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.

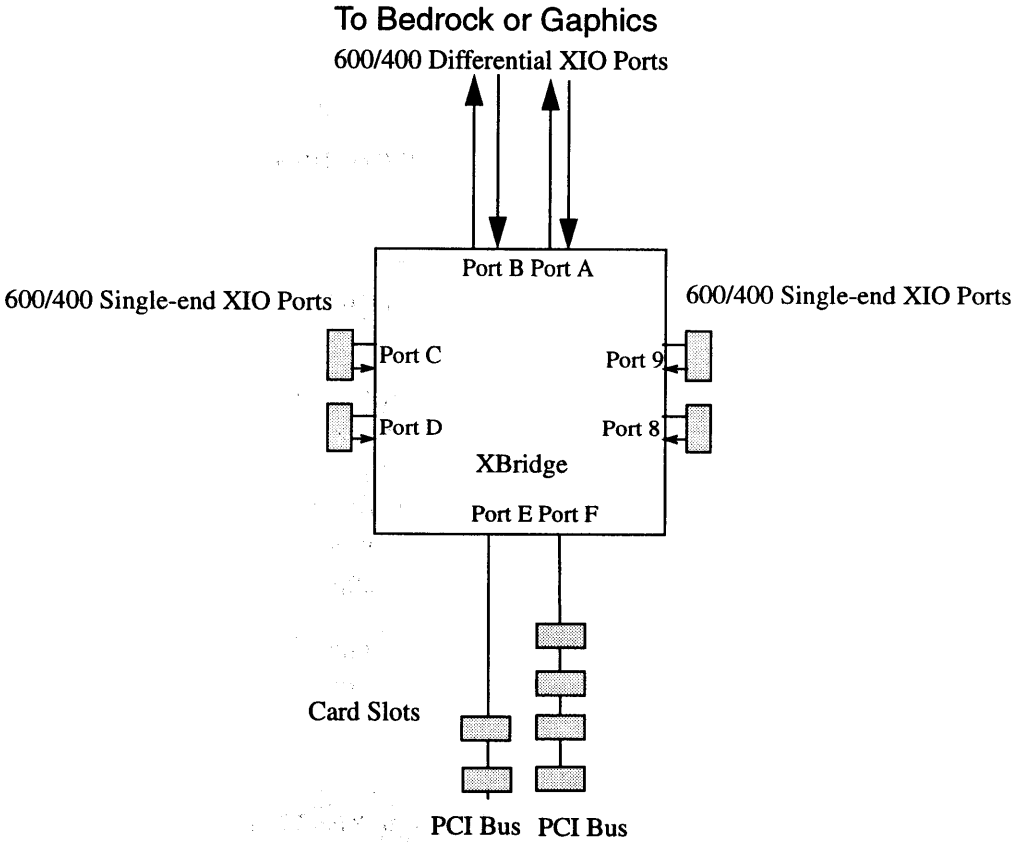

Figure 1 XBridge System Block Diagram

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.

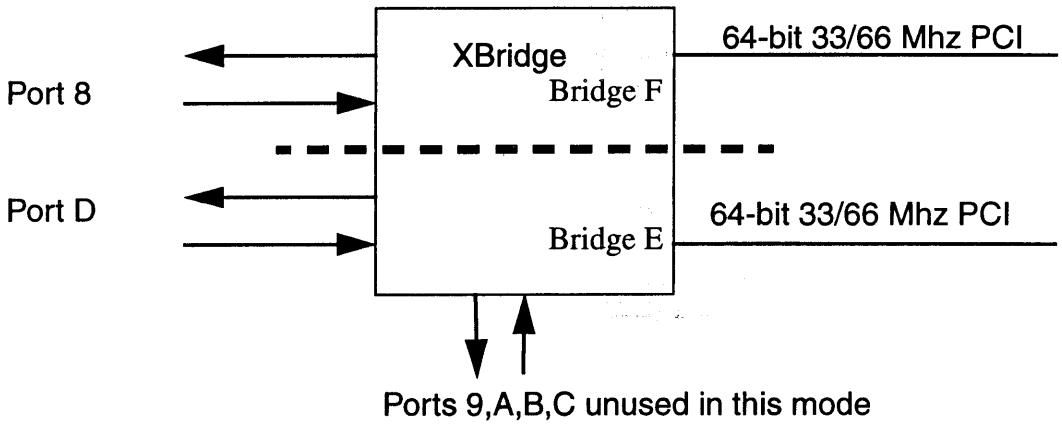

Figure 2 Super Bridge Mode Block Diagram

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.

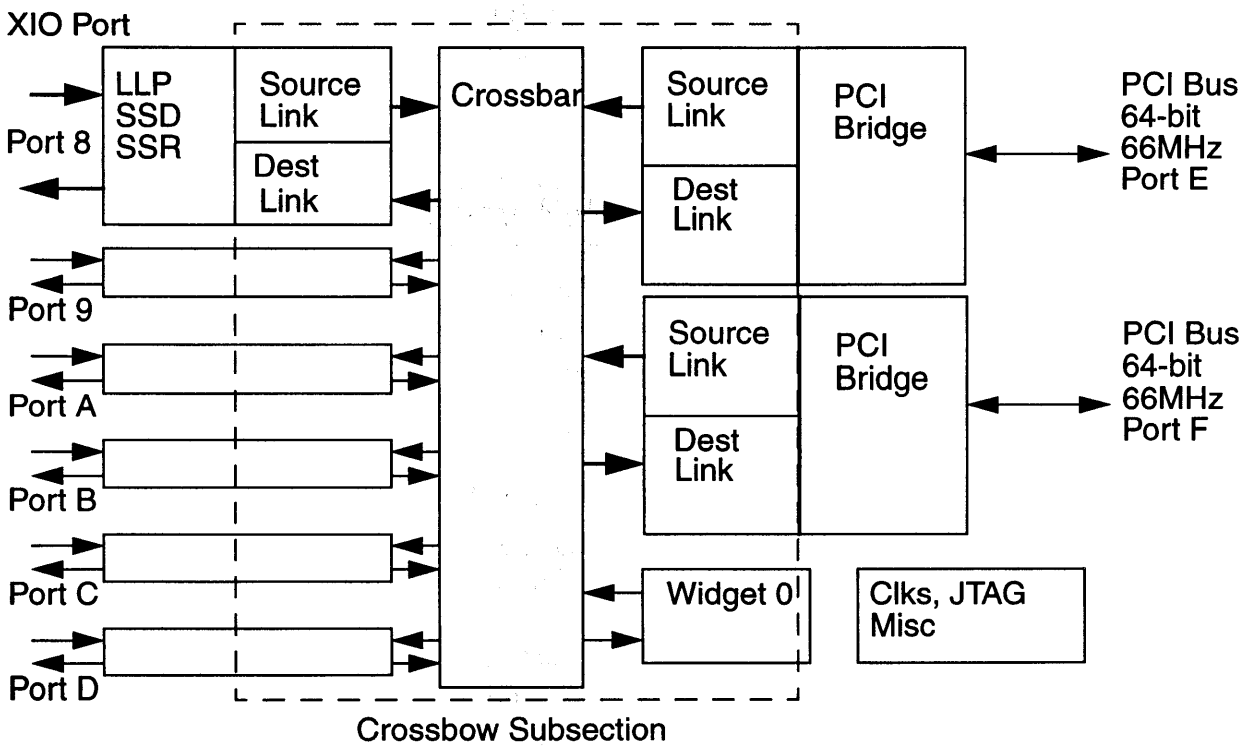

Figure 3 XBridge Block Diagram

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 ar-

biter 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.

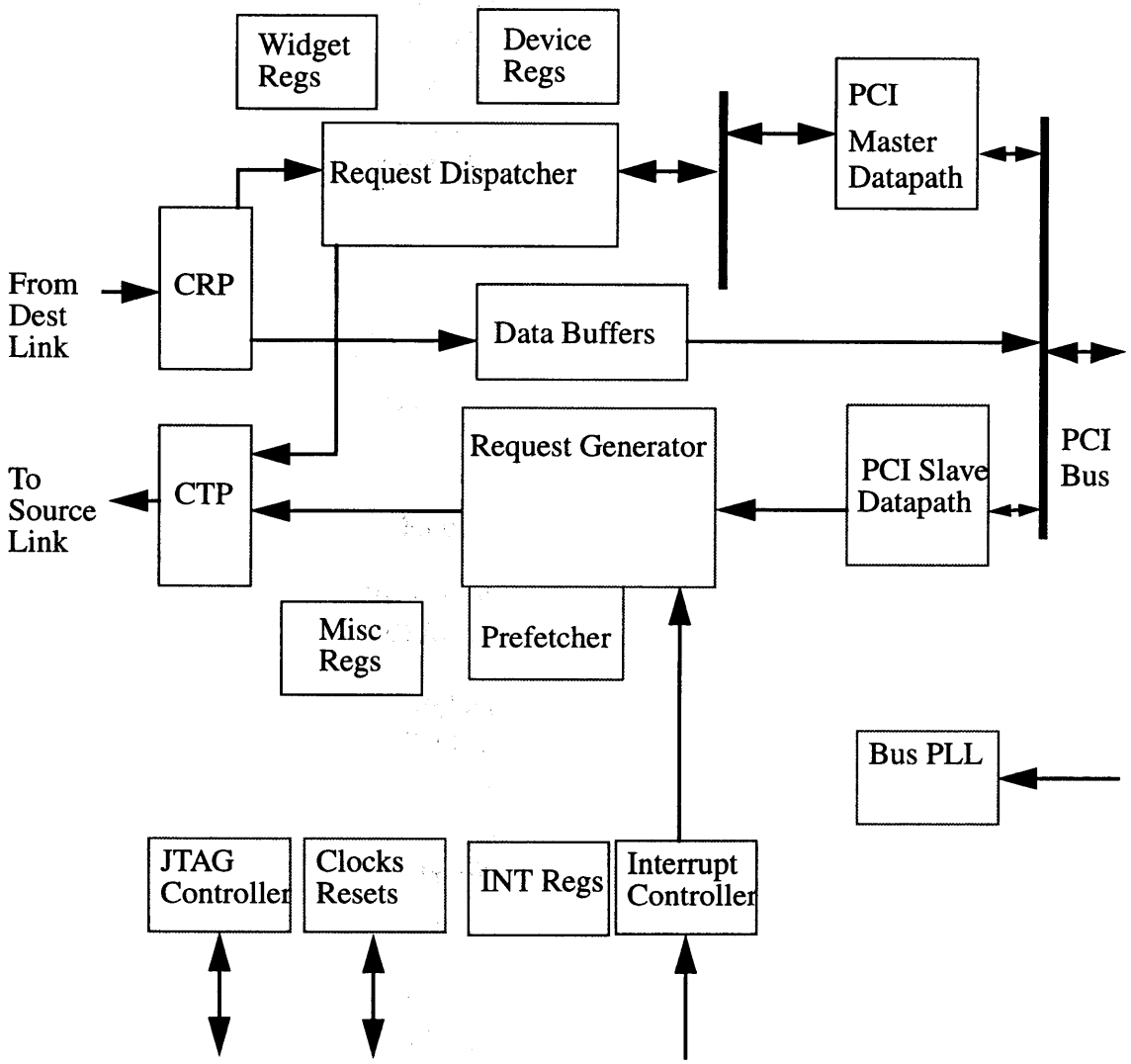

Figure 4 *XBridge* Bridge Subsection Block Diagram

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