitration Lower
- Link(x) Status (Read and Clear)
- Link(x) Reset
- Link(x) Auxiliary Status

Registers common to the entire Crossbow Subsection of XBridge:

- Crossbow Identification
- Crossbow Error Command Word
- Crossbow Error Upper Address
- Crossbow Error Lower Address

- Crossbow Interrupt Destination Upper Address
- Crossbow Interrupt Destination Lower Address
- Crossbow Arbitration Reload Register
- Crossbow Packet Time-out Register
- Crossbow Widget 0 Control Register
- Crossbow Widget 0 Status Register
- Crossbow LLP Control Register
- Crossbow Performance Counter
- Crossbow NIC register
- Crossbow Widget 0 Reset Fence
- Crossbow Link 8 Reset Fence
- Crossbow Link 9 Reset Fence
- Crossbow Link A Reset Fence
- Crossbow Link B Reset Fence
- Crossbow Link C Reset Fence
- Crossbow Link D Reset Fence
- Crossbow Link E Reset Fence
- Crossbow Link F Reset Fence
- Crossbow Lock Register

There are 3 major functions associated with the register set found in the Crossbow Subsection of XBridge: Error handling, Buffer Flushing, and Arbitration Control.

4.1.1 Error Handling

In general when an error occurs in a link controller, the nature of the error is indicated by flags which are asserted in the Link(x) Status register. If there is packet information associated with the error, the information is logged in the Crossbow Error Command Word register, Error Upper Address register, and Error Lower Address register. The Link's Control register contains interrupt enable bits associated with most error conditions on the link. If the interrupt enable bit is set, an interrupt to the host processor will be generated. Only one outstanding interrupt will be generated to the host at anytime regardless of the number of errors that have occurred within the Crossbow subsection.

An interrupt to the host processor consists of a write to an interrupt register in the host. When issuing an interrupt, the Crossbow Subsection of

XBridge will generate two double word write request packets. The first packet is sent when the interrupt condition occurs. The second packet is transmitted when the interrupt condition is cleared in the appropriate status register. The programmer should not disable an interrupt bit. The packets are routed from widget 0, the crossbow, to the destination of the interrupt which is typically the host processor. This destination is indicated by the target id field found in the Crossbow Interrupt Destination Upper Address register. The packets are sent with the address of the destination's interrupt register which is taken from the values programmed into the Crossbow Interrupt Destination Upper Address and Lower Address registers. The data sent with the first packet consists of the interrupt vector value in bits [7:0] of the data field and bit [8] set to a one. When the packet is received by the interrupt destination, bit 8 in the data field indicates it should decode the interrupt vector and set the appropriate bit in its interrupt register. The same data is sent with the second packet with the exception of bit [8] of the data field which is set to a zero indicating to the interrupt destination that it should clear the appropriate bit in its interrupt register. The interrupt vector value is programmed into the interrupt vector field of the Crossbow Interrupt Destination Upper Address register.

When an error occurs, the Crossbow Subsection of XBridge will update the global error registers if necessary and assert error flags in the appropriate Link(x) Status register. If an interrupt for this error condition is enabled, the first interrupt packet to the host will be dispatched. If subsequent errors requiring interrupts occur before the first interrupt condition is serviced, no further interrupts will be dispatched. However, **all** interrupt conditions must be cleared (in the Crossbar subsection) before the second interrupt packet will be dispatched. Interrupt conditions are cleared by a read of the Link(x) Status register at its read and clear address. This operation will return the value of the register and then clear all error condition and pending interrupts. Once all the Link(x) Status registers have been accessed, all interrupt requests should be cleared and the second interrupt packet will be dispatched. A read of the interrupt register in the host will indicate a cleared interrupt condition. If this is not the case it implies that another interrupt has occurred before the prior interrupt service routine could complete.

In a similar manner the global error registers, Crossbow Error Command Word register, Error Upper Address register, and Error Lower Address register, can only accept information from the first erring packet. A write to the Crossbow Error Command Word register will clear the global registers and re-arm them to accept information from future erring packets.

The types of actual errors that can be reported by a link controller are:

- Retry on Link
- Max Retry Limit exceeded
- Packet Time-out
- Overallocated input buffer
- Under allocated Bandwidth
- Invalid Micropacket

The LLP module uses a "go back'n" retry policy to ensure reliable data transmission. This LLP module will indicate when a retry is occurring on the transmit side of the link. It will also indicate when data is received incorrectly on the receive side of the link. To monitor the general integrity of the link, there are two counters in the Link(x) Status register which count the number of retries which have occurred on both the receiving and transmitting link. There are bits in the Link(x) Control which enable interrupt requests to be generated every time a retry occurs or every time a retry counter rolls over. No packet information is logged for this error.

A max retry error occurs when the transmitter has attempted a number of times to transmit the same micropacket and failed. The maximum retry number is based on a programmable value in the Crossbow LLP Control register. When the number is exceeded a flag will be set in the Link(x) Status register. If the max retry interrupt enable bit is set in the Link(x) Control register, an interrupt request will occur. Once a link fails due to a max retry its retry buffer is usually backed up. This causes stale packets destined for the port to remained backed up in source input packet buffers. For this reason the buffer stall signal is removed when a max retry error occurs, thus allowing the packets to be flushed out of their respective source input packet buffers. The packets are essentially dropped. The sources affected when this occurs are indicated by the time-out error location field in the Link(x) Auxiliary Status register.

In addition to max retry error handling, each link controller is responsible for detecting packet time-out errors. There are two basic types of packet time-out errors. A max request time-out error and a source time-out.

A max request time-out occurs when, due to a crosstalk protocol error or the inability of the widget to return request credits on its link, no request packets are allowed to be forwarded from a source to a given destination. When a destination reaches its max request limit a watchdog timer is set. If the destination remains at its max request limit for more than one time-out interval, the watchdog timer will detect this. When this condition occurs, the max request time-out bit in the Link(x) Status register will be set and an interrupt request is generated provided the interrupt condition is enabled. In addition, the port will then accept request packets to allow the packets to be flushed from the source input packet buffers and sent to the destination. All request packet will require proper credits to complete transit to the destination. As before, when this error occurs, the sources affected are indicated by the time-out error location field in the Link(x) Auxiliary Status register.

The other type of packet time-out that can occur is a source time-out. A source time-out occurs when a packet is being streamed through the Crossbow Subsection of XBridge to its destination and transmission of the packet from the source stalls for the same given maximum time interval.

If a source time-out occurs, the Crossbow Subsection of XBridge will complete transmission of the packet with "filler" data and set the appropriate error bits in the packet. The appropriate flag in the Link(x) Status register is asserted, and an interrupt request is generated, provided interrupts are enabled.

The time-out value is set globally for all links by programming the interval into the Crossbow Packet Time-out register. Interrupts on time-out detection may be disabled on a link by link basis by setting the proper values in the Link(x) Control registers.

An overallocated input buffer error occurs when a source widget attempts to transfer a packet to the Crossbow Subsection of XBridge and there is no space available to accept the packet in the input packet buffers. When this condition occurs, the input over-allocation flag is asserted in the Link(x) Status register. An interrupt request will then be generated, provided the proper interrupt enable bit is asserted in the Link(x) Control register.

Please refer to section 3.1.3 Arbitration Control for information on the under allocated bandwidth error.

There are also error conditions specific to widget 0. An error condition occurs when a widget attempts to read from or write to an illegal address in widget 0. An error condition will also occur when widget 0 receives a request packet of any size other than a double word packet or a write request to a read only register. Widget 0 can also experience packet time-out errors when attempting to send response packets and interrupt request packets. This error is logged in the Crossbow Widget 0 Status register, the packet header information is logged in the global error registers (Error Command Word, Error Upper Address, Error Lower Address), and an interrupt request is generated, provided that the Interrupt on Error bit is set in the Widget 0 Control register.

Please refer to Table 3 for a summary of link controller errors, which are errors which can occur during the routing of a Crosstalk packet through the Crossbar. Please refer to Table 4 for a summary of Widget 0 errors that can occur when the Crossbow Subsection of XBridge processes Crosstalk packets during internal register accesses.

LLP asserts squash data	***LLP Receiver Error*** bit in the Link(x) Status register is set. If the ***Enable Interrupt on LLP Receiver Error*** bit is set in the Link(x) Control register an interrupt will be generated.
Receiver retry counter increments from FF -> 00	The ***LLP Receiver Retry Overflow*** bit in the Link (x) Status register is set. If the ***Enable Interrupt on LLP Receiver Retry counter overflow*** bit is set in the Link(x) Control register, an interrupt will be generated.
LLP asserts transmit retry	***LLP Transmitter Retry*** bit in the Link(x) Status register is set. If the ***Enable Interrupt on LLP Transmitter Error*** bit is set in the Link(x) Control register an interrupt will be generated.
Transmit retry counter increments from FF -> 00	The ***LLP Transmit Retry counter overflow*** bit in the interrupt status register is set. If the ***Enable Interrupt on LLP Transmit Retry counter overflow*** bit is set in the Link(x) Control register, an interrupt will be generated.
Transmitter max retry occurs	Fatal Error condition. LLP shuts down, requires link reset from external widget or the port to be reset by a write to the Link(x) Reset register to restart. The ***LLP Max Transmitter Retry*** bit in the Link (x) Status register is set. If the ***Enable Interrupt on LLP Transmitter Max Retry*** bit is set in the Link(x) Control register, an interrupt is generated.The ***Timed out source location*** will be updated in the Link (x) Status register to indicate other ports have been affected by this error.

Table 8 **Link Controller Error Summary**

Received Crosstalk packet cannot be routed because the destination does not map to a legal crossbow port.	The ***Illegal Destination*** bit is set in the Link(x) Status register. An interrupt will be generated if the ***Enable Interrupt on Illegal Destination*** bit in the Link(x) Control register is set. The Crossbow Command Word Error, Error Upper, and Error Lower registers will be updated.
Crosstalk packet is received when input packet buffer is full.	The ***Input Overallocation Error*** bit is set in the Link(x) Status register. An interrupt will be generated if the ***Enable Interrupt on Input Overallocation Error*** bit is set.
While receiving a Crosstalk packet the link stalls during transmission of a Crosstalk packet for a time period > packet time-out value.	The ***Packet Time-out Error Source*** bit is set in the Link(x) Status register. An interrupt will be generated if the ***Source Time-out Interrupt Enable bit*** in the Link(x) Control register is set. The packet will be forwarded to the destination and the un-received portions of the packet will be filled with data which has the Crosstalk Sideband Error bit set.
A Crosstalk destination has reached it Max Request Limit and has remained in this condition for greater than the time-out interval.	The ***Max Request Time-out Error*** bit is set in the Link(x) Status register. An interrupt will be generated if the ***Max Request Time-out Interrupt Enable bit*** in the Link(x) Control register is set. The ***Timed out source location*** will be updated in the Link (x) Status register to indicate which ports have been affected by this error.
For diagnostic purposes, if a source with guaranteed bandwidth requirements has reached its bandwidth allocation, an error will be flagged.	The ***Bandwidth Allocation Error Port ID*** will indicate which sources have been under allocated. An interrupt will be generated if the ***Enable Interrupt on Bandwidth Allocation Error bit*** is set.

Table 8 **Link Controller Error Summary**

Error	Action
Invalid Packet Type: **-Fetch and Op Packet** **-Store and Op Packet** **-Special Request Packet** **-Special Response Packet** **-Reserved Entries** **-Response packets** These are packet operations not supported by the Crossbow Subsection of XBridge. Also, the Crossbow Subsection of XBridge never issues requests which result in responses.	The ***Register Access Error*** bit in the Widget 0 Status register is set. In addition, the 48-bits of the *Crosstalk* address are stored in the Crossbow Error Address Registers and the command word is stored in the Crossbow Error Command Word register. An interrupt will be generated if the ***Interrupt on Register Access Error*** bit is set.
Request Packet Data size / Packet size mismatch or Request Packet Unsupported Data size. All Crossbow Subsection of XBridge register accesses are double word. Any other size packets which are received will cause an error.	The ***Register Access Error*** bit in the Widget 0 Status register is set. In addition, the 48-bits of the *Crosstalk* address are stored in the Crossbow Error Address Registers and the command word is stored in the Crossbow Error Command Word register. An interrupt will be generated if the ***Interrupt on Register Access Error*** bit is set.
Request Packet Command Word Error bit set or Sideband Invalid bit set.	The ***Crosstalk Error*** bit in the Widget 0 Status register is set. In addition, the 48-bits of the *Crosstalk* address are stored in the Crossbow Error Address Registers and the command word is stored in the Crossbow Error Command Word register. An interrupt will be generated if the ***Interrupt on Crosstalk Error*** bit is set.
Request Packet Invalid Address. Indicates that the request packet contains an address not supported by the Crossbow Subsection of XBridge.	The ***Register Access Error*** bit in the Widget 0 Status register is set. In addition, the 48-bits of the *Crosstalk* address are stored in the Crossbow Error Address Registers and the command word is stored in the Crossbow Error Command Word register. An interrupt will be generated if the ***Interrupt on Register Access Error*** bit is set.

Table 9 **Widget 0 Error Conditions**

Error	Action
Destination time-out. Widget 0 attempts to send a response packet back to a requestor or issue an interrupt packet, however the destination does not respond for a time period > 8 packet time-out values.	The ***Destination Time-out Error*** bit in the Widget 0 Status register is set. An interrupt will be generated if the ***Interrupt on Destination Time-out Error*** bit is set.
Receive time-out. A source begins a packet transfer to widget 0 and stalls during the transfer for a time period > packet time-out value.	The ***Receive Time-out Error*** bit in the Widget 0 Status register is set. An interrupt will be generated if the ***Interrupt on Receive Time-out Error*** bit is set.
Arbiter time-out. A source requests access to the widget 0 port and after being granted access does not relinquish port for a time period > packet time-out value.	The ***Arbiter Time-out Error*** bit in the Widget 0 Status register is set. An interrupt will be generated if the ***Interrupt on ArbiterTime-out Error*** bit is set.

Table 9 **Widget 0 Error Conditions**

4.1.2 Buffer Flush mechanism

The buffer flush mechanism in the Crossbow Subsection of XBridge allows a process to guarantee all data in the Crossbow Subsection of XBridge has been flushed from initiator to target. Each link has associated with it a Link Input Buffer Flush register. A read of this register causes all entries in the link's input buffer to be marked. The read response is delayed until all packets in the input buffer have been dispatched to their targets. Once this occurs, the read response returns a value of zero. Only one input buffer flush request may be active in the Crossbow Subsection of XBridge at a time. If another Widget attempts to access any other Link Input Buffer Flush register while a flush is in progress, the Crossbow Subsection of XBridge will return a value of one indicating that the flush request was denied the "lock" and the widget should retry later to obtain it. For examples of system scenarios where this mechanism would be employed, please refer to the Crosstalk specification, section 2.2, ordering.

4.1.3 Arbitration Control

As also detailed in the Crosstalk bus specification, there are two ring priorities of packets received by the Crossbow Subsection of XBridge. These are the guaranteed bandwidth ring (GBR) and the remainder ring (RR). The guaranteed bandwidth ring is for real- time widgets which require a deterministic worst case response time and bandwidth. The remainder ring is for widgets which are non GBR widgets and GBR widgets that have exceeded their bandwidth allocation.

Associated with each link is the Link(x) Arbitration Control Upper register and the Link(x) Arbitration Control Lower register which contain Guaranteed Bandwidth Ring Counts and Remainder Ring Weight values. In the Arbitration Control Upper and Arbitration Control Lower registers, there are seven count values which may be programmed. The seven values are associated with the seven other initiating widgets which may use the link controller's output link as the destination of a packet. The Guaranteed Bandwidth Ring Count represents the maximum number of packets that may be sent from an initiating link to this target link as guaranteed bandwidth packets within a given time period. The time period is fixed globally for all links via the Crossbow Arbitration Reload register. Each link reloads all values in its Arbitration Control register at the start of each arbitration reload interval.

Request packets sent to the Crossbow Subsection of XBridge may have a bit set in the command field requesting arbitration on the guaranteed bandwidth ring. Once a GBR packet is received at the Crossbow Subsection of XBridge's crossbar port arbiter, the link's bandwidth counter for the desired initiator/target path is checked. If the value of the counter is greater than zero, the request is treated as a guaranteed bandwidth request. The crossbar port arbiter, upon receiving the request, will service it in round-robin order with guaranteed bandwidth requests from other links. Whenever a guaranteed bandwidth request packet is serviced, the guaranteed bandwidth count associated with that initiator/target path is decremented. If the counter value reaches zero before being reloaded, all further requests for the remainder of the reload time interval will assume the same priority with the crossbar port arbiter as a remainder ring priority packet. Response packets which have their guaranteed bandwidth bit set always maintain guaranteed bandwidth priority regardless of any other factors and do not affect guaranteed bandwidth counts.

Widgets which participate on the remainder ring share any remaining bandwidth not used by guaranteed bandwidth requestors during the arbitration reload time interval. When the crossbar port arbiter receives an arbitration request from a remainder ring widget, it is serviced in a modified round robin fashion with remainder ring requests from other links. When a remainder ring widget's packet wins arbitration, it will continue to win subsequent arbitration requests provided it is not preempted by a gbr arbitration request, it is continuing to make arbitration requests for subsequent packet transfers, and it has not exceeded its remainder weight value. The remainder weight value is essentially a burst count which allows a widget to transfer n packets, where n is equal to the remainder weight value, before it has to give up the destination port to another remainder ring requestor.

For diagnostic purposes, an interrupt may be generated any time a widget reaches its guaranteed bandwidth count during a reload interval. The condition is flagged when a request packet reaches its guaranteed bandwidth limit. As with other error conditions, the condition is marked in the source Link's status register. The packet's destination ID is also logged in the Link(x) Status register so it can be determined which path was under allocated. Unlike normal error packets, the packet is eventually forwarded to its destination widget. This feature may be enabled on a link by link basis by asserting the ***Bandwidth Allocation Error Enable*** bit in the Link (x) Control register.

4.1.4 Dual Host Support

The following additions were made to Crossbow 2.0 to support dual hosting: reset fences, a resource lock register, and widget 0 watchdog timers which have been included in XBridge.

Associated with widget 0 and each link is a reset fence register. In previous versions of the Crossbow if a warmreset packet was received at any link it was simply propagated to all links and each link would perform a warm reset sequence. In version 2.0 of the Crossbow (and XBridge) one may program a link so it will only selectively accept warmresets that originate from a particular link. So, as an example, by programming the mask in link F's Reset Fence register, one may ensure that only warmresets originating from port B will be propagated to link F. In dual host system this essentially prevents one host from being able to reset a host on another port or to reset widgets associated with the second host. At power on reset the default value is that all masks are enabled. This allows the Crossbow Subsection of XBridge to behave in the same manner as previous versions with no software changes.

A lock register was added to provide a means for software implement a spin lock so resources such as widget0 registers could be shared in a dual host system. A host wishing to obtain a lock spins waiting for the lock registers to equal zero. It then should attempt to write its non-zero lock value into the register. It will then attempt to read back the value to see if the lock was obtained. If the lock was obtained it may release the lock by writing to the lock register clear address.

Watchdog timers were added to widget 0 to ensure that if one host should hang during a transaction, widget 0 will time-out instead of also hanging and still allow access by the second host. The possible time-out conditions are described in Table 9. To facilitate simulation and hardware verification of this feature, two mask bits were added to the widget 0 zero control register to allow, link time-outs to be disabled and widget 0 time-outs to be enabled. As before to match previous version of the crossbow the power on default disables widget 0 time-outs and enables link time-outs.

4.2 Address Map

All registers are 32 bits or less in size and are aligned to a 64 bit boundary with the data residing bits 31:0 (Crosstalk Data_Enable = 0x0f).

The registers can be accessed by Crosstalk double-word packet type only. An access to a reserved address in the Crossbow Subsection of XBridge or an any access other than a double word packet access will result in an error condition. Write accesses to read only and read and clear registers are also flagged as errors.

The Crossbow Subsection of XBridge common registers (Widget 0) start at crosstalk address space 0x00_0000_0000. The link 8 specific registers start at 0x00_0000_0100. Each subsequent group of link specific registers is offset 0x40 from the previous group.(ie. link 9 specific registers start at 0x00_0000_0140, etc.)

Registers	Hwreg name	Addresses	Remarks
Crossbow Identification	XB_ID	0x00_0000_0000	read-only
Crossbow Widget 0 Status	XB_STAT	0x00_0000_0008	read-only
Crossbow Error Upper Addr	XB_ERR_UPPER	0x00_0000_0010	read-only
Crossbow Error Lower Addr	XB_ERR_LOWER	0x00_0000_0018	read-only
Crossbow Widget 0 Control	XB_CTRL	0x00_0000_0020	read/write
Crossbow Packet Time-out	XB_PKT_TO	0x00_0000_0028	read/write
Crossbow Interrupt Destination Upper Address	XB_INT_UPPER	0x00_0000_0030	read/write

Table 10 **Widget 0 Register Address Map**

Registers	Hwreg name	Addresses	Remarks
Crossbow Interrupt Destination Lower Address	XB_INT_LOWER	0x00_0000_0038	read/write
Crossbow Error Command	XB_ERR_CMDWORD	0x00_0000_0040	read/write
Crossbow LLP Control	XB_LLP_CTRL	0x00_0000_0048	read/write
Crossbow Widget 0 Status	XB_STAT_CLR	0x00_0000_0050	read&clear
Crossbow Arbitration Reload	XB_ARB_RELOAD	0x00_0000_0058	read/write
Crossbow Perf Counter A	XB_PERF_CTR_A	0x00_0000_0060	read/write
Crossbow Perf Counter B	XB_PERF_CTR_B	0x00_0000_0068	read/write
Crossbow NIC register	XB_NIC	0x00_0000_0070	read/write
Crossbow Widget 0 Reset Fence	XB_W0_RST_FNC	0x00_0000_0078	read/write
Crossbow Link 8 Reset Fence	XB_L8_RST_FNC	0x00_0000_0080	read/write
Crossbow Link 9 Reset Fence	XB_L9_RST_FNC	0x00_0000_0088	read/write
Crossbow Link A Reset Fence	XB_LA_RST_FNC	0x00_0000_0090	read/write
Crossbow Link B Reset Fence	XB_LB_RST_FNC	0x00_0000_0098	read/write
Crossbow Link C Reset Fence	XB_LC_RST_FNC	0x00_0000_00A0	read/write
Crossbow Link D Reset Fence	XB_LD_RST_FNC	0x00_0000_00A8	read/write
Crossbow Link E Reset Fence	XB_LE_RST_FNC	0x00_0000_00B0	read/write
Crossbow Link F Reset Fence	XB_LF_RST_FNC	0x00_0000_00B8	read/write
Crossbow Lock register	XB_LOCK	0x00_0000_00C0	read/write

Table 10 **Widget 0 Register Address Map**

Registers	Hwreg name	Addresses	Remarks
Crossbow Lock register clear	XB_LOCK_CLR	0x00_0000_00C8	read/write
Reserved		0x00_0000_00D0 through 0x00_0000_00F8	
Link 8 Input Buffer Flush	XB_LINK_IBUF_FLUSH_8	0x00_0000_0100	read-only
Link 8 Control	XB_LINK_CTRL_8	0x00_0000_0108	read/write
Link 8 Status	XB_LINK_STAT_8	0x00_0000_0110	read-only
Link 8 Arbitration Upper	XB_LINK_ARB_UPPER_8	0x00_0000_0118	read/write
Link 8 Arbitration Lower	XB_LINK_ARB_LOWER_8	0x00_0000_0120	read/write
Link 8 Status	XB_LINK_STAT_CLR_8	0x00_0000_0128	read&clear
Link 8 Reset	XB_LINK_RESET_8	0x00_0000_0130	write-only
Link 8 Auxiliary Status	XB_LINK_AUX_STAT_8	0x00_0000_0138	read-only
Link 9 Input Buffer Flush	XB_LINK_IBUF_FLUSH_9	0x00_0000_0140	read-only
Link 9 Control	XB_LINK_CTRL_9	0x00_0000_0148	read/write
Link 9 Status	XB_LINK_STAT_9	0x00_0000_0150	read-only
Link 9 Arbitration Upper	XB_LINK_ARB_UPPER_9	0x00_0000_0158	read/write
Link 9 Arbitration Lower	XB_LINK_ARB_LOWER_9	0x00_0000_0160	read/write
Link 9 Status	XB_LINK_STAT_CLR_9	0x00_0000_0168	read&clear
Link 9 Reset	XB_LINK_RESET_9	0x00_0000_0170	write_only
Link 9 Auxiliary Status	XB_LINK_AUX_STAT_9	0x00_0000_0178	read-only
Link A Input Buffer Flush	XB_LINK_IBUF_FLUSH_A	0x00_0000_0180	read-only
Link A Control	XB_LINK_CTRL_A	0x00_0000_0188	read/write

Table 10 **Widget 0 Register Address Map**

Registers	Hwreg name	Addresses	Remarks
Link A Status	XB_LINK_STAT_A	0x00_0000_0190	read-only
Link A Arbitration Upper	XB_LINK_ARB_UPPER_A	0x00_0000_0198	read/write
Link A Arbitration Lower	XB_LINK_ARB_LOWER_A	0x00_0000_01A0	read/write
Link A Status	XB_LINK_STAT_CLR_A	0x00_0000_01A8	read&clear
Link A Reset	XB_LINK_RESET_A	0x00_0000_01B0	write-only
Link A Auxiliary Status	XB_LINK_AUX_STAT_A	0x00_0000_01B8	read-only
Link B Input Buffer Flush	XB_LINK_IBUF_FLUSH_B	0x00_0000_01C0	read-only
Link B Control	XB_LINK_CTRL_B	0x00_0000_01C8	read/write
Link B Status	XB_LINK_STAT_B	0x00_0000_01D0	read-only
Link B Arbitration Upper	XB_LINK_ARB_UPPER_B	0x00_0000_01D8	read/write
Link B Arbitration Lower	XB_LINK_ARB_LOWER_B	0x00_0000_01E0	read/write
Link B Status	XB_LINK_STAT_CLR_B	0x00_0000_01E8	read&clear
Link B Reset	XB_LINK_RESET_B	0x00_0000_01F0	write_only
Link B Auxiliary Status	XB_LINK_AUX_STAT_B	0x00_0000_01F8	read-only
Link C Input Buffer Flush	XB_LINK_IBUF_FLUSH_C	0x00_0000_0200	read-only
Link C Control	XB_LINK_CTRL_C	0x00_0000_0208	read/write
Link C Status	XB_LINK_STAT_C	0x00_0000_0210	read-only
Link C Arbitration Upper	XB_LINK_ARB_UPPER_C	0x00_0000_0218	read/write
Link C Arbitration Lower	XB_LINK_ARB_LOWER_C	0x00_0000_0220	read/write
Link C Status	XB_LINK_STAT_CLR_C	0x00_0000_0228	read&clear
Link C Reset	XB_LINK_RESET_C	0x00_0000_0230	write_only
Link C Auxiliary Status	XB_LINK_AUX_STAT_C	0x00_0000_0238	read-only

Table 10 **Widget 0 Register Address Map**

Registers	Hwreg name	Addresses	Remarks
Link D Input Buffer Flush	XB_LINK_IBUF_FLUSH_D	0x00_0000_0240	read-only
Link D Control	XB_LINK_CTRL_D	0x00_0000_0248	read/write
Link D Status	XB_LINK_STAT_D	0x00_0000_0250	read-only
Link D Arbitration Upper	XB_LINK_ARB_UPPER_D	0x00_0000_0258	read/write
Link D Arbitration Lower	XB_LINK_ARB_LOWER_D	0x00_0000_0260	read/write
Link D Status	XB_LINK_STAT_CLR_D	0x00_0000_0268	read&clear
Link D Reset	XB_LINK_RESET_D	0x00_0000_0270	write_only
Link D Auxiliary Status	XB_LINK_AUX_STAT_D	0x00_0000_0278	read-only
Link E Input Buffer Flush	XB_LINK_IBUF_FLUSH_E	0x00_0000_0280	read-only
Link E Control	XB_LINK_CTRL_E	0x00_0000_0288	read/write
Link E Status	XB_LINK_STAT_E	0x00_0000_0290	read-only
Link E Arbitration Upper	XB_LINK_ARB_UPPER_E	0x00_0000_0298	read/write
Link E Arbitration Lower	XB_LINK_ARB_LOWER_E	0x00_0000_02A0	read/write
Link E Status	XB_LINK_STAT_CLR_E	0x00_0000_02A8	read&clear
Link E Reset	XB_LINK_RESET_E	0x00_0000_02B0	write-only
Link E Auxiliary Status	XB_LINK_AUX_STAT_E	0x00_0000_02B8	read-only
Link F Input Buffer Flush	XB_LINK_IBUF_FLUSH_F	0x00_0000_02C0	read-only
Link F Control	XB_LINK_CTRL_F	0x00_0000_02C8	read/write
Link F Status	XB_LINK_STAT_F	0x00_0000_02D0	read-only
Link F Arbitration Upper	XB_LINK_ARB_UPPER_F	0x00_0000_02D8	read/write
Link F Arbitration Lower	XB_LINK_ARB_LOWER_F	0x00_0000_02E0	read/write

Table 10 **Widget 0 Register Address Map**

Registers	Hwreg name	Addresses	Remarks
Link F Status	XB_LINK_STAT_CLR_F	0x00_0000_02E8	read&clear
Link F Reset	XB_LINK_RESET_F	0x00_0000_02F0	write-only
Link F Auxiliary Status	XB_LINK_AUX_STAT_F	0x00_0000_02F8	read-only
Reserved		0x00_0000_0300 through 0x00_00FF_FFF F	

Table 10 **Widget 0 Register Address Map**

4.3 Register Definitions

Conventions: All register fields unless otherwise noted will default to zeros after power-up. Fields that are marked as reserved will be ignored when written to and return zeros when read.

4.3.1 Link (x) Input Buffer Flush

This read-only register returns a value of 0 after the input buffer for the addressed port has been flushed. If a flush operation for any port in XBridge is already in progress, a value of one is returned and the flush is not performed.

4.3.2 Link (x) Control Register

The Link Control register is a read/write register which contains the bits used to enable the Crossbow Subsection of XBridge feature set for each link..

Bits	Definition
31	***Enable interrupt on Link_Alive:*** 0 = disable, 1 = enable. default = 0. Allow an interrupt to be generated to alert host if a device comes on-line after initial start-up.

Table 11 **Link(x) Control Register**

Bits	Definition
30	***Reserved***
29:28	***Performance Monitor Mode Select:*** these bits select the performance monitor mode. 00 = no monitoring. 01 = monitor source link. Every time the source link receives a micropacket an increment is sent to the widget 0 performance counter. 10 = monitor destination link. Every time a micropacket (excluding admin packets) is sent by the destination link, an increment is sent to the widget 0 performance counter. 11 = monitor input packet buffer level. Every 100 Mhz clock cycle that the number of locations in the input packet buffer equals the value in the input packet buffer level value, an increment is sent to the widget 0 performance counter. Default = 00.
27:25	***Input Packet Buffer Level:*** for performance monitoring, a way of determining that over a period of time the amount of time there were "n"entries in the input packet buffer. When the performance monitor mode bits are set to "11", each clock cycle the number of crosstalk packets in the input packet buffer is equal to the input packet buffer level. 000 = no entries, 001 = 1 entry, 010 = 2 entries, 011 = 3 entries, 100 = 4 entries, as there are 4 input packet buffer location in the Crossbow Subsection of XBridge, values greater than 4 will be ignored. Default = 000.
24	***Send Bit Mode 8:*** When this bit is set, the link will attempt to come up in 8 bit mode only. 0 = disable, 1 = enable. default = 0.Note: this bit is not reset by a link reset. This allows the programmer to change the sense of ***Send Bit Mode 8*** and then reset the link. The link will then try to come up in the selected mode.
23	***Force Bad LLP Micro-Packet Enable*** this bit enables generation of bad LLP micro packets on first transmission of the packet. The retry will generate a valid packet.1= Force Bad, 0 = normal, default = 0. Note: This bit acts as a "one shot"; after it is set to one, it only remains set for a single cycle and then resets to zero, hence it will always return zero when read from.
22:18	***LLP Widget Credit*** This five bit value is used to determine the maximum number of outstanding request packets that can be sent to a widget. default value = 2.
17	***Enable Interrupt on Illegal Destination:*** 0 = disable, 1 = enable. default = 0.
16	***Enable Interrupt on Overallocated Input Buffer:*** 0 = disable, 1 = enable. default = 0.
15:9	***Reserved***

Table 11 **Link(x) Control Register**

Bits	Definition
8	***Enable Interrupt on Bandwidth Allocation Error***: Enables an interrupt when a Bandwidth Allocation Error occurs. 0 = disable 1 = enable. default = 0.
7	***Enable Interrupt on LLP Receiver Error counter overflow:*** Enables an interrupt to be generated when the LLP receives more than 255 erroneous micropackets. 0 = disable, 1 = enable. default = 0.
6	***Enable Interrupt on LLP Transmit Retry counter overflow:*** Enable Interrupt to be generated when the LLP retries more than 255 erroneous micropackets. 0 = disable 1 = enable. default = 0.
5	***Enable Interrupt on LLP Transmitter Max Retry:*** Enables an interrupt when the LLP indicates that the maximum number of retries to transmit a single micropacket has been reached. 0 = disable 1 = enable. default = 0.
4	***Enable Interrupt on LLP Receiver Error:*** Enables an interrupt to be generated when the LLP receives an erroneous micropacket. 0 = disable 1 = enable. default = 0.
3	***Enable Interrupt on LLP Transmitter Retry:*** Enables an interrupt when the LLP indicates a micropacket retry has occurred. 0 = disable 1 = enable. default = 0.
2	***Unused.*** This bit is writable and readable but has no functional effect.
1	***Max Request Time-out Interrupt Enable*** this bit enables an interrupt when a Max Request Time-out error occurs. 0 = disable 1 = enable, default = 0
0	***Source Time-out Interrupt Enable***: this bit enables an interrupt when the remainder of a packet has not been received from a source within the time out interval. 0 = disable 1 = enable, default = 0.

Table 11 **Link(x) Control Register**

4.3.3 Link (x) Status Register

The Link Status register contains status information for link (x) in the Crossbow Subsection of XBridge ASIC. When the Link (x) status register is read from its read and clear address, the value of the register is returned and all error bits are negated.

A note on the link_alive, max_retry, widget_present, and link_failure bits. There are four possible states for a Crossbow port: (1) it is attempting to come out of reset, (2) it has failed to come out of reset, (3) it is out of reset and functional, (4) it is non-functional due to a failure after it successfully came out of reset. If widget_present is inactive, the port will go from (1) to (2) and stay at (2). When the port is in (1), link_alive, max_retry, and link_failure_mode are all inactive. When the port is in (2), max_retry and link_failure_mode are active. When the port is in (3), link_alive is active. When the port is in (4), max_retry is active.

Bits	Definition
31	***Link_alive:*** indicates that the link came out of reset properly and is operational. 1 = link_alive. 0 = dead_puppy.
30-19	***Reserved***
18	***Multiple_error:*** indicates that while a particular error flag was set, another error of the same kind was detected. Valid for ***Illegal Destination, Input Overallocation Error, Packet Time-out Error Source, Connection Time-out Error, and Packet Time-out Destination*** flags. Multiple_errors detected = 1, no Multiple_errors detected = 0, default 0.
17	***Illegal Destination:*** indicates that the link received a packet with an illegal value in the destination field. Error = 1, no Error = 0, default 0
16	***Input Overallocation Error:*** indicates that the widget attempted to transfer a packet to the XBridge and there was no space in the input packet buffer. Error = 1, no Error = 0, default 0
15:8	***Bandwidth Allocation Error Port ID:*** when a bandwidth allocation error occurs, this field indicates which source was under allocated.Each bit is sticky. If multiple allocation errors occur, multiple bits will be set. Bit 15 = link F,..., 8 = link 8.
7	***LLP receive error counter overflow***
6	***LLP transmit retry counter overflow***
5	***LLP Max Transmitter Retry*** indicates that the max retry count was reached on the transmitter side of the LLP. This error is based on LLP micropackets, not *Crosstalk* packets. Error = 1, no Error = 0, default 0
4	***LLP Receiver error*** indicates that an error was received by the receiver section of the LLP. This error is based on LLP micropackets, not *Crosstalk* packets. Error = 1, no Error = 0, default 0

Table 12 **Link(x) Status Register**

Bits	Definition
3	***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. Error = 1, no Error = 0, default 0
2	***Reserved***
1	***Max Request Timeout:*** indicates that due to a crosstalk protocol error or the inability of the widget to return request credits on its link, this port has allowed no new outstanding requests for more than one time-out interval. Once this error occurs, request packets are allowed to pass to this destination and the sources of all packets sent are logged in the time-out error location field in the auxiliary link status register. Error = 1, no Error = 0, default 0.
0	***Packet Time-out Error Source*** indicates that as a packet was being streamed from a source widget to a destination widget, transmission from the source ceased and the packet timed out. Error = 1, no Error = 0, default 0

Table 12 **Link(x) Status Register**

4.3.4 Link (x) Auxiliary Status

This register is used for additional status information. When the Link(x) Status Register is cleared, this register is also cleared.

Bits	Description
31-24	***LLP receive retry counter*** Default 0x00
23:16	***LLP transmit retry counter*** Default 0x00
15:8	***Timed out source location:*** When a Max Request time-out error or Link Max Retry error occurs, packets are allowed to pass to the destination to allow them to be cleared from the source input packet buffers. They may be considered to be lost packets. This field indicates which sources are effected. Each bit is sticky. If multiple allocation errors occur, multiple bits will be set. Bit 15 = link F, 8 = link 8
15:7	***Reserved***

Table 13 **Link(x) Auxiliary Status**

Bits	Description
6	***Link_failure_mode:*** This bit has meaning only if ***Link_alive*** is deasserted and ***Widget Present*** is asserted. 1 = link never came out of link reset. 0 = if max_retry is set it implies link came out of reset and failed while transferring data.
5	***Widget Present:*** this bit indicates there is a widget physically attached to this port. widget present = 1, widget not present = 0.
4	***Bit mode 8*** this bit indicates whether the link is a 16 bit link or an 8 bit link. 16 bit link = 0, 8 bit link = 1
3:0	***Reserved***

Table 13 **Link(x) Auxiliary Status**

4.3.5 Link (x) Arbitration Upper

The Link Arbitration Upper register is a read/write register which contains GBR and RR arbitration information for a portion of the links in the Crossbow Subsection of XBridge ASIC. The GBR count value indicates the amount of the destination link bandwidth for a link(x) which is allocated to the other source links. Similarly, the remainder weight value is the remainder weight allocated to each of the other source links.

Please note that since a link cannot send packets to itself, (loopback) the value for link A in the Link (A) Arbitration register has no meaning. As an artifact of the Crossbow Subsection of XBridge implementation, this location will be used for widget 0 and should be programmed to 0x1F.

Remainder Weight Value (RWV) 0x0 => 1 crosstalk packet transfer is allowed from this source before next remainder ring source is serviced. RWV 0x7 => 8 crosstalk packet transfers are allowed to occur before next remainder ring source is serviced.

Guaranteed Bandwidth Ring (GBR) 0x00 => source is allowed no GBR priority. All crosstalk request packets sent to this destination from this source have remainder ring priority. GBR 0x01 => source is allowed to transfer one GBR request packet to this destination during the current GBR time interval. etc.

Bits	Definition
31-29	***Remainder Weight Value Count Link B*** Default 0x00
28-24	***Guaranteed Bandwidth Ring Count Link B*** Default 0x00
23-21	***Remainder Weight Value Count Link A*** Default 0x00
20-16	***Guaranteed Bandwidth Ring Count Link A*** Default 0x00
15-13	***Remainder Weight Value Count Link 9*** Default 0x00
12-8	***Guaranteed Bandwidth Ring Count Link 9*** Default 0x00
7-5	***Remainder Weight Value Count Link 8*** Default 0x00
4-0	***Guaranteed Bandwidth Ring Count Link 8*** Default 0x00

Table 14 **Link (x) Arbitration Register**

4.3.6 Link (x) Arbitration Lower

The Link Arbitration register is a read/write register which contains GBR arbitration information for a portion of the links in the Crossbow Subsection of XBridge ASIC.

Bits	Definition
31-29	***Remainder Weight Value Count Link F*** Default 0x00
28-24	***Guaranteed Bandwidth Ring Count Link F*** Default 0x00
23-21	***Remainder Weight Value Count Link E*** Default 0x00
20-16	***Guaranteed Bandwidth Ring Count Link E*** Default 0x00
15-13	***Remainder Weight Value Count Link D*** Default 0x00
12-8	***Guaranteed Bandwidth Ring Count Link D*** Default 0x00
7-5	***Remainder Weight Value Count Link C*** Default 0x00
4-0	***Guaranteed Bandwidth Ring Count Link C*** Default 0x00

Table 15 **Link (x) Arbitration Register**

4.3.7 Link (x) Reset

The Link(x) Reset register provides a means for software to individually reset a widget after the power on reset sequence (link reset). This signal, when asserted, will reset all local and remote LLP circuitry. It forces the LLP to assert the remote link reset signal and causes mode arbitration. A write to this address will act as a “one-shot” forcing link reset to be asserted for 2 core clock cycles. Please refer to the Crosstalk Specification concerning Widget/LLP reset and mode arbitration for greater details.

4.3.8 Crossbow Identification

The Crossbow Identification register is a read-only register used by the host cpu during configuration to determine the type of the Crossbow Subsection of XBridge. The format is the same as defined in IEEE 1149.1 JTAG Device Identification Register.

Bits	Description
31-28	***Revision Number***: current revision of the crossbow starting at 1.
27-12	***Part Number*** "Crossbow Type" = 0xD000
11-1	***Manufacturer Identity*** = 0x00
0	Always read as 1

Table 16 **Crossbow Identification Register**

4.3.9 Crossbow Error Command Word

The Error Command Word register is a read/write register that holds the command word of the packet when an errors occur. Subsequent errors are not logged until this register is written. A write to this register will clear this register and the Crossbow Error Upper and Lower Address registers. Data is ignored on a write. For a complete definition of packet command fields, please refer to the Crosstalk Interconnect specification. The command word register defaults to 0x0.

Bits	Definition
31-28	Destination ID Number (DIDN)
27-24	Source ID Number (SIDN)
23-20	Packet type(PACTYP)
19-15	Transaction NUMber (TNUM)
14	Coherent Transaction (CT)
13-12	Data size(DS)
11	Guaranteed Bandwidth Ring enable (GBR)
10	VBP Message (VBPM)
9	Error Occurred (ERROR)
8	Barrier
7-0	Reserved

Table 17 **Crossbow Error Command Word Register**

4.3.10 Crossbow Error Upper Address

The Crossbow 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 Error Command Word register is written.

Bits	Description
31-16	Reserved
15-0	Address Bits 47-32, Default 0x0

Table 18 **Crossbow Error Upper Address Register**

4.3.11 Crossbow Error Lower Address

The Crossbow 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 Error Command Word register is written.

Bits	Description
31-0	Address Bits 31-0, Default 0x0

Table 19 **Crossbow Error Lower Address Register**

4.3.12 Crossbow Interrupt Destination Upper Address

The crossbow interrupt destination upper address register is a read/write register containing the upper 16 bits of address of the register in the host to which the interrupt write request packet is targeted. The target id contains the destination of the interrupt packet and the interrupt vector contains a portion of the data to be sent in the data field of the interrupt packet.

Bits	Description
31-24	***Int_Vector***: Interrupt vector value, Default 0x00
23:20	***Reserved***
19:16	***Target ID Number*** for interrupt destination, Default 0x8 Note: bit 19 is always 1, because 0-7 are not valid interrupt target IDs.
15-0	Address Bits 47-32, Default 0x0000

Table 20 **Crossbow Interrupt Destination Upper Address Register**

4.3.13 Crossbow Interrupt Destination Lower Address

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

Bits	Description
31-0	Address Bits 31-0, Default 0x00000000

Table 21 **Crossbow Interrupt Destination Lower Address Register**

4.3.14 Crossbow Arbitration Reload

The Crossbow arbitration reload register determines the reload interval for the GBR counters Range is 0 to 80 usec in 1.28 usec steps. Value should be > 0 for proper operation. New GBR reload interval values take effect after the current GBR reload interval has completed.

Bits	Description
31-6	Reserved
5-0	***GBR Reload Interval***(1.28 usec/tic) Default = 0x002

Table 22 **Crossbow Arbitration Reload Register**

4.3.15 Crossbow Packet Time-out register

The value in this register determines the time-out interval for packets in the Crossbow. If a packet has been received in a link's input packet buffer and has not progressed through the crossbow in a minimum of one and a maximum of two timer intervals, a time-out error will occur. The time interval is computed as: (interval value x 1.28 usec).Value should be > 0 for proper operation. Note: the first time-out interval after reset is 2.56 usec. The next time-out interval will be 0xFFFFF x 1.28 usec unless a new value has been written first. New time-out interval values take effect after the current time-out interval has completed.

Bits	Description
31-20	Reserved
19-0	Packet time-out interval value(1.28usec/tic) Default = 0x0FFFFF

Table 23 **Crossbow Packet Time-out register**

4.3.16 Crossbow Widget 0 Status

This register contains the error conditions which may result when writing to internal registers in teh Crossbow subsection of XBridge. All error status bits default to 0. A read of this register from its read&clear address will clear the Register Access, Crosstalk, and, as a side effect, the Widget 0 Interrupt bit. The Link Interrupt Request bits will not be cleared until their respective status registers are cleared. An active interrupt request bit indicates only that the link or widget 0 attempted to send an interrupt; it may have been failed due to another outstanding interrupt. See the section on Error Handling for more information.

Bits	Description
31	***Link F Interrupt Request***: Link F has observed an error whose interrupt bit is enabled.
30	***Link E Interrupt Request***: Link E has observed an error whose interrupt bit is enabled.
29	***Link D Interrupt Request***: Link D has observed an error whose interrupt bit is enabled.
28	***Link C Interrupt Request***: Link C has observed an error whose interrupt bit is enabled.
27	***Link B Interrupt Request***: Link B has observed an error whose interrupt bit is enabled.
26	***Link A Interrupt Request***: Link A has observed an error whose interrupt bit is enabled.
25	***Link 9 Interrupt Request***: Link 9 has observed an error whose interrupt bit is enabled.
24	***Link 8 Interrupt Request***: Link 8 has observed an error whose interrupt bit is enabled.
23	***Widget 0 Interrupt Request***: Widget 0 has observed an error whose interrupt bit is enabled.
22-10	Reserved
9-6	***Source ID:*** This field is used for external widgets to determine the correct value to place in their source id crosstalk packet field. This field indicates the crossbow port/ link ID from which the crossbow widget 0 status register read request originated.
5	***Register Access Error:*** Indicates a register access violation, such as an attempt to access an address mapped to a reserved portion of widget 0 address space, an attempt to write a read-only register, an attempt to read a write-only register, an attempt to send a request packet size to widget 0 which is not double word size. Error = 1, No error = 0. Default = 0.

Table 24 **Crossbow Widget 0 Status Register**

Bits	Description
4	***Widget 0 Receive time-out Error***: Indicates that a source link began a transfer to widget 0 and did not complete the transfer before eight time-out intervals occurred.Error = 1, No error = 0. Default = 0
3	***Widget 0 Arbiter time-out Error***: Indicates that a source link arbitrated for widget 0 and after being granted access did not release widget 0 for eight time-out intervals. Error = 1, No error = 0. Default = 0
2	***Crosstalk Error***: A crosstalk packet was received with a sideband error bit set or the error bit set in the command word portion of the packet. Error = 1, no Error = 0, default 0
1	***Widget 0 Destination time-out Error***: Indicates that widget 0 has tried for eight time-out intervals to try to transfer a packet to a destination link.Error = 1, No error = 0. Default = 0
0	***Multiple_error:*** indicates that while a particular widget 0 error flag was set, another error of the same kind was detected. Multiple_errors detected = 1, no Multiple_errors detected = 0, default 0.

Table 24 **Crossbow Widget 0 Status Register**

4.3.17 Crossbow Widget 0 Control

This register controls error reporting for widget 0 accesses.

Bits	Description
31-8	Reserved
7	***Enable time-out counter for Links 8-F:*** Enables the watchdog time-out counters for all of the links. Enable = 1,Disable = 0. Default = 1.
6	***Enable time-out counter for Widget 0:***Enables the watchdog time-out counters for widget 0 arbiter, receiver, and destination time-out counters. Enable = 1,Disable = 0. Default = 0.
5	***Interrupt on access error:*** Enable interrupt on register access error. Enable = 1,Disable = 0. Default = 0.

Table 25 **Crossbow Widget 0 Control Register**

Bits	Description
4	***Interrupt on Widget 0 Receive time-out:*** Enable interrupts for widget 0 when a receive time-out occurs. Enable = 1, Disable = 0. Default = 0.
3	***Interrupt on Widget 0 Arbiter time-out:*** Enable interrupts for widget 0 when arbiter time-out occurs. Enable = 1, Disable = 0. Default = 0.
2	***Interrupt on Crosstalk Error:*** Enable interrupts for widget 0 when crosstalk error occurs. Enable = 1, Disable = 0. Default = 0.
1	***Interrupt on Widget 0 Destination time-out:*** Enable interrupts for widget 0 when destination time-out occurs. Enable = 1, Disable = 0. Default = 0.
0	Reserved

Table 25 **Crossbow Widget 0 Control Register**

4.3.18 Crossbow LLP Control

This register controls the parameters for the LLP modules in the crossbow.

Bits	Description
31-26	Reserved
25-16	***Max Micropacket Retry***: Represents the number of time a micropacket transmission will be retried before the link will Retry timeout. Default =0x3FF
15-10	***Null time -out:*** Idle time (in clocks) required before a LLP null micro-packet is transmitted. Minimum value is 3. Default = 0x06
9-0	***Max Micropacket Burst:*** Maximum number of micropackets that may be continuously transmitted before a null cycle is inserted. Default =0x010.

Table 26 **Crossbow LLP Control Register**

4.3.19 Crossbow Performance Counter A

This register contains the performance counter and the select signals which indicate which link's performance will be monitored. Only bits 22:20 will be affected by a write operation.

Bits	Description
31-23	Reserved
22:20	***Link Performance Monitor Select:*** determines which link will control the incrementing of the performance counter. 000 = Link 8 ,.......,111 = link F. Default = link 8
19:0	***Performance Counter Value***

Table 27 **Crossbow Performance Counter A**

4.3.20 Crossbow Performance Counter B

Same a performance counter A. Use of A and B allows monitoring of a source and a destination simultaneously.

Bits	Description
31-23	Reserved
22:20	***Link Performance Monitor Select:*** determines which link will control the incrementing of the performance counter. 000 = Link 8,.......,111 = link F. Default = link 8
19:0	***Performance Counter Value***

Table 28 **Crossbow Performance Counter B**

4.3.21 Crossbow NIC register

This register provides access to the Number-In-a-Can serial PROM that is used to provide info such as board revisions numbers. (Reference NIC datasheet Dallas 1990A).

Data comes from this interface LSB first.

Bits	Description
31-20	Reserved
19:10	***NIC_BMP:*** determines Bus Master pulse duration in uS. Default = 0
9:2	***NIC_OFSSET:*** Sampling offset relative to the deassertion of the Bus Master strobe. Default = 0
1	***NIC_DATA_VLD:*** Sample Data Valid. Default = 1.
0	***NIC_DATA:*** Sample Data. Default = 0.

Table 29 **Crossbow NIC register**

The four MicroLAN primitives are implemented by writing the specified value to the Crossbow NIC register ,then polling NIC_DATA_VLD until either the bit write completes, or the read completes with data in the NIC_DATA field, The primitives are implemented as described in the following table. All values are given in decimal.

Primitive	NIC_ BMP	NIC_OFF SET	NIC_DATA_ VLD	NIC_DATA	Description
Reset-and-Pres-ence-Pulse	500	65	0	0	The IOC3 will drive the NIC data line for 500us, and sample the NIC data line 65 us following deassertion by the IOC3.
Write-1					The IOC3 will drive the NIC data line for 5us, and sample the NIC data line 80 us following deassertion by the IOC3. The read data is don't-care.

Table 30 **MicroLAN interface usage**

Primitive	NIC_ BMP	NIC_OFF SET	NIC_DATA_ VLD	NIC_DATA	Description
Write-0	80	1	0	0	The IOC3 will drive the NIC data line for 80us, and sample the NIC data line 1 us following deassertion by the IOC3. The read data is don't-care.
Read	5	5	0	0	The IOC3 will drive the NIC data line for 5us, and sample the NIC data line 5 us fol lowing deassertion by the IOC3.

Table 30 **MicroLAN interface usage**

Bits	Field	Description
<7:0>	FAMILY_CODE<7:0>	Reads as 0x01
<55:8>	MAC_ADDR<47:0>	Unique MAC address(or board info)
<63:56>	CRC<7:0>	CRC over MAC_ADDR and FAMILY_CODE

Table 31 **Number in a can contents**

This data comes from the device least-significant-bit first. See "The Book of DS19xx Touch Memory Standards" for the CRC algorithm. That book also contains the algorithm for reading the device. Basically, the algorithm consists of issuing a Reset-and-Presence-Pulse, followed by writing a READ_ROM command (0x33) using the Write-1 and Write-0 operations, LSB-first, followed by 64 Read operations to obtain the data word.

4.3.22 Crossbow Widget 0 Reset Fence

This register determine which widgets/links widget 0 will accept warm reset from. The intent of this registers is to establish reset fences which will restrict where warm resets are propagated to in the crossbow.

Bits	Description
31-8	Reserved
7	***Warm reset from widget F Ok:*** Enable = 1, Disable = 0, Default = 1.
6	***Warm reset from widget E Ok:*** Enable = 1, Disable = 0, Default = 1.
5	***Warm reset from widget D Ok:*** Enable = 1, Disable = 0, Default = 1.
4	***Warm reset from widget C Ok:*** Enable = 1, Disable = 0, Default = 1.
3	***Warm reset from widget B Ok:*** Enable = 1, Disable = 0, Default = 1.
2	***Warm reset from widget A Ok:*** Enable = 1, Disable = 0, Default = 1.
1	***Warm reset from widget 9 Ok:*** Enable = 1, Disable = 0, Default = 1.
0	***Warm reset from widget 8 Ok:*** Enable = 1, Disable = 0, Default = 1.

Table 32 **Crossbow Widget 0 Reset Fence**

4.3.23 Crossbow Link 8 Reset Fence

This register determine which widgets/links Link 8 will accept warm reset from. The intent of this registers is to establish reset fences which will restrict where warm resets are propagated to in the crossbow.

Please note for normal operation a link 8 should always be enabled to accept a warm reset from itself. In other words, bit 0 should always be enabled. Similarly bit 1 should be enabled in the Crossbow Link 9 Reset Fence register, bit 2 should be enabled in the Crossbow Link A Reset Fence register ...etc.

The same register definition follows for Crossbow Link 9 - F Reset Fence registers.

Bits	Description
31-8	Reserved
7	***Warm reset from widget F Ok:*** Enable = 1, Disable = 0, Default = 1.
6	***Warm reset from widget E Ok:*** Enable = 1, Disable = 0, Default = 1.
5	***Warm reset from widget D Ok:*** Enable = 1, Disable = 0, Default = 1.
4	***Warm reset from widget C Ok:*** Enable = 1, Disable = 0, Default = 1.
3	***Warm reset from widget B Ok:*** Enable = 1, Disable = 0, Default = 1.
2	***Warm reset from widget A Ok:*** Enable = 1, Disable = 0, Default = 1.
1	***Warm reset from widget 9 Ok:*** Enable = 1, Disable = 0, Default = 1.
0	***Warm reset from widget 8 Ok:*** Enable = 1, Disable = 0, Default = 1.

Table 33 **Crossbow Link 8 Reset Fence**

4.3.24 Crossbow Lock Register

This register is used as a general resource for software to establish a lock on crossbow resources such as widget 0 registers. At power on reset the register is cleared. This register may only be written when the value of the register is zero. The intended use is that a "lock" may be obtained by a device writing its non-zero lock id in the register. Any other device that attempts to write the register will then be prevented from doing so. After writing the register the device should read register and compare it against its lock id to determine if it obtained the lock. Once the device wishes to give up the lock it should then write to the Crossbow Lock register clear address. The value in the Crossbow lock register may be read by a read access to either the Crossbow Lock register clear address or simply the Crossbow Lock register address.

Bits	Description
31-8	Reserved

Bits	Description
7-0	***Lock Value:*** Default = 00000000

Table 34 **Crossbow Link 8 Reset Fence**

4.4 Function Errata

4.4.1 Double Overflow

Double overflow can hang source and destination ports. An overflow occurs when all entries in the input packet buffer are filled and another packet is received. This condition may only occur if the widget is improperly programmed to believe the Crossbow subsection of XBridge has greater than 4 input packet buffers.

4.4.2 Supported Crosstalk Packet Types

(This is meant as a note for future designers that may want to use reserved crosstalk packet types)

When it receives a crosstalk packet, XBidge is hard-wired to route a fixed number of upkts for that packet, based on a decode of the packet type and data size fields (i.e. it ignores the tail bit). The following size table is currently used:

pac_typ	data_sz	upkts
0000	XX	1
0001	00	1
0001	01	3
0001	10	9
0001	11	9
0010	00	2
0010	01	3

Table 35 **Supported Crosstalk Packet types**

pac_typ	data_sz	upkts
0010	10	9
0010	11	9
0011	XX	1
0100	00	2
0101	00	1
0110	XX	1
0111	00	1
1000	00	2
1001	00	1
1010	00	1
1011	00	1
1100	00	1
1101	00	1
1110	XX	9
1111	XX	9

Table 35 **Supported Crosstalk Packet types**

CHAPTER 5

Crossbar Subsection Architectural Description

5.1 Overview

The Crossbow Subsection of XBridge chip consists of the following major blocks:

- Crossbar switch
- link controller
- Widget 0 (internal registers and error processing)

The core of the Crossbow Subsection of XBridge chip consists of a 9 port, 68 bit crossbar switch. Each port on the crossbar consists of a 68 bit output bus and a 68 bit input bus. The busses consists of 64 bits of packet data, 3 bits of sideband data (packet_start,packet_stop,invalid packet), and 1 bit of control information (xbar_data_valid). Each port, with the exception of widget 0, is connected to a link controller.

A link controller consists of all the necessary circuitry to transmit and receive data on the links via the crosstalk protocol. In addition it contains all necessary buffers and arbitration logic to control data flow to and from the links.

As shown in Figure 5, each link controller occupies one port on the crossbar switch. The Crossbow Subsection of XBridge supports six 16 bit links and 2 links to a PCI interface. Data is transferred through the crossbar at a data rate equivalent to that required for a 400 Mbaud 16 bit link.

The 8 bit ports contain rate matching buffers which allow them to transmit and receive from the crossbar at the 16 bit data rate and transmit and receive data on their physical links at half the data rate of the crossbar. 16 bit ports may be configured to run as 8 bit ports and therefore also contain rate matching buffers.

Widget 0 occupies one port on the crossbar. Widget 0 handles the accessing of Crossbow Subsection of XBridge internal registers and performs error handling. When a link controller detects an error in a Crosstalk packet, it forwards the packet to the widget 0 port and the widget 0 error handling logic updates the necessary registers and issues interrupts if so enabled. All read and write requests of internal registers in the Crossbow Subsection of XBridge are in the form of request packets sent to widget 0.

A more detailed description of these blocks follow.

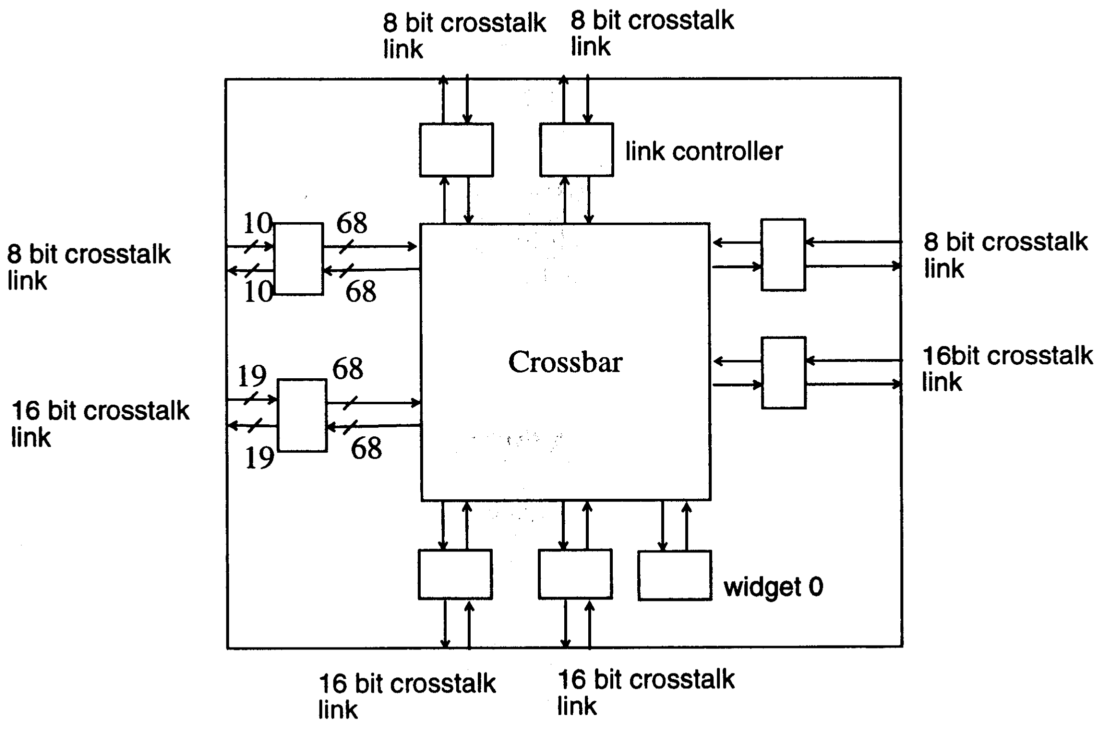


Figure 5 Crossbow Block Diagram

5.1.1 Clocking regimes

The majority of the Crossbow Subsection of XBridge operates off single 100 Mhz clock known as the core_clock. Unless otherwise noted, it should be assumed that any blocks discussed are in the core_clock clock regime.

The two notable exception are the Crosstalk link receivers and Crosstalk link transmitters which are known as respectively as the Source Synchronous Receiver (SSR) and Source Synchronous Driver(SSD).

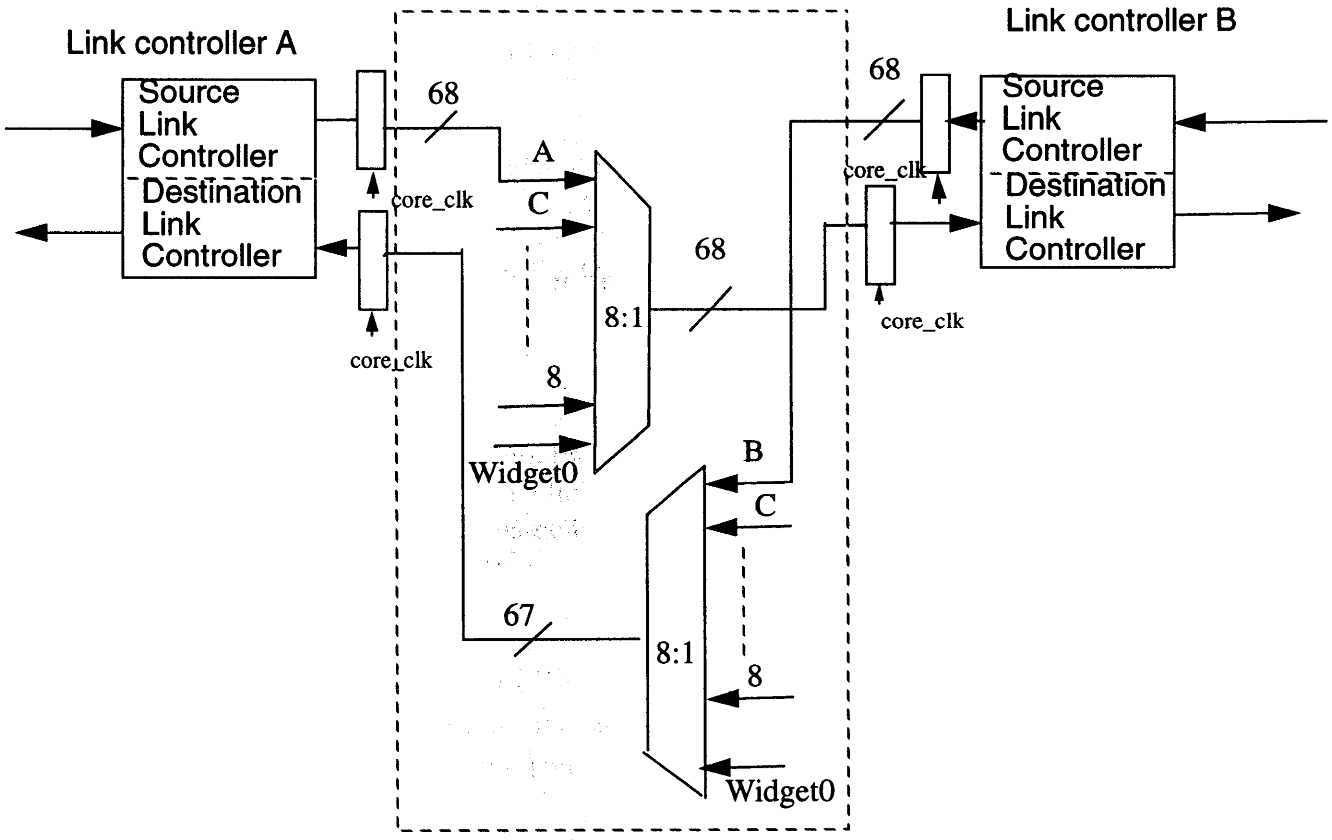
Each SSD receives a 400 Mhz clock. This in turn is used to generate a local 100 Mhz clock, local_100, which is used internal to the SSD. Since the SSD is a fixed standard cell layout, the delay from the 400 Mhz clock to the internal local_100 clock is known.

One of the received 400 Mhz clocks is used to generate the 100 Mhz core_clock. The SSD assumes that there is a fixed relationship between the local_100 clock and the core_clock. This relationship is achieved by using a PLL that will cancel the skew between one of the locally generated 100 Mhz clocks and the core_clock. The core_clock will then be delayed to have a fixed relationship between core_clock and the local 100 Mhz clocks.

Each SSR uses a clock which is transmitted with the data and there is a assumed relationship between the frequency of the transmitted clock and frequency of the core_clock. However, there is no phase relationship assumed and no skew requirements. The boundary between the two regimes may be treated as asynchronous for layout purposes

5.2 Crossbar Switch

The crossbar switch is made up of nine 68 bit wide 8:1 muxes. Any of the source ports can be connected concurrently to any of the destination ports with one significant exception. The structure of the crossbar does not allow the connection of a link controller's source crossbar port to its own destination crossbar port (no loopback). Also, only single point-to-point connections are supported (i.e. no broadcast mode). At present, it is assumed that the Crossbar switch interconnect will be traversed by data in one core clock cycle. Hence, it is necessary for source link controllers to drive the crossbar with registered outputs and it is necessary for destination link controllers to register the data in.



TOTAL: NINE 68 bit 8:1 Muxes

Figure 6 Crossbar Switch

5.3 Link Controller

There are essentially two kinds of link controllers. Link controllers differ based on the size of the link port to which they are connected. An 8 bit port link controller, as the name implies, controls traffic on 8 bit links and operates only in 8 bit mode. A 16/8 port link controller may operate in either 8 bit mode or 16 bit mode, hence they may connect to 8 bit or 16 bit widgets. In truth, the two types of link controllers differ only in their LLP modules and the controllers which interface to the LLP's.

The Link Controller actually consists of two fairly distinct modules. The Source Link Controller handles all packet traffic from the source link to the crossbar.

Conversely, the Destination Link Controller handles all traffic from the crossbar to the destination link. The two modules are more or less completely independent with the exception of status information that is passed from the Source Link Controller to the Destination Link Controller and transmitted back to the initiator widget via the destination link.

5.3.1 Source Link Controller

The source link controller consist of the following submodules:

- Source Synchronous Receiver (SSR)
- Link Level Protocol Receiver (LLPr)
- Input Packet Buffer
- Packet Receive Control
- Free Buffer Stack
- Crossbar Request Manager
- Packet Dispatch Control
- Exception module

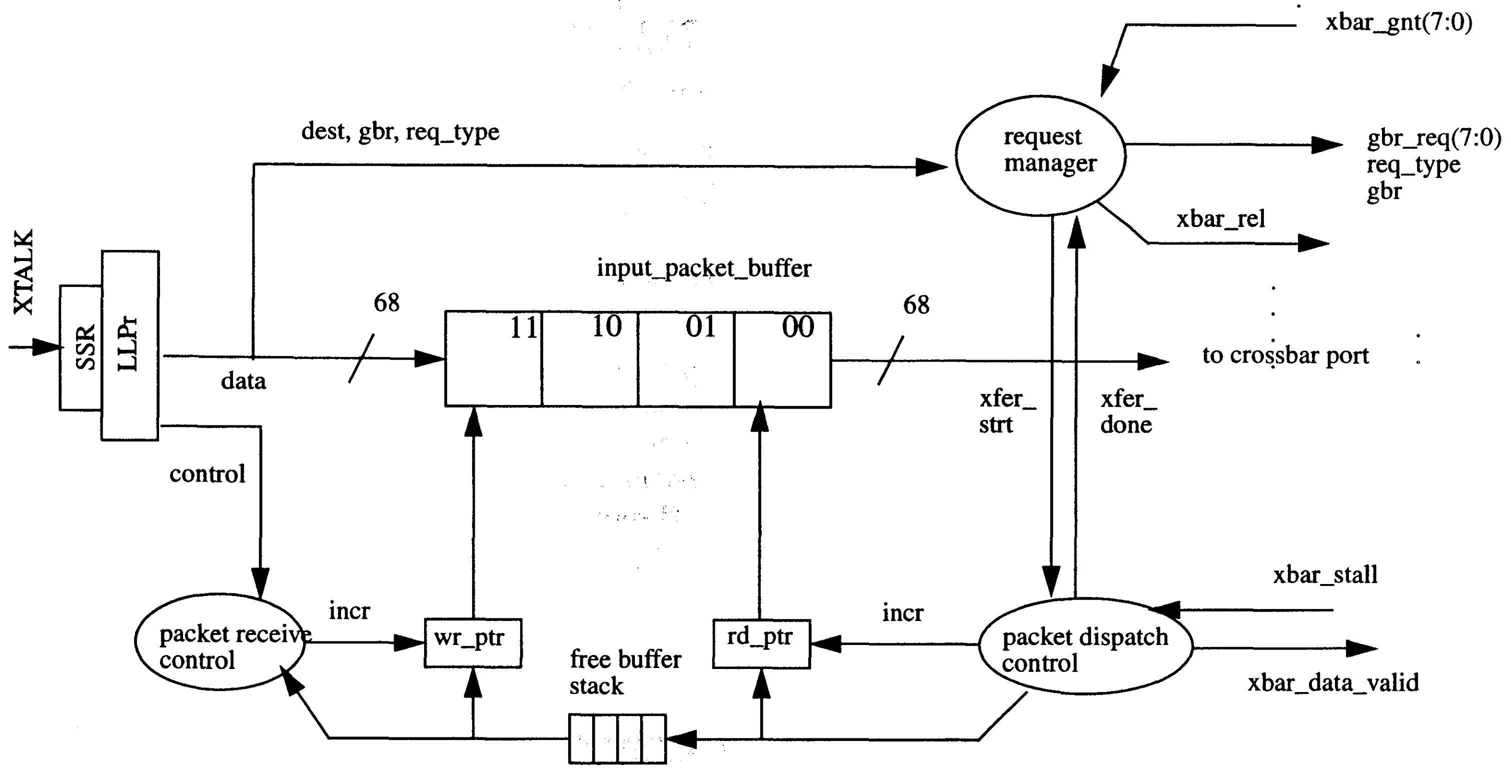


Figure 7 Source Link Controller

5.3.2 Source Synchronous Receiver and Link Level Protocol Receiver

For the purpose of this document the LLPr and SSR will essentially be treated as a black box to which the rest of the Source Link Controller interfaces. The architecture of the LLPr, LLPt, SSR and SSD are covered in great detail in the **Crosstalk Interconnect Specification** and the **Source Synchronous Driver/Receiver Specification.**

The interface between the SSR and the outside world consists of the following pins: **16 bit link:**

Data[19:0]	(i) Data, check bit, and Sequence Number (SN) information
DataReady_N	(i) Valid data is being transmitted, active low
Clk[1:0]	(i) Differential clock
8 bit link:	
Data[9:0]	(i) Data, check bits, and SN information
DataReady_N	(i) Valid data is being transmitted, active low
Clk[1:0]	(i) Differential clock

The SSR operates in the clock regime of the source synchronous differential clock which it receives(clk[1:0]). Between the LLPr and the SSR is an asynchronous boundary crossing between the received clock and the 100 Mhz core_clock. All SSR outputs interface directly with the LLPr.

The interface between the rest of the Source Link Controller and the LLPr consists of the following signals:

RcvData[63:0]	(o) Data from remote transmitter.
RcvSideband[7:0]	(o) Sideband data from remote transmitter.
RcvDataValid	(o) Data lines contain valid data.
SquashData	(o) Last micro-packet must be discarded.
RcvCBError	(o) Last micro-packet encountered check bit error.
RcvSNError	(o) Last micro-packet encountered sequence number error (only if no RcvCBError detected.)
RcvLinkReset	(o) Asserted when local receiver is in link reset state.
RcvWarmReset	(o) Asserted when local receiver is in reset state due to a remote or local WarmReset.

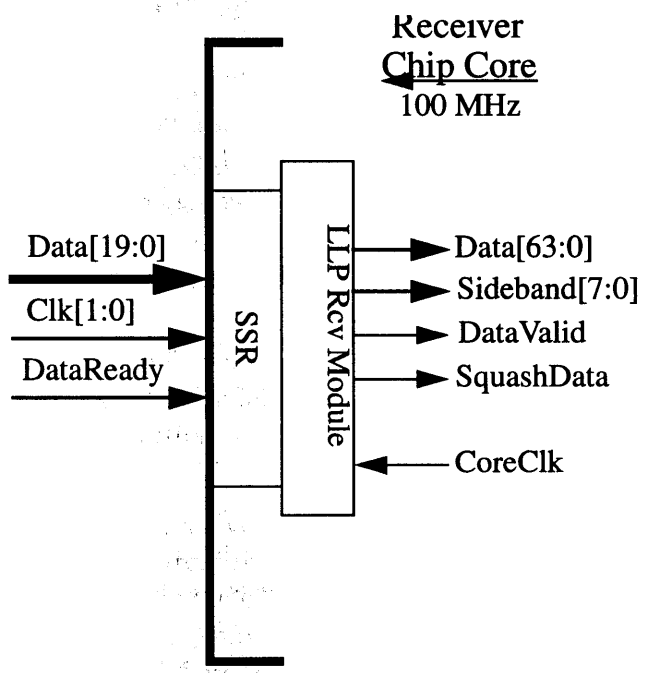


Figure 8 SSR, LLPr Pair

The reset signals from the block, namely, RcvLinkReset and RcvWarm-Reset, are interpreted by the Link Controller's exception block. The RcvCBError and RcvSNError outputs are also sent to the exception block.

The RcvData from the LLPr is received and stored in the input packet buffer until it can be forwarded through the crossbar. Command word portions of a Crosstalk packets are forwarded to the crossbar request queue. The forwarded information is used to properly route the packet. Portions of the sideband are also placed in the input packet buffer, whereas other portions of the sideband are forwarded to the destination link controller.

The packet receive control block is responsible for interpreting the control signals from the LLP and writing the input packet buffer. It also handles updating the crossbar request queue. The signals which the packet receive control block receives include RcvDataValid, SquashData, BitMode8, MicropacketHead and portions of the Sideband data.

Please refer to Figure 8 for examples of the data transfer cycle timings. In the diagrams, D0 refers to the first portion of a micropacket and D1 refers to the second portion. The CRC check code is computed for the entire micropacket. If the micropacket is found to be in error, SquashData is asserted. At this point the entire micropacket(D0 and D1) must be discarded.

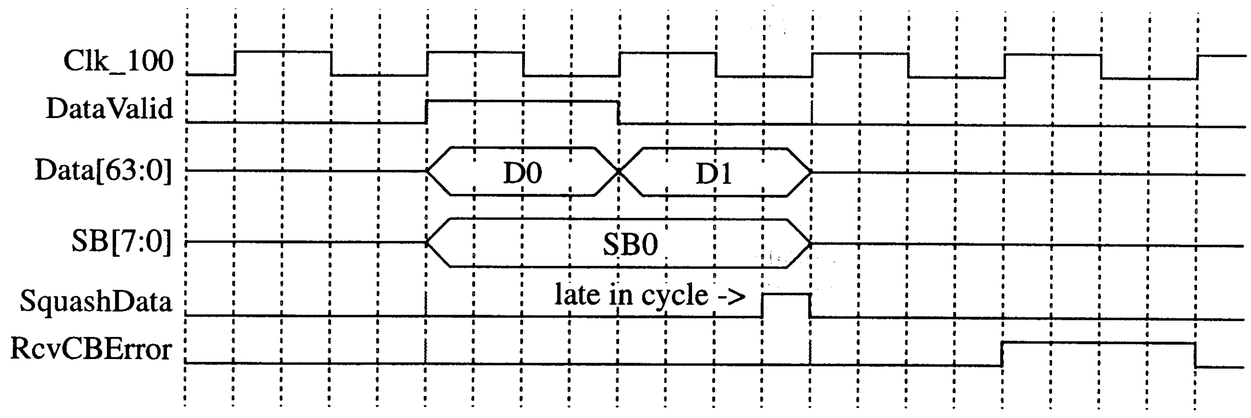


Figure 9 LLPr interface 16 bit mode, with packet error

5.3.3 Input Packet Buffer

The input packet buffer consists of an embedded SRAM 68 bits wide by 72 locations deep. The buffer is subdivided into 4 separate sub-buffers, each capable of storing a single crosstalk packet up to a full cache line in length. To simplify the design of the buffer management, each sub-buffer can store only one crosstalk packet.

The input packet buffer stores 64 bits of data from the LLP and 3 bits of sideband data (start, stop and valid). Data is written into the input packet buffer by the packet receive control, and the sub-buffers are managed by the free buffer stack. The packet dispatch module reads data out of the input packet buffer.

The exact implementation of this array is dependent on the speed and size of the selected vendor's array. The array must be capable of 10 ns read and write cycle times.

5.3.4 Packet Receive Control

The packet receive control module monitors sideband and control signals from the LLP and writes data to the input packet buffer.

A packet is delimited by the sideband signals head and tail. These bit are fully defined in the crosstalk Interconnect Specification. The packet receive control uses these signals, along with a data_valid signal from the LLP, to determine when to write to the input packet buffer. Data is written into the input packet buffer directly as soon as it is received from the LLP.

When the LLP detects a CRC error, it generates a signal called squash, which appears during the last half of a micropacket. By the time the packet receive control module detects a squash, it will have already written the micropacket. The packet receive control must maintain a shadow write pointer so that when a squash is encountered, the write pointer can be backed up to the beginning of the bad micropacket. This micropacket will be overwritten by the next micropacket.

The packet receive control also controls packet length counters. As full crosstalk packets are correctly received (no squash is detected during the stop bit), the packet receive control generates an increment signal to a counter associated with this packet buffer entry.

There is one significant exception when an administrative packet is received. An administrative packet is a Crosstalk packet which has only valid sideband information. The packet receive control logic should detect such packets and inhibit any writes to the input packet buffer.

After a crosstalk packet is correctly received, the packet receive control updates the write pointer to point to the address of the next available subbuffer in the input packet buffer.

Other miscellaneous functions provided by the packet receive control module include alerting error handling logic when errors occur. The errors detected include micropacket error detected and over allocated packet buffer. Additionally, it passes valid sideband information such as the freebuf signal onto the crossbar connection arbiter. The meaning of freebuf will be covered in the section related to the packet dispatch control.

5.3.5 Free Buffer FIFO

The free buffer fifo maintains the locations of the available cache line buffers in the input packet buffer.

As soon as the chip comes out of reset, the free buffer stack pops off the first available location in the buffer and loads the write pointer to point to that location. As complete crosstalk packets are received, the free buffer stack pops off the next location and the packet receive control loads the write pointer. As packets are read out by the packet dispatch block, free locations are pushed back onto the stack.

The free buffer stack also generates a full signal when all its input buffers are in use.

5.3.6 Crossbar Request Manager

The request manager determines priorities of requests and issues them. Requests can be generated in accelerated, scheduled or normal mode depending on the fullness of the input packet buffer and the speed of the source and destination.

The first portion of a packet contains the packet routing and priority information. As the destination is decoded, so is the priority. The decoded destination is then routed to the correct priority request generator (RR or GBR).

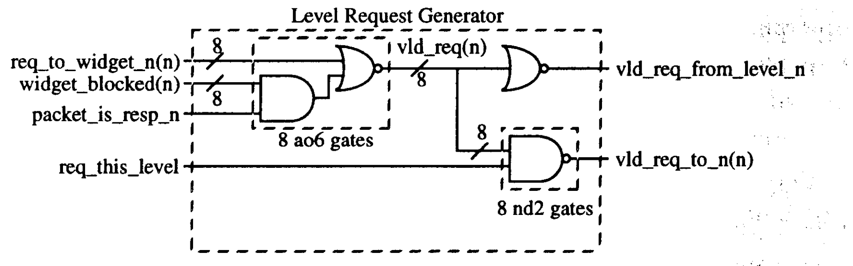
In accelerated mode, the decoded destination is passed directly to the request manager as soon as the first half of the first micropacket is received from the LLP. A request is generated the next clock cycle, at which time the second half of the micropacket and the squash signal are available. If there is no error, the request is passed through a mux and latched. If squash is asserted, a null request is muxed in and the request in the request manager is purged. Accelerated mode saves 2 clock cycles over normal mode by not waiting for the squash signal to determine if a request will be pushed into the request manager. Accelerated mode will be used in the cases summarized in Figure 8.

In normal mode, the request is held until the first micropacket has been received correctly. If a squash signal is not detected during reception of the second half of the first micropacket, the request is pushed into the request manager.

A third mode is used for transfers from an 8 bit port to a 16 bit port when a full cache line is being transferred. In scheduled mode, the request is held until half the cache-line is received. This allows the source link to build up enough data so as to not slow down the destination link to 8 bit speed. (The destination link would be waiting for the 8 bit source link to receive data.)

SRC	DEST	INPUT BUFFER EMPTY	Scheduler action
16	16	Y	Select accelerated xbar request
16	16	N	Select normal xbar request
16	8	Y	Select accelerated xbar request
16	8	N	Select normal xbar request
8	8	Y	Select accelerated xbar request
8	8	N	Select normal xbar request
8	16	N	Packet size in micropackets: 1,2,3 Select normal xbar request 9 Select scheduled request
8	16	Y	Packet size in micropackets: 1 Select accelerated xbar request 2 Select normal xbar request 3 Select normal xbar request 9 Select scheduled request

Figure 10 XBAR Request Dispatch



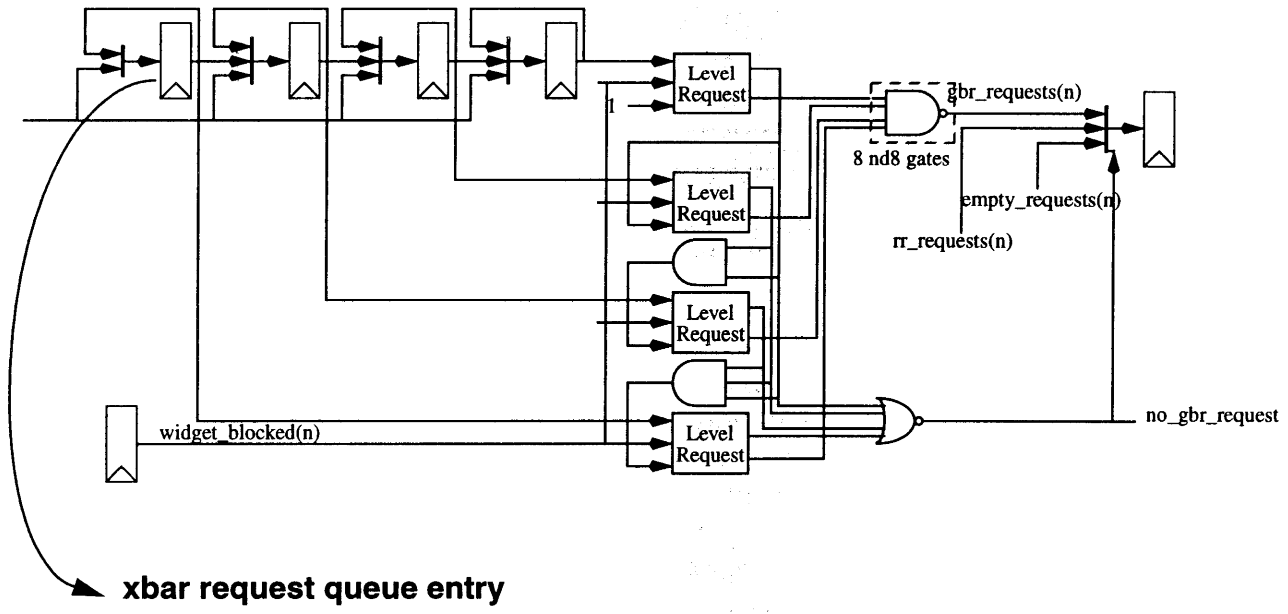


Figure 11 Request Manager Detail

Each priority request generator consists of a fifo-like array of registers. The head of the fifo is the newest request and the tail is the oldest. A request is generated based on the following criteria:

1. If a packet's destination is blocked, meaning it can accept no more request packets, only response packets destined for that destination may be sent. A request packet cannot be selected until its destination can accept more request packets.

2. The oldest packet in a request generator is promoted ahead of younger packets.

A destination is blocked if it has received the maximum number of requests or if the source has depleted its bandwidth to the target. Counters tracking maximum request and bandwidth available are maintained in the destination port arbiter. The arbiter sends a block signal to all source ports indicating if that source is blocked from sending to this destination.

Each request generator (RR and GBR) independently determines its next request. By taking into account whether the destination is blocked and if the packet is a response or request, the request generators determine the winning request in one clock cycle. A mux at a higher level selects between GBR and RR requests. If there are GBR requests pending, then GBR requests will get priority over RR requests. Please reference Figure 8.

The request manager continues to assert the request until status conditions change causing another packet in the input packet buffer to assume higher priority or the crossbar connection request is granted.

Once the packet has been selected for crossbar connection arbitration, it will alert the packet dispatch control which will take appropriate action to prepare for transferring the packet out of the input packet buffer and into the crossbar.

When the request manager receives a grant from the crossbar connection arbiter, it will clear the packet's valid bit in the crossbar request queue, thus removing it from arbitration consideration. When the packet dispatch control is nearing completion of the current transfer, it will signal the request manager and a new crossbar connection request will be dispatched if one is ready. In this manner, crossbar connection arbitration for the following packet may overlap the transfer of the current packet.

As mentioned before, the age of the packet plays a part in the request manager's algorithm. The reason for this is that there is a requirement that request packet issue order be maintained. If this were not the case and two request packets of the same priority were sent to the same destination but dispatched in an order different from the order in which they were received, a Write After Read hazard could occur.

This problem is circumvented by always selecting the oldest packet. Using a fifo structure to generate requests, the highest priority is given to the request at the head of the fifo.

In the event that a change in status occurs among the entries in the input packet buffer and a crosstalk packet assumes a higher priority than the one currently being selected for crossbar arbitration, the request manager will perform a request context switch. One example of when this would occur would be when the input packet buffer contains only remainder ring priority packets and then a guaranteed bandwidth packet is received. When a context switch occurs, the request manager will withdraw all crossbar connection requests. It will wait for a period of time equal to the request to grant latency. If a grant is received during the waiting period, the lower priority packet which is associated with the previous request will be sent. If no grant is received, a new request will be issued to the destination associated with the higher priority packet.

Size	Bit name	Signal Definitions
1	**Pt**	***Packet timer flag***: when a packet timer interrupt occurs and this entry is valid, this bit will be set. When the xbar request associated with this entry is acknowledged, this bit will be cleared.
1	**Dto**	***Destination time-out flag***: when a packet timer interrupt occurs and the packet timer flag is set, this bit will be set. This bit is cleared when the xbar request associated with this entry is acknowledged or by the exception processing logic.
1	**Valid**	***Valid:*** indicates a valid entry. When this bit is active, a crossbar connection arbitration cycle will begin.
1	**Flush**	***Input Buffer Flush*** indicates that this entry was present when a flush command was issued. When an input buffer flush command is issued, all current entries are marked. When a packet is forwarded through the crossbar, the entry is retired and this bit is reset. When all marked entries have been retired, the flush is completed.
1	**ReqType**	***Request type*** indicates whether the packet is a request or a response.
8	**Dest**	***Destination field*** contains the packet's target id field, used to indicate the proper link controller to send the crossbar connection request to.
1	**GBR**	***Guaranteed Bandwidth Ring*** indicates this is a high priority packet for purposes of crossbar arbitration. High priority =1, Normal priority = 0.
2	**BufLoc**	***Buffer location:*** indicates which input packet buffer location is associated with this entry.

Table 36 **Xbar Request Queue Entry**

5.3.7 Packet Dispatch Control

The packet dispatch control module is responsible for transferring crosstalk packets out of the input packet buffer and into the crossbar router. Once a crossbar connection request is issued, the request manager will forward a pointer which indicates which input packet buffer the data should be transferred from. This pointer is used to determine the starting address of the data in the input packet buffer and indicates which input packet buffer status register should be accessed.

The packet dispatch control module will, upon receiving this pointer, pull the first value out of the input packet buffer and load it into the crossbar input port. This reduces the start up latency when a crossbar grant is received. The packet dispatch control then waits until a crossbar grant is received. At this time it will also assert the xbar_data_valid signal and decrement the packet count value in the input packet buffer status registers.

When the crossbar grant is received, the module will begin reading from the input packet buffer and transfer the packet into the crossbar. As each piece is transferred, the packet count is decremented. The presence of a complete packet in the input packet buffer is indicated by the presence of the tail flag in the buffer's associated input packet buffer status register. When the tail flag is present and the count has reached zero, the complete packet has been transferred.

When the end of the transfer is near, the module will issue a crossbar release signal. This will allow the request manager to schedule the next crossbar arbitration request. It will also allow the crossbar connection arbiter to give the port away before the current transfer is complete.

In conjunction with releasing the crossbar connection arbiter, the packet dispatch control will make the location in the input packet buffer available. It will do this by advancing the head of the fifo in the circular buffer. Additionally, it will send a signal to the transmit side of the link controller (Destination Link Controller) to issue a freebuf signal on the outgoing sideband data.

The widget must have prior knowledge of the number of crosstalk packet buffers in the crossbow's input packet buffer. When a packet is dispatched from the widget to the Crossbow Subsection of XBridge the counter corresponding to the number of available sub-buffers is decremented. When the widget receives a Crosstalk credit in the sideband portion of an LLP micropacket from the XBridge, this indicates a new sub-buffer has been made available and the counter is incremented. In this manner, packet flow control is achieved between the widget and the crossbow input packet buffer.

The four separate counters in the input packet buffer status registers allow the Crossbow Subsection of XBridge to simultaneously stream data into and out of a packet buffer location. If a source time-out occurs, the tail flag will never get set and instead the source time-out error flag will be set. If, in the course of a transfer, the dispatch controller detects this error flag is set, it should continue to transfer data until it has transferred all existing data and then "pad" the packet with the last piece of data received until the required number of transfers has occurred. The invalid sideband bit should be set for all padded data transferred. The correct number of transfers to perform is indicated by the expected packet size value which is determined from the packet header information and stored in the input packet buffer status register.

If the dispatch controller receives an abort signal it indicates an error has occurred.When this occurs, the controller should simply retire the input packet buffer location on in the input fifo.

There is a possibility that the destination may not have enough available buffers to store the packet in progress. In this situation, the destination will assert a stall signal. The dispatch controller must then wait for a micropacket boundary before it stops sending the packet. When the stall signal is de-asserted, the dispatch controller resumes transmission of the packet.

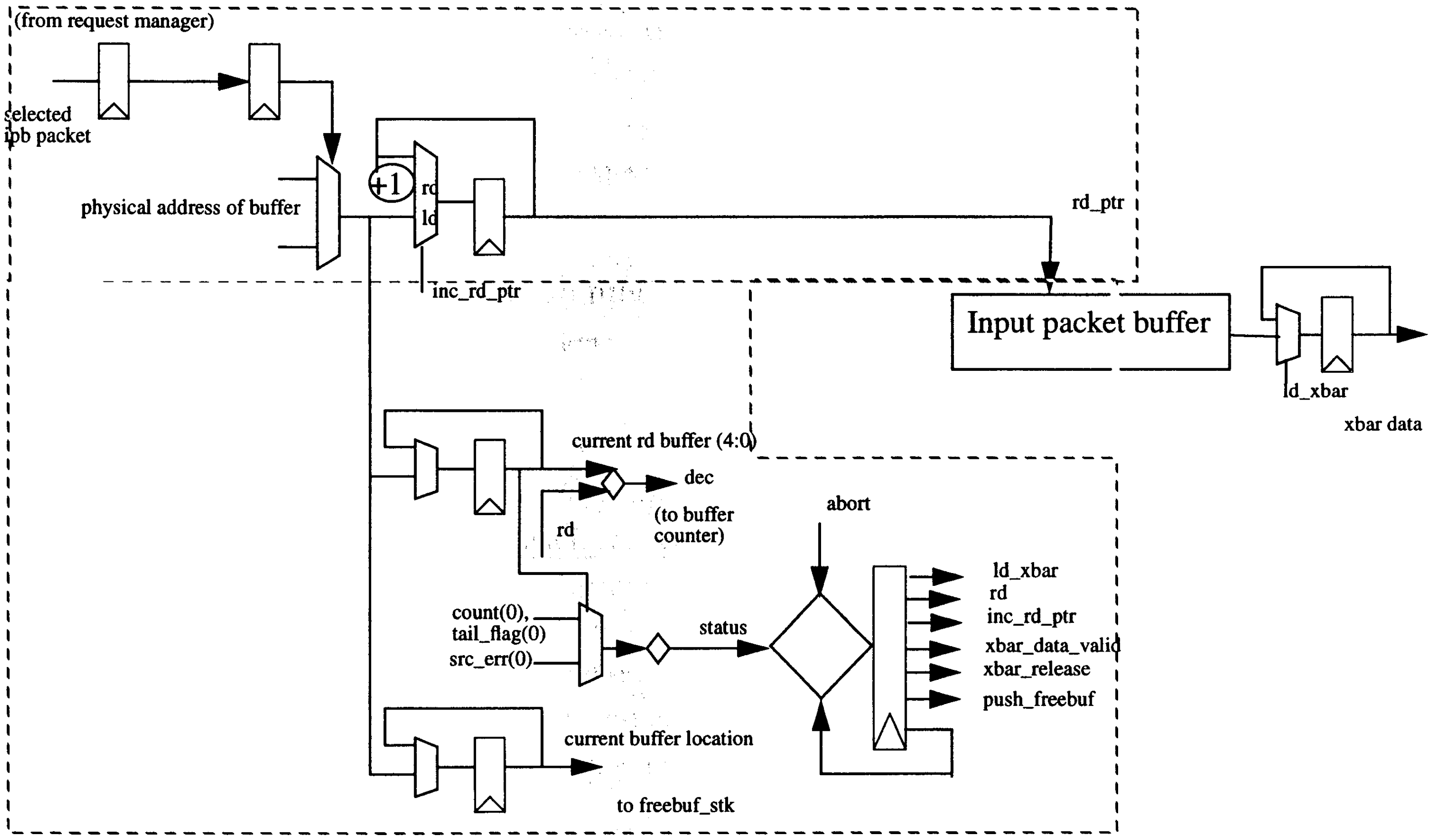


Figure 12 Packet Dispatch Control

5.3.8 Exception module/Error handling

The exception module is responsible for overseeing all error conditions which may occur on the link. The detection processing of error conditions is somewhat distributed throughout the module in the link controller and their interaction is discussed in this section also. These error conditions include:

- Data Error occurred, receiver
- Data Error occurred, transmitter
- Packet time-out destination
- Packet time-out source
- Illegal destination
- Retry time-out

- Over-allocated input packet buffer
- Under-allocated bandwidth
- Link Reset

A receiver data error occurs when the packet receive control logic detects a corrupted micropacket that is being received by the LLP (SquashData active). In this case, the packet receive control module will signal the exception module. The exception will update the retry counter in the Link Status register. If the Interrupt on LLP Receiver Error bit is set in the Link(x) Control register, the exception module will send an interrupt request signal to widget 0 each time a receiver error occurs. If the Interrupt on LLP Receiver Error Counter Overflow bit is set, an interrupt request will be generated whenever the counter overflows. When widget 0 receives this signal, it will issue an interrupt packet to the host if there are none currently outstanding.

Transmitter data errors which are indicated by the LLPt MicroPktRetry signal are similarly reported and logged.

Destination time-out errors occur when a packet has been received in the input packet buffer, but has not been dispatched to its destination in a reasonable period of time. The way this is determined is by means of time-out clock. The time-out clock is a central timer in widget 0 which sends a timer interrupt to all the link controllers at a programmable interval.When the time-out clock "interrupt" occurs, all valid entries in the request manager's xbar request queue are marked by having their packet timer flag set. As the xbar requests are honored, the entries are cleared out of the request queue. If another time-out interrupt occurs and there are crossbar request queue entries with their packet time-out flags set, these packets are considered timed-out. At this point, the exception module will detect this condition and pass a halt request to the request manager.

The request manager will finish its current transaction, halt, and send an acknowledgment back to the exception module. The exception module will then send a signal to the request manager which will cause all timed out packets in the queue to have their error bit set and their destination redirected to widget 0. The exception module will then release the request manager.

The request manager will then go about normal processing. When it arrives at the entry for a timed-out packet and makes the xbar port request to widget 0, the error request bit will be set. If the global error registers are available in widget 0, the xbar port request is sent to widget 0. Widget 0 will acknowledge the request. The packet dispatch controller will transmit the packet through the crossbar, retire the buffer location onto the freebuf stack, and signal the exception module. The exception module will update the Link(x) Status register. Widget 0, upon receiving the packet, will update the global error registers. Once the registers have been written, the global_err_reg_avail flag is de-asserted.

As subsequent time-out packets are processed only if the global_err_reg_avail flag is de-asserted. The exception module will intervene and suppress the crossbar port request from being sent to widget 0 if a time-out occurs. It will also "fake" a xbar port grant to the request manager such that the entry is removed from the queue. The packet dispatch controller when it detects that the error registers are not available, it will simply retire the packet and alert the exception module.

A source time-out occurs when a packet has been partially received, and is being forwarded to the destination. Partial transmission will occur when the LLP goes into retry. When the head of a packet is received the packet receive control module will note this. If the packet receive control logic detects two time-out clock interrupts before the tail of the packet is received, then the packet will be considered timed-out. The packet receive control will set the packet time-out bit in the packet count registers and prepare to accept the start of the next crosstalk packet. When the dispatch control logic detects that the time-out bit is set, once it has dispatched all available packet data it will pad the rest of the packet and set the invalid sideband bit in each piece of data until the correct number of transfers has occurred. It will then set the **Packet Time-out Source** bit in the Link(x) Status register which will cause an interrupt request to be generated if the interrupt is unmasked. Please reference for a flow diagram hardware handling of source and destination time-outs.

A destination time-out occurs when a connection between a source and destination has been established but, the packet transfer does not complete before two time-out clock interrupts have occurred. This situation can occur when a destination link stalls indefinitely due to multiple link errors or a MaxRetry situation occurring. When this occurs, the **Packet Time-out Destination** bit in the Link(x) Status register will be set and an interrupt request will be generated if the interrupt is unmasked. Additionally, the location of the destination which timed-out will be stored in the **Timed_Out Destination** field of the Link(x) Auxiliary Status register. The packet's location in the input packet buffer will then be retired.

An illegal destination error occurs when a packet is received and its destination field does not correspond to any crossbow port destinations. This condition is detected, by the destination mapping logic. When an illegal destination is detected the destination mapping logic will map the destination to widget 0 and set the destination error bit. This information will then be entered into the request manager's xbar request queue. From this point, the packet is handled the same as a destination time-out packet with the notable exception that the **Illegal Destination** bit is set in the Link(x) Status register.

A retry time-out occurs when the LLPt has tried to send a packet for the programmed maximum number of retry times. When this occurs, the **RetryTimeout** signal on the LLPt will become active. This will set the ***LLP Max Transmitter Retry*** bit in the Link(x) Status register and create a interrupt request that is sent to widget 0.

An over-allocated input buffer occurs when a packet arrives at the input and no entries are available in the freebuf fifo. When this condition arises, the packet receive control will let the packet fall on the floor and set the ***Input Overrallocation Error*** bit in the Link(x) Status register. This will send an interrupt request to widget provided the ***Enable Interrupt on Overallocated Input Buffer*** bit is sent in the Link(x) Control register.

An under-allocated bandwidth error occurs when a GBR request comes in and its associated bandwidth counter indicates it has used all of its allocated transfers for the current guaranteed bandwidth interval. The crossbar connection arbiter will detect that this condition has occurred and set the ***Bandwidth Allocation Error*** bit in the Link(x) Status register. In addition, it will indicate which source the request(s) came from. The contents of this location will not change, regardless of further under-allocation occurring until the error condition is cleared by a read of the Link(x) Status register at its read and clear address.

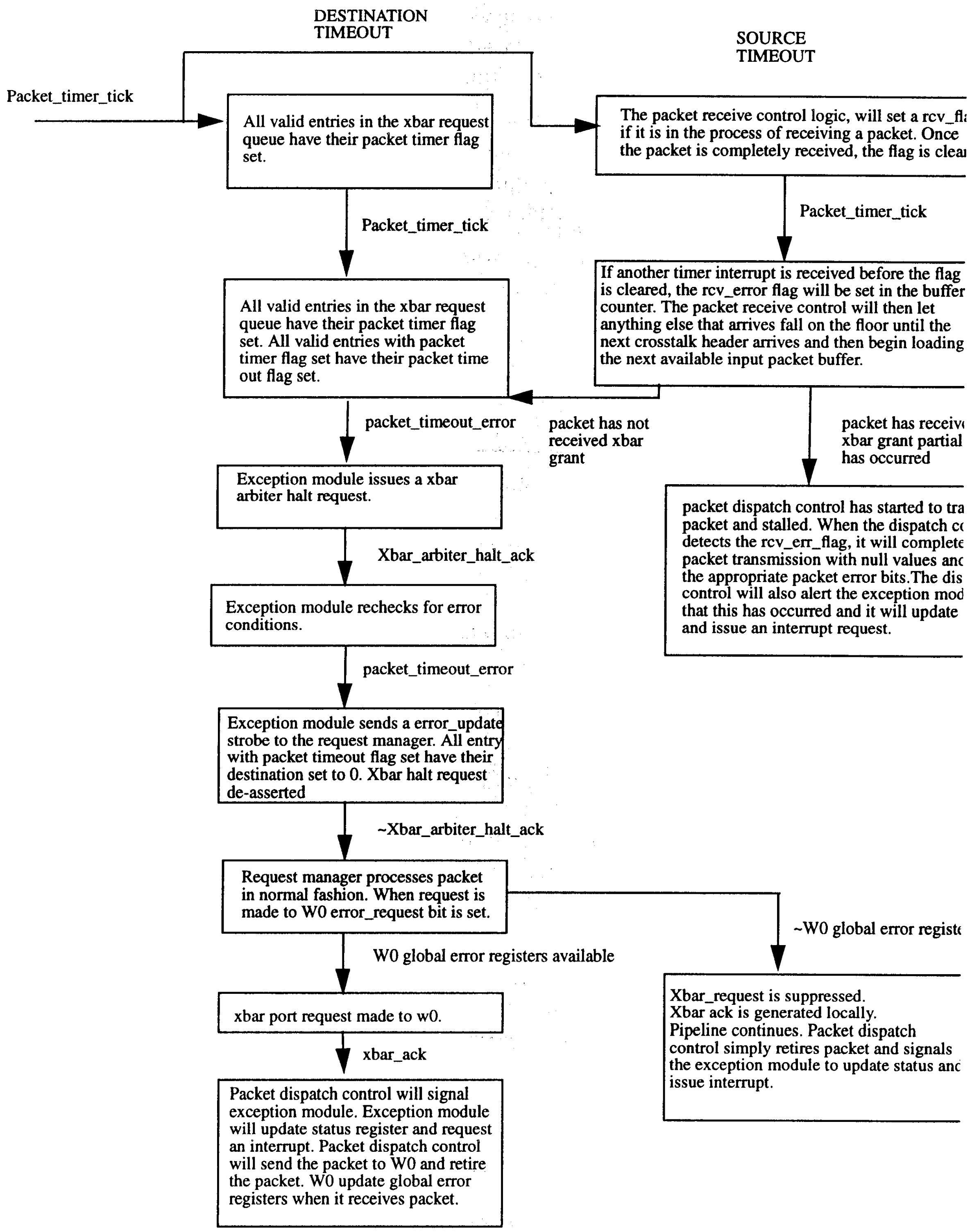

DESTINATION TIMEOUT
SOURCE TIMEOUT
Packet_timer_tick
All valid entries in the xbar request queue have their packet timer flag set.
The packet receive control logic, will set a rcv_fl
if it is in the process of receiving a packet. Once the packet is completely received, the flag is clea
Packet_timer_tick
Packet_timer_tick
All valid entries in the xbar request queue have their packet timer flag set. All valid entries with packet timer flag set have their packet time out flag set.
If another timer interrupt is received before the flag is cleared, the rcv_error flag will be set in the buffer counter. The packet receive control will then let anything else that arrives fall on the floor until the next crosstalk header arrives and then begin loading the next available input packet buffer.
packet_timeout_error
packet has not received xbar grant
packet has receiv xbar grant partial has occurred
Exception module issues a xbar arbiter halt request.
packet dispatch control has started to tra packet and stalled. When the dispatch c detects the rcv_err_flag, it will complete packet transmission with null values anc the appropriate packet error bits.The dis control will also alert the exception mod that this has occurred and it will update and issue an interrupt request.
Xbar_arbiter_halt_ack
Exception module rechecks for error conditions.
packet_timeout_error
Exception module sends a error_update strobe to the request manager. All entry with packet timeout flag set have their destination set to 0. Xbar halt request de-asserted
~Xbar_arbiter_halt_ack
Request manager processes packet in normal fashion. When request is made to W0 error_request bit is set.
~W0 global error regist
W0 global error registers available
xbar port request made to w0.
Xbar_request is suppressed. Xbar ack is generated locally. Pipeline continues. Packet dispatch control simply retires packet and signals the exception module to update status anc issue interrupt.
xbar_ack
Packet dispatch control will signal exception module. Exception module will update status register and request an interrupt. Packet dispatch control will send the packet to W0 and retire the packet. W0 update global error registers when it receives packet.

Figure 13 Flow diagram packet time-out processing

5.3.9 Destination Link Controller

The destination link controller consist of the following submodules:

- Crossbar Connection Arbiter
- Freebuf Issue Logic and Crossbar Flow Control
- Sideband Datapath
- Link Level Protocol Transmitter. LLPt
- Source Synchronous Driver, (SSD)

5.3.10 Crossbar Connection Arbitration

Each destination link controller contains a crossbar connection arbiter. The crossbar connection arbiter controls access to the destination crossbar port associated with its link controller. It receives requests from the request managers of all source link controllers that want to establish a path through the crossbar to transfer a packet to the destination link. The crossbar connection arbiter will arbitrate from among the outstanding requests, grant the connection, and establish the connection through the crossbar.

A crossbar arbitration cycle begins when one or more Crossbar connection requests are received from the source link controller request manager of the other ports. The crossbar connection arbiter will send a grant acknowledge signal back to the winning link controller and switch the mux control signals to the crossbar to enable the path from the source link controller to its destination crossbar port. The path is maintained until the source link controller relinquishes the connection by sending a crossbar release signal to the crossbar arbiter. At this time, the arbiter will service other requests if there are any outstanding. If there are no requests outstanding, the crossbar arbiter will put the datapath in idle mode.

The crossbar arbitration scheme consists of two round robin ring arbiters. From each request manager, the crossbar arbiter receives a request signal which indicates the priority of the crossbar connection request. The highest priority requests are guaranteed bandwidth requests. All other requests are remainder ring requests. Guaranteed bandwidth requests are serviced first in round robin order starting at one greater than the last link controller which won arbitration. If there are no guaranteed bandwidth requests, remainder ring requests are then serviced. Please refer to Figure 14 and Figure 16.

The crossbar connection arbiter maintains status signals which are used by a source link controller's request manager to determine in which order to send the packets present in its input packet buffer. One bit that is maintained by the crossbar arbiter is the MAX_REQ bit. As previously mentioned in the request management section, this bit indicates that the target widget has received its maximum number of outstanding crosstalk request packets. This maximum number is equal to the number of request packet buffers available in the target widget. When a crossbar connection request is made, the source link controller sends a signal indicating whether the connection request is for a response or request packet transfer. If the transfer is to be a request packet transfer, when the request is granted a counter is incremented. When the count value is equal to the programmed maximum value, the MAX_REQ bit is asserted. When this flag is set, source link controllers will only send response packets to this target. The target widget will indicate when more request packet buffer space is available by sending the freebuf signal in the sideband data bits, which will eventually decrement the request packet counter and change the MAX_REQ bit.

The crossbar connection arbiter also maintains 8 status bits which indicate the status of each source link controller's guaranteed bandwidth counter. In each crossbar connection arbiter there is a counter associated with each source link controller. This counter contains the number of packet transfers from an initiator widget to this target widget that will be treated as guaranteed bandwidth requests during a programmable time period. All guaranteed bandwidth counters associated with all crossbar connection arbiters are reloaded with their initial values at a fixed interval which is determined by the value in the arbitration reload register. Each time a guaranteed bandwidth ring request packet is granted a connection, the value of the guaranteed bandwidth counter associated with the packet source is decremented. Response packet transfers have no effect on the counters. If the value of a guaranteed bandwidth counter reaches zero before the reload interval, the BW_aval bit associated with the counter is negated. This bit remains negated until the counter is reloaded. As guaranteed bandwidth requests are received at a source link controller, if the BW_aval bit for the source- destination pair is negated, the request will be passed over to the remainder ring.

Widgets which participate on the remainder ring share any remaining bandwidth not used by guaranteed bandwidth requestors during the arbitration reload time interval. When the crossbar connection arbiter receives an arbitration request from a remainder ring widget it is serviced in a modified round robin fashion with remainder ring requests from other links. When a remainder ring widget's packet wins arbitration, it will continue to win subsequent arbitration requests provided it is not preempted by a gbr arbitration request, it is continuing to make arbitration requests for subsequent packet transfers, and it has not exceeded its remainder weight value. The remainder weight value is essentially a burst count which allows a widget to transfer n packets, where n is equal to the remainder weight value, before it has to give up the destination port to another remainder ring requestor.

The implementation of this ring is similar to the guaranteed bandwidth ring. When a member of the ring wins arbitration, a counter is incremented. The value of counter is compared to the remainder weight value associated with its position on the ring. The ring priority will remain at this position until one of two conditions occur. If the counter value reaches the remainder weight value, the counter is cleared and the "priority token" is passed to the next widget on the ring. Also, if another remainder ring widget wins arbitration, the priority token is passed to that widget and the counter is set to one. The second condition would occur when the remainder ring widget which previously won does not re-request for a connection.

Please note that while a gbr packet may interrupt a remainder ring burst, the remainder ring device does not give up its priority on the remainder ring. After all gbr requests have been satisfied, the remainder ring burst will complete.

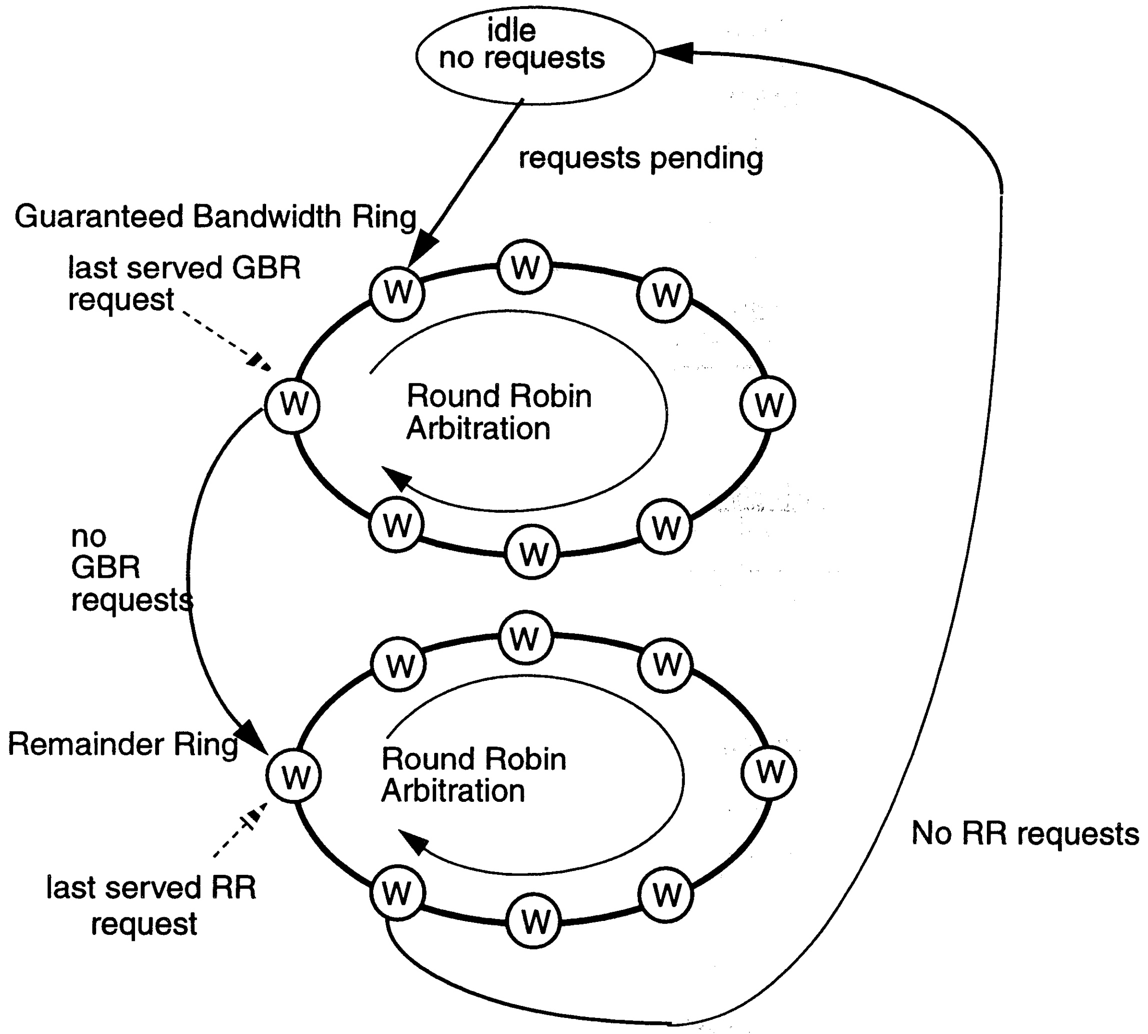


Figure 14 Crossbar Connection Arbitration Diagram

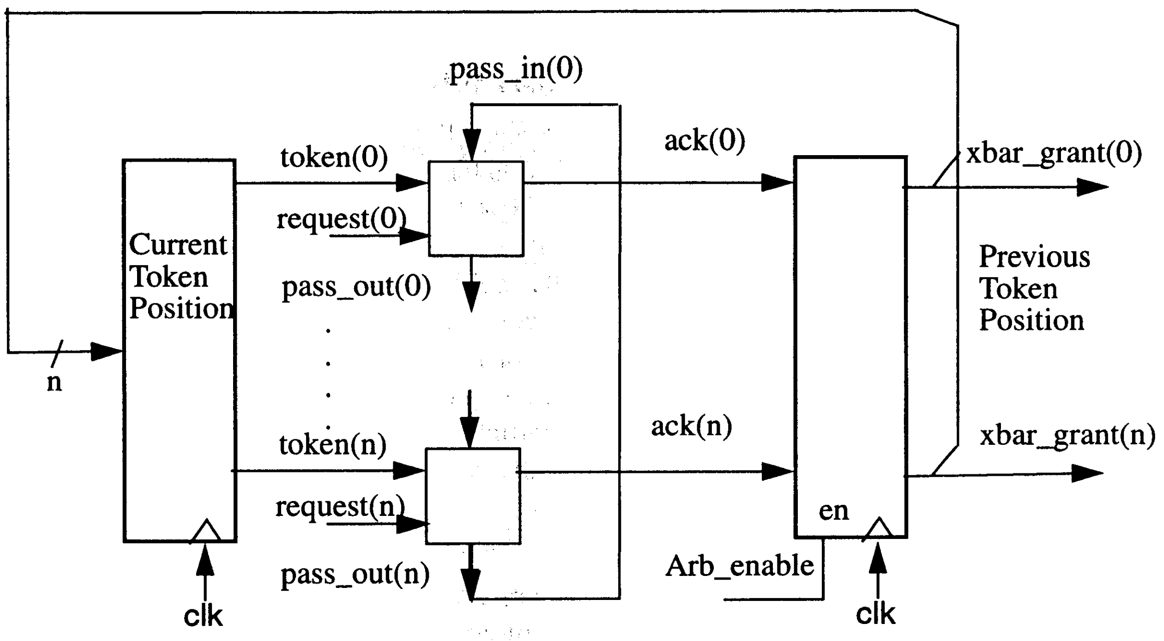


Figure 15 General Ring Arbitration Implementation

5.3.11 Free buffer issue logic and Crossbar Flow Control

As crosstalk packets are dispatched throughout the crossbow, locations in the input packet buffer become available. The Crossbow Subsection of XBridge will signal the widget which sent a packet that there is space available for another packet to be transferred into the Crossbow Subsection of XBridge. This is accomplished by setting a bit in the micropacket sideband data flowing back to the initiating widget. If there is crosstalk packet information flowing back to the initiating widget, the credit is simply sent along with the normal stream of data. If there is no traffic on the port, the free buffer issue logic will simply generate a micropacket with a credit and a sideband administrative bit set. When the widget receives the micropacket, it will simply discard the micropacket data and use the credit information in the sideband data.

As packets are dispatched across the crossbar, they are received and passed directly to the LLPt. In the LLPt, there is a retry buffer which serves a dual purpose. For normal LLPt functionality, it functions as a retry buffer. As the LLPt transmits micropackets it saves them in the buffer until the receiver acknowledges their receipt. If the acknowledgment does not come in a given time period, the packets are retransmitted.

During chip to chip communication, normal turn-around to transmit a packet and receive an acknowledgment uses only a portion of the retry buffer. The remaining buffer location are used as a rating-matching buffer to match transfer rates between the 16 bit sources and 8 bit destinations.

During a normal 8 bit transfer there is sufficient space left over in the retry buffer to contain a full cache line packet. This means a 16 bit source transferring to various destinations is not limited by the transfer rate of the destination. It simply dumps the cache line at full 16 bit rate to the 8 bit port and then goes to service another destination.

In the event the acknowledgment is not received and the buffer fills up, a stall signal is sent out which is monitored by the transmitting source. The current skid distance through the crossbar is 2 micropackets. The LLPt provides a count indicating the amount of free space remaining. When this number reaches 2 micropackets, the stall signal is asserted.

5.3.12 Link Level Protocol Transmitter and Source Synchronous Driver

For the purpose of this document the LLPt and SSD will essentially be treated as a black box to which the rest of the Destination Link Controller interfaces. The architecture of the LLPr, LLPt, SSR and SSD are covered in great detail in (reiterate the specific pertinent documents).

The interface between the SSD and the outside world consists of the following pins: (16 bit link)

Data[19:0]	(i) Data, check bit, and SN information
DataReady_N	(i) Valid data is being transmitted, active low
Clk[1:0]	(i) Differential clock

The interface to the LLPt consists of the following signals:

SendData[63:0]	(i) Data to be transmitted
SendSideband[7:0]	(i) Sideband data which accompanies each micropacket
SendDataValid	(i) Data lines contain valid data
BufFull	(o) DataValid must be de-asserted on the next clock, as the LLP transmit buffer is now full.
CancelDataValid	(i) Stomp current micro-packet and do not write it to xmit buffer. Available only in 16 bit mode, this signal

can be asserted during the second clock of a micro-packet.

SendDataError (i) Force bad checkbits on first transmit attempt. Retransmissions will not force errors. Asserted on first clock of micro-packet.

MicroPktRetry (o) Pulses each time the retransmit timer expires, indicating that data is being retransmitted due to lack of an acknowledge from the receiver. This signal, along with the receiver error signals, can be used to gather statistics on link integrity.

RetryTimeout (o) Asserted when a micro-packet has been retried MaxRetry times. Under this condition, the link will shut down all data transmission, de-assert BufFull, and wait until a LinkReset is received.

MaxRetry[9:0] (i) Total retry attempts a micro-packet can fail before RetryTimeout is asserted. This also controls the number of times a link will send reset micro-packet bursts without acknowledge before failing to reset.

MaxBurst[9:0] (i) Longest continuous data burst (in clocks) the LLP will send. Minimum value is 4. This value can be adjusted to account for slightly mismatched clocks.

BuffersEmpty[3:0] (o) Number of micropacket locations remaining in the Micropacket retry buffer. At time 0, this signal equals the usable LLP xmit buffer size. This signal is a registered output of the LLP, and is updated in response to new data from the core on the cycle following DataValid.

RequestBitMode8 (i) Indicates that the local physical port is 8 bits wide. Should be set to 0 for 16 bit ports.

BitMode8 (o) Port size currently being used by LLP. 1 indicates 8 bit, 0 indicates 16 bit.

Reset (i) Chip power on reset: resets all local LLP logic, begins LinkReset process.

SendLinkReset (i) Resets all local and remote LLP logic and asserts remote LinkReset signal. Forces mode arbitration.

SendWarmReset(i) Same as LinkReset, except causes remote WarmReset line to pulse instead of LinkReset line.

WarmResetAfterReset(i) This signal tells the LLP which type of reset-packet to send after the chip Reset signal de-asserts. If this signal is a '1' a WarmReset negotiation packet will be issued. If the signal is a '0',a LinkReset negotiation packet will be issued. The size request by the negotiation packet will reflect the existing RequestBitMode8 LLP input signal. In XBridge it is tied to a '0'.

NullTimeout[5:0] (i) Idle time (in clocks) required before a LLP null micro-packet is transmitted. Minimum value is 3.

CoreClk (i) 100 Mhz clock used by core logic.

The SendData signal comes from the rate matching buffer. SendSideband consists of data from the rate matching buffer and the sideband datapath. SendDataValid is provided by the rate matching buffer control. Please refer to Figure 16 for examples of the relationship of these signals

CancelDataValid is not used.

SendDataError is used for diagnostic purposes. It causes a micropacket, when initially sent, to be sent with the check bits corrupted. On retry transmission, the packet is sent correctly.

MicropktRetry and the RetryTimeout signal are sent to the exception module which handles these condition.

MaxRetry, MaxBurst, and Nulltimeout are all provided by values in the Crossbow LLP Control register.

5.4 Widget 0

Widget 0 occupies one crossbar port. All Crossbow internal register accesses are directed to this port. Widget 0 handles the writing of all internal registers and generation of read responses with register information. Additionally, packet transfers which have resulted in errors are directed to widget 0. Widget 0 then handles the updating of the global error registers and the issuing of interrupts, if necessary. Link controllers wishing to send a packet to widget 0 send arbitration requests as they would to obtain access to a link. Arbitration for the port is done in a simple round-robin fashion.

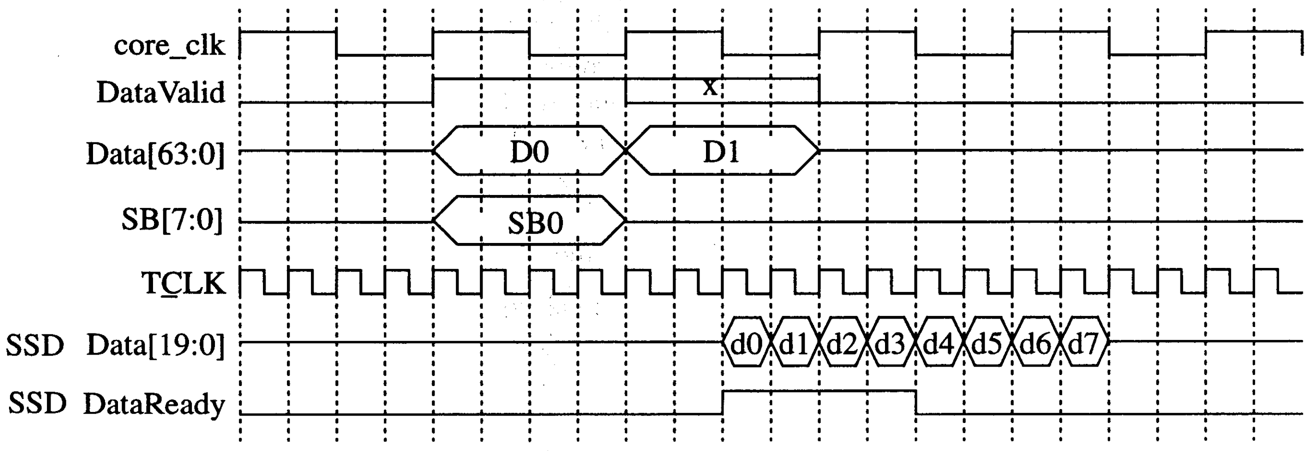


Figure 16 LLPt transmitting

5.4.1 Internal Register Access

All Crossbow internal registers are 32 bit. Therefore, all Crossbow registers accesses are assumed to be double word packets.

When a link controller wishes to transfer a packet to Widget 0, it arbitrates for the resource as in the case of a packet destined for any other link controller. A request is made to the crossbar connection controller, the controller acknowledges the requests in round robin order disregarding any priority flags and sends signals to the crossbar to establish the link between initiator and target.

When the packet arrives at Widget 0, the command portion of the packet is stripped and placed in registers and the address and request type are decoded. If the request is a write request, the data is stored in a temporary register and then written to the appropriate register when the address decode has completed. The access to remote link registers is accomplished via a separate 32 bit datapath which connects each link controller to widget 0 and widget 0 to the link controllers. Please refer to Figure 8. If a write response is requested, widget 0 will then arbitrate for the connection back to the initiating link. Once the connection is granted, the response packet will be formed using the information in the temporary registers and transferred. In the case of a read request, the pertinent request information is stored. Once address decode and register accesses are complete, the read data is placed in a temporary register. As before, arbitration for a connection to the initiating link is requested and the response packet is transferred.

Widget 0 typically processes only one transaction at a time. The one exception to this is a buffer flush read request which is done as a split transaction. When a flush read request is received, the necessary information to generate the read response is stored in a set of registers dedicated to this purpose. A flush request is sent to the link controller associated with the input packet buffer that is to be flushed. While the flush is occurring, other requests to widget 0 may be serviced. Once the flush has completed, the link controller will signal widget 0. Widget 0 will complete any transaction in process and then arbitrate and dispatch the response. Since the Crossbow Subsection of XBridge supports only one outstanding flush request at a time from all requesting widgets, it is assumed one set of temporary registers to store information for the read response is sufficient.

5.4.2 Error Processing

When an error occurs at a link controller, the erring packet is forwarded to widget 0. This typically occurs for packet time-out errors. When the W0 crossbar connection arbiter receives the port request, an additional signal will be set the error request bit which indicates that the information being transferred will be for an update of the global error registers namely the Crossbow Error Command Word, Crossbow Error Upper Address, and Crossbow Error Lower Address registers.

When the packet is received, the command and address portions of the packet are extracted and stored in the global error registers. The global_err_reg_avail bit will then be de-asserted. The link controllers will detect the de-assertion of the signal and respond accordingly, meaning no more error information will be forwarded to w0. The w0 crossbar connection will then be released to service other requests. The global_err_avail flag is reset by writing the Crossbow Error Command Word register. Writing the Crossbow Error Command Word register also has the side effect of clearing the global error registers.

There are also errors that are specific to widget 0. These include illegal register accesses and destination packet time-outs. These errors are handled in much the same manner as an error on a link controller. The packet information is logged if the global error registers are available and an interrupt request is generated if the interrupt for the error condition is enabled.

5.4.3 Interrupt processing

Each link controller and widget 0 itself generate interrupts by means of an interrupt request signal which is sent to the widget 0 interrupt processing logic. The interrupt requests are essentially all "or" ed by widget 0. Each interrupt request line will remain active until its error condition bit in the Link(x) Status register is cleared or its associated interrupt mask in the Link(x) Control register is enabled. When widget 0 detects the first interrupt, it will send an interrupt halt to the widget 0 xbar connection arbiter. After any current transaction is completed, the arbiter will grant no more crossbar connection requests until released by the interrupt control logic. Widget 0 will then form an "interrupt on" write request packet, arbitrate for the host destination port, and transfer the packet. At this point it will release the widget 0 crossbar connection arbiter.

When all interrupt requests have been de-asserted, the same process will occur with the exception of the fact that the "interrupts off" interrupt packet will contain different data.

5.4.4 Buffer Flush Handling

As discussed in the programmer's interface section of this document, it sometimes becomes necessary to determine when all data in an input packet buffer has been flushed to a particular destination. As a simplification, the flush mechanism actually verifies that all valid entries existing in the input packet buffer when the flush request was made will be flushed regardless of their destinations.

A widget wishing to flush an input packet buffer will send a read request to the link input buffer flush register associated with the packet buffer it wishes to flush. This request is sent to the widget 0 destination. The widget 0 controller will determine whether a buffer flush is currently in progress by checking internal flags. If no flushes are in progress, the widget 0 controller will assert the flag which indicates a flush is in progress. The widget 0 controller will then save the necessary information to form a read response packet. It then issues a flush request to the appropriate link controller. The flush is performed as a split read request transaction. While the flush is in progress, widget 0 continues to process future requests. The flush management logic in the link controller will then mark all valid buffer entries by setting the flush flag in their request manager's crossbar request queue entry. As the entries are retired, this flag is cleared. When all flags have been cleared, the flush management logic signals widget 0. Widget 0 then sends a read response to the requesting widget indicating the flush has completed (a data value of 0) and de-asserts the flush in progress flag.

If a flush is in progress, the widget 0 controller will form a read response and route it back to the requesting widget with a value of one. This indicates that the flush request was denied and the widget should retry the request later. Allowing only one outstanding flush in the Crossbow at a time eliminates the need for keeping contexts for multiple split transactions.

5.4.5 Timers

Widget 0 also contains the timer for the arbitration reload interval and the timer which sets the input packet buffer time-out interval. The timer outputs are sent to and monitored by all link controllers. The timer clock consists of a pre-scaled version of core_clock which provides 1.28 usecs per "tic". The arbitration reload timer is a 6 bit counter and the input packet buffer time-out timer consists of a 20 bit counter.

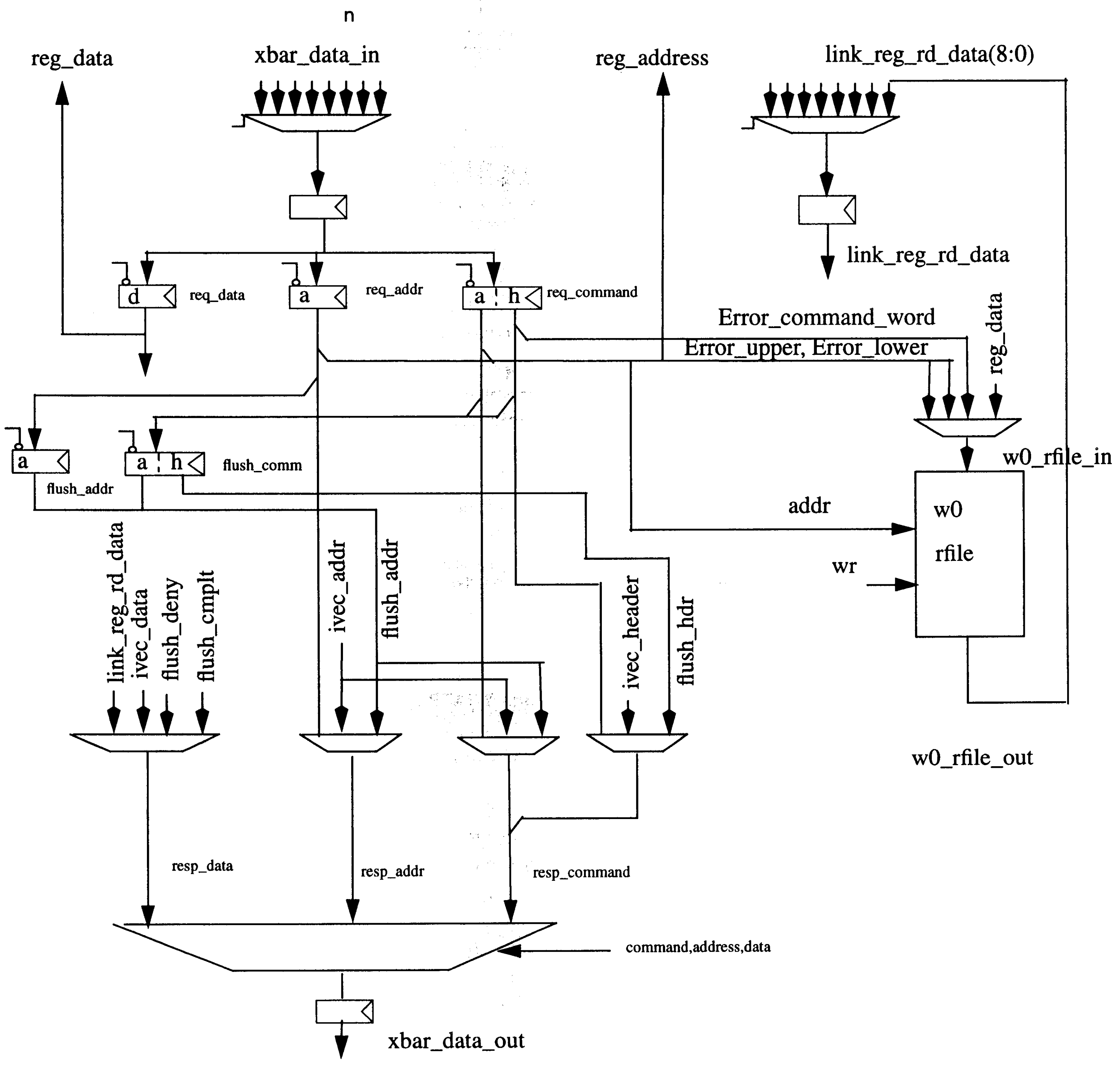


Figure 17 Widget 0 Datapath

5.5 Reset of Device

There are three reset conditions for the Crossbow Subsection of XBridge:

- Power-on Reset
- Link Reset
- Warm Reset

When power on reset is asserted, (from the PWR pin) all pertinent registers in the Crossbow Subsection of XBridge will be initialized. Power on reset will also be sent to the LLP blocks in all link controllers. Power on Reset is then de-asserted and the LLP will begin their reset sequence.

The reset sequence for the LPP is described as follows:

The LLP send module can be reset with either Reset (power on), LinkReset, or WarmReset. Each of these resets will clear the sender's transmit buffer and all other LLP internal state. During the reset state, the LLP also transmits special reset packets to the remote LLP, forcing that LLP module into reset as well. LinkReset and WarmReset both exist so that a higher level protocol can decide if just the local LLP link should be reset, or if a reset pulse should be propagated throughout the entire system. Note that the power-on Reset will look like a LinkReset to remote LLP receivers.

Before the LLP can exit the reset state, it must negotiate the link bit mode with the remote LLP. LinkReset will cause the LLP sender to transmit a continuous stream of reset micro-packets, with a small time gap between back-to-back packets. These packets contain the data width of the transmitter (8 or 16 bits), and continue to transmit until a bit mode has been negotiated. Since the physical data link width of the remote receiver is not known, reset packets are sent alternately in 8 and 16 bit modes. If a bit mode is not negotiated within MaxRetry reset micro-packet bursts, the link is assumed dead, and RetryTimeout is asserted.

Once a reset packet is received, the LLP knows it is connected to a live link. The LLP will then remain in a reset state for a number of clocks sufficient to cover the transmission of several reset micro-packets. This allows the LLP to negotiate for data link transmission size. Each side of the link transmits its actual physical size, and at the completion of the reset the data transmission size is set.

After power-on reset, the LLP will eventually come out of reset (i.e Buffull de-asserted, RcvLinkReset,RcvWarmReset, de-asserted), and begin normal operation. Those that are not connected or failed to come out of reset will have their RetryTimeout signals active. Active link connections are indicated by the value of the ***LLP Max Transmitter Retry*** bit in the Link(x) Status Register.

An individual XBridge port may be reset by an external widget when the widget sends a Link Rest pattern out onto the Crosstalk link. The LLPr in the crossbow will receive the pattern and generate a RcvLinkReset signal. The exception control block will cause the crossbar release signal to become active for a number of cycles to release any crossbar connections. The Link Controller associated with the port is then reset. No other links or Link Controllers are affected.

The XBridge itself may reset an individual port. This is done by setting the Link Reset bit in the Link(x) Control Register. Writing the bit causes a "one-shot effect" whereby SendWarmReset is asserted for a few cycles and then the Link Reset bit is cleared. This cause the LLP to begin transmitting warm reset patterns on the crosstalk interconnect. In a similar manner, the exception control block will force a disconnect from the crossbar and reset the Link Controller.

The Crossbow Subsection of XBridge may also receive a WarmReset pattern from an LLPr. A WarmReset differs from a Link reset in that it is propagated throughout the Crossbow Subsection of XBridge. When the RcvWarmReset signal becomes active from any LLPr, for all intents the Crossbow Subsection of XBridge will consider this a power on reset and reset all core logic. It will also propagate the reset and assert a SendWarmReset to all LLPt's in the XBridge.

As the WarmReset is propagated out, WarmResets are received back. These reset loops are disabled by the LLP which after detecting the first reset pattern, will ignore all subsequent reset requests for 1024 cycles.

CHAPTER 6

PCI Bridge Internal Registers

This section contains the register definitions for the various sections of the PCI bridge section of the XBridge 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 invalid address error. All the register descriptions contain a reset value field. An X in this field indicate a undefined state after reset.

6.1 Internal Register Address Map

The internal registers in the PCI 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. There are 2 PCI Bridge sections which occupy widget id number 0xe and 0xf. The addresses in the Table 37 are *Crosstalk* addresses.

All registers in the PCI 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			
PCI Bridge Identification Register	0x0000_0000	0x0F	read-only
PCI Bridge Status Register	0x0000_0008	0x0F	read-only
PCI Bridge Error Upper Address Register	0x0000_0010	0x0F	read-only
PCI Bridge Error Lower Address Register	0x0000_0018	0x0F	read-only
PCI Bridge Control Register	0x0000_0020	0x0F	read/write
PCI Bridge Request Time-out Value Register	0x0000_0028	0x0F	read/write
PCI Bridge Interrupt Destination Upper Address Register	0x0000_0030	0x0F	read/write
PCI Bridge Interrupt Destination Lower Address Register	0x0000_0038	0x0F	read/write
PCI Bridge Error Command Word Register	0x0000_0040	0x0F	read-only
PCI Bridge LLP Configuration Register	0x0000_0048	0x0F	read/write
PCI Bridge Target Flush Register	0x0000_0050	0x0F	read-only
Aux Error Command Word Register	0x0000_0058	0x0F	read-only
Response Buffer Error Upper Address Register	0x0000_0060	0x0F	read-only
Response Buffer Error 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
Map Fault Address Register	0x0000_0090	0x0F	read/write
Arbitration			
Arbitration Priority Register	0x0000_00A0	0x0F	read/write
Number In A Can			

Table 37 **PCI Bridge Register Address Map**

Registers	Double Word Addresses	Data Enables	Remarks
NIC Register	0x0000_00B0	0x0F	read/write
PCI			
PCI Time-out Register	0x0000_00C0	0x0F	read/write
PCI Type 1 Configuration Register	0x0000_00C8	0x0F	read/write
PCI Error Upper Address Register	0x0000_00D0	0x0F	read only
PCI Error Lower Address Register	0x0000_00D8	0x0F	read only
Interrupt			
PCI Bridge Interrupt Status Register	0x0000_0100	0x0F	read only
PCI 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 1 Host 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
Error Interrupt View Register	0x0000_0170	0x0F	read only
Multiple Interrupt Register	0x0000_0178	0x0F	read only

Table 37 PCI Bridge Register Address Map

Registers	Double Word Addresses	Data Enables	Remarks
Force Always Interrupt 0 Register	0x0000_0180	0x0F	write only
Force Always Interrupt 1 Register	0x0000_0188	0x0F	write only
Force Always Interrupt 2 Register	0x0000_0190	0x0F	write only
Force Always Interrupt 3 Register	0x0000_0198	0x0F	write only
Force Always Interrupt 4 Register	0x0000_01A0	0x0F	write only
Force Always Interrupt 5 Register	0x0000_01A8	0x0F	write only
Force Always Interrupt 6 Register	0x0000_01B0	0x0F	write only
Force Always Interrupt 7 Register	0x0000_01B8	0x0F	write only
Force w/Pin Interrupt 0 Register	0x0000_01C0	0x0F	write only
Force w/Pin Interrupt 1 Register	0x0000_01C8	0x0F	write only
Force w/Pin Interrupt 2 Register	0x0000_01D0	0x0F	write only
Force w/Pin Interrupt 3 Register	0x0000_01D8	0x0F	write only
Force w/Pin Interrupt 4 Register	0x0000_01E0	0x0F	write only
Force w/Pin Interrupt 5 Register	0x0000_01E8	0x0F	write only
Force w/Pin Interrupt 6 Register	0x0000_01F0	0x0F	write only
Force w/Pin Interrupt 7 Register	0x0000_01F8	0x0F	write only
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

Table 37 PCI Bridge Register Address Map

Registers	Double Word Addresses	Data Enables	Remarks
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
Buffer Address Match Registers			
Buffer 0 Upper Address Match Register	0x0000_0300	0x0F	read only
Buffer 0 Lower Address Match Register	0x0000_0308	0x0F	read only
Buffer 1 Upper Address Match Register	0x0000_0310	0x0F	read only
Buffer 1 Lower Address Match Register	0x0000_0318	0x0F	read only
Buffer 2 Upper Address Match Register	0x0000_0320	0x0F	read only
Buffer 2 Lower Address Match Register	0x0000_0328	0x0F	read only
Buffer 3 Upper Address Match Register	0x0000_0330	0x0F	read only
Buffer 3 Lower Address Match Register	0x0000_0338	0x0F	read only

Table 37 PCI Bridge Register Address Map

Registers	Double Word Addresses	Data Enables	Remarks
Buffer 4 Upper Address Match Register	0x0000_0340	0x0F	read only
Buffer 4 Lower Address Match Register	0x0000_0348	0x0F	read only
Buffer 5 Upper Address Match Register	0x0000_0350	0x0F	read only
Buffer 5 Lower Address Match Register	0x0000_0358	0x0F	read only
Buffer 6 Upper Address Match Register	0x0000_0360	0x0F	read only
Buffer 6 Lower Address Match Register	0x0000_0368	0x0F	read only
Buffer 7 Upper Address Match Register	0x0000_0370	0x0F	read only
Buffer 7 Lower Address Match Register	0x0000_0378	0x0F	read only
Buffer 8 Upper Address Match Register	0x0000_0380	0x0F	read only
Buffer 8 Lower Address Match Register	0x0000_0388	0x0F	read only
Buffer 9 Upper Address Match Register	0x0000_0390	0x0F	read only
Buffer 9 Lower Address Match Register	0x0000_0398	0x0F	read only
Buffer 10 Upper Address Match Register	0x0000_03A0	0x0F	read only
Buffer 10 Lower Address Match Register	0x0000_03A8	0x0F	read only
Buffer 11 Upper Address Match Register	0x0000_03B0	0x0F	read only
Buffer 11 Lower Address Match Register	0x0000_03B8	0x0F	read only
Buffer 12 Upper Address Match Register	0x0000_03C0	0x0F	read only
Buffer 12 Lower Address Match Register	0x0000_03C8	0x0F	read only
Buffer 13 Upper Address Match Register	0x0000_03D0	0x0F	read only
Buffer 13 Lower Address Match Register	0x0000_03D8	0x0F	read only
Buffer 14 Upper Address Match Register	0x0000_03E0	0x0F	read only
Buffer 14 Lower Address Match Register	0x0000_03E8	0x0F	read only
Buffer 15 Upper Address Match Register	0x0000_03F0	0x0F	read only

Table 37 **PCI Bridge Register Address Map**

Registers	Double Word Addresses	Data Enables	Remarks
Buffer 15 Lower Address Match Register	0x0000_03F8	0x0F	read only
Performance Monitor Registers			
Buffer 0 Flush Count with Data Touch Register	0x0000_0400	0x0F	read/write
Buffer 0 Flush Count w/o Data Touch Register	0x0000_0408	0x0F	read/write
Buffer 0 Request in Flight Count Register	0x0000_0410	0x0F	read/write
Buffer 0 Prefetch Request Count Register	0x0000_0418	0x0F	read/write
Buffer 0 Total PCI Retry Count Register	0x0000_0420	0x0F	read/write
Buffer 0 Max PCI Retry Count Register	0x0000_0428	0x0F	read/write
Buffer 0 Max Latency Count Register	0x0000_0430	0x0F	read/write
Buffer 0 Clear All Register	0x0000_0438	0x0F	read/write
Buffer 2 Flush Count with Data Touch Register	0x0000_0440	0x0F	read/write
Buffer 2 Flush Count w/o Data Touch Register	0x0000_0448	0x0F	read/write
Buffer 2 Request in Flight Count Register	0x0000_0450	0x0F	read/write
Buffer 2 Prefetch Request Count Register	0x0000_0458	0x0F	read/write
Buffer 2 Total PCI Retry Count Register	0x0000_0460	0x0F	read/write
Buffer 2 Max PCI Retry Count Register	0x0000_0468	0x0F	read/write
Buffer 2 Max Latency Count Register	0x0000_0470	0x0F	read/write
Buffer 2 Clear All Register	0x0000_0478	0x0F	read/write
Buffer 4 Flush Count with Data Touch Register	0x0000_0480	0x0F	read/write
Buffer 4 Flush Count w/o Data Touch Register	0x0000_0488	0x0F	read/write
Buffer 4 Request in Flight Count Register	0x0000_0490	0x0F	read/write
Buffer 4 Prefetch Request Count Register	0x0000_0498	0x0F	read/write
Buffer 4 Total PCI Retry Count Register	0x0000_04A0	0x0F	read/write

Table 37 PCI Bridge Register Address Map

Registers	Double Word Addresses	Data Enables	Remarks
Buffer 4 Max PCI Retry Count Register	0x0000_04A8	0x0F	read/write
Buffer 4 Max Latency Count Register	0x0000_04B0	0x0F	read/write
Buffer 4 Clear All Register	0x0000_04B8	0x0F	read/write
Buffer 6 Flush Count with Data Touch Register	0x0000_04C0	0x0F	read/write
Buffer 6 Flush Count w/o Data Touch Register	0x0000_04C8	0x0F	read/write
Buffer 6 Request in Flight Count Register	0x0000_04D0	0x0F	read/write
Buffer 6 Prefetch Request Count Register	0x0000_04D8	0x0F	read/write
Buffer 6 Total PCI Retry Count Register	0x0000_04E0	0x0F	read/write
Buffer 6 Max PCI Retry Count Register	0x0000_04E8	0x0F	read/write
Buffer 6 Max Latency Count Register	0x0000_04F0	0x0F	read/write
Buffer 6 Clear All Register	0x0000_04F8	0x0F	read/write
Buffer 8 Flush Count with Data Touch Register	0x0000_0500	0x0F	read/write
Buffer 8 Flush Count w/o Data Touch Register	0x0000_0508	0x0F	read/write
Buffer 8 Request in Flight Count Register	0x0000_0510	0x0F	read/write
Buffer 8 Prefetch Request Count Register	0x0000_0518	0x0F	read/write
Buffer 8 Total PCI Retry Count Register	0x0000_0520	0x0F	read/write
Buffer 8 Max PCI Retry Count Register	0x0000_0528	0x0F	read/write
Buffer 8 Max Latency Count Register	0x0000_0530	0x0F	read/write
Buffer 8 Clear All Register	0x0000_0538	0x0F	read/write
Buffer 10 Flush Count with Data Touch Register	0x0000_0540	0x0F	read/write
Buffer 10 Flush Count w/o Data Touch Register	0x0000_0548	0x0F	read/write
Buffer 10 Request in Flight Count Register	0x0000_0550	0x0F	read/write
Buffer 10 Prefetch Request Count Register	0x0000_0558	0x0F	read/write

Table 37 **PCI Bridge Register Address Map**

Registers	Double Word Addresses	Data Enables	Remarks
Buffer 10 Total PCI Retry Count Register	0x0000_0560	0x0F	read/write
Buffer 10 Max PCI Retry Count Register	0x0000_0568	0x0F	read/write
Buffer 10 Max Latency Count Register	0x0000_0570	0x0F	read/write
Buffer 10 Clear All Register	0x0000_0578	0x0F	read/write
Buffer 12 Flush Count with Data Touch Register	0x0000_0580	0x0F	read/write
Buffer 12 Flush Count w/o Data Touch Register	0x0000_0588	0x0F	read/write
Buffer 12 Request in Flight Count Register	0x0000_0590	0x0F	read/write
Buffer 12 Prefetch Request Count Register	0x0000_0598	0x0F	read/write
Buffer 12 Total PCI Retry Count Register	0x0000_05A0	0x0F	read/write
Buffer 12 Max PCI Retry Count Register	0x0000_05A8	0x0F	read/write
Buffer 12 Max Latency Count Register	0x0000_05B0	0x0F	read/write
Buffer 12 Clear All Register	0x0000_05B8	0x0F	read/write
Buffer 14 Flush Count with Data Touch Register	0x0000_05C0	0x0F	read/write
Buffer 14 Flush Count w/o Data Touch Register	0x0000_05C8	0x0F	read/write
Buffer 14 Request in Flight Count Register	0x0000_05D0	0x0F	read/write
Buffer 14 Prefetch Request Count Register	0x0000_05D8	0x0F	read/write
Buffer 14 Total PCI Retry Count Register	0x0000_05E0	0x0F	read/write
Buffer 14 Max PCI Retry Count Register	0x0000_05E8	0x0F	read/write
Buffer 14 Max Latency Count Register	0x0000_05F0	0x0F	read/write
Buffer 14 Clear All Register	0x0000_05F8	0x0F	read/write

Table 37 PCI Bridge Register Address Map

6.2 *Crosstalk* Registers

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

- PCI Bridge Identification Register
- PCI Bridge Status Register
- PCI Bridge Error Upper Address Register
- PCI Bridge Error Lower Address Register
- PCI Bridge Control Register
- PCI Bridge Request Time-Out Value Register
- PCI Bridge Interrupt Destination Upper Address Register
- PCI Bridge Interrupt Destination Lower Address Register
- PCI Bridge Error Command Word Register
- LLP Configuration Register
- PCI Bridge Target Flush Register
- Aux Error Command Register
- Response Buffer Error Upper Address Register
- Response Buffer Error Lower Address Register
- Test Pin Control Register

6.2.1 PCI Bridge Identification Register

The PCI 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	0xD002	Part Number “Widget Type”
MFG_NUM	11:1	00000100100	Manufacturer Identity
	0	1	Always read as 1

Table 38 PCI Bridge Identification Register

6.2.2 PCI Bridge Status Register

The status register is a read register which holds status information of the PCI 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
RX_CREDIT_CNT	15:12		Receive Credit Count
TX_CREDIT_CNT	11:8		Transmit Credit Count
PCI_MISC_INPUT	7:0		Misc input pins

Table 39 PCI Bridge Status Register

6.2.3 PCI Bridge Error Upper Address

The PCI 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 PCI 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 40 PCI Bridge Error Upper Address Register

6.2.4 PCI Bridge Error Lower Address

The PCI 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 PCI 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 41 **PCI Bridge Error Lower Address Register**

6.2.5 PCI Bridge Control Register

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

Name	Bits	Reset Value	Definition
	31:28		Reserved
RST_PIN_N(3:0)	27:24	0xF	Software programmable reset pins. A "1" provides a high level on the rst_pin_n().
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.
	20:17		Reserved
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.

Table 42 **PCI Bridge Control Register**

Name	Bits	Reset Value	Definition
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
	8:6		Reserved
PCI_SPEED	5:4		Read Only Pci Speed Select Input
WIDGET_ID	3:0	0xF	Widget ID number.

Table 42 PCI Bridge Control Register

6.2.6 PCI 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 43 PCI Bridge Request Time-out Value Register

6.2.7 PCI Bridge Interrupt Destination Upper Address Register

The PCI 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 44 PCI Bridge Interrupt Destination Upper Address Register

6.2.8 PCI Bridge Interrupt Destination Lower Address Register

The PCI 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 45 PCI Bridge Interrupt Destination Lower Address Register

6.2.9 PCI Bridge Error Command Word Register

The PCI 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 46 PCI Bridge Error Command Word Register

6.2.10 PCI Bridge LLP Configuration Register

This register contains the configuration information for the LLP modules **only in Super Bridge Mode**.

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 47 PCI Bridge LLP Configuration Register

6.2.11 PCI Bridge Target Flush Register

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

6.2.12 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 48 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 48 **Aux Error Command Word Register**

6.2.13 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 49 **Response Buffer Error Upper Address Register**

6.2.14 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 50 Response Buffer Error Lower Address Register

6.2.15 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
	31:24	0	Reserved
TDATA_OUT	23:16	0	Test Data Out to Pins
SEL_TPIN_7	15:14	0	Select function for test pin 7 11 => LLP Max Retry Timeout 10 => CRP_GRP 01 => TDATA_OUT (7) 00 = >Quickswitch enable mode
SEL_TPIN_6	13:12	0	Select function for test pin 6 11 => LLP Buff Full 10 => REQ_DSP_GRP 01 => TDATA_OUT (6) 00 = > Quickswitch enable mode
SEL_TPIN_5	11:10	0	Select function for test pin 5 11 =>LLP_GRP 10 => PCI_GRP 01 => TDATA_OUT (5) 00 = > Quickswitch enable mode

Table 51 Test Pin Control Register

Name	Bits	Reset Value	Description
SEL_TPIN_4	9:8	0	Select function for test pin 4 11 => RESP_BUF_GRP 10 => PMU_GRP 01 =>TDATA_OUT (4) 00 = >Quickswitch enable mode
SEL_TPIN_3	7:6	0	Select function for test pin 3 11 => 0 10 => 0 01 =>TDATA_OUT (3) 00 = >Quickswitch enable mode
SEL_TPIN_2	5:4	0	Select function for test pin 2 11 => 0 10 => 0 01 =>TDATA_OUT (2) 00 = >Quickswitch enable mode
SEL_TPIN_1	3:2	0	Select function for test pin 1 11 => 0 10 => 0 01 =>TDATA_OUT (1) 00 = >Quickswitch enable mode
SEL_TPIN_0	1:0	0	Select function for test pin 0 11 => 0 10 => 0 01 =>TDATA_OUT (0) 00 = >Quickswitch enable mode

Table 51 **Test Pin Control Register**

6.3 Mapping Registers

6.3.1 PCI Bridge Direct Mapping Register

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

Name	Bits	Reset Value	Description
	31:24		Reserved
DIR_W_ID	23:20	0	Direct space target widget ID
	19:18		Reserved
DIR_ADD512	17	1	Direct add 512MB offset only when DIR_OFF = 0x000
DIR_OFF	16:0	0x000	Direct Map Offset

Table 52 **PCI Bridge Auxiliary Space and Direct Mapping Register**

6.4 Arbitration Register

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

Name	Bits	Reset Value	Description
	31:18	0	Reserved
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

Table 53 **ARB_PRIORITY Register**

Name	Bits	Reset Value	Description
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	0	Reserved
FREEZE_GNT	6	0	When this bit is set by host, no grant will be issued to the PCI 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 53 ARB_PRIORITY Register

6.5 NIC Register

This register contains the number in a can support. XBridge uses the Crossbow NIC in XBridge Mode and this NIC (on Bridge port F only) in Super Bridge Mode.

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 54 NIC Register

6.6 PCI

PCI registers are those registers which are associated with the PCI bus operation.

- PCI Time-out Register
- PCI Type 1 Configuration Register
- PCI Error Upper Address Register
- PCI Error Lower Address Register

6.6.1 Time-out Register

This register determines retry hold off and max retries allowed for PIO accesses to PCI.

Name	Bits	Reset Value	Description
	31:21	0	Reserved
PCI_RETRY_HLD	20:16	1	The PCI retry hold is the amount of time the PCI 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.
	15:10	0	Reserved
PCI_RETRY_CNT	9:0	0	The PCI retry count determines the number of retry operations the PCI Bridge will attempt as a master before taking an error exception. If this value is 0, PCI Bridge will always retry the command until it completes.

Table 55 PCI Time-out Register

6.6.2 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 56 PCI Type 1 Configuration Register

6.6.3 PCI Error Upper Address Register

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

Name	Bits	Reset Value	Description
	31:28	X	Reserved
PCI_XTALK_DID	27:24	X	Target Xtalk ID
	23:22	X	Reserved
PCI_DAC	21	X	Dual Address Cycle
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.

Name	Bits	Reset Value	Description
PCI_UADDR_ERR	15:0	X	This is the address of the current PCI bus command, PCI address 47:32.

Table 57 PCI Error Upper Address Register

6.6.4 PCI Error Lower Address Register

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

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

Table 58 PCI Error Lower Address Register

6.7 Interrupt Registers

Interrupt registers are those registers which are associated with the control of interrupts inside the PCI Bridge.

- PCI Bridge Interrupt Status Register
- PCI Bridge Interrupt Enable Register
- Reset Interrupt Status Register
- Interrupt Mode Register
- Interrupt Device Register
- Host Error Field Register
- Interrupt (x) Host Address Register
- Error Interrupt View Register
- Multiple Interrupt Register
- Force Always Interrupt (x) Register
- Force w/Pin Interrupt (x) Register

6.7.1 INT_STATUS Register

This is the current interrupt status register which maintains the current status of all the interrupting devices which generated a n interrupt. 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
	31	X	Reserved
PMU_PAGE_FAULT	30	0	Indicates an Invalid Page Access
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.
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.

Table 59 **Bridge INT_STATUS Register**

Name	Bits	Reset Value	Description
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 micro-packet. 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 0xff 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.
LLP_TCTY	17	0	LLP Transmitter Retry Count Interrupt is generates when the transmitter retry counter rolls from 0xff to 0.
	16	0	Reserved
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 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.
	8	0	Reserved
INT_STATE	7:0	0	Status of INT_N[7:0]. A 1 means INT_N is low.

Table 59 Bridge INT_STATUS Register

6.7.2 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
	31	X	Reserved
PMU_PAGE_FAULT	30	0	Indicates an Invalid Page Access
EN_UNEXPECTED_RESP	29	0	Enables unexpected response interrupt.
EN_BAD_XRESP_PACKET	28	0	Enables bad Crosstalk packet interrupt.
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.
	16	0	Reserved
EN_PCI_ABORT	15	0	Enables PCI master/slave abort condition interrupt

Table 60 **Bridge INT_ENABLE Register**

Name	Bits	Reset Value	Description
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 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.
	8	0	Reserved
EN_INT_STATE	7:0	0	Enables INT_N[7:0].

Table 60 **Bridge INT_ENABLE Register**

6.7.3 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	ERROR_VIEW
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

Table 61 **Bridge INT_RESET Register**

Name	Bits	Reset Value	Writing a one clears the following:
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
PMU_PAGE_FAULT	1	0	Page_Fault
PCI_GRP_CLR	0	0	PCI_ABORT, PCI_PARITY, PCI_SERR, PCI_PERR, PCI_MASTER_TOUT, PCI_RETRY_CNT

Table 61 **Bridge INT_RESET Register**

6.7.4 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 62 **INT_MODE register**

6.7.5 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).
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 63 **INT_DEV register**

6.7.6 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 PCI Bridge error in host interrupt register.

Table 64 HOST_ERR_FIELD register

6.7.7 Interrupt (x) Host Address 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 65 Interrupt (x) Host register

6.7.8 Error Interrupt View Register

This register contains the view of which interrupt occur even if they are not currently enabled. The group clear is used to clear these bits just like the interrupt status register bits.

Name	Bits	Reset Value	Description
	31	X	Reserved
PMU_PAGE_FAULT	30	0	Indicates an Invalid Page Access
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.
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.

Table 66 Error Interrupt View Register

Name	Bits	Reset Value	Description
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 micro-packet. 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 0xff 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.
LLP_TCTY	17	0	LLP Transmitter Retry Count Interrupt is generates when the transmitter retry counter rolls from 0xff to 0.
	16	0	Reserved
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 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.

Table 66 **Error Interrupt View Register**

Name	Bits	Reset Value	Description
	8:0	0	Reserved

Table 66 Error Interrupt View Register

6.7.9 Multiple Interrupt Register

This register indicates if any interrupt occurs more than once without being cleared.

Name	Bits	Reset Value	Description
	31	X	Reserved
PMU_PAGE_FAULT	30	0	Indicates an Invalid Page Access
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.
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.

Table 67 Bridge INT_STATUS Register

Name	Bits	Reset Value	Description
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 micro-packet. 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 0xff 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.
LLP_TCTY	17	0	LLP Transmitter Retry Count Interrupt is gener-ates when the transmitter retry counter rolls from 0xff to 0.
	16	0	Reserved
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 parity error interrupt
PCI_MASTER_TOUT	11	0	Indicates a device select timeout on the PCI bus. or a GIO timeout.

Table 67 Bridge INT_STATUS Register

Name	Bits	Reset Value	Description
PCI_RETRY_CNT	10	0	PCI retry operation count exhausted.
XREAD_REQ_TOUT	9	0	PCI to Crosstalk read request timeout.
	8	0	Reserved
INT_STATE	7:0	0	Status of INT_N[7:0]. A 1 means INT_N is low.

Table 67 Bridge INT_STATUS Register

6.7.10 Force Always Interrupt (x) Register

A write to this data independent write only register will force a set interrupt to occur as if the interrupt line had transitioned. If the interrupt line is already active an addition set interrupt packet is set. All buffer flush operations also occur on this operation.

6.7.11 Force Interrupt (x) Register

A write to this data independent write only register in conjunction with the assertion of the corresponding interrupt line will force a set interrupt to occur as if the interrupt line had transitioned. The interrupt line must be active for this operation to generate a set packet, otherwise the write PIO is ignored. All buffer flush operations also occur when the set packet is sent on this operation.

6.8 Device Registers

Device registers are those registers which are associated with the control of PCI buffer and mapping operations on a per device basis.

- Device (x) Register
- Device (x) Write Request Buffer Register
- Even Device Response Buffer Register
- Odd Device Response Buffer Register
- Read Response Buffer Status Register
- Read Response Buffer Clear Register
- Buffer Address Match Registers

- Buffer (x) Upper Address Match Register
- Buffer (x) Lower Address Match Register

6.8.1 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.
	24		Reserved
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.
REAL_TIME	21	0	Real Time Device Enable 1 => enabled. This bit is programmed to select which arbitration ring the device operates in.
	20		Reserved
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. 32-bit direct space

Table 68 Device (x) Registers

Name	Bits	Reset Value	Description
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 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 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.
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 32-bit 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.

Table 68 Device (x) Registers

Name	Bits	Reset Value	Description
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 bus.

Table 68 Device (x) Registers

6.8.2 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.

6.8.3 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.

Table 69 Even Device Read Response Buffer Register

Name	Bits	Reset Value	Description
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.
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 69 Even Device Read Response Buffer Register

6.8.4 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.
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.

Table 70 Odd Device Read Response Buffer Register

Name	Bits	Reset Value	Description
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 70 Odd Device Read Response Buffer Register

6.8.5 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 71 Read Response Buffer Status Register

6.8.6 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 72 Read Response Buffer Clear Register

6.8.7 PCI Bridge Buffer (x) Upper address Register

The PCI Bridge buffer upper address register is a read only register which contains the upper 16-bits of the address and status used to select the buffer for a PCI transaction.

Name	Bits	Reset Value	Description
FILLED	31	X	Filled indicates that the buffer has valid data
ARMED	30	X	Armed indicates that request has been sent but data has not arrived
FLUSH	29	X	Flush indicates that current outstanding buffer is marked to be flushed as soon as data arrives
XERR	28	X	Xerr indicates that a xtalk cw error bit or invalid sideband error occurred
PKTERR	27	X	Pkterr indicates that the incoming packet was the wrong size
TIMEOUT	26	X	Timeout indicates the response has not arrived within the time limits
PREFETCH	25	X	Request type Prefetch
PRECISE	24	X	Request type Precise
DW_BE	23:16	X	Byte Enables used for precise transactions.
UPP_ADDR	15:0	X	Address Bits 47:32

Table 73 PCI Bridge Buffer Upper Address Register

6.8.8 PCI Bridge Buffer (x) Lower Address Register

The PCU Bridge buffer lower address register is a read only register which contains the lower 32-bits of the address used to selection buffer to a PCI transaction.

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

Table 74 PCI Bridge Buffer Lower Address Register

6.9 Performance Registers

The Performance registers are those registers which are associated with monitoring the performance of PCI generated reads to the host environment. Because of the size of the register file only the even registers were instrumented.

- Buffer (x) Flush Count with Data Touch Register
- Buffer (x) Flush Count w/o Data Touch Register
- Buffer (x) Request in Flight Count Register
- Buffer (x) Prefetch Request Count Register
- Buffer (x) Total PCI Retry Count Register
- Buffer (x) Max PCI Retry Count Register
- Buffer (x) Max Latency Count Register
- Buffer (x) Clear All Register

6.9.1 Buffer (x) Flush Count with Data Touch Register

This counter is incremented each time the corresponding response buffer is flushed after at least a single data element in the buffer is used. A word write to this address clears the count.

Name	Bits	Reset Value	Description
	31:16		Reserved
TOUCH_CNT	15:0	X	Count of Buffer Flushes with Data Touch

Table 75 **Flush Count with Data Touch Register**

6.9.2 Buffer (x) Flush Count w/o Data Touch Register

This counter is incremented each time the corresponding response buffer is flushed without any data element in the buffer being used. A word write to this address clears the count.

Name	Bits	Reset Value	Description
	31:16		Reserved
NOTOUCH_CNT	15:0	X	Count of Buffer Flushes without Data Touch

Table 76 **Flush Count without Data Touch Register**

6.9.3 Buffer (x) Request in Flight Count Register

This counter is incremented on each bus clock while the request is in-flight. A word write to this address clears the count.

Name	Bits	Reset Value	Description
	31:16		Reserved
INFLIGHT_CNT	15:0	X	In-flight Count

Table 77 **In-flight Count Register**

6.9.4 Buffer (x) Prefetch Request Count Register

This counter is incremented each time the request using this buffer was generated from the prefetcher. A word write to this address clears the count.

Name	Bits	Reset Value	Description
	31:16		Reserved
PREFETCH_CNT	15:0	X	Prefetch Count

Table 78 **Prefetch Count Register**

6.9.5 Buffer (x) Total PCI Retry Count Register

This counter is incremented each time a PCI bus retry occurs and the address matches the tag for the selected buffer. The buffer must also has this request in-flight. A word write to this address clears the count.

Name	Bits	Reset Value	Description
	31:16		Reserved
RETRY_CNT	15:0	X	Retry Count

Table 79 **PCI Retry Count Register**

6.9.6 Buffer (x) Max PCI Retry Count Register

This counter is contains the maximum retry count for a single request which was in-flight for this buffer. A word write to this address clears the count.

Name	Bits	Reset Value	Description
	31:16		Reserved
MAX_RETRY_CNT	15:0	X	Max Retry Count

Table 80 **Max PCI Retry Count Register**

6.9.7 Buffer (x) Max Latency Count Register

This counter is contains the maximum count (in bus clocks) for a single request which was in-flight for this buffer. A word write to this address clears the count.

Name	Bits	Reset Value	Description
	31:16		Reserved
MAX_LATENCY_CNT	15:0	X	Max Latency Count

Table 81 Max Latency Count Register

6.9.8 Buffer (x) Clear All Register

Any access to this register clears all the count values for the (x) registers.

CHAPTER 7

PCI Bridge Address Mapping

The Bridge section of the XBridge ASIC performs two types of address mapping, *Crosstalk* interconnect to PCI bus {this direction of flow is referred to as *pio mode*} and PCI 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 each Bridge section of the XBridge uses the lower 8 Giga Bytes. The PCI bus specification defines multiple address spaces and sizes which are supported in the Bridge section of the XBridge and are defined later in this chapter.

The pio mode bus translation mechanism consists of generating a 32-bit PCI 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.

7.1 Bus Address Maps

7.1.1 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 allowed 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 section of the XBridge 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 section of the XBridge 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 - 0x407f_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 82 Bridge section of the XBridge PCI Memory Space Usage

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

7.1.2 *Crosstalk* System Address Map

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

7.2 *Crosstalk* to PCI Bus Mapping

The Bridge section of the XBridge ASIC performs address translation from *Crosstalk* system addresses to PCI 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.

7.2.1 *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 section of the XBridge 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 section of the XBridge 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 section of the XBridge will not allow loopback mapping.

The *Crosstalk* interconnect specification defines a 16 MByte section for each widget called Widget Space. (See Figure 18) Widget space is used to access internal registers, and PCI devices. Widget space in the Bridge section of the XBridge ASIC is divided into 10 sections, six 1 MByte sections, three 2 MByte sections, and one 4MByte section. (Figure 18) The first 2 MByte section is used for Bridge section of the XBridge ASIC internal registers, internal address translation entry ram, PCI interrupt acknowledge cycle generation, PCI special cycle generation and PCI configuration space. The 4 MByte section is translated to either pci memory space addesses 0x0 to 0x3f_ffff or external flash port depending on which pci bridge. This address is translated from the processor boot address of 00_1fc0_0000 by the memory controller. The remaining 8 sections are used to access 8 PCI devices. The six 1 MByte sections are used for PCI devices 2 through 7. The two 2 MByte sections are used for PCI devices 0 and 1.

The Bridge section of the XBridge ASIC also supports mapping a larger address then the 16MByte widget space. The larger mapping consists of an eight Giga-byte 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.

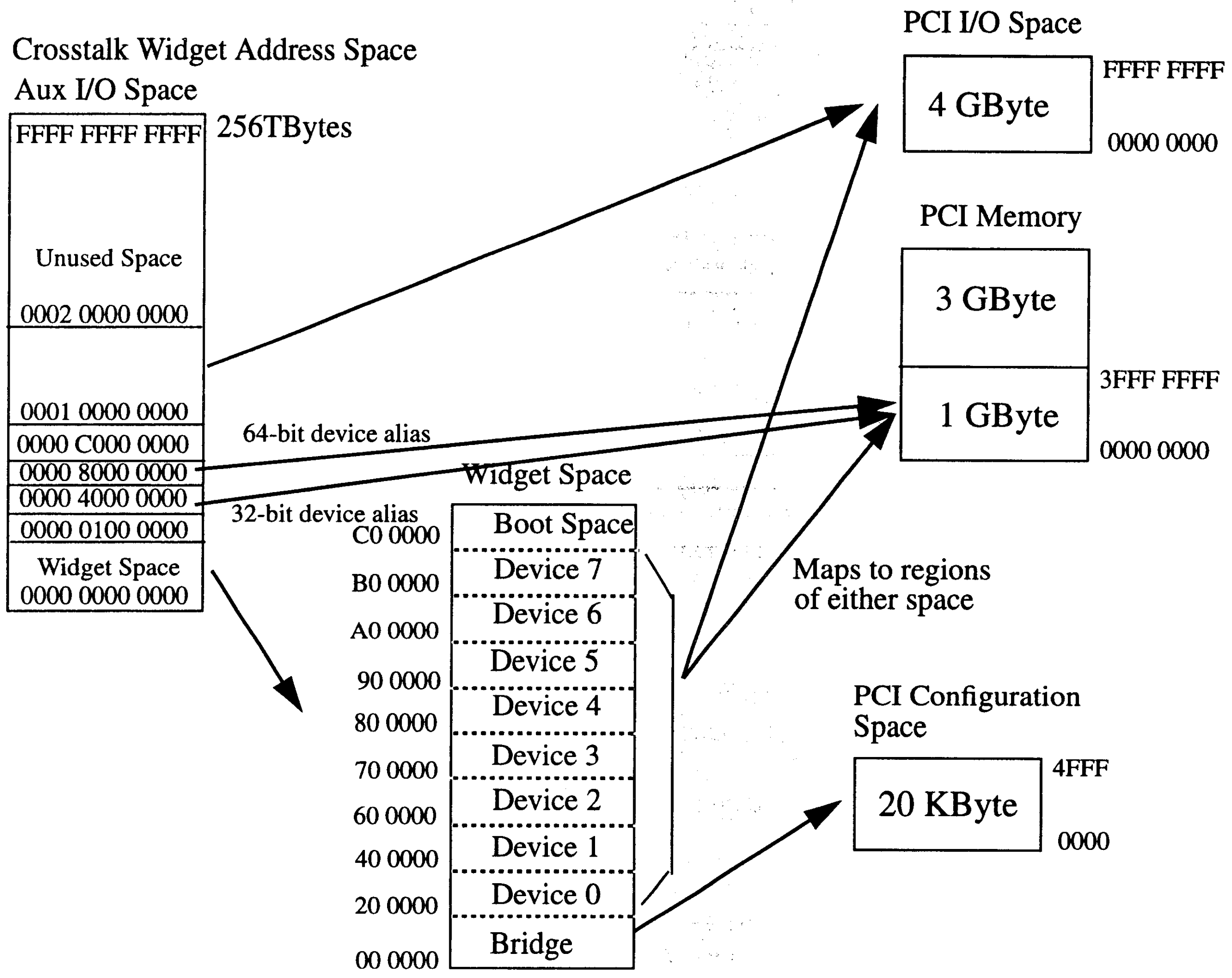


Figure 18 Crosstalk View of the PCI Bus

7.2.2 *Crosstalk* Mapping Registers

The *Crosstalk* mapping is controlled by the eight Device Registers. The Bridge section of the XBridge uses *Crosstalk* addresses to determine if the access is to widget space, or other areas mapped through the Bridge section of the XBridge . As shown in Figure 18, widget space contains sections which are mapped to the address spaces on the PCI buses.

In widget space, the Device Register bit DEV_IO_MEM is used to enable either memory or I/O space. 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
7	0 i/o space	11:0	31:20

Table 83 **Device Offset Index DEV_OFF**

The MEM_SWAP bit is used to control byte swapping for accesses to PCI memory 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.

7.2.3 *Crosstalk* Widget Space Address Map

Crosstalk Address (Widget Space)	Description
0x00_0000 -> 0x00_FFFF	Local Registers See Chapter 2
0x01_0000 -> 0x01_0FFF	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 -> 0x1F_FFFF	Reserved
0x20_0000 -> 0x3F_FFFF	PCI Device 0 Space
0x40_0000 -> 0x5F_FFFF	PCI Device 1 Space
0x60_0000 -> 0x6F_FFFF	PCI Device 2 Space
0x70_0000 -> 0x7F_FFFF	PCI Device 3 Space
0x80_0000 -> 0x8F_FFFF	PCI Device 4 Space
0x90_0000 -> 0x9F_FFFF	PCI Device 5 Space
0xA0_0000 -> 0xAF_FFFF	PCI Device 6 Space
0xB0_0000 -> 0xBF_FFFF	PCI Device 7 Space
0xC0_0000 -> 0xFF_FFFF	PCI Bus 0x0

Table 84 **Widget Space Address Map**

Table 84 contains the mapping for the Bridge section of the XBridge widget space. Table 85 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	
0x30	Expansion ROM Base Address			
0x34 -> 0x38	Reserved			
0x3C	Max_Lat	Min_Gnt	Int Pin	Int Line
0x40 -> 0x100	Device Specific			

Table 85 PCI Configuration Space

7.3 PCI Bus to *Crosstalk* Mapping

The Bridge section of the XBridge ASIC supports two mechanisms to translate PCI 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 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 bus address. Since some modes of the PCI 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 512 4KByte or 16KByte pages, anywhere in *Crosstalk* system space for a total of 8 MByte of mapped space. Both the page mapped and direct schemes provide coherency, prefetching, and precise attributes to enhance Bridge section of the XBridge performance. Each of these attributes are selectable for each page in the page mapping area or per device in the direct map area. Section 7.3.3 provides a explanation of the attribute functionality.

7.3.1 PCI View of *Crosstalk*

A PCI bus device can only access the *Crosstalk* interconnect address space through the PCI memory address space. Figure 19 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 section of the XBridge 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 section of the XBridge maps 256 TBytes through using the upper address bits as attributes for the transaction.

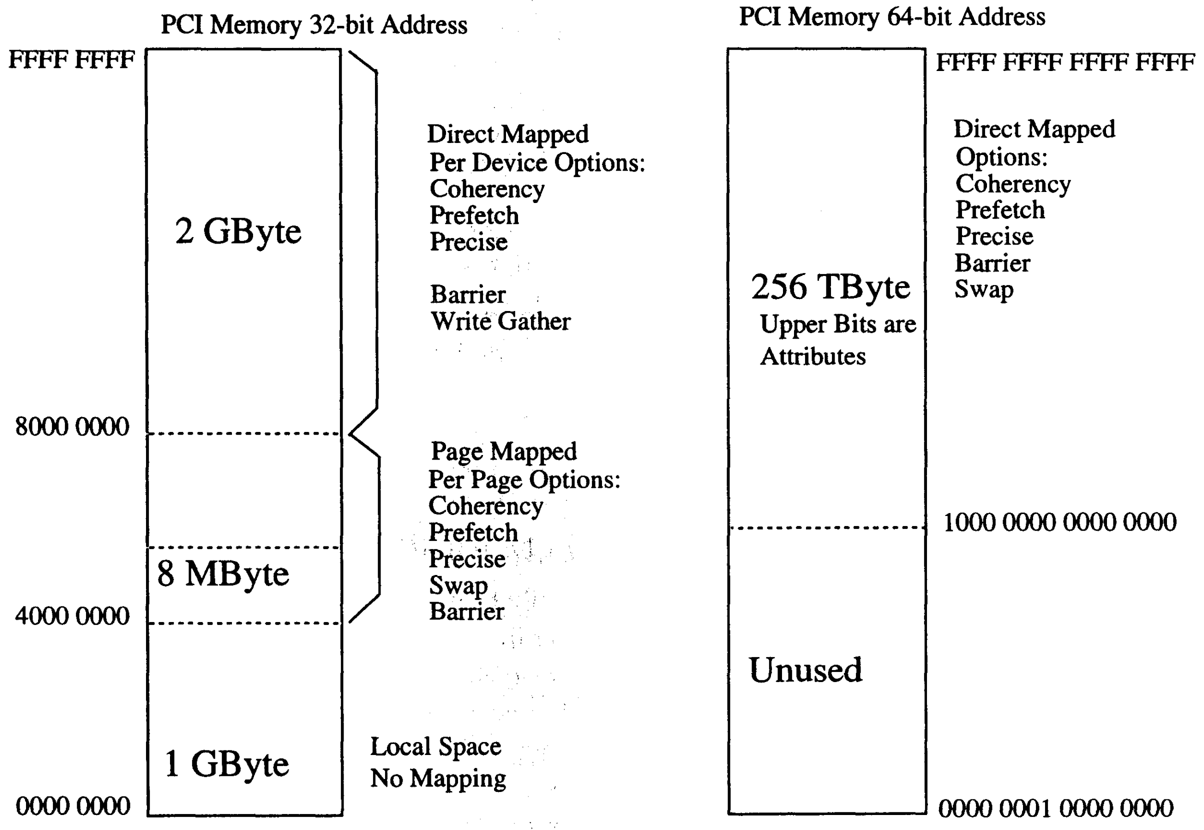


Figure 19 32-bit Address PCI View of Crosstalk

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.

7.3.2 PCI Direct Mapping

The Bridge section of the XBridge 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 devices to map system resources. As shown in Figure 19, direct mapping remaps a section of the PCI memory space to a section of the *Crosstalk* system space.

In 32-bit direct mapped mode, access to the upper 2 GBytes of PCI 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 86 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
0x00000	0x0000_0000_0000 - 0x0000_7FFF_FFFF

Table 86 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.

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 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	Swap
54:48	Unused

Table 87 **64-bit Address Direct Map Attributes**

Note: The Bridge section of the XBridge 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 in the xtalk packet is always set to 0 in all modes. 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 88 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 bus. Normally a cache line read buffer is invalidated after the current cycle is completed, either FRAME_N negated. 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 bus transactions to a single buffer. For a complete invalidation rules see PCI 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.

Table 88 **Direct Mapping Attributes Per Device**

Name	Description
SWAP	Device Enable Swapping 1 => enabled. When enabled the byte swapper exchanges data per Appendix A. 64-bit address mode selectable.
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 88 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 89 Address Attributes In 32 Bit Address Direct Space

Attribute	How Attribute Is Specified
Prefetch	Bit 59 of the address.
Swap	Bit 55 of the address.
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 90 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 ATE
Coherent	Coherent bit in page map entry.
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 91 Address Attributes In Page Mapped Space

7.3.3 PCI Page Mapping

The PCI PMU (Page Mapping Unit) is the mechanism which performs the page mapping of PCI bus addresses to *Crosstalk* system addresses. The PMU translates a 8 MByte 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 section of the XBridge 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 92. The ATE provides the translated address, remote map field and attributes.

The Bridge section of the XBridge supports an internal RAM map containing 512 entries starting from the address 0x4000_0000_0000_0000. Also pages mapped with the internal map are **ALWAYS WRITE GATHER DISABLED.**

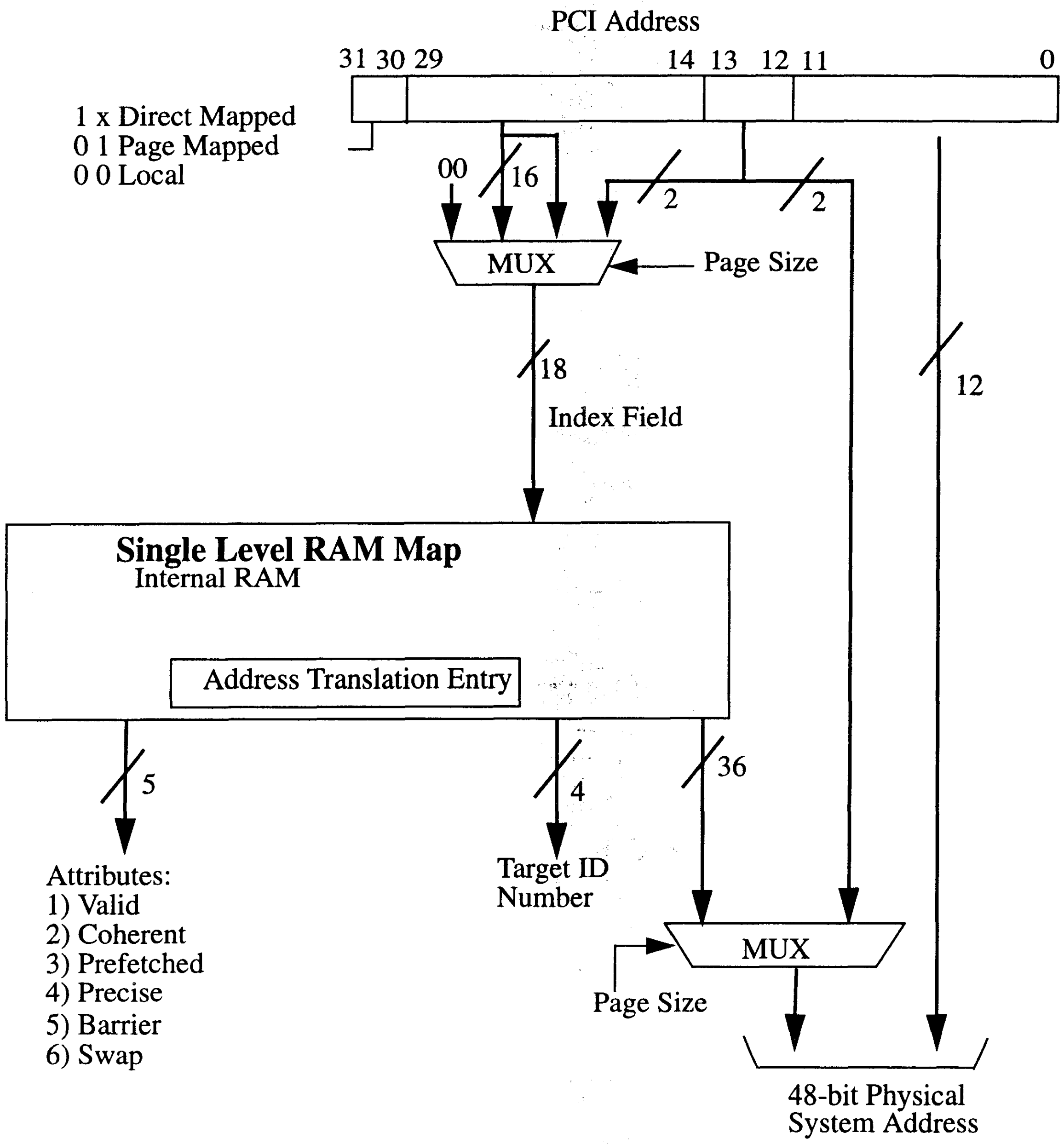


Figure 20 PCI PMU

Programmers note: The page 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.

Figure 20 contains the translation path for a PCI address into a *Crosstalk* interconnect system address. Bits 31:30 of the PCI 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.

Table 92 contains a description of the 8-byte (64-bit) address translation entry. The GBR bit is controlled from the device registers.

Bits	Name	Description
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.
5	Swap	Swap Data 1 => byte swap . This bit is causes the hardware to perform a byte swap on the data.
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.
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 bus. Normally a cache line read buffer is invalidated after the current cycle is completed, either FRAME_N negated . 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 bus transactions to a single buffer. For a complete invalidation rules see PCI 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.

Bits	Name	Description
0	Entry Valid	Entry Valid bit indicates that the address translation entry contains valid mapping information.

Table 92 **PMU Address Translation Entry**

7.3.4 Packetization of PCI Operations

PCI 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 section of the XBridge 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 section of the XBridge uses the PCI command ***Read Line*** and ***Read Multiple*** like pci read operation.

7.3.5 PCI Address Translation Flow Chart

Figure 21, Figure 22, and Figure 23 contain flow charts of PCI to *Crosstalk* address translation.

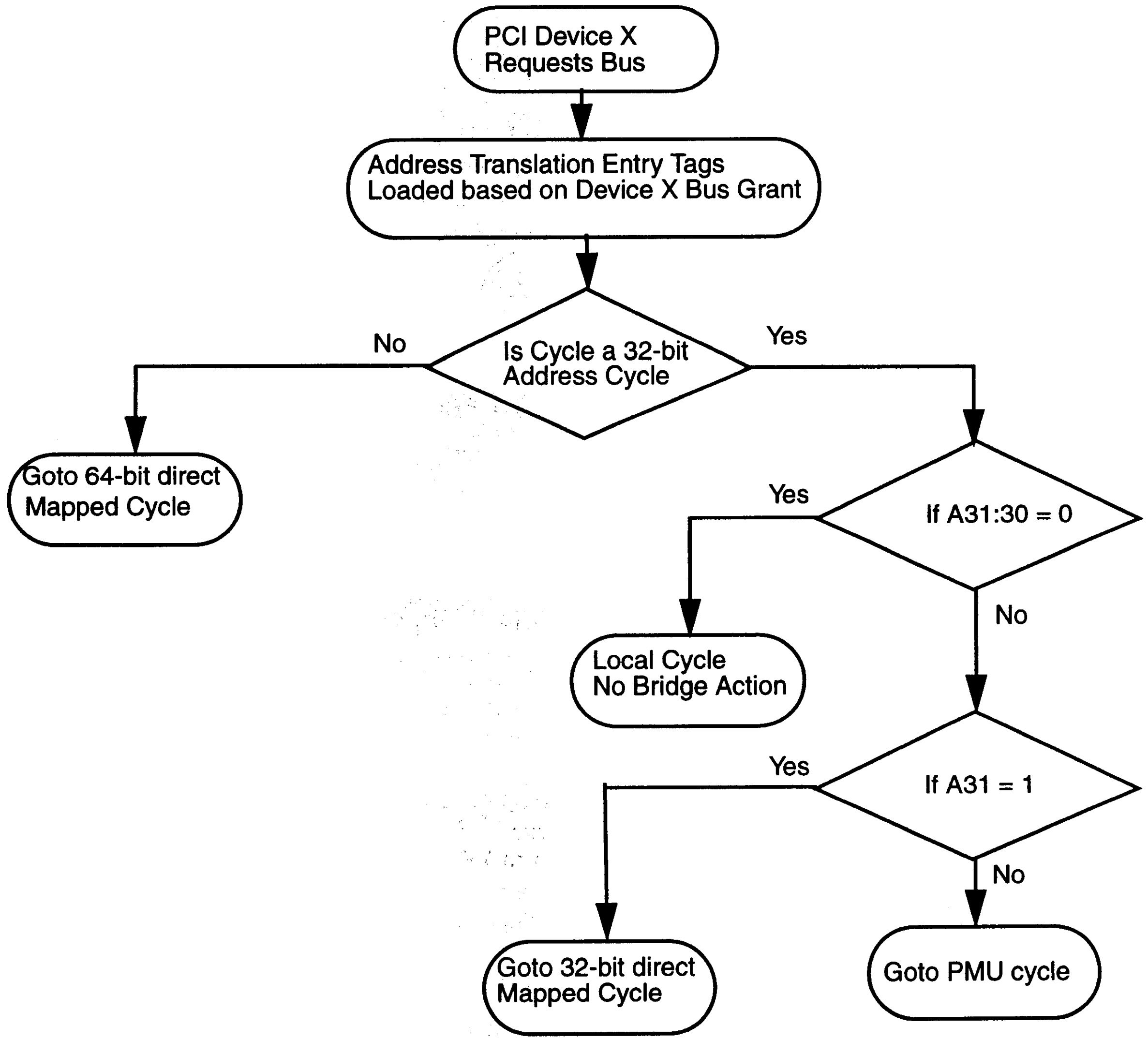

PCI Device X Requests Bus
Address Translation Entry Tags Loaded based on Device X Bus Grant
Is Cycle a 32-bit Address Cycle
No
Yes
Goto 64-bit direct Mapped Cycle
If A31:30 = 0
Yes
No
Local Cycle No Bridge Action
If A31 = 1
Yes
No
Goto 32-bit direct Mapped Cycle
Goto PMU cycle

Figure 21 PCI Address Translation Flow Chart 1

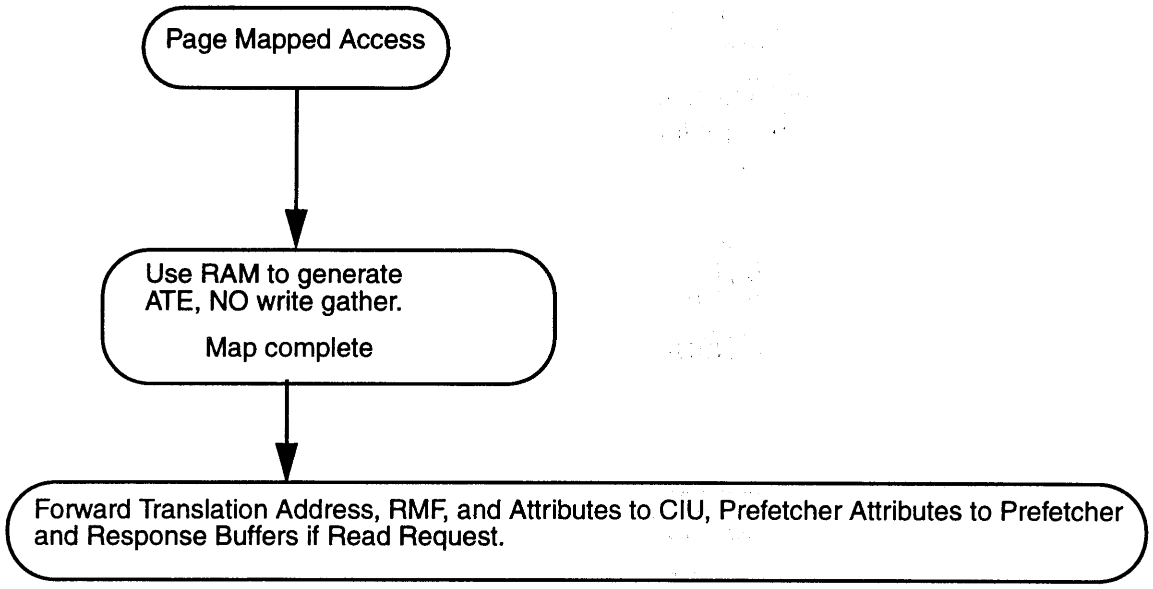

Page Mapped Access
Use RAM to generate ATE, NO write gather.
Map complete
Forward Translation Address, RMF, and Attributes to CIU, Prefetcher Attributes to Prefetcher and Response Buffers if Read Request.

Figure 22 PMU Address Translation Flow Chart

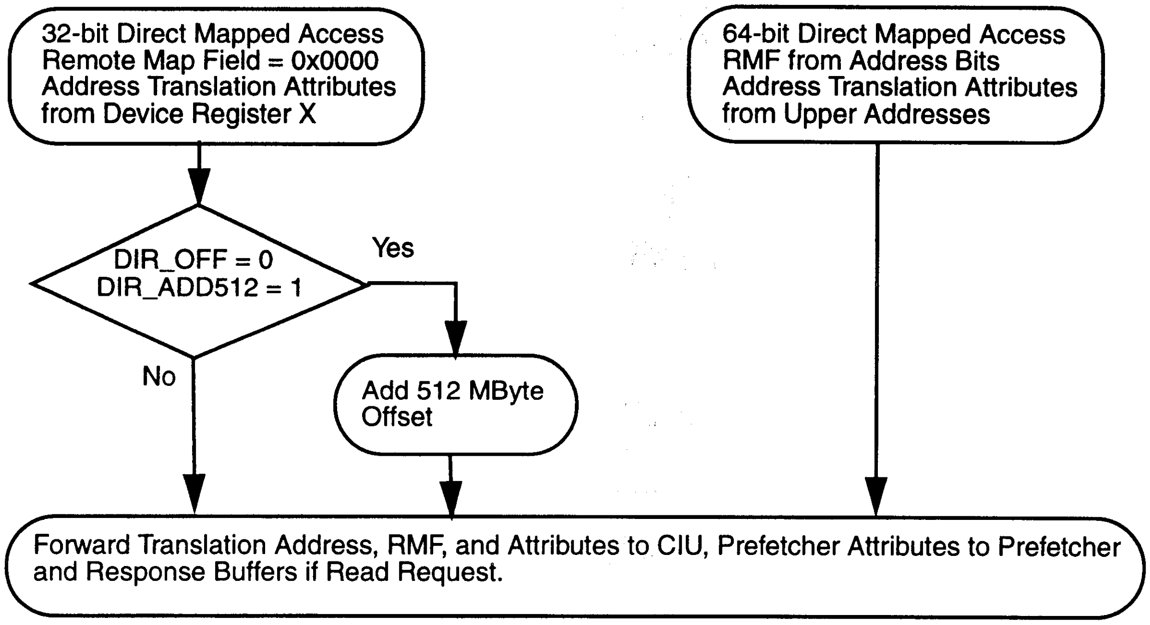

32-bit Direct Mapped Access
Remote Map Field = 0x0000
Address Translation Attributes from Device Register X
64-bit Direct Mapped Access
RMF from Address Bits
Address Translation Attributes from Upper Addresses
DIR_OFF = 0
DIR_ADD512 = 1
Yes
No
Add 512 MByte Offset
Forward Translation Address, RMF, and Attributes to CIU, Prefetcher Attributes to Prefetcher and Response Buffers if Read Request.

Figure 23 Direct Address Translation Flow Chart

CHAPTER 8 PCI Interface Unit and It's Operation

PCI Interface Unit interfaces between PCI Bus and PCI 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 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 93**.

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 93 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 command. 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 24. 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 validated 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].

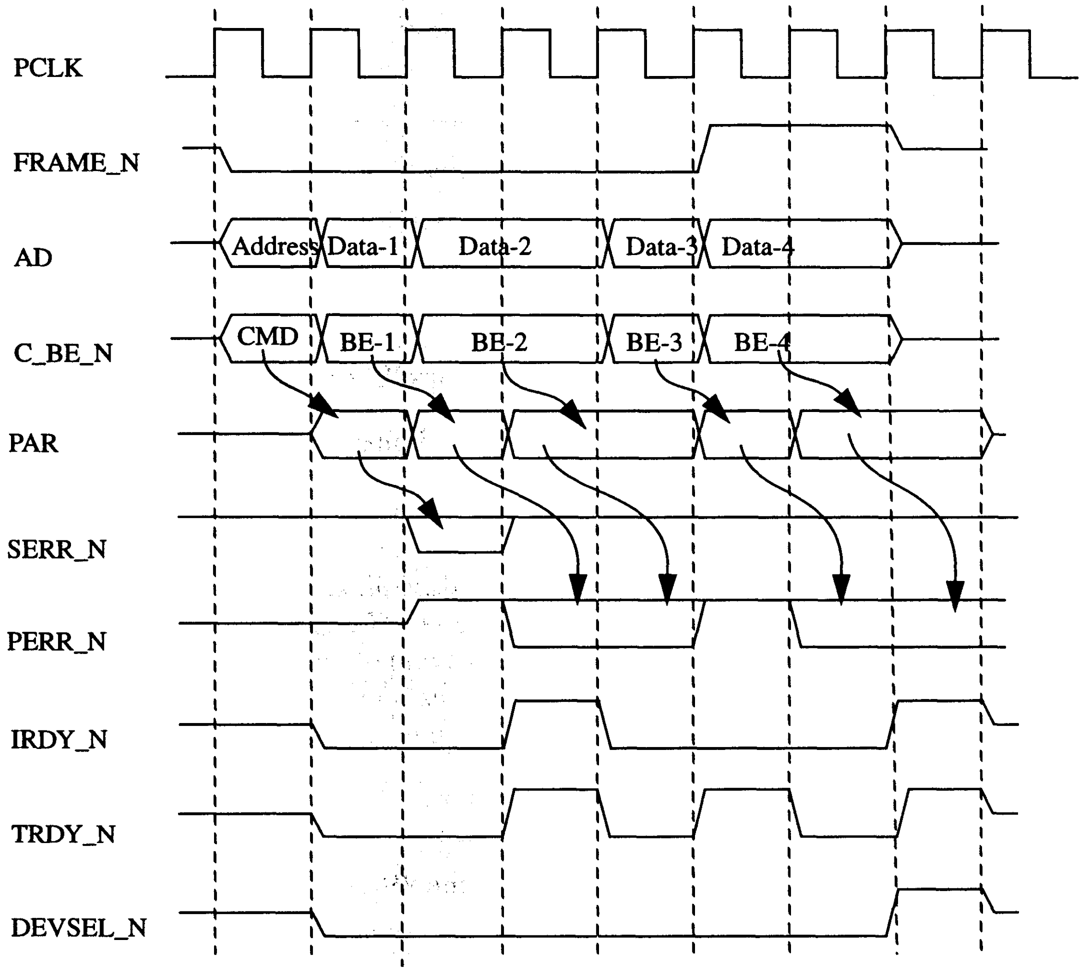


Figure 24 PCI Write (32-bit address)

8.1.2.2 Read Transaction

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.

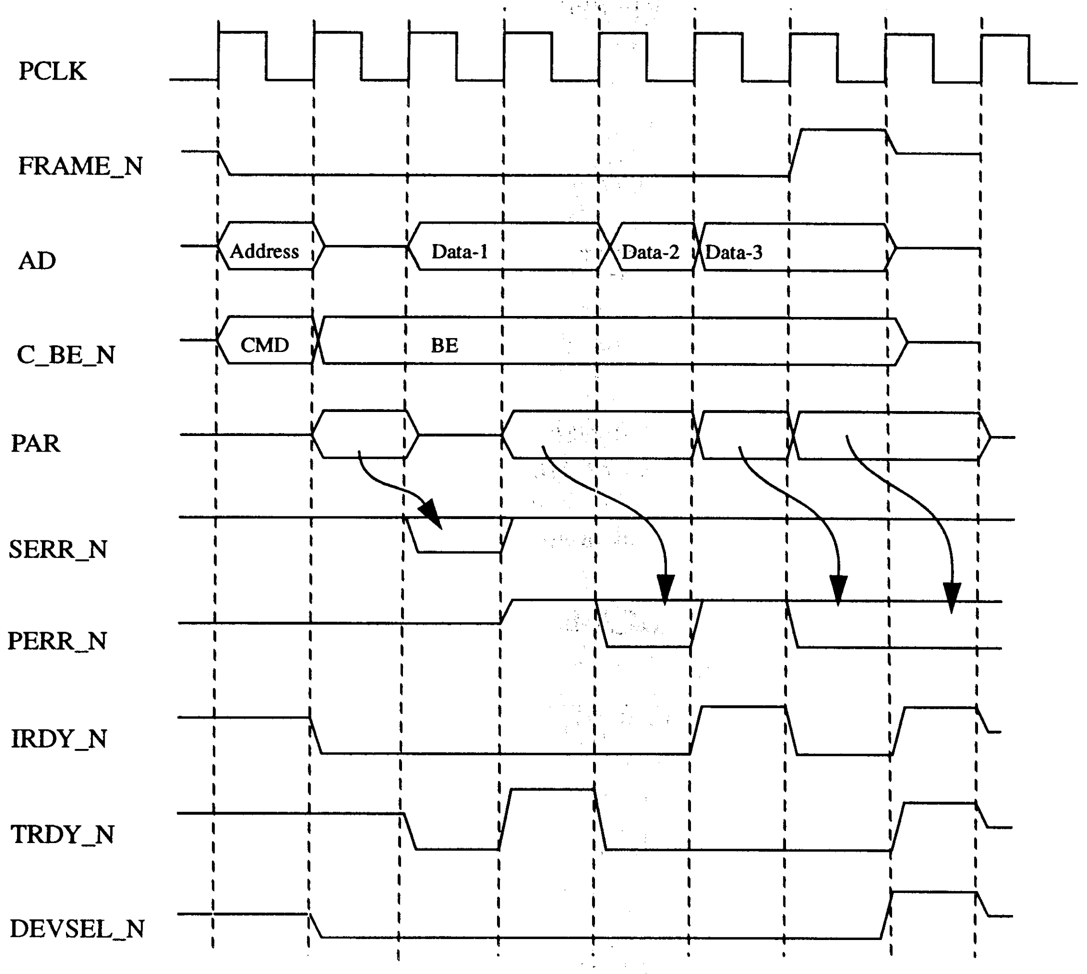


Figure 25 PCI Read (32-bit address)

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 termination 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.

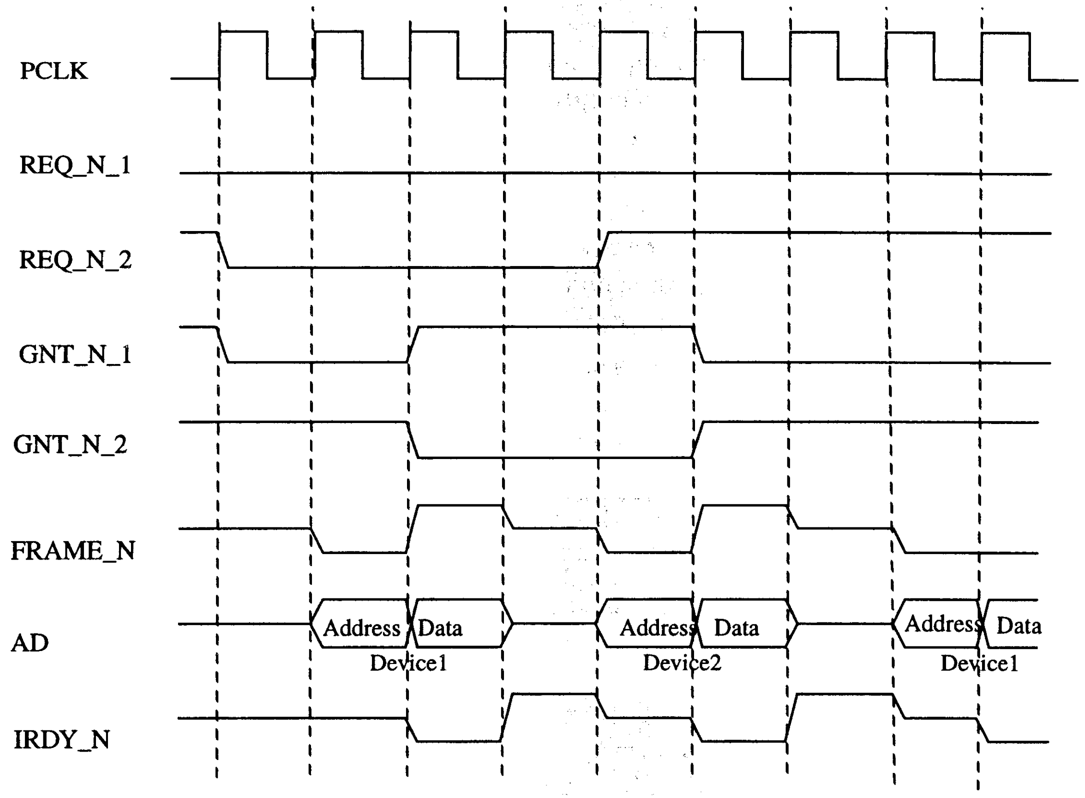


Figure 26 PCI Arbitration

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 support: 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.

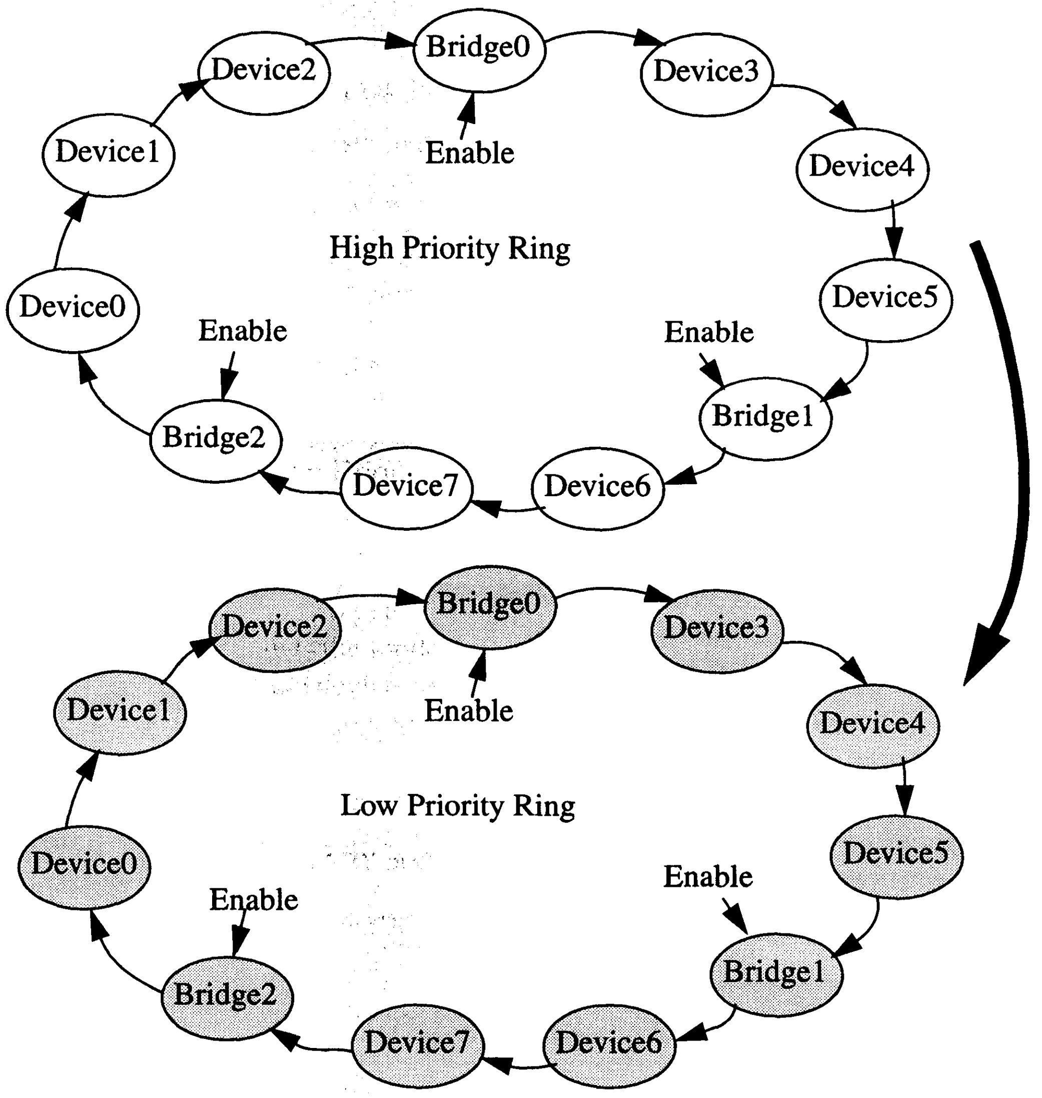


Figure 27 Arbitration Priority Rings

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.

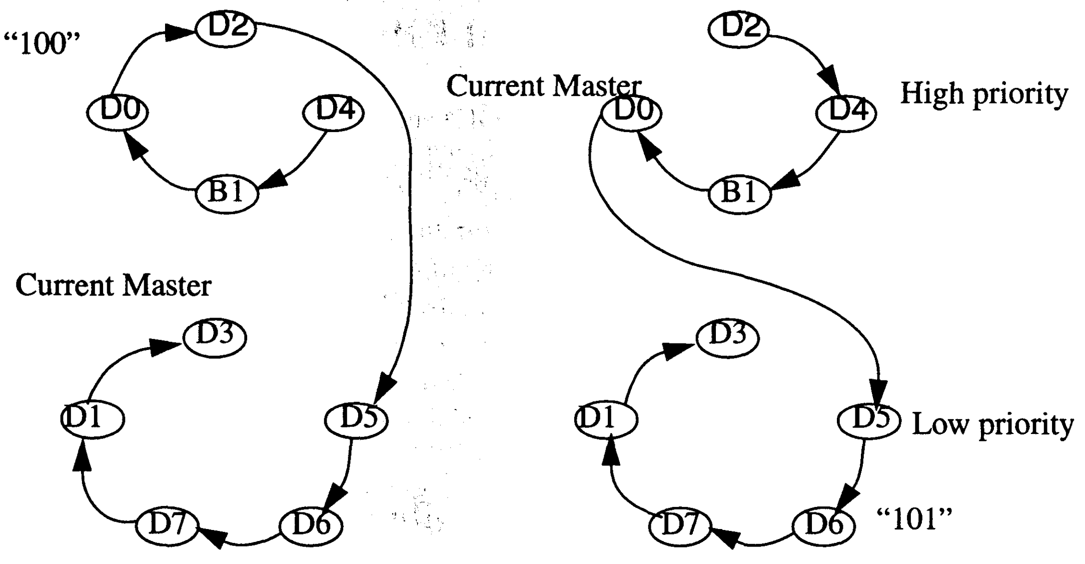


Figure 28 Arbitration Examples

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 chapter 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 94 **Configuration ID Select Lines**

CHAPTER 9

PCI Bridge Subsection Data Buffers

9.1 Data Buffer Overview

The Bridge subsection of the XBridge 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 bus (dma mode). An example of these operation would be a SCSI DMA.

The buffering in the Bridge subsection of the XBridge ASIC allows peak performance on both the PCI 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.

9.2 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 subsection of the

XBridge 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 automatically handle the packing and proper credit management of smaller packets. If the operation is to the PCI bus, the ping-pong buffer provides an additional two operation buffer. If the operation requires a response, the PCI bus has a ping-pong response buffer.

9.3 PCI Master Buffers

There are two types of PCI master buffers used in the Bridge subsection of the XBridge 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.

9.3.1 Read Response Buffers

The Bridge subsection of the XBridge 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 subsection of the XBridge 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 subsection of the XBridge 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.

9.3.1.1 Precise Operations

A precise PCI read operation causes the Bridge subsection of the XBridge 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 subsection of the XBridge ASIC will continue to issue retries until the response arrives and then the next read operation will provide the requested data. The Bridge subsection of the XBridge 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 9.3.1.4)

9.3.1.2 Non-Precise Operations

A non-precise PCI read operation behaves like the precise transaction except that the Bridge subsection of the XBridge 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 9.3.1.4)

9.3.1.3 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 9.3.1.4)

9.3.1.4 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.

9.3.1.5 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 subsection of the XBridge 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”.

9.3.2 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 subsection of the XBridge 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 10 PCI Interrupts

10.1 Bridge Interrupt Introduction

Bridge Subsection can either accept interrupt requests from the 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.

10.2 Interrupt Operations

The Bridge only supports level triggered interrupts shown in Figure 29. 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 Bus clock at the PCI 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 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.

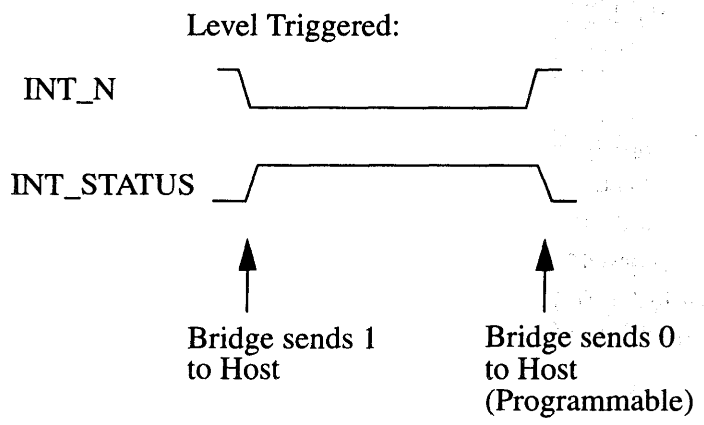


Figure 29 Level Triggered Interrupts

CHAPTER 11

PCI Bridge Subsection Error Cases

The Bridge subsection of the XBridge 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.

11.1 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.

11.1.1 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 subsection of the XBridge 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 subsection of the XBridge 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 subsection of the XBridge 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 95 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 95 Response Packet Error Conditions

11.1.2 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 96 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 96 Request Packet Error Conditions

11.1.3 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 97 Receive Link Errors

11.2 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.

11.2.1 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 98 Receive Link Errors

11.3 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.

11.4 PCI Errors

11.4.1 *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 99 **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 99 PCI Master Errors

11.4.2 *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 100 PCI Slave Errors

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 *XBridge* ASIC supports the PCI bus which uses Little Endian address mode. The system can be of either addressing mode as well and therefore swapping may be desirable.

The *XBridge* ASIC provides byte swapping between the *Crosstalk* Interconnect and the PCI bus. Byte swapping can be enabled per device for *Crosstalk* to PCI accesses with the device registers. Byte swapping can be enabled for PCI 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 *XBridge* 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 101 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
	A	B						0x1	0x0		B	A	

Table 102 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					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 102 **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 103 **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 104 **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 105 **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 106 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 107 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 108 **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 109 **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 110 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 111 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 112 **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 113 **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 114 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 115 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 116 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 117 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 118 **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 119 **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 120 **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 121 **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 122 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 123 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 124 **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 125 **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 126 **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 126 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 127 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 128 **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 129 **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 130 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 131 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 132 Double Word Data Little Endian SysAD to PCI-64 No Swap

APPENDIX B Revision Log

B.1 Rev1.0

APPENDIX C

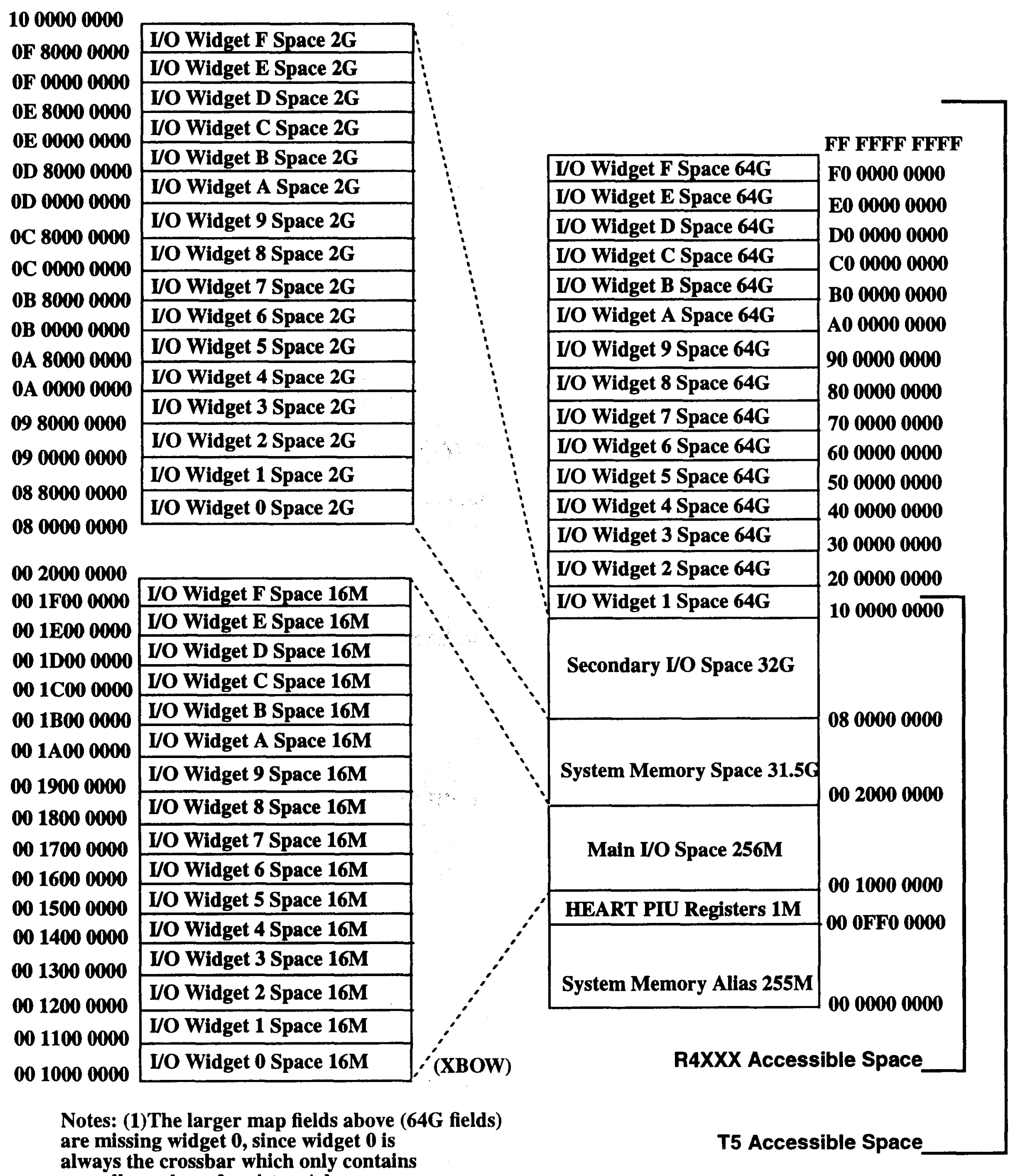
Software Notes and Restrictions

C.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 30 contains the processor address maps for the different ranges. Table 133contains 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 30 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 133 **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 134 **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 134 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 134 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 134 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 134 Address Map From Base Address

11.5 Current Anomalies in Rev1 Si

- Symptom: