RapidIO 2.1 Serial Endpoint IP Core User's Guide

June 2011
IPUG84_01.3

Table of Contents

The RapidIO 2.1 interconnect specification is an open standard developed by the RapidIO Trade Association. It has been designed to offer a reliable high-performance, packet-switched, interconnect for the embedded industry providing scalable chip-to-chip and board-to-board communication for microprocessors, digital signal processors (DSPs), network processors, and memories. Rapid I/O offers the generic memory and device addressing concepts of a processor bus at the user level where at the Serial RapidIO transaction level, processing is managed completely by the protocol and hidden from the user's application. The LatticeECP3™ family of low-cost, low-power FPGA devices provides a very economical platform for developing Serial RapidIO interconnect strategies.

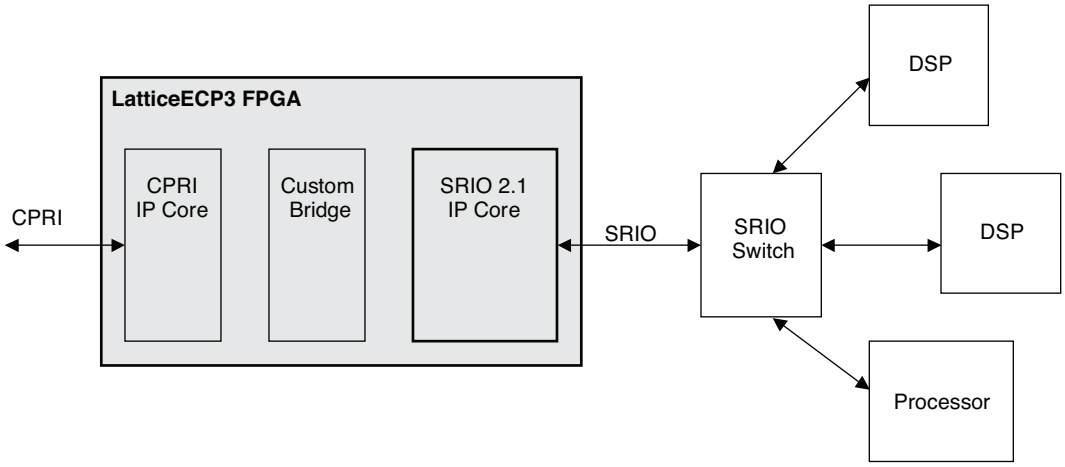
The Lattice Serial RapidIO (SRIO) Endpoint IP core is fully compliant to the following RapidIO version 2.1 Interconnect Specifications:

- Part 1: Input/Output Logical Specification
- Part 2: Message Passing Logical Specification
- Part 3: Common Transport Specification
- Part 6: LP-Serial Physical Layer Specification
- Part 8: Error Management Extensions Specification

Because the core is completely compliant to these specifications, this user's guide relies heavily upon them to aid in the explanation of core behavior and functional content. Required RapidIO behavior and functionality explained in these specifications is not replicated in this user's guide. It is recommended that users have access to, and a working level knowledge of these specifications before using the SRIO IP core.

Figure 1-1 shows an example of a typical system application.

Figure 1-1. System Application



Quick Facts

Table 1-1 gives quick facts about the SRIO IP core.

Table 1-1. Quick Facts

Core Requirements	FPGA Families Supported	SRIO IP Configuration			
		1X Mode / 2X Mode / 4X Mode			
		LatticeECP3			
Resources Utilization	Target Device	LFE3-35EA	LFE3-70EA	LFE3-95EA	LFE3-150EA
	Data Path Width	64			
	LUTs	15,749			
	sysMEM™ EBRs	16			
	Registers	10,199			
Design Tool Support	Lattice Implementation	Lattice Diamond™ 1.2			
	Synthesis	Synplicity Simplify Pro® for Lattice E-2010.09L-SP2			
	Simulation	Aldec® Active-HDL® 8.2 SP1 Lattice Edition			
		Mentor Graphics® ModelSim® 6.5B			

Features

General Features

- Compliant with RapidIO Trade Association, RapidIO Interconnect Specification, Revision 2.1, August 2009
- Includes compliance with Interconnect Specification Part 8: Error Management Extensions
- Supports all capability (CARs) and configuration and status registers (CSRs) for all three RapidIO layers
- Supports 8-bit or 16-bit device IDs
- Supports incoming and outgoing multi-cast events
- Transmitter controlled flow control
- Generic 32-bit processor interface for local register access
- Generic 32-bit interface supporting user definable register expansion accessible locally or remotely via the SRIO link.

Physical Layer Features

- 1x/2x/4x serial lane support with integrated transceivers
- Serial data rates supported: 1.25, 2.5, 3.125 GBaud
- Receive/transmit packet buffering, flow control, error detection, packet assembly and delineation
- Automatic freeing of resources used by acknowledged packets
- Automatic retransmission of retried packets
- Autonomous error recovery
- Full control over integrated transceiver parameters via SERDES SCI interface

Transport Layer Features

- Supports up to three logical layer user ports.
- Scheduling of transmission from logical layer ports based on two different selectable priority schemes - fixed or rotating.

Logical Layer Features

- Built-in maintenance request decoder/response encoder supporting SRIO link access of local CAR and CSR registers.
- Autonomous Doorbell generation on event.
- Built-in logical layer packet source/sink capability referred to as the Soft Packet Interface (SPI) accessed via a local processor control interface (AMI).

Errata

The logN_rlnk_dst_dsc_n signal may not function as expected with small sized packets. It is not recommended to use this signal in your design.

Supported Transactions

All legal SRIO logical layer transaction types are supported via the Logical Layer user interface ports and though the Soft Packet Interface.

What Is Not Supported

The core does not support the following optional features:

- Software Assisted Error Recovery
- Critical Request Flow (CRF)
- Baud Rate Discovery
- Lane n Status 1 to 7 CSRs
- Enable Inactive Lanes
- Port Initialization Test Mode
- Virtual Channels (VCs) 1 - 8

Conventions

The nomenclature used in this document is based on the Verilog language. This includes radix indications, and logical operators.

Data Ordering and Data Types

Data ordering is per the RapidIO specification, and also follows PowerPC conventions. This includes:

- Byte ordering is big-endian
- The most significant bit within data types is numbered 0

Table 1-2. Data Types

Name	Size
byte or octet	8 bits
hword – half word	16 bits
word	32 bits
dword – double word	64 bits

Signal Names

- Signal names that end with “_n” or “_N” are active low.

Functional Description

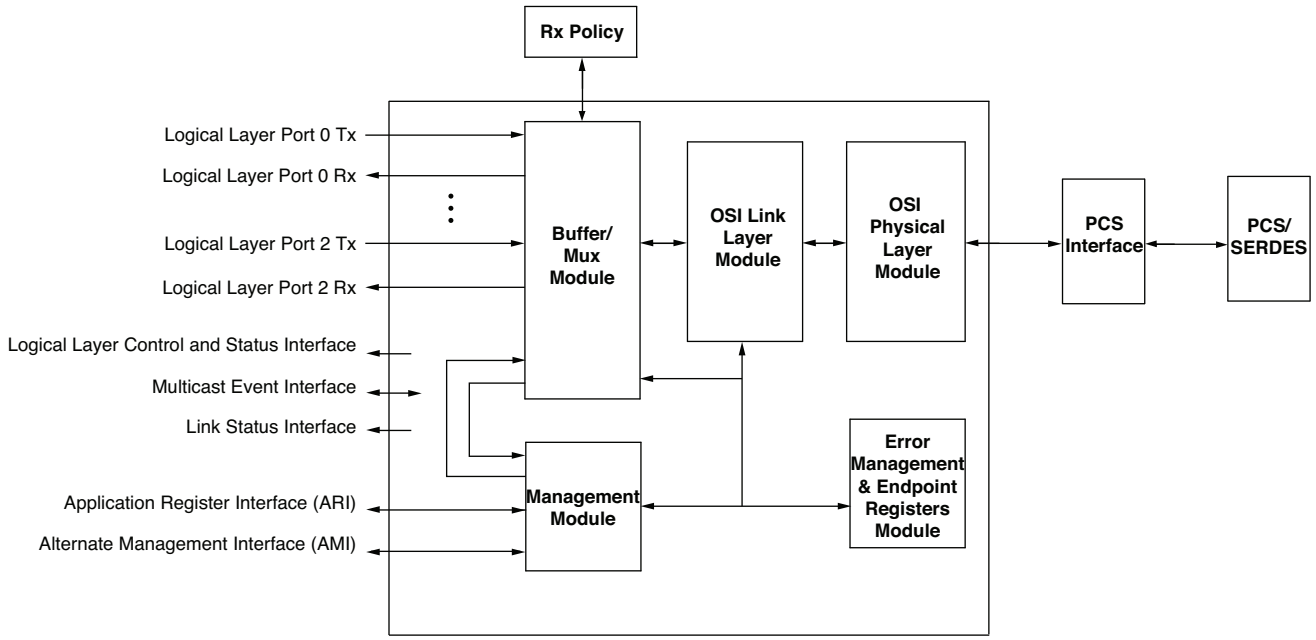
This section provides detailed descriptions of functionality that interfaces with user logic.

Overview

The SRIO IP core implements the functionality described in the RapidIO 2.1 specification for the Physical, Transport, Error Management, and Logical layers. Figure 2-1 shows the major functional blocks that make up the core as well as how it interfaces to the high-speed transceivers and user-defined logical layer functions. Built-in logical layer functionality is provided for maintenance transaction support and for a low bandwidth control plane based packet source/sink feature referred to as the Soft Packet Interface (SPI). Three bi-directional logical layer interface ports are available for user defined functions such as Initiator, Target, and Doorbell functions. The core consists of the following major functional blocks:

- **OSI Physical Layer Module** – Implements RapidIO physical layer functions that are defined in the OSI physical layer.
- **OSI Link Layer Module** – Implements RapidIO physical layer functions that are defined in the OSI link layer.
- **Buffer Mux Module** – Implements buffering for packet transmission and reception. It also manages the multiplexing and de-multiplexing of traffic to and from the Logical Layer ports.
- **Management Module** – Implements the functions needed to access Control and Status Registers (CSRs) both locally or remotely and provides an external interface for user defined register expansion. It also includes a subsystem used to monitor interface events.
- **Error Management and Endpoint Registers Module** – Implements logical layer error management CSRs as well as various endpoint CSRs.

Figure 2-1. LP-Serial Endpoint Core Architecture



User Interface

The core interacts with the user application through eight types of interfaces:

- **Logical Layer Ports** – The logical layer ports are used to connect user implemented logical layer functions to the core. Each Logical layer port consists of a 64-bit receive and transmit interface, an error reporting interface supporting error management extensions, and a User Event interface.
- **Alternate Management Interface** – This interface is used to access local CSRs within the core and user defined registers outside the core via the ARI described below. It is also used to access the Soft-Packet Interface (SPI) FIFO feature that allows creation and termination of RapidIO packets though software.
- **Application Register Interface** – This interface is used by the core to access user defined CSRs. This interface provides an efficient way for users to add additional registers to their design and can be accessed through maintenance commands as well from the far-end.
- **Logical Layer Common Control and Status Interface** – This interface presents information contained in the logical layer control CSRs contained in the core, transmit flow control information, and general logical port interface status.
- **Multicast Event Interface** – This interface provides support for generating RapidIO control symbols containing the multicast event function code. It also indicates when a control symbol has been received that contains the multicast event function code.
- **Link Status Interface** – The status interface includes both generic information about Tx and Rx activity and RapidIO port state as well as interface training information specific to LP-Serial ports.
- **Receive Policy Interface** – This interface controls the de-multiplexing of request and response packets received from the RapidIO link to Logical link and Management Rx ports on the Buffer Mux Module.
- **Transceiver Interface** – The transceiver interface connects the SRIO IP core to the PCS interface logic that interfaces with the built-in SERDES.

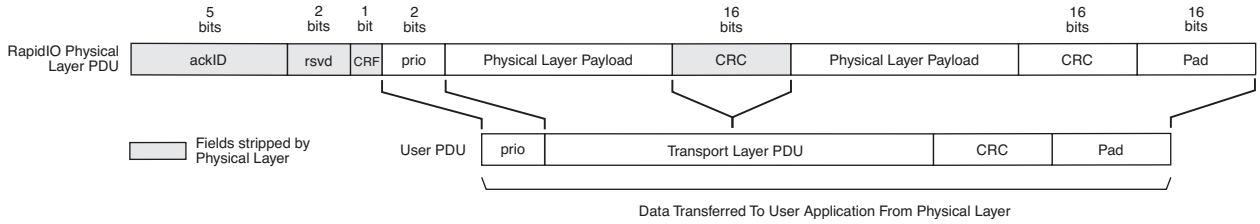
In addition to the functions included in the core proper, there is a user customizable Receive Policy module that is delivered as Verilog source code. A Physical Coding Sub-layer interface module is also provided in Verilog format and provides the interface between the core and the device built-in SERDES functions.

Packet Transmission and Reception

The SRIO core presents a reliable transport to logical layer functions connected to the logical layer ports. This means that link retries due to error recovery and congestion are handled by the physical layer and are transparent to logical layer functions.

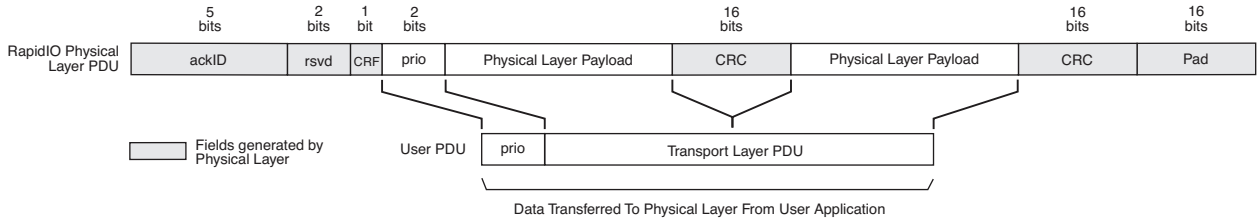
Received packets have the first eight bits of the physical layer overhead stripped along with the mid-span CRC, if present. The final CRC and possible pad are not stripped and can be discarded by the logical layer. Figure 2-2 illustrates the handling of received packets.

Figure 2-2. Receive Packet Handling



For transmitted packets the physical layer core adds the first octet of the physical layer overhead along with the mid-span CRC, final CRC, and pad as needed. Figure 2-3 illustrates the handling of transmitted packets.

Figure 2-3. Transmit Packet Handling



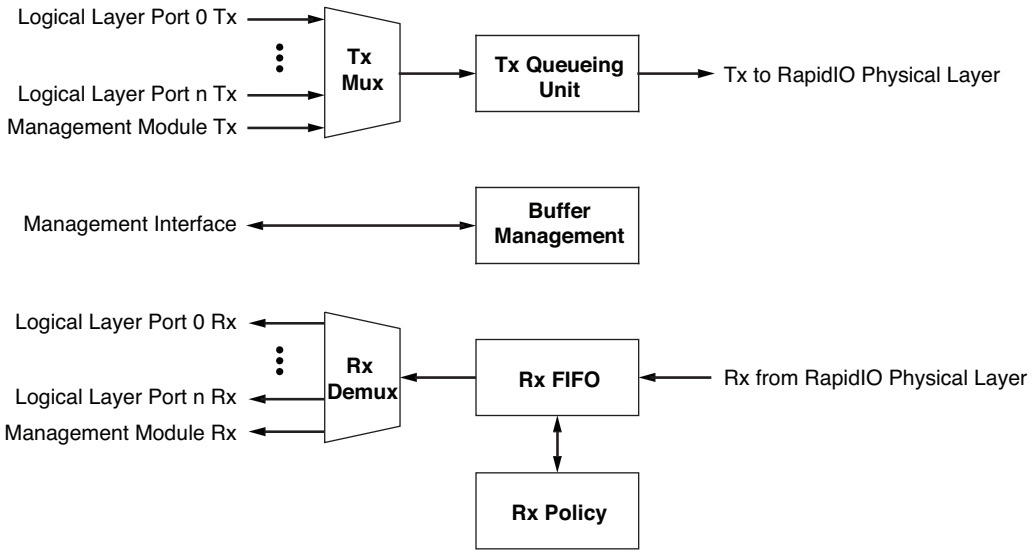
Module Descriptions

Buffer Mux Module

The Buffer/Mux module contains the following functional blocks:

- **Tx Queuing Unit** – Buffers outgoing RapidIO Logical layer packets for transmission, and handles retransmission in response to retry and error indications from the far end of the link. The maximum number of maximum length RapidIO packets that can be buffered is fixed in this version of the IP core to 14.
- **Tx Mux** – Multiplexes packets from the external logical layer functions, and the internal Management Module to the Tx FIFO.
- **Rx FIFO** – Buffers incoming RapidIO Logical layer packets. This buffer not only provides elasticity, but it also filters out packets with receive errors so they are not visible to logical layer functions. The FIFO can store up to seven, maximum length, RapidIO packets.
- **Rx Demux** – De-multiplexes incoming packets to one of the Logical layer Rx ports or to the internal Management Module Rx port.
- **Rx Policy** – Controls the operation of the Rx Demux and determines whether a packet will be stored in the Rx FIFO based on user defined criteria. This block is implemented outside the core itself. A reference implementation of an Rx Policy in Verilog source code form is included with the core.

Figure 2-4. Buffer Mux Module Block Diagram

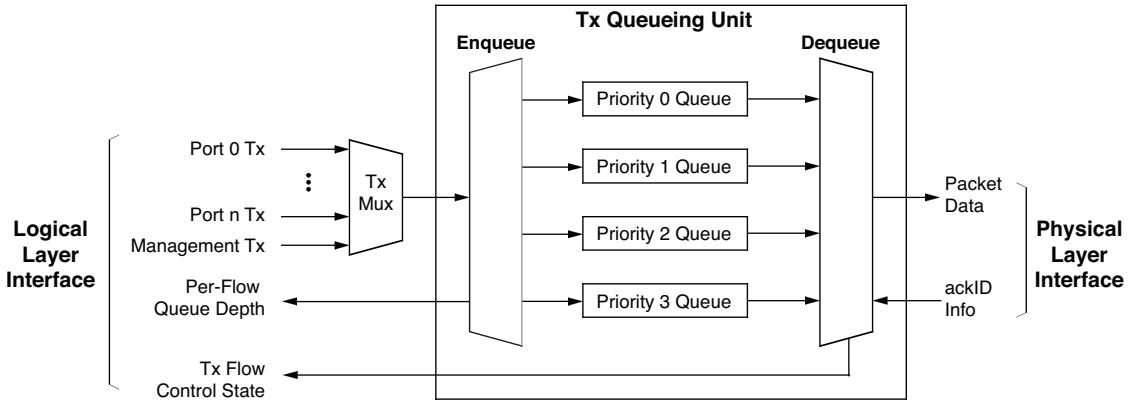


Packet Transmission

The transmission logic multiplexes logical layer packets generated by logical layer functions and queues them for transmission. This buffering eliminates the need for the logical layer functions to resend packets in the case of link errors or retries due to link congestion. There is no cut-through forwarding of packets. That is to say all packets pre-

sented at the logical layer Tx ports are completely loaded into the Tx queues before they are eligible for transmission.

Figure 2-5. Buffer Mux Tx Datapath Functional Block Diagram



Transmit Multiplexer – Tx Mux

The transmit multiplexer selects the next packet for en-queue from the packets that are currently presented on the Logical Port Transmit Interfaces as well as the management interface Transmit link. In cases where multiple packets are queued on these interfaces for transmission, the Transmit Multiplexer selects the next packet for en-queue using the algorithm shown in Figure 2-6. While the packet that was selected is being en-queued, traffic presented at the other logical layer ports is stalled. This serializes the en-queue of packets from the logical layer ports.

Arbitration is affected by the RapidIO priority of traffic queued at the multiplexer ports, and the relative priority of the Transmit Multiplexer ports themselves. Arbitration effectively occurs in two phases with the first being RapidIO priority arbitration, and the second port priority arbitration.

RapidIO Priority Arbitration

The first thing considered in the arbitration algorithm is the RapidIO priority of queued traffic. This priority information is taken from log_tlnk_d[0:1] bits from each port. Packets are forwarded in strict RapidIO priority order with RapidIO priority 3 being the highest. The highest RapidIO priority of packets enqueued at the start of an arbitration cycle is referred to as the winning RapidIO priority. If there is only one packet queued at the winning RapidIO priority then that packet will be enqueued.

Port Priority Arbitration

If there are multiple packets queued at the winning RapidIO priority, then the packet selected for forwarding is based on the relative port priorities of ports with packets enqueued at the winning RapidIO priority. There are two schemes supported (fixed and rotating) for port arbitration, with the scheme that is used configured via GUI selection during core generation:

- **Fixed Priority** – Each port has a fixed relative priority with port 0 having the highest priority and the management port having the lowest.
- **Rotating Priority** – Each port has a fixed relative priority, but the absolute priorities may update after an arbitration cycle. This is described in more detail below.

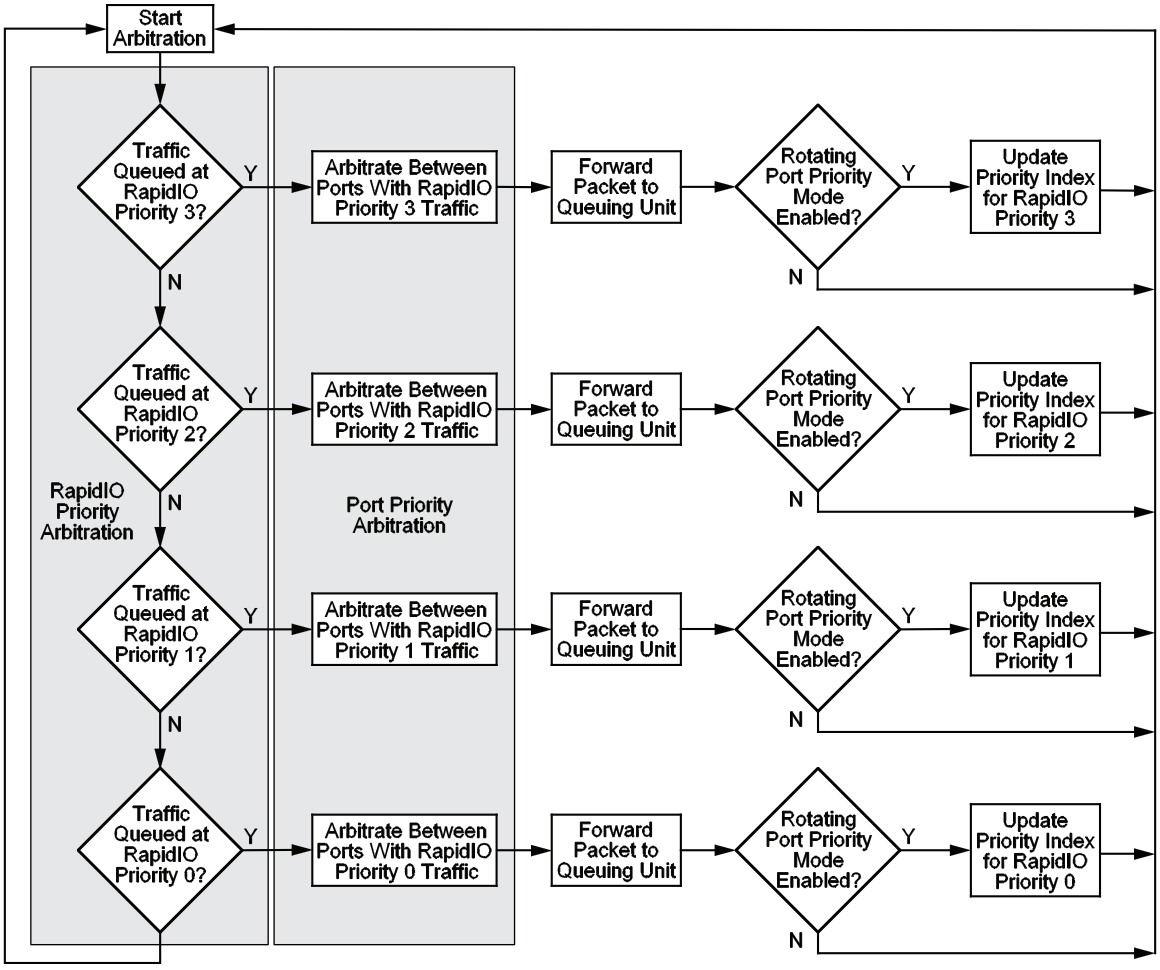
When configured for rotating priority the relative priorities of the ports remain constant, but the port that won the last port arbitration contest is assigned the lowest priority for the next contest. This has the effect of rotating the port priority. An internal variable called the priority index determines which port is assigned the highest port priority during arbitration. After each arbitration contest this value is updated to equal the winning port number plus one. A separate priority index is kept for each RapidIO priority level.

Table 2-1 shows an example of how the priority index would be updated and port priorities change over several contests at the same RapidIO priority level.

Table 2-1. Rotating Port Priority Example

	Arc	Contest 1	Contest 2	Contest 3	Contest 4
Port Priority	Highest	Logical Port 0	Logical Port 3	Management Port	Logical Port 2
		Logical Port 1	Management Port	Logical Port 0	Logical Port 3
		Logical Port 2	Logical Port 0	Logical Port 1	Management Port
		Logical Port 3	Logical Port 1	Logical Port 2	Logical Port 0
	Lowest	Management Port	Logical Port 2	Logical Port 3	Logical Port 1
Priority Index		Port 0	Port 3	Management Port	Logical Port 2
Winning Port		Port 2	Port 3	Port 1	

Figure 2-6. Transmit Multiplexer Arbitration Algorithm



Queuing Unit

The Queuing Unit manages the buffering of packets for transmission. Packets are stored in the Tx buffer memory implemented inside the core. This memory is divided into fixed size packet buffers of 280 bytes each. Each of these buffers is capable of storing a maximum sized RapidIO packet. The total number of buffers available for storage across all priority queues (see below) is equal to the transmit buffer memory size in bytes mod 280. For the initial design, the buffer memory is fixed at 4096 bytes and so a total of 14 buffers are available.

Packets are stored in one of four logical queues, one for each of the four RapidIO priority levels. Each of these queues can contain up to the fourteen packet maximum for the initial IP core release. It is up to user logical layer functions to control the maximum number of packets that should be loaded at each priority level so that there are buffers available when they are needed for other priorities. It may not make sense, for example, to use up all 14 buffers at the low priority queue level if other priority traffic must also be supported. In addition, at least one of the fourteen buffers should always be left available for response packets associated with the Management unit (maintenance requests) in order to avoid dead-lock conditions. This procedure is referred to as the ingress queue threshold policy.

The ingress queue policy should also take into consideration the fill levels of buffers at the far-end of the link when Transmitter Flow Control is enabled. In order support the implementation of the ingress queue threshold policy the core provides the following information to the user logical layer functions via the Logical Layer Common Control and Status Interface:

- The current depth of each of the local priority queues as well as a total depth. This information is presented on the tx_flow_depth vector.
- The current state of the Tx flow control state machine. This information is presented on the tx_flow_ctrl_state vector and provides the fill levels of the buffers at the far-end of the link when Transmitter Flow Control is being used. If Transmitter Flow Control is not being used, this vector can be ignored.

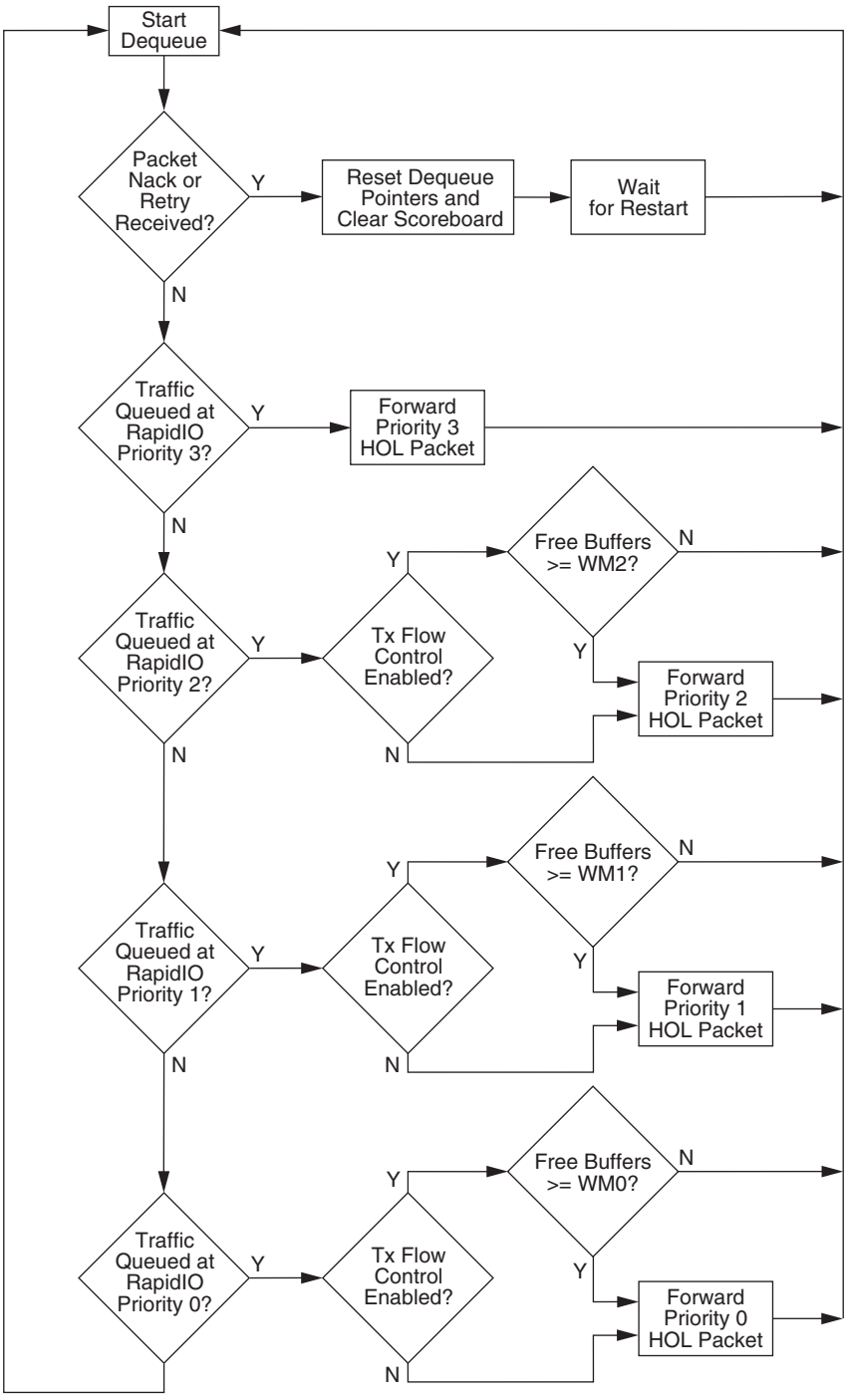
Enqueue Policy
Packets delivered by the Tx Mux to the Queuing Unit are queued based on their RapidIO priority. This priority information is captured from log_tlnk_d[0:1] on the first beat of a packet transfer. Packets are not accepted for enqueue when all packet buffers in the queuing unit are currently filled.

Dequeue Policy
The Dequeue Policy selects queued packets for forwarding over the link. Factors that affect which packet is selected include:

- Packet priorities
- Tx flow control, if enabled
- Retransmission state based on errors or retries

The algorithm used by the Dequeue Policy for packet selection is shown in Figure 2-7.

Figure 2-7. Dequeue Policy Algorithm



Packet Reception

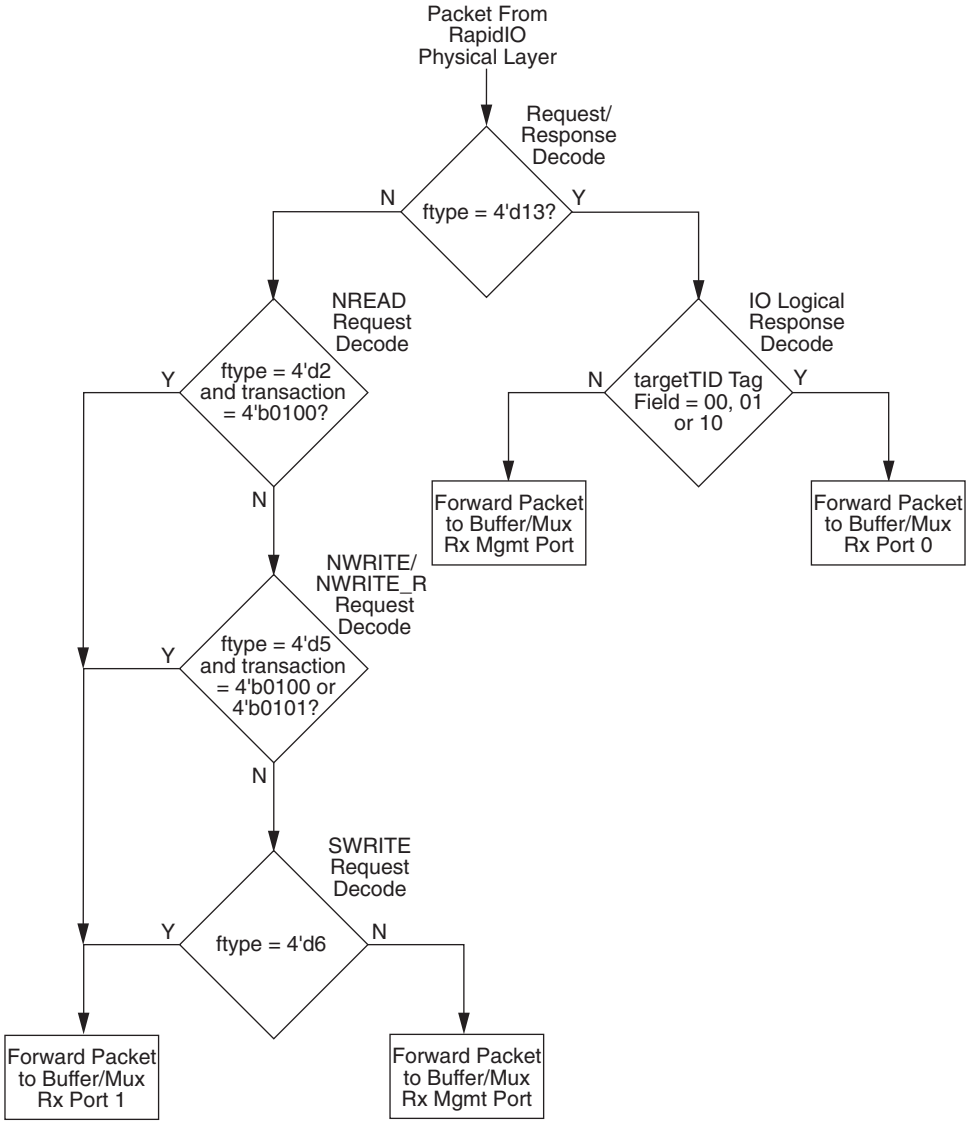
Rx Policy

This section describes how the Receive Policy is used to de-multiplex received packets to the Logical Layer ports and Management unit. The policy description shown in Figure 2-8 assumes the following configuration of logical layer functions:

- I/O Logical Layer initiator block connected to Logical Layer Port 0.
- I/O Logical Layer target block connected to Logical Layer Port 1.
- No logical layer functions connected to Logical Layer Port 2.

The reference receive policy does not force any retries of received packets. The receive policy that is provided as a reference design with the IP core design package is slightly different than the policy described below in that it also supports a doorbell unit connected to Logical Layer port 2. This reference policy can be modified to suit the arrangement of user logical layer functions connected to the core.

Figure 2-8. Reference Policy Demux Decision Tree

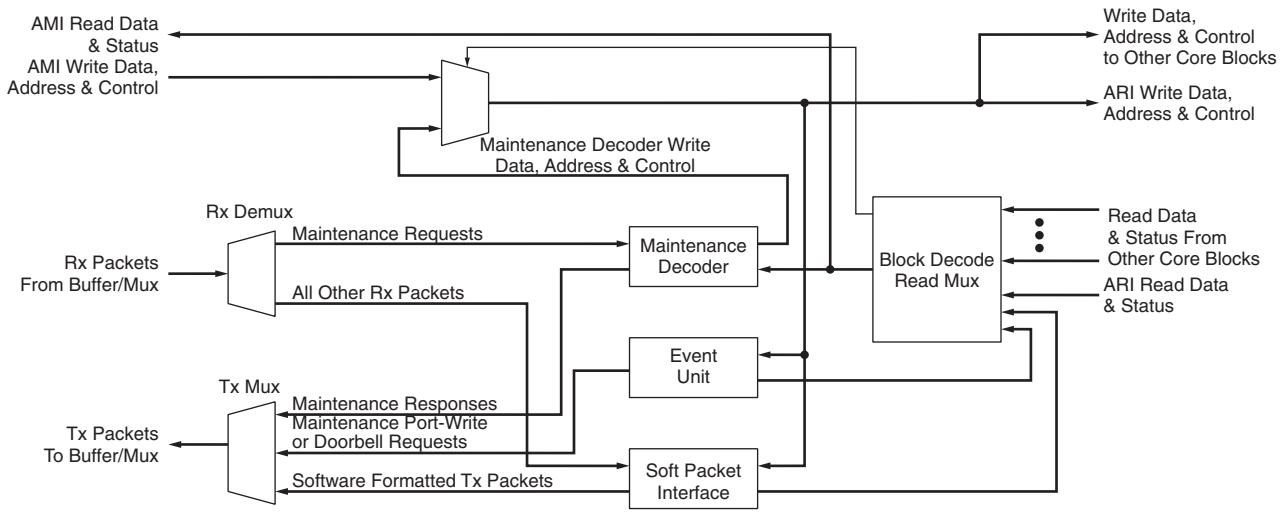


Management Module

The Management Module Implements the infrastructure needed to manage an endpoint remotely using RapidIO maintenance transactions or locally through the AMI processor interface. It also contains facilities for remote status reporting and software packet generation and decoding. The Management Module consists of the following sub-modules:

- **Maintenance Decoder** – Decoding and responding to maintenance transactions received from the RapidIO link.
- **Event Unit** – Implements logging and reporting of system events.
- **Block Decode / Read Mux** – Block level address decoding for modules. Arbitrating access to CSRs between the maintenance decoder and the Alternate Management Interface (AMI).
- **Soft Packet Interface (SPI)** – This module provides a means of generating and decoding RapidIO packets under software control.

Figure 2-9. Management Module Block Diagram



Maintenance Decoder

The maintenance decoder gives remote endpoints access to the control and status registers in the core. It does this by decoding incoming maintenance packets and generates read or write cycles on the management interface. In the case of maintenance read transactions the decoder formats response packets with the read data.

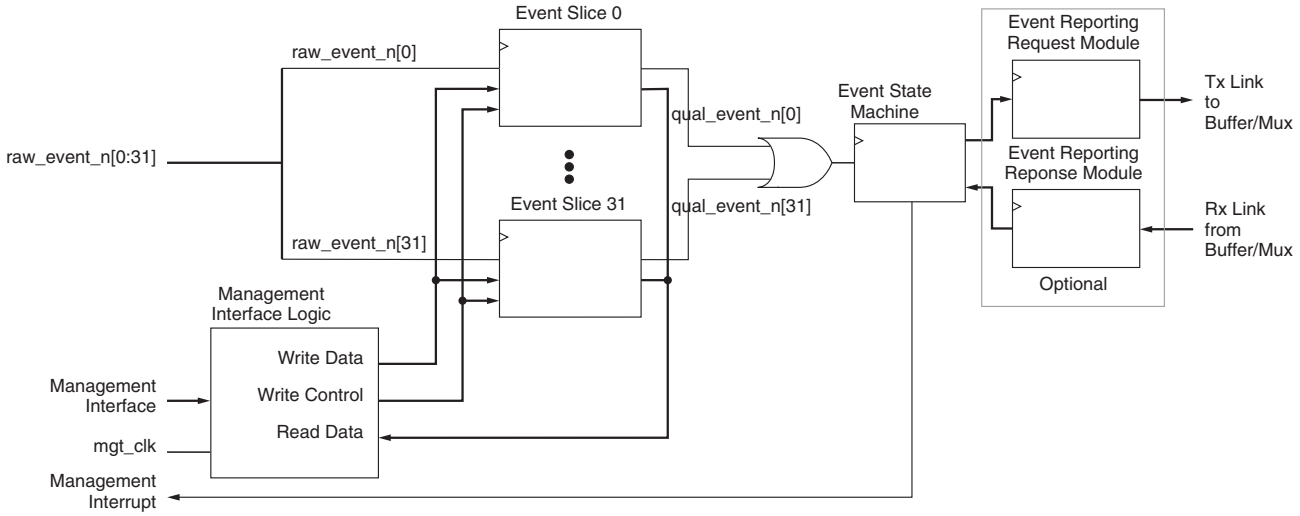
Event Unit

The Event unit logs and reports the various system level events that can occur within the endpoint. These events include:

- **Error Management Events** – These include errors at the physical, transport, and logical layer as defined in the Error Management Extensions specification.
- **Soft Packet Interface Events** – These include reception of a packet, as well as the completion of packet transmission.
- **User-Defined Events** – Each logical layer interface includes an event indication that can be used to flag events that are significant to the user application.

A complete list of events, both standard and optional, is shown in Table 2-2. Event logging consists of capturing the assertion of events on the event bus in the Event Status Register, and in some cases the Event Overflow Register. Depending on whether the event has been masked it will be reported to the system locally and remotely as described below. The structure of the Event Unit is shown in Figure 2-10.

Figure 2-10. Event Unit Block Diagram

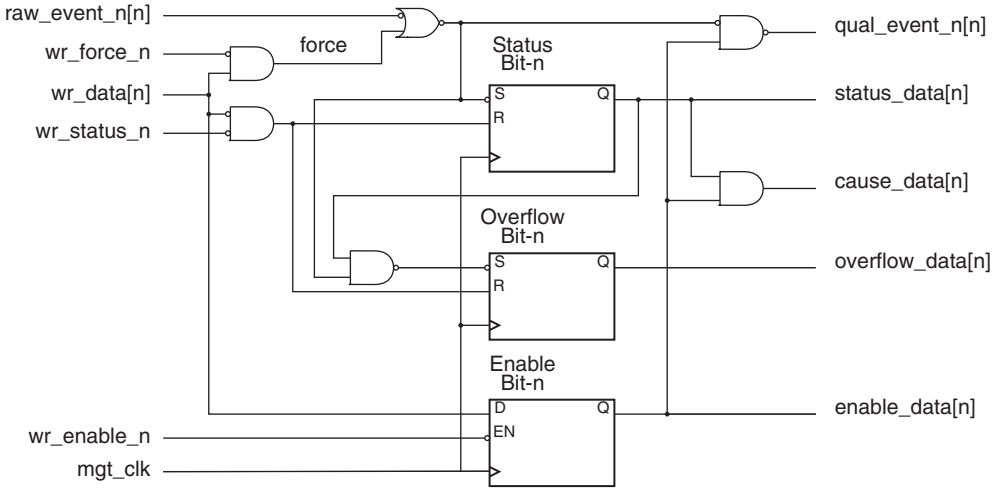


Event Logging

The logging of events is managed by five CSRs located in the event unit. Each bit in each of the CSRs corresponds to one of the system events.

- **Status Register** – The Status CSR shows which events have been asserted. This is a read/write register, and events can be cleared by writing to this register.
- **Overflow Register** – The Overflow CSR shows which events have occurred more than once since the corresponding bit was set in the status register.
- **Mask Register** – The Mask CSR controls whether the assertion of an event results in its being reported by one of the standard event reporting mechanisms.
- **Cause Register** – The Cause CSR shows which events caused an event to be reported. Essentially this CSR shows the state of the Status CSR masked by the state of the Mask CSR.
- **Force Register** – The Force CSR is used to force the assertion of one or more events, and is typically used for testing.

Figure 2-11. Event Slice Logic Block Diagram



Event Reporting

There are two methods of event reporting:

- Assertion of the interrupt signal on the Alternate Management Interface.
- Transmission of a Doorbell message. The doorbell message will contain the event number which corresponds to the active event listed in Table 2-2. The lower five bits of the information field is used to carry the event number. Care should be taken if using Doorbells in the system for other purposes to reserve this portion of the doorbell information field. In the initial version of the core, there is no way to disable generation of the doorbell message. This feature will be added in a future release of the core.

  Transmission of a Port-Write maintenance packet as described in the Error Management Extensions specification is also not available in the initial version of the core. This feature will also be made available as an option in a future release.

Table 2-2. Endpoint Layer Core Events

Event Number	Description
0	Physical Layer Error Management Event
1	Logical/Transport Layer Error Management Event
2	Soft Packet Interface Rx Event
3	Soft Packet Interface Tx Event
4-15	Reserved
16	Logical Port 0, User Event 0
17	Logical Port 0, User Event 1
18	Logical Port 0, User Event 2
19	Logical Port 0, User Event 3
20	Logical Port 1, User Event 0
21	Logical Port 1, User Event 1
22	Logical Port 1, User Event 2
23	Logical Port 1, User Event 3
24	Logical Port 2, User Event 0
25	Logical Port 2, User Event 1
26	Logical Port 2, User Event 2
27	Logical Port 2, User Event 3
28	Logical Port 3, User Event 0
29	Logical Port 3, User Event 1
30	Logical Port 3, User Event 2
31	Logical Port 3, User Event 3

Soft Packet Interface Module

The Soft Packet Interface (SPI) provides a means of sending and receiving RapidIO packets under software control. Formatting and decoding of these packets is performed by system software. The format of packet data transmitted and received is the same as that used on the Logical Layer port interfaces used to connected logical layer functions, and is shown in Figure 2-2 and Figure 2-3.

Specific examples of RapidIO Logical Layer packets can be seen in “Packet Formats and Descriptions” on page 25. Transmission and reception of packets by software is done using the CSRs summarized in Table 2-3. The Soft Packet Interface (SPI) supports packet transfers based on either polled or interrupt driven schemes.

Table 2-3. Soft Packet Interface CSR Summary

CSR Name	Summary Description	Details in Section
SP_TX_CTRL	Soft Packet Transmit Control	"Soft Packet FIFO Transmit Control (IR_SP_TX_CTRL) CSR" on page 75
SP_TX_STAT	Soft Packet Transmit Status	"Soft Packet FIFO Transmit Status (IR_SP_TX_STAT) CSR" on page 76
SP_TX_DATA	Soft Packet Transmit Data	"Soft Packet FIFO Receive Data (IR_SP_RX_DATA) CSR" on page 77
SP_RX_CTRL	Soft Packet Receive Control	"Soft Packet FIFO Receive Control (IR_SP_RX_CTRL) CSR" on page 76
SP_RX_STAT	Soft Packet Receive Status	"Soft Packet FIFO Receive Status (IR_SP_RX_STAT) CSR" on page 77
SP_RX_DATA	Soft Packet Receive Data	"Soft Packet FIFO Receive Data (IR_SP_RX_DATA) CSR" on page 77

Packet Transmission

Transmission of packets using the soft packet FIFO consists of the following steps:

1. Verify that the FIFO is ready to accept another packet by reading the Transmit Status CSR.
2. Write control information to Transmit Control CSR.
3. Write packet data to the Transmit Data CSR.

Loading of packet data in the transmit queue is managed by the Transmit FIFO State Machine. It is important to understand the operation of this state machine in order to write software that interacts with it. Figure shows the states and transitions that this machine implements and Table 2-17 describes the conditions that cause state transitions.

Table 2-4. Transmit FIFO State Machine State Diagram

Arc	Current State	Next State	Cause	Comments
1	Idle	Idle	The state machine is parked in this state until there is a write to the Transmit Control CSR.	This is the initial state after reset.
2	Idle	Armed	There has been a write to the Transmit Control CSR.	
3	Armed	Armed	The state machine is parked in this state until there is a write to the Transmit Data CSR.	The number of bytes in the packet to be transmitted is written to the
4	Armed	Active	There has been a write to the Transmit Data CSR.	
5	Active	Active	The state machine is parked in this state until the Octets Remaining field of the Transmit Status CSR equals zero.	Octets Remaining field of the Transmit Status CSR is decremented by four each time there is a write to the Transmit Data CSR.
6	Active	Idle	The Octets Remaining field of the Transmit Status CSR equals zero	If the Event Enable bit is set, a Soft Packet Interface Tx Event is generated during this transition.

Packet Reception

Reception of packets using the soft packet FIFO consists of the following steps:

1. Verify that the FIFO has packet data ready by reading the Receive Status CSR.
2. Read packet data from the Receive Data CSR.

Unloading packet data from the receive queue is managed by the Receive FIFO State Machine. It is important to understand the operation of this state machine in order to write software that interacts with it. Figure 2-12 shows the states and transitions that this machine implements and Table 2-5 describes the conditions that cause state transitions.

Figure 2-12. Receive FIFO State Machine State Diagram

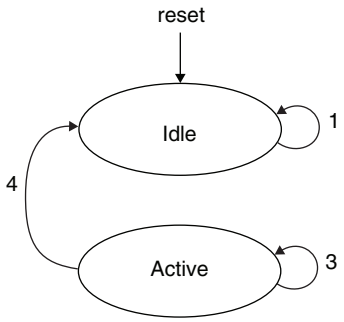


Table 2-5. Receive FIFO State Machine Transition Table

Arc	Current State	Next State	Cause	Comments
1	Idle	Idle	The state machine is parked in this state until there is a read of the Receive Data CSR.	This is the initial state after reset.
2	Idle	Active	There has been a read of the Receive Data CSR.	
3	Active	Active	The state machine is parked in this state until the Octets Remaining field of the Receive Status CSR equals zero.	Octets Remaining field of the Receive Status CSR is decremented by four each time there is a read of the Receive Data CSR.
4	Active	Idle	The Octets Remaining field of the Receive Status CSR equals zero	

Error Management

The core includes support for the RapidIO Error Management Extensions Specification. Error management support consists of the following components:

- A complete implementation of the physical layer error management extensions implemented in the OLLM and OPLM blocks.
- Implementation of logical and transport layer extensions for logical layer functions implemented in the core.
- Infrastructure for handling logical layer errors detected by user defined logical layer functions external to the core. Please refer to "Logical Port Error Capture Interface" on page 23 for detail. Errors detected by the Logical Layer are reported to the core using this 128-bit vector format.

Physical Layer Extensions

The core implements all of the physical layer error management extension described in the specification. These extensions are summarized in Table 2-6.

Table 2-6. Physical Layer Error Management Extension Summary

Error	Comments
Received control symbol	Received a control symbol with a bad CRC value.
Received out-of-sequence acknowledge control symbol	Received an acknowledge control symbol with an unexpected ackID (packet-accepted or packet_retry).
Received packet-not-accepted control symbol	
Received packet with unexpected ackID	Received packet with an ackID value that was either out-of-sequence or not outstanding.
Received packet with bad CRC	
Received packet exceeds 276 Bytes	Received packet which exceeds the maximum allowed size.

Table 2-6. Physical Layer Error Management Extension Summary (Continued)

Received Non-outstanding ackID	Link_response received with an ackID that is not outstanding.
Protocol error	An unexpected packet or control symbol was received.
Delineation error	Received unaligned /SC/ or /PD/ or undefined code-group.
Unsolicited acknowledge control symbol	An unexpected acknowledge control symbol was received.
Link time-out	An acknowledge or link-response control symbol is not received within the specified time-out interval.

The physical layer extensions are controlled by the following CSRs, which appear in the Error Reporting Block:

- Port 0 Error Detect CSR, described in section "Port 0 Error Detect (ERB_ERR_DET) CSR" on page 68
- Port 0 Error Rate Enable CSR, described in section "Port 0 Error Rate Enable (ERB_ERR_RATE_EN) CSR" on page 69
- Port 0 Attributes Capture CSR, described in section "Port 0 Attributes Capture (ERB_ATTR_CAPT) CSR" on page 70
- Port 0 Packet/Control Symbol Capture CSR, described in section "Port 0 Packet/Control Symbol Capture (ERB_PACK_SYM_CAPT) CSR" on page 71
- Port 0 Packet Capture CSR 1, described in section "Port 0 Packet Capture 1 (ERB_PACK_CAPT_1) CSR" on page 71
- Port 0 Port Packet Error Capture CSR 2, described in section "Port 0 Packet Capture 2 (ERB_PACK_CAPT_2) CSR" on page 71
- Port Packet Error Capture CSR 3, described in section "Port 0 Packet Capture 3 (ERB_PACK_CAPT_3) CSR" on page 71

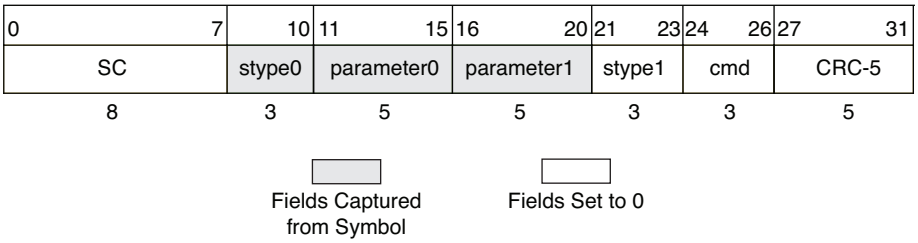
Physical Layer Error Decoding and Capture

RapidIO Physical layer errors are decoded by either receive or transmit logic depending on the error type. This has an effect on the data that is captured when the error occurs. In the case of errors decoded by the transmit logic only the fields relevant to the specific error are capture in the Port Packet/Control Symbol Error Capture CSR 0. Table 2-7 itemizes which errors are decode by the receive logic and which are decoded by the transmit logic.

Table 2-7. Physical Layer Error Capture

Error Name	Decoded By	What is captured
Corrupt Control Symbol	Rx Logic	Complete Control Symbol
Acknowledgement with Unexpected ackID	Tx Logic	Partial Control Symbol
Packet-not-accepted	Tx Logic	Partial Control Symbol
Packet with Unexpected ackID	Rx Logic	Complete Packet Header
Packet with Bad CRC	Rx Logic	Complete Packet Header
Packet greater than 276 bytes	Rx Logic	Complete Packet Header
Illegal or Invalid Character	**Not implemented**	**Not implemented**
Data Character in Idle 1 Sequence	**Not implemented**	**Not implemented**
Loss of Descrambler Synchronization	**Not implemented**	**Not implemented**
Link_response with non-outstanding ackID	Tx Logic	Partial Control Symbol
Protocol error	Tx Logic	Partial Control Symbol
Deliniation error	Rx Logic	Complete Control Symbol
Unsolicited acknowledgement	Tx Logic	Partial Control Symbol
Link time-out	Tx Logic	**Partial Packet Header**

Figure 2-13. Partial Control Symbol Capture Fields



User-Implemented Logical Transport Extensions

The core provides a common infrastructure for reporting logical layer errors detected by user defined logic connected to the core. Interaction between the core and these functions is done through the following interfaces:

- Logical Layer Common Control and Status Interface
- Logical Port Error Capture Interface

Logical Layer Common Control and Status Interface

The Logical Layer Common Control and Status Interface presents information contained in the logical layer control CSRs contained in the core. From the standpoint of error management there are two signals on this interface that are significant:

- **lt_error_en[0:31]** – This vector reflects the contents of the Logical/Transport Layer Error Enable CSR. User defined logical layer functions must monitor this vector to verify that they have detected has been enabled before reporting it.
- **resp_time_out[0:23]** – This vector reflects the state of bits 0-23 of the Port Response Time-out Control CSR. RapidIO logical layer functions that initiate transactions must use this value to set the timeout period for responses.

Logical Port Error Capture Interface

This interface is used to report errors detected by the user defined logical layer functions back to the core for reporting. When a logical layer error is detected that has not been masked by the corresponding bit in the lt_error_en vector, the user function should present the error information on the logN_error_capt_data vector and assert the logN_error_capt_trigd_req_n indication. This information must be held stable until acknowledged by the logN_error_capt_trigd_ack_n indication. The format of the logN_error_capt_data vector is shown in 10. Note that if another logical layer error was already reported and the Logical/Transport Capture were loaded with that error data and locked, this error data will be discarded.

Table 2-8. Error Capture Vector Field Definitions

Bit Field	Name	Description
logN_error_capt_data[0:31]	address[0:31]	Most significant 32 bits of the address associated with the error (for requests, for responses if available). This field is loaded into bits 0 to 31 of the Logical/Transport Layer High Address Capture CSR.
logN_error_capt_data[32:60]	address[32:60]	Least significant 29 bits of the address associated with the error (for requests, for responses if available). This field is loaded into bits 0 to 28 of the Logical/Transport Layer Address Capture CSR.
logN_error_capt_data[61]	reserved	Reserved field. Implementations should set this bit-field to 0. This field is loaded into bit 29 of the Logical/Transport Layer Address Capture CSR.

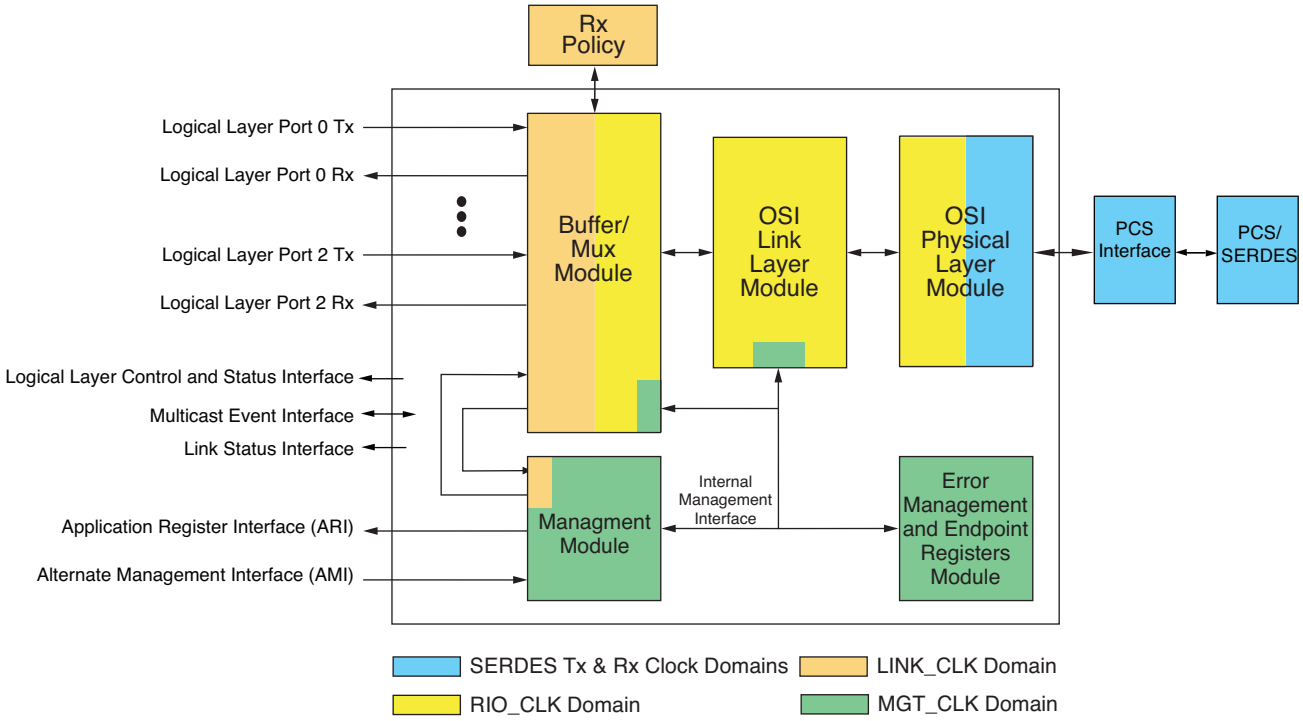
Table 2-8. Error Capture Vector Field Definitions (Continued)

logN_error_capt_data[62:63]	xamsbs	Extended address bits of the address associated with the error (for requests, for responses if available). This field is loaded into bits 30 to 31 of the Logical/Transport Layer Address Capture CSR.
logN_error_capt_data[64:71]	MSB destinationID	Most significant byte of the destinationID associated with the error (large transport systems only). This field is loaded into bits 0 to 7 of the Logical/Transport Layer Device ID Capture CSR.
logN_error_capt_data[72:79]	destinationID	The destinationID associated with the error. This field is loaded into bits 8 to 15 of the Logical/Transport Layer Device ID Capture CSR.
logN_error_capt_data[80:87]	MSB sourceID	Most significant byte of the sourceID associated with the error (large transport systems only). This field is loaded into bits 16 to 23 of the Logical/Transport Layer Device ID Capture CSR.
logN_error_capt_data[88:95]	sourceID	The sourceID associated with the error. This field is loaded into bits 24 to 31 of the Logical/Transport Layer Device ID Capture CSR.
logN_error_capt_data[96:99]	ftype	Format type associated with the error. This field is loaded into bits 0 to 3 of the Logical/Transport Layer Control Capture CSR.
logN_error_capt_data[100:103]	ttype	Transaction type associated with the error. This field is loaded into bits 4 to 7 of the Logical/Transport Layer Control Capture CSR.
logN_error_capt_data[104:111]	msg info	letter, mbox, and msgseg for the last Message request received for the mailbox that had an error (Message errors only). This field is loaded into bits 8 to 15 of the Logical/Transport Layer Control Capture CSR.
logN_error_capt_data[112:122]	implementation defined	Implementation defined field. This field is loaded into bits 16 to 26 of the Logical/Transport Layer Control Capture CSR.
logN_error_capt_data[123:127]	error type	This field indicates the type of error that has been detected. It is used to set the appropriate bit (indexed by this 5-bit vector) in the Logical/Transport Layer Error Detect CSR. It is also loaded into bits 27 to 31 of the Logical/Transport Layer Control Capture CSR. This 5-bit vector defines the bit number (0 to 31) to set in the Logical Layer Error Detect CSR.

Regional Clocking

The core contains five major clock domains as shown in Figure 2-14 (the SERDES clock domain contains both a receive and a transmit clock domain). Descriptions for each of these clocks are provided in "Interface Description and Timing Diagrams" on page 29.

Figure 2-14. LP-Serial Endpoint Core Clock Domains



Real-Time Time-Bases

The core generates two internal time-base signals used for timing link initialization and error management activities. One toggles every microsecond and the second toggles every millisecond. These time-bases are derived from the mgt_clk, and the user must configure an internal divider that divides the management clock to generate the microsecond time-base. The microsecond time-base is further divided down to create the millisecond time-base.

The mgt_clk divide ratio is configured through the gui. The required divide ratio can be calculated using the following formula:

divide_ratio = 1000 ns / mgt_clk period in ns = mgt_clk frequency in MHz

Packet Formats and Descriptions

This section describes the format of some common logical layer packet types as they appear on the Logical Layer Port transmit and receive interfaces. There are two things to note about these figures:

1. Only the format of small transport systems is shown (systems with less than or equal to 256 devices). Large transport systems will have all fields following the source and destination IDs shifted by 16-bits.

2. These figures do not show the final CRC/pad that will be present on received packets.

Maintenance Request/Response Formats

This section provides formats and field descriptions for some of the maintenance request/response transactions. Shaded rows in Table 2-9 indicate 8-byte D-word boundaries.

Table 2-9. Maintenance Read/Write Format Description

Bits	Value	Field Name	Rio Layer	Notes
0:1	b'00	**pri**ority	Phy	Priority: 00=lowest, 11= highest
2:3	b'00	tt	Transport	Device ID lengths 00=8b, 01=16b, 10, 11=unused (8 and 16b supported)
4:7	b'1000	**f**ormat **type**	Logical	Format Type 8=Maintenance
8:15	h'00	destID/Addr	Transport	Destination ID (endpoint ID)
16:23	h'00	SrcID/Addr	Transport	Source ID. Logical layer obtains this value from the Base Device ID CSR or via port array on the Control and Status interface.
24:27	b'0000	**trans**action	Logical	0=read, 1=write, 2=read response, 3=write response, 4=port write
28:31	b'1000	rdsize, wrsize or **Status** (resp)	Logical	Read[2]/Write[1] size in bytes (4 bytes shown here), DWords, or multiple DWords up to a 64 byte maximum. See also wdptr below and Tables 4.3 and 4.4 of Part 1 Spec for encodings. For responses, this is a status field (0=Done successfully, 7=Error, 0xc-0xf implementation defined).
32:39	h'00	**src**TID or **targetTID** (resp)	Logical	Source Transaction ID - Incremented for each transaction at the source and can be used for sequencing. For responses, the target uses the srcTID field received in the request to supply the targetTID field.
40:47	h'00	**hop**_Count	Transport	Destination resolution used only for maintenance packets routed though a switch network. Set to 0 for the core sims. Must be 0xff on responses per spec.
48:63	h'0000	**conf**ig**_off**set	Logical	Configuration register offset - 21b DWord address (least significant 3b of the 24b byte address are not sent and are encoded in wdptr and size fields)
0:4				
5	b'0	wdptr	Logical	Word Data Pointer, 0 = upper four byte lanes of a DWord, 1 = lower our byte lanes.
6:7	b'00	**res**erved	Logical	Reserved.
For Write Transactions				
8:63	H'00000000_000000	Data	Logical	Write data field. All data payloads are big-endian double-word aligned. Sub-double-word payloads must be padded and properly aligned within the 8-byte boundary.
0:7	H'00			

1. Writes greater than one DWord up to 64 bytes can be supported at all DWord lengths but must end on a DWord boundary.
2. Reads greater than one DWord up to 64 bytes can be supported at only the lengths listed in Table 4-3 – Part 1 specification (16, 32, and 64) and end on a DWord boundary.
3. Maintenance responses will occur for read/write formats following the same format as that described above (ftype=8, with associated transaction type (2=read response, 3=write response). Port-Write maintenance commands do not have a response.

Figure 2-15. Maintenance Read

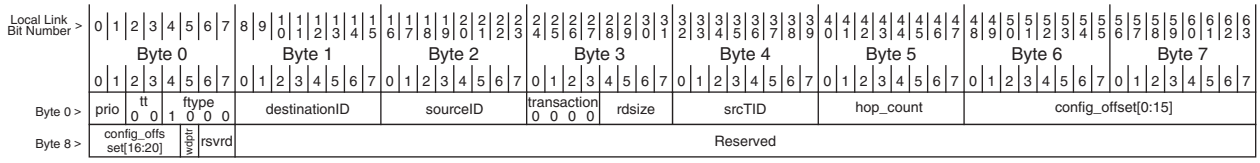


Figure 2-16. Maintenance Write

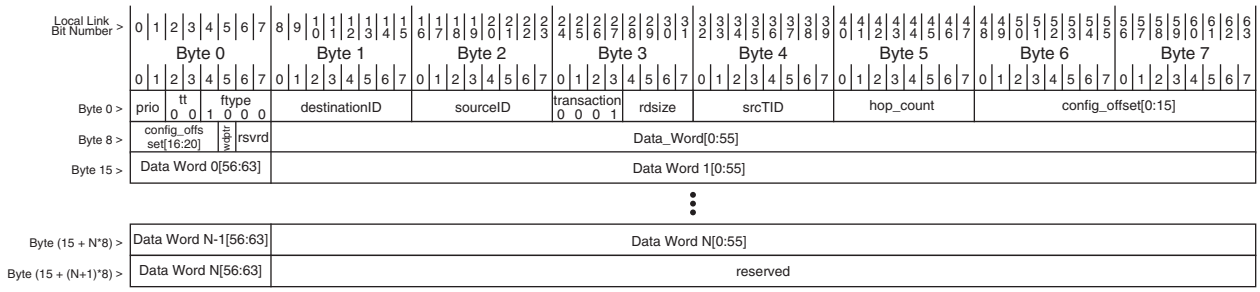


Figure 2-17. Maintenance Read Response

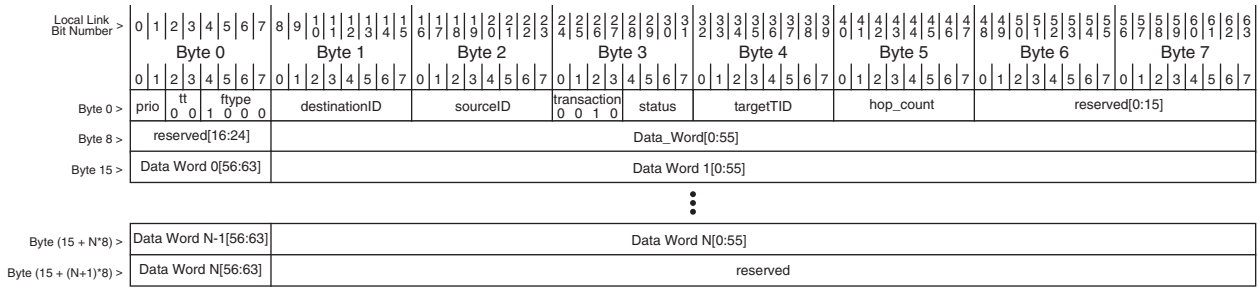
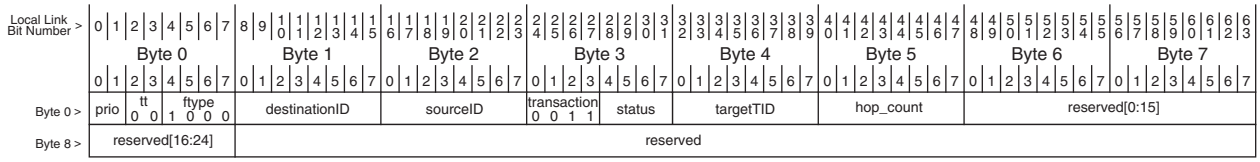


Figure 2-18. Maintenance Write Response



Read, Write and SWrite Request/Response Format Descriptions

This section provides formats and field descriptions for read, write, and swrite transactions. Shaded rows in Table 2-10 indicate 8-byte D-word boundaries.

Table 2-10. Read, Write and SWrite Request/Response Formats

Bits	Value	Field Name	Rio Layer	Notes
0:1	b'00	**pri**ority	Phy	Priority: 00=lowest, 11= highest
2:3	b'00	tt	Transport	Device ID lengths 00=8b, 01=16b, 10, 11=unused (8 and 16b supported)
4:7	b'1000	**f**ormat **type**	Logical	Format Type 2=NRead, 5=NWrite/NWrite_R, 6=SWrite, 13=Response Packet.
8:15	h'00	destID/Addr	Transport	Destination ID (endpoint ID)
16:23	h'00	SrcID/Addr	Transport	Source ID. Logical layer obtains this value from the Base Device ID CSR or via port array on the Control and Status interface.
24:27	b'0000	**trans**action	Logical	Transaction Type. When ftype=2, 4=NRead, others define Atomic. When ftype=5, 4=NWrite, 5=NWrite_R, others define Atomic. When ftype=6, there is no transaction type field in the packet. When ftype=13, 0=response with no payload, 8=response with payload.

Table 2-10. Read, Write and SWrite Request/Response Formats (Continued)

Bits	Value	Field Name	Rio Layer	Notes
28:31	b'1000	rdsize, wrsize or **Status** (response)	Logical	Read[2]/Write[1] size in bytes (4 bytes shown here), DWords, or multiple DWords up to a 256 byte maximum. See also wdptr below and Tables 4.3 and 4.4 of Part 1 Spec for encodings. For responses, this is a status field (0=Done successfully, 7=Error, 0xc-0xf implementation defined).
32:39	h'00	**src**TID or **targetTID** (response)	Logical	Source Transaction ID - Incremented for each transaction at the source and can be used for sequencing. For responses, the target uses the srcTID field received in the request to supply the targetTID field.
40:63	h'0000	address	Logical	Address - 29b DWord address (least significant 3b of the 32b byte address are not sent and are encoded in wdptr and size fields). The address length and whether the "extended address" is contained is specified as a global system parameter (Proc Element Logical Layer Control CSR). 32b addressing shown here.
0:4				
5	b'0	wdptr	Logical	Word Data Pointer, 0 = upper four byte lanes of a DWord, 1 = lower our byte lanes.
6:7	b'00	xamsbs	Logical	Extended address most significant two bits. Supports 34b addressing for the formats shown here.
For Write Transactions				
8:63	H'00000000_000000	Data	Logical	Write data field. All data payloads are big-endian double-word aligned. Sub-double-word payloads must be padded and properly aligned within the 8-byte boundary.
0:7	H'00			

1. Writes greater than one DWord up to 256 bytes can be supported at all DWord lengths but must end on a DWord boundary.
2. Reads greater than one DWord up to 256 bytes can be supported at only the lengths listed in Table 4-3 – Part 1 specification (16, 32, and 64, …) and end on a DWord boundary.
3. Non-maintenance requests use a separate specific ftype (0x13) for responses.

Figure 2-19. NRead

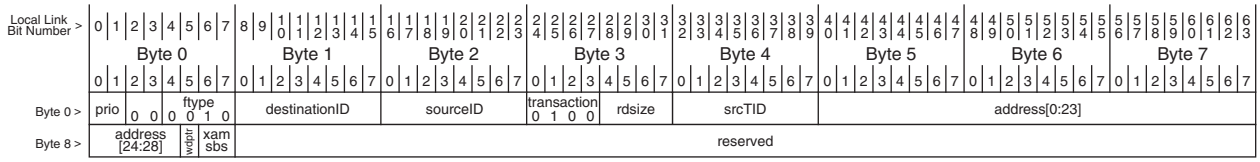


Figure 2-20. NWrite

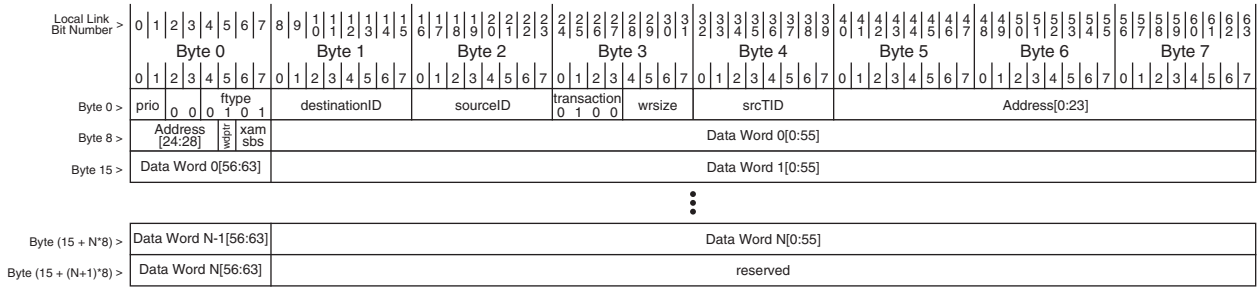


Figure 2-21. NWrite_R

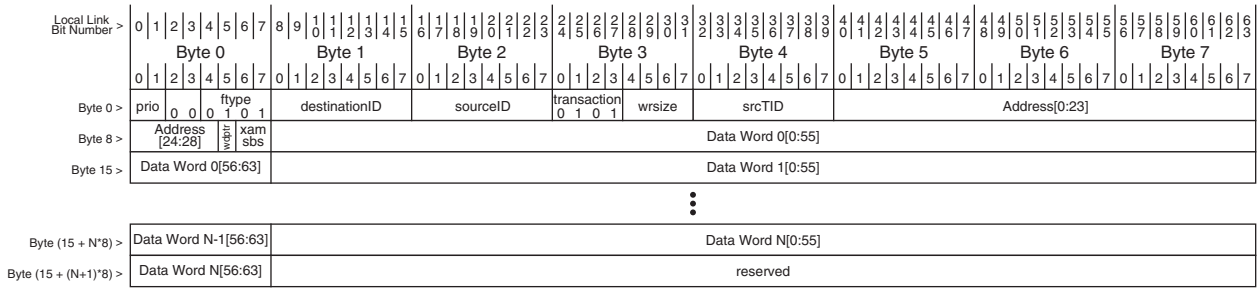


Figure 2-22. SWrite

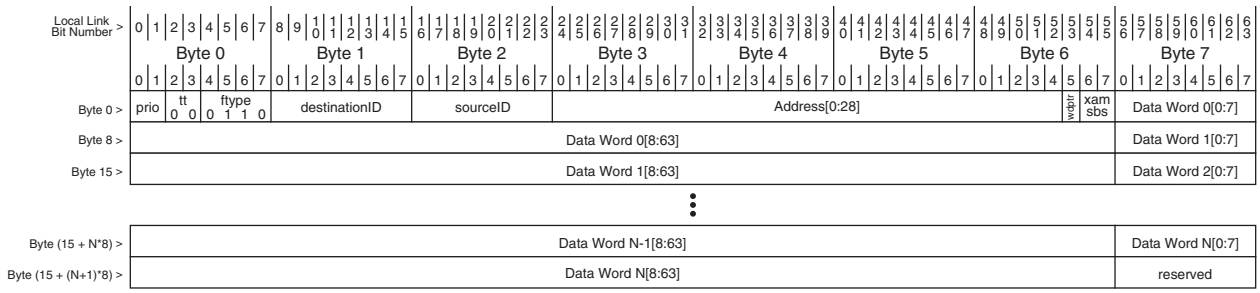


Figure 2-23. Response

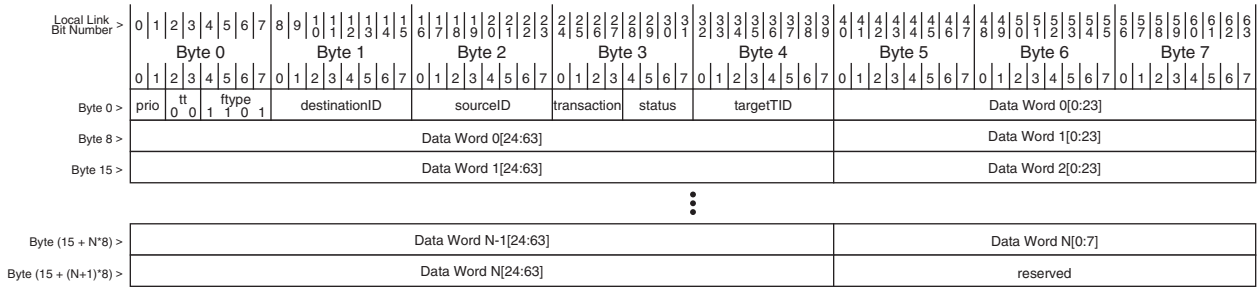
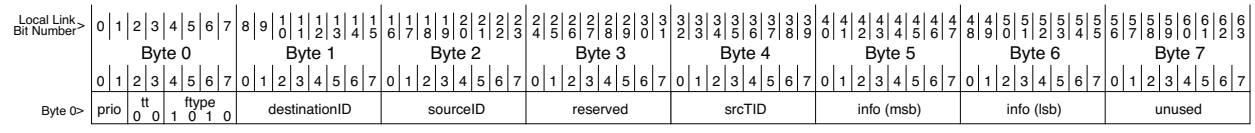


Figure 2-24. Doorbell



Interface Description and Timing Diagrams

This section describes the external interfaces of the core, including port names and functions. The core implements the following interfaces:

- Clocks, Resets, and Miscellaneous
- Logical Port Interface
- Logical Layer Common Control and Status Interface
- Alternate Management Interface (AMI)
- Application Register Interface (ARI)
- Multicast Event Interface

- Rx Policy Interface
- Link Status Interface
- Link Trace Interface
- Transceiver Interface

Clocks Resets and Miscellaneous

This section contains signal port descriptions for clocks, resets and other miscellaneous signals that are not associated with a specific interface. Also see "Regional Clocking" on page 25 for additional details.

The core has four reset inputs and one reset output as shown in Table 2-11. All of the reset inputs are asynchronous resets, however it is recommended that they be asserted asynchronously and de-asserted synchronously. This is illustrated in Figure 2-25.

Figure 2-25. Transmit FIFO State Machine State Diagram

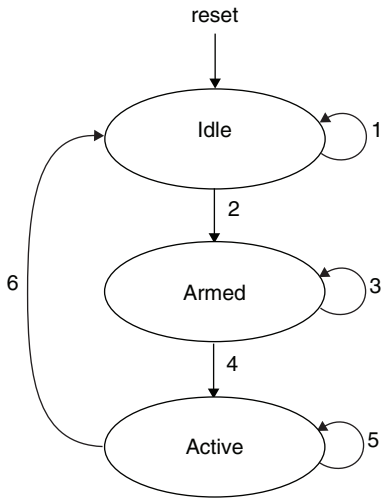


Table 2-11. Clocks, Resets and Miscellaneous Signal Descriptions

Signal	Direction	Clock	Description
Clocks and Resets			
recover_clk	Input	Up to 156.25MHz	Recover clock from PCS RX
recover_clk_reset_n	Input	Sync	Active low reset for the recover clock domain logic.
rx_clk (pcs_if_clk)	Input	Up to 156.25 MHz	Receive data clock to RX SRIO core.
rx_reset_n	Input	Sync	Active low reset for the transceiver receive interface logic.
tx_clk (pcs_if_clk)	Input	Same as rx_clk	Transmit SRIO data clock to transceiver (16-bit PCS interface).
tx_reset_n	Input	Sync	Active low reset for the transceiver transmit interface logic.
rio_clk	Input	Up to 156.25MHz	This clock is used to sequence the bulk of the receive logic in the RapidIO Physical layer interface. This clock is derived from a fraction of the SERDES reference clock, based on link width. The relationship between this clock and the transceiver clocks are managed to ensure synchronous transfer between them.
rio_clk_reset_n	Input	Sync	Active low reset for logic in the rio_clk domain.

Table 2-11. Clocks, Resets and Miscellaneous Signal Descriptions (Continued)

Signal	Direction	Clock	Description
lnk_clk	Input	Same as rio_clk	Transfers across the logical layer interfaces are made with respect to this clock. This clock must be synchronous with respect to rio_clk. In this implementation of the IP core, rio_clk = lnk_clk.
lnk_clk_reset_n	Input	Sync	Active low reset for logic in the lnk_clk domain.
mgt_clk	Input	User-provided rate	This is the clock for the management logic in the RapidIO core. This clock is asynchronous with respect to the other clocks. The rate can chosen by the user to match their control plane bus logic. Typical rates would be 50 MHz to 100 MHz. *Note: A management clock must be provided even if the AMI or ARI ports are not used, in order to respond to maintenance transactions received on the link.*
mgt_clk_reset_n	Input	Sync	Assertion of this active low input resets all of the storage elements in the Link-Request Reset-Device state machine. This signal is typically connected to the power-on reset or push-button reset in the system.
sys_reseti_n	Input	async	Active low input resetting all of the storage elements in the Link-Request Reset-Device state machine. This signal is typically connected to the power-on reset or push-button reset in the system.
sys_reseto_n	Output	rio_clk	This is an active low output from the Link-Request Reset-Device state machine. It is asserted under two conditions: · The sys_reseti_n input has been asserted. · A valid Link-Request Reset-Device sequence has been detected. When either of these conditions occurs, sys_reseto_n will be asserted for 64 mgt_clk cycles."

The following tables shows the relationship between pcs_if_clk and rio_clk with different combinations of baud rates and numbers of lanes.

Table 2-12. 1.25G Baud Rate

Clock	x1	x2	x4	Units
pcs_if_clk	62.5	62.5	62.5	MHz
rio_clk/lnk_clk	15.625	31.25	62.5	MHz

Table 2-13. 2.5G Baud Rate

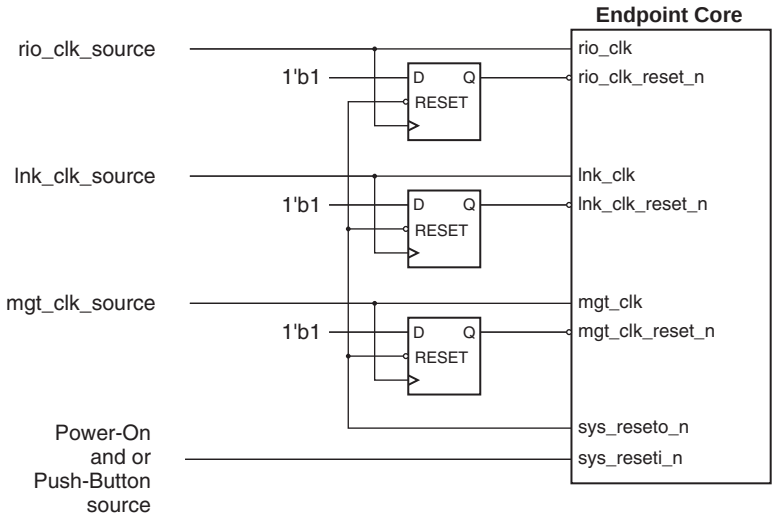
Clock	x1	x2	x4	Units
pcs_if_clk	125	125	125	MHz
rio_clk/lnk_clk	31.25	62.5	125	MHz

Table 2-14. 3.125G Baud Rate

Clock	x1	x2	x4	Units
pcs_if_clk	156.25	156.25	156.25	MHz
rio_clk/lnk_clk	39.0625	78.125	156.25	MHz

Figure 2-26 shows an example of a reset strategy for driving the various resets. The core also includes support for resetting itself and other logic upon reception of a Link-Request Reset-Device command sequence as defined in the RapidIO LP-Serial Physical Layer Specification. In order to take advantage of this functionality the reset signals should be connected as shown in Figure 2-26.

Figure 2-26. Example Reset Strategy



Logical Port Interface

There are three logical port interfaces available on the core. Each of these port interfaces support a 64-bit bi-directional data path, error capture, and user defined event functions. The error capture functions (supporting error management extension) and user the user defined event functions have been described earlier in the document. Inputs associated with unused ports should be tied inactive for control signals.

Table 2-15. Logical Port Interface Signal Descriptions

Signal	Direction	Clock	Description
Logical Port Transmit			
logN_tlnk_d[0:63]	Input	lnk_clk	Transmit data.
logN_tlnk_rem[0:2]	Input	lnk_clk	Transmit remainder. This value plus one, describes how many bytes are valid on the final beat of a transfer.
logN_tlnk_sof_n	Input	lnk_clk	Transmit start of frame indication. When asserted, this signal indicates that the current data beat is the first beat of a packet.
logN_tlnk_eof_n	Input	lnk_clk	Transmit end of frame indication. When asserted, this signal indicates that the current data beat is the last beat of a packet.
logN_tlnk_src_rdy_n	Input	lnk_clk	Transmit source ready indication. When asserted, this signal indicates that the user logic has presented valid data on the link.
logN_tlnk_src_dsc_n	Input	lnk_clk	Transmit source discontinue indication. When asserted any time during packet transmission, this signal indicates that the core should discard the current packet.
logN_tlnk_dst_rdy_n	Output	lnk_clk	Transmit destination ready indication. When asserted, this signal indicates that the core is ready to accept data during this clock cycle. This signal will be de-asserted by the core for the following reasons: The queuing unit is currently enqueing a packet from another logical port. There are already fifteen packets in the priority transmission queue at the priority level contained in the packet header. All packet buffers in the queuing unit are currently filled. Stalls are not asserted by the core once a packet transfer has started on that port.

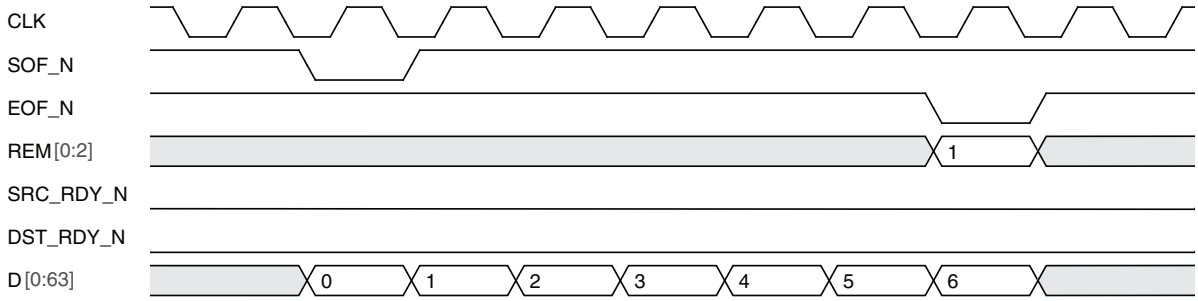
Table 2-15. Logical Port Interface Signal Descriptions (Continued)

Signal	Direction	Clock	Description
Logical Port Receive			
logN_rlnk_d[0:63]	Output	lnk_clk	Receive data.
logN_rlnk_rem[0:2]	Output	lnk_clk	Receive remainder. This value plus one, describes how many bytes are valid on the final beat of a transfer.
logN_rlnk_beats[0:7]	Output	lnk_clk	Transfer beat count. This signal is asserted at the beginning of the transfer and indicates the total number of data beats that make up the transfer.
logN_rlnk_sof_n	Output	lnk_clk	Receive start of frame indication. When asserted, this signal indicates that the current data beat is the first beat of a packet.
logN_rlnk_eof_n	Output	lnk_clk	Receive end of frame indication. When asserted, this signal indicates that the current data beat is the last beat of a packet.
logN_rlnk_src_rdy_n	Output	lnk_clk	Receive source ready indication. When asserted, this signal indicates that the core has presented valid data on the link.
logN_rlnk_dst_rdy_n	Input	lnk_clk	Receive destination ready indication. When asserted, this signal indicates that the user logic is ready to accept data during this clock cycle.
logN_rlnk_dst_dsc_n	Input	lnk_clk	Receive destination discontinue indication. When asserted any time during packet transmission, this signal indicates that the core should stop sending the current packet. The core will then resend the packet from the beginning. See "Errata" on page 8 for caveats on this control signal's usage.
Logical Port Error Capture			
logN_port_id[0:1]	Output	static	Logical port ID. This vector carries the logical port number of the logical port.
logN_error_capt_data[0:127]	Input	async	Error management data. Data to be loaded into the Logical/Transport Layer capture registers. See "Logical Port Error Capture Interface" on page 23 for a description of the 128-bit vector.
logN_error_capt_req_n	Input	async	Error capture request indication. When asserted This signal indicates that the logical layer function attached to the port have detected a logical/transport layer error and that valid capture data is present on logN_error_capt_data. The core assumes that this signal is asynchronous and synchronizes it internally.
logN_error_capt_ack_n	Output	async	Error capture request acknowledge indication. When asserted this signal indicates that the core has captured the error data presented on logN_error_capt_data.
Logical Port User Event			
logN_event_req_n[0:3]	Input	async	Event request indication. When asserted this signal indicates that the logical layer function attached to the port has detected a user defined event. The core assumes that this signal is asynchronous and synchronizes it internally.
logN_event_ack_n[0:3]	Output	async	Event request acknowledge indication. When asserted this signal indicates that the core has recognized the user event.

Logical Port Interface Data Path Timing Diagrams

A transfer consists of one or more data beats. A data beat is the time to transfer data presented on the interface from source to destination. When the flow control signals are not asserted this is one clock cycle. Figure 2-27 shows a basic PDU transfer across the interface, which is applicable to either receive or transmit directions, without the assertion of flow control. In this example, a total of fifty octets of data are transferred in seven data beats. Eight octets are transferred on each of the first six data beats and two octets during the final one. A key thing to note in that all byte lanes must be fully populated during all data beats except for the final one.

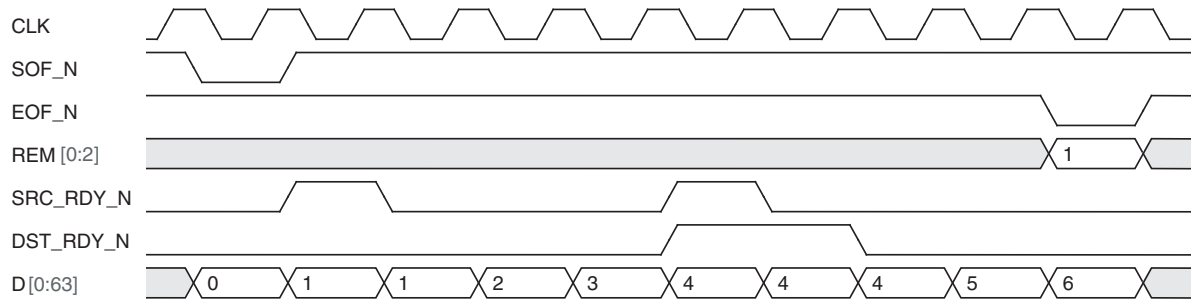
Figure 2-27. Logical Port Data Transfer Diagram



The flow control mechanism that is included as part of the logical port interface specification is a transfer stall scheme that is controlled with the DST_RDY_N and SRC_RDY_N signals. De-assertion of either of these signals stalls the progress of data beats. These signals may be independently de-asserted at any time. Figure 2-28 shows the same PDU transfer that was shown in Figure 2-27 with stalls due to the de-assertion of various combinations of the DST_RDY_N and SRC_RDY_N signals.

The first stall that occurs in the transfer is due to the de-assertion of SRC_RDY_N and results in the extension of data beat one to two clock cycles. The second stall is due to the de-assertion of both DST_RDY_N and SRC_RDY_N for one clock cycle followed by the de-assertion of DST_RDY_N only for one clock cycle. This second stall results in the extension of data beat four to three clock cycles.

Figure 2-28. Logical Port Data Transfer Diagram With Stalls



Logical Layer Common Control and Status Interface

This interface provides control and status information to the logical layer functions connected to the logical ports.

Table 2-16. Logical Layer Control and Status Interface Signal Descriptions

Signal	Direction	Clock	Description
tx_flow_ctrl_state[0:4]	Output	lnk_clk	This vector indicates the current state of the Tx flow-control state machine. The states are encoded as follows: **5`b00001** – Tx flow control is disabled either due to the state of the Tx Flow Control Enable bit in the Buffer Configuration CSR or it is not supported by the link partner. **5`b00010** – The free buffer count is greater than or equal to the WM0 field in the Buffer Configuration CSR and the Dequeue Unit is forwarding traffic at all priority levels. **5`b00100** – The free buffer count is greater than or equal to the WM1 value and less than the WM0 value in the Buffer Configuration CSR and the Dequeue Unit is forwarding traffic at priority levels 1, 2, and 3. **5`b01000** – The free buffer count is greater than or equal to the WM2 value and less than the WM1 value in the Buffer Configuration CSR and the Dequeue Unit is forwarding traffic at priority levels 2, and 3. **5`b10000** – The free buffer count is less than the WM2 value in the Buffer Configuration CSR and the Dequeue Unit is forwarding traffic at priority level 3 only.
tx_flow_depth[0:21]	Output	lnk_clk	This vector contains the current depths of the transmission queues. The vector is encoded as follows: tx_flow_depth[0:3] – Depth of priority 0 transmit queue tx_flow_depth[4:7] – Depth of priority 1 transmit queue tx_flow_depth[8:11] – Depth of priority 2 transmit queue tx_flow_depth[12:15] – Depth of priority 3 transmit queue tx_flow_depth[16:21] – Total depth of all queues
base_deviceid[0:7]	Output	mgt_clk	Small transport base deviceID. This port reflects the state of bits 8 to 15 of the Base Device ID CSR.
large_base_deviceid[0:15]	Output	mgt_clk	Large transport base deviceID. This port reflects the state of bits 16 to 31 of the Base Device ID CSR.
lt_error_en[0:31]	Output	mgt_clk	Logical transport error enable. Lt_error_en[0:31] reflects the state of the Logical/Transport Layer Error Enable CSR.
pe_ll_ctrl[0:31]	Output	mgt_clk	Logical layer control. pe_ll_ctrl[0:31] reflects the state of the Processing Element Logical Layer Control CSR.
resp_time_out[0:23]	Output	mgt_clk	Logical layer response time-out. resp_time_out[0:23] reflects the state of bits 0 to 23 of the Port Response Time-out Control CSR.
master_enable	Output	mgt_clk	Master enable indication. This port reflects the state of bit 1 the Port general Control CSR.
lcsh_bar[0:63]	Output	mgt_clk	Local configuration space base address. This vector presents the data in the Local Configuration Space Base Address 0 and 1 CSRs. lcsh_bar[0:31] reflects the state of bits 0 to 31 of Local Configuration Space Base Address 0. lcsh_bar[32:63] reflects the state of bits 0 to 31 of Local Configuration Space Base Address 1.
raw_event_n[0:31]	Output	mgt_clk	Raw event vector. Each active-low bit is asserted for one clock cycle when an event condition is decoded. The mapping of events to the bits in this vector is shown earlier in the document.

Alternate Management Interface (AMI)

The Alternate Management Interface is used by user defined management hardware to access CSRs within the core or the Application Register Block. Table 2-17 describes the signals that make up this interface. Figure 2-29 illustrates an AMI read cycle while Figure 2-30 shows a write cycle.

Table 2-17. Alternate Management Interface Signal Descriptions

Signal	Direction	Clock	Description
ami_mgt_di[0:31]	Input	mgt_clk	Management data in. Write data to the core is driven onto this interface.
ami_mgt_do[0:31]	Output	mgt_clk	Management data out. Data from read cycles is delivered from this port.
ami_mgt_a [0:21]	Input	mgt_clk	Management register address. This word address selects the register target for read and write cycles.
ami_mgt_sel_n	Input	mgt_clk	Management device select. This signal is asserted to start a read or write cycle.
ami_mgt_wr_n [0:3]	Input	mgt_clk	Management register write. There is one write signal for each byte of the write interface. The write signals are mapped to the write data word as follows: ami_mgt_wr [0] – Write signal for ami_mgt_di[0:7] ami_mgt_wr [1] – Write signal for ami_mgt_di[8:15] ami_mgt_wr [2] – Write signal for ami_mgt_di[16:23] ami_mgt_wr [3] – Write signal for ami_mgt_di[24 :31]
ami_mgt_rd_n	Input	mgt_clk	Management register read. This signal is asserted for read cycles.
ami_mgt_rdy_n	Output	mgt_clk	Management ready signal. This signal is asserted when write data has been accepted during write cycles, or valid read data is present on ami_mgt_do[0:31] during read cycles. The interface master should deassert ami_mgt_sel_n in the next cycle.
ami_mgt_int_n	Output	mgt_clk	Management interrupt signal. This signal is asserted each time there is an unmasked event.
ami_mgt_timeout_n	Output	mgt_clk	Management Interface Time Out. This signal is asserted coincidentally with the ami_mgt_rdy_n signal when there is a timeout on the management interface
ami_error_capt_data	Input	mgt_clk	AMI error management data. Data to be loaded into the Logical/Transport Layer capture registers. This port is user to capture user-defined error information.
ami_error_capt_trigd_req_n	Input	mgt_clk	Error capture request indication. When asserted, this signal indicates that the user defined AMI interface logic has detected an error and that valid capture data is present on ami_error_capt_data.
ami_error_capt_trigd_ack_n	Output	mgt_clk	Error capture request acknowledge indication. When asserted this signal indicates that the core has captured the error data presented on ami_error_capt_data.

Figure 2-29. AMI Read Cycle Example

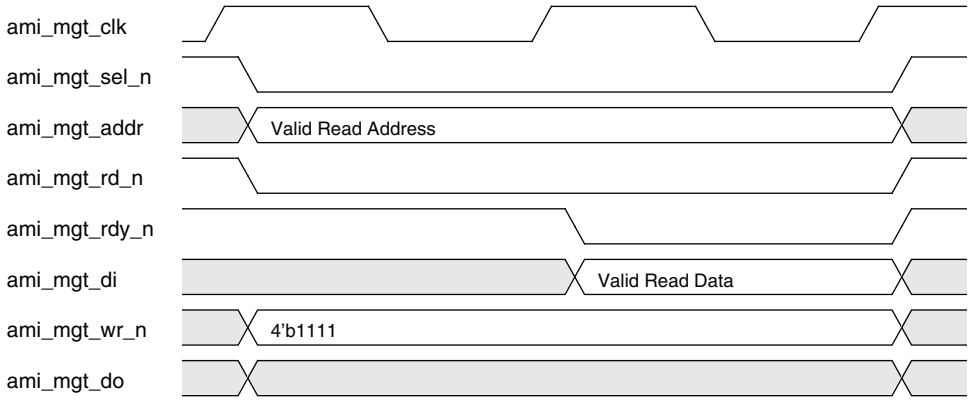
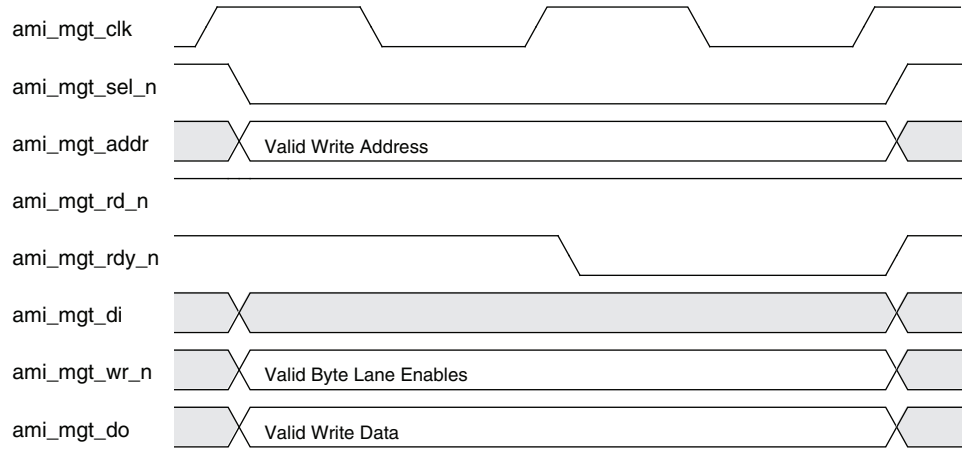


Figure 2-30. AMI Write Cycle Example



Application Register Interface (ARI)

The Application Register Interface connects the core to a customer defined block that implements application specific control and status registers. Registers implemented in this block are accessible though RapidIO maintenance transactions or through the Alternate Management Interface. Table 2-18 describes the signals that make up this interface. This interface uses the same timing and protocol as the Alternate Management Interface. Note that the logN port names used here have no correlation to the three Logical Layer ports of the SRIO core. For example, a write cycle to log0_mgt_di does not use the core's Logical Layer Port0 transmit path. The reason for these three logN ports is to provide built-in muxing for up to three separate user register logic blocks (perhaps corresponding to user-supplied Logical Layer blocks), but all data/control flow is still through the Management port.

Table 2-18. Application Register Interface Signal Descriptions

Signal	Direction	Clock	Description
ari_log0_mgt_di[0:31], ari_log1_mgt_di[0:31], ari_log2_mgt_di[0:31]	Input	mgt_clk	Management data in. During read cycles, data from one of the user register logic blocks is returned to the core via these ports.
ari_mgt_do[0:31]	Output	mgt_clk	Management data out. Write data from the core is driven onto this interface.
ari_mgt_a [0:21]	Output	mgt_clk	Management register address. This word address selects the register target for read and write cycles.
ari_mgt_sel_n	Output	mgt_clk	Management device select. This signal is asserted to start a read or write cycle.

Table 2-18. Application Register Interface Signal Descriptions (Continued)

Signal	Direction	Clock	Description
ari_mgt_wr_n [0:3]	Output	mgt_clk	Management register write. There is one write signal for each byte of the write interface. The write signals are mapped to the write data word as follows: ari_mgt_wr [0] – Write signal for ari_mgt_di[0:7] ari_mgt_wr [1] – Write signal for ari_mgt_di[8:15] ari_mgt_wr [2] – Write signal for ari_mgt_di[16:23] ari_mgt_wr [3] – Write signal for ari_mgt_di[24 :31]
ari_mgt_rd_n	Output	mgt_clk	Management register read. This signal is asserted for read cycles.
ari_log0_mgt_rdy_n, ari_log1_mgt_rdy_n, ari_log2_mgt_rdy_n,	Input	mgt_clk	Management ready signal. This signal is asserted when write data has been accepted during write cycles, or valid read data is present on ari_log[n]_mgt_do[0:31] during read cycles. The interface master should de-assert ari_log[n]_mgt_sel_n in the next cycle.
ari_log0_mgt_int_n, ari_log1_mgt_int_n, ari_log2_mgt_int_n,	Input	mgt_clk	Management interrupt signal. This signal is asserted when there is an event requiring attention.

Figure 2-31. ARI Read Cycle Example

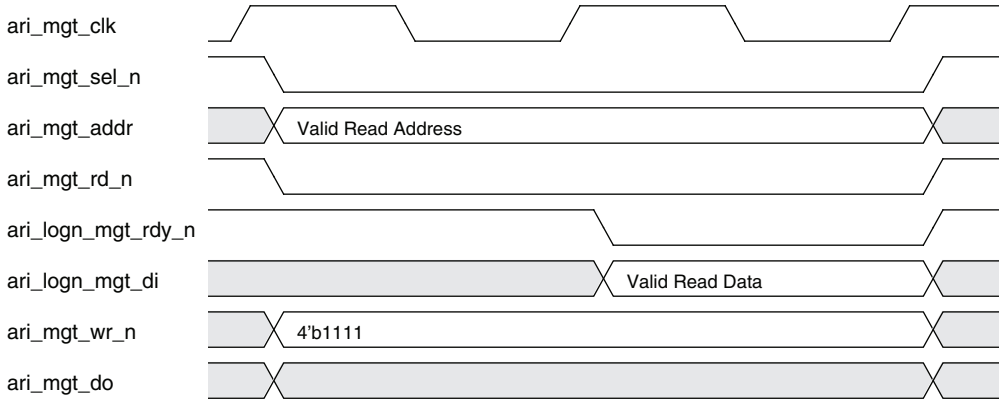
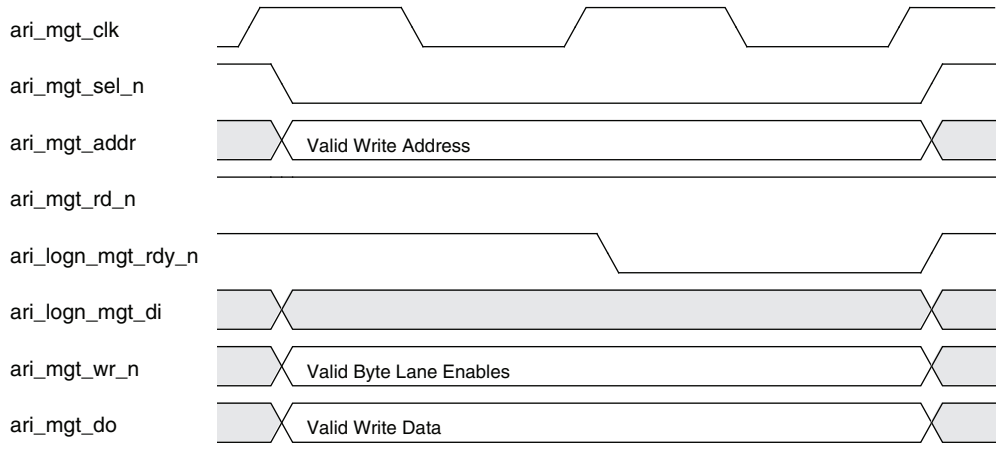


Figure 2-32. ARI Write Cycle Example



Multicast Event Interface

The core provides support for generating RapidIO control symbols containing the multicast event function code. In addition it indicates when a control symbol has been received that contains the multicast event function code.

Table 2-19. Multicast Event Interface Signal Descriptions

Signal	Direction	Clock	Description
lnk_mce_rx_req_n	Output	async	Multicast event reception request. This signal is asserted when a multicast event control symbol has been received. It remains asserted until lnk_mce_rx_ack_n is asserted by user logic.
lnk_mce_rx_ack_n	Input	async	Multicast event reception acknowledgement. User logic asserts this signal to acknowledge the assertion of the lnk_mce_rx_ack_n signal. It must remain asserted until lnk_mce_rx_ack_n is deasserted.
lnk_mce_tx_req_n	Input	async	Multicast event transmit request. This signal is asserted when user logic wants to transmit a multicast event control. It must remain asserted until lnk_mce_tx_ack_n is asserted by the core.
lnk_mce_tx_ack_n	Output	async	Multicast event transmit acknowledgement. The core asserts this signal to acknowledge the assertion of the lnk_mce_tx_req_n signal. It will remain asserted until lnk_mce_tx_req_n is deasserted.

Receive Policy Interface

The Receive Policy Interface is used to allow a user defined policy to control the acceptance and de-multiplexing of incoming packet traffic to the Logical Layer Ports on the core as described in "Packet Reception" on page 16.

Table 2-20. Rx Policy Interface Signal Descriptions

Signal	Direction	Clock	Description
rxp_data[0:63]	Output	rio_clk	Rx packet header data. This vector contains the received packet data stream.
rxp_sof_n	Output	rio_clk	Rx packet header start of packet indication. This signal indicates that the data on rxp_data[0:63] is the first beat of the packet.
rxp_eof_n	Output	rio_clk	Rx packet header end of packet indication. This signal indicates that the data on rxp_data[0:63] is the last beat of the packet.
rxp_advance_n	Output	rio_clk	Rx packet header advance indication. When this signal is asserted, rxp_data[0:63] is valid.
rxp_discard_n	Output	rio_clk	Rx packet header discard indication. This signal is asserted at the same time that rxp_eof_n is asserted if the packet is to be discarded. This signal is asserted for the following reasons: - A CRC error was detected on the packet. - The port is in the input-retry or input-error state.
rxp_rdy_n	Input	rio_clk	Rx packet policy ready indication. This signal is asserted by the user-defined policy when it has presented a valid destination logical port number
rxp_demux_port[0:3]	Input	rio_clk	Rx packet destination port number. This vector contains the logical port number that the user-defined rx policy wants the packet to be forwarded to.
rxp_force_retry_n	Input	rio_clk	Rx packet retry indication. The user-defined demux policy asserts this signal when the packet is to be retried. *Note: This is an advanced feature. It is not required to force a packet retry when the receive queues are full. Flow control is done automatically by the core.*

The current implementation supports a single clock cycle of header inspection before the user-defined receive policy must assert rxp_rdy_n. This means that the policy can inspect the first 64-bits of the packet before deciding whether to force a retry of the packet or which local port to forward it to if it is accepted. This is illustrated in

Figure 2-33, which shows the Receiver policy mapping a 6 data beat packet which has been accepted by the OLLM module.

Figure 2-33. Receive Policy Interface Timing – Accepted Packet / Normal Operation

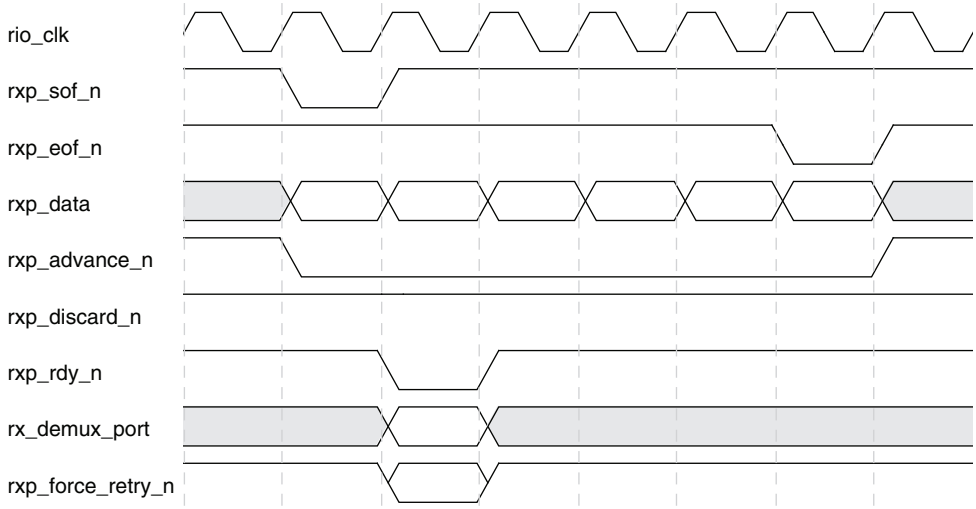
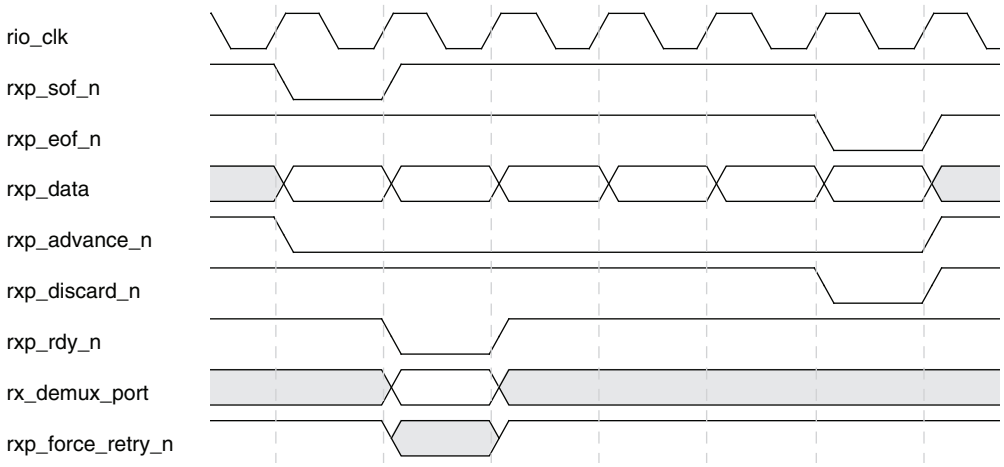


Figure 2-34 shows the timing of a packet that is eventually marked for discard by the OLLM module. This packet is discarded and retried independent of the state of the rxp_force_retry signal when rxp_rdy_n is asserted.

Figure 2-34. Receive Policy Interface Timing – Discarded Packet



Link Status Interface

The status interface includes both generic information about Tx and Rx activity (status[8:11]) and RapidIO port state (status[0:7]) as well as interface training information specific to LP-Serial ports. Table 2-21 shows the signals that make up this interface.

Table 2-21. RapidIO LP-Serial Status Port Signals

Signal	Direction	Clock	Description
status[0:7]	Output	mgt_clk	The bits of this vector correspond to the state of several bits in the Port n Error and Status CSR. The mapping of these register bits to the status vector is shown in Table 2-22.
status[8]	Output	phy_tclk	This active low signal is asserted for one clock cycle whenever a packet has passed the Logical Layer Port transmit interface.
status[9]	Output	phy_tclk	This active low signal is asserted simultaneously with status[8] for one clock cycle if the packet being transmitted has been discarded due to congestion or an error.
status[10]	Output	phy_rclk	This active low signal is asserted for one clock cycle whenever a packet has passed the Logical Layer Port receive interface.
status[11]	Output	phy_rclk	This active low signal is asserted simultaneously with status[10] for one clock cycle if the packet being received has been discarded due to congestion or an error.
status[12:21]	Output	phy_rclk	These bits contain the state of the 1x/2x/Nx Initialization State Machine described in Section 4.12.4.8.1 of the RapidIO Interconnect Specification Part 6: Physical Layer LP-Serial Specification version 2.1. The mapping of states to the values encoded on this vector is shown in Table 2-23.

Table 2-22. Port n Error and Status CSR to Status Port Signal Mapping

Status Vector Bit Number	Port n Error and Status CSR Bit Number	Port n Error and Status CSR Bit Description
0	11	Output Retry-encountered
1	12	Output Retried
2	13	Output Retry-stopped
3	14	Output Error-encountered
4	15	Output Error-stopped
5	21	Input Retry-stopped
6	22	Input Error-encountered
7	23	Input Error-stopped

Bits 12 to 21 of the status interface bring out the state of the 1x/2x/Nx Initialization State Machine described in Section 4.12.4.8.1 of the RapidIO Interconnect Specification Part 6: Physical Layer LP-Serial Specification version 2.1. Table 2-23 shows the enumeration of state machine states on these status bits.

Table 2-23. LP-Serial Initialization State Machine State Enumeration

Status[12:21] Value	State
10'b0000000001	SILENT
10'b0000000010	SEEK
10'b0000000100	DISCOVERY
10'b0000001000	1X_MODE_LANE0
10'b0000010000	1X_MODE_LANE1
10'b0000100000	1X_MODE_LANE2
10'b0001000000	1X_RECOVERY
10'b0010000000	2X_MODE
10'b0100000000	2X_RECOVERY
10'b1000000000	NX_MODE

Link Trace Interface

The Link Trace Interface is a debugging interface that makes the transmit and receive data streams between the OLLM and the OPLM accessible for protocol trace purposes. Reveal can be used to capture these outputs and then write out the captured data to a text file. The text file can then be post-processed by software to provide a log file revealing high-level SRIO transactions requests and symbol exchanges over the link.

Table 2-24. Link Trace Interface Signal Description

Signal	Signal Direction	Clock Domain	Description
phy_tlnk_rdy_n	Output	rio_clk	Transmit link ready indication. This is essentially a link up indication.
phy_trdy_n	Output	rio_clk	Transmit advance indication. This is stall indication for the Tx link.
phy_td[0:63]	Output	rio_clk	Transmit data. This vector contains multiplexed packet data and control symbol information. This data is valid when both phy_tlnk_rdy_n and phy_trdy_n are asserted.
phy_tvalid_h_n	Output	rio_clk	Transmit valid on high word indication. When asserted, this signal indicates that either packet or control symbol data are being presented on phy_td[0:31].
phy_tsym_h_n	Output	rio_clk	Transmit symbol on high word indication. When asserted, this signal indicates that control symbol data is being presented on phy_td[0:31].
phy_tvalid_l_n	Output	rio_clk	Transmit valid on low word indication. When asserted, this signal indicates that either packet or control symbol data are being presented on phy_td[32:63].
phy_tsym_l_n	Output	rio_clk	Transmit symbol on low word indication. When asserted, this signal indicates that control symbol data is being presented on phy_td[32:63].
phy_rlnk_rdy_n	Output	rio_clk	Receive link ready indication. This is essentially a link up indication.
phy_rrdy_n	Output	rio_clk	Receive advance indication. This is stall indication for the Rx link.
phy_rd[0:63]	Output	rio_clk	Receive data. This vector contains multiplexed packet data and control symbol information. This data is valid when both phy_rlnk_rdy_n and phy_rrdy_n are asserted.
phy_rvalid_h_n	Output	rio_clk	Receive valid on high word indication. When asserted, this signal indicates that either packet or control symbol data are being presented on phy_rd[0:31].
phy_rsym_h_n	Output	rio_clk	Receive symbol on high word indication. When asserted, this signal indicates that control symbol data is being presented on phy_rd[0:31].
phy_rvalid_l_n	Output	rio_clk	Receive valid on low word indication. When asserted, this signal indicates that either packet or control symbol data are being presented on phy_rd[32:63].
phy_rsym_l_n	Output	rio_clk	Receive symbol on low word indication. When asserted, this signal indicates that control symbol data is being presented on phy_rd[32:63].

Transceiver Interface

The transceiver interface connects the SRIO IP core to the PCS interface logic that interfaces with the built-in SERDES.

Table 2-25. Transceiver Interface Signal Descriptions

Signal	Signal Direction	Clock Domain	Description
pd_stat[0:31]	Input	mgt_clk	The bits of this vector are readable in the Platform Dependent PHY Status CSR. These bits are used to monitor platform specific transceiver status. They are customer configured, and are strapped to 32'd0 in the distribution version of the transceiver glue.
pd_ctrl[0:31]	Output	mgt_clk	The bits of this vector are readable in the Platform Dependent PHY Control CSR. These bits are used to control platform specific transceiver features. They are customer configured, and are not connected in the distribution version of the transceiver glue.

Table 2-25. Transceiver Interface Signal Descriptions (Continued)

Signal	Signal Direction	Clock Domain	Description
baud_support[0:4]	Input	tx_clk	Indicates which baud rates are supported by the transceivers. 5`b00001 - 1.25 GBaud 5`b00010 - 2.5 GBaud 5`b00100 - 3.125 GBaud 5`b01000 - 5.0 GBaud 5`b10000 - 6.25 GBaud
baud_select[0:3]	Output	tx_clk	Indicates the selected baud rate of the port 4`b0000 - Reserved 4`b0001 - 1.25 GBaud 4`b0010 - 2.5 GBaud 4`b0011 - 3.125 GBaud 4`b0100 - 5.0 GBaud 4`b0101 - 6.25 GBaud 4`b0110 - 4`b1111 - Reserved
port_disable	Output	tx_clk	This signal reflects the state of the bit-8 in the Port 0 Control CSR. It is typically used to enable a platform specific powerdown mode. The powerdown mode used is configured in the transceiver glue, and is disabled by default.
loopback[0:2]	Output	mgt_clk	Loopback mode control vector. This signal is controlled by bits [2:4] of the Platform Independent PHY Control CSR. The encoding of this vector is as follows: 3'b000No Loopback 3'b001Near-End PCS Loopback 3'b010 Near-End PMA Loopback 3'b011Far-End PMA Loopback 3'b100Far-End PCS Loopback
rx_init_n	Output	rx_clk	Active low, software controlled, reset signal for transceiver Rx logic. This signal is controlled by bit-1 of the Platform Independent PHY Control CSR.
rx_data[0:63 or 31]	Input	rx_clk	Receive data from transceivers.
rx_charisk[0:7 or 3]	Input	rx_clk	Indicates which octets in rx_data are data or K characters.
rx_8b10b_err[0:7 or 3]	Input	rx_clk	Indicates that an 8b10b decoding error was detected on the corresponding rx_data byte.
rx_lane_type[0:7]	Input	rx_clk	Indicates the lane type: 2`b00 - short run 2`b01 - medium run 2`b10 - long run 2`b11 – Reserved The bits on this interface are assigned as follows: rx_lane_type[0:1] – Lane 0 rx_lane_type[2:3] – Lane 1 rx_lane_type[4:5] – Lane 2 rx_lane_type[6:7] – Lane 3
rx_lane_trained[0:3]	Input	rx_clk	Indicates the state of the Rx equalization logic. The state of this signal appears in the Receiver trained bit in the Lane n Status 0 CSRs, and the “Receiver trained” bit in the CS Field transmitted by the lane.
rx_lane_sync[0:3]	Input	rx_clk	This active high signal indicates that synchronization has been achieved on each of the lanes.
rx_lane_inv[0:3]	Input	rx_clk	This active high signal indicates that the transceiver has detected that the polarity of the input signal has been inverted and has inverted the polarity of the receiver input to correct this.
tx_init_n	Output	mgt_clk	Active low, software controlled, reset signal for transceiver Tx logic. This signal is controlled by bit-0 of the Platform Independent PHY Control CSR.

Table 2-25. Transceiver Interface Signal Descriptions (Continued)

Signal	Signal Direction	Clock Domain	Description
tx_data[0:63 or 31]	Output	tx_clk	Transmit data to transceivers.
tx_charisk[0:7 or 3]	Output	tx_clk	Indicates which octets in txdata are data or K characters.
tx_lane_type[0:3]	Input	tx_clk	Indicates the lane type: 1`b0 - short run 1`b1 - long run
tx_lane_mode[0:3]	Input	tx_clk	Indicates the current operating mode of the lane: 1`b0 - short run 1`b1 - long run
tx_lane_silence[0:3]	Output	tx_clk	Each bit of this vector, when asserted, forces the corresponding transmitter to a marking state.
rcv_baud_lt_xmt_baud	input	tx_clk	This active high signal indicates that the current line side received baud rate is less that the current baud rate being used for transmission. This signal is used for baud rate discovery when enabled.
tport_enable_n	input	rio_clk	When asserted this signal forces the core to drive out test data to the transceivers. The test data consists of character data from the tport_tdi[0:63] input, and K character select data from the tport_char_is_k[0:7]. Clock compensation sequences are not multiplexed into the transmit data stream when the test port is enabled, and the core must be reset after tport_enable_n is de-asserted.
tport_tdi[0:63]	input	rio_clk	Test data to be driven onto tx_data port.
tport_char_is_k[0:7]	input	rio_clk	Indicates that the corresponding octets in tport_tdi are K characters when driven to 1`b1.

RapidIO 2.1 LP-Serial Core Registers

This section describes the Capability Registers (CARs) and Command and Status Registers (CSRs) that are implemented in the core. These registers can be accessed through maintenance transactions entering from the serial RapidIO interface, or through the Alternate Master Interface. Both interfaces provide the same level of read/write access. One does not provide more or less access rights over the other.

Reserved registers are registers that contain only reserved bits. Reserved bits are intended for control and or status functions in future implementations. Software should not assume a value for reserved bits when read. Software should write zero to reserved bits when writing to registers that contain reserved bits.

Register Blocks

The core provides a single register window of 16 Mbytes. Registers are 32 or 64-bit, and they are word or double-word addressable. These registers fall into the following broad categories:

- RapidIO Logical and Common Transport Layer Registers
- RapidIO Physical Layer Registers
- RapidIO Error Management Extensions Registers
- Implementation Registers – These are registers not defined in the RapidIO specifications, but defined in this specification, and implemented in the core.
- Application Registers – These are registers not defined in this specification or the RapidIO specifications, and implemented in Application Register Block.

Table 2-26. Core Register Block Mapping

Address Range	Description
0x000000 to 0x000088	Logical and Common Transport Layer Registers
0x00008C to 0x0000FC	Reserved
0x000100 to 0x00024C	Physical Layer LP-Serial Registers
0x000250 to 0x000FFC	Reserved
0x001000 to 0x00126C	Error Management Extensions Registers
0x001270 to 0x00FFFC	Reserved
0x010000 to 0x1FFFFC	Implementation Registers
0x200000 to 0xFFFFFC	Application Registers

Address Map

Table 2-27. RapidIO 2.x LP-Serial Core Register Address Map

Byte Address Offset	Register	Description	Section
Logical and Common Transport Layer Registers			
0x000000	DEV_ID	Device Identity CAR	
0x000004	DEV_INFO	Device Information CAR	
0x000008	ASMBLY_ID	Assembly Identity CAR	
0x00000C	ASMBLY_INFO	Assembly Information CAR	
0x000010	PE_FEAT	Processing Element Features CAR	
0x000014	Reserved		
0x000018	SRC_OPS	Source Operations CAR	
0x00001C	DST_OPS	Destination Operations CAR	
0x000020 to 0x000048	Reserved		
0x00004C	PE_LLC	Processing Element Logical Layer Control CSR	
0x000050 to 0x000054	Reserved		
0x000058	LCS_BA0	Local Configuration Space Base Address 0	
0x00005C	LCS_BA1	Local Configuration Space Base Address 1	
0x000060	B_DEV_ID	Base Device ID CSR	
0x000064	Reserved		
0x000068	HB_DEV_ID_LOCK	Host Base Device ID Lock CSR	
0x00006C	COMP_TAG	Component Tag CSR	
0x000070 to 0x0000FC	Reserved		
Physical Layer LP-Serial Registers			
0x000100	LP_SERIAL_HDR	LP-Serial Register Block Header	
0x000104 to 0x00011C	Reserved		
0x000120	PORT_LT_CTRL	Port Link Time-out Control CSR	
0x000124	PORT_RT_CTRL	Port Response Time-out Control CSR	
0x000128 to 0x000138	Reserved		
0x00013C	PORT_GEN_CTRL	Port General Control CSR	
0x000140 to 0x000154	Reserved		
0x000154	PORT_0_CTRL2	Port 0 Error and Status CSR	
0x000158	PORT_0_ERR_STAT	Port 0 Error and Status CSR	
0x00015C	PORT_0_CTRL	Port 0 Control CSR	

Table 2-27. RapidIO 2.x LP-Serial Core Register Address Map (Continued)

Byte Address Offset	Register	Description	Section
0x000160 to 0x000FFC	Reserved		
0x001000	LP_SERIAL_LANE_HDR	LP-Serial Lane Register Block Header	
0x001004 to 0x00100C	Reserved		
0x001010	LANE_0_STATUS_0	Lane 0 Status 0 CSR	
0x001014 to 0x00102C	Reserved		
0x001030	LANE_1_STATUS_0	Lane 1 Status 0 CSR	
0x001034 to 0x00104C	Reserved		
0x001050	LANE_2_STATUS_0	Lane 2 Status 0 CSR	
0x001054 to 0x00106C	Reserved		
0x001070	LANE_3_STATUS_0	Lane 3 Status 0 CSR	
0x001074 to 0x001FFC	Reserved		
Error Management Extensions Registers			
0x002000	ERB_HDR	Error Management Extensions Block Header	
0x002004	Reserved		
0x002008	ERB_LT_ERR_DET	Logical/Transport Layer Error Detect	
0x00200C	ERB_LT_ERR_EN	Logical/Transport Layer Error Enable	
0x002010	ERB_LT_ADDR_CAPT_H	Logical/Transport Layer High Address Capture	
0x002014	ERB_LT_ADDR_CAPT	Logical/Transport Layer Address Capture	
0x002018	ERB_LT_DEV_ID_CAPT	Logical/Transport Layer Device ID Capture	
0x00201C	ERB_LT_CTRL_CAP	Logical/Transport Layer Control Capture	
0x002020 to 0x002024	Reserved		
0x002028	ERB_LT_DEV_ID	Port-write Target Device ID	
0x00202C to 0x00203C	Reserved		
0x002040	ERB_P_ERR_DET	Port Error Detect CSR	
0x002044	ERB_P_ERR_RATE_EN	Port Error Rate Enable CSR	
0x002048	ERB_P_ATTR_ERR_CAPT	Port Attributes Error Capture CSR	
0x00204C	ERB_P_PACK_SYM_ERR_CAPT_0	Port Packet/Control Symbol Error Capture CSR 0	
0x002050	ERB_P_PACK_ERR_CAPT_1	Port Packet Error Capture CSR 1	
0x002054	ERB_P_PACK_ERR_CAPT_2	Port Packet Error Capture CSR 2	
0x002058	ERB_P_PACK_ERR_CAPT_3	Port Packet Error Capture CSR 3	
0x00205C to 0x002064	Reserved		
0x002068	ERB_P_ERR_RATE	Port Error Rate CSR	
0x00206C	ERB_P_ERR_RATE_THRESH	Port Error Rate Threshold CSR	
0x002070 to 0x101FFC	Reserved		
Implementation Registers			
0x102000 to 0x1060FC	Reserved		
0x102000	IR_BUFFER_CONFIG	Buffer Configuration	
0x102004 to 0x1060FC			
0x106100	IR_EVENT_STAT	Error Event Status	
0x106104	IR_EVENT_ENBL	Error Event Enable	
0x106108	IR_EVENT_FORCE	Error Event Force	
0x10610C	IR_EVENT_CAUSE	Error Event Cause	
0x106110	IR_EVENT_OVRFLOW	Error Event Overflow	

Table 2-27. RapidIO 2.x LP-Serial Core Register Address Map (Continued)

Byte Address Offset	Register	Description	Section
0x106114 to 0x106FFC	Reserved		
0x107000	IR_SP_TX_CTRL	Soft Packet FIFO Transmit Control	
0x107004	IR_SP_TX_STAT	Soft Packet FIFO Transmit Status	
0x107008	IR_SP_TX_DATA	Soft Packet FIFO Transmit Data	
0x10700C	IR_SP_RX_CTRL	Soft Packet FIFO Receive Control	
0x107010	IR_SP_RX_STAT	Soft Packet FIFO Receive Status	
0x107014	IR_SP_RX_DATA	Soft Packet FIFO Receive Data	
0x107018 to 0x10701C	Reserved		
0x107020	IR_PI_PHY_CTRL	Platform Independent PHY Control	
0x107024	IR_PI_PHY_STAT	Platform Independent PHY Status	
0x107028	IR_PD_PHY_CTRL	Platform Dependant PHY Control	
0x10702C	IR_PD_PHY_STAT	Platform Dependent PHY Status	
0x107030 to 0x1FFFFC	Reserved		
Application Registers			
0x200000 to 0xFFFFFC	Reserved for Application		

Device Identity (DEV_ID) CAR

Address: 0x000000

Description: The DeviceVendorIdentity field identifies the vendor that manufactured the device containing the processing element. A value for the DeviceVendorIdentity field is uniquely assigned to a device vendor by the registration authority of the RapidIO Trade Association. The core presents the value requested by the customer in the core license agreement.

The DeviceIdentity field is intended to uniquely identify the type of device from the vendor specified by the DeviceVendorIdentity field. The contents of this register reflect the state of GUI selections made during IP core generation.

Reference: This register implements the functionality described in Section 5.4.1 of the Input/Output Logical Specification Rev. 2.1.

Table 2-28. Device Identity CAR Bit-Fields

Bit Field	Name	Type	Reset State	Description
00:15	DeviceIdentity	Read Only	GUI Selection	Device identifier
16:31	DeviceVendorIdentity	Read Only	GUI Selection	Device vendor identifier

Device Information (DEV_INFO) CAR

Address: 0x000004

Description: The DeviceRev field is intended to identify the revision level of the device. The contents of this register reflect the state of GUI selections made during IP core generation.

Reference: This register implements the functionality described in Section 5.4.2 of the Input/Output Logical Specification Rev. 2.1.

Table 2-29. Device Information CAR Bit-Fields

Bit Field	Name	Type	Reset State	Description
00:31	DeviceRev	Read Only	GUI Selection	Device revision level

Assembly Identity (ASMBLY_ID) CAR

Address: 0x000008

Description: The AssyVendorIdentity field identifies the vendor that manufactured the assembly or subsystem containing the device. A value for the AssyVendorIdentity field is uniquely assigned to a device vendor by the registration authority of the RapidIO Trade Association. The core presents the value requested by the customer in the core license agreement.

The AssyIdentity field is intended to uniquely identify the type of device from the vendor specified by the AssyVendorIdentity field. The contents of this register reflect the state of GUI selections made during IP core generation.

Reference: This register implements the functionality described in Section 5.4.3 of the Input/Output Logical Specification Rev. 2.1.

Table 2-30. Assembly Identity CAR Bit-Fields

Bit Field	Name	Type	Reset State	Description
00:15	AssyIdentity	Read Only	GUI Selection	Assembly identifier
16:31	AssyVendorIdentity	Read Only	GUI Selection	Assembly vendor identifier

Assembly Information (ASMBLY_INFO) CAR

Address: 0x00000C

Description: This register includes assembly revision level information, and a pointer to the first entry in the extended features list. In this implementation this is a pointer to physical layer register block. The AssyRev field is intended to uniquely identify the revision of the assembly or subsystem containing the device. The contents of the AssyRev field reflect the state of GUI selections made during IP core generation.

Reference: This register implements the functionality described in Section 5.4.4 of the Input/Output Logical Specification Rev. 2.1.

Table 2-31. Assembly Information CAR Bit-Fields

Bit Field	Name	Type	Reset State	Description
00:15	AssyRev	Read Only	GUI Selection	Assembly revision level
16:31	ExtendedFeaturesPtr	Read Only	Set to 16'h0100	Pointer to the first entry in the extended features list

Processing Element Features (PE_FEAT) CAR

Address: 0x000010

Description: This register identifies the major functionality provided by the processing element. The contents of this register reflect the state of GUI selections made during IP core generation. The user must configure these selections to reflect the physical layer, transport layer, and logical layer functionality implemented in the design.

Reference: This register implements the functionality described in the following documents:

1. Section 5.4.5 of the Input/Output Logical Specification Rev. 2.1.
2. Section 6.4.1 of the LP-Serial Physical Layer Specification Rev. 2.1
3. Section 3.4.1 of the Common Transport Specification Rev 2.1

Table 2-32. Processing Element Features CAR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0	Bridge	Read Only		Indicates whether the PE can bridge to another interface such as PCI.
1	Memory	Read Only	GUI Selection	Indicates whether the PE contains memory.
2	Processor	Read Only	GUI Selection	Indicates whether the PE contains a local processor.
3	Switch	Read Only	GUI Selection	Indicates whether the PE can bridge to another external RapidIO interface.
4	Multi-port	Read Only	GUI Selection	The bit shall be implemented by devices that support the LP-Serial IDLE2 sequence, but is optional for devices that do not support the LP-Serial IDLE2 sequence. If this bit is not implemented it is Reserved. If this bit is implemented, the Switch Port Information CAR at Configuration Space Offset 0x14 (see *RapidIO Part 1: I/O Logical Specification*) shall be implemented regardless of the state of bit 3 of the Processing Element Features CAR. Indicates whether the PE implements multiple external RapidIO ports 1`b0 -PE does not implement multiple external RapidIO ports 1`b1 -PE implements multiple external RapidIO ports
5:25	Reserved	Read Only	GUI Selection	
26	CRF Support	Read Only	1`b0	PE supports the Critical Request Flow (CRF) indicator 1`b0 - Critical Request Flow is not supported 1`b1 - Critical Request Flow is supported
27	Common transport large system support	Read Only	GUI Selection	1`b0 - PE does not support common transport large systems 1`b1 - PE supports common transport large systems
28	Extended features	Read Only	1`b1	Indicates whether the PE has extended features list; the extended features pointer is valid.
29:31	Extended addressing support	Read Only	GUI Selection	Indicates the number address bits supported by the PE both as a source and target of an operation. All PEs shall at a minimum support 34 bit addresses. 3`b111 - PE supports 66, 50, and 34 bit addresses 3`b101 - PE supports 66 and 34 bit addresses 3`b011 - PE supports 50 and 34 bit addresses 3`b001 - PE supports 34 bit addresses All other encodings reserved

Source Operations (SRC_OPS) CAR

Address: 0x000018

Description: This register defines the set of RapidIO I/O logical operations that can be issued by this processing element. It is assumed that a processing element can generate I/O logical maintenance read and write requests if it is required to access CARs and CSRs in other processing elements. The contents of this register reflect the state of GUI selections made during IP core generation. The user must configure these selections to reflect the logical layer functionality implemented in the design.

Reference: This register implements the functionality described in the following documents:

1. Section 5.4.7 of the Input/Output Logical Specification Rev. 2.1.
2. Section 5.4.1 of the Message Passing Logical Specification Rev. 2.1.

Table 2-33. Source Operations CAR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:13	Reserved	Read Only	GUI Selection	
14:15	Implementation Defined	Read Only	GUI Selection	These bits are implemented as reserved bits.
16	Read	Read Only	GUI Selection	PE can support a read operation
17	Write	Read Only	GUI Selection	PE can support a write operation
18	Streaming-write	Read Only	GUI Selection	PE can support a streaming-write operation
19	Write-with-response	Read Only	GUI Selection	PE can support a write-with-response operation
20	Data message	Read Only	GUI Selection	PE can support a data message operation
21	Doorbell	Read Only	GUI Selection	PE can support a doorbell operation
22	Atomic (compare-and-swap)	Read Only	GUI Selection	PE can support an atomic compare-and-swap operation
23	Atomic (test-and-swap)	Read Only	GUI Selection	PE can support an atomic test-and-swap operation
24	Atomic (increment)	Read Only	GUI Selection	PE can support an atomic increment operation
25	Atomic (decrement)	Read Only	GUI Selection	PE can support an atomic decrement operation
26	Atomic (set)	Read Only	GUI Selection	PE can support an atomic set operation
27	Atomic (clear)	Read Only	GUI Selection	PE can support an atomic clear operation
28	Atomic (swap)	Read Only	GUI Selection	PE can support an atomic swap operation
29	Port-write	Read Only	GUI Selection	PE can support a port-write operation
30:31	Implementation Defined	Read Only	GUI Selection	These bits are implemented as reserved bits.

Destination Operations (DST_OPS) CAR

Address: 0x00001C

Description: This register defines the set of logical layer operations that can be supported by the logical layer functions connected to the core. The contents of this register reflect the state of GUI selections made during IP core generation. The user must configure these selections to reflect the logical layer functionality implemented in the design.

Reference: This register implements the functionality described in the following documents:

1. Section 5.4.8 of the Input/Output Logical Specification Rev. 2.1.
2. Section 5.4.2 of the Message Passing Logical Specification Rev. 2.1.

Table 2-34. Destination Operations CAR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:13	Reserved	Read Only	GUI Selection	
14:15	Implementation Defined	Read Only	GUI Selection	These bits are implemented as reserved bits.
16	Read	Read Only	GUI Selection	PE can support a read operation
17	Write	Read Only	GUI Selection	PE can support a write operation
18	Streaming-write	Read Only	GUI Selection	PE can support a streaming-write operation
19	Write-with-response	Read Only	GUI Selection	PE can support a write-with-response operation
20	Data message	Read Only	GUI Selection	PE can support a data message operation
21	Doorbell	Read Only	GUI Selection	PE can support a doorbell operation
22	Atomic (compare-and-swap)	Read Only	GUI Selection	PE can support an atomic compare-and-swap operation
23	Atomic (test-and-swap)	Read Only	GUI Selection	PE can support an atomic test-and-swap operation
24	Atomic (increment)	Read Only	GUI Selection	PE can support an atomic increment operation
25	Atomic (decrement)	Read Only	GUI Selection	PE can support an atomic decrement operation
26	Atomic (set)	Read Only	GUI Selection	PE can support an atomic set operation
27	Atomic (clear)	Read Only	GUI Selection	PE can support an atomic clear operation
28	Atomic (swap)	Read Only	GUI Selection	PE can support an atomic swap operation
29	Port-write	Read Only	GUI Selection	PE can support a port-write operation
30:31	Implementation Defined	Read Only	GUI Selection	These bits are implemented as reserved bits.

Processing Element Logical Layer Control (PE_LLC) CSR

Address: 0x00004C

Description: The Processing Element Logical Layer Control CSR is used for general command and status information for the logical interface. The contents of bits 29 to 31 are visible on bits 29 to 31 of the pe_ll_ctrl[0:31] vector

Reference: This register implements the functionality described in Section 5.5.1 of the Input/Output Logical Specification Rev. 2.1.

Table 2-35. Processing Element Logical Layer Control CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:28	Reserved	Read Only	GUI Selection	
29:31	Extended addressing control	Read Write	GUI Selection	Controls the number of address bits generated by the PE as a source and processed by the PE as the target of an operation. 3`b100 - PE supports 66 bit addresses 3`b010 - PE supports 50 bit addresses 3`b001 - PE supports 34 bit addresses (default) All other encodings reserved

Local Configuration Space Base Address 0 (LCS_BA0) CSR

Address: 0x000058

Description: The local configuration space base address 0 register specifies the most significant bits of the local physical address double-word offset for the processing element's configuration register space. The contents of bits 0 to 31 are visible as bits 0 to 31 on the lcsh_bar[0:63] vector.

Reference: This register implements the functionality described in Section 5.5.2 of the Input/Output Logical Specification Rev. 2.1.

Table 2-36. Processing Element Logical Layer Control CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0	Reserved	Read Write	GUI Selection	Although this bit is writeable, it always reads as 1`b0 for compatibility with the RapidIO specification.
1:16	LCSBA	Read Write	GUI Selection	Reserved for a 34-bit local physical address Reserved for a 50-bit local physical address Bits 0-15 of a 66-bit local physical address
17:31	LCSBA	Read Write	GUI Selection	Reserved for a 34-bit local physical address Bits 0-14 of a 50-bit local physical address Bits 16-30 of a 66-bit local physical address

Local Configuration Space Base Address 1 (LCS_BA1) CSR

Address: 0x00005C

Description: The local configuration space base address 1 register specifies the least significant bits of the local physical address double-word offset for the processing element's configuration register space, allowing the configuration register space to be physically mapped in the processing element. This register allows configuration and maintenance of a processing element through regular read and write operations rather than maintenance operations. The double-word offset is right-justified in the register. The contents of bits 0 to 31 are visible as bits 32 to 63 on the lcsh_bar[0:63] vector.

Reference: This register implements the functionality described in Section 5.5.3 of the Input/Output Logical Specification Rev. 2.1.

Table 2-37. Processing Element Logical Layer Control CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0	LCSBA	Read Write	GUI Selection	Reserved for a 34-bit local physical address Bit 15 of a 50-bit local physical address Bit 31 of a 66-bit local physical address
1:31	LCSBA	Read Write	GUI Selection	Bits 0-30 of a 34-bit local physical address Bits 16-46 of a 50-bit local physical address Bits 32-62 of a 66-bit local physical address

Base Device ID (B_DEV_ID) CSR

Address: 0x000060

Description: The base device ID CSR contains the base device ID values for the processing element. A device may have multiple device ID values, but these are not defined in a standard CSR.

Reference: This register implements the functionality described in Section 3.5.1 of the Common Transport Specification Rev. 2.1.

Table 2-38. Host Base Device ID Lock CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:7	Reserved	Read Only	Set to 8'd0.	
8:15	Base_deviceID	Read Write	GUI Selection	This is the base ID of the device in a small common transport system (end point devices only).
16:31	Large_base_deviceID	Read Write	GUI Selection	This is the base ID of the device in a large common transport system (only valid for end point device and if bit 27 of the Processing Element Features CAR is set).

Host Base Device ID Lock (HB_DEV_ID_LOCK) CSR

Address: 0x000068

Description: The host base device ID lock CSR contains the base device ID value for the processing element in the system that is responsible for initializing this processing element. The Host_base_deviceID field is a write-once/reset-able field which provides a lock function. Once the Host_base_deviceID field is written, all subsequent writes to the field are ignored, except in the case that the value written matches the value contained in the field. In this case, the register is re-initialized to 0xFFFF. After writing the Host_base_deviceID field a processing element must then read the Host Base Device ID Lock CSR to verify that it owns the lock before attempting to initialize this processing element.

Reference: This register implements the functionality described in Section 3.5.2 of the Common Transport Specification Rev. 2.1.

Table 2-39. Host Base Device ID Lock CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:15	Reserved	Read Only	Set to 16'h0000.	
16:31	Host_base_deviceID	Read Write	Set to 16'hFFFF.	This is the base device ID for the PE that is initializing this PE.

Component Tag (COMP_TAG) CSR

Address: 0x00006C

Description: The component tag CSR contains a component tag value for the processing element and can be assigned by software when the device is initialized. It is especially useful for labeling and identifying devices that are not end points and do not have device ID registers.

Reference: This register implements the functionality described in Section 3.5.3 of the Common Transport Specification Rev. 2.1.

Table 2-40. Component Tag CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	component_tag	Read Write	Set to 32'h00000000.	Component tag for the PE.

LP-Serial Register Block Header (LP_SERIAL_HDR)

Address: 0x000100

Description: The LP-Serial register block header contains the EF_PTR to the next extended features block and the EF_ID that identifies this as the generic end point LP-Serial register block header. In this implementation this is a pointer to the LP-Serial lane register block.

Reference: This register implements the functionality described in Section 6.6.1 of the LP-Serial Physical Layer Specification Rev. 2.1.

Table 2-41. LP-Serial Block Header Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:15	EF_PTR	Read Only	Set to 16'h1000	Pointer to the next block in the extended features data structure.
16:31	EF_ID	Read Only	Set to 16'h0001.	Extended Features ID

Port Link Time-out Control (PORT_LT_CTRL) CSR

Address: 0x000120

Description: The port link time-out control register contains the time-out timer value for all ports on a device. This time-out is for link events such as sending a packet to receiving the corresponding acknowledge, and sending a link-request to receiving the corresponding link-response. The reset value is the maximum time-out interval, and represents between 3 and 6 seconds.

Reference: This register implements the functionality described in Section 6.6.2 of the LP-Serial Physical Layer Specification Rev. 2.1.

Table 2-42. Port Link Time-out Control CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:23	time-out_value	Read Only	Set to 23'hFFFFFF	Time-out interval value.
24:31	reserved	Read Only	Set to 8'h00	

Port Response Time-out Control (PORT_RT_CTRL) CSR

Address: 0x000124

Description: The port response time-out control register contains the time-out timer count for all ports on a device. This time-out is for sending a request packet to receiving the corresponding response packet. The reset value is the maximum time-out interval, and represents between 3 and 6 seconds. The value stored in this register is presented to the logical layer functions on the timeout[0:23] vector.

Reference: This register implements the functionality described in Section 6.6.3 of the LP-Serial Physical Layer Specification Rev. 2.1.

Table 2-43. Port Response Time-out Control CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:23	time-out_value	Read Only	Set to 23'hFFFFFF	Time-out interval value.
24:31	reserved	Read Only	Set to 8'h00	

Port General Control (PORT_GEN_CTRL) CSR

Address: 0x00013C

Description: The bits accessible through the Port General Control CSR are bits that apply to all ports on a device. There is a single copy of each such bit per device. These bits are also accessible through the Port General Control CSR of any other physical layers implemented on a device.

Reference: This register implements the functionality described in Section 6.6.4 of the LP-Serial Physical Layer Specification Rev. 2.1.

Table 2-44. Port General Control CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0	Host	Read Only	Defined by GUI Selection	A Host device is a device that is responsible for system exploration, initialization, and maintenance. Agent or slave devices are typically initialized by Host devices. 1`b0 - agent or slave device 1`b1 - host device
1	Master Enable	Read Write	Defined by GUI Selection	The Master Enable bit controls whether or not a device is allowed to issue requests into the system. If the Master Enable is not set, the device may only respond to requests. 1`b0 - processing element cannot issue requests 1`b1 - processing element can issue requests
2	Discovered	Read Write	Defined by GUI Selection	This device has been located by the processing element responsible for system configuration 1`b0 - The device has not been previously discovered 1`b1 - The device has been discovered by another processing element
3:31	reserved	Read Only	Set to 29'd0.	

Port 0 Control 2 (PORT_0_CTRL2) CSR

Address: 0x000154

Description: This register contains assorted control bits.

Reference: This register implements the functionality described in the following document:

1. Section 6.6.10 of the LP-Serial Physical Layer Specification Rev. 2.1.

Table 2-45. Port 0 Control 2 CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:3	Selected Baudrate	Read Write	4`b0000	Indicates the initialized baudrate of the port 4`b0000 - no rate selected 4`b0001 - 1.25 GBaud 4`b0010 - 2.5 GBaud 4`b0011 - 3.125 GBaud 4`b0100 - 5.0 GBaud 4`b0101 - 6.25 GBaud 4`b0110 - 4`b1111 - Reserved
4	Baudrate Discovery Support	Read Only	GUI Selection	Indicates whether automatic baudrate discovery is supported. 1`b0 - Automatic baudrate discovery not supported 1`b1 - Automatic baudrate discovery supported
5	Baudrate Discovery Enable	Read Write	GUI Selection	Controls whether automatic baudrate discovery is enabled. 1`b0 - Automatic baudrate discovery disabled 1`b1 - Automatic baudrate discovery enable The port does not allow this bit to be set unless it supports baudrate discovery.
6	1.25 GBaud Support	Read Only	GUI Selection	Indicates whether port operation at 1.25 GBaud is supported. 1`b0 - 1.25 GBaud operation not supported 1`b1 - 1.25 GBaud operation supported

Table 2-45. Port 0 Control 2 CSR Bit-Fields (Continued)

Bit Field	Name	Type	Reset State	Description
7	1.25 GBaud Enable	Read Write	GUI Selection	Controls whether port operation at 1.25 GBaud is enabled. 1`b0 - 1.25 GBaud operation disabled 1`b1 - 1.25 GBaud operation enabled The port does not allow this bit to be set unless it supports 1.25 GBaud.
8	2.5 GBaud Support	Read Only	GUI Selection	Indicates whether port operation at 2.5 GBaud is supported. 1`b0 - 2.5 GBaud operation not supported 1`b1 - 2.5 GBaud operation supported
9	2.5 GBaud Enable	Read Write	GUI Selection	Controls whether port operation at 2.5 GBaud is enabled. 1`b0 - 2.5 GBaud operation disabled 1`b1 - 2.5 GBaud operation enabled The port does not allow this bit to be set unless it supports 2.5 GBaud.
10	3.125 GBaud Support	Read Only	GUI Selection	Indicates whether port operation at 3.125 GBaud is supported. 1`b0 - 3.125 GBaud operation not supported 1`b1 - 3.125 GBaud operation supported
11	3.125 GBaud Enable	Read Write	GUI Selection	Controls whether port operation at 3.125 GBaud is enabled. 1`b0 - 3.125 GBaud operation disabled 1`b1 - 3.125 GBaud operation enabled The port does not allow this bit to be set unless it supports 3.125 GBaud.
12	5.0 GBaud Support	Read Only	GUI Selection	Indicates whether port operation at 5.0 GBaud is supported. 1`b0 - 5.0 GBaud operation not supported 1`b1 - 5.0 GBaud operation supported
13	5.0 GBaud Enable	Read Write	GUI Selection	Controls whether port operation at 5.0 GBaud is enabled. 1`b0 - 5.0 GBaud operation disabled 1`b1 - 5.0 GBaud operation enabled The port does not allow this bit to be set unless it supports 5.0 GBaud.
14	6.25 GBaud Support	Read Only	GUI Selection	Indicates whether port operation at 6.25 GBaud is supported. 1`b0 - 6.25 GBaud operation not supported 1`b1 - 6.25 GBaud operation supported
15	6.25 GBaud Enable	Read Write	GUI Selection	Controls whether port operation at 6.25 GBaud is enabled. 1`b0 - 6.25 GBaud operation disabled 1`b1 - 6.25 GBaud operation enabled The port does not allow this bit to be set unless it supports 6.25 GBaud.
16:28	reserved	Read Only	13'd0	
29	Data scrambling disable	Read Write	1`b0	This bit is for test use only and shall not be asserted during normal operation. 1`b0 -transmit data scrambler and receive data descrambler are enabled. 1`b1 -transmit data scrambler and receive data descrambler are disabled.
30	Remote Transmit Emphasis Control Support	Read Only	GUI Selection	Indicates whether the port is able to transmit commands to control the transmit emphasis in the connected port. 1`b0 -The port does not support transmit emphasis adjustment in the connected port 1`b1 -The port supports transmit emphasis adjustment in the connected port
31	Remote Transmit Emphasis Control Enable	Read Write	GUI Selection	Controls whether the port may adjust the transmit emphasis in the connected port. 1`b0 -Remote transmit emphasis control is disabled 1`b1 -Remote transmit emphasis control is enabled The port shall not let this bit be set unless remote transmit emphasis control is supported and the link to which the port is connect is using idle sequence 2 (IDLE2).

Port 0 Error and Status (PORT_0_ERR_STAT) CSR

Address: 0x000158

Description: This register contains assorted port error and status bits.

Reference: This register implements the functionality described in the following documents:

1. Section 6.6.8 of the LP-Serial Physical Layer Specification Rev. 2.1.
2. Section 2.2 of the Error Management Extensions Specification Rev. 2.1.

Table 2-46. Port 0 Error and Status CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0	Idle Sequence 2 Support	Read Only	GUI Selection	Indicates whether the port supports idle sequence 2 for baudrates of less than 5.5 GBaud. 1`b0 -Idle sequence 2 not supported for baudrates < 5.5 GBaud. 1`b1 -Idle sequence 2 supported for baudrates < 5.5 GBaud
I	Idle Sequence 2 Enable	Read Write	GUI Selection	Controls whether idle sequence 2 is enabled for baudrates of less than 5.5 GBaud. 1`b0 -Idle sequence 2 disabled for baudrates < 5.5 GBaud. 1`b1 -Idle sequence 2 enabled for baudrates < 5.5 GBaud. The port does not allow this bit to be set unless idle sequence 2 is supported and shall not allow this bit to be cleared if only idle sequence 2 is supported.
2	Idle Sequence	Read Only	Set to 1`b0	Indicates which idle is active. 1`b0 -Idle sequence 1 is active. 1`b1 -Idle sequence 2 is active.
3:4	reserved	Read Only	Set to 2'd0	
5	Output Packet-dropped	Read Write	Set to 1`b0	Output port has discarded a packet. Once set remains set until written with a logical 1 to clear.
6	Output Failed-encountered	Read Write	Set to 1`b0	Output port has encountered a failed condition, meaning that the port's failed error threshold has been reached in the Port *n* Error Rate Threshold register. Once set remains set until written with a logical 1 to clear.
7	Output Degraded-encountered	Read Write	Set to 1`b0	Output port has encountered a degraded condition, meaning that the port's degraded error threshold has been reached in the Port *n* Error Rate Threshold register. Once set remains set until written with a logical 1 to clear.
8:10	reserved	Read Only	Set to 3'd0	
11	Output Retry-encountered	Read Write	Set to 1`b0	Output port has encountered a retry condition. This bit is set when bit 13 is set. Once set, remains set until written with a logical 1 to clear.
12	Output Retried	Read Only	Set to 1`b0	Output port has received a packet-retry control symbol and can not make forward progress. This bit is set when bit 13 is set and is cleared when a packet-accepted or a packet-not-accepted control symbol is received.
13	Output Retry-stopped	Read Only	Set to 1`b0	Output port has received a packet-retry control symbol and is in the "output retry-stopped" state.
14	Output Error-encountered	Read Write	Set to 1`b0	Output port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 15 is set. Once set, remains set until written with a logical 1 to clear.
15	Output Error-stopped	Read Only	Set to 1`b0	Output is in the "output error-stopped" state.
16:20	reserved	Read Only	Set to 5'd0	

Table 2-46. Port 0 Error and Status CSR Bit-Fields (Continued)

Bit Field	Name	Type	Reset State	Description
21	Input Retry-stopped	Read Only	Set to 1`b0	Input port is in the “input retry-stopped” state.
22	Input Error-encountered	Read Write	Set to 1`b0	Input port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 23 is set. Once set, remains set until written with a logical 1 to clear.
23	Input Error-stopped	Read Only	Set to 1`b0	Input port is in the “input error-stopped” state.
24:26	reserved	Read Only	Set to 3’d0	
27	Port-write Pending	Read Write	Set to 1`b0	Port has encountered a condition which required it to initiate a Maintenance Port-write operation This bit is only valid if the device is capable of issuing a maintenance port-write transaction. Once set remains set until written with a logical 1 to clear.
28	Port Unavailable	Read Only	GUI Selection	Indicates whether or not the port is available (read only). The port’s resources may have been merged with another port to support wider links. 1`b0 -The port is available for use. 1`b1 -The port is not available for use.
29	Port Error	Read Write	Set to 1`b0	Input or output port has encountered an error from which hardware was unable to recover. Once set, remains set until written with a logical 1 to clear.
30	Port OK	Read Only	Set to 1`b0	The input and output ports are initialized and the port is exchanging error-free control symbols with the attached device.
31	Port Uninitialized	Read Only	Set to 1`b1	Input and output ports are not initialized. This bit and bit 30 are mutually exclusive.

Port 0 Control (PORT_0_CTRL) CSR

Address: 0x00015C

Description: This register contains assorted control bits.

Reference: This register implements the functionality described in the following documents:

1. Section 6.6.9 of the LP-Serial Physical Layer Specification Rev. 2.1.
2. Section 2.2 of the Error Management Extensions Specification Rev. 2.1.

Table 2-47. Port 0 Control CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:1	Port Width Support	Read Only	GUI Selection	Indicates port width modes supported by the port (read-only). This field is used in conjunction with the Extended Port Width Support field of this register. The bits of these two fields collectively indicate the port width modes supported by the port in addition to 1x mode which is supported by all ports. Bit 0: 1`b0 -4x mode not supported 1`b1 -4x mode supported Bit 1: 1`b0 -2x mode not supported 1`b1 -2x mode supported
2:4	Initialized Port Width	Read Only	Set to 3`b000	Width of the ports after initialized: 3`b000 - Single-lane port 3`b001 - Single-lane port, lane R 3`b010 - Four-lane port 3`b011 - Two-lane port 3`b100 - Eight-lane port 3`b101 - Sixteen-lane port 3`b110 - 3`b111 - Reserved
5:7	Port Width Override	Read Write	Set to 3`b000	Soft port configuration to control the width modes available for port initialization. The meaning of bits 6 and 7 depend on the value of bit 5. Control of addition width modes is provided in the Extended Port Width Override field of this register. 3`b000 -No override, all supported mode widths enabled. 3`b001 -Reserved 3`b010 - Force single lane 3`b011 - Force single lane, lane R 3`b100 -Reserved 3`b101 - 2x mode enabled, 4x mode disabled 3`b110 - 4x mode enabled, 2x mode disabled 3`b111 - 2x and 4x modes enabled The port does not allow the enabling of a width mode that is not supported by the port. A change in the value of the Port Width Override field causes the port to re-initialize using the new field value.
8	Port Disable	Read Write	Set to 1`b0	Port disable: 1`b0 - port receivers/drivers are enabled 1`b1 - port receivers/drivers are disabled and are unable to receive or transmit any packets or control symbols
9	Output Port Enable	Read Write	GUI Selection	Output port transmit enable: 1`b0 -port is stopped and not enabled to issue any packets except to route or respond to I/O logical maintenance packets. Control symbols are not affected and are sent normally. This is the recommended state after device reset. 1`b1 -port is enabled to issue any packets
10	Input Port Enable	Read Write	GUI Selection	Input port receive enable: 1`b0 - port is stopped and only enabled to route or respond I/O logical MAINTENANCE packets. Other packets generate packet-not-accepted control symbols to force an error condition to be signaled by the sending device. Control symbols are not affected and are received and handled normally. This is the recommended state after device reset. 1`b1 - port is enabled to respond to any packet

Table 2-47. Port 0 Control CSR Bit-Fields (Continued)

Bit Field	Name	Type	Reset State	Description
11	Error Checking Disable	Read Write	1`b0	This bit disables all RapidIO transmission error checking: 1`b0 -Error checking and recovery is enabled 1`b1 -Error checking and recovery is disabled Device behavior when error checking and recovery is disabled and an error condition occurs is undefined.
12	Multicast-event Participant	Read Write	GUI Selection	Send incoming Multicast-event control symbols to this port (multiple port devices only).
13	reserved	Read Only	Set to 1`b0	
14	Enumeration Boundary	Read Write	GUI Selection	An enumeration boundary aware system enumeration algorithm shall honor this flag. The algorithm, on either the ingress or the egress port, shall not enumerate past a port with this bit set. This provides for software enforced enumeration domains within the RapidIO fabric.
15	reserved	Read Only	Set to 1'd0	
16:17	Extended Port Width Override	Read Only	Set to 2'd0	Extended soft port configuration to control the width modes available for port initialization. The meaning of these two bits depends on the value of bit 5 of this register. If bit 5 is set to 0 the values of bits 16:17 have no meaning and shall be ignored. If bit 5 is set to 1 bits 15:17 will be encoded as follows: Bit 15: 1`b0 - 8x mode is disabled 1`b1 - 8x mode is enabled Bit 16: 1`b0 - 16x is disabled. 1`b1 - 16x mode is enabled. The port does not allow the enabling of a width mode that is not supported by the port. When bit 5 = 1, a change in the value of this field causes the port to re-initialize using the new bit values.
18:19	Extended Port Width Support	Read Only	Set to 2'd0	Indicates additional port width modes supported by the port (read-only). This field is used in conjunction with the Port Width Support field of this register. The bits of these two fields collectively indicate the port width modes supported by the port in addition to 1x mode which is supported by all ports. Bit 18: 1`b0 - 8x mode not supported 1`b1 - 8x mode supported Bit 19: 1`b0 - 16x mode not supported 1`b1 - 16x mode supported
20:27	reserved	Read Only	Set to 8'd0	
28	Stop on Port Failed-encountered Enable	Read Write	Set to 1`b0	This bit is used with the Drop Packet Enable bit to force certain behavior when the Error Rate Failed Threshold has been met or exceeded. See Section 1.2.4 of the Part 8: Error Management Extensions for detailed requirements.
29	Drop Packet Enable	Read Write	Set to 1`b0	This bit is used with the Stop on Port Failed-encountered Enable bit to force certain behavior when the Error Rate Failed Threshold has been met or exceeded. See Section 1.2.4 of the Part 8: Error Management Extensions for detailed requirements.

Table 2-47. Port 0 Control CSR Bit-Fields (Continued)

Bit Field	Name	Type	Reset State	Description
30	Port Lockout	Read Write	Set to 1`b0	When this bit is cleared, the packets that may be received and issued are controlled by the state of the Output Port Enable and Input Port Enable bits in the Control CSR. When this bit is set, this port is stopped and is not enabled to issue or receive any packets; the input port can still follow the training procedure and can still send and respond to link-requests; all received packets return packet-not-accepted control symbols to force an error condition to be signaled by the sending device.
31	Port Type	Read Only	Set to 1`b1	This indicates the port type (read only) 1`b0 – Reserved 1`b1 - Serial port

LP-Serial Lane Register Block Header (LP_SERIAL_LANE_HDR)

Address: 0x001000

Description: The LP-Serial lane register block header contains the EF_PTR to the next extended features block and the EF_ID that identifies this as the LP-Serial lane register block header. In this implementation this is a pointer to the error management extensions register block.

Reference: This register implements the functionality described in Section 6.7.2.1 of the LP-Serial Physical Layer Specification Rev. 2.1.

Table 2-48. LP-Serial Lane Register Block Header Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:15	EF_PTR	Read Only	Set to 16'h2000	Pointer to the next block in the extended features data structure.
16:31	EF_ID	Read Only	Set to 16'h000D	

Lane n Status 0 (LP_N_STATUS_0) CSRs

Address: 0x001010 + n*0x20

Description: This register contains status information about the local lane transceiver, i.e. the lane n transceiver in the device implementing this register.

Reference: This register implements the functionality described in Section 6.7.2.2 of the LP-Serial Physical Layer Specification Rev. 2.1.

Table 2-49. Lane n Status 0 CSRs Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:7	Port Number	Read Only	Port Number[0:3] = 4'd0; Port Number[4:7] = GUI Selection	The number of the port within the device to which the lane is assigned.
8:11	Lane Number	Read Only	4'dn – where n is the lane number	The number of the lane within the port to which the lane is assigned.
12	Transmitter type	Read Only	This bit reflects the state of the tx_lane_type[n] signal on the Transceiver Interface.	Transmitter type 1`b0 – Short run 1`b1 – Long run
13	Transmitter mode	Read Only	This bit reflects the state of the tx_lane_mode[n] signal on the Transceiver Interface.	Transmitter operating mode 1`b0 – Short run 1`b1 – Long run

Table 2-49. Lane n Status 0 CSRs Bit-Fields (Continued)

Bit Field	Name	Type	Reset State	Description
14:15	Receiver type	Read Only	This bit reflects the state of the rx_lane_type[2n:2n+1] signals on the Transceiver Interface.	Receiver type 2`b00 – Short run 2`b01 – Medium run 2`b10 – Long run 2`b11 – Reserved
16	Receiver input inverted	Read Only	This bit reflects the state of the rx_lane_inv[n] signal on the Transceiver Interface.	This bit indicates whether the lane receiver has detected that the polarity of its input signal is inverted and has inverted its receiver input to correct the polarity. 1`b0 – Receiver input not inverted 1`b1 – Receiver input inverted
17	Receiver trained	Read Only	This bit reflects the state of the rx_lane_trained[n] signal on the Transceiver Interface.	When the lane receiver controls any transmit or receive adaptive equalization, this bit indicates whether or not all adaptive equalizers controlled by the lane receiver are trained. If the lane supports the IDLE2 sequence, the value of this bit shall be the same as the value in the “Receiver trained” bit in the CS Field transmitted by the lane. 1`b0 – One or more adaptive equalizers are controlled by the lane receiver and at least one of those adaptive equalizers is not trained 1`b1 – The lane receiver controls no adaptive equalizers or all of the adaptive equalizers controlled by the lane receiver are trained
18	Receiver lane sync	Read Only	This bit reflects the state of the rx_lane_sync[n] signal on the Transceiver Interface.	This bit indicates the state of the lane’s lane_sync signal. 1`b0 – Lane_sync FALSE 1`b1 – Lane_sync TRUE
19	Receiver lane ready	Read Only	This bit reflects the logical and of the rx_lane_trained[n] and the rx_lane_sync[n] signals on the Transceiver Interface.	This bit indicates the state of the lane’s lane_ready signal. 1`b0 – Lane_ready FALSE 1`b1 – Lane_ready TRUE

Table 2-49. Lane n Status 0 CSRs Bit-Fields (Continued)

Bit Field	Name	Type	Reset State	Description
20:23	8b10b decoding errors	Read Only	4'd0	This field indicates the number of 8b10b decoding errors that have been detected for this lane since this register was last read. The field is reset to 4'h0 when the register is read. 4'h0 – No 8b10b decoding errors have been detected since the register was last read. 4'h1 – One 8b10b decoding error has been detected since this register was last read. 4'h2 – Two 8b10b decoding errors have been detected since this register was last read. ... 4'hD – 13 8b10b decoding errors have been detected since the register was last read. 4'hE – 14 8b10b decoding errors have been detected since this register was last read. 4'hF – At least 15 8b10b decoding errors have been detected since this register was last read.
24	Lane_sync state change	Read Only	1`b0	Indicates whether the lane_sync signal for this lane has changed state since the bit was last read. This bit is reset to 1`b0 when the register is read. This bit provides an indication of the burstiness of the transmission errors detected by the lane receiver. 1`b0 – The state of lane_sync has not changed since this register was last read 1`b1 – The state of lane_sync has changed since this register was last read
25	Rcvr_trained state change	Read Only	1`b0	Indicates whether the rcvr_trained signal for this lane has changed state since the bit was last read. This bit is reset to 1`b0 when the register is read. A change in state of rcvr_trained indicates that the training state of the adaptive equalization under the control of this receiver has changed. Frequent changes of the training state suggest a problem with the lane. 1`b0 – The state of rcvr_trained has not changed since this register was last read 1`b1 – The state of rcvr_trained has changed since this register was last read
26:27	reserved	Read Only	2'd0	

Table 2-49. Lane n Status 0 CSRs Bit-Fields (Continued)

Bit Field	Name	Type	Reset State	Description
28	Status 1 CSR implemented	Read Only	1`b0	This bit indicates whether or not the Status 1 CSR is implemented for this lane 1`b0 – The Status 1 CSR is not implemented for this lane 1`b1 – The Status 1 CSR is implemented for this lane
29:31	Status 2-7 CSRs implemented	Read Only	3'd0	This field indicates the number of implementation specific Status 2-7 CSRs that are implemented for this lane 3`b000 – None of the Status 2-7 CSRs are implemented for this lane 3`b001 – The Status 2 CSR is implemented for this lane 3`b010 – The Status 2 and 3 CSRs are implemented for this lane 3`b011 – The Status 2 through 4 CSRs are implemented for this lane 3`b100 – The Status 2 through 5 CSRs are implemented for this lane 3`b101 – The Status 2 through 6 CSRs are implemented for this lane 3`b110 – The Status 2 through 7 CSRs are implemented for this lane 3`b111 – Reserved

Error Reporting Block Header (ERB_HDR) CSR

Address: 0x002000

Description: The error management extensions block header register contains the EF_PTR to the next EF_BLK and the EF_ID that identifies this as the error management extensions block header. In this implementation this is the last block and therefore the EF_PTR is a NULL pointer.

Reference: This register implements the functionality described in Section 2.3.2.1 of the Error Management Extensions Specification Rev. 2.1.

Table 2-50. Error Reporting Block Header CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:15	EF_PTR	Read Only	Set to 16'h0000	Pointer to the next block in the extended features data structure.
16:31	EF_ID	Read Only	Set to 16'h0007	

Logical/Transport Layer Error Detect (ERB_LT_ERR_DET) CSR

Address: 0x002008

Description: This register indicates the error that was detected by the Logical or Transport logic layer. Multiple bits may get set in the register if simultaneous errors are detected during the same clock cycle that the errors are logged.

Reference: This register implements the functionality described in Section 2.3.2.2 of the Error Management Extensions Specification Rev. 2.1.

Table 2-51. Logical/Transport Layer Error Detect CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0	I/O error response	Read Write	Set to 1`b0	Received a response of 'ERROR' for an I/O Logical Layer Request.
1	Message error response	Read Write	Set to 1`b0	Received a response of 'ERROR' for an MSG Logical Layer Request.
2	GSM error response	Read Write	Set to 1`b0	Received a response of 'ERROR' for a GSM Logical Layer Request.
3	Message Format Error	Read Write	Set to 1`b0	Received MESSAGE packet data payload with an invalid size or segment (MSG logical)
4	Illegal transaction decode	Read Write	Set to 1`b0	Received illegal fields in the request/response packet for a supported transaction (I/O/MSG/GSM logical)
5	Illegal transaction target error	Read Write	Set to 1`b0	Received a packet that contained a destination ID that is not defined for this end point. End points with multiple ports and a built-in switch function may not report this as an error (Transport)
6	Message Request Time-out	Read Write	Set to 1`b0	A required message request has not been received within the specified time-out interval (MSG logical)
7	Packet Response Time-out	Read Write	Set to 1`b0	A required response has not been received within the specified time out interval (I/O/MSG/GSM logical)
8	Unsolicited Response	Read Write	Set to 1`b0	An unsolicited/unexpected Response packet was received (I/O/MSG/GSM logical)
9	Unsupported Transaction	Read Write	Set to 1`b0	A transaction is received that is not supported in the Destination Operations CAR (I/O/MSG/GSM logical)
10-23	Reserved	Read Only	Set to 14`d0	
24-31	Implementation Specific error enable	Read Only	Set to 8`d0	An implementation specific error has occurred. These indications can be generated by user defined logical layer functions.

Logical/Transport Layer Error Enable (ERB_LT_ERR_EN) CSR

Address: 0x00200C

Description: This register contains the bits that control if an error condition locks the Logical /Transport Layer Error Detect and Capture registers and is reported to the system host.

Reference: This register implements the functionality described in Section 2.3.2.3 of the Error Management Extensions Specification Rev. 2.1.

Table 2-52. Logical/Transport Layer Error Enable CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0	I/O error response enable	Read Write	Set to 1`b0	Enable reporting of an I/O error response. Save and lock original request transaction information in all Logical/Transport Layer Capture CSRs.
1	Message error response enable	Read Write	Set to 1`b0	Enable reporting of a Message error response. Save and lock original request transaction information in all Logical/Transport Layer Capture CSRs.
2	GSM error response enable	Read Write	Set to 1`b0	Enable reporting of a GSM error response. Save and lock original request transaction capture information in all Logical/Transport Layer Capture CSRs.
3	Message Format Error enable	Read Write	Set to 1`b0	Enable reporting of a message format error. Save and lock transaction capture information in Logical/Transport Layer Device ID and Control Capture CSRs.

Table 2-52. Logical/Transport Layer Error Enable CSR Bit-Fields (Continued)

Bit Field	Name	Type	Reset State	Description
4	Illegal transaction decode enable	Read Write	Set to 1`b0	Enable reporting of an illegal transaction decode error Save and lock transaction capture information in Logical/Transport Layer Device ID and Control Capture CSRs.
5	Illegal transaction target error enable	Read Write	Set to 1`b0	Enable reporting of an illegal transaction target error. Save and lock transaction capture information in Logical/Transport Layer Device ID and Control Capture CSRs.
6	Message Request time-out enable	Read Write	Set to 1`b0	Enable reporting of a Message Request time-out error. Save and lock transaction capture information in Logical/Transport Layer Device ID and Control Capture CSRs for the last Message request segment packet received.
7	Packet Response Time-out error enable	Read Write	Set to 1`b0	Enable reporting of a packet response time-out error. Save and lock original request address in Logical/Transport Layer Address Capture CSRs. Save and lock original request Destination ID in Logical/Transport Layer Device ID Capture CSR.
8	Unsolicited Response error enable	Read Write	Set to 1`b0	Enable reporting of an unsolicited response error. Save and lock transaction capture information in Logical/Transport Layer Device ID and Control Capture CSRs. (switch or endpoint device)
9	Unsupported Transaction error enable	Read Write	Set to 1`b0	Enable reporting of an unsupported transaction error. Save and lock transaction capture information in Logical/Transport Layer Device ID and Control Capture CSRs. (switch or endpoint device)
10-23	Reserved	Read Only	Set to 14`d0	
24-31	Implementation Specific error enable	Read Only	Set to 8`d0	Enable reporting of an implementation specific error. Save and lock capture information in appropriate Logical/Transport Layer Capture CSRs.

Logical/Transport Layer High Address Capture (ERB_LT_ADDR_CAPT_H) CSR

Address: 0x002010

Description: This register contains error information. It is locked when a Logical/Transport error is detected and the corresponding enable bit is set. This register is only required for end point devices that support 66 or 50 bit addresses.

Reference: This register implements the functionality described in Section 2.3.2.4 of the Error Management Extensions Specification Rev. 2.1.

Table 2-53. Logical/Transport Layer High Address Capture CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	address[0-31]	Read Write	Set to 32'd0	Most significant 32 bits of the address associated with the error (for requests, for responses if available)

Logical/Transport Layer Address Capture (ERB_LT_ADDR_CAPT) CSR

Address: 0x002014

Description: This register contains error information. It is locked when a Logical/Transport error is detected and the corresponding enable bit is set.

Reference: This register implements the functionality described in Section 2.3.2.5 of the Error Management Extensions Specification Rev. 2.1.

Table 2-54. Logical/Transport Layer Address Capture CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:28	address[32-60]	Read Write	Set to 29'd0	Least significant 29 bits of the address associated with the error (for requests, for responses if available)
29	reserved	Read Only	Set to 1`b0	
30:31	xamsbs	Read Write	Set to 2'd0	Extended address bits of the address associated with the error (for requests, for responses if available)

Logical/Transport Layer Device ID Capture (ERB_LT_DEV_ID_CAPT) CSR

Address: 0x002018

Description: This register contains error information. It is locked when a Logical/Transport error is detected and the corresponding enable bit is set.

Reference: This register implements the functionality described in Section 2.3.2.6 of the Error Management Extensions Specification Rev. 2.1.

Table 2-55. Logical/Transport Layer Device ID Capture CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:7	MSB destinationID	Read Write	Set to 29'd0	Most significant byte of the destinationID associated with the error (large transport systems only)
8:15	destinationID	Read Write	Set to 29'd0	The destinationID associated with the error
16:23	MSB sourceID	Read Write	Set to 29'd0	Most significant byte of the sourceID associated with the error (large transport systems only)
24:31	sourceID	Read Write	Set to 29'd0	The sourceID associated with the error

Logical/Transport Layer Control Capture (ERB_LT_CTRL_CAPT) CSR

Address: 0x00201C

Description: This register contains error information. It is locked when a Logical/Transport error is detected and the corresponding enable bit is set.

Reference: This register implements the functionality described in Section 2.3.2.7 of the Error Management Extensions Specification Rev. 2.1.

Table 2-56. Logical/Transport Layer Control Capture CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:3	ftype	Read Write	Set to 4'd0	Format type associated with the error
4:7	ttype	Read Write	Set to 4'd0	Transaction type associated with the error
8:15	msg info	Read Write	Set to 8'd0	letter, mbox, and msgseg for the last Message request received for the mailbox that had an error (Message errors only)
16:26	implementation defined	Read Write	Set to 9'd0	Implementation defined field from logN_error_capt_data[112:122]. In the reference design, bit [25:26] are the logical port number that reported the error.
27:31	error type	Read Write	Set to 5'd0	Error type indication taken from bits 123 to 127 of the logN_error_capt_data vector on the logical port that reported the error.

Port-write Target deviceID (ERB_LT_DEV_ID) CSR

Address: 0x002028

Description: This register contains the target deviceID to be used when a device generates a Maintenance port-write operation to report errors to a system host.

Reference: This register implements the functionality described in Section 2.3.2.8 of the Error Management Extensions Specification Rev. 2.1.

Table 2-57. Port-write Target deviceID CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:7	deviceID_msb	Read Write	Set to 8'd0	This is the most significant byte of the port-write target deviceID (large transport systems only)
8:15	deviceID	Read Write	Set to 8'd0	This is the port-write target deviceID
16	large_transport	Read Write	Set to 1`b0	deviceID size to use for a port-write 1`b0 - use the small transport deviceID 1`b1 - use the large transport deviceID
17:31	reserved	Read Only	Set to 15'd0	

Port 0 Error Detect (ERB_ERR_DET) CSR

Address: 0x002040

Description: The Port 0 Error Detect Register indicates physical layer transmission errors detected by the hardware.

Reference: This register implements the functionality described in Section 2.3.2.10 of the Error Management Extensions Specification Rev. 2.1.

Table 2-58. Error Detect CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0	Implementation specific error	Read Write	Set to 1`b0	An implementation specific error has been detected
1:8	reserved	Read Only	Set to 8'd0	
9	Received corrupt control symbol	Read Write	Set to 1`b0	Received a control symbol with a bad CRC value
10	Received acknowledge control symbol with unexpected ackID	Read Write	Set to 1`b0	Received an acknowledge control symbol with an unexpected ackID (packet-accepted or packet_retry)
11	Received packet-not-accepted control symbol	Read Write	Set to 1`b0	Received packet-not-accepted acknowledge control symbol
12	Received packet with unexpected ackID	Read Write	Set to 1`b0	Received packet with unexpected ackID value - out-of-sequence ackID
13	Received packet with bad CRC	Read Write	Set to 1`b0	Received packet with a bad CRC value
14	Received packet exceeds 276 Bytes	Read Write	Set to 1`b0	Received packet which exceeds the maximum allowed size
15	Received illegal or invalid character	Read Write	Set to 1`b0	Received an 8b10b code-group that is invalid (a code-group that does not have a 8b10b decode given the current running disparity) or illegal (a code-group that is valid, but whose use is not allowed by the LP-Serial protocol). This bit may be set in conjunction with bit 29 Delineation error.
16	Received data character in IDLE1 sequence	Read Write	Set to 1`b0	Received a data character in an IDLE1 sequence. This bit may be set in conjunction with bit 29 Delineation error.

Table 2-58. Error Detect CSR Bit-Fields (Continued)

Bit Field	Name	Type	Reset State	Description
17	Loss of descrambler synchronization	Read Write	Set to 1`b0	Loss of receiver descrambler synchronization while receiving scrambled control symbol and packet data.
18:25	reserved	Read Only	Set to 8'd0	
26	Non-outstanding ackID	Read Write	Set to 1`b0	Link_response received with an ackID that is not outstanding
27	Protocol error	Read Write	Set to 1`b0	An unexpected packet or control symbol was received
28	reserved	Read Only	Set to 1`b0	
29	Delineation error	Read Write	Set to 1`b0	Received an 8b10b code-group that is invalid (a code-group that does not have a 8b10b decode given the current running disparity), illegal (a code-group that is valid, but whose use is not allowed by the LP-Serial protocol) or that is in a position in the received code-group stream that is not allowed by the LP-Serial protocol.
30	Unsolicited acknowledge control symbol	Read Write	Set to 1`b0	An unexpected packet acknowledgement control symbol was received
31	Link time-out	Read Write	Set to 1`b0	A packet acknowledgement or link-response control symbol was not received within the specified time-out interval not received within the specified time-out interval

Port 0 Error Rate Enable (ERB_ERR_RATE_EN) CSR

Address: 0x002044

Description: This register contains the bits that control when an error condition is allowed to increment the error rate counter in the Port n Error Rate Threshold Register and lock the Error Capture registers.

Reference: This register implements the functionality described in Section 2.3.2.11 of the Error Management Extensions Specification Rev. 2.1.

Table 2-59. Error Rate Enable CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0	Implementation specific error	Read Write	Set to 1`b0	An implementation specific error has been detected
1:8	reserved	Read Only	Set to 8'd0	
9	Received control symbol with bad CRC enable	Read Write	Set to 1`b0	Enable error rate counting of a corrupt control symbol
10	Received out-of-sequence acknowledge control symbol enable	Read Write	Set to 1`b0	Enable error rate counting of an acknowledge control symbol with an unexpected ackID
11	Received packet-not-accepted control symbol enable	Read Write	Set to 1`b0	Enable error rate counting of received packet-not-accepted control symbols
12	Received packet with unexpected ackID enable	Read Write	Set to 1`b0	Enable error rate counting of packet with unexpected ackID value - out-of-sequence ackID
13	Received packet with Bad CRC enable	Read Write	Set to 1`b0	Enable error rate counting of packet with a bad CRC value
14	Received packet exceeds 276 Bytes enable	Read Write	Set to 1`b0	Enable error rate counting of packet which exceeds the maximum allowed size
15	Received illegal or invalid character enable	Read Write	Set to 1`b0	Enable error rate counting of reception of an 8b10b code-group that is invalid or illegal.

Table 2-59. Error Rate Enable CSR Bit-Fields (Continued)

Bit Field	Name	Type	Reset State	Description
16	Received data character in an IDLE1 sequence enable	Read Write	Set to 1`b0	Enable error rate counting of reception of a data character in an IDLE1 sequence.
17	Loss of descrambler synchronization enable	Read Write	Set to 1`b0	Enable error rate counting of loss of receiver descrambler synchronization when scrambled control symbol and packet data is being received.
18:25	reserved	Read Only	Set to 9'd0	
26	Non-outstanding ackID enable	Read Write	Set to 1`b0	Enable error rate counting of link-responses received with an ackID that is not outstanding
27	Protocol error enable	Read Write	Set to 1`b0	Enable error rate counting of protocol errors
28	Frame toggle edge error enable	Read Write	Set to 1`b0	Enable error rate counting of frame toggle edge errors
29	Delineation error enable	Read Write	Set to 1`b0	Enable error rate counting of reception of an 8b10b code-group that is invalid, illegal or that is in a position in the received code-group stream that is not allowed by the LP-Serial protocol.
30	Unsolicited acknowledgement control symbol enable	Read Write	Set to 1`b0	Enable error rate counting of received unsolicited packet acknowledgement control symbols.
31	Link time-out enable	Read Write	Set to 1`b0	Enable error rate counting of link time-out errors.

Port 0 Attributes Capture (ERB_ATTR_CAPT) CSR

Address: 0x002048

Description: This register contains the bits that control when an error condition is allowed to increment the error rate counter in the Port n Error Rate Threshold Register and lock the Error Capture registers.

Reference: This register implements the functionality described in Section 2.3.2.12 of the Error Management Extensions Specification Rev. 2.1.

Table 2-60. Attributes Capture CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:2	Info type	Read Write	Set to 2`b000	Type of information logged 3`b000 – Packet 3`b001 – Reserved 3`b010 – Short control symbol 3`b011 – Long control symbol 3`b100 – Implementation specific (capture register contents are implementation specific) 3`b101 – Reserved 3`b110 – Undefined (S-bit error), capture as if a packet (parallel physical layer only) 3`b111 – Reserved
3:7	Error type	Read Write	Set to 5'd0	The encoded value of the bit in the Port *n* Error Detect CSR that describes the error captured in the Port *n* Packet/Control Symbol Capture 0-3 CSRs.
8:27	Implementation Dependent	Read Write	Set to 20'd0	Implementation Dependent Error Information
28:30	reserved	Read Only	Set to 3`b000	
31	Capture valid info	Read Write	Set to 1`b0	This bit is set by hardware to indicate that the Port *n* Packet/Control Symbol Capture 0-3 CSRs and the other bits in this register contain valid information and are locked. This bit is cleared and the Port *n* Packet/Control Symbol Capture 0-3 CSRs and the other bits in this register are unlocked by software writing 0b0 to the bit.

Port 0 Packet/Control Symbol Capture (ERB_PACK_SYM_CAPT) CSR

Address: 0x00204C

Description: This register contains either captured control symbol information or the first 4 bytes of captured packet information.

Reference: This register implements the functionality described in Section 2.3.2.13 of the Error Management Extensions Specification Rev. 2.1.

Table 2-61. Attributes Capture CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	Capture 0	Read Write	Set to 32'd0	If the info_type field of the Port n Attributes Capture CSR is "long control symbol" or "short control symbol" or Delimited short or long control symbol bytes 0-3. If the info_type field of the Port *n* Attributes Capture CSR is "packet", packet bytes 0-3.

Port 0 Packet Capture 1 (ERB_PACK_CAPT_1) CSR

Address: 0x002050

Description: Error capture register 1 contains bytes 4 through 7 of the packet header.

Reference: This register implements the functionality described in Section 2.3.2.14 of the Error Management Extensions Specification Rev. 2.1.

Table 2-62. Packet Capture 1 CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	Capture 1	Read Write	Set to 32'd0	If the info_type field of the Port n Attributes Capture CSR is "long control symbol", delimited long control symbol Bytes 4-7. If the info_type field of the Port n Attributes Capture CSR is "packet", packet Bytes 4-7. Otherwise these bits are set to 32'd0.

Port 0 Packet Capture 2 (ERB_PACK_CAPT_2) CSR

Address: 0x002054

Description: Error capture register 2 contains bytes 8 through 11 of the packet header.

Reference: This register implements the functionality described in Section 2.3.2.15 of the Error Management Extensions Specification Rev. 2.1.

Table 2-63. Packet Capture 2 CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	Capture 2	Read Write	Set to 32'd0	If the info_type field of the Port n Attributes Capture CSR is "packet", packet Bytes 8-11. Otherwise these bits are set to 32'd0.

Port 0 Packet Capture 3 (ERB_PACK_CAPT_3) CSR

Address: 0x002058

Description: Error capture register 3 contains bytes 12 through 15 of the packet header.

Reference: This register implements the functionality described in Section 2.3.2.16 of the Error Management Extensions Specification Rev. 2.1.

Table 2-64. Packet Capture 3 CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	Capture 3	Read Write	Set to 32'd0	If the info_type field of the Port n Attributes Capture CSR is "packet", packet Bytes 12-15. Otherwise these bits are set to 32'd0.

Error Rate (ERB_ERR_RATE) CSR

Address: 0x002068

Description: The Port n Error Rate register is a 32-bit register used with the Error Rate Threshold register to monitor and control the reporting of transmission errors.

Reference: This register implements the functionality described in Section 2.3.2.17 of the Error Management Extensions Specification Rev. 2.1.

Table 2-65. Error Rate CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:7	Error Rate Bias	Read Write	Set to 8'h80	This field specifies the rate at which the Error Rate Counter is decremented (the error rate bias value). 8'h 00 – Do not decrement the error rate counter 8'h 01 – Decrement every 1ms (+/-34%) 8'h 02 – Decrement every 10ms (+/-34%) 8'h 04 – Decrement every 100ms (+/-34%) 8'h 08 – Decrement every 1s (+/-34%) 8'h 10 – Decrement every 10s (+/-34%) 8'h 20 – Decrement every 100s (+/-34%) 8'h 40 – Decrement every 1000s (+/-34%) 8'h 80 – Decrement every 10000s (+/-34%) Other values are reserved
8:13	reserved	Read Only	Set to 6'd0	
14:15	Error Rate Recovery	Read Write	Set to 2`b00	The value of this field limits the incrementing of the Error Rate Counter above the failed threshold trigger. 2`b00 – Only count 2 errors above 2`b01 – Only count 4 errors above 2`b10 – Only count 16 error above 2`b11 – Do not limit incrementing the error rate count
16:23	Peak Error Rate	Read Write	Set to 8'h00	This field contains the peak value attained by the error rate counter since the field was last reset.
24:31	Error Rate Counter	Read Write	Set to 8'h00	This field contains a count of the number of transmission errors that have been detected by the port, decremented by the Error Rate Bias mechanism, to create an indication of the link error rate.

Error Rate Threshold (ERB_ERR_RATE_THR) CSR

Address: 0x00206C

Description: The Port n Error Rate Threshold register is a 32-bit register used to control the reporting of the link status to the system host.

Reference: This register implements the functionality described in Section 2.3.2.18 of the Error Management Extensions Specification Rev. 2.1.

Table 2-66. Error Rate Threshold CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:7	Error Rate Failed Threshold Trigger	Read Write	Set to 8'hff	This field contains the threshold value for reporting an error condition due to a possibly broken link. 0x00 - Disable the Error Rate Failed Threshold Trigger 0x01 - Set the error reporting threshold to 1 0x02 - Set the error reporting threshold to 2 ... 0xFF - Set the error reporting threshold to 255
8-15	Error Rate Degraded Threshold Trigger	Read Write	Set to 8'hff	This field contains the threshold value for reporting an error condition due to a degrading link. 0x00 - Disable the Error Rate Degraded Threshold Trigger 0x01 - Set the error reporting threshold to 1 0x02 - Set the error reporting threshold to 2 ... 0xFF - Set the error reporting threshold to 255
16-31	reserved	Read Only	Set to 16'd0	

Buffer Configuration (IR_BUFFER_CONFIG) CSR

Address: 0x102000

Description: This register controls the operation of the Tx and Rx buffer logic.

Table 2-67. Buffer Configuration CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
00:03	WM0	Read Write	Set to 4'd4	These fields control the Transmitter-Controlled flow control watermark thresholds as described in Section 5.9.2.1 of the LP-Serial Specification. The values contained in these fields are zero-extended before being used internally.
04:07	WM1	Read Write	Set to 4'd3	
08:11	WM2	Read Write	Set to 4'd2	
12:28	reserved	Read Only	Set to 17'd0	
29	Tx Flow Control Enable	Read Write	GUI Selection	Controls whether Transmitter-Controlled flow control is enabled. 0 – Disabled 1 – Enabled
30	Tx Synchronization Mode	Read Write	Set to 1`b1	Controls whether the synchronizers are enabled between the link_clk domain and the rio_clk domain. In the transmit direction. 0 – Synchronizers are enabled 1 – Synchronizers are disabled
31	Rx Synchronization Mode	Read Write	Set to 1`b1	Controls whether the synchronizers are enabled between the link_clk domain and the rio_clk domain. In the receive direction. 0 – Synchronizers are enabled 1 – Synchronizers are disabled

Event Status (IR_EVENT_STAT) CSR

Address: 0x106100

Description: This register displays the unmasked status of the system events. Table 2-68 shows how each bit in this register maps to specific events.

Table 2-68. Event Status CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	Event Status	Read Write	Set to 32'h00000000	Each bit displays the unmasked status of one of the error events. The bit is set when an event occurs. Each event is cleared by writing a 1`b0 to the bit or bits that are set. Bits corresponding to unimplemented event conditions are treated as reserved.

Event Enable (IR_EVENT_ENBL) CSR

Address: 0x106104

Description:This register enables the generation of event messages and or an interrupt upon the occurrence of system events. Table 2-69 shows how each bit in this register maps to specific events.

Table 2-69. Event Enable CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	Event Enable	Read Write	Set to 32'h00000000	When set, each bit enables one of the event types. Bits corresponding to unimplemented event conditions are treated as reserved.

Event Force (IR_EVENT_FORCE) CSR

Address: 0x106108

Description: This register is used to force the generation of event messages. Table 2-70 shows how each bit in this register maps to specific events.

Table 2-70. Event Force CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	Event Force	Write Only reads as 32'h00000000	Set to 32'h00000000	When set each bit generates one of the event types if it is unmasked. Bits corresponding to unimplemented event conditions are treated as reserved.

Event Cause (IR_EVENT_CAUSE) CSR

Address: 0x10610C

Description: This register displays the status of the error events masked by the contents of the IR_EVENT_ENBL register. Table 2-71 shows how each bit in this register maps to specific events.

Table 2-71. Event Cause CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	Event Cause	Read Write	Set to 32'h00000000	Each bit displays the masked status of one of the error events. The bit is set when an event occurs and it has not been masked. Bits corresponding to unimplemented event conditions are treated as reserved.

Event Overflow (IR_EVENT_OVRFLOW) CSR

Address: 0x106110

Description: This register displays if there has been one or more occurrences of event an event since the corresponding event bit was set in the IR_EVENT_STAT register. Table 2-72 shows how each bit in this register maps to specific events.

Table 2-72. Event Force CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	Event Overflow	Read Write	Set to 32'h00000000	Each bit displays the overflow state of an event. Bits corresponding to unimplemented event conditions are treated as reserved.

Event Configuration (IR_EVENT_CONFIG) CSR

Address: 0x106114

Description: This register controls the operation of the event unit.

Table 2-73. Event Configuration CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:15	Dest ID	Read Write	Set to 16'h0000	When the core is configured for in-band event reporting, this field configures the destinationID in outgoing doorbell or port write packets.
16:29	reserved	Read Only	Set to 14'd0	
30	Event Message Transport Type	Read Write	Set to 1`b0	When the core is configured for in-band event reporting, this bit controls whether small or large transport format is used for doorbell or port write packets. 0 – Use small transport 1 – Use large transport
31	Event Message Enable	Read Write	Set to 1`b0	When the core is configured for in-band event reporting, this bit controls whether doorbell or port write packets are generated in response to events. 0 – Disable event messages 1 – Enable event messages

Soft Packet FIFO Transmit Control (IR_SP_TX_CTRL) CSR

Address: 0x107000

Description: This register is used to configure and control the transmission of packets using the soft packet FIFO.

Table 2-74. Soft Packet FIFO Transmit Control CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:15	Octets to Send	Read Write	Set to 16'd0	Writing a non-zero value (N) to this field arms the packet FIFO for packet transmission. The FIFO control logic will transmit the next N octets written to the Soft Packet FIFO Transmit Data register over the RapidIO link as a single packet.
16:31	reserved	Read Only	Set to 16'd0	

Soft Packet FIFO Transmit Status (IR_SP_TX_STAT) CSR

Address: 0x107004

Description:T his register is used to monitor the transmission of packets using the soft packet FIFO.

Table 2-75. Soft Packet FIFO Transmit Status CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:15	Octets Remaining	Read Only	Set to 16'd0	This field shows how many octets are still to be loaded in the current packet.
16:19	Buffers Filled	Read Only	Set to 4'd0	This field indicates how many complete packets are stored in the Tx FIFO.
20:26	reserved	Read Only	Set to 7'd0	
27	Full	Read Only	Set to 1`b0	This bit is set when the value of Buffers Filled equals the number of available transmission buffers.
28:31	Tx FIFO State	Read Only	Set to 4'd0	These bits display the state of the state machine that controls loading of packet data into the FIFO. The enumeration of states are as follows: 4`b0000 –Idle 4`b0001 –Armed 4`b0010 –Active All other states are reserved.

Soft Packet FIFO Transmit Data (IR_SP_TX_DATA) CSR

Address: 0x107008

Description: This register is used to write data to the soft packet FIFO.

Table 2-76. Soft Packet FIFO Transmit Data CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	Packet Data	Write Only	Set to 32'd0	This register is used to write packet data to the Tx FIFO. Reads of this register will return zero.

Soft Packet FIFO Receive Control (IR_SP_RX_CTRL) CSR

Address: 0x10700C

Description: This register is used to configure and control the reception of packets using the soft packet FIFO.

Table 2-77. Soft Packet FIFO Receive Control CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:30	reserved	Read Only	Set to 31'd0	
31	Overflow Mode	Read Write	Set to 1`b0	This bit controls the handling of received packets when the Rx FIFO is full. 1`b0 –The packet is discarded 1`b1 –The packet is not discarded

Soft Packet FIFO Receive Status (IR_SP_RX_STAT) CSR

Address: 0x107010

Description: This register is used to monitor the reception of packets using the soft packet FIFO.

Table 2-78. Soft Packet FIFO Receive Status CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:15	Octets Remaining	Read Only	Set to 16'd0	This field shows how many octets are head of line packet buffer in the Rx FIFO.
16:19	Buffers Filled	Read Only	Set to 4'd0	This field indicates how many complete packets are stored in the Rx FIFO.
20:26	reserved	Read Only	Set to 7'd0	
27	Full	Read Only	Set to 1`b0	This bit is set when the value of Buffers Filled equals the number of available reception buffers.
28:31	Rx FIFO state	Read Only	Set to 4`b0000	This field displays the current state of the RxFIFO: 4`b000 – Idle 4`b001 – Active All other states are reserved.

Soft Packet FIFO Receive Data (IR_SP_RX_DATA) CSR

Address: 0x107014

Description: This register is used to read data from the soft packet FIFO.

Table 2-79. Soft Packet FIFO Receive Data CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	Packet Data	Read Only	Set to 32'd0	This register is used to read packet data from the Rx FIFO.

Platform Independent PHY Control (IR_PI_PHY_CTRL) CSR

Address: 0x107020

Description: This register is used to control platform independent operating modes of the transceivers. These control bits are uniform across all platforms.

Table 2-80. Platform Independent PHY Control CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0	Tx Reset	Read Write	Set to 1`b1	This bit controls the state of the tx_init_n signal on the transceiver interface.
1	Rx Reset	Read Write	Set to 1`b1	This bit controls the state of the rx_init_n signal on the transceiver interface.
2:4	Loopback	Read Write	Set to 3`b000	These bits control the state of the loopback control vector on the transceiver interface. The loopback modes are enumerated as follows: 3`b000 – No Loopback 3`b001 – Near End PCS Loopback 3`b010 – Near End PMA Loopback 3`b011 – Far End PCS Loopback 3`b100 – Far End PMA Loopback
5:31	reserved	Read Only	Set to 27'd0	

Platform Independent PHY Status (IR_PI_PHY_STAT) CSR

Address: 0x107024

Description: This register is used to monitor platform independent operating status of the transceivers. These control bits are uniform across all platforms.

Table 2-81. Platform Independent PHY Status CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:21	reserved	Read Only	Set to 28'd0	
22:31	Initialization State	Read Only		These bits contain the state of the 1x/2x/Nx Initialization State Machine described in Section 4.12.4.8.1 of the RapidIO Interconnect Specification Part 6: Physical Layer LP-Serial Specification version 2.1. The mapping of states to the values encoded on this vector is shown in Table 2-82.

Table 2-82. LP-Serial Mode Initialization State Machine State Enumeration

IR_PI_PHY_STAT[22:31]	State
10`b0000000001	SILENT
10`b0000000010	SEEK
10`b0000000100	DISCOVERY
10`b0000001000	1X_MODE_LANE0
10`b0000010000	1X_MODE_LANE1
10`b0000100000	1X_MODE_LANE2
10`b0001000000	1X_RECOVERY
10`b0010000000	2X_MODE
10`b0100000000	2X_RECOVERY
10`b1000000000	NX_MODE

Platform Dependent PHY Control (IR_PD_PHY_CTRL) CSR

Address: 0x107028

Description: This register is used to control platform dependent operating modes of the transceivers. The functionality of these control bits is specific to each platform.

Table 2-83. Platform Dependent PHY Control CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	PD Control	Read write	Set to 32'd0	The state of these bits controls the state of the pd_ctrl vector on the transceiver interface.

Platform Dependent PHY Status (IR_PD_PHY_STAT) CSR

Address: 0x10702C

Description: This register is used to monitor platform dependent operating status of the transceivers. The functionality of these status bits is specific to each platform.

Table 2-84. Platform Dependent PHY Control CSR Bit-Fields

Bit Field	Name	Type	Reset State	Description
0:31	PD Status	Read Only	Set to 32'd0	The state of these bits reflects the state of the pd_stat vector on the transceiver interface.

Parameter Settings

The IPexpress™ tool is used to create all IP and architectural modules in the Diamond and ispLEVER software. For the Serial RapidIO IP core, numerous options are separated into multiple tabs. Table 3-1 provides the list of user-configurable parameters for the core the remainder of the section provides information on how implement the SRIO IP core parameters.

Table 3-1. IP Core Parameters

Parameter	Input Range	Default
Global		
LANE_SEL – Number of SERDES lanes	1, 2, 4	1
BAUD_RATE – Baud Rate	1.25, 2.5, 3.125	1.25
TxFLOW_CTL – Transmit flow control	Enable/Disable	Enabled
TxPRIORITY – Transmit priority method	Fixed/Rotating	Fixed
TIME_BASE – Time base ratio	0 - 255	64
EventMsgType	Port Write/Doorbell	Doorbell
Port General Control CSR		
HOST – Host/Slave endpoint type	Host/Salve	Slave
MASTER – Master/Respond only	Enable/Disable	Disable
Lane n Status 0 CSR		
PORT_ID – RapidIO Port identifier	0 - 15	0
Port 0 Control CSR		
OPORT_ENABLE – Output port enable	Enable/Disable	Disable
IPORT_ENABLE – Input port enable	Enable/Disable	Enable
Base Device ID CSR		
BASE_DEV_ID – Base device identifier	0x0-0xff	0x0
Assembly Identity and Information CARs		
ASSY_ID – Assembly identifier	0x0-0xffff	0x0
ASSY_VEND_ID – Assembly vendor identifier	0x0-0xffff	0x0
ASSY_REV_ID – Assembly revision identifier	0x0-0xffff	0x0
Processing Element Features CAR		
BRIDGE – General bridge identifier	No/Yes	No
Switch – RapidIO Switch/Bridge identifier	No/Yes	No
PE_ELEMENT – Processing element identifier	No/Yes	No
MEM_ELEMENT – Memory element identifier	No/Yes	No
LARGE_SYSTEMS – Large systems identifier	Enabled/Disable	Disable
ADDR_SUP – Address width supported identifier	34, 34 and 50, 34 and 66, 34 and 50 and 66	34
Source Operations CAR		
SO_READ – Source operations read capability	Yes/No	No
SO_WRITE - Source operations write capability	Yes/No	No
SO_SWRITE - Source operations swrite capability	Yes/No	No
SO_WRITE_RESP - Source operations swrite capability	Yes/No	No
SO_MESSAGE - Source operations messaging capability	Yes/No	No
SO_DOORBELL - Source operations doorbell capability	Yes/No	No

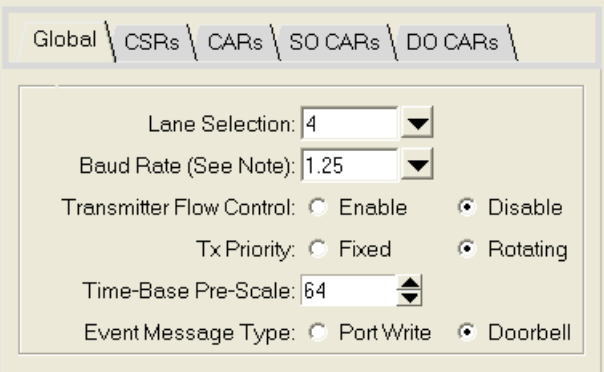
Table 3-1. IP Core Parameters (Continued)

Parameter	Input Range	Default
SO_ACMP_SWAP - Source operations atomic compare and swap capability	Yes/No	No
SO_ATEST_SWAP - Source operations atomic test and swap capability	Yes/No	No
SO_AINCR - Source operations atomic increment capability	Yes/No	No
SO_ADECR - Source operations atomic decrement capability	Yes/No	No
SO_ASET - Source operations atomic set capability	Yes/No	No
SO_ACLR - Source operations atomic clear capability	Yes/No	No
SO_ASWAP - Source operations atomic swap capability	Yes/No	No
SO_PORT_WRITE - Source operations port-write capability	Yes/No	No
Destination Operations CAR		
DO_READ – Destination operations read capability	Yes/No	No
DO_WRITE - Destination operations write capability	Yes/No	No
DO_SWRITE - Destination operations swrite capability	Yes/No	No
DO_WRITE_RESP - Destination operations swrite capability	Yes/No	No
DO_MESSAGE - Destination operations messaging capability	Yes/No	No
DO_DOORBELL - Destination operations doorbell capability	Yes/No	No
DO_ACMP_SWAP - Destination operations atomic compare and swap capability	Yes/No	No
DO_ATEST_SWAP - Destination operations atomic test and swap capability	Yes/No	No
DO_AINCR - Destination operations atomic increment capability	Yes/No	No
DO_ADECR - Destination operations atomic decrement capability	Yes/No	No
DO_ASET - Destination operations atomic set capability	Yes/No	No
DO_ACLR - Destination operations atomic clear capability	Yes/No	No
DO_ASWAP - Destination operations atomic swap capability	Yes/No	No
DO_PORT_WRITE - Destination operations port-write capability	Yes/No	No

The following sub-sections describe the various GUI tabs used to configure IP core operation to specific needs of the user applications. Note that only the Global tab contains selections that affect the behavior of the SRIO IP core. The remaining tabs provide mostly read-only register bit values and default register values for some selected read/write bits that advertise the configuration and capabilities of the SRIO IP core to the logical layer functions and to the SRIO system via maintenance transactions. Please refer to the Functional Description section RapidIO 2.1 LP-Serial Core Registers for complete descriptions of register contents.

Global Tab

Figure 3-1. Global Tab



Lane Selection

This option selects the number of SERDES lanes used on the SRIO link. It has an affect on the simulation evaluation capability only. Regardless of user configuration, the IP core itself that is generated supports all three link width combinations 1x, 2x, and 4x. See section titled “Setting Up the Core” for information on how to set up for x1 and x2 modes of operation external to the core.

Baud Rate

This option selects the SERDES line rate.

Transmitter Flow Control

This option selects whether Transmitter-Controlled flow control is enabled or disabled. When set to disabled, the system uses Receiver-Controlled flow as the default. See section “Functional Description” for details.

Transmit Priority

This option selects whether Fixed or Rotating priority is used. See “Functional Description” section for details.

Time-Base Pre-Scale

This option selects the time-base pre-scale value that is applied to the Management clock to obtain one microsecond and one millisecond time-base signals for internal core use. See section “Functional Description” for details.

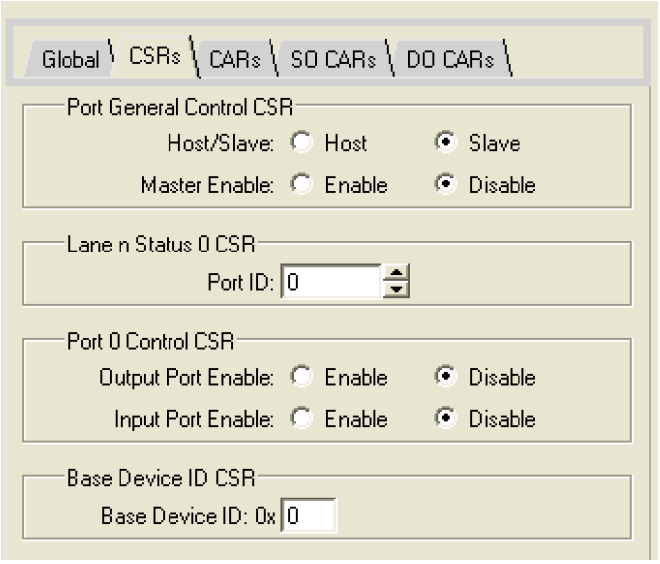
Event Message Type

This option selects the format for the generated event message. See section "Event Reporting".

CSRs Tab

The SRIO Configuration and Status Registers tab allows users to specify values for mostly read-only register bits and default values for some selected read/write bits. Please refer to the Functional Description section “RapidIO 2.1 LP-Serial Core Registers” for complete descriptions of register contents.

Figure 3-2. CSRs Tab



Host/Slave

This option specifies whether or not this SRIO endpoint is a Host device which is responsible for system exploration, initialization, and maintenance or an Agent/Slave device which is typically initialized by Host devices. This bit can be found in the Port General Control CSR.

Master Enable

This option selects whether or not a device is allowed to issue requests into the system. If the Master Enable is not set, the device may only respond to requests. This bit can be found in the Port General Control CSR.

SRIO Port ID

This option specifies the SRIO Port ID number of the device to which the lane/s of this core instance are assigned. These bits can be found in the Lane n Status 0 CSR.

Output Port Enable

This option specifies whether or not the port is stopped and not enabled to issue packets except to route or respond to I/O logical maintenance packets. Control symbols are not affected and are sent normally. This bit can be found in the Port 0 Control CSR.

Input Port Enable

This option specifies whether or not the port is stopped and only enabled to route or respond to I/O logical Maintenance packets. Other packets generate packet-not-accepted control symbols to force an error condition to be signaled by the sending device. Control symbols are not affected and are received and handled normally This bit can be found in the Port 0 Control CSR.

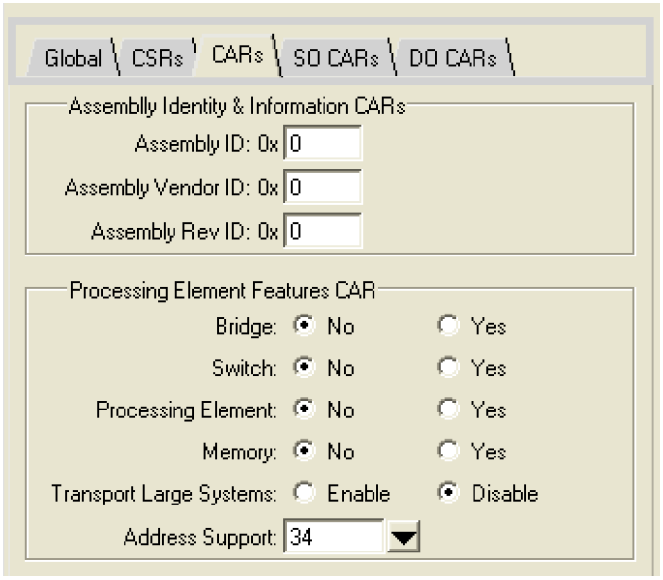
Base Device ID

This option specifies the base ID of the device in a small common transport system. This bit field can be found in the Base Device ID CSR.

CARs Tab

The SRIO Capability Registers tab allows users to specify values for read-only capability register bits. Please refer to the Functional Description section RapidIO 2.1 LP-Serial Core Registers for complete descriptions of register contents.

Figure 3-3. CARs Tab



Assembly ID

This option specifies the Assembly Identity field which is intended to uniquely identify the type of assembly from the vendor specified by the Assembly Vendor Identity field. The values for the Assembly Identity field are assigned and managed by the respective vendor. This bit field can be found in the Assembly Identity CAR.

Assembly Vendor ID

The option specifies the Assembly Vendor Identity field identifies the vendor that manufactured the assembly or subsystem containing the device. A value for the Assembly Vendor Identity field is uniquely assigned to a assembly vendor by the registration authority of the RapidIO Trade Association. This bit field can be found in the Assembly Identity CAR.

Assembly Rev ID

This option specifies the Assembly revision level managed by the respective vendor. This bit field can be found in the Assembly Information CAR.

Bridge

This option specifies whether or not the endpoint can bridge to another interface. Examples are PCI, proprietary processor buses, etc. This bit can be found in the Processing Elements Features CAR.

Switch

This option specifies whether or not the endpoint can bridge to another external RapidIO interface. This bit can be found in the Processing Elements Features CAR.

Processing Elements

This option specifies whether or not the endpoint physically contains a local processor or similar device that executes code. This bit can be found in the Processing Elements Features CAR.

Memory

This option specifies whether or not the endpoint has physically addressable local address space and can be accessed as an end point through non-maintenance (i.e. non-coherent read and write) operations. This local address space may be limited to local configuration registers, or could be on-chip SRAM. This bit field can be found in the Processing Elements Features CAR.

Transport Large Systems

This option specifies whether or not the endpoint supports common transport large systems (16b source and destination IDs). This bit can be found in the Processing Elements Features CAR.

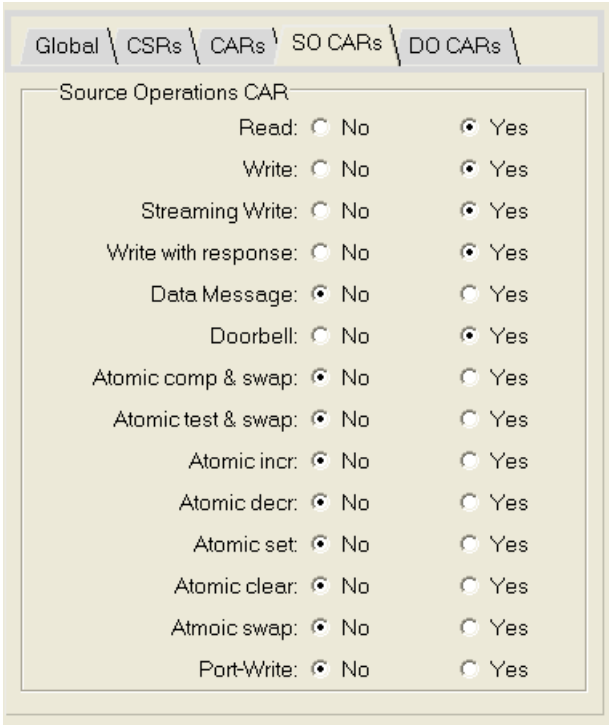
Address Support

This option specifies the number address bits supported by the PE both as a source and target of an operation. This bit field can be found in the Processing Elements Features CAR.

Source Operation CARs Tab

The SRIO Source Operations Capability Registers tab allows users to specify values for read-only capability register bits. Please refer to the Functional Description section RapidIO 2.1 LP-Serial Core Registers for complete descriptions of register contents. These bit fields can be found in the Source Operations CAR.

Figure 3-4. Source Operations Tab



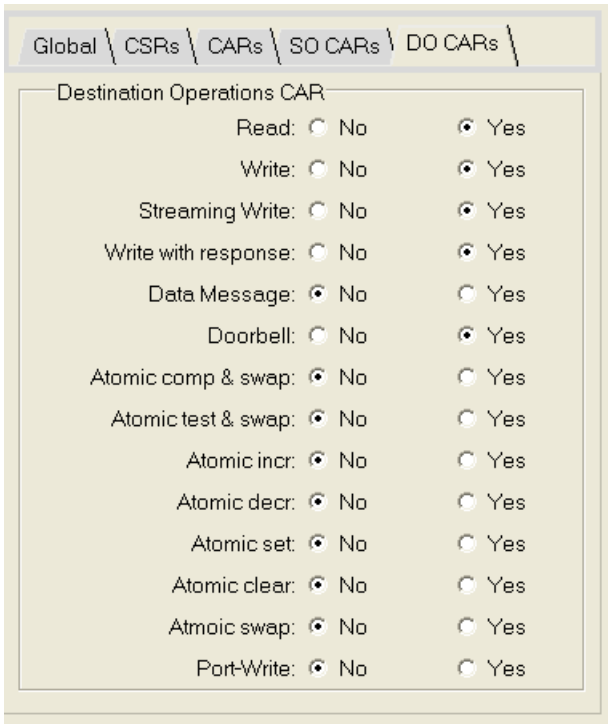
These options specify whether the endpoint can support:

- Read operations
- Write operations
- Streaming write operations
- Write with response operations
- Data message operations
- Doorbell operations
- Atomic compare and swap operations
- Atomic test and swap operations
- Atomic increment operations
- Atomic decrement operations
- Atomic set operations
- Atomic clear operations
- Atomic swap operations
- Port-write operations

Destination Operations CARs Tab

The SRIO Destination Operations Capability Registers tab allows users to specify values for read-only capability register bits. Please refer to the Functional Description section RapidIO 2.1 LP-Serial Core Registers for complete descriptions of register contents. These bit fields can be found in the Source Operations CAR

Figure 3-5. Destination Operations Tab



These options specify whether or not the endpoint can support:

- Read operations
- Write operations
- Streaming write operations
- Write with response operations
- Data Message operations
- Doorbell operations
- Atomic compare and swap operations
- Atomic test and swap operations
- Atomic increment operations
- Atomic decrement operations
- Atomic set operations
- Atomic clear operations
- Atomic swap operations
- Port-write operations

IP Core Generation

This chapter provides information on how to generate the Lattice SRIO IP core using the Diamond or ispLEVER software IPexpress tool, and how to include the core in a top-level design. The Lattice SRIO IP core can be used in the LatticeECP3 device family. An IP core and device-specific license is required to enable full use of the SRIO IP core in a complete, top-level design. Contact Lattice sales to obtain information on how to obtain the required licenses.

Licensing the IP Core

An IP core- and device-specific license is required to enable full, unrestricted use of the SRIO IP core in a complete, top-level design. Instructions on how to obtain licenses for Lattice IP cores are given at:

http://www.latticesemi.com/products/intellectualproperty/aboutip/isplevercoreonlinepurchas.cfm

Users may download and generate the SRIO IP core and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The SRIO IP core also supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited time (approximately four hours) without requiring an IP license. See "Hardware Evaluation" on page 98 for further details. However, a license is required to enable timing simulation, to open the design in the Diamond or ispLEVER EPIC tool, and to generate bitstreams that do not include the hardware evaluation timeout limitation.

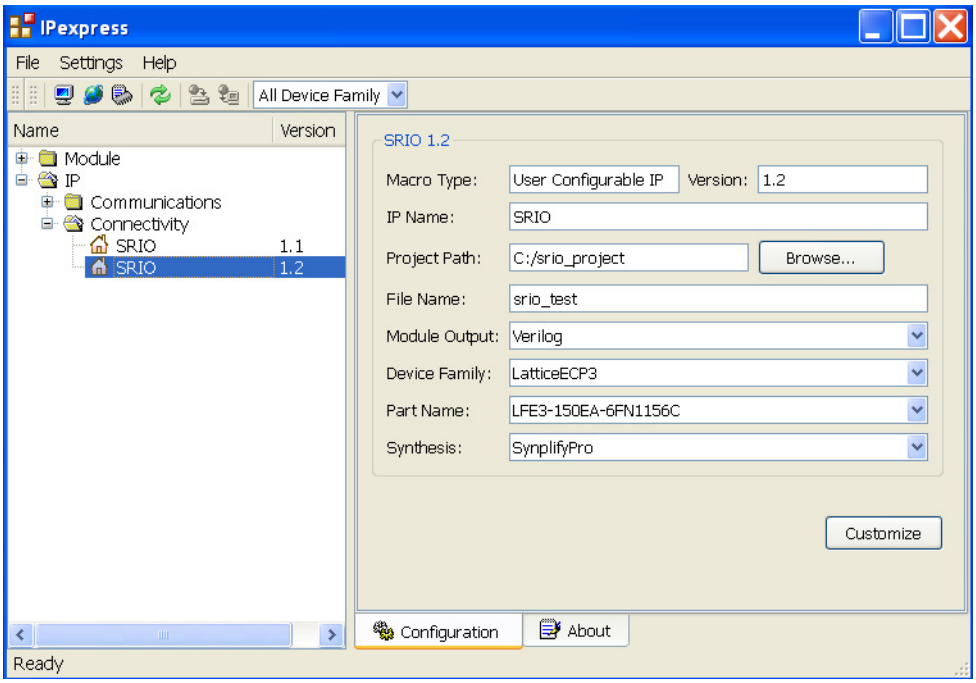
Getting Started

The SRIO IP core is available for download from the Lattice IP server using the IPexpress tool. The IP files are automatically installed using ispUPDATE technology in any customer specified directory. After the IP core has been installed, the IP core will be available in the IPexpress GUI dialog box shown in Figure 4-1.

The ispLEVER IPexpress GUI dialog box for the SRIO IP core is shown in Figure 4-1. To generate a specific IP core configuration the user specifies:

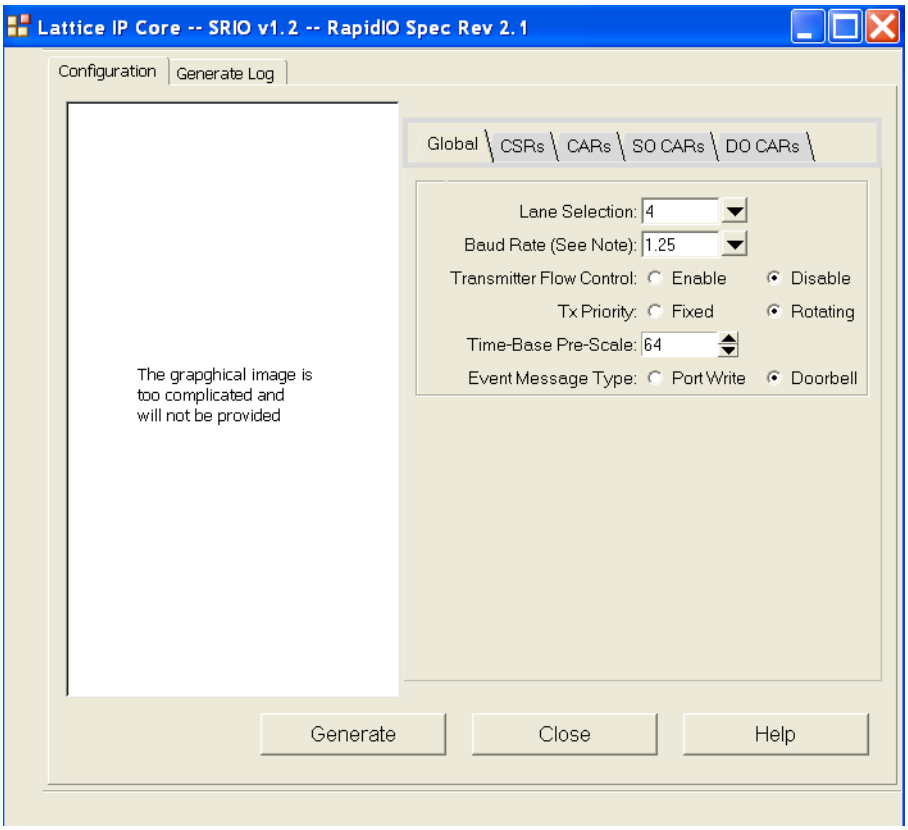
- **Project Path** – Path to the directory where the generated IP files will be loaded.
- **File Name** – "username" designation given to the generated IP core and corresponding folders and files.
- **(Diamond) Module Output** – Verilog or VHDL.
- **(ispLEVER) Design Entry Type** – Verilog HDL or VHDL.
- **Device Family** – Device family to which IP is to be targeted (e.g. LatticeECP3). Only families that support the particular IP core are listed.
- **Part Name** – Specific targeted part within the selected device family.

Figure 4-1. IPexpress Dialog Box (Diamond Version)



Note that if IPexpress is called from within an existing project, Project Path, Module Output (Design Entry in isp-LEVER), Device Family and Part Name default to the specified project parameters. Refer to the IPexpress online help for further information. To create a custom configuration, the user clicks the Customize button in the IPexpress Dialog Box to display the SRIO IP core Configuration GUI, as shown in Figure 4-2. From this dialog box, the user can select the IP parameter options specific to their application. Refer to section titled “Parameter Settings” for more information on the SRIO parameter settings.

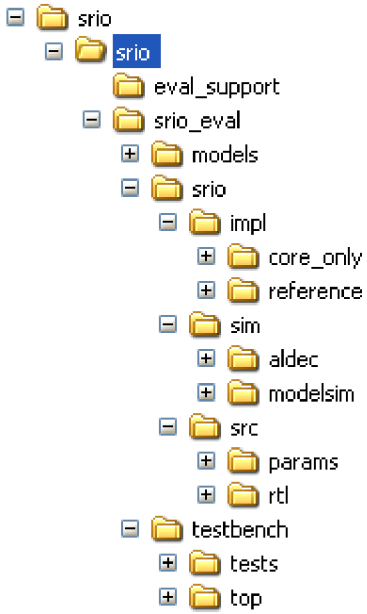
Figure 4-2. Configuration GUI (Diamond Version)



IPexpress-Created Files and Top Level Directory Structure

When the user clicks the Generate button in the IP Configuration Dialog Box, the IP core and supporting files are generated in the specified “Project Path” directory. The directory structure of the generated files is shown in Figure 4-3.

Figure 4-3. LatticeECP3 SRIO Core Directory Structure



The design flow for IP created with IPexpress uses a post-synthesized module (NGO) for synthesis and a protected model for simulation. The post-synthesized module is customized and created during IPexpress generation. The protected simulation model is not customized during IPexpress and relies on parameters provided to customize behavior during simulation.

Table 4-1 provides a list of key files and directories created by IPexpress and how they are used. IPexpress creates several files that are used throughout the design cycle. The names of most of the created files created are customized to the user's module name specified in IPexpress.

Table 4-1. IP Core Parameters

File	Sim	Synthesis	Description
<user name>_inst.v	Yes		This file provides a verilog instance template for the IP.
<user name>_inst.vhd	Yes		This file provides a vhdl instance template for the IP.
<user name>.v	Yes		This file provides the SRIO IP core wrapper for simulation. It instantiates and configures <user name>_beh.v below.
srio_core_beh.v	Yes		This file provides the simulation model for the SRIO IP core.
<user name>_bb.v		Yes	This file provides the synthesis black box for the user's synthesis.
<user name>.ngo		Yes	This file provides the GUI configured synthesized IP core.
<user name>.lpc			This file contains the IPexpress options used to recreate or modify the core in IPexpress.
<username>.ipx			The IPX file holds references to all of the elements of an IP or Module after it is generated from the IPexpress tool (Diamond version only). The file is used to bring in the appropriate files during the design implementation and analysis. It is also used to re-load parameter settings into the IP/Module generation GUI when an IP/Module is being re-generated.
<user name>_eval			This directory contains a sample reference design that utilizes the IP core. This design supports simulation and implementation using Diamond or isp-LEVER.
eval_support			This directory contains simulation models and .ngo support for the reference design that utilizes the IP core.

Most of the files required to use the SRIO IP core in a user's design reside in the root directory \<user_name> created by IPexpress. This includes the simulation model supporting simulation and a synthesis black box and .ngo file for implementing the design in Diamond or ispLEVER. The \<user_name> folder (SRIO in this example) contains files/folders with content specific to the <username> configuration. This directory is created by IPexpress each time the core is generated and regenerated each time the core with the same file name is regenerated. A separate \<username> directory is generated for cores with different names, e.g. \<my_core_0>, \<my_core_1>, etc.

The \<user_name>_eval (srio_eval in this example) and subtending directories provide files supporting SRIO IP core evaluation that includes both simulation and implementation via Diamond or ispLEVER. The \<user_name>_eval directory contains files/folders with content that is constant for all configurations of the SRIO IP core. The \srio_eval directory is created by IPexpress the first time the core is generated and updated each time the core is regenerated.

The implementation part of user evaluation supports both core only and reference design options using Synplify (Precesion supported in next release). Separate directories located at \<username>\srio_eval\<username>\impl\[core_only or reference] are provided for implementation (synthesis, map, place and route) using Diamond or ispLEVER and contain specific pre-built project files. Implementations performed in the core_only directory contain only the core and can therefore be used to show the size of the core. The reference directory implementation option reveals a much larger design because it contains additional example user logic functions. The models directory provides library elements such as the SERDES/PCS and supporting PLL moules.

The simulation part of user evaluation provides project files and test-cases supporting RTL simulation for both the Active-HDL and ModelSim simulators. Separate directories located at \<username>\srio_eval\<username>\sim\[Aldec or ModelSim]\rtl are provided and contain specific pre-built simulation project files. See section "Running Functional Simulation" below for more details.

Directory \<user_name>\eval_support provides simulation model and .ngo build support files for the reference design example user logic functions described above.

Instantiating the Core

The generated SRIO IP core package includes a black-box (<username>_bb.v) and an instance (<username>_inst.v/vhd) template that can be used to instantiate the core in a top-level design. Two example RTL top-level source files (srio_core_only_top.v and srio_reference_top.v) are provided and can also be used for instantiation template information of the IP core. These files can be found at \<username>\srio_eval\<username>\src\rtl\top\<device_family>. Users may also use the top-level reference version as the starting point for their top-level design.

Running Functional Simulation

Simulation support for the SRIO IP core is provided for Aldec and ModelSim simulators. The SRIO IP core simulation model is generated from IPexpress with the name <user name>.v. This file calls <user name>_beh.v which contains the obfuscated simulation model. An obfuscated simulation model is Lattice's unique IP protection technique which scrambles the Verilog HDL while maintaining logical equivalence. VHDL users will use the same Verilog model for simulation.

The functional simulation includes a configuration-specific behavioral model of the SRIO IP Core that is instantiated in an FPGA top level source file along with simulation support modules (see section "Simulation Strategies" below for details).

The top-level file supporting ModelSim eval simulation is provided in:

```
\<project_dir>\srio_eval\<username>\rtl\<device_family>\srio_reference_top.v.
```

This FPGA top is instantiated in an evaluation test-bench provided in:

```
\<project_dir>\srio_eval\testbench\top\<device_family>\tb.v.
```

It configures FPGA test logic registers and SRIO IP core control and status registers via an included test file provided in:

```
\<project_dir>\srio_eval\testbench\tests\<device_family>\testcase.v.
```

Note the user can edit the testcase.v file to configure and monitor registers as desired.

Users may run the eval simulation by doing the following:

1. Open ModelSim.
2. Under the File tab, select **Change Directory** and choose folder

```
\<project_dir>\srio_eval\<username>\sim\modelsim\rtl.
```

3. Under the Tools tab, select **Execute Macro** and execute one of the ModelSim “do” scripts shown.

The top-level file supporting Aldec Active-HDL simulation is provided in:

```
\<project_dir>\srio_eval\<username>\sim\aldec\rtl.
```

This FPGA top is instantiated in an eval testbench provided in:

```
\<project_dir>\srio_eval\testbench\top\<device_family>
```

that configures FPGA test logic registers and SRIO IP core control and status registers via an included test file provided in

```
\<project_dir>\srio_eval\testbench\tests\<device_family>\testcase.v.
```

Note the user can edit the testcase.v file to configure and monitor registers as desired.

Users may run the eval simulation by doing the following:

1. Open Active-HDL.
2. Under the Console tab change the directory to:

```
\<project_dir>\srio_eval\<username>\sim\aldec\rtl
```

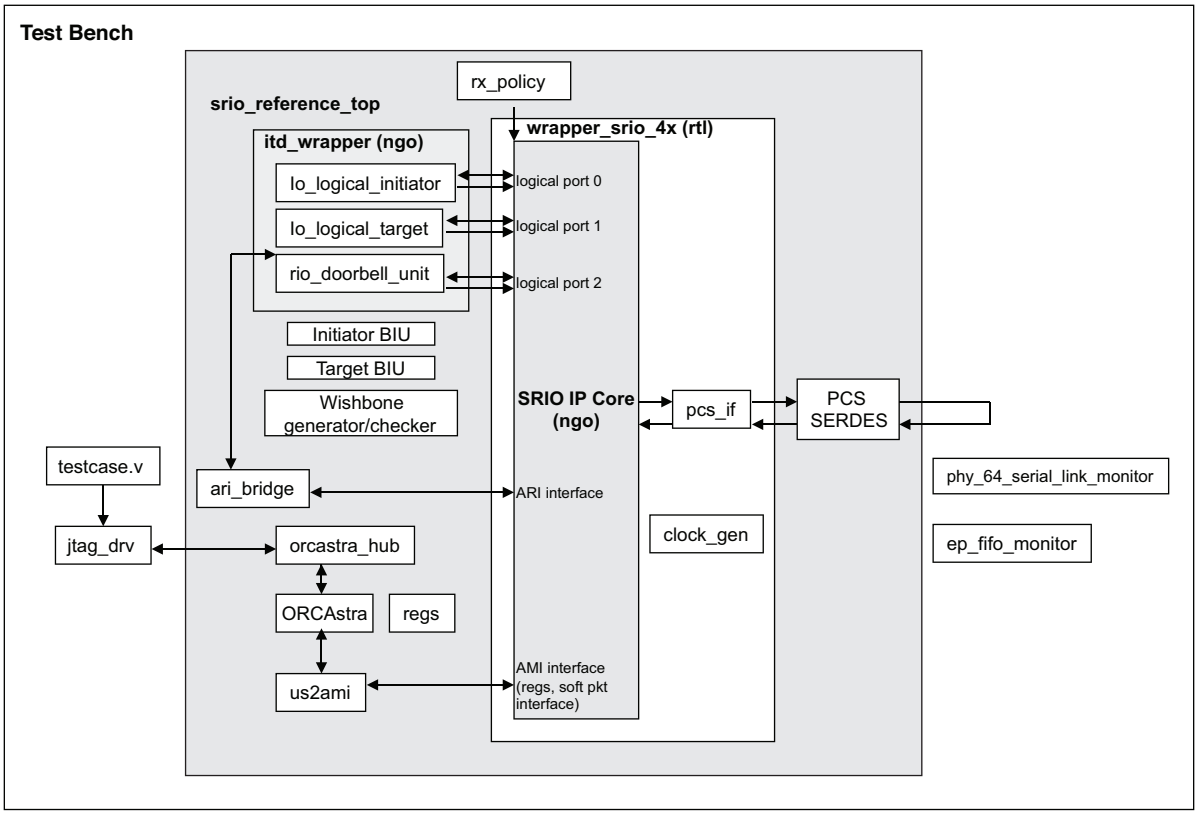
3. Execute the Active-HDL “do” scripts shown.

The simulation waveform results will be displayed in the Active-HDL Wave window.

Simulation Strategies

This section describes the simulation environment which demonstrates basic SRIO functionality. Figure 4-4 shows a block diagram of the simulation environment.

Figure 4-4. Simulation Environment Block Diagram



Simulation Environment Overview

The simulation environment is made up of various driver and monitor functions that are used to generate SRIO traffic and provide read/write access of the IP core for initialization and verification of proper operation. Driver/monitor functions connected to the Logical Layer interface ports (Initiator, Target, and Door-Bell Unit) generate SRIO read/write request/response transaction pairs that are transmitted by the SRIO IP core and looped at the transceiver interface back into the receiver. A receiver policy is used to route received request/response packets to the proper logical layer port.

Since the transceiver interface is connected in loopback, this single SRIO IP Core can be used to demonstrate both local operations such as reads and writes of local registers, and remote operations, such as maintenance reads and writes of far-end registers.

An ORCAstra module is used to provide a microprocessor interface to the SRIO IP core through the Alternate Management Interface (AMI). The AMI interface is used to:

- Access local Control and Status Registers (CSRs) through the Management Module
- Access alternate registers via the Alternate Register Interface (ARI)
- Generate SRIO read/write request/response pairs via the Soft Packet Interface (SPI) FIFO

The SPI can be used to perform maintenance transactions (reads and writes) of the far-end CSRs, and to remotely read and write the target memory.

Key Components

The simulation environment for the SRIO IP core consists of the following key components:

1. **Initiator (io_logical_layer_initiator)** – This module initiates write transactions that can be destined for the far-end IO_Logical_Layer_Target or Soft-Packet FIFO. It includes a programmable DMA unit that is controlled through register access of the Alternate Register interface. It also checks for the appropriate response based on the transaction type (response, non-response).

2. **Target (io_logical_layer_targer)** – This module provides a fully contained I/O logical layer target endpoint that contains a read/write memory. The Target converts I/O logical layer requests received from the RapidIO link into read or write cycles of the memory. Both the Initiator and the soft-packet interface FIFO have the ability to make requests of the target to first write the target's memory, and then read the target's memory to verify that both the write and read operations were successful. The Target generates the appropriate response packet for both read and write transaction types.

3. **Door-Bell (rio_doorbell_unit)** – This module generates door-bell messages and is controlled by writing registers through the ARI.

4. **jtag_drv , orcastra, orcastra_hub** – These modules provide a processor read/write interface via the JTAG port. The jtag_drv Verilog task provides a connection to the target device as well as to the display interface for user observations. It is supported by lower level tasks such as lword_read, lword_write, reg_vrfy, etc. In hardware lab testing the jtag_drv module is replaced with companion ORCAstra software running on a host PC. Orcastra_hub allows ORCAstra and Reveal to share the JTAG interface during hardware lab testing.

5. **Device Top (srio_reference_top.v)** – This is the device top level file. It can be synthesized, placed, routed, and loaded into a device for testing on hardware.

6. **Test-Bench Top (tb.v)** – This is the test-bench top-level Verilog source file used to interconnect simulation drivers to the target device and to provide system level clocks and resets.

7. **Testcase.v** – This is a sequence of verilog tasks used during simulation which initialize the SRIO core and perform various operations as examples of the core operation. These include reading and writing local CSRs, reading and writing remote CSRs, NWRITE and NREAD operations sourced from the soft packet interface to the memory in the io_logical_target. NWRITE operations sourced from the io_logical_initiator. Doorbells sourced from the rio_doorbell_unit.

8. **Simulation monitors** – ep_fifo_monitor.v, phy_64_serial_link_monitor.v: Monitors are included in the simulation environment to log the request/response transactions to log files. Packets which are sent and received on the SRIO are logged in the following files: port_rx_log.log and port_tx_log.log record each packet that appears on the SRIO link. port_rx_phy.log and port_tx_phy.log record record each symbol that appears on the SRIO link. The rx and tx log files are similar to each other since the tx port is directly looped back into the rx port. All write accesses to the soft-packet FIFO are recorded in the spf_fifo_monitor.log file. This file contains the simulation timestamp, address, and data of each SPI write operation.

Operational Overview

The JTAG simulation driver provided is an intelligent driver/monitor that is controlled through separate Verilog script files which direct the driver to perform macro level functions such as read a register, initialize the core, send an NWRITE_R I/O operation, etc. The drivers manage all of the low level functions related to the processor interface autonomously. Users can easily add to the script files to provide additional testing in interested areas.

In this simulation environment the SRIO transmit link is looped back to the receive link, so the same core contains both the local and remote CSRs. Physically there is only one set of CSRs. The difference between local and remote accesses is how the access is performed. Local CSRs are accessed directly through the AMI interface. Example Verilog tasks for accessing local CSRs are csr_rd, csr_wr, and csr_rd_vrfy.

Remote CSR accesses utilize the soft packet interface FIFO to send a read/write request to the far end. The SPI is accessed via the AMI. A Verilog task creates a maintenance read or write packet and writes it into the SPI. Once the complete packet is written into the SPI the packet is sent on the SRIO transmit link. At the far-end (SRIO rx port in this simulation environment) the packet is received and the Rx Policy directs the packet to the Management Mod-

ule. The desired CSR is read/written, and any necessary response is returned to the local SPI, where it is read via the AMI.

Synthesizing and Implementing the Core in a Top-Level Design

The SRIO IP core itself is synthesized and provided in NGO format when the core is generated through IPexpress. Users may use the core in their own top-level design by instantiating the core in their top-level as described previously and then synthesizing the entire design with either Synplify or Precision (future release) RTL Synthesis.

The top-level file srio_core_only_top.v provided in \<project_dir>\srio_eval\<username>\src\rtl\top\<device_family> supports the ability to implement the SRIO IP Express core in isolation. Push-button implementation of this top-level design with either Synplify® or Precision® (future release) RTL Synthesis is supported via the Diamond or ispLEVER project files srio_core_only_eval.syn located in the \<project_dir>\srio_eval\<username>\impl\core_only\ directory.

To use this project file in Diamond:

1. Choose **File > Open > Project**.
2. Browse to \<project_dir>\srio_eval\<username>\impl\(synplify or precision) in the Open Project dialog box.
3. Select and open <username>.ldf. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Select the Process tab in the left-hand GUI window.
5. Implement the complete design via the standard Diamond GUI flow.

To use this project file in ispLEVER:

1. Choose **File > Open Project**.
2. Browse to \<project_dir>\srio_eval\<username>\impl\(synplify or precision) in the Open Project dialog box.
3. Select and open <username>.syn. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Select the device top-level entry in the left-hand GUI window.
5. Implement the complete design via the standard ispLEVER GUI flow.

Setting Up the Core

This section describes how to set up the SRIO IP core for various link width combinations.

Overview

The SRIO IP core is capable of operating in 1x, 2x, and 4x link width combinations. The internal link state machines provide a degree of LP-Serial link auto-negotiation capability that ensures that any connected combination of 1x, 2x, and 4x ports that are configured in the same manner (buad rate, etc.) shall find a link width over which they can communicate.

How To Set Up for X1, X2, X4 Operation

The SRIO IP core provided will support all three link width combinations regardless of user GUI configuration. The example case provided in the reference evaluation contains connections from the SERDES/PCS to the PCS interface module required for x4 operation. Four 16-bit SERDES channels are used. In order to support fixed x1 or x2 modes of operation, the unused input connections/channels of the PCS interfaces module (PCS_if.v) need to be disconnected and forced to logic 0.

Setting Design Constraints

There are several design constraints that are required for the IP core. These constraints must be placed as preferences in the .lpf file located in the Diamond or ispLEVER project directory. These preferences can be entered in the .lpf file through the Preference Editing View in Diamond, the Design Planner in ispLEVER, or directly in the text based .lpf file. The preferences listed below are the key preferences required to achieve advertised performance. It is recommended that the reference design .lpf file be consulted for a complete and up to date list of preferences.

Frequency Preferences:

```
FREQUENCY NET "rio_clk" 125 MHz PAR_ADJ 10 ;
FREQUENCY NET "tx_full_clk" 312.5 MHz ;
FREQUENCY NET "pcs_if_clk" 156.25 MHz ;
FREQUENCY NET "rx_half_clk_ch0" 156.25 MHz ;
FREQUENCY NET "rx_half_clk_ch1" 156.25 MHz ;
FREQUENCY NET "rx_half_clk_ch2" 156.25 MHz ;
FREQUENCY NET "wrapper_srio_4x/recover_clk" 156.25 MHz ;
FREQUENCY NET "mgt_clk" 65 MHz ;
FREQUENCY NET "refck2core" 125 MHz ;
```

Use Preferences:

```
USE SECONDARY NET "rx_half_clk_ch0";
USE SECONDARY NET "rx_half_clk_ch1";
USE SECONDARY NET "rx_half_clk_ch2";
USE PRIMARY NET "wrapper_srio_4x/recover_clk" QUADRANT_BL; or QUADRANT_BR
USE PRIMARY NET "pcs_if_clk" ; QUADRANT_BL QUADRANT_BR; or only one QUARDRANT
USE PRIMARY NET "rio_clk";
```

Block/Multicycle Preferences:

```
BLOCK PATH FROM CLKNET "rx_half_clk_ch0" TO CLKNET "wrapper_srio_4x/recover_clk" ;
BLOCK PATH FROM CLKNET "rx_half_clk_ch1" TO CLKNET "wrapper_srio_4x/recover_clk" ;
BLOCK PATH FROM CLKNET "rx_half_clk_ch2" TO CLKNET "wrapper_srio_4x/recover_clk" ;
BLOCK PATH FROM CLKNET "pcs_if_clk" TO CLKNET "wrapper_srio_4x/recover_clk" ;
BLOCK PATH FROM CLKNET "wrapper_srio_4x/recover_clk" TO CLKNET "pcs_if_clk" ;
BLOCK PATH FROM CLKNET "wrapper_srio_4x/recover_clk" TO CLKNET "mgt_clk" ;
BLOCK PATH FROM CLKNET "mgt_clk" TO CLKNET "refck2core" ;
BLOCK PATH FROM CLKNET "refck2core" TO CLKNET "mgt_clk" ;
BLOCK NET "lnk_clk_reset_n";
BLOCK JTAGPATHS ;
BLOCK    PATH    FROM    CELL    "*oplm2_mgt_1x_2x_nx_init_sm?init_state*"    TO    CELL
"*oplm2_mgt_1x_2x_nx_tx_int*";
```

```
BLOCK NET "switch*";
BLOCK PATH FROM CLKNET "sci_rd" TO CLKNET "mgt_clk" ;
MULTICYCLE FROM CLKNET "pcs_if_clk" TO CLKNET "rio_clk" 1.000000 X_SOURCE ;
MULTICYCLE FROM CLKNET "rio_clk" TO CLKNET "pcs_if_clk" 1.000000 X_DEST ;
MAXDELAY FROM CELL "*wrapper*ollm_serial_tx_link_sched*local_reset_n" 16ns;
MULTICYCLE FROM CELL "*wrapper_srio_4x*rio_maintenance_module*bus_interface*mgt_a*"
2.000000 X ;
MULTICYCLE FROM ASIC "wrapper_srio_4x/wrapper/core/ollm/ollm_64_ack_fifo/tx_fifo_core/
mem_data_mem_data_0_0" PIN "DO*" 4x;
MULTICYCLE FROM ASIC "wrapper_srio_4x/wrap-
per/core/ollm/ollm_64_ack_fifo/tx_fifo_core/mem_data_mem_data_0_0" PIN "DO*" TO CELL
"*ollm_tx_error_mgmt*" 1x;
MULTICYCLE FROM CELL "*ollm_serial_rx_sym_dec*sym_crc_err_n" TO CELL "*ollm_rx_packet_sm*"
2.000000 X ;
```

Errors and Warnings

The following warnings will be present during the map stage of implementation for both core_only and the reference design implementation examples and should be ignored. They have to do with RAM initialization which is not required for FIFO applications.

WARNING - map: The reset of EBR
'wrapper/core/rio_ep_mgmt_entity_rio_ep_packet_fifo/spf_rx_fifo_size_cou
nt_fifo_mem_data_mem_data_0_0' cannot be controlled. The local reset
is not connected to any control signal and set to GND. The global reset
is disabled via GSR property. To control the EBR reset, either connect
the local reset to a control signal or force the GSR property to be
enabled.

The following warnings will be present during the map stage of implementation for the reference design implementation example and should be ignored. These warnings have to do with the JTAG port which has dedicated I/O buffers and do not require explicit instantiation.

WARNING - map: I/O buffer missing for top level port tdi...logic will be discarded.

WARNING - map: I/O buffer missing for top level port tms...logic will be discarded.

WARNING - map: I/O buffer missing for top level port tck...logic will be discarded.

Hardware Evaluation

The SRIO IP core supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of IP cores that operate in hardware for a limited period of time (approximately four hours) without requiring the purchase on an IP license. It may also be used to evaluate the core in hardware in user-defined designs.

Enabling Hardware Evaluation in Diamond

Choose **Project > Active Strategy > Translate Design Settings**. The hardware evaluation capability may be enabled/disabled in the Strategy dialog box. It is enabled by default.

Enabling Hardware Evaluation in ispLEVER

In the **Processes for Current Source** pane, right-click the **Build Database** process and choose **Properties** from the dropdown menu. The hardware evaluation capability may be enabled/disabled in the Properties dialog box. It is enabled by default.

Updating/Regenerating the IP Core

By regenerating an IP core with the IPexpress tool, you can modify any of its settings including: device type, design entry method, and any of the options specific to the IP core. Regenerating can be done to modify an existing IP core or to create a new but similar one.

Regenerating an IP Core in Diamond

To regenerate an IP core in Diamond:

1. In IPexpress, click the **Regenerate** button.
2. In the **Regenerate** view of IPexpress, choose the IPX source file of the module or IP you wish to regenerate.
3. IPexpress shows the current settings for the module or IP in the Source box. Make your new settings in the Target box.
4. If you want to generate a new set of files in a new location, set the new location in the IPX Target File box. The base of the file name will be the base of all the new file names. The IPX Target File must end with an .ipx extension.
5. Click **Regenerate**. The module's dialog box opens showing the current option settings.
6. In the dialog box, choose the desired options. To get information about the options, click **Help**. Also, check the **About** tab in IPexpress for links to technical notes and user guides. IP may come with additional information. As the options change, the schematic diagram of the module changes to show the I/O and the device resources the module will need.
7. To import the module into your project, if it's not already there, select **Import IPX to Diamond Project** (not available in stand-alone mode).
8. Click **Generate**.
9. Check the **Generate Log** tab to check for warnings and error messages.
10. Click **Close**.

The IPexpress package file (.ipx) supported by Diamond holds references to all of the elements of the generated IP core required to support simulation, synthesis and implementation. The IP core may be included in a user's design by importing the .ipx file to the associated Diamond project. To change the option settings of a module or IP that is already in a design project, double-click the module's .ipx file in the File List view. This opens IPexpress and the module's dialog box showing the current option settings. Then go to step 6 above.

Regenerating an IP Core in ispLEVER

To regenerate an IP core in ispLEVER:

1. In the IPexpress tool, choose **Tools > Regenerate IP/Module**.
2. In the **Select a Parameter File** dialog box, choose the Lattice Parameter Configuration (.lpc) file of the IP core you wish to regenerate, and click **Open**.
3. The **Select Target Core Version, Design Entry, and Device** dialog box shows the current settings for the IP core in the **Source Value** box. Make your new settings in the **Target Value** box.
4. If you want to generate a new set of files in a new location, set the location in the **LPC Target File** box. The base of the .lpc file name will be the base of all the new file names. The LPC Target File must end with an .lpc extension.
5. Click **Next**. The IP core's dialog box opens showing the current option settings.

6. In the dialog box, choose desired options. To get information about the options, click **Help**. Also, check the **About** tab in the IPexpress tool for links to technical notes and user guides. The IP core might come with additional information. As the options change, the schematic diagram of the IP core changes to show the I/O and the device resources the IP core will need.
7. Click **Generate**.
8. Click the **Generate Log** tab to check for warnings and error messages.

SERDES/PCS

PCS Configuration

The “pcs_serdes.lpc” configuration file used to generate the SERDES/PCS module from IPexpress is provided in the IP evaluation package making further user-customization possible without starting from scratch. The base design operates the SERDES/PCS module at 1.25G line rate and requires a 125 MHz reference clock. Table 5-1 below shows some of the settings used.

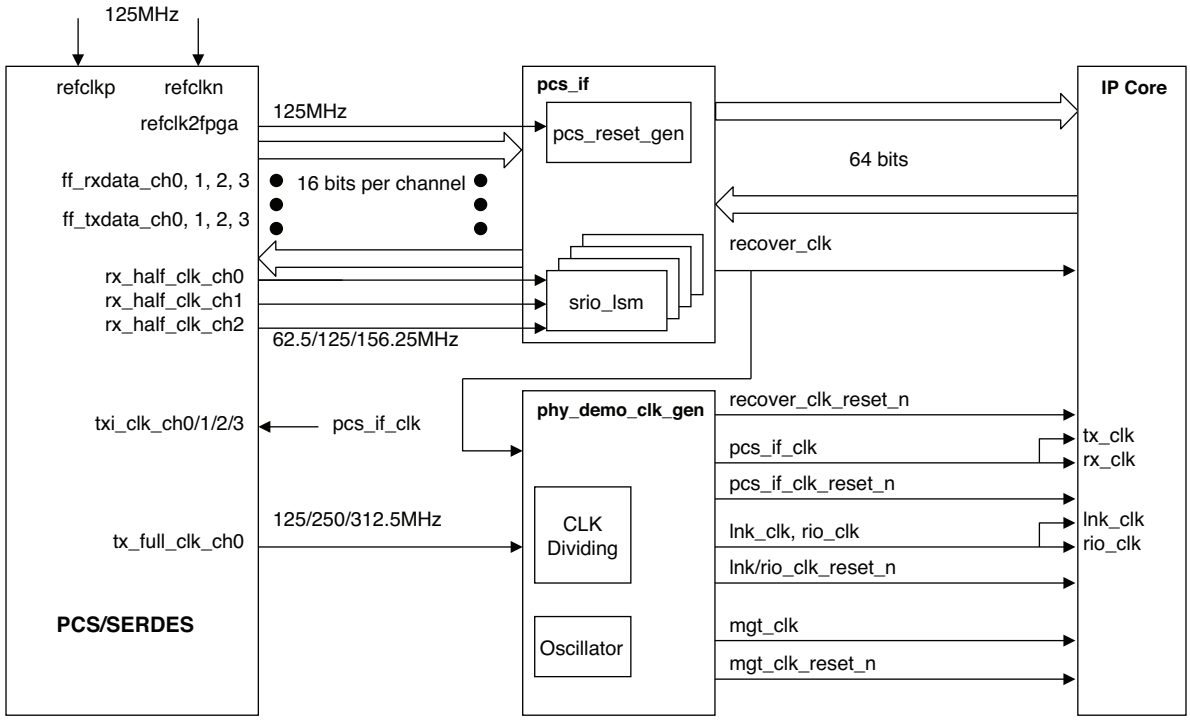
Table 5-1. SRIO SERDES/PCS Configuration

Bit Rate		1.25G	2.5G	3.125G
Protocol		SRIO	SRIO	SRIO
Reference Clock Multiplier		10X	20X	25X
refclk		125	125	125
16-Bit Interface	ff_rxhalfclk_ch[0:3]	62.5	125	156.25
	ff_txfullclk_ch0	125	250	312.5
RX PCS	Word Aligner	On	On	On
	8b/10b Decoder	On	On	On
	FPGA Bridge FIFO	On	On	On
	CTC	Bypass	Bypass	Bypass
	LSM	Bypass	Bypass	Bypass
TX PCS	8b/10b Encoder	On	On	On
	FPGA Bridge FIFO	On	On	On

PCS Connections

In the SRIO evaluation reference design, the SERDES/PCS is configured with a 16-bit interface on a per channel basis. The PCS fabric interface clock scheme used follows “Case II_b” described in TN1176, LatticeECP3 SERDES/PCS Usage Guide. Figure 5-1 provides a block diagram of the connections between the SERDES/PCS module, the PCS_if.v module, and the SRIO IP core.

Figure 5-1. SRIO SERDES/PCS Connections Block Diagram



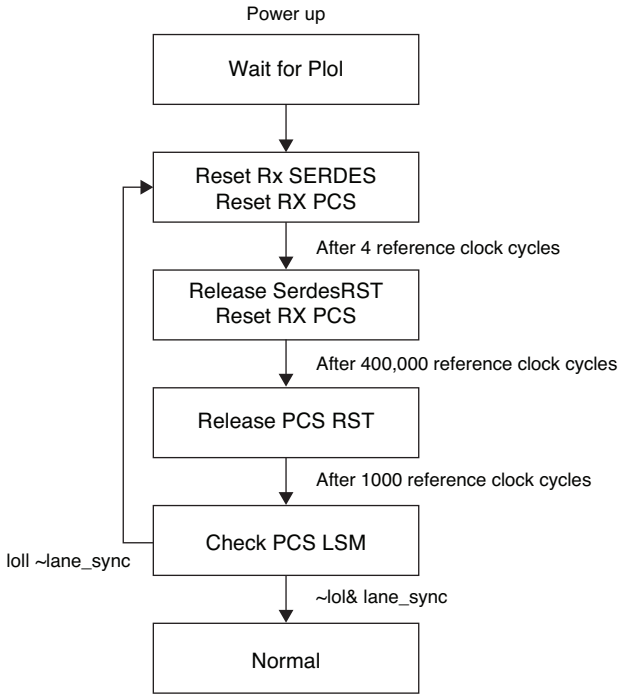
pcs_if module – This module performs the bridging function between hard SERDES/PCS module and IP core. It contains the PCS SRIO link state machines on a per channel basis, and generates the proper reset sequences for the SERDES/PCS block. The PCS side of the interface operates over a single 16-bit data interface, two 16-bit data interfaces, or four 16-bit data interfaces depending upon x1, x2, or x4 mode. The core side of the interface operates over a fixed 64-bit data path. The data path and LSMs are clocked with the recover clock on a per-channel basis.

phy_demo_clk_gen module – This module passes though the tx_full_clk_ch0 to the generate lnk_clk and rio_clk inputs of the core, generates the mgt_clk, and generates the resets for the core. For reset generation, a delay counter is used to delay the release of tx pcs lane reset from pcs_if module and to release the resets on each of the following clocks: tx_halfclk_ch0 (tx_rst_n and rx_rst_n), rio_clk (rio_clk_rst_n), lnk_clk (lnk_clk_rst_n), and mgt_clk (mgt_clk_rst_n) after the tx_fullclk_ch0 and tx_halfclk_ch0 are stable.

SERDES/PCS Initialization

For reset initialization, follow the procedures described in the SERDES/PCS Reset section of TN1176, LatticeECP3 SERDES/PCS Usage Guide. It covers the basic requirements for the reset functionality of SERDES/PCS. The transmitter reset initialization follows these procedures exactly. For the receiver, there is slight difference in that the reference design does not check rlos from SERDES/PCS. Instead, the reference design includes the protocol level CDR lock status check circuit. The modified procedure is shown below.

Figure 5-2. PCS/SERDES RX Reset Generation State Machine



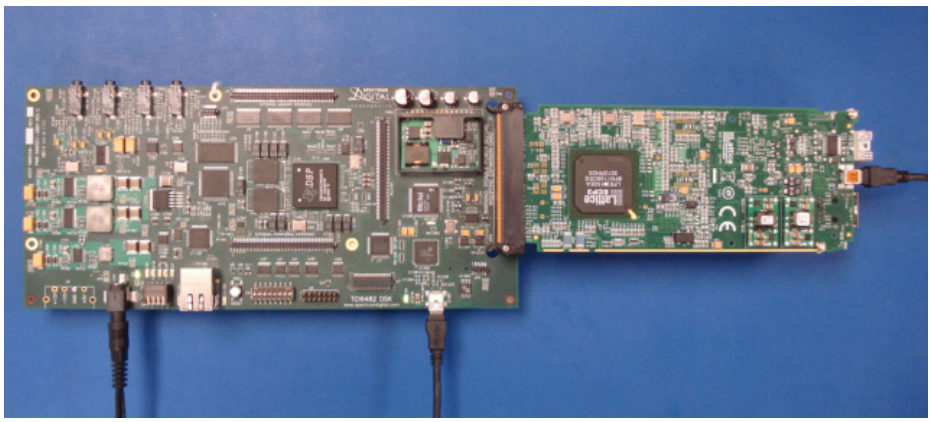
Core Verification

Serial RapidIO (SRIO) is the interface of choice for Texas Instruments (TI) DSP families, which are widely used in the wireless world. Lattice has developed an interoperability testing platform for the Lattice SRIO IP core and the TI 6482 DSP. The platform was designed to support the uTCA form factor, which simplifies connectivity and increases the flexibility of the testing platform. Additionally, the LatticeMico32™ soft processor is utilized in the design to initiate transactions between the SRIO core and the TI DSP.

The test platform shown in Figure 6-1 consists of a Spectrum Digital TCI6482 DSK board containing the DSP, and a LatticeECP3™ AMC card plugged into the AMC connector on the DSK. A LatticeMico32 processor in the FPGA is used to initiate transactions from the SRIO IP core with the DSP acting as a target. Testing involves the following:

1. Identifying and enumerating the DSP using RapidIO maintenance read and write transactions.
2. Performing DMA transfers to the DSP's L2 memory using RapidIO NWRITE, SWRITE, and NWRITE_R transactions. The success of these transactions was verified by the LatticeMico32 processor using NREAD transactions.

Figure 6-1. SRIO Interoperability Testing Platform



Support Resources

Lattice Technical Support

There are a number of ways to receive technical support as listed below.

Online Forums

The first place to look is Lattice Forums (www.latticesemi.com/support/forums.cfm). Lattice Forums contain a wealth of knowledge and are actively monitored by Lattice Applications Engineers.

Telephone Support Hotline

Receive direct technical support for all Lattice products by calling Lattice Applications from 5:30 a.m. to 6 p.m. Pacific Time.

- For USA and Canada: 1-800-LATTICE (528-8423)
- For other locations: +1 503 268 8001

In Asia, call Lattice Applications from 8:30 a.m. to 5:30 p.m. Beijing Time (CST), +0800 UTC. Chinese and English language only.

- For Asia: +86 21 52989090

E-mail Support

- techsupport@latticesemi.com
- techsupport-asia@latticesemi.com

Local Support

Contact your nearest Lattice sales office.

Internet

www.latticesemi.com

References

- DS1021, LatticeECP3 Family Data Sheet
- TN1176, LatticeECP3 SERDES/PCS Usage Guide
- RapidIO 2.1 Specification: www.rapidio.org/specs/current

Revision History

Date	Document Version	IP Core Version	Change Summary
March 2010	01.0	0.1	Initial release.
September 2010	01.1	1.1	Added support for Diamond software throughout.
			Added integrated support for x4 @ 2.5G. Added more flexible clocking scheme for x1, x2 modes supporting link rates up to 3.125G.
April 2011	01.2	1.2	Added support for x4 @3.125G.
June 2011	01.3	1.2	Updated "Clocks, Resets and Miscellaneous Signal Descriptions" on page 30.

Resource Utilization

LatticeECP3 Utilization

Table A-1 lists resource utilization for LatticeECP3 FPGAs using the SRIO IP core.

Table A-1. Resource Utilization[1]

IPexpress User-Configuration	Slices	LUTs	Registers	sysMEM EBRs	f_{MAX} (MHz)
All Configurations	10,539	15,749	10,199	16	156.25[2]

1. Performance and utilization data are generated using an LFE3-70EA-8FN672CES device with Lattice Diamond 1.2 and Synplify Pro E-2010.09L-SP2 software. Performance may vary when using a different software version or targeting a different device density or speed gradewithin the LatticeECP3 family.
2. f_{MAX} shown is for x4 operation at 3.125Gbaud using -8 speed grade devices.

Ordering Part Number

The Ordering Part Number (OPN) for the SRIO IP core targeting LatticeECP3 devices is SRIO-E3-U1.

Configuration and Licensing

IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the Diamond and ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and Diamond or ispLEVER help system. For more information on the Diamond or ispLEVER design tools, visit the Lattice web site at: www.latticesemi.com/software.