



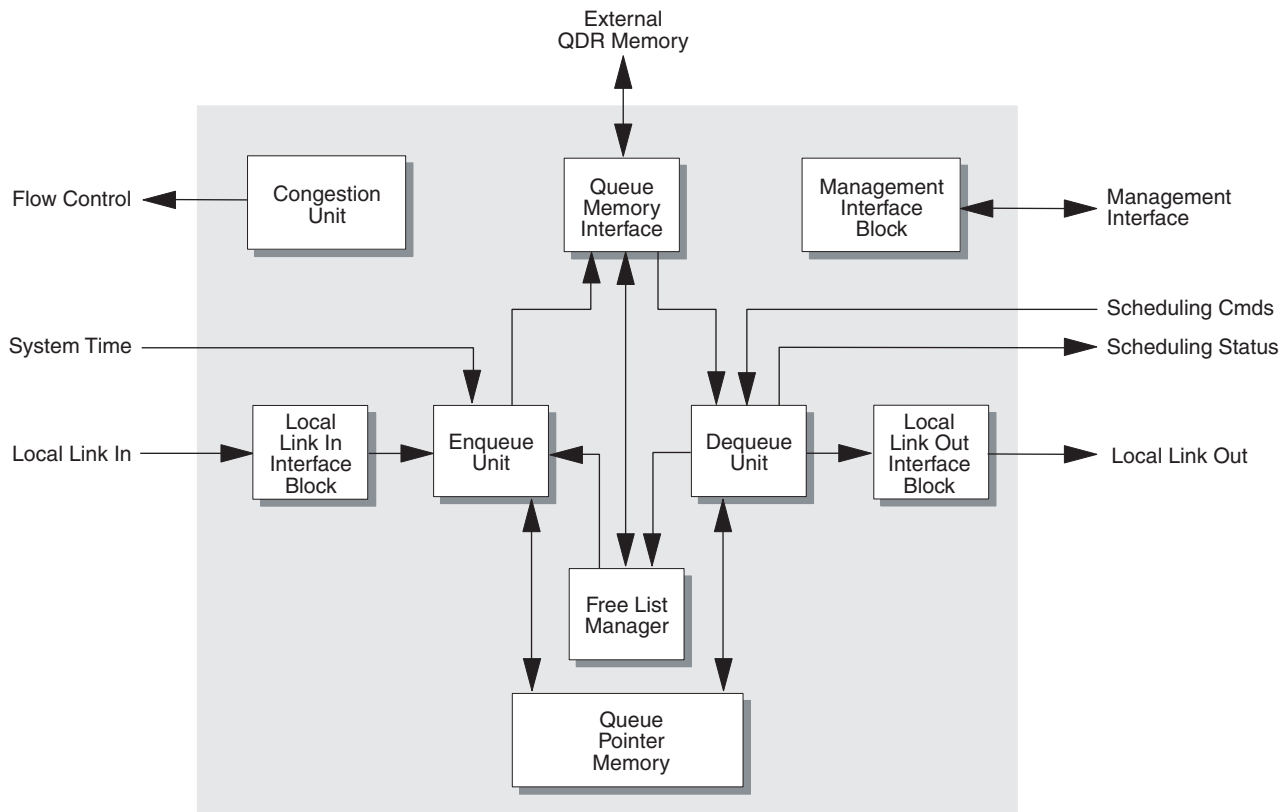
XAPP511 (v1.1) May 4, 2007

## Queue Manager Reference Design

### Summary

The Queue Manager Reference Design (QMRD) illustrates per-flow queuing for network processing applications, along with class-based flow control.

The QMRD segments variable length frames into fixed length Fabric Protocol Data Units (PDUs) when configured for ingress queuing, and reassembles fixed length Fabric PDUs into variable length frames when configured for egress queuing. It provides command and status interfaces that can be connected to a traffic scheduler, providing a complete traffic queuing and scheduling solution. [Figure 1](#) illustrates the overall architecture of the QMRD.



X511\_01\_040907

Figure 1: QMRD Block Diagram

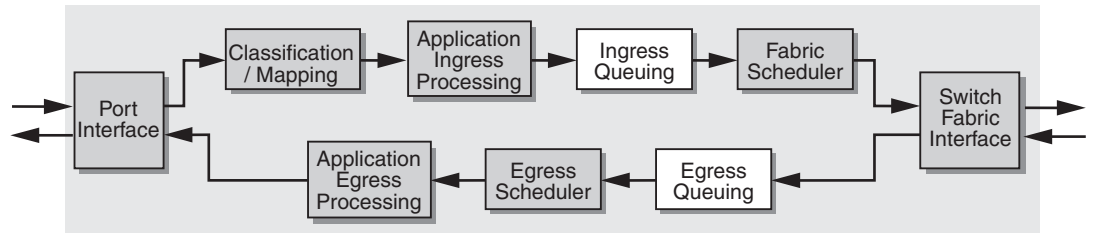
The QMRD source code can be downloaded from the following location on the web:

<http://www.xilinx.com/bvdocs/apnotes/xapp511.zip>

Registration is required to download the source code.

## Introduction

At various points in a network, it's common to encounter streams of data comprised of different flows, where a flow can be defined as a set of data with a specific destination. Queue management enables separate buffering of each flow, which allows data to be forwarded according to specific network policies and quality of service requirements. The role of queue management in a switch fabric port is shown in [Figure 2](#). The QMRD can be configured to implement queue management in both ingress and egress paths.



X511\_02\_040907

*Figure 2: Network Processing Framework*

When used for ingress queuing applications the QMRD accepts Port PDUs on its LocalLink Input Interface, segments them into Buffer PDUs and places each in the appropriate queue. Buffer PDUs are de-queued under control of the Fabric Scheduler and transferred to the Switch Fabric Interface.

When used for egress queuing applications the QMRD accepts Fabric PDUs from the Switch Fabric Interface, converts them into Buffer PDUs and places them into what are typically traffic class queues. Buffer PDUs are de-queued under control of the Egress Scheduler and transferred to the Application Egress Processing functions.

## Terminology

Certain terms are used with specific meaning within this document. These terms are defined in [Table 1](#). Acronyms are spelled out in [Table 2](#).

*Table 1: Definition of Terms*

Term	Definition
Packet	A transmission across a LocalLink interface, delimited by SOF and EOF.
Port PDU	The contents of a variable-sized packet on the port interface side of the QMRD (input during ingress, output during egress).
Fabric PDU	The contents of a fixed-size packet on the switch fabric side of the QMRD (output during ingress, input during egress).
Buffer PDU	The contents of one storage entry in a linked list.
Buffer Slot	A block in memory in which one Buffer PDU can be stored.
Flow Queue	A queue in QDR memory in which Buffer PDUs are stored, corresponding to a specific Queue ID.
Assembly Queue	A queue in QDR memory in which Fabric PDUs are reassembled in egress mode, before being attached to a flow queue.
Free List	A queue in QDR memory in which unused Buffer Slots are stored.
Line Rate	This is the peak data rate at which the Input and Output LocalLink Interfaces can operate, which is 10 Gbps.
LINK_CLK	The clock rate at which the LocalLink interfaces operate.
DCR_CLK	The clock rate at which the DCR Slave interface operates.
CORE_CLK	The internal clock rate of the QMRD.

Table 1: Definition of Terms (Continued)

Term	Definition
Queue Capacity	The maximum of number of Queues which can be used.
Class Capacity	The maximum number of classes which can be used.

Table 2: Meaning of Acronyms

Acronym	Meaning
BRAM	Block RAM
DCR	Device Control Register (Bus)
EOF	End Of Frame
FIFO	First-In First-Out (Memory)
PDU	Protocol Data Unit
QDR	Quad Data Rate (Memory)
QMRD	Queue Manager Reference Design
SAR	Segmentation and Reassembly
SOF	Start of Frame
TCP	Transmission Control Protocol
WRED	Weighted Random Early Discard

## Features

The following list outlines the high-level features of the QMRD:

- Supports from 256 to 64K queues
- Supports up to eight traffic classes
- Segment size (buffer slot size) configurable as 64 or 256 bytes
- Command and status interface for connection to traffic scheduler
- Configurable class-based flow control, including WRED

## Architecture

The QMRD splits variable sized packets (Port PDUs) into fixed-sized segments (Buffer PDUs), which are stored as linked lists. Maintaining a linked list for each queue allows memory to be allocated and used dynamically as needed, reducing wasted memory capacity.

As shown in [Figure 1](#), the key functional blocks are:

- LocalLink Input Interface
- LocalLink Output Interface
- Management Interface
- Queue Memory Interface
- Free List Manager
- Queue Pointer Memory
- Congestion Unit
- Enqueue Unit
- Dequeue Unit

The LocalLink Input and LocalLink Output Interfaces implement the functionality described in the LocalLink Interface Specification [Reference Item 1]. The Management Interface operates as a DCR bus slave [Reference Item 3].

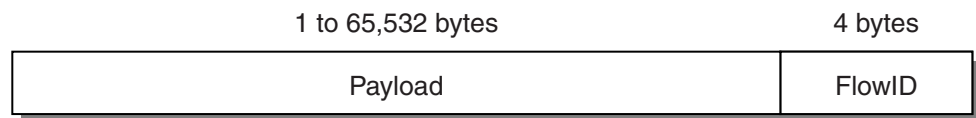
The Queue Memory Interface provides access to external QDR memory for queueing. The Free List Manager maintains a linked list of unused buffer slots, which can be allocated for PDU storage. The Queue Pointer Memory stores head and tail pointers for all queues, and the Congestion Unit implements flow control measures.

The Enqueue Unit accepts input PDUs and stores them in queues by attaching them to linked lists. The Enqueue Unit segments Port PDUs in ingress mode, and reassembles Fabric PDUs in egress mode.

The Dequeue Unit outputs PDUs after removing them from queues. Fabric PDUs are output in ingress mode, and Port PDUs are output in egress mode.

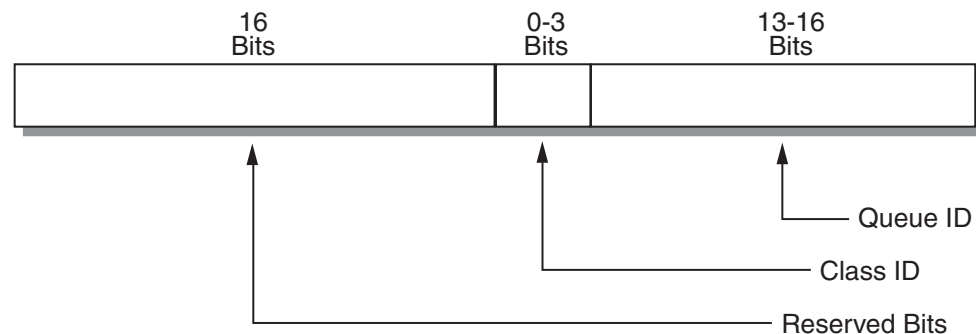
## PDU Formats

Port PDUs are variable length packets used on the Port Interface side, which are input in ingress mode and output in egress mode. The only Port PDU content that is examined by the QMRD is the Flow ID descriptor that appears as the first four bytes. Port PDUs can be from 5 to 65,536 bytes in length. Figure 3 shows the format of Port PDUs, and Figure 4 shows the format of the Flow ID field. The upper 16 bits of the Flow ID aren't used by the QMRD, but they are carried all the way through unchanged, including to the Scheduling Status Vector. The lower 16 bits carry Queue ID and Class ID with a configurable partition. The sizes of the Queue ID and Class ID fields can change over a small range, as long as the total is always 16 bits.



X511\_03\_030904

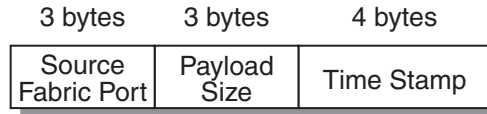
Figure 3: Port PDU Format



x511\_04\_040907

Figure 4: Flow ID Format

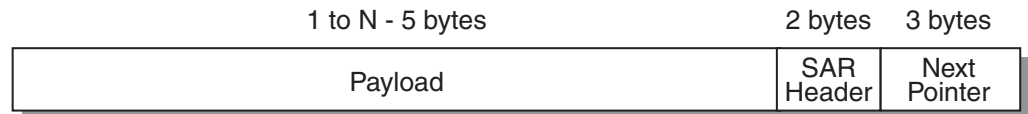
Before segmentation Port PDUs are tagged with a header that contains information needed for the switching process. The format of this tag is shown in Figure 5. The Time Stamp field contains the system time value captured from the SYS\_TIME[31:0] port when the PDU arrives at the Enqueue Unit. The Payload Size field contains the number of bytes in the Port PDU including the Flow ID header.



x511\_05\_040907

Figure 5: Port PDU Tag

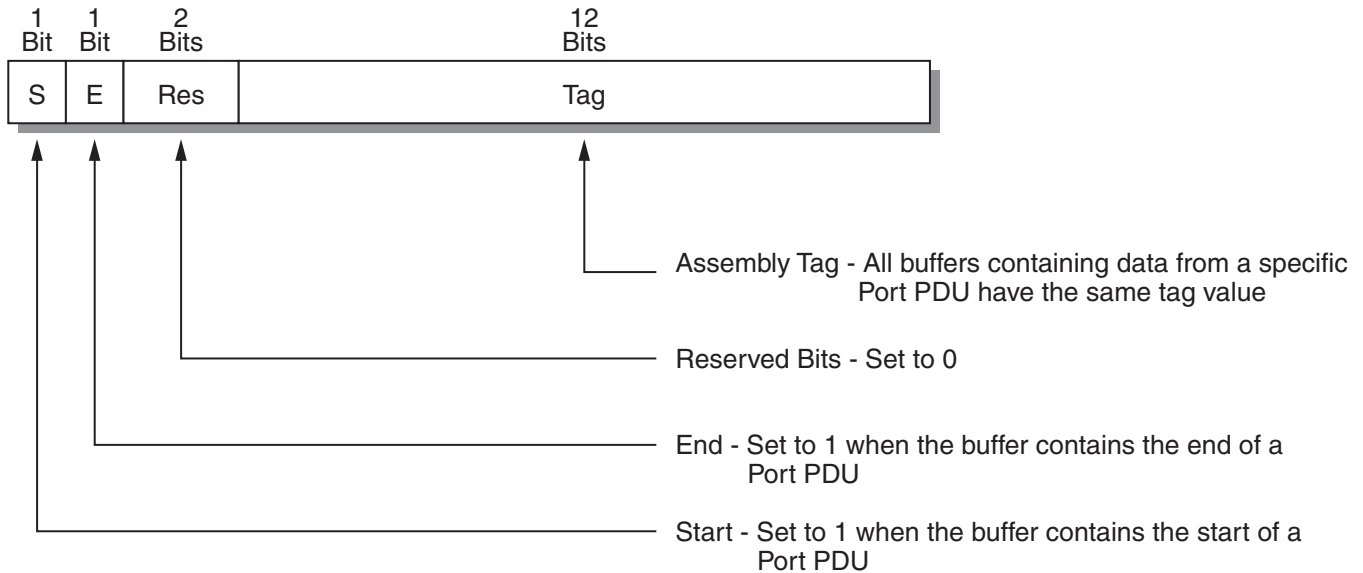
Buffer PDUs are used to store segmented Port PDUs in the queues. Buffer PDU headers consist of two components: a SAR Header used to control segmentation and reassembly, and a Next Pointer used to link to the next buffer in the queue. Buffer PDUs can be from 6 to N-5 bytes in length, where N is the size of a buffer slot. Figure 6 shows the format of Buffer PDUs.



x511\_06\_040907

Figure 6: Buffer PDU Format

The SAR Header is two bytes in length and contains the fields shown in Figure 7. The header contains all of the information needed to reassemble a segmented Port PDU.

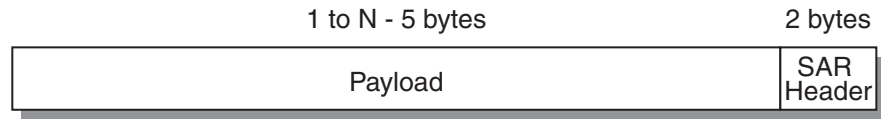


x511\_07\_040907

Figure 7: SAR Header Format

The Next Pointer is a 24-bit buffer pointer that addresses the next Buffer PDU in the queue. If the Buffer PDU is at the tail of a queue the Next Pointer will be set to zero. All buffer pointers address Queue Memory with 64-byte granularity. As a result buffer pointers can address up to 16.8 million packet buffers when buffer slot size is 64 and 4.2 million when buffer slot size is 256.

Fabric PDUs are used to transfer data between the QMRD and the switch fabric. The difference between a Fabric PDU and a Buffer PDU is that the Next Pointer has been stripped. Figure 8 shows the format of Fabric PDUs.

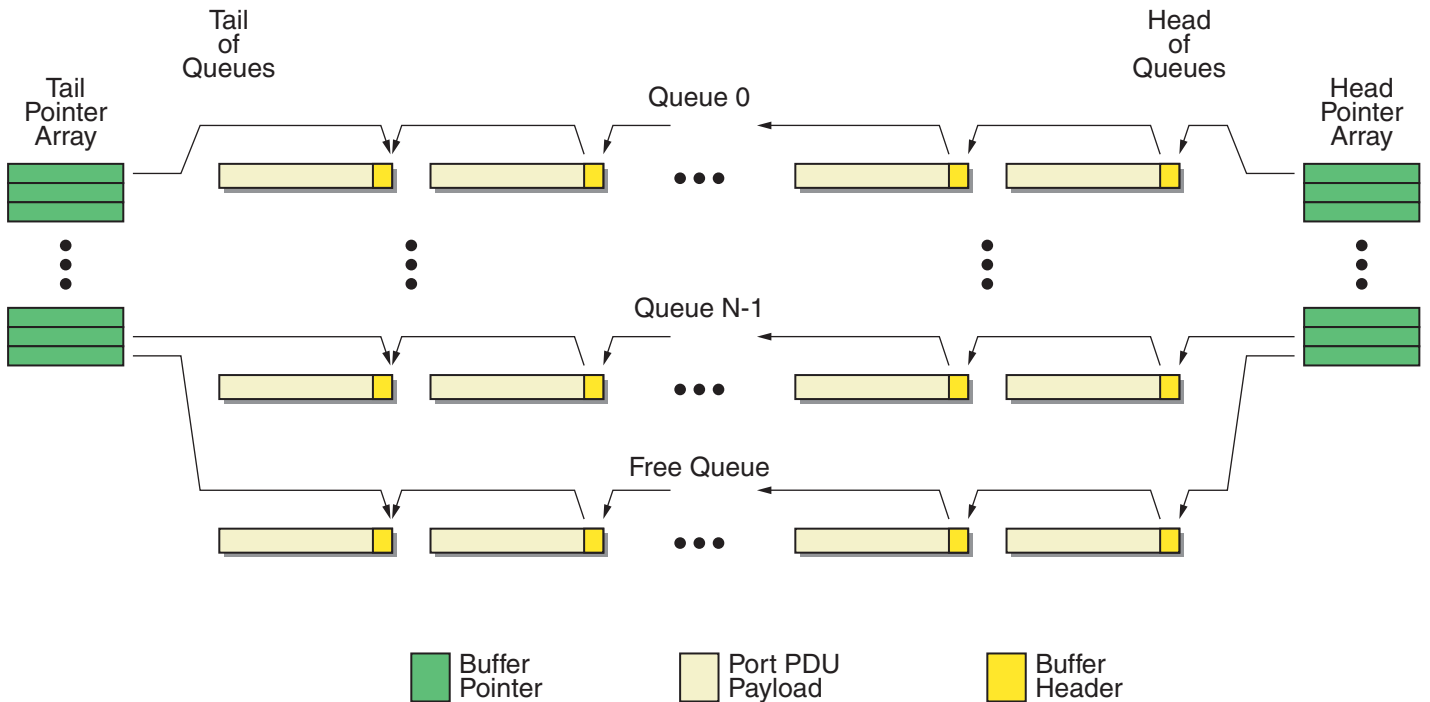


x511\_08\_040907

Figure 8: Fabric PDU Format

### Storage Schema

The QMRD stores queues in external memory in the form of linked lists of fixed length buffers. Two arrays are used to store buffer pointers to the Buffer PDUs at the head and tail of each queue. Figure 9 illustrates how these data structures are used to create the queues.



x511\_09\_040907

Figure 9: Fabric PDU Format

In addition to the flow queues there is one additional queue called the Free List that is used to store unused buffers. This queue is only accessed by the Enqueue and Dequeue Units.

### Ingress Mode PDU Processing

This section describes the processing of traffic that occurs when the QMRD is configured for ingress operation. Figure 10 illustrates how data is mapped to the various PDU formats at each stage of processing.

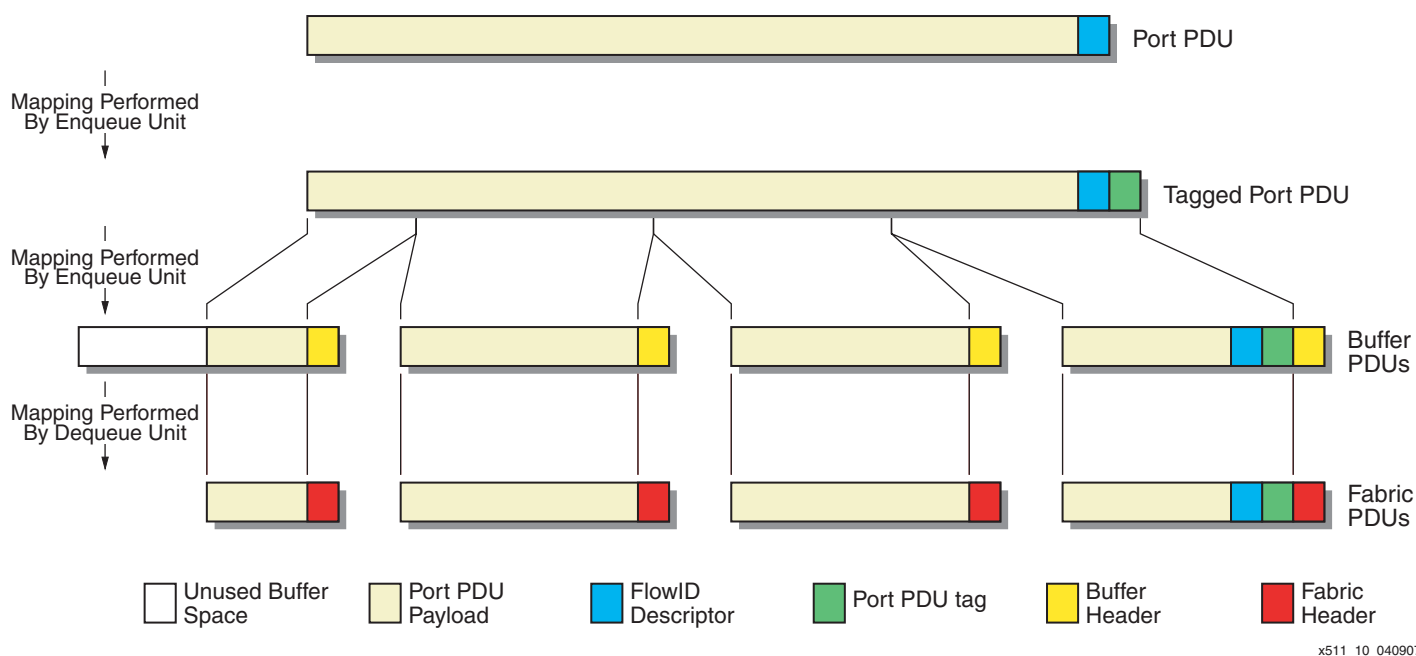


Figure 10: PDU Processing for Ingress Configuration

The first operation performed is the tagging of each incoming Port PDU by the Enqueue Unit. The tag is prepended to the Port PDU. The value used for the Time Stamp field is captured from the SYS\_TIME[31:0] port when the Port PDU is enqueued. The Payload Size field is annotated into the buffer memory after the Port PDU has been completely received.

The next operations performed by the Enqueue Unit are segmentation and enqueue. Segmentation converts variable length Port PDUs into one or more Buffer PDUs. The Buffer PDUs are then enqueued by linking them to the tail of a queue based on the Flow ID. Once the complete Port PDU is enqueued, an enqueue Scheduling Status Vector is generated.

As shown in Figure 4 the Flow ID can contain both queue and class identifiers. The Queue ID is exclusively used for queue selection, while the Class ID is used to enforce flow control policies.

The next stage of processing occurs when the first Buffer PDU of a Port PDU appears at the head of a queue. When the Dequeue Unit receives a dequeue command from the Traffic Scheduler, it unlinks the next Buffer PDU from the appropriate queue and forwards it to the LocalLink Output Interface. As each Buffer PDU is transferred it is converted into a Fabric PDU by stripping the Next Pointer from the header.

## Egress Mode PDU Processing

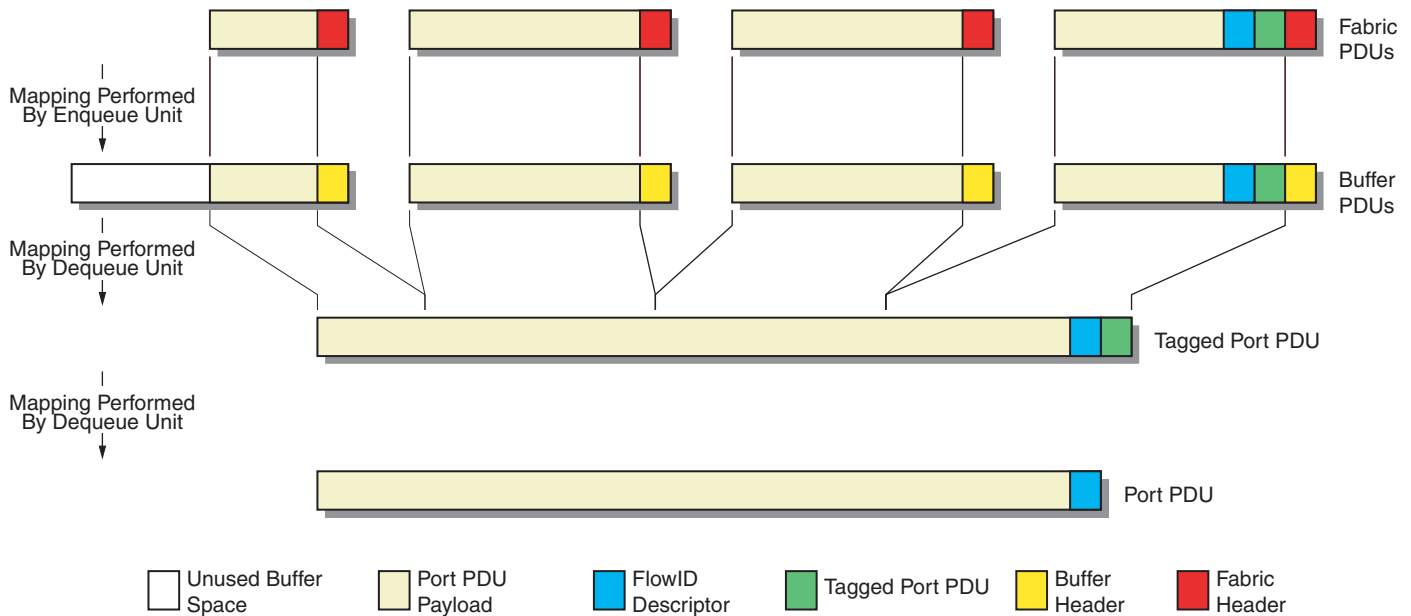
This section describes the processing of traffic that occurs when the QMRD is configured for egress operation.

The first operation performed by the Enqueue Unit is linking the incoming Fabric PDUs to the tail of the appropriate assembly queues. Assembly queues are used to collect all of the Fabric PDUs corresponding to a particular Port PDU in a linked list before they are linked into a flow queue. This is needed since Fabric PDUs from multiple Port PDUs may be arriving in an interleaved fashion from other ports on the switch fabric. The use of assembly queues ensures that all of the Buffer PDUs corresponding to a single Port PDU are queued together.

Assembly queue assignment is based on the Assembly Tag in the SAR header. The Enqueue Unit can manage up to 4096 assembly queues at one time. Each time a Fabric PDU is received that has the Start bit in the SAR header set, the Enqueue Unit stores the association between the Assembly Tag, the Flow ID of the Port PDU, and buffer pointers to the head and tail of the

assembly queue in a lookup table. This mapping is used to associate subsequent Fabric PDUs with the correct assembly queue. When a Fabric PDU arrives with the End bit in the SAR header set, the assembly queue is linked to the appropriate flow queue and an enqueue Scheduling Status Vector is generated.

The next stage of processing occurs when the first Buffer PDU of a Port PDU appears at the head of a queue. When the Dequeue Unit receives a dequeue command from the Traffic Scheduler, it unlinks the entire Port PDU and forwards it across the LocalLink Output Interface. The Port PDU Tag is stripped from the Port PDU as it is transferred. Figure 11 illustrates how data is mapped to the various PDU formats at each stage of processing.



x511\_11\_040907

Figure 11: PDU Processing for Egress Configuration

## Flow Control

The QMRD implements a three-stage scheme to handle queue congestion. These mechanisms are designed to maximize throughput and minimize data loss. Each stage of congestion handling is based on programmable thresholds that are configured on a per-class basis. The three stages are as follows:

- Stage 1 - Flow Control Messages
- Stage 2 - Weighted Random Early Discard (WRED)
- Stage 3 - Tail Discard

Each of these mechanisms is activated for a particular traffic class when the total number of Buffer PDUs queued within that traffic class is greater than or equal to the corresponding threshold in the Flow Control Management Registers. Each time a PDU arrives on the LocalLink Input Interface these thresholds are checked and the appropriate action is taken before the PDU is enqueued.

The thresholds are checked in the following order: Stage 3, then Stage 2, then Stage 1. This is consistent with normal expected usage, in which the Stage 1 threshold will be the lowest and the Stage 3 threshold the highest. No more than one of the three stages will be implemented on any PDU.



## Flow Control Messages

When the Flow Control Message threshold has been exceeded, the Congestion Unit issues a Flow Control Message through the Flow Control Interface, and the PDU is enqueued. A Flow Control Message is only sent upon the transition above the threshold. A second Flow Control Message for the same class will not be sent until the PDU count drops below the threshold and then increases past it again.

## WRED

WRED is a technique for reducing the transmission rate of TCP traffic flows. When the WRED threshold has been exceeded, the WRED parameters are retrieved from the WRED Control Register. The WRED Weight parameter and the TCP flag are used to determine how the Port PDU will be handled. If the TCP flag is set, the Weight parameter is compared with a pseudo random value generated within the Congestion Unit. If the pseudo random value is greater than the weight value for the queue, the Port PDU is discarded. This comparison is only done once, at the start of the Port PDU.

## Tail Discard

When the Tail Discard threshold has been exceeded, the Congestion Unit discards all Port PDUs corresponding to that traffic class.

## Scheduling

The QMRD relies on an external Scheduler to manage the order in which Port PDUs are dequeued. In order to do this a Scheduler will need to maintain models of the current loading of each queue. In order to support an external Scheduler, the QMRD does the following:

- Issues a Status Message to the Scheduler after each Port PDU is enqueued.
- Initiates dequeue of Port PDUs in response to Command Messages from the Scheduler.
- Issues a Status Message to the Scheduler upon completion of each dequeue operation.

## Error Handling

The QMRD shall detect protocol errors and parity errors as defined in [Table 3](#) and [Table 4](#).

*Table 3: Protocol Error Handling*

Protocol Error	Recovery
Missing Packet SOF	All data is discarded until SOF
Missing Packet EOF	Erroneous PDU and following PDU are discarded
Undersized PDU (Size < 5 bytes Ingress, Size < 3 bytes Egress)	PDU is discarded
Oversized PDU (Payload Size > 65532)	PDU is discarded
Incorrect Fabric PDU Size (Egress Mode)	PDU is discarded
Assembly Format Error (During assembly in egress mode, missing Start, missing End, or oversized Port PDU)	Entire Port PDU is discarded

Table 4: Parity Error Handling

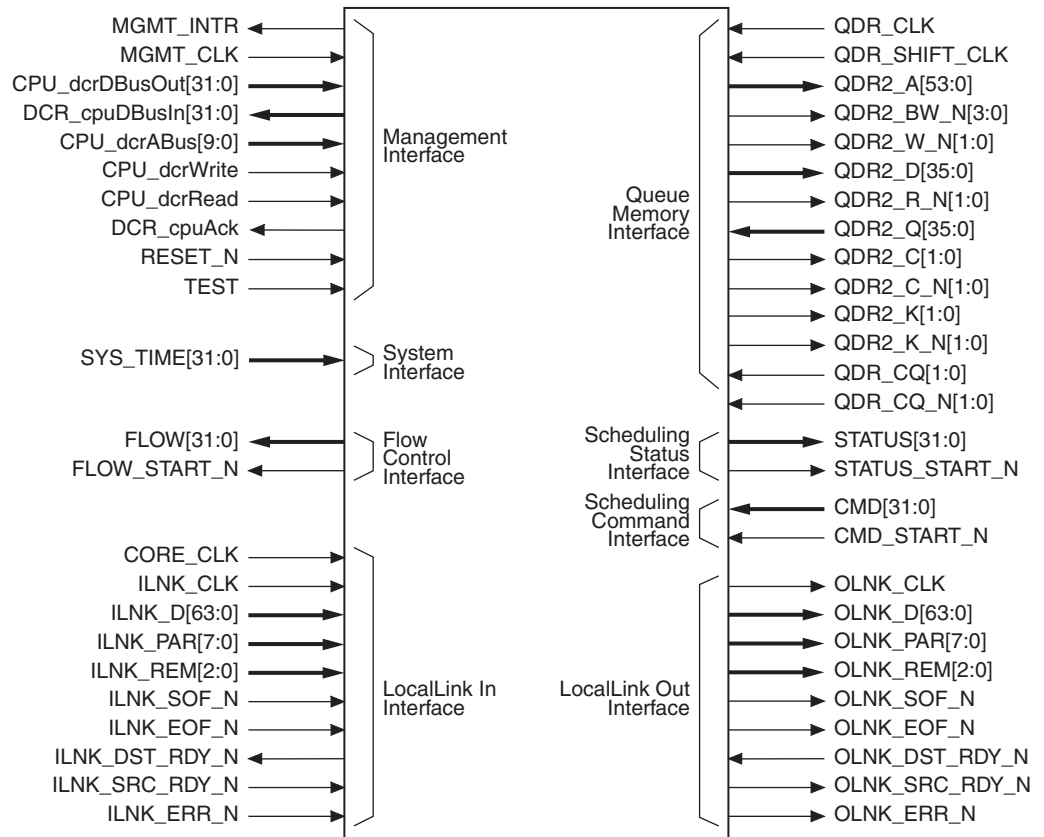
Parity Error Source	Recovery
LocalLink Input or Input Rate Adaption FIFO	PDU is flagged as erroneous, and OLNK_ERR is asserted at LocalLink Output upon dequeue
Assembly Queue (Egress Mode)	Fatal Error (Linked List Corruption)
Pointer Memory	Fatal Error (Linked List Corruption)
QDR Critical (Pointer in erroneous word)	Fatal Error (Linked List Corruption)
QDR Non-Critical (No pointer in erroneous word)	OLNK_ERR is asserted at LocalLink Output upon dequeue
Dequeue Status Memory (Ingress Mode)	Fatal Error (Linked List Corruption caused by incorrect payload size)

It should be noted that assertion of the ILNK\_ERR signal during reception of a PDU shall be treated identically to a parity error. The PDU shall be flagged as erroneous and OLNK\_ERR asserted at the output upon dequeue.

Certain error events are fatal in that normal operation cannot be continued. The primary example of this situation is corruption of a linked list. Upon occurrence of a fatal error, the QMRD shall assert its interrupt signal and shall cease operation except for the Management Interface. The shutdown is implemented by halting both LocalLink interfaces at the next PDU boundary.

## Interface Description

The QMRD connects to the rest of the system through eight interfaces. [Figure 12](#) shows the signals that make up the QMRD interfaces.



x511\_12\_040907

Figure 12: QMRD Pinout

### LocalLink Input Interface

The LocalLink Input Interface is used to deliver frames to the QMRD for storage. It is compliant with the LocalLink Interface Specification [Reference Item 1]. Table 5 lists the signals supported by this implementation.

Table 5: LocalLink Input Interface Signals

Signal	Direction	Description
ILNK_CLK	Output	<b>Input link clock.</b> Data and control signals on this link must meet setup and hold times with respect to the rising edge of this clock.
ILNK_D[63:0]	Input	<b>Input data.</b> The width of this bus is fixed at 64 bits.
ILNK_PAR[7:0]	Input	<b>Input Parity.</b> This optional signal provides even byte parity coverage for ILNK_D. Parity errors detected on this interface will result in the assertion of the OLNK_ERR_N when the data that created the error is presented on OLNK_D.
ILNK_REM[2:0]	Input	<b>Data remainder indication.</b> The number of bytes that are valid during the last beat of a packet transfer is equal to this value plus one.
ILNK_SOF_N	Input	<b>Start of frame indication.</b> Used to indicate that the word being presented on the ILNK_D signals is the first beat of a packet transfer.

Table 5: LocalLink Input Interface Signals (Continued)

Signal	Direction	Description
ILNK_EOF_N	Input	<b>End of frame indication.</b> Used to indicate that the data being presented on the ILNK_D signals is the last beat of a packet transfer.
ILNK_ERR_N	Input	<b>Input parity error.</b> Indicates that a parity error was detected in the current word by an upstream device.
ILNK_DST_RDY_N	Output	<b>Destination ready indication.</b> Used to indicate that the QMRD can accept the data presented on ILNK_D during this clock cycle.
ILNK_SRC_RDY_N	Input	<b>Source ready indication.</b> Used to indicate to the QMRD that the data presented on ILNK_D is valid during this clock cycle.

### LocalLink Output Interface

The LocalLink Output Interface is used to deliver PDUs stored in the queues. It is compliant with the LocalLink Interface Specification [[Reference Item 1](#)]. [Table 6](#) lists the signals supported by this implementation.

Table 6: LocalLink Output Interface Signals

Signal	Direction	Description
OLNK_CLK	Output	<b>Output link clock.</b> Data and control signals on this link are referenced with respect to the rising edge of this clock.
OLNK_D[63:0]	Output	<b>Output link data.</b> The width of this bus is fixed at 64 bits.
OLNK_PAR[7:0]	Output	<b>Output link parity.</b> This optional signal provides even byte parity coverage for OLNK_D.
OLNK_REM[2:0]	Output	<b>Data link remainder indication.</b> The number of bytes that are valid during the last beat of a packet transfer is equal to this value plus one.
OLNK_SOF_N	Output	<b>Start of frame indication.</b> Used to indicate that the word being presented on the OLNK_D signals is the first beat of a frame transfer.
OLNK_EOF_N	Output	<b>End of frame indication.</b> Used to indicate that the data being presented on the OLNK_D signals is the last beat of a packet.
OLNK_DST_RDY_N	Input	<b>Destination ready indication.</b> Used to indicate that the destination can accept the data presented on OLNK_D during this clock cycle.
OLNK_SRC_RDY_N	Output	<b>Source ready indication.</b> Used to indicate that the data presented on OLNK_D is valid during this clock cycle.
OLNK_ERR_N	Output	<b>Frame error indication.</b> Used to indicate to the next processing element that a parity error has been detected during reception of the frame.

### System Time Interface

The System Interface is used to deliver the current system time in the form of a 32-bit count to the QMRD. [Table 7](#) shows the specifics of this signal.

Table 7: System Interface Signals

Signal	Direction	Description
SYS_TIME[31:0]	Input	<b>System time.</b> This value is the system time base and is sampled by the QMRD each time a PDU enters the Enqueue Unit. This input must be updated synchronously with respect to CORE_CLK.

### Flow Control Interface

The Flow Control Interface is used to transfer flow control messages to the Fabric or Port Interface Cores. Table 8 shows the signals that make up this interface.

Table 8: Flow Control Interface Signals

Signal	Direction	Description
FLOW[31:0]	Output	<b>Flow Command.</b> The flow control vector is transferred across this bus. The transfer is synchronous with respect to the ILNK_CLK signal on the LocalLink Input Interface. The Flow Command word is encoded as follows: 31:27 Reserved 26:24 Class ID 23:0 Class Size (Total number of Buffer PDUs enqueued within this class.)
FLOW_START_N	Output	<b>Scheduling Command Start.</b> This signal is asserted to indicate that a valid flow control vector is presented on the FLOW[31:0] bus.

### Scheduling Interface

The Scheduling Interface is used to transfer Scheduling Status words to the Traffic Scheduler, and to receive Scheduling Commands. The signals comprising the Scheduling Interface are defined in Table 9.

In order to optimize data rate, the Scheduling Status and Command interchange is pipelined. That is, the Scheduler can issue commands before the QMRD completes and acknowledges previous commands. The QMRD maintains a pipeline depth of three, meaning that up to three Scheduling Commands can be issued without waiting for acknowledgement. Any additional commands written when the pipeline is full are discarded, and cause an interrupt to be generated.

The formats of Scheduling Status and Command words are shown in Table 10 and Table 11, respectively. In the case of Scheduling Status, Word 1 is sent first. The QMRD requires that at least one inactive cycle (CMD\_START\_N negated) is present between the end of one Scheduling Command and the start of the next.

Table 9: Scheduling Interface Signals

Signal	Direction	Description
STATUS[31:0]	Output	<b>Scheduling Status.</b> The scheduling status vector is transferred across this bus. The transfer is synchronous with respect to the OLNK_CLK signal.
STATUS_START_N	Output	<b>Scheduling Status Start.</b> This signal is asserted during the cycle that the first word of the status vector is presented on the STATUS[31:0] bus.

Table 9: Scheduling Interface Signals (Continued)

Signal	Direction	Description
CMD[31:0]	Input	<b>Scheduling Command.</b> The scheduling command vector is transferred across this bus. The transfer is synchronous with respect to the OLNK_CLK signal.
CMD_START_N	Input	<b>Scheduling Command Start.</b> This signal is asserted during the cycle that the first word of the command vector is presented on the CMD[31:0] bus.

Table 10: Scheduling Status Encoding

Word	Bit Field	Enqueue Definition	Dequeue Definition
1	31	0	1
	30	Reserved	Acknowledgement <b>0:</b> Dequeued Successfully <b>1:</b> Dequeued with Error (Queue doesn't exist, or queue is empty)
	29:17	Reserved	Reserved
	16:0	Payload Size in Bytes	Payload Size in Bytes
2	31:0	Flow ID <i>Only the least significant 16 bits are actually used, as shown in Figure 4. The upper 16 bits are passed through unchanged.</i>	

Table 11: Scheduling Command Encoding

Bit Field	Definition
31:16	Reserved
15:0	Queue ID/Class ID [15:0] ( <i>Encoded as in Flow ID</i> )

## Queue Memory Interface

The Queue Memory Interface supports QDR-II SRAMs, organized as two banks with a total width of 36 bits. The Queue Memory Interface signals are listed in [Table 12](#).

Table 12: Queue Memory Interface Signals

Signal Name	Direction	Description
QDR2_A[53:0]	Output	Address (27-bit address replicated for two banks)
QDR2_D[35:0]	Output	Write data
QDR2_Q[35:0]	Input	Read data
QDR2_R_N[1:0]	Output	Read enable
QDR2_W_N[1:0]	Output	Write enable
QDR2_BW_N[3:0]	Output	Byte lane select for writes
QDR2_K[1:0]	Output	Write clock
QDR2_K_N[1:0]	Output	Alternate phase write clock
QDR2_C[1:0]	See Note	Read clock
QDR2_C_N[1:0]	See Note	Alternate phase read clock
QDR2_CQ[1:0]	Input	Read echo clock

Table 12: Queue Memory Interface Signals (Continued)

Signal Name	Direction	Description
QDR2_CQ_N[1:0]	Input	Alternate phase read echo clock

**Note:** QDR SRAM devices support a number of different clocking techniques, particularly for capturing read data. The best approach for any given board will depend largely on the routed trace lengths of the clocks and data. The board used to test the QMRD has the read clocks wired directly to the write clocks at the QDR devices. Therefore, the read clocks are used as inputs in the QMRD although these are usually outputs from a QDR interface.

## Management Interface

The Management Interface is used to access the Queue Memory and the control registers contained in the QMRD. It is compliant with the CoreConnect DCR Bus Specification [Reference Item 3]. Table 13 lists the signals supported by this implementation.

Table 13: Management Interface Signals

Signal Name	Direction	Description
CPU_dcrABus[9:0]	Input	CPU DCR address bus
CPU_dcrDBusOut[31:0]	Input	CPU DCR data bus out
CPU_dcrRead	Input	CPU DCR read
CPU_dcrWrite	Input	CPU DCR write
DCR_cpuAck	Output	DCR CPU acknowledge
DCR_cpuDBusIn[31:0]	Output	DCR CPU data bus in
MGMT_CLK	Input	DCR clock
MGMT_INTR	Output	Interrupt
RESET_N	Input	Hard reset for QMRD
TEST	Input	When asserted, enables access to test and debug features in the QMRD. Tie Low for normal operation.

## Configuration

The configuration options supported by the QMRD are described in Table 14. It should be noted that two rules are automatically enforced on maximum queue and class capacity:

1. Illegal values for MAX\_QUEUE\_CAPACITY are automatically trimmed to the maximum legal value.
2. Queue index and class index must total no more than 16 bits, meaning that the product of MAX\_QUEUE\_CAPACITY and MAX\_CLASS\_CAPACITY must not exceed 64K. MAX\_CLASS\_CAPACITY will be automatically trimmed to accommodate this if needed.

These values determine the partitioning of the Queue ID and Class ID fields within the Flow ID, as shown in Figure 4.

Table 14: Configuration Parameters

Parameter	Values
INGRESS_MODE	Conditional Define: If this is parameter is defined, the QMRD operates in Ingress mode. Else, it operates in Egress Mode.
BUFFER_SIZE	Size of a buffer slot: 64B (0) 256B (1)
MAX_QUEUE_CAPACITY	256 Queues (0h) 512 Queues (1h) 1K Queues (2h) 2K Queues (3h) 4K Queues (4h) 8K Queues (5h) 16K Queues (6h) 32K Queues (7h) 64K Queues (8h) Illegal: 9h – Fh
MAX_CLASS_CAPACITY	1 Class (0h) 2 Classes (1h) 4 Classes (2h) 8 Classes (3h)
DCR_BASEADDR DCR_HIGHADDR	The range specified by these parameters must comprise a complete, contiguous power of two range: $(DCR\_HIGHADDR - DCR\_BASEADDR) = Range - 1$ $Range = 2^N$ The N least significant bits of DCR_BASEADDR must be zero, and N for the QMRD must be at least 7.
EN_FLOW_CONTROL	Flow Control Messages Disabled (0) Flow Control Messages Enabled (1)
EN_WRED	WRED Disabled (0) WRED Enabled (1)
EN_TAIL_DISCARD	Tail Discard Disabled (0) Tail Discard Enabled (1)
EN_STATS_REG	Statistics Registers Disabled (0) Statistics Registers Enabled (1) These include Class Size Registers, Class Statistics Registers and Parity Error Count Register.



## Implementation High-Level

The high-level operation of the QMRD is shown in [Figure 13](#). A more detailed description of the states and state transitions can be found in [Table 15](#) and [Table 16](#).

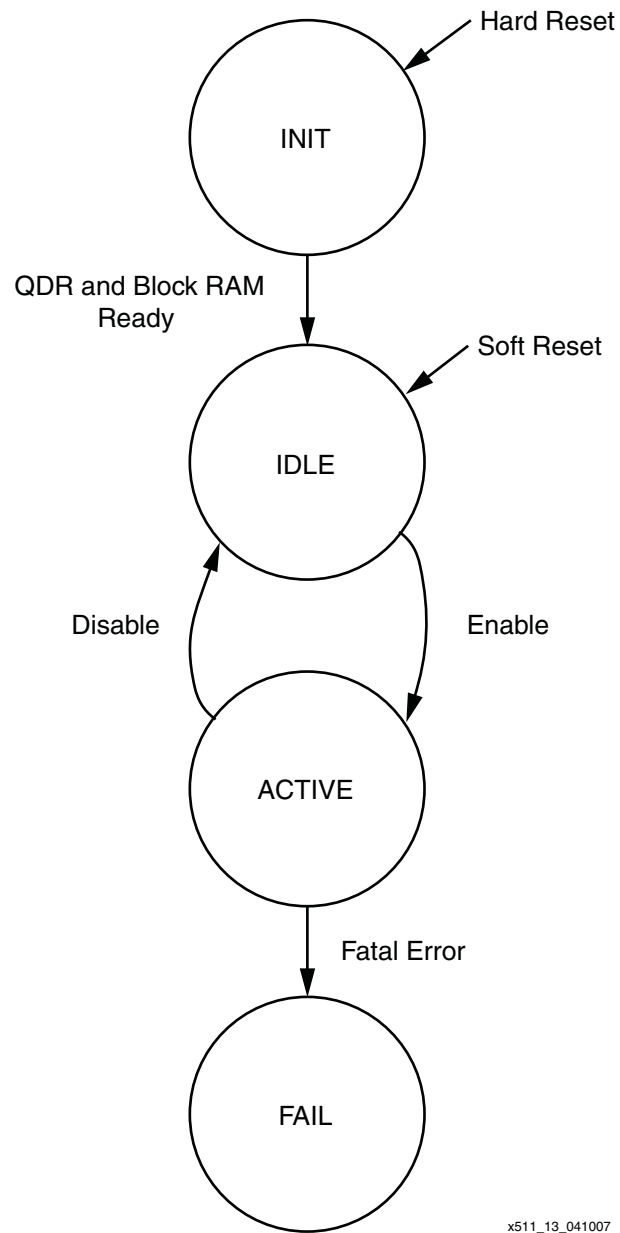


Figure 13: QMRD High-Level Operation

Table 15: High-Level State Descriptions

State	Description
Init	The Queue Memory Interface is not operational yet, and the RAM Ready register bit is negated. BRAM requiring initialization is not yet ready. The Management Interface is functioning, except that writes to the QMRD Control Register are ignored. The Input LocalLink Destination Ready signal is negated, to prevent traffic arrival.
Idle	The QDR DCM is phase locked, the Queue Memory Interface is operational, and the RAM Ready register bit is asserted. BRAM requiring initialization is ready. The Management Interface is fully functioning. The Input LocalLink Destination Ready signal (ILNK_DST_RDY_N) is negated, to prevent traffic arrival. Dequeue operations are disabled by preventing access to the Queue Memory.
Active	The Management Interface is functioning, except for the following limitations Queue Memory accesses are prohibited Queue Configuration Register is read-only Free List Status Register is read-only The Input LocalLink Destination Ready signal (ILNK_DST_RDY_N) is asserted, to allow traffic arrival. Dequeue operations are enabled.
Fail	Basically identical to the Idle state, except that direct transition to the Active state is prohibited. Exit from this state can only be achieved through a reset.

Table 16: High-Level State Transitions

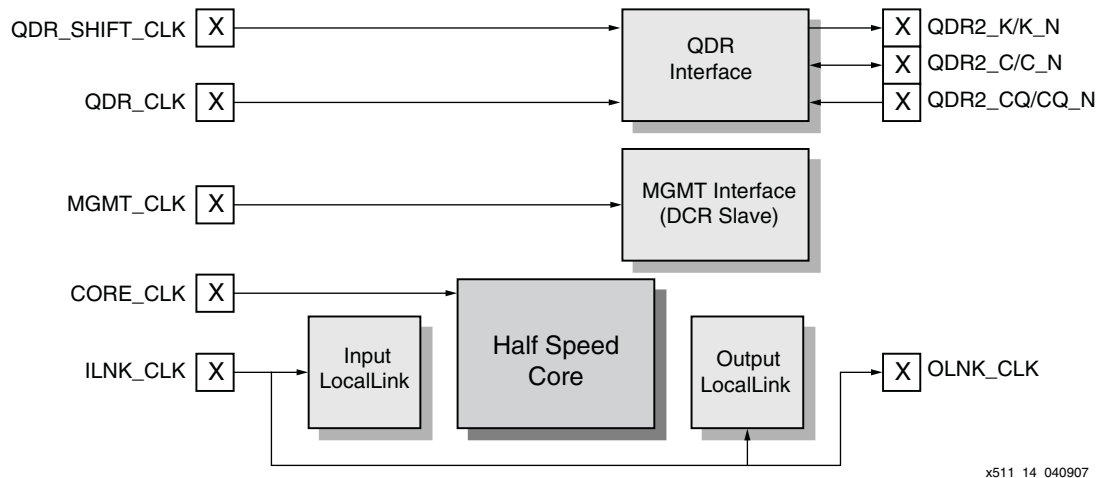
Transition	Description
Hard Reset	Occurs when the RESET signal is asserted.
Soft Reset	Occurs when the Soft_Reset register bit is set.
<b>Init→Idle</b>	Occurs when the memories becomes operational.
<b>Idle→Active</b>	Occurs when the Enable register bit is set through the Management Interface.
<b>Active→Idle</b>	Occurs when the Enable register bit is cleared through the Management Interface.
<b>Active→Fail</b>	Occurs when a fatal error is detected.

## Clocking

There are a number of different clock domains in the QMRD, as shown in [Figure 14](#). The bulk of the QMRD logic is clocked by CORE\_CLK, which must be half the speed of ILNK\_CLK and edge-aligned.

QDR\_CLK is the basis for the Queue Memory Interface, and must be identical to ILNK\_CLK. QDR\_SHIFT\_CLK is a delayed version of QDR\_CLK used to align write data sampling in the QDR interface. (QDR\_SHIFT\_CLK is delayed 90 degrees in QMRD testing.)

MGMT\_CLK is used for the Management Interface DCR slave port, and requires no alignment with other QMRD clocks.



x511\_14\_040907

Figure 14: QMRD Clock Usage

## Programming Model

The following sections define the access to the QMRD that is supported by the Management Interface.

### Memory Map

The memory map supported by the Management Interface is shown in Table 17. Accesses to reserved addresses will result in a Management Interface time-out. The addresses in Table 17 are all relative to the DCR base address, which is set by configuration parameters.

Table 17: Management Interface Address Map

Address Offset	Usage
048h : 07Fh	Unused
040h : 047h	Class Size Registers
030h : 03Fh	Class Statistics Registers
010h : 02Fh	Flow Control Management Registers
00Fh	Reserved for Test and Debug
009h : 00Eh	Unused
008h	Queue Memory Data Register
007h	Queue Memory Address Register
006h	Free List Status Register
005h	Parity Error Count Register
004h	Interrupt Mask Register
003h	Interrupt Source Register
002h	Queue Configuration Register
001h	QMRD Control Register
000h	Revision Register

## Queue Memory Access

The Queue Memory is much larger than the available Management Interface address space, so it must be accessed indirectly. Reads and writes of the Queue Memory are accomplished by reading and writing the Queue Memory Data Register. The memory address used is set up in the Queue Memory Address Register. This register can automatically increment after each Queue Memory Data Register access, although reads and writes cannot be intermixed in this fashion. Management Interface access to the Queue Memory is enabled only when the QMRD is in the Idle state.

During Management Interface reads of the Queue Memory, parity errors will cause a Management Interface time out and will increment the parity error count.

## Register Definitions

The following sections provide definitions of the registers accessible via the Management Interface. The following abbreviations are used to describe access to the register fields:

RW: Read-Write

RO: Read Only

RCO: Read Only, Cleared By Read

RCW: Read, Bit-wise clear by writing "1"

## Revision Register

It is expected that variants of the QMRD will be developed for different applications. In anticipation of this, the Revision Register ([Table 18](#)) provides a way for software to determine which variant and version of the QMRD are being used. In addition, this register can be accessed to determine how the QMRD has been configured.

*Table 18: Revision Register*

Bit Field	Access	Definition
31:24	RO	Variant: The design variant (i.e., specific set of features) implemented. Encoded as <b>01h</b> in the QMRD.
23:16	RO	Revision: The specific version of a given variant implemented. Encoded as <b>10h</b> in the QMRD.
15:0	RO	Configuration: A record of the compile-time configuration choices, as follows: <b>15:12</b> : Unused <b>11:10</b> : MAX_CLASS_CAPACITY <b>9:6</b> : MAX_QUEUE_CAPACITY <b>5</b> : EN_FLOW_CONTROL <b>4</b> : EN_WRED <b>3</b> : EN_TAIL_DISCARD <b>2</b> : EN_STATS_REG <b>1</b> : INGRESS_MODE (1=Ingress, 0=Egress) <b>0</b> : BUFFER_SIZE Note that the values of MAX_CLASS_CAPACITY and MAX_QUEUE_CAPACITY reflect the automatic enforcement of constraints. (They may differ from the user settings.)

### Queue Manager Control Register

The Queue Manager Control Register (Table 19) supports high-level control of the QMRD.

Table 19: Queue Manager Control Register

Bit Field	Reset	Access	Definition
31:8	0	RW	Source Fabric Port: The value used to fill the Source Fabric Port field in the Port PDU Tag in Ingress Mode.
7:3	0	RO	Reserved.
2	0	RO	RAM Ready <b>1:</b> QDR I/F and BRAM are operational. <b>0:</b> QDR I/F and BRAM are not operational
1	0	RW	Queue Manager Enable <b>Write 1:</b> Triggers transition from Idle to Active state. <b>Write 0:</b> Triggers transition from Active to Idle state. Read returns 1 for Active state, 0 for Idle state.
0	0	RW	Queue Manager Soft Reset <b>Write 1:</b> Triggers Soft Reset. <b>Write 0:</b> No effect. Read always returns 0.

### Queue Configuration Register

The Queue Configuration Register (Table 20) sets the actual queue and class capacities. The following illegal accesses will result in a Management Interface time-out:

- Attempt to write this register when the QMRD is in the Active state.
- Attempt to write a configuration which violates the Maximum Queue Capacity or Maximum Class Capacity

It should be noted that if a PDU arrives with a Queue ID exceeding the programmed Queue Capacity, it will simply be wrapped around. That is, with Queue Capacity set to 512, a PDU with Queue ID = 512 will be stored in queue 0.

Table 20: Queue Configuration Register

Bit Field	Reset	Access	Definition
31:6	0	RO	Reserved
5:4	0	RW	Class Capacity: <b>00</b> : 1 Class <b>01</b> : 2 Classes <b>10</b> : 4 Classes <b>11</b> : 8 Classes
3:0	0	RW	Queue Capacity <b>0h</b> : 256 Queues <b>1h</b> : 512 Queues <b>2h</b> : 1K Queues <b>3h</b> : 2K Queues <b>4h</b> : 4K Queues <b>5h</b> : 8K Queues <b>6h</b> : 16K Queues <b>7h</b> : 32K Queues <b>8h</b> : 64K Queues <b>&gt;8h</b> : Reserved

### Interrupt Source Register

The bits in the Interrupt Source Register (Table 21) indicate occurrence of the corresponding condition when they have the value "1". The non-fatal interrupts can be cleared in bit-wise fashion by writing "1". Fatal interrupts can only be cleared by a reset.

Table 21: Interrupt Source Register

Bit Field	Reset	Access	Definition
31:15	0	RO	Reserved
14	0	RCW	Scheduling Interface Pipeline Overflow
13	0	RCW	DCR Access Failure
12	0	RCW	Free List Exhaustion
11	0	RCW	Class PDU Count Overflow
10	0	RCW	Parity Error
9	0	RCW	Assembly Format Error
8	0	RCW	Oversized Port PDU Protocol Error
7	0	RCW	Undersized Port PDU Protocol Error
6	0	RCW	Incorrect Fabric PDU Size Protocol Error
5	0	RCW	Missing EOF Protocol Error
4	0	RCW	Missing SOF Protocol Error
3	0	RO	Dequeue Status Corruption (Fatal Error)
2	0	RO	Buffer Memory Linked List Corruption (Fatal Error)
1	0	RO	Pointer Memory Linked List Corruption (Fatal Error)
0	0	RO	Assembly Queue Linked List Corruption (Fatal Error)

### ***Interrupt Mask Register***

The bits in the Interrupt Mask Register (Table 22) are set to "1" to enable the corresponding interrupt, and cleared to "0" to disable it.

**Table 22: Interrupt Mask Register**

Bit Field	Reset	Access	Definition
31:15	0	RO	Reserved
14	0	RW	Scheduling Interface Pipeline Overflow
13	0	RW	DCR Access Failure
12	0	RW	Free List Exhaustion
11	0	RW	Class PDU Count Overflow
10	0	RW	Parity Error
9	0	RW	Assembly Format Error
8	0	RW	Oversized PDU Protocol Error
7	0	RW	Undersized PDU Protocol Error
6	0	RW	Incorrect Fabric PDU Size Protocol Error
5	0	RW	Missing EOF Protocol Error
4	0	RW	Missing SOF Protocol Error
3:0	Fh	RO	Reserved (Fatal errors non-maskable)

### ***Parity Error Count Register***

The Parity Error Count Register (Table 23) indicates the total number of parity errors from all sources. The count is cleared atomically by a read, to prevent missing any occurrences.

**Table 23: Parity Error Count Register**

Bit Field	Reset	Access	Definition
31:16	0	RO	Reserved
15:0	0	RCO	Parity Error Count

### ***Free List Status Register***

The Free List Status Register (Table 24) indicates the length in Buffer Slots of the Free List. Since the Free List is initialized by software, the length and the head and tail pointers must be initialized as well.

The Free List Length is writable only when the QMRD is in the Idle state. At the transition to the Active state, the Free List Length value is used to initialize the Free List pointers only if Free List Pointer Init is set. During operation, the Free List Length is incremented and decremented to reflect the actual state of the Free List. (Writing the Free List Length must be done *after* the linked list is set up in Queue Memory.)

The intent is to support proper initialization of the Free List, while allowing temporary disabling of the QMRD with the Free List intact for debugging.

Table 24: Free List Status Register

Bit Field	Reset	Access	Definition
31	0	RW	Free List Pointer Init <b>0:</b> Make no changes to the Free List upon entry to the Active state. <b>1:</b> Upon entry to the Active state, initialize Free List pointers based on the Free List Length field.
30:24	0	RO	Reserved
23:0	0	RW*	Free List Length (Buffer Slots) Writable only when QMRD is in Idle state.

### Queue Memory Address Register

The Queue Memory Address Register (Table 25) contains the location within the Queue Memory that is referenced by the Queue Memory Data Register. This location is a word address, i.e., it has a granularity of four bytes.

The Queue Memory Address can be automatically incremented after each access to the Queue Memory Data Register. The default increment is one word, and it can also be set to reflect the Buffer Size. The latter setting is useful for initializing the Free List. Auto-increment should not be used when intermixing reads and writes. Whenever the access direction is changed, the Queue Memory Address Register should be rewritten.

Table 25: Queue Memory Address Register

Bit Field	Reset	Access	Definition
31	0	RW	Auto-Increment Step: <b>0:</b> Increment by 1 <b>1:</b> Increment by Buffer Size
30	0	RW	Auto-Increment Enable: <b>0:</b> Auto-increment disabled <b>1:</b> Auto-increment enabled
29:28	0	RO	Reserved
27:0	0	RW	Queue Memory Address

### Queue Memory Data Register

The Queue Memory Data Register (Table 26) represents the Queue Memory location referenced by the Queue Memory Address Register. Reads and writes of the Queue Memory Data Register take effect at the corresponding location in the Queue Memory. Read data is actually prefetched when the Queue Memory Address Register is written (or auto-incremented) to avoid a Management Interface time out.

Table 26: Queue Memory Data Register

Bit Field	Reset	Access	Definition
31:0	0	RW	Queue Memory Data



### Flow Control Management Registers

The Flow Control Management Registers are organized in banks according to class, as shown in Table 27. The definitions of the individual registers are shown in Table 28 through Table 30. A seed value is included for the WRED random number generator, to facilitate testing. (The seed is only sampled at the transition into Active state.)

Table 27: Flow Control Management Register Addresses

Address	Usage
02Fh : 02Ch	Class 7 Registers
02Bh : 028h	Class 6 Registers
027h : 024h	Class 5 Registers
023h : 020h	Class 4 Registers
01Fh : 01Ch	Class 3 Registers
01Bh : 018h	Class 2 Registers
017h : 014h	Class 1 Registers
013h	Class 0 WRED Control Register
012h	Class 0 Stage 3 Threshold Register
011h	Class 0 Stage 2 Threshold Register
010h	Class 0 Stage 1 Threshold Register

Table 28: Stage 1 Threshold Register

Bit Field	Reset	Access	Definition
31:24	0	RO	Reserved
23:0	FFFFFFh	RW	Threshold used for Flow Control Messages

Table 29: Stage 3 Threshold Register

Bit Field	Reset	Access	Definition
31:24	0	RO	Reserved
23:0	FFFFFFh	RW	Threshold used for Tail Discard

Table 30: WRED Control Register

Bit Field	Reset	Access	Definition
31:24	0	RO	Reserved
23:16	FFh	RW	WRED Random Number Seed
15:9	0	RO	Reserved
8	0	RW	TCP Flag 1: WRED Enabled 0: WRED Disabled
7:0	0	RW	WRED Weight

### Class Statistics Registers

The Class Statistics Registers are organized in banks according to class, as shown in [Table 31](#). The definitions of the individual registers are shown in [Table 32](#) and [Table 33](#). Note that the discard counts include only Port PDUs discarded by congestion management, not those discarded for protocol errors.

**Table 31: Class Statistics Register Addresses**

Address	Usage
03Fh : 03Eh	Class 7 Registers
03Dh : 03Ch	Class 6 Registers
03Bh : 03Ah	Class 5 Registers
039h : 038h	Class 4 Registers
037h : 036h	Class 3 Registers
035h : 034h	Class 2 Registers
033h : 032h	Class 1 Registers
031h	Class 0 Discard Count Register
030h	Class 0 PDU Count Register

**Table 32: Class PDU Count Register**

Bit Field	Reset	Access	Definition
31	0	RCO	Rollover Bit 0: No rollover has occurred 1: Rollover has occurred
30:0	0	RO	Total Number of Port PDUs passed.

**Table 33: Class Discard Count Register**

Bit Field	Reset	Access	Definition
31	0	RCO	Rollover Bit 0: No rollover has occurred 1: Rollover has occurred
30:24	0	RO	Reserved
23:0	0	RO	Total Number of Port PDUs discarded.

### Class Size Registers

The Class Size Registers ([Table 34](#)) indicate the current number of Buffer PDUs that are stored for each class. The Size Registers for classes 0 through 7 are at addresses **040h** through **047h**, respectively.

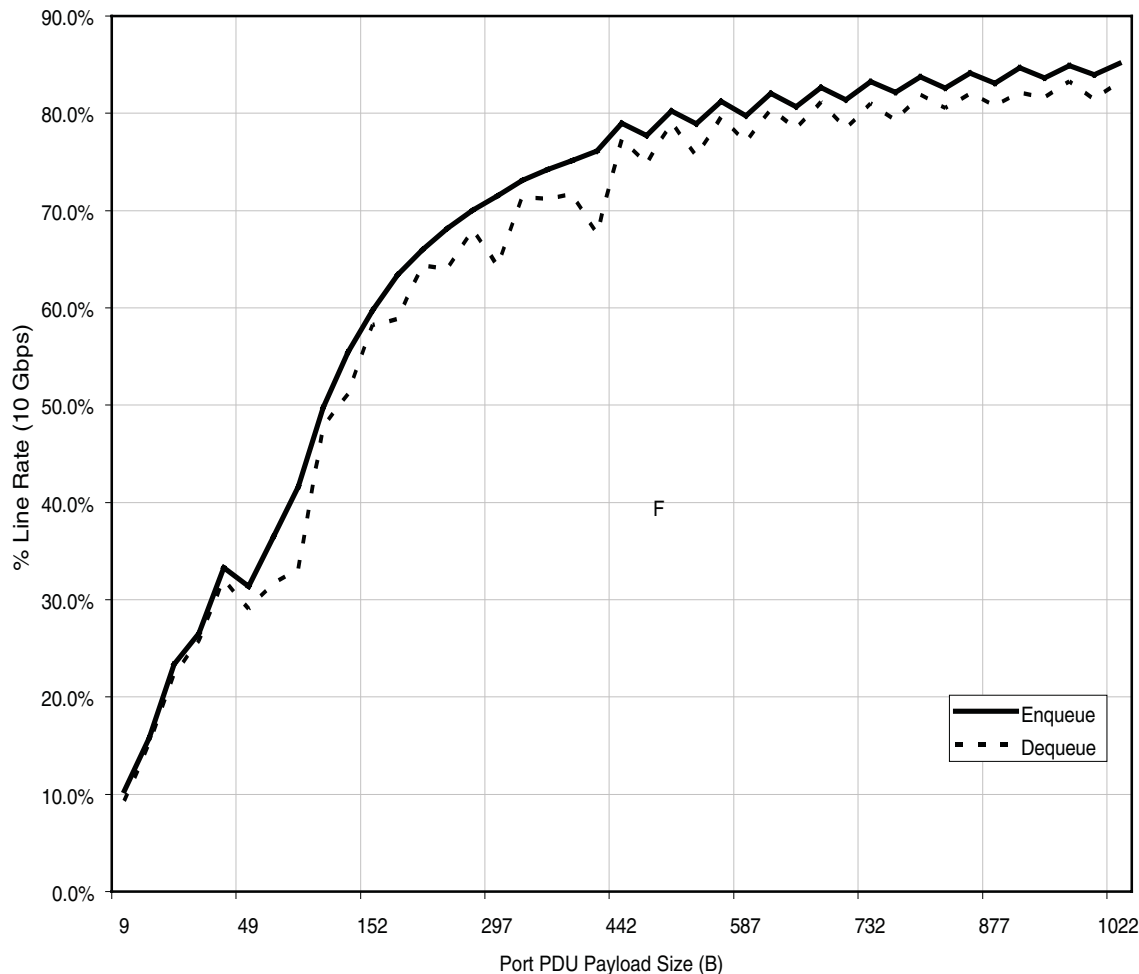
**Table 34: Class Size Registers**

Bit Field	Reset	Access	Definition
31:24	0	RO	Reserved
23:0	0	RO	Total Number of Buffer PDUs currently stored.

## Performance

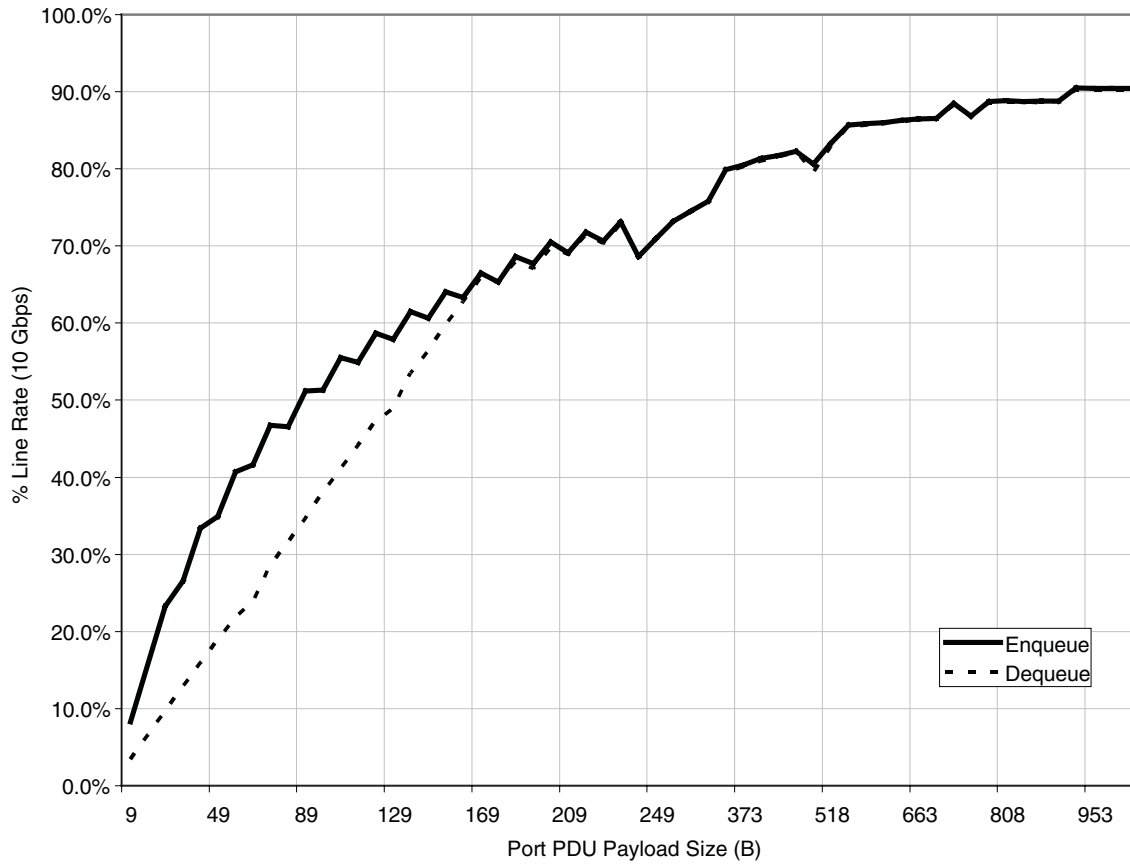
The data rates that can be sustained by the QMRD are plotted in Figure 15 through Figure 18 over different Port PDU sizes. These curves are based on measurements made in simulation, and represent actual times to transfer a known payload. Data rates are plotted separately for enqueue and dequeue, although due to the nature of the test the dequeue rate is limited by the enqueue rate.

There are a number of issues which account for data rate limitations, but the main culprit is latency in accesses to the Queue Memory. The Queue Memory is accessed by the Enqueue Unit, Dequeue Unit and Free List Manager, and the resulting arbitration and collisions exacerbate latency. This is especially true when traversing a linked list of small Buffer PDUs, because accesses cannot be pipelined. Limited performance in egress mode with 64B buffer slots is a prime example. In addition, these performance numbers are effected by packet overhead induced by the various PDU headers.



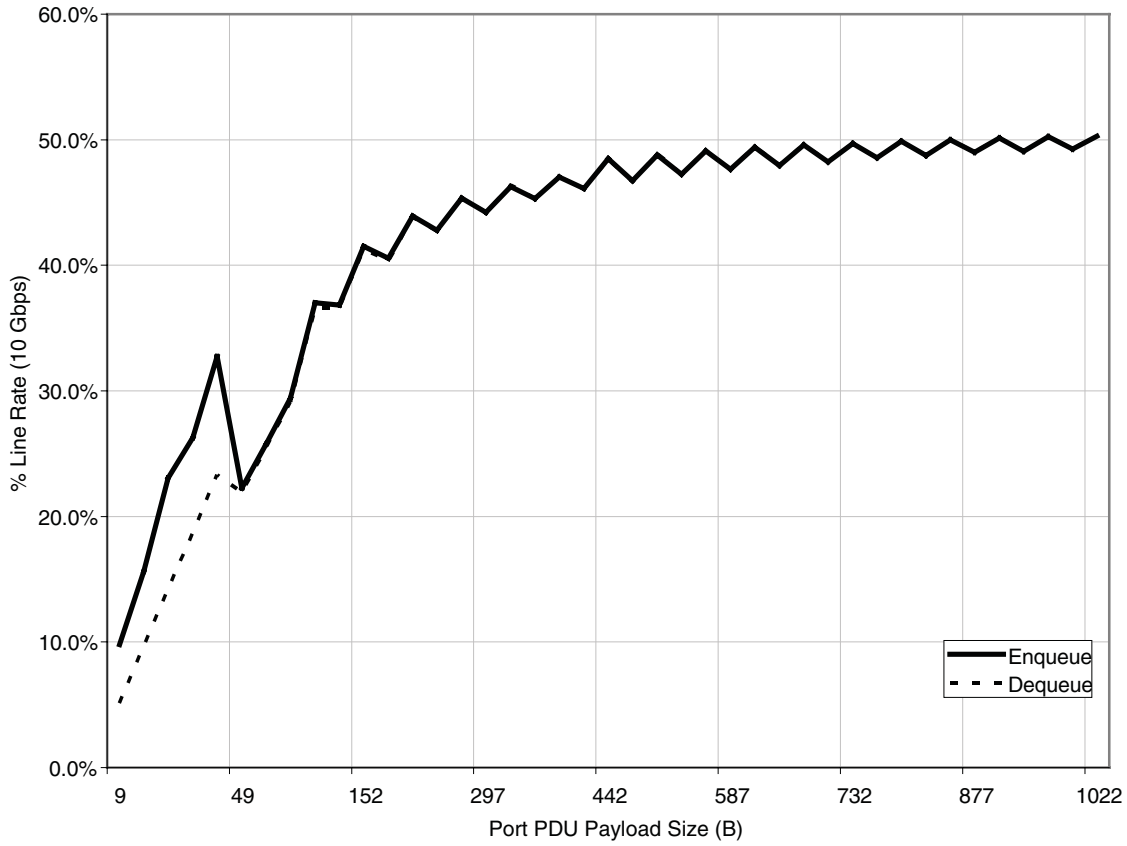
XAPP511\_15\_041007

Figure 15: QMRD Ingress Performance – 64B Buffer Slots



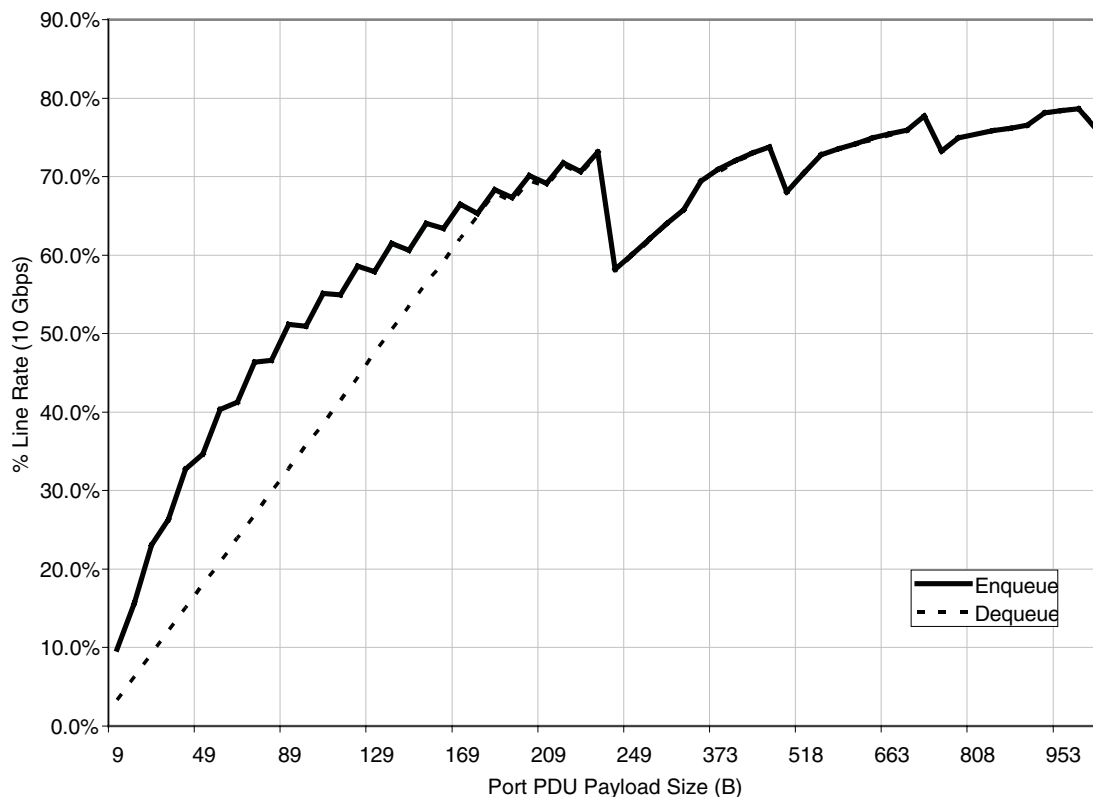
XAPP511\_16\_041007

Figure 16: QMRD Ingress Performance – 256B Buffer Slots



XAPP511\_17\_041007

Figure 17: QMRD Egress Performance – 64B Buffer Slots



XAPP511\_18\_041007

Figure 18: QMRD Egress Performance – 256B Buffer Slots

## Resource Usage

QMRD resource usage is summarized in [Table 35](#) through [Table 37](#). The GPIO total listed includes only external I/O, not internal connections to other modules. The only external connections are for the Queue Memory Interface.

In order to ease system integration, clocking resources (DCMs and global clock buffers) have largely been kept external to the QMRD itself. The exceptions are a single DCM and clock buffer in the Queue Memory Interface. QDR clocking architecture may change depending on the board on which the QMRD is used.

[Table 36](#) and [Table 37](#) describe the extremes of QMRD resource usage, covering the extent of configurable features. [Figure 19](#) provides a more detailed picture of how BRAM usage varies with Maximum Queue Capacity. [Figure 20](#) offers a representative view of logic usage variation, illustrating flip-flop usage for a subset of configurations (egress mode, 64K queue capacity, 64 byte buffer size). The different curves represent the following feature sets:

- All Disabled: All flow control features disabled; statistics disabled.
- Stats Disabled: All flow control features enabled; statistics disabled.
- All Enabled: All flow control features enabled; statistics enabled.

It should be noted that the majority of flow control logic is dedicated to WRED, therefore separate curves for the different flow control options are not shown.

As an example, both an Ingress and an Egress Queue Manager, configured for 4096 queues and eight classes with all features enabled, will fit in an XC2V3000 device.

Table 35: Clock and I/O Usage

Design Configuration	DCM	GCLK	MGT	GPIO
Ingress	1	1	0	146
Egress	1	1	0	146

Table 36: Memory Usage

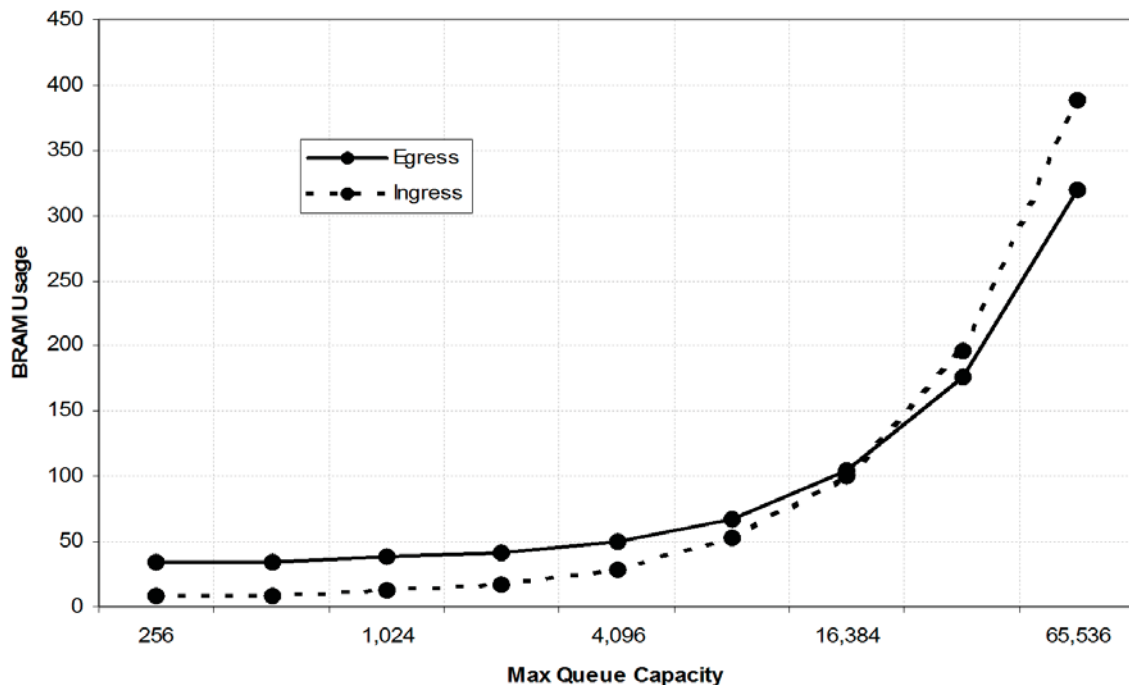
Design Configuration	BRAM	MEM16	DPRAM	DPRAM32
Ingress Buffer Size=64 Queue Capacity=256 Class Capacity=1 Flow Control Fully Disabled	9	85	156	156
Egress Buffer Size=64 Queue Capacity=256 Class Capacity=1 Flow Control Fully Disabled	35	57	156	156
Ingress Buffer Size=256 Queue Capacity=64K Class Capacity=8 Flow Control Fully Enabled	389	85	156	156
Egress Buffer Size=256 Queue Capacity=64K Class Capacity=8 Flow Control Fully Enabled	320	57	156	156

Table 37: Logic Usage

Design Configuration	FG	CY	DFF	BUFT	FG5	FG6	FG7	FG8
Ingress Buffer Size=64 Queue Capacity=256 Class Capacity=1 Flow Control Fully Disabled	5798	580	5562	2112	280	15	1	0
Egress Buffer Size=64 Queue Capacity=256 Class Capacity=1 Flow Control Fully Disabled	5885	796	5613	2112	346	101	18	0

Table 37: Logic Usage (Continued)

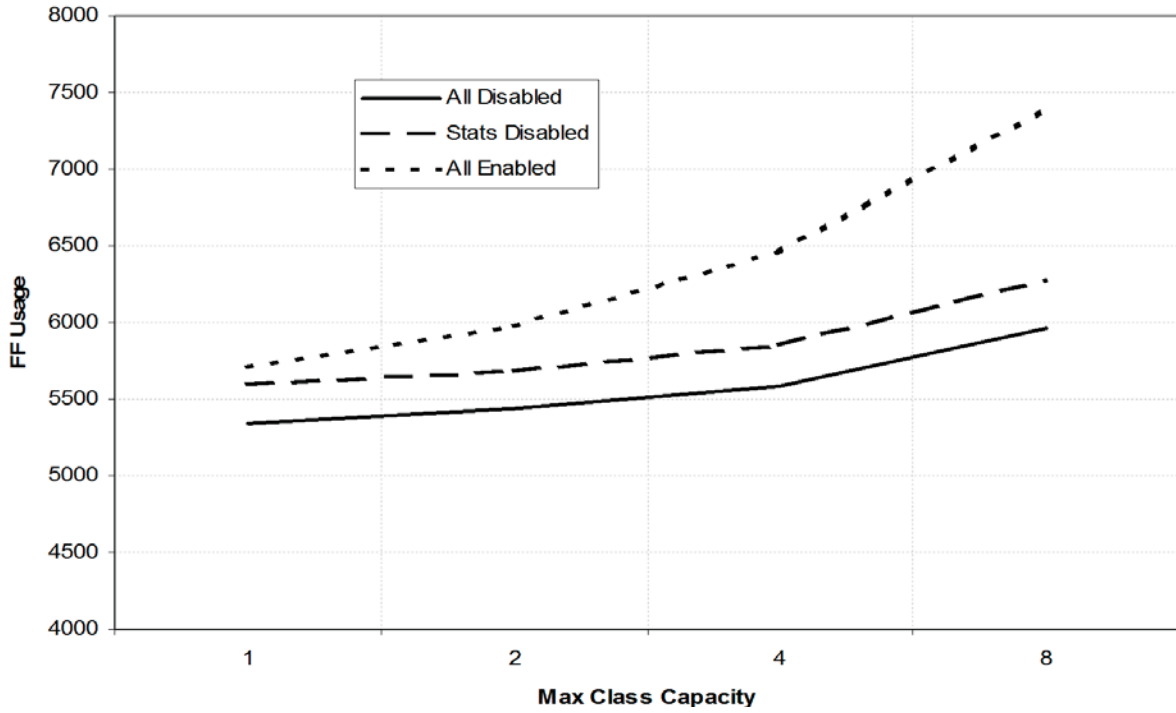
Design Configuration	FG	CY	DFF	BUFT	FG5	FG6	FG7	FG8
Ingress Buffer Size=256 Queue Capacity=64K Class Capacity=8 Flow Control Fully Enabled	10592	1901	8186	2112	278	90	12	216
Egress Buffer Size=256 Queue Capacity=64K Class Capacity=8 Flow Control Fully Enabled	10731	2102	8091	2112	351	134	10	162



XAPP511\_19\_042607

Figure 19: Block RAM Usage vs. Maximum Queue Capacity





XAPP511\_20\_042607

Figure 20: Flip-Flop Usage vs. Maximum Class Capacity

## Validation

Verification using both simulation and actual hardware was a project requirement. Unfortunately, the lack of infrastructure made creation of a real test system impractical. This was especially true at the high data rates involved. In order to meet the hardware validation requirement, a synthesizable test harness has been created as shown in Figure 21. This supports running the same tests in both simulation and in hardware.

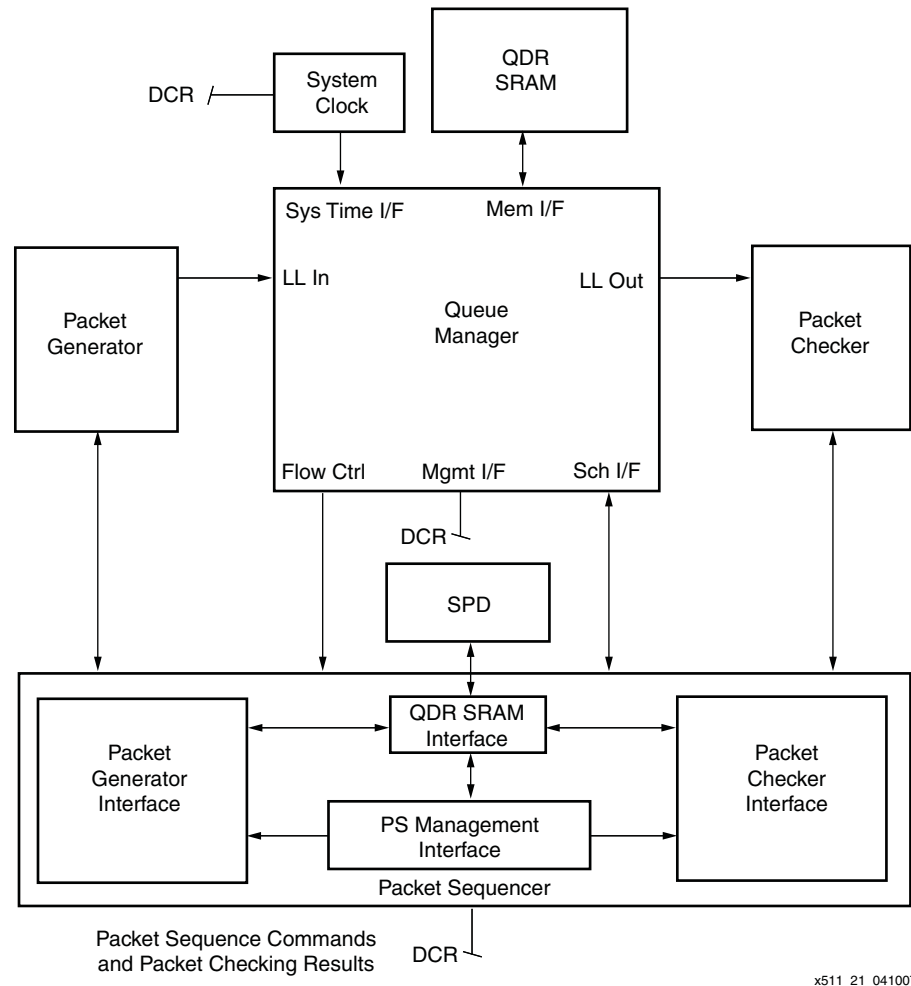


Figure 21: QMRD Test Environment

The test harness is based primarily on automatic packet generation and packet checking, under the supervision of the Packet Sequencer. The Sent Packet Database (SPD) is used to manage packet characteristics. The test harness is a DCR slave which can be accessed using C code, via the PLI interface in simulation or from a MicroBlaze™ processor in actual hardware. Hardware testing was done on the Xilinx Virtex™-II QDR board, which provides QDR memories and a serial port for interfacing with the MicroBlaze.

### DCR Access Test

This test confirms basic operation of the Management Interface and the programmable features in the test harness.

### Single Packet Test

Basic operation of the QMRD is confirmed with one Port PDU at a time, to a single queue.

**Multiple Packets and Queues Test**

Testing is expanded to cover multiple PDUs back-to-back over multiple queues.

**Input LocalLink Interface Protocol Test**

The Input LocalLink Interface is tested rigorously, including introduction of errors and wait states.

**Parity Error Test**

Response to parity errors is tested, using programmable test and debug features in the QMRD.

**Flow Control Test**

The various flow control mechanisms are tested, including the related configuration options.

**Miscellaneous Test**

This test encompasses a number of minor features, including the function of various registers and header fields during operation.

**Random Traffic Test**

A comprehensive test in which packet sizes, timing and error introductions vary randomly. This is run for extended periods in hardware.

## Design Hierarchy

The directory structure of the QMRD is shown below, with the various directories.

source	Verilog source and implementation files
___ {root}	Global definitions file containing defines for global debugging features
___ coregen_qmc	Files generated by Coregen for cores used by the QMRD
___ implementation	User constraints file, bitfile configuration, and ISE compile scripts
___ cfg1_ingress	- For the various test configurations
___ cfg1_egress	
___ cfg4_ingress	
___ cfg4_egress	
___ rtl	Verilog source tree (for QMRD files)
___ congestion_unit	
___ dcms	
___ dequeue_unit	
___ enqueue_unit	
___ fifos	
___ freelist_manager	
___ local_link_if	
___ management_if	
___ qdr_sram_if	
___ queue_ptr_mem	
___ ram	
___ sync	
___ traffic_scheduler_if	
___ xilinx_cores	
___ synplify	Synthesis compile script and results (after synthesis)
___ utilities	Revision control scripts

## References

1. LocalLink Interface Specification, [Xilinx DS290, LocalLink Interface Specification, Version 1.0, October 18, 2002](#).
2. AMIRIX Systems Inc., DOC-003445, Queue Manager Reference Design Test Report.
3. CoreConnect 32-Bit Device Control Register Bus Architectural Specification Version 2.9.

## Customization

Use of the QMRD in actual systems is likely to involve customization, typically to interfaces and packet header formats. In addition, customization of the Queue Memory Interface to match a specific board implementation is very likely.

AMIRIX Systems is a development partner with Xilinx and offers customization services for the QMRD.

Contact:

AMIRIX Systems Inc.

77 Chain Lake Drive

Halifax, NS, Canada B3S 1E1

[www.amirix.com](http://www.amirix.com)

Tel: (902) 450-1700

Toll Free: 1-866-426-4749

Fax: (902) 450-1704

E-mail: [info@amirix.com](mailto:info@amirix.com)

---

---

## Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/09/04	1.0	Initial Xilinx release.
05/04/07	1.1	Updated link to reference design file.