Product Not Recommended for New Designs

Mesh Fabric Reference Design Application Note

XAPP698 (v1.2) February 15, 2005

XILINX®

Mesh Fabric Reference Design Application Note
XAPP698 (v1.2) February 15, 2005

The following table shows the revision history for this document..

	Version	Revision
03/05/04	1.0	Initial Xilinx release.
03/11/04	1.1	Added "Backplane Interface" section.
02/15/05	1.2	Revised Figure 4-2, Figure 5-3, and Figure 5-4; made other minor changes throughout.

Table of Contents

Schedule of Figures 7

Schedule of Tables 9

Preface: About This Guide

Guide Contents 11
Additional Resources 11
Conventions 12
Typographical 12
Online Document 13

Chapter 1: Introduction

General Description 15
Feature Summary 17

Chapter 2: Architecture

Overview 19
Block Level Descriptions 20
Ingress System Interface (itm_if) 22
Input Registers 22
Output Registers 23
Destination Lookup 23
Multicast Lookup 23
Ingress Buffering (itm_fifo) 24
FIFO Depth Control 26
Local Flow Control 26
Cell Transmit (cell_tx) 26
Cell Receive (cell_rx) 27
To TM or Upstream Flow Control Generation (itm_fc, itm_fifo, egress_sched, cell_rx) 27
Output Queueing (egress_fifo, egress_q, egress_sched) 28
Memory Access Mux (egress_fifo) 31
Output Buffer Controller (egress_sched) 32
Output Queue Controller (egress_q). 32
Fabric Flow Control Generation (egress_fc) 33
Priority Scheduler (egress_sched) 33
Port Scheduler (egress_sched) 34

To Fabric or Downstream Flow Control Generation (etm_if, egress_sched, cell_rx) 34
Egress System Interface (egress_if) 34
Input Registers 34
Output Registers 34
Egress Multiplexing 34
System Interfaces and Signal Descriptions 35
Cascade Interfaces 35
Egress TM Interface 37
Ingress TM Interface 37
SERDES Interface 37
Management Interface 38
System Signal Interface 38
Backplane Interface 39

Chapter 3: Implementation

Memory Map 41
Accessing the MFRD Register Space 42
Accessible Register Descriptions 42
Read Address Register (RAReg) 42
Write Address Register (WAReg) 42
Read Data Register (RDReg) 43
Write Data Register (WDReg) 43
Control & Status Register (CSReg) 43
Auto-Address Operation 44
Internal MFRD Register Descriptions 45
General Configuration Registers 46
Ver_Reg 46
Config_Reg_1 46
Config_Reg_2 47
Port_Sel_Reg 47
DCM_Rst_Reg 48
DCM_Shift_Reg 49
DCM_Shift_Dir 49
FIFO_Grp_Tbl_Sel 49
Egrs_Thresh_Reg 50
Egrs_FIFO_En 51
Dwn_strm_Port_Cnt 52
General Status Registers 52
Full-Mesh Status Register (Status_Reg) 52
Error Status Register (Error_Stat) 53
Device Structure Summary (Dev_Struct) 53

SERDES Configuration Registers 54
SERDES_Config 54
SERDES Status Register 54
Egress Buffer Management 55
Buff_Ent_Lst 56
Free_Head_Ptr 56
Free_Tail_Ptr 56
Free_Depth 56
Free_Dph_Thres_Tbl 57
Free_Dph_Msk_Tbl 57
Group ID Mapping Tables 57
ID_Map_Tbl 57

Egress Buffer Control Initialization 58
Parameterization 59
Implementation Size Based on SERDES and FPGA Selection 63
Design Tools 64

Chapter 4: Performance Characteristics

Performance 65
Clocking 68

Chapter 5: MFRD Demo

Demo Components 69
Demo Configuration 70
PPC System 71
Local Link FIFOs 71
Local Link Shim 71
DCR Modules 74
Pin Descriptions for the Demo System 77
Demo Operation 81
Demo Reference Files 84

Glossary 85

Schedule of Figures

Chapter 1: Introduction

Figure 1-1: **Mesh Topology** 15
Figure 1-2: **Cascading Multiple FPGAs** 16

Chapter 2: Architecture

Figure 2-1: **Reference Design Block Diagram** 20
Figure 2-2: **Data and Flow Control Paths** 21
Figure 2-3: **itm_if** 22
Figure 2-4: **System Interface Input Regs** 23
Figure 2-5: **Group Map Lookup** 24
Figure 2-6: **Ingress Buffer** 25
Figure 2-7: **cell_rx and cell_tx** 27
Figure 2-8: **ITM Flow Control** 28
Figure 2-9: **Output Queueing** 29
Figure 2-10: **Egress Memory Buffer and Datapath** 30
Figure 2-11: **Memory Access Mux** 31
Figure 2-12: **Egress Flow Control Thresholds** 32
Figure 2-13: **Cascade Interface Cell Header Format** 35
Figure 2-14: **Cascade Interface Connections** 36

Chapter 3: Implementation

Figure 3-1: **Write with Auto-Increment** 44
Figure 3-2: **Read with Auto-Increment** 45

Chapter 4: Performance Characteristics

Figure 4-1: **Data Path through Mesh Fabric Reference Design** 65
Figure 4-2: **Mesh Switch Clocking Scheme** 68

Chapter 5: MFRD Demo

Figure 5-1: **Mesh Fabric System Block Diagram** 69
Figure 5-2: **Timestamp Details** 72
Figure 5-3: **CPU Communication between Upstream and Downstream FPGAs (Write)** 75
Figure 5-4: **CPU Communication between Upstream and Downstream FPGAs (Read)** 76
Figure 5-5: **Initialization Menu** 81
Figure 5-6: **Demo Menu** 82
Figure 5-7: **Demo Results** 83
Figure 5-8: **Debug Menu** 84

Schedule of Tables

Chapter 1: Introduction

Chapter 2: Architecture

Table 2-1: **Ingress Buffer Capacity versus Cell Payload Size** 26
Table 2-2: **Egress Buffering Block RAM Allocation** 29
Table 2-3: **Cascade Interface Signals** 36
Table 2-4: **Egress TM Interface Signals** 37
Table 2-5: **Ingress TM Interface Signals** 37
Table 2-6: **SERDES Interface Signals** 37
Table 2-7: **DCR Interface Pins** 38
Table 2-8: **System Signal** 38
Table 2-9: **Backplane Cell Header Format** 39

Chapter 3: Implementation

Table 3-1: **MFRD Memory Map** 41
Table 3-2: **Internal Control Register Summary** 45
Table 3-3: **SERDES to FIFO Mapping** 50
Table 3-4: **Egress Memory and Width Based on Device** 58
Table 3-5: **Reference Design Parameters** 59
Table 3-6: **Maximum FFs, LUTs, BRAMs, and MGTs Based on SERDES and FPGA** . . . 63
Table 3-7: **Block by Block Utilization** 63

Chapter 4: Performance Characteristics

Table 4-1: **Total User and SERDES Clocks** 66
Table 4-2: **SERDES Clock Domain Timing** 66
Table 4-3: **User Clock Domain Timing** 66
Table 4-4: **Observed Upstream and Downstream Loopback Values** 67

Chapter 5: MFRD Demo

Table 5-1: **Shim DCR Registers** 72
Table 5-2: **CON Bit Assignments** 74
Table 5-3: **MFRD Reference Design Pinout** 77
Table 5-4: **Control Word Organization** 82

About This Guide

The Xilinx Mesh Fabric Reference Design is a development vehicle for full mesh line cards based on Virtex™-II Pro devices. The design is a fully parameterized IP component that lets designers partition a mesh fabric design into any combination of Virtex-II Pro devices.

Guide Contents

This manual contains the following chapters:

- Chapter 1, "Introduction," provides an overview of the mesh fabric reference design along with its key features.
- Chapter 2, "Architecture," describes in detail the major interfaces and functional blocks of the reference design. Signal descriptions are also provided.
- Chapter 3, "Implementation," provides implementation-specific information on the reference design, including the memory map, register descriptions, and customizable parameters.
- Chapter 4, "Performance Characteristics," provides operational data for the mesh fabric reference design, including performance and clocking.
- Chapter 5, "MFRD Demo," describes the components, configuration, and operation of the Mesh Fabric reference design demonstration board.

Additional Resources

For additional information, go to http://support.xilinx.com. The following table lists some of the resources you can access from this website. You can also directly access these resources using the provided URLs.

Resource	Description/URL
Tutorials	Tutorials covering Xilinx design flows, from design entry to verification and debugging http://support.xilinx.com/support/techsup/tutorials/index.htm
Answer Browser	Database of Xilinx solution records http://support.xilinx.com/xlnx/xil_ans_browser.jsp
Application Notes	Descriptions of device-specific design techniques and approaches http://support.xilinx.com/apps/appsweb.htm

Resource	Description/URL
Data Sheets	Device-specific information on Xilinx device characteristics, including readback, boundary scan, configuration, length count, and debugging http://support.xilinx.com/xlnx/xweb/xil_publications_index.jsp
Problem Solvers	Interactive tools that allow you to troubleshoot your design issues http://support.xilinx.com/support/troubleshoot/psolvers.htm
Tech Tips	Latest news, design tips, and patch information for the Xilinx design environment http://www.support.xilinx.com/xlnx/xil_tt_home.jsp

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
`Courier font`	Messages, prompts, and program files that the system displays	`speed grade: - 100`
`Courier bold`	Literal commands that you enter in a syntactical statement	**`ngdbuild`** *`design_name`*
Helvetica bold	Commands that you select from a menu	**File →Open**
	Keyboard shortcuts	**Ctrl+C**
Italic font	Variables in a syntax statement for which you must supply values	**`ngdbuild`** *`design_name`*
	References to other manuals	See the *Development System Reference Guide* for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as **`bus[7:0]`**, they are required.	**`ngdbuild`** `[`*`option_name`*`]` *`design_name`*
Braces { }	A list of items from which you must choose one or more	**`lowpwr`** `={`**`on`**`|`**`off`**`}`

Convention	Meaning or Use	Example
Vertical bar |	Separates items in a list of choices	**`lowpwr ={on|off}`**
Vertical ellipsis . . .	Repetitive material that has been omitted	`IOB #1: Name = QOUT'` `IOB #2: Name = CLKIN'` . . .
Horizontal ellipsis . . .	Repetitive material that has been omitted	**`allow block`** *`block_name loc1 loc2 ... locn;`*

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Resources" for details. Refer to "Title Formats" in Chapter 1 for details.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the *Virtex-II Handbook.*
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

Chapter 1

Introduction

This chapter contains the following sections:

- "General Description"
- "Feature Summary"

General Description

Mesh switch fabrics are becoming popular with the emergence of inexpensive, high-speed serializers/deserializers (SERDES) as well as FPGA devices with embedded SERDES blocks. This document describes an implementation of a mesh switch fabric using Xilinx Virtex-II Pro devices.

Figure 1-1 illustrates the concept of a mesh switch. Essentially, it provides a private connection between any pair of modules where all modules are interconnected. In exchange, there is no central switch fabric.

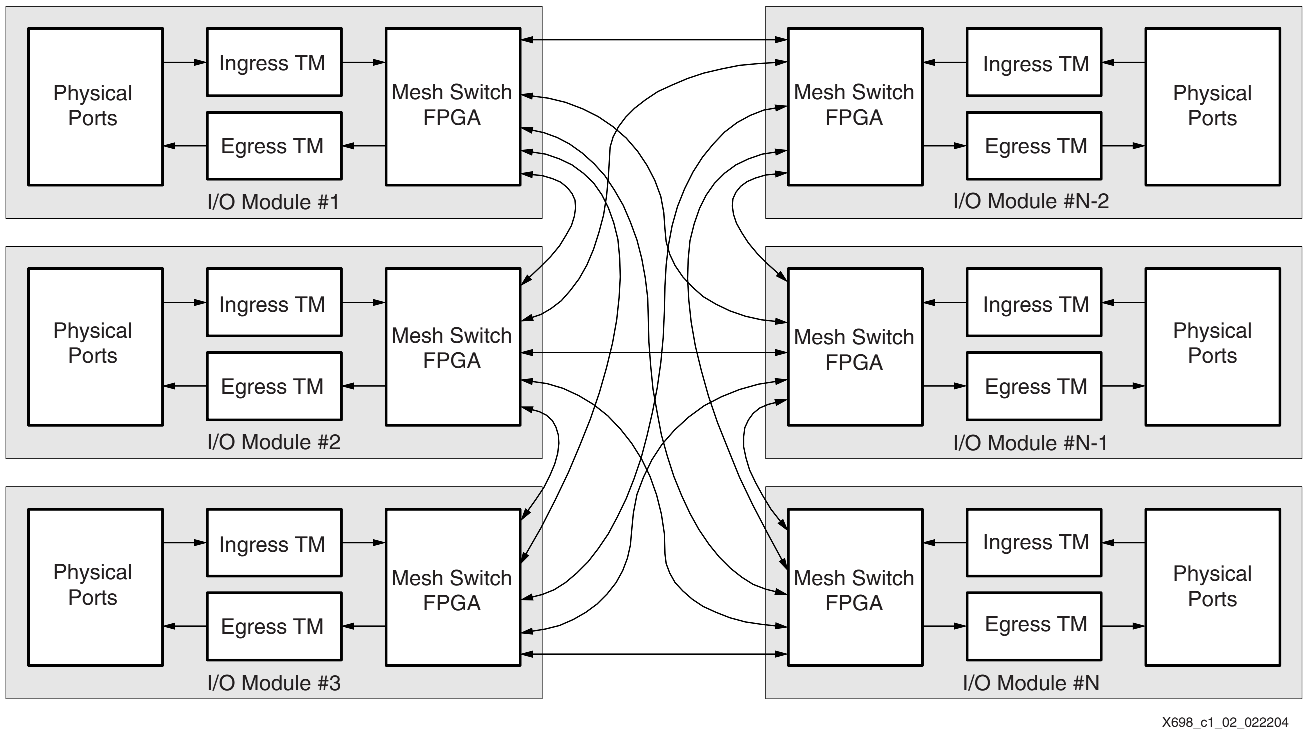


Figure 1-1: **Mesh Topology**

This reference design supports all device sizes from an XC2VP4 upward. No device requires any speed grade beyond the –5 (slowest available speed grade). All SERDES blocks in each device can be used to implement switch ports, although not all SERDES blocks need to be used if a custom application requires some available SERDES blocks for other functions. Devices can be cascaded to implement more switch ports. The configuration image for each device in the chain is identical (presuming all devices in the chain are the same). Different size devices can be cascaded with the obvious requirement of a unique image per device type. The last device in a cascade can have a different configuration from the other devices because its cascade pins are not used. Alternatively, if the last device has the same configuration, its pins must be handled appropriately on the board (tied off or left open).

To easily scale the ratio of logic to high-speed serial I/O, this reference design provides a way to cascade multiple Virtex-II Pro devices but present one interface to the traffic manager. Thus multiple, smaller FPGAs can be used to build up a system with many SERDES blocks, providing an overall cost savings compared with using a larger FPGA to get the required number of SERDES blocks.

For example, the XC2VP7 device is an extremely economical choice offering eight SERDES blocks. Applications needing 16 ports can cascade two XC2VP7 parts. A 12-port solution can mix a XC2VP7 together with a XC2VP4. Unique configuration images are required but the same HDL source can be compiled into either target.

Latency through the cascade path is limited to a small number of clock cycles. This is very important because the flow control budget must include the ripple delay through multiple parts.

Figure 1-2 illustrates the concept of cascading multiple FPGAs yet presenting a single interface to the traffic manager.

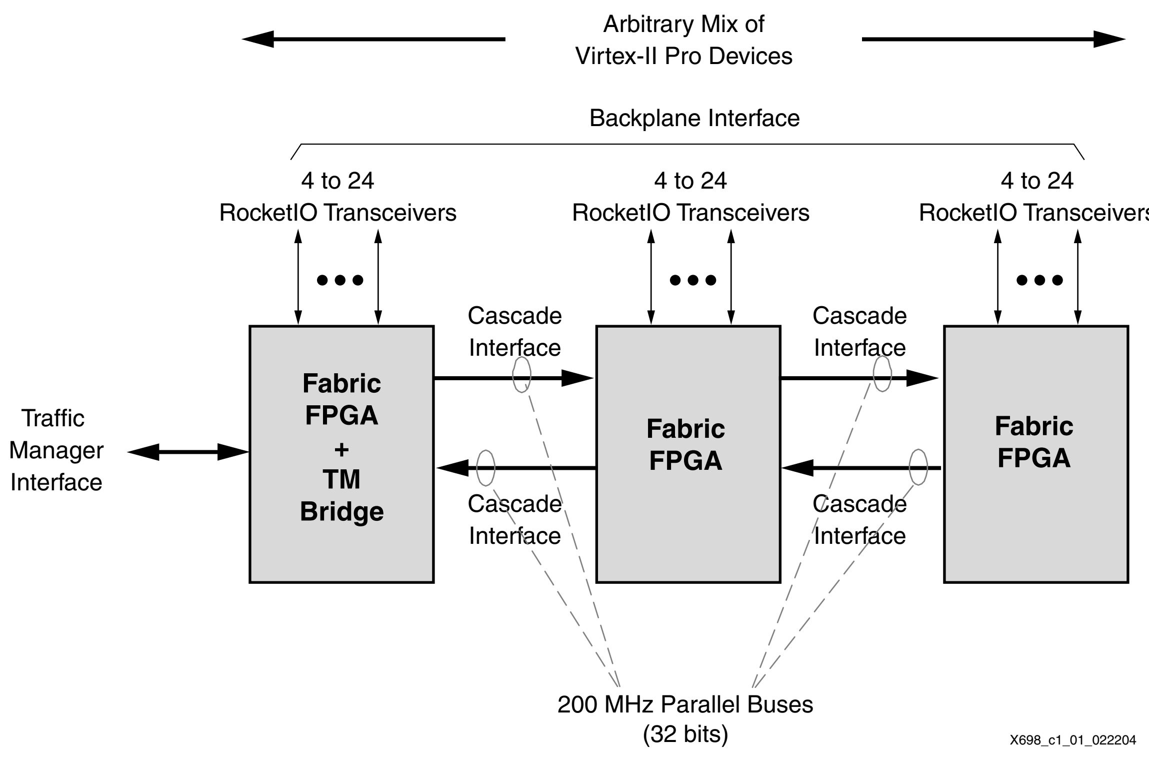

Arbitrary Mix of Virtex-II Pro Devices
Backplane Interface
4 to 24 RocketIO Transceivers
4 to 24 RocketIO Transceivers
4 to 24 RocketIO Transceivers
Traffic Manager Interface
Fabric FPGA + TM Bridge
Cascade Interface
Cascade Interface
Fabric FPGA
Cascade Interface
Cascade Interface
Fabric FPGA
200 MHz Parallel Buses (32 bits)
X698_c1_01_022204

Figure 1-2: **Cascading Multiple FPGAs**

To provide flexibility in quality of service, this reference design supports up to 16 traffic priorities. Each application can specify how many priorities to implement. Smaller devices might require fewer priorities because of block RAM (BRAM) limitations.

The device closest to the Traffic Manager (TM) typically includes logic to bridge the cascade interfaces to the interface used by the Traffic Manager. In this case, the configuration for the FPGA with the TM logic is different than the others in the cascade chain.

Feature Summary

The key features of the mesh fabric reference design are as follows:

- Support for 4 to 256 total switch ports
- Support for any Virtex-II Pro device from XC2VP4 to XC2VP100
- Fabrics can be built from any combination of Virtex-II Pro devices
- Backplane cell size is programmable from 48 to 136 bytes
- Flexible traffic scheduling with up to 16 traffic priorities, with Weighted Round Robin (WRR) between Egress FIFOs and strict priority scheduling between priorities on egress
- 32-bit Cascade interface
- No external memory required
- Multicast priority support
- Multiple parts of any size can be cascaded to achieve desired port count
- Loosely formatted system channel can be tailored to each application easily
- Out-of-band flow control

Chapter 2

Architecture

This chapter contains the following sections:

- "Overview"
- "Block Level Descriptions"
- "System Interfaces and Signal Descriptions"

Overview

Figure 2-1 shows the overall architecture of the reference design. The major functional blocks that make up the design can be broken up into four groups:

- **Ingress Datapath**
  These blocks enqueue cells to be forwarded to other line cards.
- **Egress Datapath**
  These blocks buffer and schedule traffic arriving from other line cards.
- **Backplane Interface**
  These blocks handle the transmission and reception of data across the backplane links.
- **Management**
  This block implements the interface used to access internal control and status registers. The reference design implements the DCR bus. Customers can customize this block to connect to different interfaces.

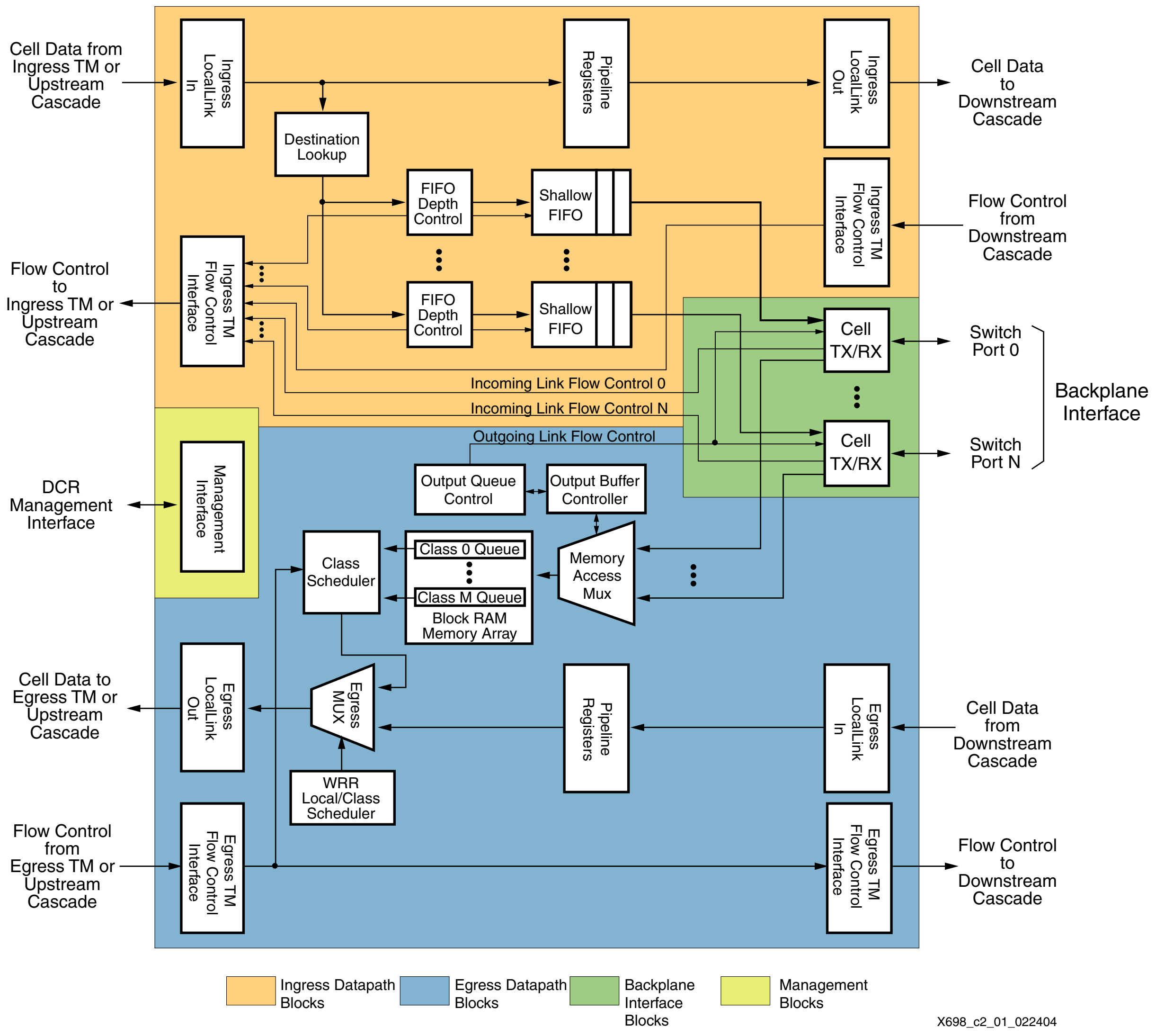

Cell Data from Ingress TM or Upstream Cascade
Ingress LocalLink In
Destination Lookup
Pipeline Registers
Ingress LocalLink Out
Cell Data to Downstream Cascade
FIFO Depth Control
Shallow FIFO
Ingress TM Flow Control Interface
Flow Control from Downstream Cascade
Flow Control to Ingress TM or Upstream Cascade
Cell TX/RX
Switch Port 0
Backplane Interface
Incoming Link Flow Control 0
Incoming Link Flow Control N
Outgoing Link Flow Control
Switch Port N
DCR Management Interface
Management Interface
Output Queue Control
Output Buffer Controller
Class Scheduler
Class 0 Queue
Class M Queue
Block RAM Memory Array
Memory Access Mux
Cell Data to Egress TM or Upstream Cascade
Egress LocalLink Out
Egress MUX
WRR Local/Class Scheduler
Egress LocalLink In
Cell Data from Downstream Cascade
Flow Control from Egress TM or Upstream Cascade
Egress TM Flow Control Interface
Flow Control to Downstream Cascade
Ingress Datapath Blocks
Egress Datapath Blocks
Backplane Interface Blocks
Management Blocks
X698_c2_01_022404

Figure 2-1: **Reference Design Block Diagram**

Block Level Descriptions

Figure 2-2 shows the data and control paths for the reference design.

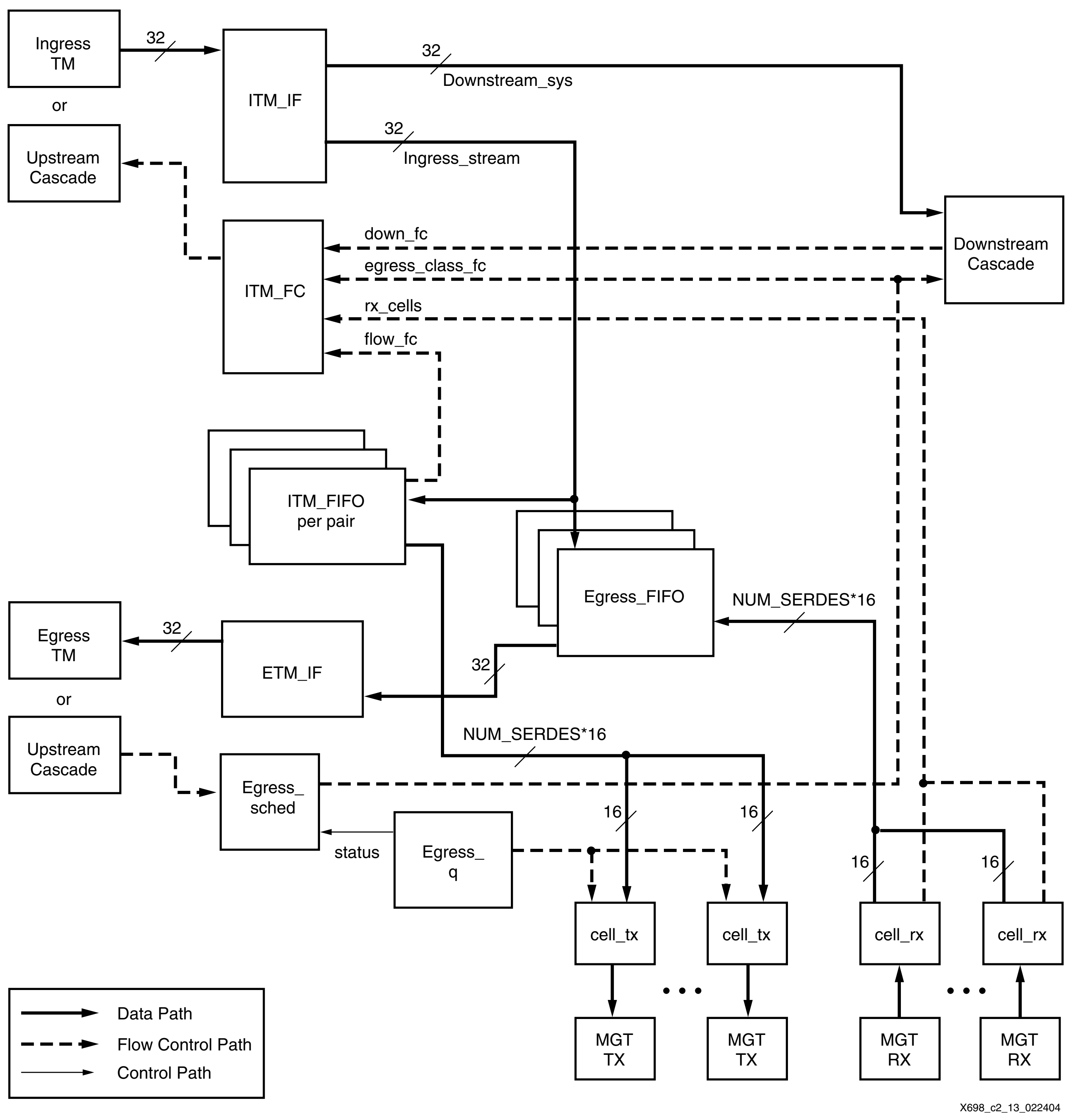


Figure 2-2: **Data and Flow Control Paths**

The following subsections provide details on specific mesh switch functional blocks. The entity names from the HDL source code are indicated in each subsection. The interfaces to connecting blocks are illustrated.

- "Ingress System Interface (itm_if)"
- "Ingress Buffering (itm_fifo)"
- "Cell Transmit (cell_tx)"

- "Cell Receive (cell_rx)"
- "To TM or Upstream Flow Control Generation (itm_fc, itm_fifo, egress_sched, cell_rx)"
- "Output Queueing (egress_fifo, egress_q, egress_sched)"
- "Priority Scheduler (egress_sched)"
- "Port Scheduler (egress_sched)"
- "To Fabric or Downstream Flow Control Generation (etm_if, egress_sched, cell_rx)"
- "Egress System Interface (egress_if)"

Ingress System Interface (itm_if)

The itm_if interface provides the downstream Cascade interface and lookup functions for itm_fifo blocks. Cells arrive on this interface from either a TM or an upstream FPGA, and are sent out a particular SERDES block or set of SERDES blocks.

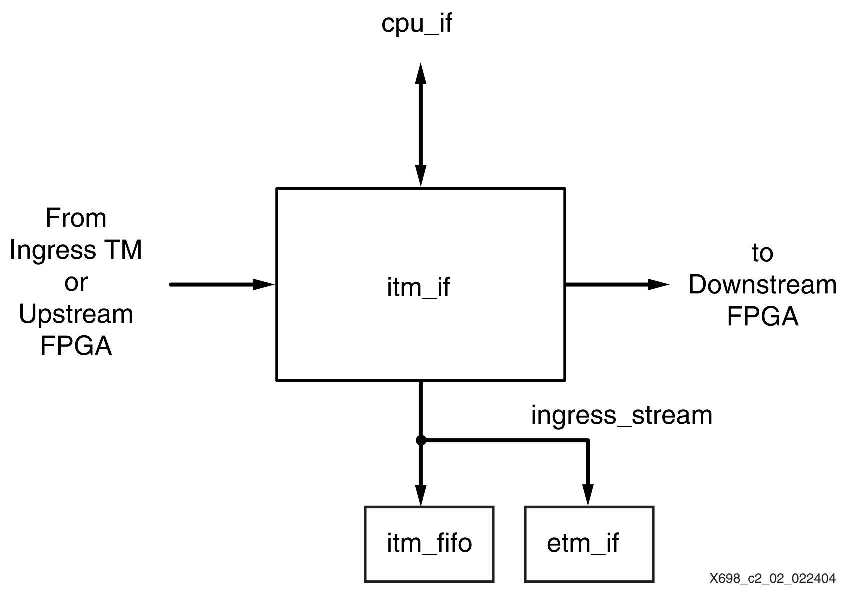


Figure 2-3: **itm_if**

Input Registers

The input data and SOC from the ingress traffic manager (or upstream cascade) can be DDR or SDR. When DDR is used, a DCM phase-shifts the incoming clock and doubles it for use inside the FPGA. This clock is used for the entire ingress data path, all the way to the ITM FIFO. It also is the downstream cascade data path clock. When SDR is used for the incoming data, the DCM is removed. Refer to Figure 2-4.

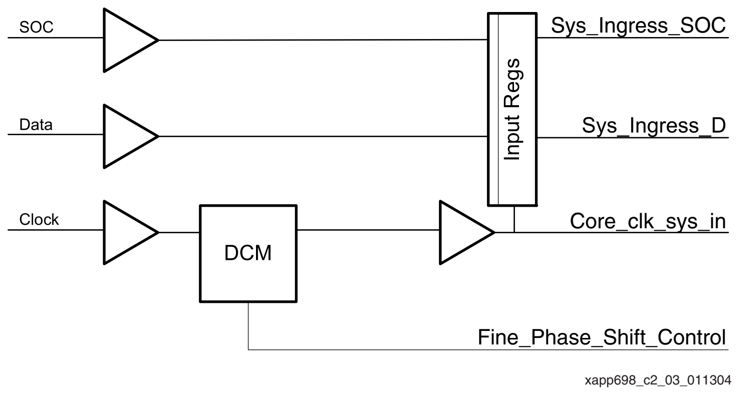


Figure 2-4: **System Interface Input Regs**

The DCM used in the DDR data recovery supports dynamic phase shifting in order to reliably sample the data. The end-user is expected to set the default value to a setting that works in that system after the interface has been tuned once.

As many stages of pipelining as are required can be inserted between the input registers and cell writing logic. It is also reasonable to insert register stages as the input data propagates further through the part. In other words, two SERDES pairs can tap off immediately while two SERDES pairs tap off after another register. Care must be taken because this delay affects the flow-control budget. Some floorplanning must be performed to assist timing closure with as few pipeline stages as possible.

Output Registers

This block echoes everything received from the input pins back to the cascade output registers. The clock output and the data must come from flip-flops to match timing. There is no requirement to match the clock phase between input and output. The output registers must tap from the input after as few pipelining registers as possible.

Destination Lookup

Each FPGA in a cascade chain is programmed to drive a given set of destination ports, one per SERDES. For maximum flexibility, a table is used that is indexed by port number. The table output includes the targeted SERDES or a null indication if the destination is serviced by another FPGA in the chain.

Multicast Lookup

For cells designated as multicast, address lookup is performed using a single block RAM per four MGTs. Each BRAM, organized as 4K x 4, uses the Group ID field from the system header to look up which MGT, if any, gets a copy of the data. The CPU initializes the BRAM with the correct information.

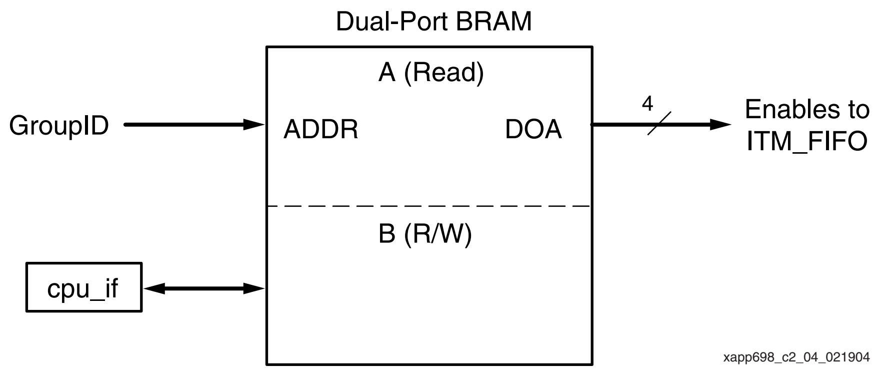


Figure 2-5: **Group Map Lookup**

Multicast cells use a single block RAM per four SERDES blocks organized as 4K x 4. The Group ID is used as an address into the RAM. The RAM output is a mask of which SERDES receives a copy of the cell.

Ingress Buffering (itm_fifo)

A small amount of ingress buffering is required because the system bus is much faster than the SERDES. The goals are to use as little RAM as possible and have all traffic priorities flow through one ingress buffer. Because of HOL blocking, egress flow control does not affect this buffer since blocking any one priority blocks all others.

The buffering must be deep enough to allow reasonable flow control response time to the ingress TM and yet maintain line-rate flow over the SERDES. If too small, the buffer either overruns, causing cell loss, or underruns, causing performance loss. If the buffering is too large, the output buffering in the egress has to be increased to support the draining of this buffer because it is not affected by flow control. Figure 2-6 shows the input buffer logic instanced one for every two SERDES blocks.

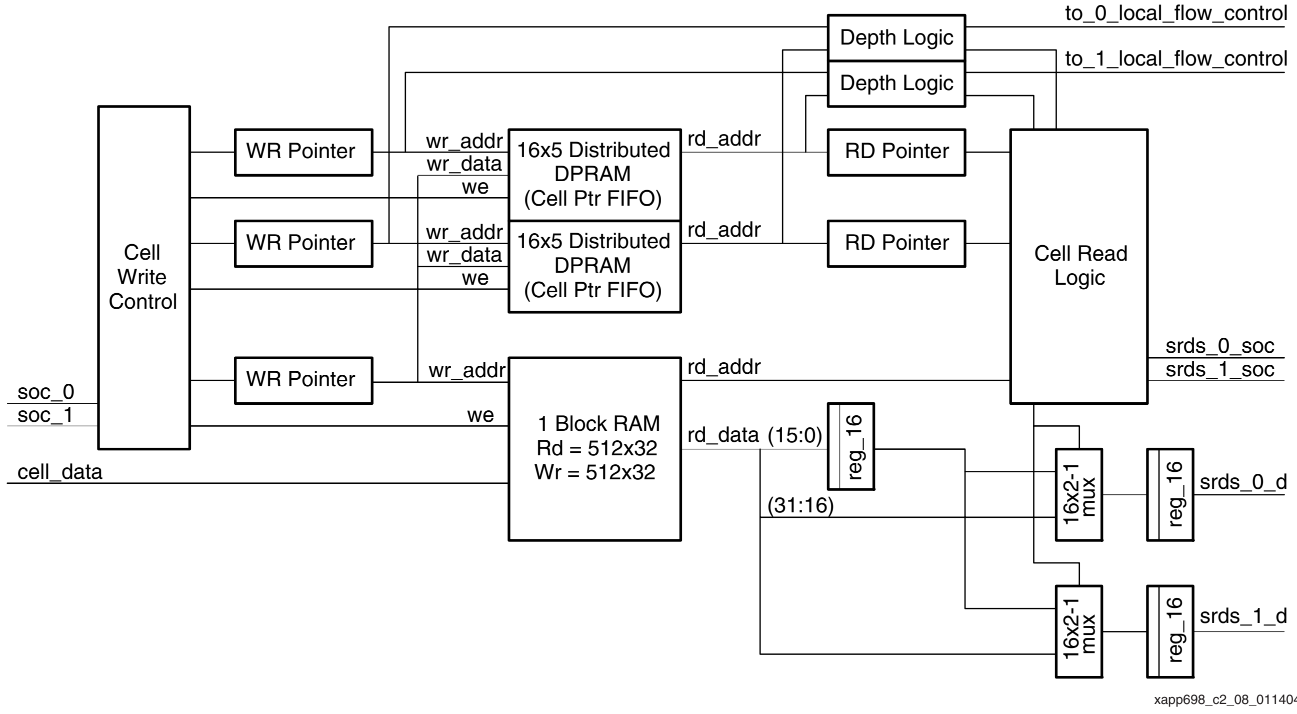


Figure 2-6: **Ingress Buffer**

The TM is allowed 32 clocks of response time from receiving a flow control off signal to stop cell flow. When coupled with a 16-clock budget for transmitting flow control (assuming 16 priorities and 16 destinations) plus another 16 clocks of pipeline delay and synchronizer delay, 64 clocks in total expire from flow control activated to no more cells arriving. With a 200 MHz system clock and a 156.25 MHz SERDES clock, 50 SERDES clocks occur during that interval. At most, the system can provide 256 bytes of new data while the SERDES can drain 100 bytes, meaning FIFO depth grows by 156 bytes after the flow control off threshold is reached. For full cells, this corresponds to less than 2.5 cells at the smallest cells size of 64 bytes.

Because the TM is allowed to be clock efficient on non-full cells, care must be taken to avoid sustained bursts of short cells into a single SERDES. This goal can be accomplished with a simple SOC pacing mechanism in the ingress TM.

Two SERDES transmitters share a single block RAM. The RAM is organized as 512 x 32 on both interfaces. Ingress system data is written directly into the block RAM. As new cells arrive from the system interface, if the cell is destined to either or both of the SERDES blocks serviced by this buffer, it is written into the RAM consecutively behind the previous cell. A pointer to the newly arrived cell in memory is stored in a separate FIFO per SERDES. This technique allows the two SERDES blocks to multicast cells from a single RAM without having to write two copies of the cell into the RAM.

The read interface is time interleaved between the two SERDES blocks. 32 bits are read at a time, where 16 bits go directly to the SERDES and 16 bits are stored for the next clock.

FIFO Depth Control

Depth is counted in each pointer FIFO and flow control for that SERDES destination is generated from the depth. Table 2-1 shows the number of cells stored based on cell payload size.

Table 2-1: **Ingress Buffer Capacity versus Cell Payload Size**

Size	Cells in RAM	Cells per Port
40	46	23
64	30	15
80	24	12
96	20	10
112	17	8
128	16	8

The cells are always evenly split between the two SERDES blocks. A five-bit depth counter provides highest possible depth count.

Local Flow Control

When the depth of each FIFO crosses a programmed threshold, flow control for all priorities to that destination is asserted back to the ingress TM. Normally this threshold is set to 3. The maximum depth must never exceed six cells for a destination. The extra cells (two to nine based on cell size) can be used to allow the TM to stream in short cells. Use caution, however, because the total number of cells needs to be reflected in the egress buffer's skid distance because flow control does not stop this buffer from draining.

Cell Transmit (cell_tx)

The cell_tx block works in conjunction with the ingress-buffering block to generate cells to be sent out a SERDES. The ingress-buffering block provides a cell start signal together with the first 16 bits of data for the cell (system-side header information). In each subsequent clock, the ingress-buffering block provides another 16 bits of data.

This block reformats the header information to build the fabric header, insert flow control information, and calculate header and payload CRC values (using $x^8 + x^2 + x + 1$).

When no cells are available to transmit, the block builds and sends IDLE cells. These cells are used to establish a working full-duplex link as well as maintain the propagation of flow control information.

The link-quality state machine also is contained in this block. Flow control to the ingress TM is held active as long as this block does not detect good cells arriving from the far end with the LOCKED bit set in the header.

The good_cells signal is input into cell_tx from cell_rx (see Figure 2-7). When TRUE, it indicates the cell that was just received was a valid cell. A cell is considered valid if there is no header CRC failure. If no cell was received within a given amount of time, the good_cells indicator becomes FALSE. The cell_tx module uses good_cells as the LOCKED bit in the cell header.

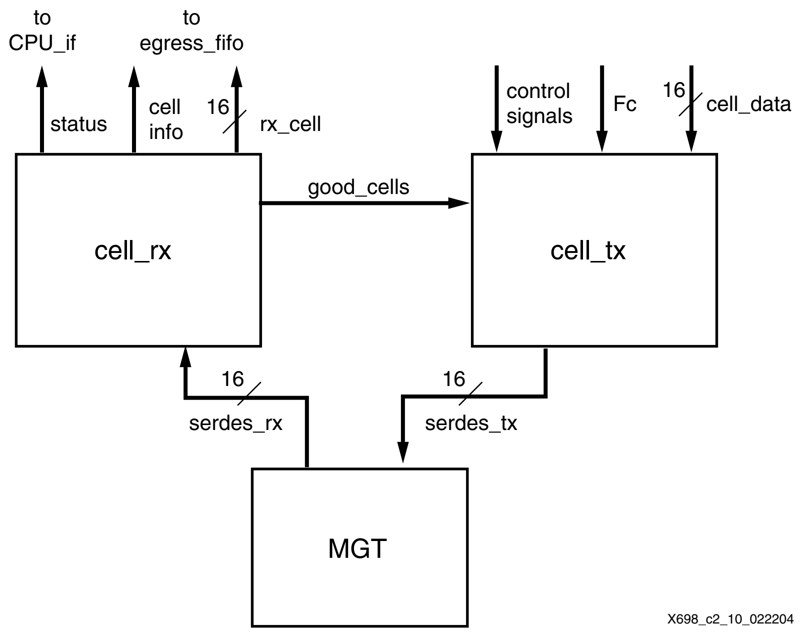


Figure 2-7: **cell_rx and cell_tx**

The cell_tx block also provides the transmit interface to the MGT megacell.

Cell Receive (cell_rx)

This block performs the receive interface to the MGT megacell. Its primary function is to perform cell-alignment by detecting the SOC comma character. It also completes the link-quality state machine with the cell_tx block.

Cells are received from the MGT, the data and flow control information are extracted, and status is collected and sent to the rest of the system. Status information includes the locked and good_cells signals, whether there was a header or data CRC error, and whether there was a symbol error. These signals are available to the software from the MFRD SERDES Status Register. The receive block also checks that cells are continuously received, and, if it fails to see valid cells within 16 clocks, the receive block deasserts the locked and good_cells signals. Any time the locked signal is deasserted, the ITM_FC module asserts flow control for all priorities on that port.

To TM or Upstream Flow Control Generation (itm_fc, itm_fifo, egress_sched, cell_rx)

Each cell received includes flow control information for each traffic priority. This block extracts that information and forwards it directly to the ingress TM. The path delay must be minimized. While the SERDES blocks indicate no full-duplex link established, or in the event of a header CRC error, flow control is assumed to be active.

Flow control also comes from the downstream cascade, egress_sched, and itm_fifo blocks. The itm_fc block consolidates these sources of flow control into one data stream, and creates a signal to indicate which flow control data goes with which port. When itm_fc_start is asserted, port itm_fc represents port zero. On each subsequent rising edge of itm_fc_clk, the port number is increased by one. If IFC_FOR_MCAST is set in `switch_scale_pack.vhd`, then there is a multicast version of flow control after the last port in the system, which is a logical-OR of all ports for each class. Flow control continuously cycles in this manner. Figure 2-8 shows the itm_fc flow control.

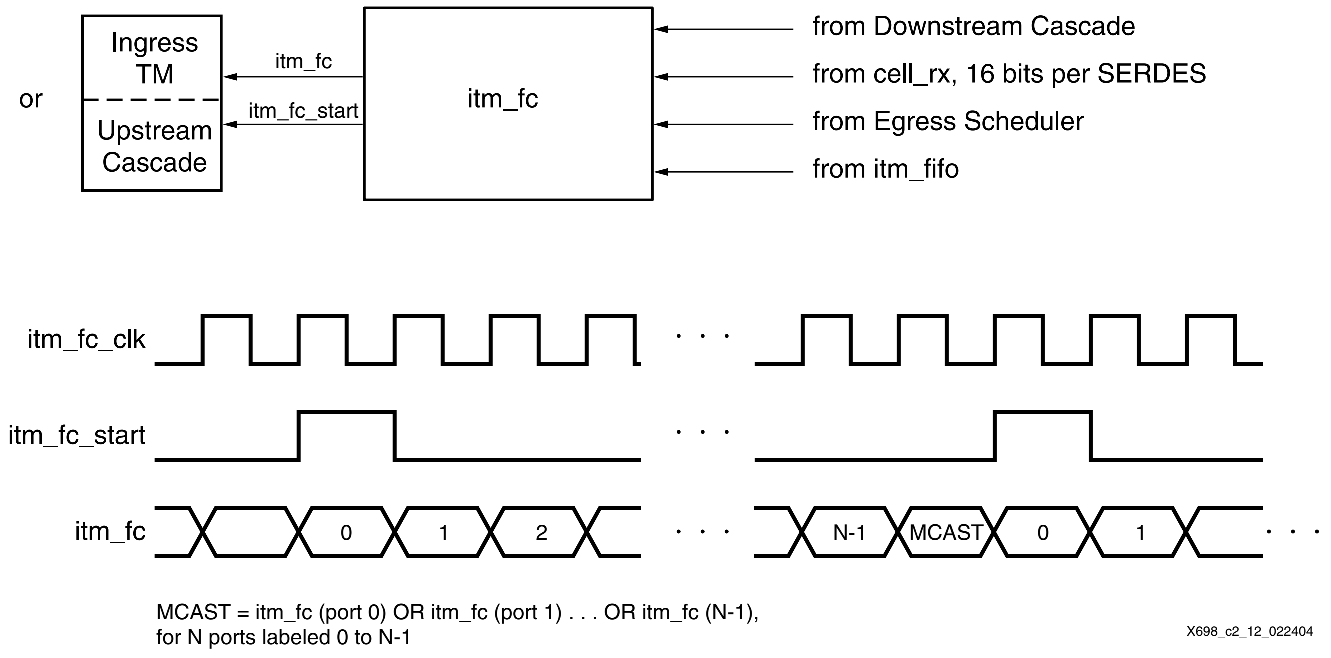


Figure 2-8: **ITM Flow Control**

A flow control stream is generated with one bit per destination and priority. Thus a 16-port switch with 16 priorities has 256 flow control bits. The width of the flow control stream is implementation dependent, with the RTL supporting three widths (number of priorities, one-half the number of priorities, and two times the number of priorities). It is up to each designer to select a width based on the number of used ports and priorities and the desired flow control communication time.

Flow control requests received from the ingress buffer generate an active bit for all priorities for the selected port because the ingress buffer runs all priorities through a single FIFO. Flow control requests received from the serial Rx blocks are fully fanned out (destination and priority) and are propagated as received.

The ingress TM needs to stop sending multicast cells destined to a given GroupID whenever active flow control is received for any port in the fanout list of that ID.

Output Queueing (egress_fifo, egress_q, egress_sched)

Each egress_fifo receives cells from up to eight cell_rx blocks simultaneously. Cells are written as soon as they arrive into preallocated buffers.

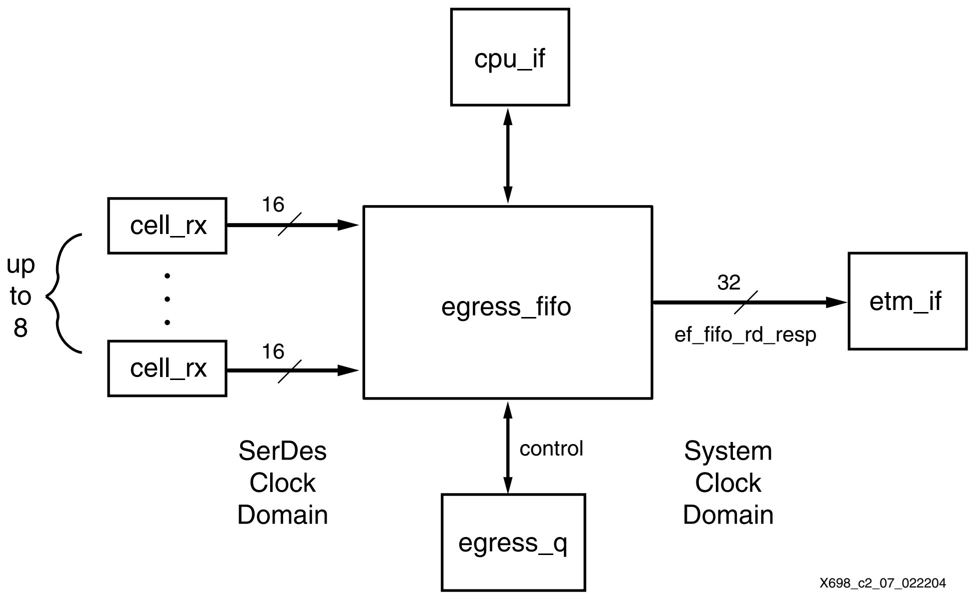


Figure 2-9: **Output Queueing**

One to three sets of block RAMs implement the egress storage. The RAMs are arranged in several possible fashions. One set of RAMs supports up to eight SERDES blocks. The set is split into eight columns of 16 bits where each column writes data from one SERDES at a time. Each SERDES writes the first word of a cell into whatever column it is connected to at the time. The alignment is stored in the output queue control to properly align the cell at output. This mechanism allows the support of any cell size.

For 9 to 16 SERDES designs, there are two banks of eight columns. There is no need to have an even number in one bank or the same amount in both banks. Splitting evenly, however, helps utilize memory more fully. For 17 to 24 SERDES designs, there are three banks of eight columns. When multiple memory banks are used, there is a second level scheduler to select egress cells from the banks.

Table 2-2: **Egress Buffering Block RAM Allocation**

Part	SERDES	RAMs Used	Banks	Columns	Read Width	Cell Buffers
XC2VP4	4	16	1	4	32	227 – 682
XC2VP7	8	32	1	8	32	455 – 1365
XC2VP20	8	64	1	8	64	910 – 2730
XC2VP30	8	64	1	8	64	910 – 2730
XC2VP40	12	128	2	8	64	1820 – 5460
XC2VP50	16	128	2	8	64	1820 – 5460
XC2VP70	20	192	3	8	64	2730 – 8190
XC2VP100	20	192	3	8	64	2730 – 8190

Note: Currently the XC2VP4 solution uses 16 RAMs to keep the design consistent with all others. Because this is such a small amount of memory, it is simple to support four additional RAMs in a second bank. Also, the design only requires four columns, saving logic.

Table 2-2 summarizes the memory structure based on part type and maximum SERDES count. Figure 2-10 illustrates the egress memory function. Cell memory is consumed 16 bytes at a time for implementations with more than four MGTs. Implementations with only four MGTs use four columns, thereby consuming 8 bytes at a time. For cells that do not completely fill a row, the remainder of that row in memory is unused. Thus the memory efficiency is a function of selected cell payload size. To calculate the number of buffers available based on cell size:

cell_size_div4 = ((system_payload_size + 8) / 4) – 1

BYTES_PER_COLUMN = NUM_COLUMNS * 2

WORDS_PER_BUF = ((cell_size_div4 * 4) + (BYTES_PER_COLUMN – 1)) / BYTES_PER_COLUMN

NUM_BUFFERS = NUM_EGRESS_FIFO_WORDS / WORDS_PER_BUF

In the equations:

- NUM_COLUMNS is 4 for designs using four or fewer SERDES blocks and 8 for designs using more than four SERDES blocks.
- NUM_EGRESS_FIFO_WORDS is 8192 for devices larger than XC2VP7 devices and 4096 for XC2VP4 or XC2VP7 devices.
- system_payload_size is between 40 and 128 bytes.

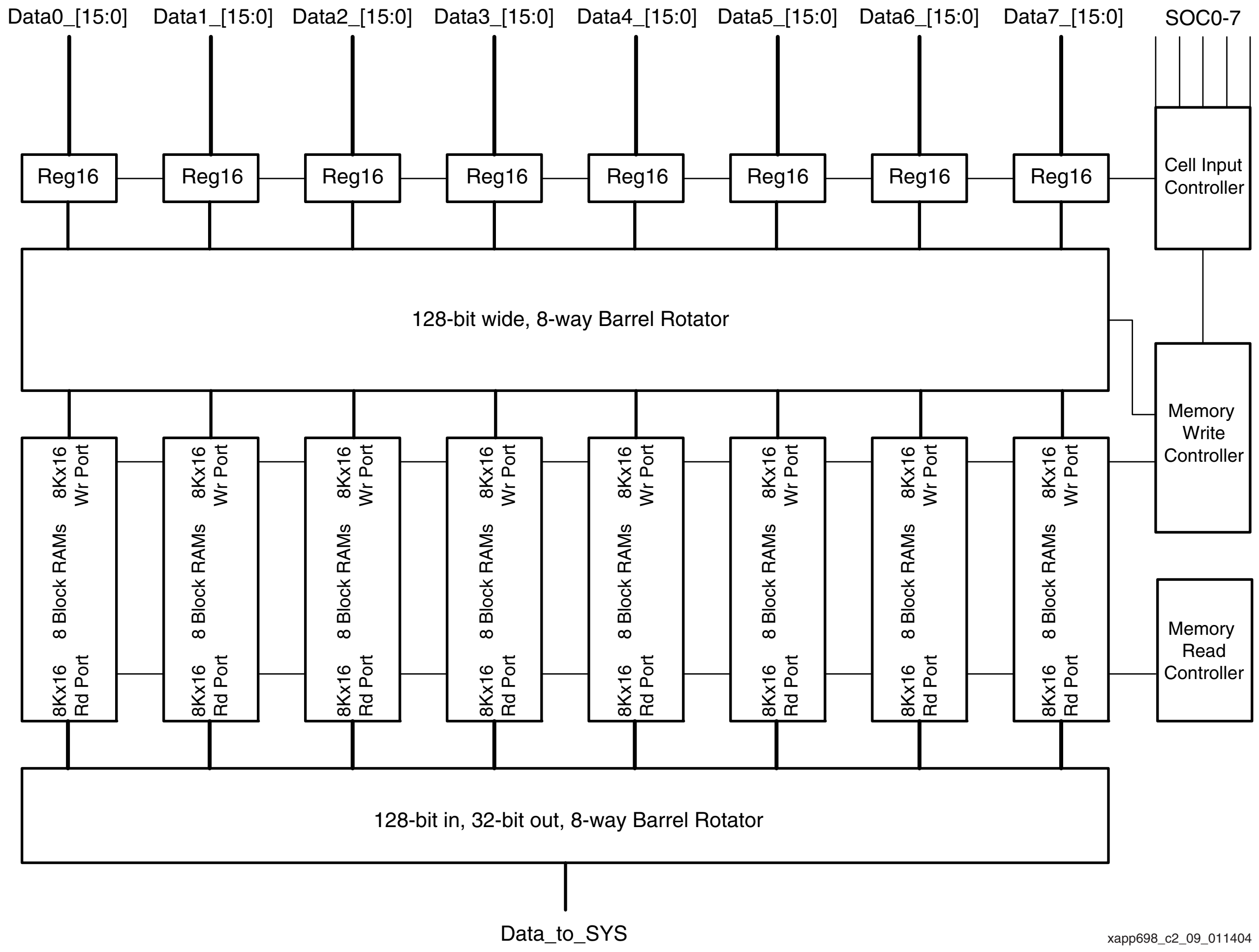


Figure 2-10: **Egress Memory Buffer and Datapath**

Memory Access Mux (egress_fifo)

This mux connects one SERDES at a time to a given subportion of the output buffer RAM. Each SERDES writes its 16 bits of data every clock into one of the RAMs, as shown in Figure 2-11. The upper half of Figure 2-11 shows an ordering of writes to the FIFO from each SERDES interface. When there is incoming cell data, the cell words are written to the column for that particular round. The number of columns (NUM_COLUMNS) depends on the number of SERDES blocks in the device (refer to the NUM_COLUMNS description for valid values).

The lower part of Figure 2-11 illustrates the buffering for a theoretical cell size of 12 bytes (six 16-bit words). Only the first four SERDES buffers (S0 through S3) are shown. S0 and S2 have completed writing a cell to buffers, where each buffer holds a complete cell. S0 happened to align its first word to the first SERDES in the FIFO. S2 had a cell whose first word was written to column 4. The buffer size must be one row larger than the cell size to allow for unaligned cells. S1 is in the process of writing a cell into the buffer. S3 is awaiting a new cell to arrive. Refer to the WORDS_PER_BUF equation for the number of rows required per buffer.

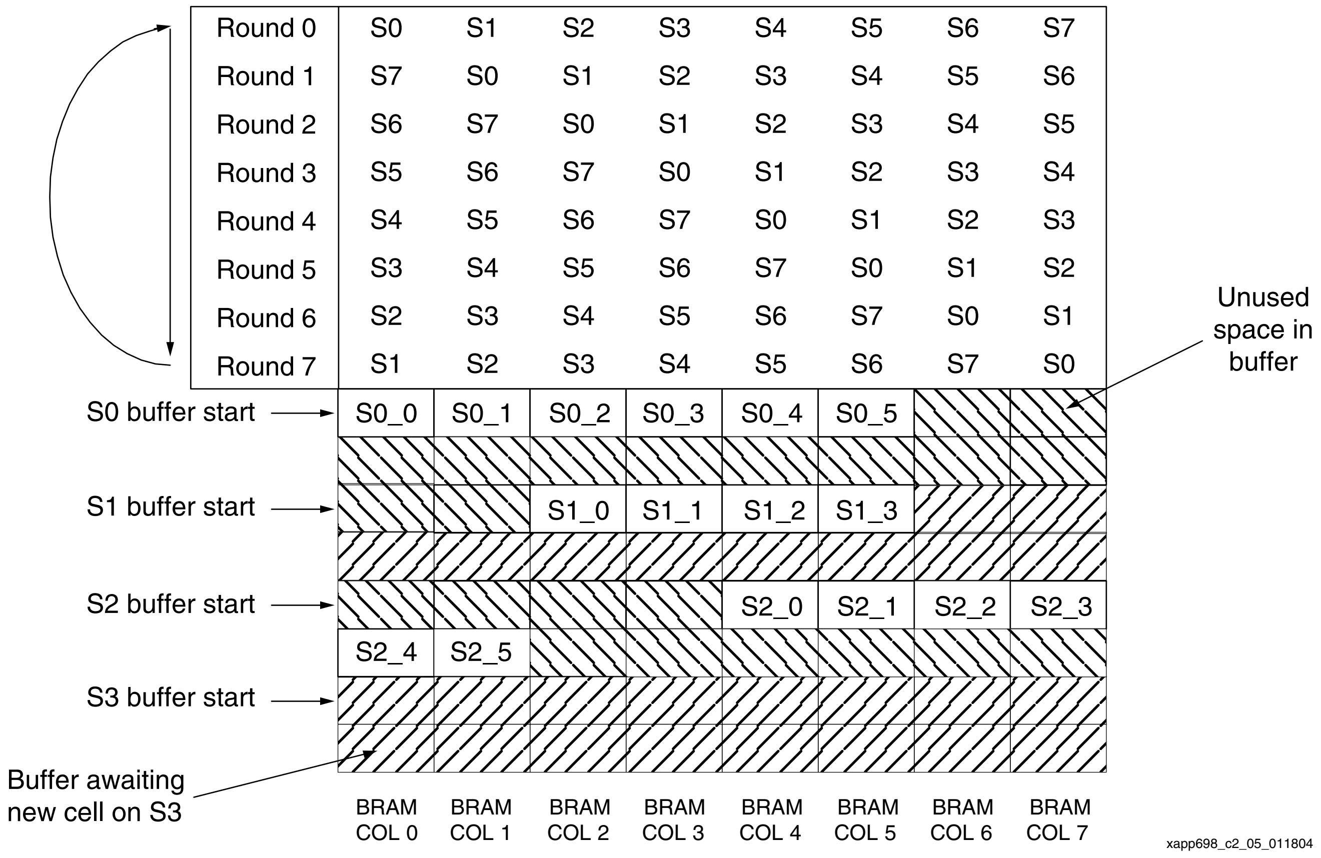

Round 0 S0 S1 S2 S3 S4 S5 S6 S7
Round 1 S7 S0 S1 S2 S3 S4 S5 S6
Round 2 S6 S7 S0 S1 S2 S3 S4 S5
Round 3 S5 S6 S7 S0 S1 S2 S3 S4
Round 4 S4 S5 S6 S7 S0 S1 S2 S3
Round 5 S3 S4 S5 S6 S7 S0 S1 S2
Round 6 S2 S3 S4 S5 S6 S7 S0 S1
Round 7 S1 S2 S3 S4 S5 S6 S7 S0
S0 buffer start
S0_0 S0_1 S0_2 S0_3 S0_4 S0_5
S1 buffer start
S1_0 S1_1 S1_2 S1_3
S2 buffer start
S2_0 S2_1 S2_2 S2_3
S2_4 S2_5
S3 buffer start
Buffer awaiting new cell on S3
Unused space in buffer
BRAM COL 0
BRAM COL 1
BRAM COL 2
BRAM COL 3
BRAM COL 4
BRAM COL 5
BRAM COL 6
BRAM COL 7
xapp698_c2_05_011804

Figure 2-11: **Memory Access Mux**

Because cells can arrive at any time on the Rx SERDES, it is not likely that a SERDES will be connected to the first Block RAM in that clock. To cover this case, each cell is allowed to start with any alignment with respect to a 16-byte memory row. When the buffer used to store the cell is queued into an output queue, the alignment at cell start is stored as well. The read interface uses this alignment to restore the cell to the expected order.

Output Buffer Controller (egress_sched)

Buffers are allocated dynamically from a common pool. Programmable thresholds are provided to control backpressure caused by pool limits. As each threshold is crossed, flow control is asserted to all sources on a programmable set of priorities, where each set is unique to that threshold. This is meant to represent "lower priority traffic queues", "higher priority traffic queues", "highest priority queue". This latter cutoff attempts to ensure the highest traffic priority can continue to flow through a heavily congested fabric, which is important for both TDM traffic as well as control plane traffic for the system. Up to 15 thresholds are supported, but any number (from 0 to 15) can be used. Ultimately, if all thresholds are exceeded, cells are dropped.

The smallest supportable cell size is a function of managing block RAM accesses for the free list. Because an allocation operation is dependent on the results of the previous allocation, the RAM cannot support allocations on successive clocks. Allowing a three-clock latency through the RAM structure means that eight SERDES blocks generating a new cell every three clocks works out to a total of 24 clocks per SERDES, or 48 bytes where eight bytes are in the cell header, leaving a 40-byte payload.

In order to support all SERDES blocks starting a cell simultaneously, each SERDES pre-allocates a buffer and refills its pending buffer as soon as a cell starts.

Output Queue Controller (egress_q)

Each queue has a programmed threshold at which flow control is activated for that queue. It also has another ceiling register above which cells are dropped instead of enqueued. Normally, this limit is not bit hit. It is an indication that threshold registers are programmed incorrectly somewhere in the system.

The drain rate out of a queue is limited by the egress system interface. The flow control on threshold is set as low as possible but not too low to avoid underrunning the maximum transfer rate achievable on this interface. Expect this level to be around 15 cells. The number of cells that can still arrive after this point is a function of how many SERDES blocks are supported in this part times the number of cells that can still arrive after flow control is signaled. This value can range from 4 to 240 cells (4 to 24 SERDES blocks). Figure 2-12 shows the egress flow control thresholds.

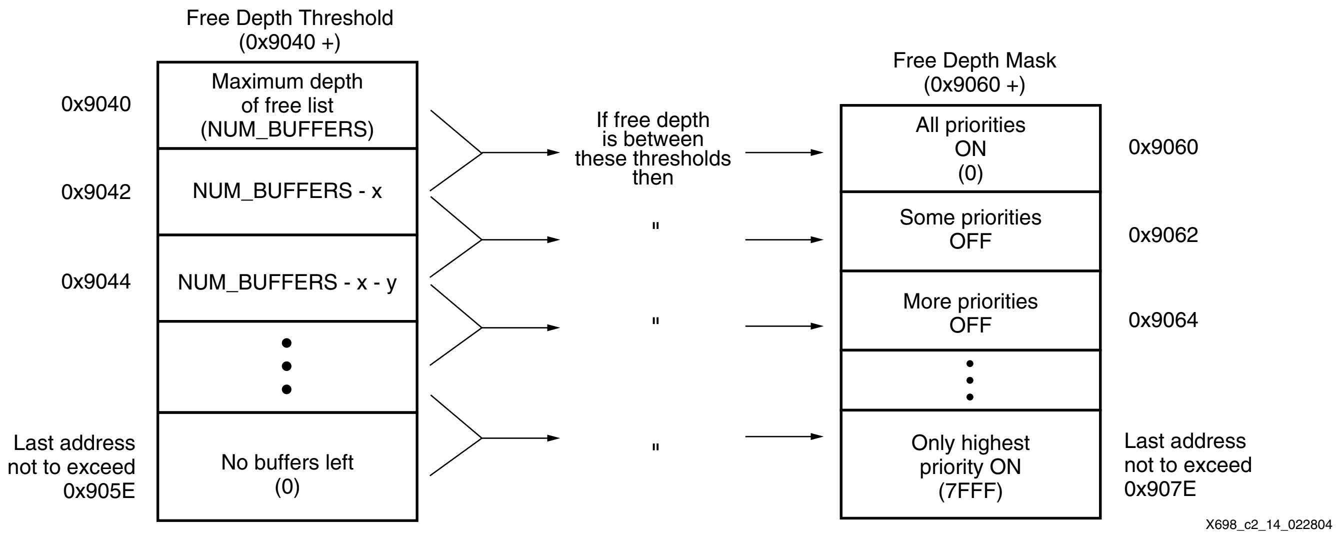


Figure 2-12: **Egress Flow Control Thresholds**

The output queues are currently written to support collapsing all of the ports serviced by an egress memory buffer into a single set of queues. This simplifies the scheduler dramatically because only a WRR priority scheduler needs to be implemented as opposed to a full priority and port scheduler.

Alternately, the queues for each port can be kept separately at the expense of requiring two levels of WRR scheduling.

Note: The current implementation has reserved block RAM space for each port with unique queues, but all ports in a FIFO block are collapsed into one set of queues.

Fabric Flow Control Generation (egress_fc)

When the mesh switch is first turned on and configured, the Free Depth register contains the total number of buffers available (NUM_BUFFERS). Each time a buffer is allocated, the contents of the Free Depth register are decremented. When the buffer is released, the Free Depth register is incremented.

When the Free Depth register's contents are between the values contained in the Free Depth Threshold register at addresses 0x9040 and 0x9042, then all priorities are on as specified in the Free Depth Mask register at address 0x9060, and there is no flow control activated. Once the Free Depth register value falls below the value in address 0x9042, yet is above the value contained in 0x9044, then some priorities have flow control asserted. The Free Depth Mask register at 0x9062 is read to determine which flow control priorities are turned on. This process continues until the Free Depth value falls below the last threshold, causing flow control to be asserted for all except the highest priority.

As queue depths for a single output queue cross the programmed threshold, flow control is returned in the cell header back to the source port for that queue or all source ports serviced by an egress memory block, if port queues are collapsed.

As the buffer thresholds are crossed, flow control is returned in the cell header for all source ports serviced by an egress memory block. The threshold bits are accepted from the buffer memory controller and merged with the per-queue bits.

Priority Scheduler (egress_sched)

A weighted round robin (WRR) scheduler selects which output queue to service next. Five bit weights are desired, but the implementation might choose to drop to four bit weights if the logic space is too high.

The WRR algorithm grooms the output so that the cell emission is smoothed evenly over a full round of weights. If this function cannot be implemented in a XC2VP4 device because of logic space restrictions, a simple "parking" style WRR can be implemented.

Note: The current implementation uses strict priority for the priority scheduler.

Note: The current implementation uses a four-clock scheduling cycle, meaning cell start events are generated no more frequently than every four clocks. A small elastic store (five entries) holds scheduled cells that are waiting to go out the egress interface. Because the number of clocks used on the egress interface is variable (from 2 to 33), the scheduler generally runs well ahead of actual cell transmission time. A long stream of very small cells causes the egress bus to become under-utilized.

Port Scheduler (egress_sched)

At a minimum, designs with more than one egress buffer block (more than eight SERDES blocks) requires a scheduler to select cells from one block at a time. If the implementation keeps output queues separate per port, a full WRR port scheduler is required. When collapsed, there is no need for WRR. In that event, a simple RR scheduler is used.

To Fabric or Downstream Flow Control Generation (etm_if, egress_sched, cell_rx)

The egress TM receives traffic for just one port so it needs to send only one flow control bit per priority back to the fabric. When asserted, the priority scheduler in all egress memory blocks will stop servicing that priority. Response time to flow control must be less than five clock cycles, not counting any cell currently in progress.

This interface also can propagate flow control downstream in a cascaded configuration. Because the current implementation collapses all port queues into one set of queues, the downstream flow control needs only include one bit per priority. It operates identically to TM received flow control.

The upstream FPGA generates this flow control when it is not ready to accept more cells of a given priority from the downstream device.

Egress System Interface (egress_if)

The egress system interface handles sending cells to the egress TM or to an upstream FPGA in a cascaded system.

Input Registers

As with the Ingress System Interface, the signals that come into the Egress System interface from the downstream FPGA can be either SDR or DDR. For DDR, a DCM phase-shifts the clock to reliably sample the incoming signals.

Cells received on this interface are held in a small retiming buffer (two cells maximum can be in distributed RAM). Any cell received is forwarded automatically to the output registers immediately after any cell in progress finishes. All FPGAs in a chain must frequency lock their core clocks to avoid the retiming buffer from over-running or under-running.

If this is the final FPGA in a chain, cells from the Ingress System Interface selected to go out the local egress (based on normal destination lookups) are looped through this interface.

Output Registers

Unlike the output registers of the ingress buffer block, these output registers are clocked by an always-running core clock (or the Ingress System Interface recovered clock). Because an FPGA might not have a downstream partner, it cannot have a running clock from the input register stage.

Egress Multiplexing

This block generates downstream backpressure based on a WRR scheduler to select local cells versus downstream cells. Each priority is scheduled separately.

The weights for the WRR algorithm are calculated based on the number of ports being serviced locally versus the number of ports downstream. The local weight is (local_ports / total_ports) while the downstream weight is ((total_ports – local_ports) / total_ports).

System Interfaces and Signal Descriptions

The reference design interfaces to the system through the following interfaces:

- **Cascade Interfaces**
  These interfaces transfer cell data between fabric devices and between the head-end fabric device and the Traffic Manager or Traffic Manager Bridge.
- **Egress TM Interfaces**
  The Egress TM uses these interfaces to communicate per-priority backpressure information to the fabric.
- **Ingress TM Interfaces**
  The fabric uses these interfaces to communicate per-port backpressure information to the Ingress TM.
- **SERDES Interfaces**
  These interfaces provide the differential clock and data signals to communicate with the MGTs.
- **Management Interface**
  This interface provides access to control and status registers within the fabric.
- **System Signal Interface**
  This interface provides the reset signal.
- **Backplane Interface**
  This full-duplex interface connects two mesh switch FPGAs together.

Cascade Interfaces

The Cascade interfaces transfer cell data between fabric devices and between the head-end fabric device and the Traffic Manager or Traffic Manager Bridge. Data transfers across the Cascade interfaces consist of a 32-bit cell header followed by from 40 to 128 bytes of user cell data. The format of the Cell Header is illustrated in Figure 2-13. The length field is zero-based, so for the minimum payload of 40 bytes, the length field should be 39 in binary.

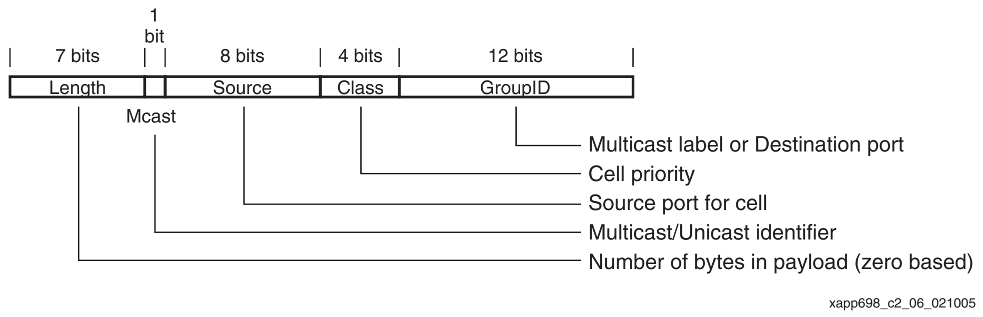


Figure 2-13: **Cascade Interface Cell Header Format**

Figure 2-14 shows the interconnecting signals between two cascading FPGAs.

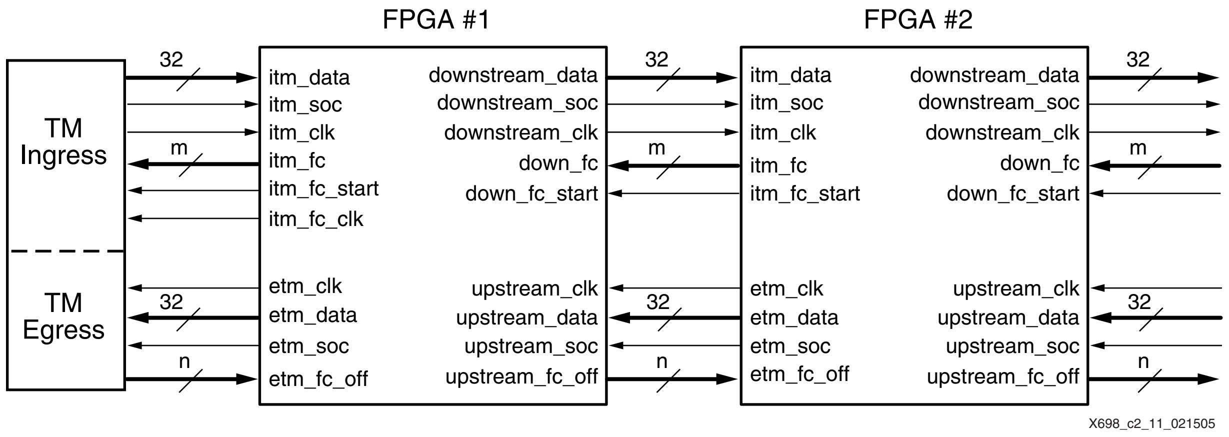


Figure 2-14: **Cascade Interface Connections**

Table 2-3 describes the Cascade interface signals.

Table 2-3: **Cascade Interface Signals**

Signal	Type	Description
Upstream_data[31:0]	Input	Cell data traveling upstream.
Upstream_soc	Input	Start of Cell indicator for data traveling upstream.
Upstream_clk	Input	Clock for data and SOC traveling upstream. The frequency can be up to 200 MHz.
Upstream_fc_off[0:n-1]	Output	Flow control from the downstream to the upstream FPGA. The width is configurable based on NUM_CLASSES.
Downstream_data[31:0]	Output	Cell data traveling downstream.
Downstream_soc	Output	Start of Cell indicator for data traveling downstream.
Downstream_clk	Output	Clock for data and SOC traveling downstream. The frequency can be up to 200 MHz.
Down_fc[0:m-1]	Input	Flow control from upstream to downstream. The width is configurable to be equal to NUM_CLASSES, NUM_CLASSES * 2, or NUM_CLASSES / 2.
Down_fc_start	Input	Signal that indicates which down_fc value corresponds to port number 0.

Egress TM Interface

Table 2-4 describes the Egress TM interface signals.

Table 2-4: **Egress TM Interface Signals**

Signal	Type	Description
Etm_data[31:0]	Output	Cell data to TM or upstream FPGA.
Etm_soc	Output	Start of Cell indicator for data to TM or upstream FPGA.
Etm_clk	Output	Clock for data and SOC to TM or upstream FPGA. The frequency can be up to 200 MHz.
Etm_fc_off[0:n-1]	Input	Flow control from the TM or upstream FPGA. The width is configurable based on NUM_CLASSES.

Ingress TM Interface

Table 2-5 describes the Ingress TM interface signals.

Table 2-5: **Ingress TM Interface Signals**

Signal	Type	Description
Itm_data[31:0]	Input	Cell data from TM or Upstream FPGA.
Itm_soc	Input	Start of Cell indicator for data from TM or upstream FPGA.
Itm_clk	Input	Clock for data and SOC from TM or upstream FPGA. The frequency can be up to 200 MHz.
Itm_fc(0:m-1)	Output	Flow control to the TM or upstream FPGA. The width is configurable to be equal to NUM_CLASSES, NUM_CLASSES * 2, or NUM_CLASSES / 2.
Itm_fc_start	Output	Signal that indicates which itm_fc value corresponds to port number 0.

SERDES Interface

Table 2-6 describes the SERDES interface signals.

Table 2-6: **SERDES Interface Signals**

Signal	Type	Description
serdes_refclk_p[0:p], serdes_refclk_n[0:p]	Input	Differential reference clocks for MGTs. The width is configurable based on EDGES, a variable set in switch_scale_pack. The frequency can be up to 156 MHz.
out_refclk_p, out_refclk_n	Output	Differential reference clocks for MGTs used only when EDGES = 1. Connect out_refclk_p and out_refclk_n outside the FPGA to serdes_refclk_p[1] and serdes_refclk_n[1], respectively.
TXN[0:r], TXP[0:r]	Output	Differential transmit signals from the MGTs. The width is configurable based on NUM_SERDES.
RXN[0:r], RXP[0:r]	Input	Differential receive signals to the MGTs. The width is configurable based on NUM_SERDES.

Management Interface

The Management Interface is used to access internal status and control registers. This implementation is based on the CoreConnect Device Control Register (DCR) bus. Consult the IBM 32-Bit Device Control Register Bus Specification for a description of the architecture of the DCR bus.

Table 2-7: **DCR Interface Pins**

Signal	Type	Description
DCR_Clock	In	DCR Clock. This input is the CPU clock, which can run at up to 100 MHz.
DCR_Write	In	Write transaction. This signal indicates that a write transaction is in progress and that the address on DCR_ABus is valid. Once this input is asserted, the CPU waits a maximum of 16 clocks for DCR_Ack to be asserted. If DCR_Ack is not asserted within 16 clocks, the CPU times out and executes the next instruction. This signal stays asserted until DCR_Ack is asserted.
DCR_Read	In	Read transaction. This signal indicates that a read transaction is in progress and that the address on DCR_ABus is valid. Once this input is asserted, the CPU waits a maximum of 16 clocks for DCR_Ack to be asserted. If DCR_Ack is not asserted within 16 clocks, the CPU times out and executes the next instruction. This signal stays asserted until DCR_Ack is asserted.
DCR_Abus[9:0]	In	Address bus. This bus is valid whenever DCR_write or DCR_read is asserted, and is invalid otherwise. The address remains stable during a write or read transaction.
DCR_Ack	Out	DCR transfer acknowledge back to CPU. This output indicates that the DCR slave logic has written the data from DCR_DBusIn on a write cycle, or that read data is available on DCR_DBusOut for a read cycle. DCR_Ack must be kept asserted until DCR_Write or DCR_Read is deasserted, then it must be deasserted. The CPU waits for DCR_Ack to be deasserted before starting a new operation.
DCR_DBusOut[31:0]	Out	DCR read data bus to CPU or another DCR block. The MFRD drives this bus with the contents of the selected register during a read operation, as long as the address on DCR_ABus matches one of the MFRD's addresses and DCR_read is asserted. Data must be valid when DCR_Ack is asserted and remain valid as long as the MFRD is selected. When the MFRD is not selected, this bus is driven with DCR_DBusIn.
DCR_DBusIn[31:0]	In	DCR write data bus to DCR from CPU or another DCR block. Data on this bus is written to the location specified by DCR_Abus as long as DCR_write is active. This signal can come from the CPU or from another DCR unit's bypass.

System Signal Interface

Table 2-8 describes the system signal.

Table 2-8: **System Signal**

Signal	Type	Description
Sys_reset	Input	Active-High reset for the entire design.

Backplane Interface

This full-duplex interface connects two mesh switch FPGAs together. Data cells flow from one to the other for a given source-to-destination mapping. Flow control goes in the reverse direction from the data. Cells are a fixed size, set in the MFRD configuration register Config_Reg_1.

Fabric cells include eight bytes of overhead and a programmed amount of payload. The overhead controls cell flow and communicates backpressure. Table 2-9 defines the format of the cell overhead.

Table 2-9: **Backplane Cell Header Format**

Signal	Bit Range	Number of Bits	Function
SOC	63:56	8	Start of cell delimiter (8B/10B comma character)
Locked	55	1	Indicates the full-duplex path is established, and the link is okay to use
Full	54	1	If set, valid payload length = cell size. Otherwise, the last byte of the payload field is the length of valid data.
Idle	53	1	0 = valid payload, 1 = idle cell
Mcast	52	1	0 = unicast cell, 1 = multicast
reserved	51:48	4	For future expansion
Class	47:44	4	Traffic class for cell
GroupID	43:32	12	Multicast label
BPmap	31:16	16	Per class, backpressure-asserted bitmap
HCRC	15:8	8	Header CRC
Length	7:0	8	Used when Full = 0, otherwise the payload starts here
Payload	-	320 to 1024	Cell user payload (40 to 128 bytes)
PCRC	-	8	Payload CRC (located at end of cell)

In the future, the reserved bits can be used to carry a small sequence number (in order to assist link aggregation), SOF/EOF markers (for a packet fabric instead of a cell fabric), etc.

If more bits are needed, the multicast bit can be encoded into the class bits, where certain classes are reserved for multicast.

Periodically, the cell transmitter must include a gap between cells. The gap pattern is a repeating, two-symbol pattern for a total of four symbols, set to “F7F7”. This gap can be shrunk or grown by the serial receive logic in the FPGA to allow for clock-rate adjustment between the receiver and sender.

Chapter 3

Implementation

This chapter contains the following sections:

- "Memory Map"
- "Accessing the MFRD Register Space"
- "Internal MFRD Register Descriptions"
- "Egress Buffer Control Initialization"
- "Parameterization"
- "Implementation Size Based on SERDES and FPGA Selection"
- "Design Tools"

Memory Map

The MFRD registers are accessed through the DCR bus. The DCR Address bus provides 1K 32-bit locations. The MFRD uses five addresses. The Base Address can be located on any DCR address, modulo 8, for example, 0x000, 0x008, 0x010, 0x018. The Base Address is a constant that can be set in `mesh_switch_pack.vhd`. Table 3-1 summarizes all the available registers.

Table 3-1: **MFRD Memory Map**

Address Range (Hex)	Function
0x0000 – 0x1FFF	General configuration registers
0x2000 – 0x3FFF	General status registers
0x4000 – 0x5FFF	SERDES configuration registers
0x6000 – 0x7FFF	SERDES status registers
0x8000 – 0x9FFF	Egress buffer manager configuration
0xA000 – 0xBFFF	Group ID mapping tables
0xC000 – 0xFFFF	Null – Reserved

In this chapter, each register is depicted as a three-row table. The first row breaks down the register bits, the second row describes each bit's function, and the third row shows the bit values after reset. Reserved fields always read as 0s.

Accessing the MFRD Register Space

The DCR interface uses a 10-bit address bus and a 32-bit data bus. Each MFRD instantiation uses five addresses from the DCR to access the MFRD register space. The internal registers are accessed through these five registers:

- RAReg - Read Address Register
- WAReg - Write Address Register
- RDReg - Read Data Register
- WDReg - Write Data Register
- CSReg - Control & Status Register

All the registers except CSReg are 16 bits wide. CSReg can use the full 32 bit width.

The DCR Base value by default is 32 decimal, or 20 hex.

Accessible Register Descriptions

Read Address Register (RAReg)

DCR Base + 0x0
Read/Write

31:16	15:0
Reserved	MFRD Read Address
0000	0000

MFRD Read Address: This field contains the read address for internal MFRD registers. The CPU writes the read address to this location, and the read data is returned to RDReg.

Write Address Register (WAReg)

DCR Base + 0x1
Read/Write

31:16	15:0
Reserved	MFRD Write Address
0000	0000

MFRD Write Address: This field contains the write address for internal MFRD registers. The CPU writes the MFRD write address to this location, and then writes the MFRD data to WDReg.

Read Data Register (RDReg)

DCR Base + 0x2
Read

31:16	15:0
Reserved	MFRD Read Data
0000	0000

MFRD Read Data: This field contains the read data from internal MFRD registers. It returns data that corresponds to the address in RAReg.

Write Data Register (WDReg)

DCR Base + 0x3
Read/Write

31:16	15:0
Reserved	MFRD Write Data
0000	0000

MFRD Write Data: This field contains the write data for internal MFRD registers. The CPU writes the data that is subsequently written to the MFRD register whose address resides in WAReg.

Control & Status Register (CSReg)

DCR Base + 0x4
Read/Write

31:4	3	2	1	0
Reserved	WAD	WAI	RAD	RAI
0000000000000000000000000000	0	0	0	0

The Control & Status register (CSReg) allows the software to provide one address and perform multiple reads or writes to consecutive addresses. To perform a write, first the software writes CSReg and sets the appropriate bit depending on whether the address is incremented (WAI) or decremented (WAD). After setting CSReg, the software writes the starting address to WAReg and then writes the first data word to WDReg. At the conclusion of the first write, the address in WAReg is updated with the next address to be written.

Similarly, for a read, the software sets either the RAI or the RAD bit, writes to the starting address into RAReg, and then reads RDReg. After reading RDReg, the address in RAReg is updated with the next address to be read. When the software is through with the auto-address transaction, it needs to write again to CSReg to turn off the auto-address bits.

This feature is most efficiently utilized when at least five transactions are performed, because there is some overhead to setting up the process. If more controls are added to

CSReg, it is recommended that the software performs a read-modify-write operation on CSReg before modifying the bits.

RAI: This bit enables auto-increment for RAReg. Software writes a 1 to this bit to turn on auto-increment by 2 for the address stored in RAReg. The new address generated by the auto-increment logic is stored into RAReg for software to read. To disable auto-increment, software needs to write a 0 to this bit. Use for five or more increments.

RAD: This bit enables auto-decrement for RAReg. Software writes a 1 to this bit to turn on auto-decrement by 2 for the address stored in RAReg. The new address generated by the auto-decrement logic is stored into RAReg for software to read. To disable auto-decrement, software needs to write a 0 to this bit. Use for five or more decrements.

WAI: This bit enables auto-increment for WAReg. Software writes a 1 to this bit to turn on auto-increment by 2 for the address stored in WAReg. The new address generated by the auto-increment logic is stored into WAReg for software to read. To disable auto-increment, software needs to write a 0 to this bit. Use for five or more increments.

WAD: This bit enables auto-decrement for WAReg. Software writes a 1 to this bit to turn on auto-decrement by 2 for the address stored in WAReg. The new address generated by the auto-decrement logic is stored into WAReg for software to read. To disable auto-decrement, software needs to write a 0 to this bit. Use for five or more decrements.

Auto-Address Operation

This section shows the waveforms for reads and writes to internal registers using CSReg auto-addressing. Figure 3-1 shows a write operation using auto-incrementing. Figure 3-2 shows a read operation using auto-incrementing.

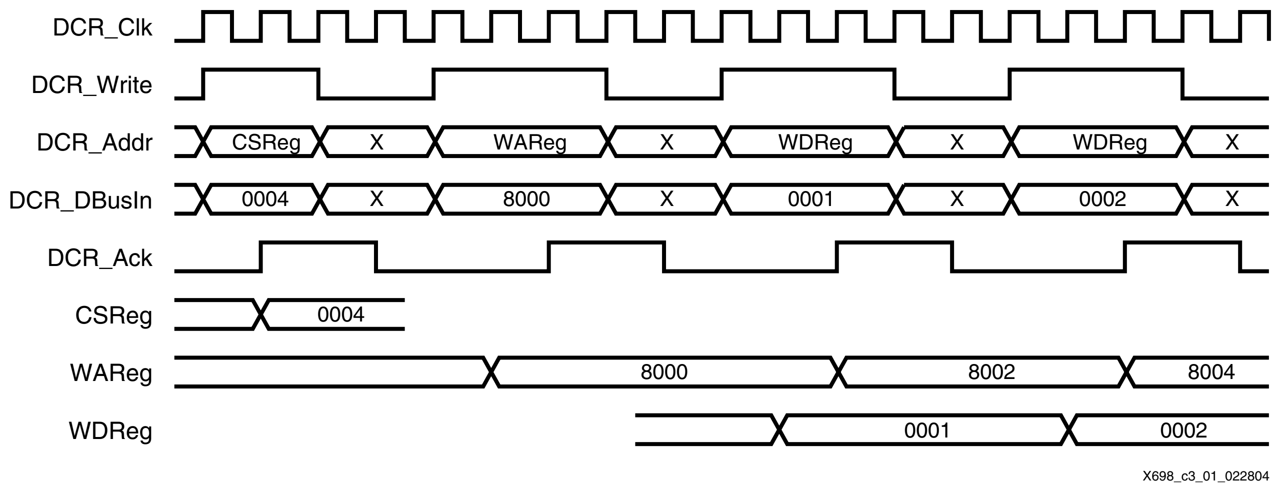


Figure 3-1: **Write with Auto-Increment**

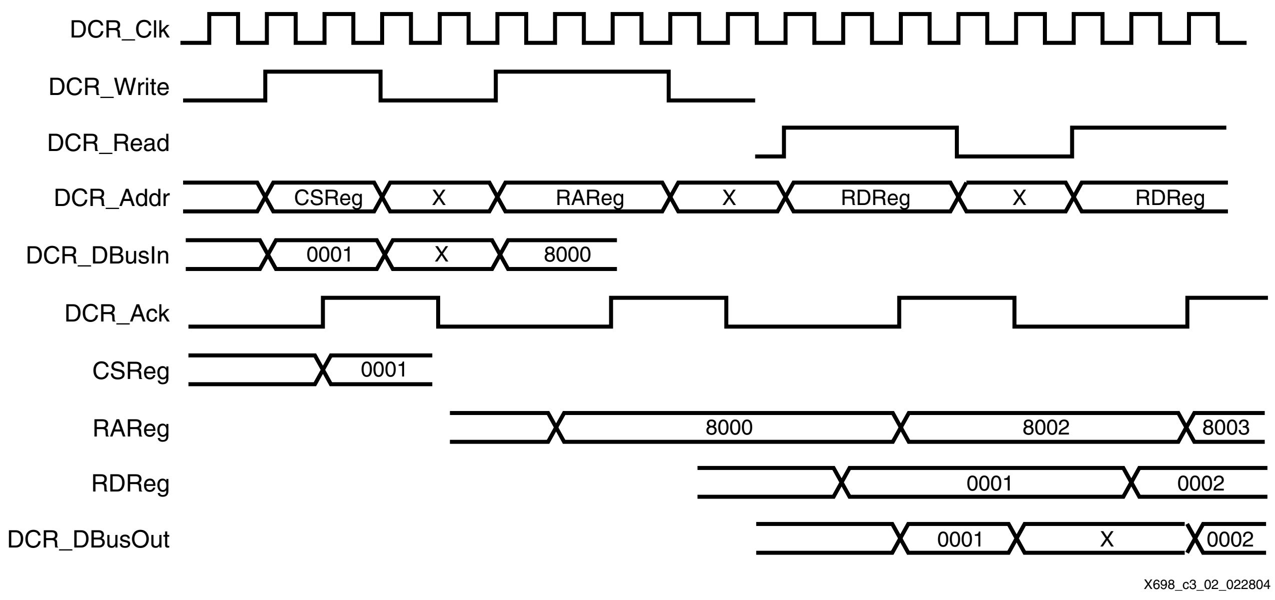


Figure 3-2: **Read with Auto-Increment**

Internal MFRD Register Descriptions

Each internal MFRD register is detailed in this section. Table 3-2 summarizes the registers.

Table 3-2: **Internal Control Register Summary**

Offset Address (Hex)	Name	Description
General Configuration Registers		
0x0000	Ver_Reg	Full-Mesh Version
0x0002	Config_Reg_1	Configuration register 1
0x0004	Config_Reg_2	Configuration register 2
0x0006	Port_Sel_Reg	Port Selection register
0x0008	DCM_Rst_Reg	DCM Reset register
0x000A	DCM_Shift_Reg	DCM Phase Shift register
0x000C	DCM_Shift_Dir	DCM Phase Shift Direction register
0x000E	FIFO_Grp_Tbl_Sel	Egress FIFO/Group Table Select
0x0010	Egrs_Thresh_Reg	Egress Thresholds (flow control depth and max queue depth)
0x0012	Egrs_FIFO_En	Egress FIFO Enables
0x0014	Dwn_strm_Port_Cnt	Downstream Port Count
General Status Registers		
0x2000	Status_Reg	Full-Mesh Status registers
0x2002	Error_Stat	Error Status Register
0x2004	Dev_Struct	Device Structure Summary

Table 3-2: **Internal Control Register Summary *(Continued)***

Offset Address (Hex)	Name	Description
SERDES Configuration Registers		
0x4000+	SERDES_Config	SERDES Configuration (one register per SERDES)
SERDES Status Registers		
0x6000+	SERDES_Status	SERDES Status registers
Egress Buffer Manager Configuration		
0x8000+	Buff_Ent_Lst	Buffer Entry List Address
0x9000	Free_Head_Ptr	First Free Buffer
0x9002	Free_Tail_Ptr	Last Free Buffer
0x9004	Free_Depth	Number of Buffers in the list –1
0x9040+	Free_Dph_Thres_Tbl	Free Depth Threshold Table (maximum of 15 registers)
0x9060+	Free_Dph_Msk_Tbl	Free Depth Mask Table (maximum of 15 registers)
Group ID Mapping Tables		
0xA000+	ID_Map_Tbl	Group ID Mapping Table

General Configuration Registers

Ver_Reg

Offset Address = 0x0000
Read Only

31:16	15:8	7:0
Reserved	Version	Revision

Config_Reg_1

Offset Address = 0x0002
Write/Read

31:13	12	11	10	9	8	7:6	5:0
Reserved	Loop My Port	Gen ITM FC	Has Down	Force Tx Idle	Enable Tx	Reserved	Cell_size_ div_4
0	0	0	0	0	0	00	001011

Loop_My_Port: When this bit is set, cells sent into this part destined to this port (as defined by my_port) are looped through the FPGA directly to the egress TM. This bit must only be set in the last device of a multi-FPGA chain. This provision avoids wasting a SERDES to loop traffic from this port's ingress to egress as well as saving the TMs from providing this loopback function.

Gen_ITM_FC: When this bit is set, flow control frames are generated to the ingress TM. Otherwise the FC_START flag never is asserted.

Has_Down: When this bit is set, another FPGA is downstream of this one. It is used by the ingress TM flow control generator to control generation of FC_START. The last FPGA in a chain creates the flow-control timing. All upstream devices lock to that timing.

Force_Tx_Idle: When this bit is set, no cells are transmitted from any of the SERDES blocks. Instead, continuous /I2/ patterns are sent (K28.5, D16.2). MGT macrocells automatically convert D16.2 into D5.6 as required for disparity.

Enable_Tx: When this bit is set, cells with payload are sent from the SERDES. When this bit is cleared, cells received from the ingress TM are stored in the ingress FIFO but not sent. Instead only EMPTY cells are sent.

Cell_size_div_4: This field controls the maximum payload size sent in one cell. The full cell size equals (Cell_size_div_4 + 1) * 4. The default value is 0xB (d11), which corresponds to 48 byte cells, implying 40 bytes of payload maximum. Do not set this field below 0xB or over 0x20 (d32). Cells do not need to be fully filled. This just controls what is sent between the SERDES blocks as a fixed unit.

Config_Reg_2

Offset Address = 0x0004
Write/Read

31:16	15:0
Reserved	Ingress FIFO Enables (per SERDES pair)
0	0000000000000000

Each pair of SERDES TX blocks is served by one ingress FIFO element (a block RAM split into three FIFOs, where one FIFO is dedicated to each SERDES and the third FIFO holds multicast cells common to both SERDES blocks. Each pair has one enable bit that controls the admission of cells from the ingress TM. If the enable bit is off (0), cells destined to either SERDES are discarded and the FIFO is flushed. If the enable bit is on (1), cells are accepted and stored.

The total number of bits actually supported in this register is:

(number of SERDES blocks used in the design + 1) / 2.

Once a bit is inactive, software must it inactive for at least eight SERDES clock periods to ensure hardware is fully initialized.

Port_Sel_Reg

Offset Address = 0x0006
Write/Read

31:16	15:8	7:0
Reserved	Base_Port	My_Port
0	00000000	00000000

My_Port: Physical port number in a switch system that corresponds to this switch instance's port number. For example, in a system with 16 ports total, there are 16 unique instances of mesh switch FPGAs or multi-FPGA chains. Each FPGA or all FPGAs in a chain are given a unique My_Port number that sequences from 0 to 15. Each switch instance only uses 15 SERDES blocks because the local port is looped back without use of a SERDES.

When the ingress TM sends in a cell whose destination is My_Port, that cell is not forwarded to any SERDES. Instead it is looped directly to the egress TM (if LOOP_MY_PORT is enabled in the last device of the chain). The loopback FPGA sets My_Port into the SOURCE field of the cell header automatically.

Base_Port: This value is the port number assigned to SERDES instance 0 in any given FPGA. The mesh switch FPGAs always map the N SERDES blocks linearly from Base_Port up. Assume XC2VP50 devices are used in the same 16-port example. Each XC2VP50 device uses 15 SERDES blocks. If Base_Port is set to zero, SERDES instance 0 maps to physical port 0, instance 1 maps to port 1, and so on. The FPGA serving port 0 has My_Port set to 0. In this case, Base_Port is set to 1, implying SERDES instance 0 maps to physical port 1, SERDES instance 1 maps to physical port 2, and so on.

For the cases where My_Port lands within the span of ports covered by the SERDES, a gap is introduced to cover that situation. For example, using the same system scenario, the device corresponding to My_Port = n (where n > 0) maps SERDES instance 0 to physical port 0, up to SERDES instance n – 1 mapped to physical port n – 1. Physical port n is then served via the local loopback. SERDES instance n maps to physical port n + 1 up to SERDES instance 14 mapping to physical port 15. The gap is supported in all aspects of SERDES blocks and flow control mapping.

There are instances where the absolute mapping can cause PCB difficulties. In that event, relative port mapping, where My_Port is always 0, could be used.

DCM_Rst_Reg

Offset Address = 0x0008
Write/Read

31:3	2	1	0
Reserved	Upstream Cells DCM	Ingress TM DCM	SERDES Refclk DCM
0	0	0	0

Each DCM can be reset manually.

SERDES Refclk DCM: This bit indicates if the DCM is attached to REFCLK_P/N input 0. While two REFCLK_P/N inputs are required (one for the top of the die, one for the bottom), only the top one connects to a DCM to generate the system timing.

Ingress TM DCM: When this bit is set, the system cell stream input uses a DCM to recover clock and double its frequency for data sampling. Because the clock is edge-aligned, a phase shift is required. Dynamic shifting is supported to support fine-tuning. The default shift value loaded in the device's UCF file must correspond to a quadrature delay.

Upstream Cells DCM: When this bit is set, cell data received from a downstream FPGA uses a DCM to recover data similarly to the ingress TM. It must be tuned accordingly. The design depends on the frequencies being identical (common reference) while the phases can vary.

DCM_Shift_Reg

Offset Address = 0x000A
Write/Read

31:3	2	1	0
Reserved	Upstream Cells DCM	Ingress TM DCM	SERDES Refclk DCM
0	0	0	0

Each DCM can be fine-phase shifted dynamically by writing a one to the appropriate bit (multiple bits can be set simultaneously). Because the bit automatically clears after one clock, it never returns a value of 1 when read. Each write corresponds to one shift command. Software needs to check the DCM_READY status register before applying a shift command.

DCM_Shift_Dir

Offset Address = 0x000C
Write/Read

31:1	0
Reserved	Shift Up
0	0

When this bit is cleared, the DCM is shifted down (less phase shift) with each shift command. When this bit is set, the DCM is shifted up (more phase shift). All DCMs share the same shift direction.

FIFO_Grp_Tbl_Sel

Offset Address = 0x000E
Write/Read

31:2	1:0
Reserved	Page Select
0	00

Each group of 16 ports requires a unique group ID mapping table. Further, no more than eight ports can be assigned to a single egress FIFO structure, where each FIFO structure consumes 8K worth of memory space. This register allows the CPU access to one particular instance at a time.

For the Group ID mapping table, one bit per SERDES is required plus one bit for the local loopback port. This is kept internally as a single vector indexed from 0 to NUM_SERDES. When Page_Select = 0, the CPU has access to bits 15 down to 0 in this vector. When Page_Select = 1, the CPU has access to bits 31 down to 16. Bit 0 always corresponds to

SERDES 0, while bit NUM_SERDES (one past the last SERDES) always corresponds to the local loopback port.

For the egress FIFO structures, the Page_Select field corresponds to a particular set of SERDES blocks based on the total number of SERDES blocks. Mapping to physical ports is identical to ingress mapping based on opening a hole for the local port. Table 3-3 shows which SERDES blocks are mapped to which egress FIFO structure for a given total number of SERDES blocks. The FIFO block number corresponds to the Page_Select field.

Table 3-3: **SERDES to FIFO Mapping**

Total SERDES Blocks in Part	FIFO Block #0	FIFO Block #1	FIFO Block #2
1 to 8	0 to 7	na	na
9	0 to 4	5 to 8	na
10	0 to 4	5 to 9	na
11	0 to 5	6 to 10	na
12	0 to 5	6 to 11	na
13	0 to 6	7 to 12	na
14	0 to 6	7 to 13	na
15	0 to 7	8 to 14	na
16	0 to 7	8 to 15	na
17	0 to 5	6 to 11	12 to 16
18	0 to 5	6 to 11	12 to 17
19	0 to 6	7 to 12	13 to 18
20	0 to 6	7 to 13	14 to 19
21	0 to 6	7 to 13	14 to 20
22	0 to 7	8 to 14	15 to 21
23	0 to 7	8 to 15	16 to 22
24	0 to 7	8 to 15	16 to 23

Note: This data is not required for initialization. It is provided for information only.

Egrs_Thresh_Reg

Offset Address = 0x0010
Write/Read

31:16	15:8	7:0
Reserved	Flow_Control_Depth	Max_Queue_Depth
0	00000000	00000000

Flow_Control_Depth: When the number of cells in a given Q equals or exceeds this depth, flow control is asserted and sent back through all SERDES blocks fed by this FIFO structure. This value corresponds to a specific priority because each priority has a unique queue.

Max_Queue_Depth: When the number of cells in a given Q equals this value, no more cells to that queue are accepted. Any new cells are discarded. This action should never happen as it indicates flow control was ignored. If it does happen, check if the two limits are set too closely together.

Egrs_FIFO_En

Offset Address = 0x0012
Write/Read

31:16	15	14:8	7:0
Reserved	Enable_Sched	Enable_Allocation	Enable_Queue
0	0	0000000	00000000

Enable_Queue: Each FIFO structure has one bit in this field. Bit zero corresponds to FIFO 0, bit one corresponds to FIFO 1, and so on. Normally no more than three bits are used because no part has more than 24 SERDES blocks. When one of these bits is set, cells are stored into output queues as expected. When one of these bits is cleared, the output queue memory structures are initialized and all cells are removed.

Note: If cells were queued, the buffers are not returned to the free list. Software is responsible for restoring the egress memory structures if this field is used during normal operation.

Once disabled, a FIFO structure must be left disabled for N x TM_CLOCK_PERIOD, where N is the number of queues supported. Currently that value corresponds to 16 clock periods because 16 traffic priorities are supported with simple priority based queues.

Enable_Allocation: Each FIFO structure has one bit in this field. Bit zero corresponds to FIFO 0, bit one corresponds to FIFO 1, and so on. When one of these bits is set, buffers can be removed from the free list and given to SERDES receivers. When one of these bits is cleared, buffer requests from the SERDES are ignored implying that any received cells are ignored. Allocation MUST be disabled while queues are disabled. If not, buffers are lost.

The proper initialization sequence is:

1. Disable allocation and disable queues for all FIFOs (default state after reset).
2. Initialize the free list via the CPU.
3. Initialize the FREE_HEAD, FREE_TAIL and FREE_DEPTH registers. At this point, the output queues are guaranteed to be fully initialized.
4. Enable output queues.
5. Enable allocation for the desired queues.

Enable_Sched: If set, egress scheduler is allowed to send cells to the egress TM interface. If clear, cells are buffered but not sent. Upstream cells still flow through, but they can be stopped using the Enable_Sched bit in the downstream part.

Dwn_strm_Port_Cnt

Offset Address = 0x0014
Write/Read

31:8	7:0
Reserved	Downstream_Port_Count
0	00000000

Downstream_Port_Count: This field is set to the number of ports served by FPGAs downstream of this FPGA in a multi-FPGA cascaded switch. It is used by the egress cell arbitration to fairly weight cells between FPGAs. For example, consider a switch built from an XC2VP7 device with eight ports cascaded in front of an XC2VP4 device with four SERDES ports and one local port looped back. In that case, Downstream_Port_Count is set to 5, and the XC2VP7 is guaranteed 7/5 of the total bandwidth of the egress TM interface.

In order to save on block RAMs, the upstream part uses a single block RAM as a FIFO for downstream cells headed to the egress TM. To prevent HOL blocking, any cells present in this FIFO are immediately forwarded. Fairness between FPGAs is accomplished via a weighted bucket that operates over an average number of cells and asserts backpressure using the depth of the queues for the local ports.

General Status Registers

The following registers belong to the General Status memory region.

Full-Mesh Status Register (Status_Reg)

Offset Address = 0x2000
Read

31:11	10	9	8	7:3	2	1	0
Reserved	Upstrm DCM Busy	Ingress TM DCM Busy	SERDES DCM Busy	Reserved	Upstrm DCM Locked	Ingress TM DCM Locked	SERDES DCM Locked

SERDES_DCM_Locked: When this bit is set, the DCM on the SERDES reference clock is locked.

Ingress_TM_DCM_Locked: When this bit is set, the DCM on the ingress TM clock is locked.

Upstrm_DCM_Locked: When this bit is set, the DCM on the upstream interface clock is locked.

SERDES_DCM_Busy: When this bit is set, the DCM on the SERDES reference clock is busy executing a phase shift command.

Ingress_TM_DCM_Busy: When this bit is set, the DCM on the ingress TM clock is busy executing a phase shift command.

Upstrm_DCM_Busy: When this bit is set, the DCM on the upstream interface clock is busy executing a phase shift command.

Error Status Register (Error_Stat)

Offset Address = 0x2002
Write/Read

All of the bits in this register are sticky. Any write to the register clears them. Once set, these bits stay set.

31:16	15	14	13:12	11:10	9:8	7:6	5:4	3:2	1	0
Reserved	ITM FIFO Ovf	Reserved	Egress Ovf(6)	Egress Ovf(5)	Egress Ovf(4)	Egress Ovf(3)	Egress Ovf(2)	Egress Ovf(1)	Queue Ovf(0)	Buffer Ovf(0)
0	0	0	00	00	00	00	00	00	0	0

ITM_Fifo_Ovf: When this bit is set, one of the ingress FIFO blocks has deleted a cell because of overflow. This condition is not normal and indicates an ingress TM not responding to flow control.

Egress_Ovf: Each egress FIFO structure provides two overflow status indications. Up to seven FIFO structures (servicing up to 56 ports in one FPGA) can reflect status through this register. Devices use only as many bits as are required. Unimplemented FIFO blocks return zero.

Queue_Ovf: When this bit is set, an output queue within this FIFO element exceeded the maximum programmed depth, indicating the depth threshold is set to low or the far-end device(s) are not responding to fabric flow control. It is not possible to determine which device as queues are shared by multiple devices.

Buffer_Ovf: When this bit is set, cell(s) were dropped because the shared buffer pool was exceeded, indicating the buffer full thresholds are set to high or far-end device(s) are not responding to flow control.

Device Structure Summary (Dev_Struct)

Offset Address = 0x2004
Read

31:9	8	7:4	3:0
Reserved	Deep Rams	Num FIFOs	Num Columns

Num_Columns: Valid values for this field are 3 (four columns) or 7 (eight columns). This field indicates the number of memory stripes used to make up the egress buffer. Software uses this field to verify how many buffers can be made. Normally only solutions using four or fewer total SERDES blocks return a value of 3.

Num_FIFOs: This field returns the number of egress FIFO structures used - 1. Because each FIFO structure can hold at most eight ports, this field returns (NUM_SERDES + 7) / 8.

Deep_Rams: This bit is set when each egress FIFO RAM has 8K entries. This bit is cleared when each FIFO RAM has 4K entries. It is set only for XC2VP4 and XC2VP7 solutions. Software uses this bit to determine the number of buffers that can be made.

SERDES Configuration Registers

SERDES_Config

Offset Address = 0x4000+
Write/Read

31:14	13	12:11	10	9	8	7:4	3	2	1	0
Reserved	Rx Swap	Loop back	Plus Align	Minus Align	Rx Reset	Reserved	Tx Swap	Tx Reset	Tx Inh	Power down
0	0	00	0	0	1	0	0	1	0	0

Each SERDES has a single configuration register. SERDES 0 is at the base of the block (address = 0x40), SERDES 1 is at address 0x41, up to SERDES n at 0x40 + n. Refer to the MGT specification for full details on these control pins, including timing and interactions.

Power_down: When this bit is set, the MGT is shut down in low-power mode.

Tx_Inh: When this bit is set, TXP/TXN are disabled (TXP = 1, TXN =0).

TxReset: When this bit is set, reset is applied to the Tx section of the SERDES.

TxSwap: When this bit is set, polarity of TXP/TXN is reversed.

RxReset: When this bit is set, reset is applied to the Rx section of the SERDES.

MinusAlign: When this bit is set, comma- causes 8B/10B alignment.

PlusAlign: When this bit is set, comma+ causes 8B/10B alignment.

Loopback: Refer to the MGT specification for details on this bit.

RxSwap: When this bit is set, the polarity of RXP/RXN is reversed.

SERDES Status Register

The following register belongs in the SERDES Status memory region.

Offset Address: 0x6000+
Write/Read

Each SERDES block consumes two bytes. SERDES 0 is located at address 0 with each SERDES spaced by two bytes.

31:13	12	11	10	9	8	7	6:5	4:3	2:0
Reserved	Symbol Error	Data CRC Error	Header CRC Error	Good Cells	Lock	Tx Buffer Error	Loss of Sync	Rx Buffer Status	Clock Correction Count
0	0	0	0	0	0				

Clock_Correction_Count: The MGT block reports this value. Refer to the MGT specification for details.

Rx_Buffer_Status: The MGT block reports this value. Bit 1 is set to indicate buffer overflow, and bit 0 is set to indicate the buffer is half full.

Loss_of_Sync: The MGT block reports this value. Refer to the MGT specification for details.

Tx_Buffer_Error: The MGT block reports this value. This bit is set when the transmit buffer overflows. If this condition occurs, MGT TX_RESET must be applied.

Lock: Recovered Locked flag from cell header received from far end. If set, both ends are in cell synchronization and can send data.

Good_Cells: When this bit is set, the SERDES is receiving good cells. A lock indication follows.

Header_CRC_Error: When this bit is set, at least one cell with a header CRC error was received. This sticky status flag does not clear until the status register is written. The value written is ignored. Any write clears this status.

Data_CRC_Error: When this bit is set, at least one cell with a data CRC error was received. This sticky status flag is cleared by writing the status register with any value.

Symbol_Error: When this bit is set, at least one symbol not in the MGT's 8B/10B table was received. This sticky status flag is cleared by writing the status register with any value.

Egress Buffer Management

Each egress FIFO uses several data structures for buffer and queue management. The first is a 2K x 16 buffer list that has one entry per buffer. Each entry points to the next buffer in a list. When a buffer is free, it points to the next free buffer. When a buffer is allocated onto an output queue, it points to the next buffer in that queue. The free list includes a head pointer, tail pointer, and depth register. The head points to the first free buffer, while the tail points to the last. The depth is set to the number of buffers in the list – 1 to prevent the hardware from fully popping the list, simplifying the allocation logic. Software initializes all of these registers.

The output queues are maintained in a block RAM. Each queue has an entry that includes head pointer, tail pointer, alignment, and depth. These fields work identically to the free list control fields. The alignment field controls which 16 bits within the 128-bit FIFO word hold the first SERDES word for a cell. These structures are initialized automatically.

The last structure is a 16-entry threshold RAM. Each entry holds a depth count and a priority mask. Entries must be placed in the RAM in descending order. For example, Entry 0 normally holds the total number of buffers in the list. Entry 1 holds the depth at which the first level of backpressure based on shared depth limits is crossed. Software can use as many or as few thresholds as are desired.

Two block RAMs hold the free list (again to save block RAMs), which limits the maximum number of buffers in a list to 2K. The full FIFO memory (8K entries) can hold more than 2K cells when the cell size is 48 bytes or smaller. The extra memory goes unused in this case.

To reduce the address location requirements on the DCR bus, the Buffer Entry List is accessed by writing to the Buffer Entry List Address register and then reading from or writing to the Buffer Entry List Data Register.

Note: The Free Head Pointer, the Free Tail Pointer, and the Free Depth Pointer registers are repeated one to three times based on the number of FIFO instances in design. Each set is selected using the PAGE_SELECT register.

Buff_Ent_Lst

Offset Address = 0x8000+
Write/Read

31:15	14	13:11	10:0
Reserved	Alloc	Align	Next_Buffer

Next_Buffer: This field is initialized by software to the index of the next buffer in the list.

Align: Software does not initialize this field. It is set by hardware automatically when a buffer is forwarded.

Alloc: Software initializes this bit to 0. This bit can be used to track lost buffers. Hardware sets this bit when a buffer is allocated. Hardware clears this bit when a buffer is returned to the free list. Software reads this bit to track for lost buffers.

Free_Head_Ptr

Offset Address = 0x9000
Write/Read

31:11	10:0
Reserved	Free List Head Pointer
0	00000000000

The CPU can write this register only when allocation is disabled.

Free_Tail_Ptr

Offset Address = 0x9002
Write/Read

31:11	10:0
Reserved	Free List Tail Pointer
0	00000000000

The CPU can write this register only when allocation is disabled.

Free_Depth

Offset Address = 0x9004
Write/Read

31:11	10:0
Reserved	Free List Depth
0	00000000000

The CPU can write this register only when allocation is disabled.

Free_Dph_Thres_Tbl

Offset Address = 0x9040+
Write only

31:11	10:0
Reserved	Depth Threshold
0	00000000000

Each entry in the table represents a depth threshold. The thresholds must be loaded into the table in descending order, where the largest depth limit occurs first. Further, the first entry (address 0) is the maximum depth of the free list. Additionally, the last entry (can be any address up to 15) *MUST* be 0.

The hardware keeps a flow control threshold level (the address into the table) that always corresponds to the current free depth being less than or equal to the threshold at current level and greater than the threshold at the next level. When the free depth goes below the next threshold, the level advances. Similarly when it is not less than the current threshold, the level decreases (unless it is already at 0).

Priority-based flow control is then sent to each SERDES serviced by this FIFO corresponding to the mask table contents at the current level. Thus each threshold can pass a unique priority mask, shutting off any mix of priorities.

Any number of thresholds from 1 to 15 is supported based on how software initializes the table.

Free_Dph_Msk_Tbl

Offset Address = 0x9060+
Write only

31:16	15:0
Reserved	Priority Mask

Priority Mask: This field contains a flow control mask per threshold for egress buffer memory depth limits. Bits map directly across, where bit 0 corresponds to priority 0 and so forth. When fewer than 16 priorities are used, the remaining bits are reserved.

Group ID Mapping Tables

ID_Map_Tbl

Offset Address = 0xA000+
Write/Read

31:16	15:0
Reserved	Local Loop MAP bit + SERDES MAP

The group ID for multicasting supports up to 4K entries. Each entry is programmed 16 bits at a time (matching the CPU interface width). Each set of 16 bits is paged to support indefinite extension as the number of SERDES blocks grows.

The full map requires one bit per SERDES block + one bit for the local loopback port. Bits are organized with bit 0 corresponding to SERDES 0 and continuing up. The local loopback bit is always the next bit up after all the SERDES bits. While this makes the software a bit more complex with the requirement for extracting and rearranging bit fields, it simplifies the hardware.

Pages beyond page 0 correspond to higher order bits in the map. Page 0 corresponds to bits 15 down to 0, page 1 to bits 31 down to 16, and so on.

Egress Buffer Control Initialization

A 2048-entry RAM holds one entry for each buffer. The content is a link to the next buffer in a list. Software has access to the full 2K entry table as well as the Free_Head_Ptr, Free_Tail_Ptr, and Free_Depth registers for the free list. It does not have access to any of the lists maintained for output queues.

The actual number of buffers that can be fit into the FIFO memory is a function of how much memory is available and the configured cell size. Table 3-4 shows how much egress memory is available and its width, based on selected device size.

Table 3-4: **Egress Memory and Width Based on Device**

Device	Memory Columns	Memory Rows	Bytes Per Row
XC2VP4	4	4K	8
XC2VP7	8	4K	16
All others	8	8K	16

Next calculate the number of memory rows required as a function of fabric cell size:

```
MEM_ROWS_PER_CELL = ROUND_UP(CELL_SIZE_IN_BYTES /
BYTES_PER_ROW)
```

Calculate the number of buffers in an egress FIFO:

```
NUM_BUFFERS = ROUND_DOWN(MEMORY_ROWS / MEM_ROWS_PER_CELL)
```

Each buffer is then numbered from 0 to (NUM_BUFFERS – 1). Program flow is as follows:

```
FREE_HEAD = 0
FREE_TAIL = NUM_BUFFERS – 1
FREE_DEPTH = NUM_BUFFERS – 1
For (I = 0; I < NUM_BUFFERS – 1; I++)
    Buffer_list(i) = I + 1
```

Any configuration that places four or fewer SERDES blocks into an egress FIFO uses only four columns and eight byte rows as opposed to eight columns. Software can interrogate the size of each FIFO instance via status registers.

Parameterization

The reference design includes several customizable parameters that control the port configuration and operating modes. All control constants guiding the design are in one package file: `switch_scale_pack.vhd`. The first section is made up of constants that a user can tune to change the characteristics of the switch. The second section gives many immediately derived types used throughout the switch. None of these should require modification. Table 3-5 describes all supported parameters.

Table 3-5: **Reference Design Parameters**

Constant	Valid Values	Effect
NUM_PORTS	1 to 256	This constant is the number of ports in a system. It can be any arbitrary value and represents the number of unique, physical destinations that are addressable by the switch fabric. There is a small gain in area by setting this number as low as possible.
NUM_SERDES	1 to 24	This constant is the number of SERDES blocks used for switch ports in the FPGA. It can be any arbitrary value that is equal to or less than the number of SERDES blocks available in the target FPGA device. This value is one less than NUM_PORTS for non-cascaded solutions, implying that all ports other than the local port (the one served by a given FPGA instance) are reached through this FPGA instance. This value is set to the number of ports reachable through this given FPGA instance for cascaded solutions. Different FPGA types can be used within one cascade chain. Different physical ports can use different cascading solutions. For that matter, different physical ports need not be fully populated (a non-full mesh).
NUM_CLASSES	1 to 16	This parameter is the number of different traffic priorities supported in the switch fabric. Currently CLASS is used identically to PRIORITY, and the scheduler uses strict selection. A higher priority number has priority over lower priority numbers. Lowering this number implies fewer physical pins to communicate flow control information as well as slightly less internal logic.

Table 3-5: **Reference Design Parameters *(Continued)***

Constant	Valid Values	Effect
NUM_GROUPS	1 to 4096	This parameter is the number of multicast groups supported in the switch fabric. This constant is currently dictated by the desire to use as few block RAMs as possible for mapping between GROUP_ID and SERDES select map. The number of bits required to encode the group ID (currently 12) + the number of bits required to encode the traffic priority (currently 4) needs to be 16 or fewer. The actual split can be moved around (that is, 2K groups and 32 priorities or 8 priorities and 8K groups) because of the space available in the system cell header. Even that is easily changed. If 64-bit system buses are used, more header bits are available since the cell header always consumes one full system word. The fabric cell header has four spare bits already, so the total bit count can grow to 20 bits with no effect on the fabric cell. Changes beyond that size are more difficult to implement as the fabric cell header needs to grow.
T_Q_DEPTH	1 to 255	This parameter is the maximum depth supported in the design for any output queue. This constant is separate from the run-time configuration register that limits depth. It represents the width of the depth field implemented in the queue structures. There is no reason to change this value. Instead use the run-time programmable register to limit queues to fewer cells.
I_FC_BITS	1/2 to 2x	While expressed as a free range natural, this constant is set to one of three values: • NUM_CLASSES / 2: indicates that the flow control channel to the ingress TM is one-half the number of priorities. For example a 16-priority switch has eight flow control pins. Thus the total flow control communication time is at least NUM_PORTS * 2. • NUM_CLASSES: indicates that the flow control channel has one pin per priority. Communication time is at least NUM_PORTS. • 2*NUM_CLASSES: Indicates that the flow control channel has two pins per priority. Communication time is at least NUM_PORTS / 2. This provision allows a little flexibility between pin count and flow control communication time. TM designs that can tolerate more latency can save pins.

Table 3-5: **Reference Design Parameters *(Continued)***

Constant	Valid Values	Effect
I_FC_FOR_MCAST	FALSE, TRUE	If set to TRUE, the flow control sequence sent between the FPGA and ingress TM includes one extra word after all ports that is an or-reduction of all the ports. The ingress TM uses it to shut-off multicast flows. If set to FALSE, the flow control sequence only includes one entry for each port.
INGRESS_FIFO_FULL_THRESHOLD	0 to 150	The ingress FIFO is a single 512-word block RAM. It is split evenly into three sections of 170 entries. It is guaranteed to be drained by 16 bits at a time at 156.25 MHz (or the SERDES clock rate). The ingress FIFO is filled no faster than 32 bits at a time (or 64) at 200 MHz (or the TM clock rate). This threshold represents the point at which data is not written to the ingress FIFO and an overflow error is generated. This condition normally does not happen because it indicates ingress flow control was ignored by the TM. If data is discarded, cells are corrupted but normal flow resumes once the FIFO drains. This parameter is a hard constant in the design as opposed to run-time programmable in order to save implementation space. It need not be programmable.
INGRESS_FIFO_FC_THRESHOLD	0 to 85	This threshold represents the fill level at which flow control is generated back to the ingress TM for all priorities of traffic on this port. It is a hard constant, not programmable, which is not an issue, given the known fill and drain rates of the FIFO. Data is always taken, if available, in the FIFO. Backpressure received from the SERDES stops the ingress TM but does not stop sourcing data from this FIFO, meaning that the value must be tuned so that line-rate flows can be maintained for a reasonable mix of payload sizes without causing undue egress buffer depth limits. The higher the threshold, the higher the buffering required at the egress. The lower the threshold, the greater the likelihood that line-rate flow cannot be guaranteed because of under-running the SERDES. Data is stored in the FIFO efficiently. Thus unfilled cells consume only as many words as are required. In other words, it is not possible to know how many cells are in the FIFO at any time, only how much data.

Table 3-5: **Reference Design Parameters *(Continued)***

Constant	Valid Values	Effect
DEVICE_IS_P7_OR_LESS	FALSE, TRUE	This parameter controls the egress buffer memory structure. Set this flag to TRUE if targeting a XC2VP4 or XC2VP7 device. Set this flag to FALSE for all other targets. If not set correctly, PAR fails because of running out of block RAMs.
ALLOW_PER_PORT_QUEUES	FALSE, TRUE	When FALSE, all ports served by an egress FIFO share a common set of per-priority output queues. When TRUE, each port has a unique set of per-priority output queues. ***Note:*** The current implementation supports enough memory storage for independent queues but only generates shared queues.
T_OUTPUT_QUEUE_NUM	0 to 512	This parameter controls the width of internal fields used to store queue numbers. Set this parameter to the maximum number of ports per egress FIFO * NUM_CLASSES if output queues per port are used. Otherwise set it to NUM_CLASSES.
TOP_SERDES_NUM	0 to NUM_SERDES – 1	This parameter sets the number of MGTs to be located on the top edge of the FPGA. If non-zero, then the top MGTs are numbered 0 to TOP_SERDES_NUM – 1. If zero, then no MGTs are used on the top of the FPGA.
BOT_SERDES_NUM	0 to NUM_SERDES – 1	This parameter sets the number of MGTs to be located on the bottom edge of the FPGA. If non-zero, then the bottom MGTs are numbered TOP_SERDES_NUM to NUM_SERDES – 1. If zero, then no MGTs are used on the bottom of the FPGA.
EDGES	0 or 1	A zero value means only one edge of the FPGA is using MGTs. A one value means both the top and bottom edges are using MGTs.
TOP_IS_BREFCLK	TRUE or FALSE	When TRUE, top MGTs use BREFCLK. When FALSE, top MGTs use BREFCLK2.
BOT_IS_BREFCLK	TRUE or FALSE	When TRUE, bottom MGTs use BREFCLK. When FALSE, bottom MGTs use BREFCLK2.

Implementation Size Based on SERDES and FPGA Selection

Table 3-6 shows the maximum number of flip-flops (FFs), look-up tables (LUTs), block RAMs (BRAMs), and multi-gigabit transceivers (MGTs) that can be implemented in the mesh fabric reference design based on the number of selected SERDES blocks, the number of ports, and the Xilinx FPGA. The maximum number of priorities is also listed.

Table 3-6: **Maximum FFs, LUTs, BRAMs, and MGTs Based on SERDES and FPGA**

Number of SERDES Blocks	Number of Ports	Xilinx FPGA	Number of Priorities	FFs / %	LUTs / %	BRAMs / %	MGTs / %
20	20	XC2VP100	16	10998 / 12%	16143 / 18%	218 / 49%	20 / 100%
20	20	XC2VP70	16	10998 / 16%	16143 / 24%	218 / 66%	20 / 100%
16	16	XC2VP50	16	8521 / 18%	12287 / 26%	148 / 63%	16 / 100%
12	12	XC2VP40	16	7119 / 18%	10209 / 26%	145 / 75%	12 / 100%
8	8	XC2VP30	16	4608 / 16%	6548 / 23%	75 / 55%	8 / 100%
8	8	XC2VP20	16	4608 / 24%	6546 / 35%	75 / 85%	8 / 100%
8	8	XC2VP7	16	4608 / 46%	6544 / 66%	43 / 97%	8 / 100%
4	4	XC2VP4	16	3007 / 49%	4096 / 68%	24 / 85%	4 / 100%

Table 3-7 shows each module block's utilization. The implementation size is for the Demo system, which uses four MGTs in a XC2VP50. Certain blocks, mostly on the egress side, change in size depending on the number of MGTs and what size device is being used.

Table 3-7: **Block by Block Utilization**

Module	FG	DFF	MEM16	BRAM	DPRAM	MULT	# Times
cell_tx	125	80	16	0	0	0	4
cell_rx	138	105	0	0	0	0	4
itm_fifo_per_pair	247	242	9	1	24	0	2
egress_fifo	331	381	0	32	0	0	1
egress_sched	83	135	0	0	8	0	1
egress_q	539	596	38	3	20	1	1
etm_if	218	317	14	1	33	1	1
itm_if	63	154	32	2	0	0	1
itm_fc	113	140	0	0	17	0	1
cpu_if	348	308	0	0	0	0	1
mesh_switch	61	87	0	0	0	0	1
Total	**3302**	**3342**	**166**	**40**	**126**	**2**	

Design Tools

The recommended tools and their versions as of this printing are summarized below:

- Synthesis = Synplify Pro 7.2.2
- Simulation = ModelSim SE Plus 5.7a (VHDL license)
- Xilinx ISE = 6.1 SP 3
- Xilinx EDK = 6.1 EDK_G.14

Refer to the `readme.txt` file included with the source code for the most up-to-date information.

Chapter 4

Performance Characteristics

This chapter contains the following sections:

- "Performance"
- "Clocking"

Performance

Figure 4-1 shows the data path through the mesh fabric reference design and the number of clocks it takes to pass through each module. The clock numbers come from examining a simulation waveform and tracing one cell through the different modules and clock domains. In the itm_fifo and egress_fifo modules, there are two clock domains. On the input side of the data path, the data goes directly to a block RAM that stores the data in the input clock domain. On the output side of the data path, the data is removed from the block RAM in the output clock domain and processed.

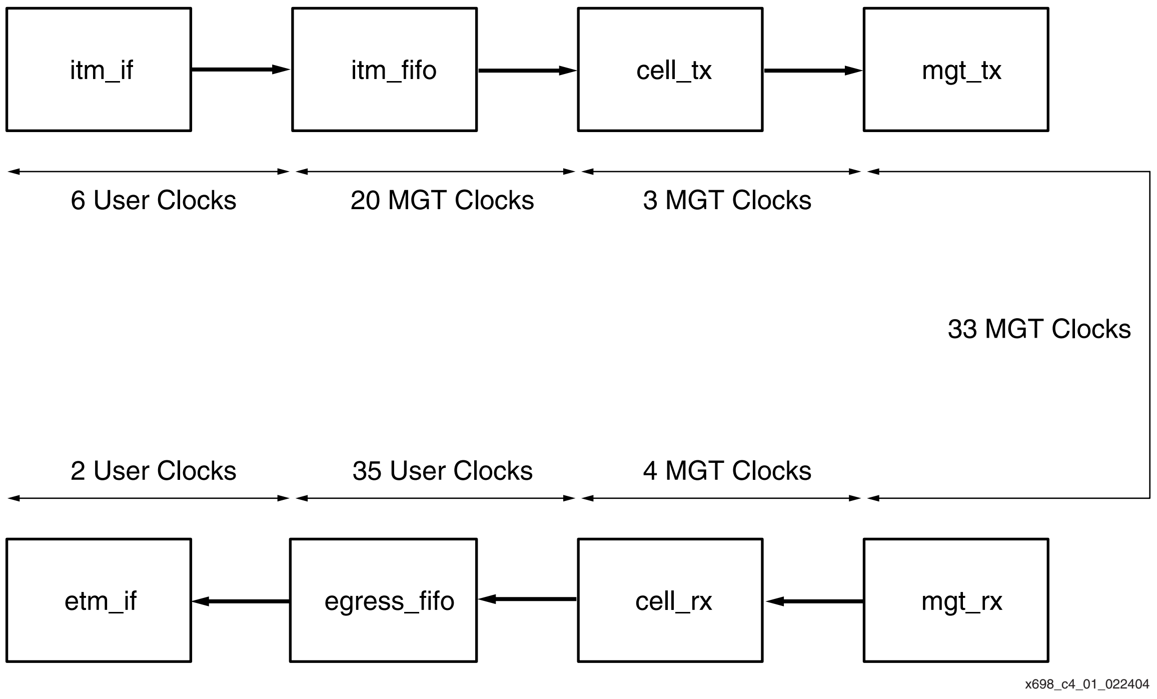


Figure 4-1: **Data Path through Mesh Fabric Reference Design**

The total time through the MFRD in the demo system, which uses a 100 MHz system clock, is on the order of 800 – 1000 ns, with the average at 870 ns, according to the timestamping analysis done by the demo software for cells to the upstream FPGA only. The demo system uses a 100 MHz user clock, because of limitations in some of the EDK components and Local Link FIFOs. However, if the MFRD system clock is run at its maximum speed of 200 MHz, the total time through the system can be extrapolated. By taking the clock cycles and frequency used for each clock, the typical time can be calculated for different frequencies.

Table 4-1 shows the calculated number of user clocks and SERDES clocks through the different modules.

Table 4-1: **Total User and SERDES Clocks**

RTL Module	Calculated User Clocks	Calculated SERDES Clocks
Itm_if	6	-
Itm_fifo	1	20
Cell_tx	-	3
MGT TX and RX	-	33
Cell_rx	-	4
Egress_fifo	35	1
Etm_if	2	-
Total	**44**	**61**

Based on a total of 61 clocks through the SERDES clock domain, the calculated time through the SERDES clock domain at 156 MHz is 390 ns (see Table 4-2).

Table 4-2: **SERDES Clock Domain Timing**

Clock Cycles	Calculated Time at 156 MHz
SERDES Clock Domain (61 clocks)	390 ns

Based on a total of 44 clocks through the user clock domain, the calculated time through the user clock domain at 100 MHz is 440 ns and at 200 MHz is 220 ns (see Table 4-2).

Table 4-3: **User Clock Domain Timing**

Clock Cycles	Calculated Time at 100 MHz	Calculated Time at 200 MHz
User clock domain (44 clocks)	440 ns	220 ns

Thus the total time through both the SERDES and user clock domains is 830 ns at 100 MHz and at is 610 ns 200 MHz.

Table 4-4 shows the observed maximum, minimum, and average values for upstream and downstream loopback.

Table 4-4: **Observed Upstream and Downstream Loopback Values**

FPGA	Observed Max (ns)	Observed Min (ns)	Observed Avg (ns)
Upstream Loopback	940	780	868
Downstream Loopback	1100	940	1008

Statistics gathered from 200 cells sent to one destination port using the Demo software to gather statistics. On the Upstream test, they were sent to Port 1, and on the Downstream test, they were sent to Port 5. The Demo uses a 100 MHz user clock and 156 MHz SERDES clock.

More detailed performance analysis, including the effects of flow control and cascade traffic still need to be performed and will be provided at a later time.

Clocking

Figure 4-2 shows a block diagram of the clocking scheme for the mesh switch design. In the figure, a dashed box indicates an optional component, where the label on the box is the VHDL generic that controls whether the component is used.

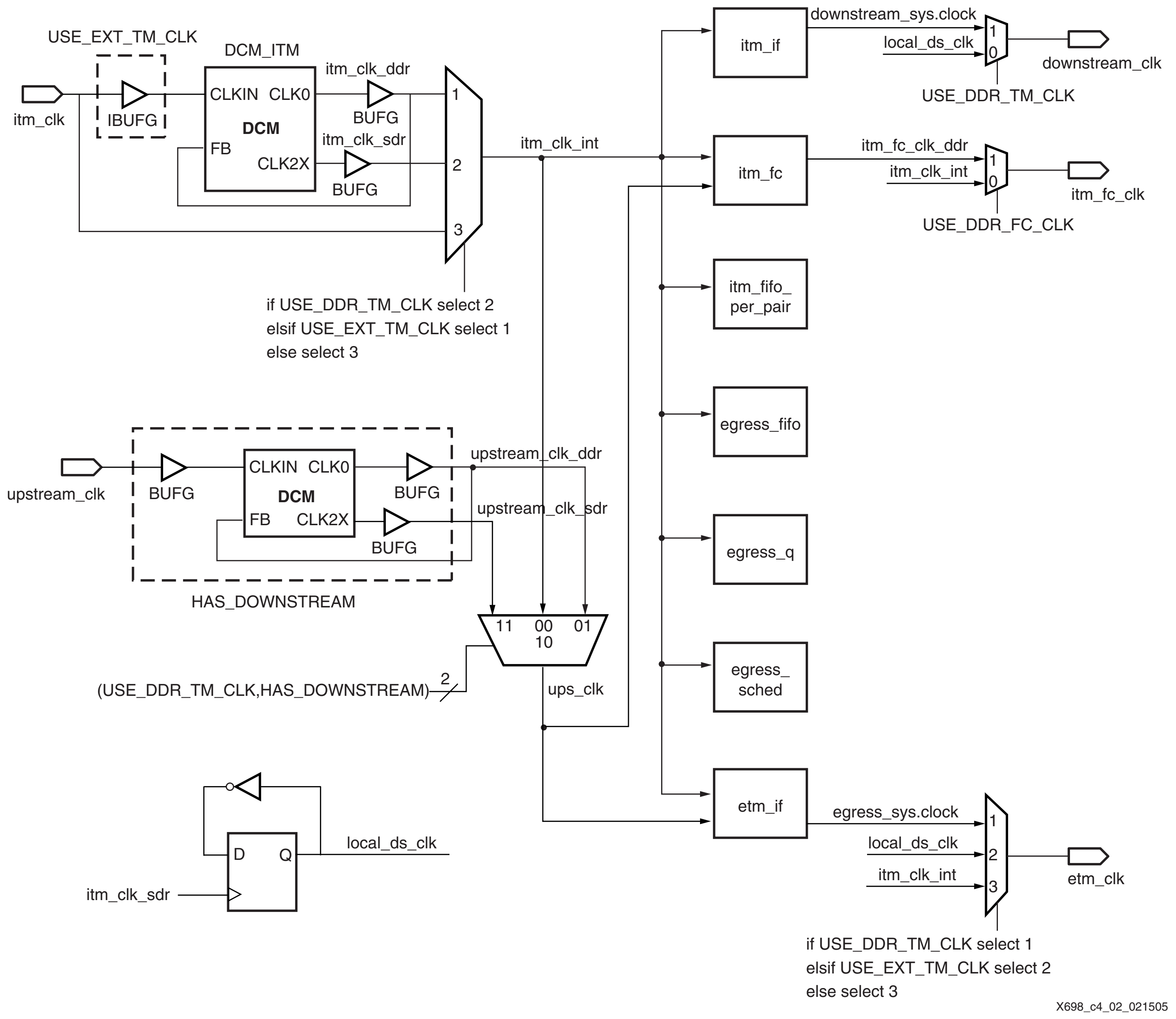


Figure 4-2: **Mesh Switch Clocking Scheme**

Chapter 5

MFRD Demo

This chapter contains the following sections:

- "Demo Components"
- "Demo Configuration"
- "Pin Descriptions for the Demo System"
- "Demo Operation"
- "Demo Reference Files"

Demo Components

To demonstrate the Mesh Fabric design working in a system, a simple subsystem was put together using two ML323 boards and various IP and reference designs. Figure 5-1 shows a high-level block diagram of the system.

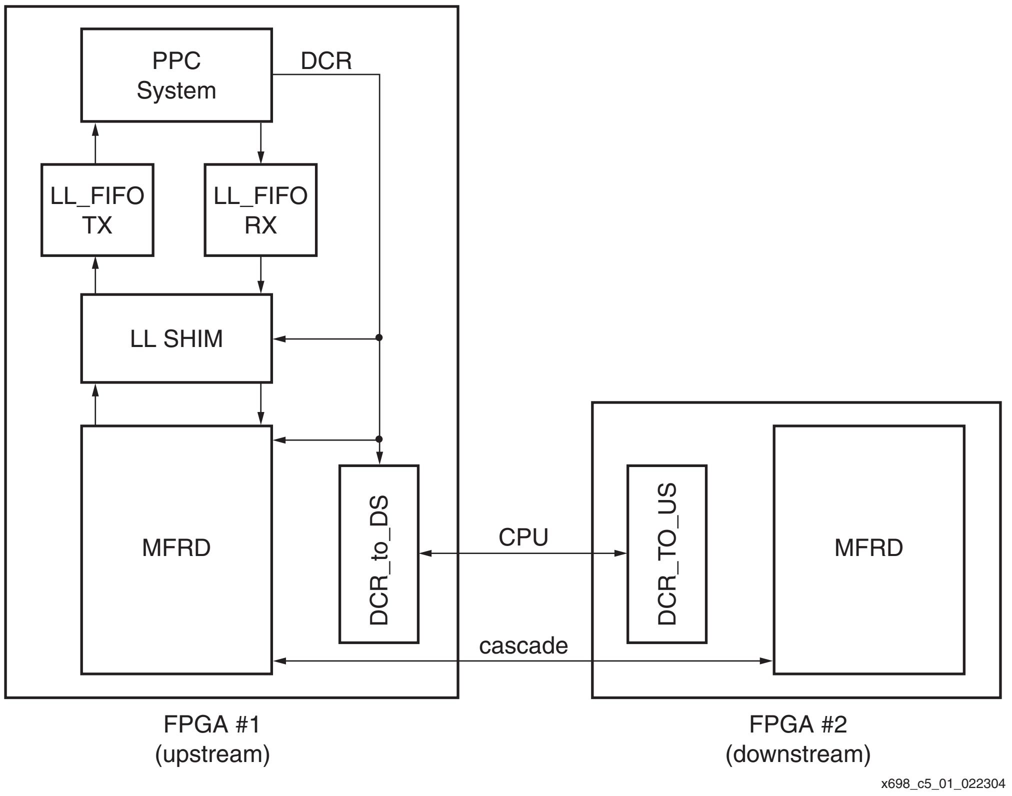


Figure 5-1: **Mesh Fabric System Block Diagram**

The components of the system are as follows:

- FPGA Components, Intellectual Property, and Reference Designs
  - A Mesh Fabric Reference Design (XAPP698)
  - An Embedded Development Kit (EDK) system including:
    - Uartlite PowerPC peripheral (included with EDK)
    - PLB to Local Link PowerPC peripheral (XAPP698)
    - DCR Slave PowerPC peripheral (included with the EDK)
    - Local Link Configurable FIFO (XAPP698)
    - Local Link to MFRD Shim (XAPP698)
    - DCR to Downstream FPGA design (XAPP698)
    - DCR to Upstream FPGA design (XAPP698)
- Software Components
  - Demo software
  - HyperTerminal (or other terminal program, such as Tera Term)
  - Xilinx EDK (not needed to run demo, only for making changes to it)
  - Xilinx ISE (not needed to run demo, only for making changes to it)
- Hardware Components
  - Single Board Option
    - 1 ML323 board
    - 8 SMA cables for MGT interconnect/loopback
    - 2 SMA cables for clock distribution
  - Two Board Option
    - 2 ML323 boards
    - 16 SMA cables for MGT interconnect/loopback
    - 4 SMA cables for clock distribution
    - 4 ribbon cables (SAMTECH P/N HCSD-32-D-13.00-01-T-N-G
    - 8 standoffs
  - Common
    - Xilinx parallel IV download cable or SystemACE card
    - null-modem serial cable

Demo Configuration

The MFRD demo has two ML323 boards, each containing a Xilinx XC2VP50 device. These boards are stacked and connected with ribbon cables to emulate a multiple-FPGA Mesh Fabric system. Each XC2VP50 uses four multi-gigabit transceivers (MGTs), for a total of eight MGTs in the system. Each MGT can be connected to an MGT on another board or can be looped back to itself. In this demo, all MGTs are looped back. This demo also can be used with one ML323 board, but the cascade capability is not shown for this case.

PPC System

The processor system consists of an EDK project, described in `system.mhs`, which connects the PowerPC processor to the following components: a DCR bus, a Uartlite component, and a custom module called `plb_ll_if`. These components allow the MFRD and Local Link Shim to communicate with the processor and the user. The PowerPC processor is configured to run at 200 MHz, and the DCR Bus operates at 50 MHz.

The MFRD demo uses the PowerPC processor inside the Virtex-II Pro FPGA to run a menu program that guides the user to set up and run the demo. The processor also acts as a simple traffic generator to send cells of various sizes and priorities to the different ports that the MFRD addresses. In addition, the processor receives the looped back data, reads two timestamps, and calculates the minimum, maximum, and average round trip times for each priority and group ID combination.

Local Link FIFOs

Because the processor cannot keep up with the MFRD performance, it requires two FIFOs to store data transfers between the two modules. The RX Local Link FIFO receives cell data from the processor and stores it until the processor has finished writing all the cells. Then the processor sets the PPC_done bit in the Shim DCR space, which is the signal for the MFRD to start emptying the FIFO. The TX Local Link FIFO receives looped back cells from the MFRD and stores them until the processor can access them. The Local Link FIFOs run at 50 MHz on the processor side and 100 MHz on the MFRD side.

Local Link Shim

The Shim module connects the MFRD to the processor system via the FIFOs. Its DCR registers configure and control the demo and status registers, which allow the processor to monitor the demo. The Shim also takes data from the RX Local Link FIFO, when the PPC_Done bit is set, and provides the itm_data and itm_soc signals to the MFRD. In the reverse direction, the Shim takes the etm_data and etm_soc signals and provides Local Link signals to the TX Local Link FIFO.

The Shim also is responsible for inserting timestamp information into the data stream, both coming out of the RX Local Link FIFO and going into the TX Local Link FIFO. The timestamp comes from a free-running 16-bit counter, running at 100 MHz. Each count represents 10 ns. The Shim generates the first timestamp, which is inserted into the data received from the RX Local Link FIFO, right after the header word. The Shim also generates the second timestamp, which is inserted into the TX Local Link data, right behind the first timestamp. These two timestamps can calculate the time for the ITM data to go through the MFRD, loop through the SMA cables, return to the MFRD, and exit on the ETM port. The difference in the two timestamps is multiplied by 10 ns to get the total time through the system. To turn the timestamp feature on, software writes to ITM_Control_Reg and ETM_Control_Reg and sets their appropriate bits.

Figure 5-2 provides a timestamp example.

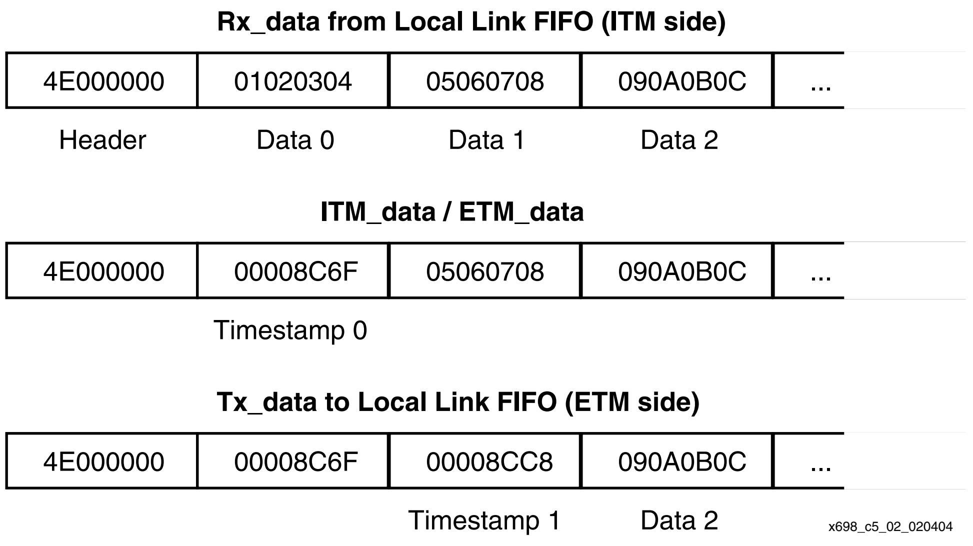


Figure 5-2: **Timestamp Details**

Table 5-1 summarizes the DCR Shim registers.

Table 5-1: **Shim DCR Registers**

DCR Address[3:0]	Register Name	R/W	Register Description
0000	DCR_Reset_Reg	R/W	0: Reset All '1' – Reset active '0' – Reset inactive
		R/W	1: Reset Shim '1': Reset active '0': Reset inactive
		R/W	2: Reset MFRD '1': Reset active '0': Reset inactive
		R/W	3: Reset RX EOF Counter '1': Reset active '0': Reset inactive
		R/W	4: Reset External '1': Reset active '0': Reset inactive
			31 - 5: Reserved

Table 5-1: **Shim DCR Registers *(Continued)***

DCR Address[3:0]	Register Name	R/W	Register Description
0001	DCR_ITM_Ctrl_Reg	R/W R/W R/W R/W R/W R/W	0: Single Cell (automatically resets when finished) '1': Take one cell from ITM LL FIFO '0': Do not use single cell method 1: Use_Len_FIFO '1': Use tx_len_ready_out to take cells from LL FIFO '0': Do not use length FIFO 2: Use_Mask '1': Read FIFO when EOF_Count = ITM_Count_Mask '0': Do not use mask to read FIFO 3: ITM Timestamp '1': Use the ITM timestamp '0': Do not use timestamp on ITM 4: Use PPC_Done (automatically resets when finished) '1': Software done writing to FIFO, time to empty FIFO '0': Wait for software to set this bit 5: Shim_loopback '1': Loops data on ETM ports back to ITM ports '0': No loopback Priority (in case of conflict): Single_cell, then length FIFO, then Mask, then PPC_Done.
0010	DCR_ETM_Ctrl_Reg	R/W	0: ETM timestamp '1': Use the ETM timestamp '0': Do not use the ETM timestamp 1 – 31: Reserved
0011	DCR_TX_Cell_Count	R	0: 11 Holds TX cell count, up to 4096 cells. Can make larger. 12 – 31: Reserved
0100	DCR_Dropped_Cell_cnt	R	0 – 11: Holds dropped cell count, up to 4096 cells. Probably too large. 12 – 31: Reserved
0101	DCR_ETM_FC_Reg	R/W	0 – 15: Holds Flow control from ETM to MFRD 16 – 31: Reserved
0110	DCR_ITM_FC_Reg	R R R	0 – 15: Holds Flow control priorities from MFRD to ITM 16 – 24: Holds binary encoded port information (maximum 256 ports) 25 – 31: Reserved
0111	DCR_EOF_Count	R	0 – 11: Number of full cells in RX FIFO 12 – 31: Reserved

Table 5-1: **Shim DCR Registers *(Continued)***

DCR Address[3:0]	Register Name	R/W	Register Description
1000	DCR_ITM_Count_Mask	R/W	0 – 11: Number of full cells before reading FIFO 12 – 31: Reserved
1001	DCR_shim_DCM_status	 R R R R	If configuration of the MFRD or Shim makes a DCM obsolete, then its lock signal is forced to a locked position. 0: TX DCM lock signal 1: Shim DCM lock signal 2: ITM FC clock lock signal 3: Downstream FPGA lock signal 4 – 31 Reserved
1010	DCR_Reset_shim_dcms	R/W R/W R/W	0: ITM DCM reset in shim 1: ETM DCM reset in shim 2: ITM_FC DCM reset in shim 3 – 31: Reserved
1011	DCR_HW_Addr_Reg	R	7: 0 Hardware Address read from I/O pins (ATCA demo version only)

DCR Modules

To configure the downstream MFRD, DCR transactions must be relayed from the upstream FPGA to the downstream FPGA, and the resulting Acknowledge from the downstream FPGA must be received by the PowerPC processor on the upstream FPGA within 16 clocks of issuing the request. Because few I/O pins are available on the ML323 headers after assignment of the MFRD cascade interface pins, the DCR signals must be encoded into a smaller number of signals. This encoding is done in the DCR_to_DS and DCR_to_US modules.

These modules take the DCR Address, Read, Write, and Data signals and encode them into a sequence of byte-wide transfers. Since the MFRD uses 16-bit address and data buses for its internal registers, only the lowest 16 bits of the DCR Address and DCR Data must be sent. Furthermore, in this demo only the least significant byte is used for DCR Address, so only one clock cycle is needed to transmit the DCR Address to the downstream FPGA. To indicate the start of a CPU transaction, the cpu_tx_sof_n signal is held Low for one clock cycle. During this time, the cpu_tx_data signal contains the control byte (CON).

Table 5-2 lists the assignment of the bits in the control byte.

Table 5-2: **CON Bit Assignments**

Bit Number	Function
0	DCR Read
1	DCR Write
2	DCR Acknowledge (Ack)
3	Parity bit (even)
4 – 7	Reserved

The Low byte of the DCR address (AL) follows the control byte. If the transaction is a write, then two more bytes are sent, corresponding to the High and Low data bytes (DH and DL). If the transaction is a read, then the sequence ends with AL. The downstream FPGA translates the received CPU signals into true DCR commands inside the DCR_to_US module. On a write, the downstream FPGA responds with an Ack (CON = 0x0C). On a read, the downstream FPGA responds with an Ack followed by High and Low data bytes. The write and read transactions are illustrated in Figure 5-3 and Figure 5-4, respectively.

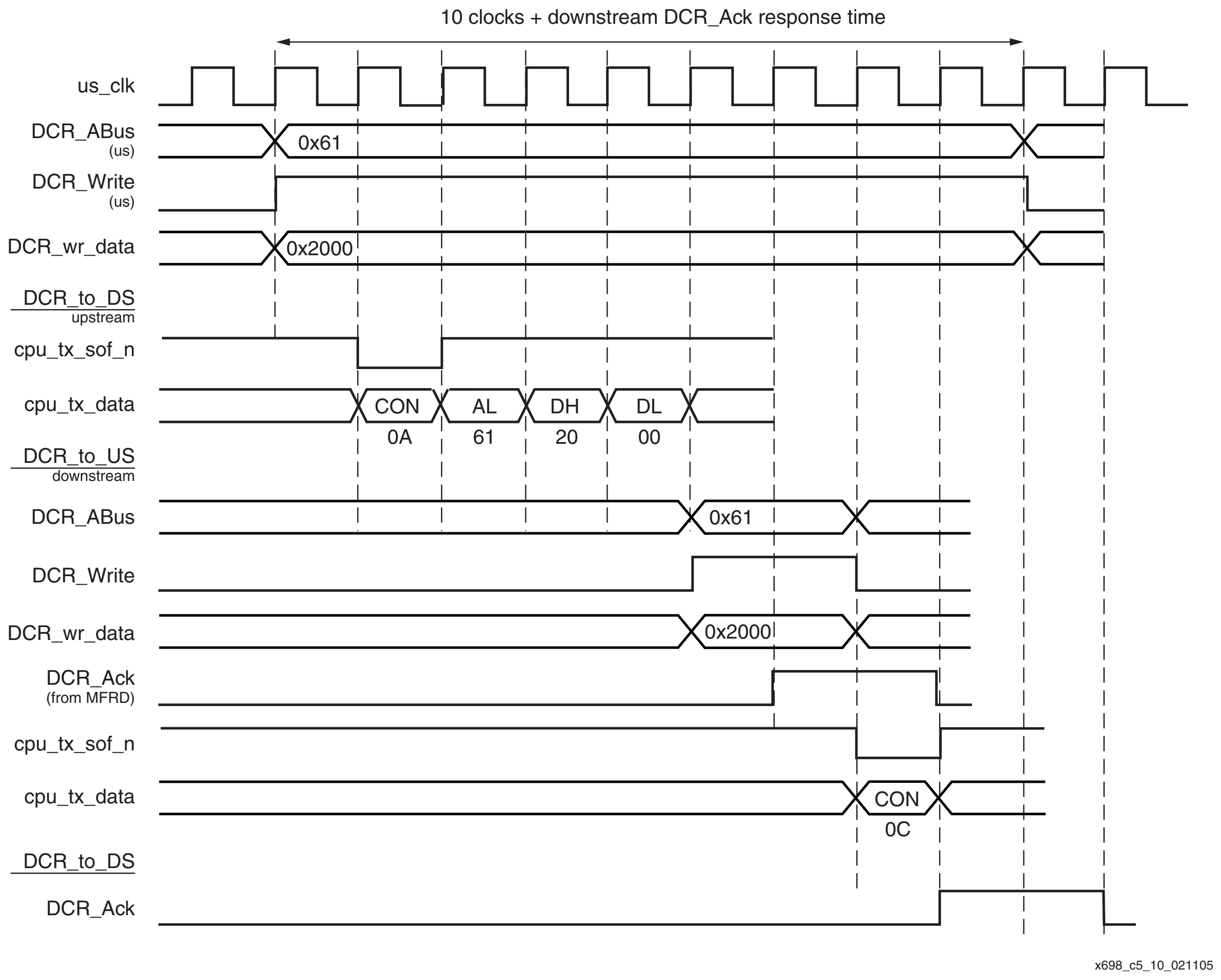


Figure 5-3: **CPU Communication between Upstream and Downstream FPGAs (Write)**

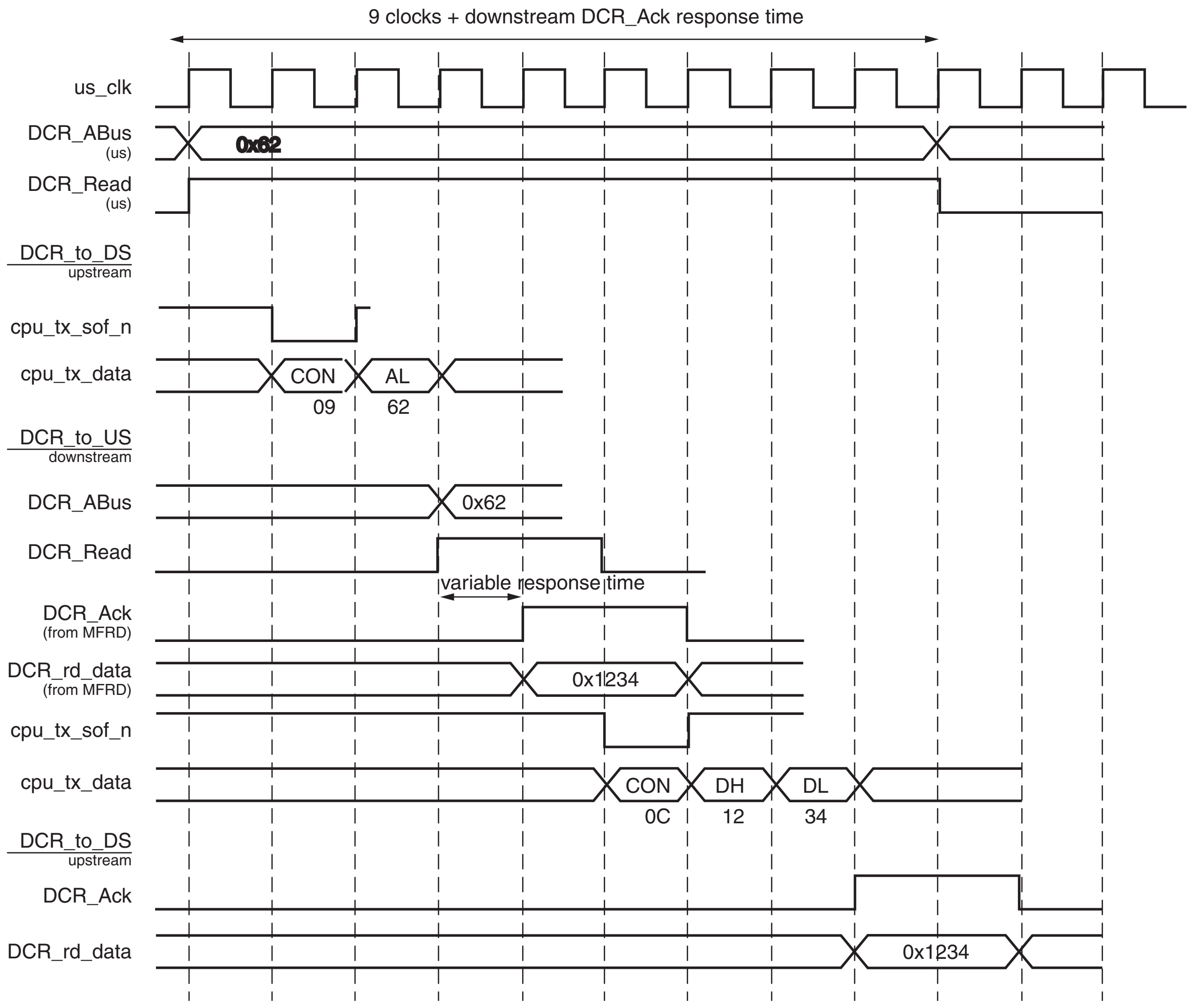


Figure 5-4: **CPU Communication between Upstream and Downstream FPGAs (Read)**

When the upstream FPGA issues the CPU transaction to the downstream FPGA, it waits for the downstream FPGA to reply with an Ack, and, in the case of a read transaction, it also expects two bytes of data. When the Ack comes from the downstream FPGA, the upstream FPGA issues a DCR_Ack to the processor. If this process takes longer than 16 clocks, then the processor times out and moves on to the next instruction.

The clock used on the downstream FPGA for the CPU interface is a 50 MHz clock, provided by the upstream FPGA via the ribbon cable. This clock is the same one used on the upstream FPGA from the PowerPC system. The rest of the downstream MFRD is clocked from the 100 MHz downstream_clk, provided by the upstream FPGAs cascade interface, which also is delivered through the ribbon cable. The upstream FPGA also provides a reset signal, controllable through a Shim DCR register.

Pin Descriptions for the Demo System

Table 5-3 provides the pin list for the MFRD demo system. The table lists the pinouts for the Upstream FPGA and the Downstream FPGA.

Table 5-3: **MFRD Reference Design Pinout**

Signal	Type	Description
Upstream FPGA Pin-out (mesh_switch_hw_test_has_ds.vhd)		
Serdes_refclk_p[0:1], Serdes_refclk_n[0:1]	Input	Differential reference clock for the MGTs, one clock each for the top and bottom MGTs. The clock frequency is 156 MHz.
Out_refclk_p, Out_refclk_n	Output	Differential reference clock output to connect with serdes_refclk_p[1] and serdes_refclk_n[1].
Shim_clk	Input	Clock from the EDK system. The clock frequency is 50 MHz.
Sys_reset	Input	System-wide asynchronous reset.
TXP[0:3], TXN[0:3]	Output	Differential MGT transmit signals.
RXP[0:3], RXN[0:3]	Input	Differential MGT receive signals.
ITM_to_FIFO_ll_clk	Input	50 MHz clock used between the EDK system and the RX Local Link FIFO. This clock is used to snoop on activity (optional).
ITM_to_FIFO_sof_n	Input	Start of Frame between the EDK system and the RX Local Link FIFO. This signal is used to snoop on activity (optional).
ITM_to_FIFO_eof_n	Input	End of Frame between the EDK system and the RX Local Link FIFO. This signal is used to snoop on activity (optional).
ITM_to_FIFO_src_rdy_n	Input	Source Ready signal between the EDK system and the RX Local Link FIFO. This signal is used to snoop on activity (optional).
FIFO_to_ITM_dst_rdy_n	Input	Destination ready signal between the EDK system and the RX Local Link FIFO. This signal is used to snoop on activity (optional).
RX_clk	Output	100 MHz clock used on the read side of the RX Local Link FIFO.
RX_dst_rdy_n	Output	Local Link signal that indicates the Shim is ready to receive data from the RX Local Link FIFO.
RX_src_rdy_n	Input	Local Link signal that indicates the RX Local Link FIFO is ready with data for the Shim.

Table 5-3: **MFRD Reference Design Pinout *(Continued)***

Signal	Type	Description
RX_data[31:0]	Input	Local Link data from the RX Local Link FIFO to the Shim.
RX_sof_n	Input	Local Link signal indicating the first word of the Local Link transfer from the RX FIFO to the Shim.
RX_eof_n	Input	Local Link signal indicating the last word of the Local Link transfer from the RX FIFO to the Shim.
RX_rem[3:0]	Input	Local Link signals indicating which bytes of the last word are valid.
TX_clk	Output	100 MHz clock from the Shim to the write port of the TX Local Link FIFO.
TX_dst_rdy_n	Input	Local Link signal that indicates the Shim is ready to transmit data to the TX Local Link FIFO.
TX_src_rdy_n	Output	Local Link signal that indicates the TX Local Link FIFO is ready to receive data from the Shim.
TX_data[31:0]	Output	Local Link data to the TX Local Link FIFO from the Shim.
TX_sof_n	Output	Local Link signal indicating the first word of the Local Link transfer to the TX FIFO.
TX_eof_n	Output	Local Link signal indicating the last word of the Local Link transfer to the TX FIFO.
TX_rem[3:0]	Output	Local Link signals indicating which bytes of the last word are valid to the TX FIFO.
Downstream_clk	Output	100 MHz clock to the downstream FPGA for the downstream data and the SOC.
Downstream_soc	Output	Start of Cell indicator for the downstream data.
Downstream_data[31:0]	Output	Data to the downstream FPGA.
Down_fc[0:15]	Input	Flow control from the downstream FPGA.
Down_fc_start	Input	Signal that indicates which down_fc value represents flow control for Port 0.
Upstream_clk	Input	100 MHz clock from the downstream FPGA for the upstream data and the SOC.
Upstream_soc	Input	Start of Cell indicator for the upstream data.
Upstream_data[31:0]	Input	Data from the downstream FPGA.

Table 5-3: **MFRD Reference Design Pinout *(Continued)***

Signal	Type	Description
Upstream_fc_off[0:15]	Output	Flow control data from the downstream FPGA.
DCR_Clk	Input	50 MHz clock from the EDK system.
CPU_dcrABus[9:0]	Input	DCR address bus from the EDK system.
CPU_dcrRead	Input	DCR read signal.
CPU_dcrWrite	Input	DCR write signal.
CPU_dcrDBusIn[31:0]	Input	DCR input data bus.
DCR_cpuDBusOut[31:0]	Output	DCR output data bus.
DCR_Ack	Output	DCR acknowledge signal.
Reset_ext	Output	Reset to external devices, such as LL FIFOs and downstream FPGAs, controllable through the Shim DCR.
Cpu_tx_data[7:0]	Output	CPU data to the downstream FPGA
Cpu_tx_sof_n	Output	Active-low start-of-frame signal to the downstream FPGA. Asserted to indicate the CON byte.
Cpu_rx_data[7:0]	Input	CPU data from the downstream FPGA.
Cpu_rx_sof_n	Input	Active-low start-of-frame signal from the downstream FPGA. Asserted to indicate the CON byte.
Us_clk	Output	50 MHz CPU clock to the downstream FPGA.
Ds_ready	Input	Lock signal from the downstream FPGA's DCM.
Error	Output	Asserted to indicate a parity error occurred in the CON byte.
Downstream FPGA Pin-out (mesh_switch_hw_test_fpga2.vhd)		
Serdes_refclk_p[0:1], Serdes_refclk_n[0:1]	Input	Differential reference clock for the MGTs, one clock each for the top and the bottom MGTs. The clock frequency is 156 MHz.
Out_refclk_p, Out_refclk_n	Output	Differential reference clock output to connect with serdes_refclk_p[1] and serdes_refclk_n[1].
Sys_clk	Input	50 MHz CPU clock from the upstream FPGA.
Sys_reset	Input	Reset from the upstream FPGA.
TXP[0:3], TXN[0:3]	Output	Differential MGT transmit signals.

Table 5-3: **MFRD Reference Design Pinout *(Continued)***

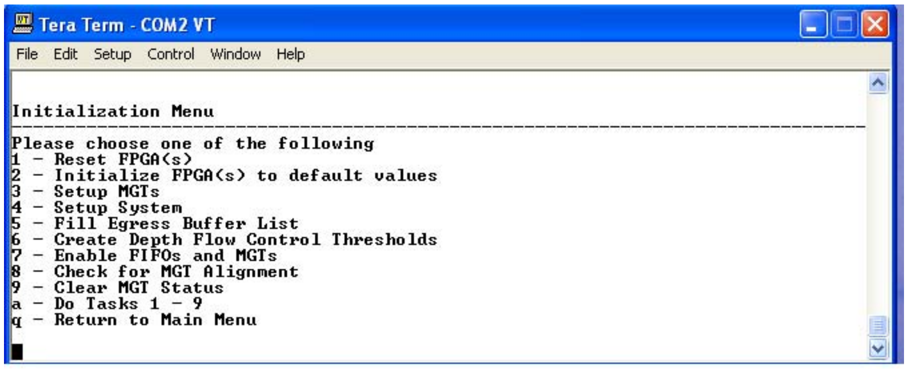
Signal	Type	Description
RXP[0:3], RXN[0:3]	Input	Differential MGT receive signals.
etm_clk	Output	100 MHz clock to the upstream FPGA for ETM data and the SOC.
etm_soc	Output	Start of Cell indicator to the upstream FPGA.
etm_data[31:0]	Output	Data to the upstream FPGA.
etm_fc[0:15]	Input	Flow control from the upstream FPGA.
etm_fc_start	Input	Signal that indicates which etm_fc value represents flow control for Port 0.
itm_clk	Input	100 MHz clock from the upstream FPGA for ITM data and the SOC.
itm_soc	Input	Start of Cell indicator for ITM data.
itm_data[31:0]	Input	Data from the upstream FPGA.
itm_fc_off[0:15]	Output	Flow control data to the upstream FPGA.
Itm_fc_clk	Output	100 MHz clock for itm_fc_off.
Cpu_tx_data[7:0]	Output	CPU data to the upstream FPGA.
Cpu_tx_sof_n	Output	Active-low start-of-frame signal to the upstream FPGA. Asserted to indicate the CON byte.
Cpu_rx_data[7:0]	Input	CPU data from the upstream FPGA.
Cpu_rx_sof_n	Input	Active-low start-of-frame signal from the upstream FPGA. Asserted to indicate the CON byte.
ready	Output	DCM lock signal to the upstream FPGA.
Error	Output	Asserted to indicate a parity error occurred in the CON byte.
Dcm_locked_led	Output	Indicator LED for the DCM lock signal.
Led_for_clock	Output	Indicator LED that blinks to show the DCR clock is working.

Demo Operation

When the demo starts, the HyperTerminal program shows a welcome screen and instructs the user to select the number of FPGAs on the line card. When two ML323 boards are stacked together, the answer is two FPGAs. The user is shown another menu with the following choices:

- Initialization Menu
- Configuration Menu
- Demo Menu
- Debug Menu

Before the demo can be successfully run, the MFRD registers must be configured through the Initialization Menu (see Figure 5-5). In the Initialization Menu, each item can be run individually or the 'a' option runs them all. This step configures the MFRD on both FPGAs to be Port 4, with a cell payload size of 40 bytes. To choose a different port or payload size, go to the Configuration Menu and make another selection. Always run the Initialization Menu 'a' option after changing the Configuration options. The initialization is successful when the status of the MGTs is 0x0308 (read through the '8' option of the Initialization Menu).

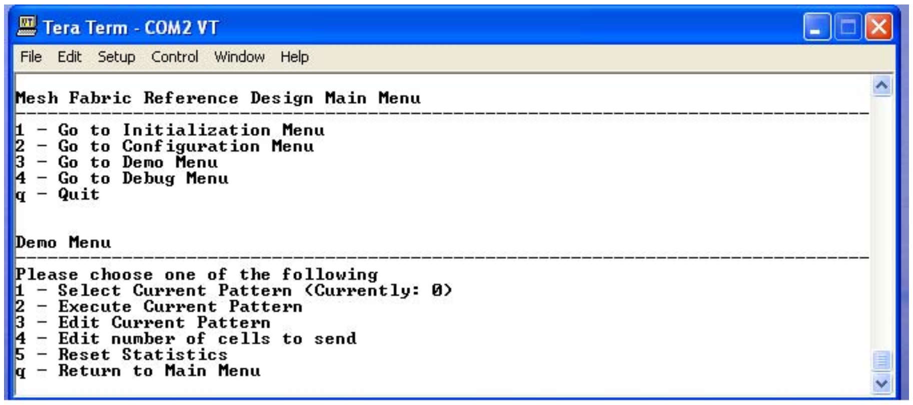


```
Initialization Menu
--------------------------------------------------------------------------------
Please choose one of the following
1 - Reset FPGA(s)
2 - Initialize FPGA(s) to default values
3 - Setup MGTs
4 - Setup System
5 - Fill Egress Buffer List
6 - Create Depth Flow Control Thresholds
7 - Enable FIFOs and MGTs
8 - Check for MGT Alignment
9 - Clear MGT Status
a - Do Tasks 1 - 9
q - Return to Main Menu
```

X698_c5_05_022304

Figure 5-5: **Initialization Menu**

When the MFRD registers and MGTs are configured, the MFRD Demo can begin. In the Demo Menu (see Figure 5-6), the user selects one of four patterns to generate traffic for the system as well as a pattern count. The pattern count controls the total number of cells generated by the Demo program. Each pattern is described by its Length, Priority, and Group ID parameters. These parameters each have subfields called Start, Stop, Increment, and Frequency. For example, a pattern can have one cell sent to each port in the system by setting the Group ID fields to (0, 8, 1, 0). This pattern generates cells to ports 0 through 8 in numerical order and then repeats the sequence for as long as the pattern count allows. If the Group ID fields are set to (0, 8, 1, 1), then the pattern is 0, 0, 1, 1, 2, 2,…, 8, 8 repeated. To send the cells in descending order, the Start value is set less than the End value, for example, (8, 0, 1, 0).



```
Mesh Fabric Reference Design Main Menu
------------------------------------------------------------------------------
1 - Go to Initialization Menu
2 - Go to Configuration Menu
3 - Go to Demo Menu
4 - Go to Debug Menu
q - Quit


Demo Menu
------------------------------------------------------------------------------
Please choose one of the following
1 - Select Current Pattern (Currently: 0)
2 - Execute Current Pattern
3 - Edit Current Pattern
4 - Edit number of cells to send
5 - Reset Statistics
q - Return to Main Menu
```

X698_c5_06_022304

Figure 5-6: **Demo Menu**

When the pattern is selected, the user can execute the current pattern. This step causes the processor to generate the requested pattern and start looking for the receive data in the Local Link FIFO. The pattern generates two types of information, data and control. The data is a sequence of 32-bit words representing the cell data, including cell header. The control word is a combination of Local Link signals. Table 5-4 shows the organization of the control word bits.

Table 5-4: **Control Word Organization**

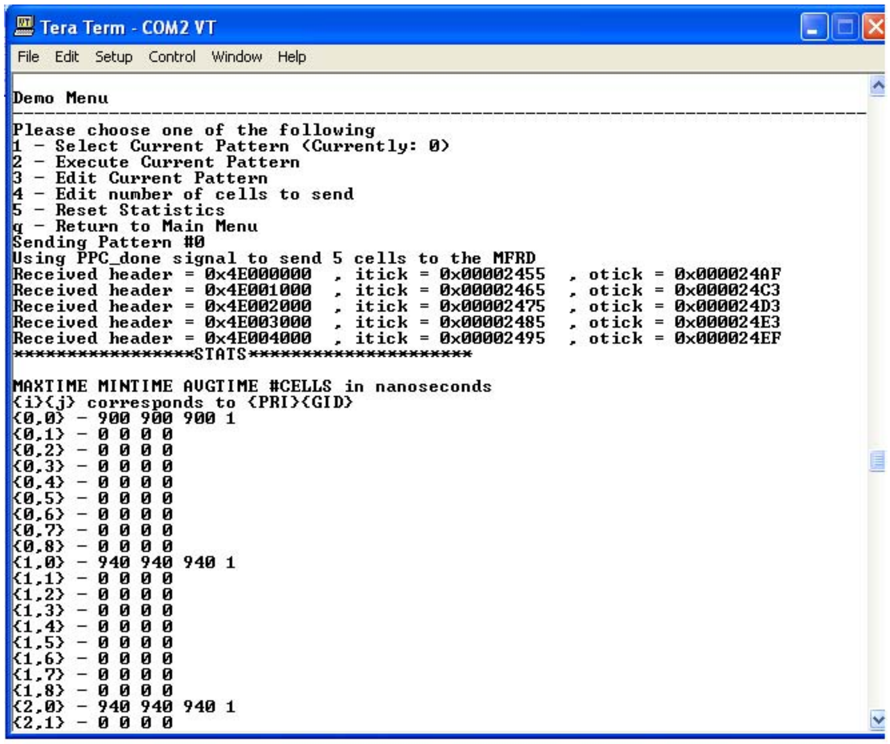
Bits	Function
0 – 3	REM
4	SRC_RDY_N
5 – 6	reserved
7	EOF_N
8	SOF_N

For example, a control word of 0x08F indicates a valid start of frame (SOF_N and SRC_RDY_N are Low), whereas a control word of 0x100 indicates a valid end of frame (EOF_N and SRC_RDY_N are Low) with all bytes valid (REM = 0). The reserved bits can be any value and can change during the loopback process.

The data portion of the pattern is written to a predetermined memory location, which is essentially the ingress Local Link FIFO. Each time data is written to this memory location, the FIFO stores the new data. The control pattern is written to a different memory location, and is translated into Local Link signals for controlling the FIFO. The software programs a DCR register in the Shim, which tells the Shim how many cells to read from the FIFO and send to the MFRD. When the FIFO contains at least one full cell, it is read and the transmission process begins. The cell is sent to one of the ports in the system, depending on the value of the Group ID field of the header, looped back through the appropriate MGT, and fed back into the MFRD. The Shim receives the ETM data and places it into the egress Local Link FIFO, after inserting the timestamp.

The software reads a predetermined memory location to determine whether the egress Local Link FIFO is ready with data, which occurs when SRC_RDY_N and SOF_N are Low in the receive control word. When the FIFO is ready, the software reads another memory location, which corresponds to the output of the egress Local Link FIFO. Then the software logs the incoming data header and two timestamps, and stores the minimum, maximum, and average travel times for each priority and Group ID pair.

The end of the transmission occurs when the egress Local Link FIFO SRC_RDY_N field has been deasserted for 100 counts. At this time, the statistics are displayed onto the terminal (see Figure 5-7), and the demo is finished.



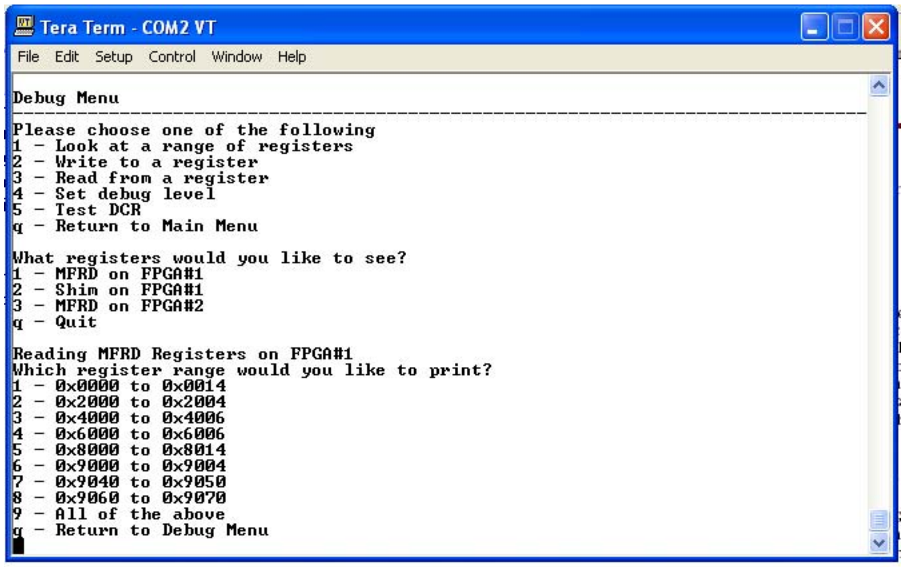
```
Demo Menu
------------------------------------------------------------------------------------
Please choose one of the following
1 - Select Current Pattern (Currently: 0)
2 - Execute Current Pattern
3 - Edit Current Pattern
4 - Edit number of cells to send
5 - Reset Statistics
q - Return to Main Menu
Sending Pattern #0
Using PPC_done signal to send 5 cells to the MFRD
Received header = 0x4E000000  , itick = 0x00002455  , otick = 0x000024AF
Received header = 0x4E001000  , itick = 0x00002465  , otick = 0x000024C3
Received header = 0x4E002000  , itick = 0x00002475  , otick = 0x000024D3
Received header = 0x4E003000  , itick = 0x00002485  , otick = 0x000024E3
Received header = 0x4E004000  , itick = 0x00002495  , otick = 0x000024EF
*****************STATS*********************

MAXTIME MINTIME AVGTIME #CELLS in nanoseconds
<i><j> corresponds to <PRI><GID>
<0,0> - 900 900 900 1
<0,1> - 0 0 0 0
<0,2> - 0 0 0 0
<0,3> - 0 0 0 0
<0,4> - 0 0 0 0
<0,5> - 0 0 0 0
<0,6> - 0 0 0 0
<0,7> - 0 0 0 0
<0,8> - 0 0 0 0
<1,0> - 940 940 940 1
<1,1> - 0 0 0 0
<1,2> - 0 0 0 0
<1,3> - 0 0 0 0
<1,4> - 0 0 0 0
<1,5> - 0 0 0 0
<1,6> - 0 0 0 0
<1,7> - 0 0 0 0
<1,8> - 0 0 0 0
<2,0> - 940 940 940 1
<2,1> - 0 0 0 0
```

X698_c5_07_022304

Figure 5-7: **Demo Results**

Figure 5-8 shows the Debug menu. The Debug menu is useful for reading back registers from the Shim or either of the FPGAs in the system. Also, the software debug level can be increased to see more outputs on the terminal program window, in case any anomalous behavior is seen. If the DCR interface on either FPGA seems broken, the software can test the DCR interface by sending consecutive writes and reads and automatically checking the read back data.



```
Debug Menu
--------------------------------------------------------------------------------
Please choose one of the following
1 - Look at a range of registers
2 - Write to a register
3 - Read from a register
4 - Set debug level
5 - Test DCR
q - Return to Main Menu

What registers would you like to see?
1 - MFRD on FPGA#1
2 - Shim on FPGA#1
3 - MFRD on FPGA#2
q - Quit

Reading MFRD Registers on FPGA#1
Which register range would you like to print?
1 - 0x0000 to 0x0014
2 - 0x2000 to 0x2004
3 - 0x4000 to 0x4006
4 - 0x6000 to 0x6006
5 - 0x8000 to 0x8014
6 - 0x9000 to 0x9004
7 - 0x9040 to 0x9050
8 - 0x9060 to 0x9070
9 - All of the above
q - Return to Debug Menu
```

X698_c5_08_022304

Figure 5-8: **Debug Menu**

Demo Reference Files

The Mesh Fabric Reference Design and Demo source files can be downloaded from

http://www.xilinx.com/esp/networks_telecom/optical/xlnx_net/mfrd/mfrd_source_code.zip

Glossary

Click on a letter, or scroll down to view the entire glossary.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

B

Backpressure

When an egress is oversubscribed, messages are sent asking the sources to stop sending new data.

BRAM

Block RAM.

Buffer

Cell storage area in block RAM representing a cell to be transferred. One buffer corresponds to one cell.

C

Cell

Data packet transferred between mesh fabric FPGAs over SERDES links. The fabric has no concept of what is transported in the cell.

D

DCR

Device Control Register.

Downstream

When multiple mesh fabric FPGAs are cascaded, downstream refers to an FPGA connected further away from the ingress TM.

E

Egress

Traffic leaving the switch and heading toward a physical port is egressing.

F

Flow Control

Synonymous with backpressure.

G

GroupID

Multicast label used to identify unique multicast domains.

H

HOL

Head Of Line.

I

Ingress

Traffic entering the switch from a physical port is ingressing.

M

Multicast, Mcast

Traffic headed to multiple destinations. Could be multiple switch ports or multiple physical ports connected to a single switch port or a mix.

P

Priority

Each port maintains a set of queues, one queue per traffic priority, which allows for multiple priority levels in traffic flow.

S

SDR

Single Data Rate.

SERDES

Serializer/Deserializer.

SF

Switch Fabric. Can refer to a single FPGA comprising an element of the fabric or the entire collection of parts.

SOC

Start of Cell.

T

TM

Traffic Manager. Device connected to the switch fabric FPGA. While the fabric is a full-duplex device, the TM can be made of two simplex parts or a single duplex one.

U

Unicast, Ucast

Traffic headed to a single destination.

Upstream

When multiple mesh fabric FPGAs are cascaded, upstream refers to an FPGA connected closer to the ingress TM.

W

WRR

Weighted Round Robin.