Virtex-5 Integrated Endpoint Block for PCI Express Designs

User Guide

UG197 (v1.3) June 2, 2008

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/06/06	1.0	Initial Xilinx release on CD.
03/20/07	1.1	Moved "Tx and Rx Buffer Layout" and "Buffer Latency" from Chapter 2 to Appendix A. Renamed Chapter 3 to "Designing with LogiCORE IP for the Endpoint Block"and replaced content. Split Error Reporting table into Table 4-3 (PCIe Block action) and Table 4-4 (User action). Added VHDL code examples to "Simulating in VHDL" in Chapter 5.
12/13/07	1.2	Revised L0PWRTURNOFFREQ description in Table 2-15, page 42 and added a footnote tied to power state D3. Clarified request types when crossing a 4 KB boundary in Table 4-2, page 71. Replaced Chapter 3, "Designing with LogiCORE IP for the Endpoint Block." Addition of "Known Restrictions," page 79.

Date	Version	Revision
06/02/08	1.3	Updated "TX Transmission Issues Due to Lack of Data Credits," page 79 including workaround. Added "Lane Reversal," page 78. Fixed LLKRXDSTREQN in "Invalid Cycles in LLKRXPREFERREDTYPE Signal," page 82. Updated "Credit Leak When Transmitting Completion TLPs," page 87. Added"Receipt of Back-to-Back ACK DLLPs," page 88.

Table of Contents

About This Guide

This guide serves as a technical reference describing the Virtex®-5 Integrated Endpoint implementation for PCI Express® designs.

Guide Contents

This guide contains the following chapters:

- Chapter 1, "Virtex-5 Integrated Endpoint Block Overview,"provides a brief introduction to the Endpoint block embedded in the Virtex-5 devices.
- Chapter 2, "Integrated Endpoint Block Functionality," gives an architectural overview of the block and detailed descriptions of each block interface.
- Chapter 3, "Designing with LogiCORE IP for the Endpoint Block," provides more information on using the CORE Generator™ GUI to generate the appropriate LogiCORE IP to implement the Virtex-5 Endpoint block in a PCI Express design.
- Chapter 4, "Designing with the Endpoint Block," provides in-depth information on various design considerations.
- Chapter 5, "Simulating with the Endpoint Block," introduces simulating with the Virtex-5 Endpoint block.
- Appendix A, "Endpoint Block Attributes," provides detailed information on the attributes that can be set on the Endpoint block. Because these attributes are all set through the CORE Generator GUI, Appendix A is provided as a reference.
- "Glossary," defines various terms used in this document.

Additional Support Resources

To search the database of silicon and software questions and answers or to create a technical support case in WebCase, see the Xilinx website at:

http://www.xilinx.com/support.

Typographical Conventions

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

Convention	Meaning or Use	Example
Italic font	References to other documents	See the *Virtex-5 Configuration Guide* for more information.
	Emphasis in text	The address (F) is asserted *after* clock event 2.
Underlined Text	Indicates a link to a web page.	http://www.xilinx.com/virtex5

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Support Resources" for details. Refer to "The PCI Express Standard" in Chapter 1 for details.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the *Virtex-5 Data Sheet*.
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest documentation.

Chapter 1

Virtex-5 Integrated Endpoint Block Overview

Summary

This chapter introduces the Integrated Endpoint block embedded in Virtex-5 devices. The sections include:

- "The PCI Express Standard"
- "The Virtex-5 Integrated Endpoint Block for PCI Express Designs"
- "Memory Requirements"
- "Use Models"

The PCI Express Standard

The PCI Express (PCIe®) standard is a next-generation evolution of the older PCI™ and PCI-X™ parallel bus standards. It is a high-performance, general-purpose interconnect architecture, designed for a wide range of computing and communications platforms. It is a packet-based, point-to-point serial interface that is backward compatible with PCI and PCI-X configurations, device drivers, and application software. Its faster, serial-bus architecture with dedicated, bidirectional I/O represents a fresh architectural approach. Table 1-1 shows the bandwidth for various lane configurations. The effective bandwidth is lower than the raw bandwidth due to the overhead of the 8B/10B encoding and decoding used by the protocol.

Table 1-1: **PCIe Bandwidth**

Link	Raw Bandwidth per Direction	Effective Bandwidth per Direction
x1	2.5 Gb/s	2 Gb/s
x2	5 Gb/s	4 Gb/s
x4	10 Gb/s	8 Gb/s
x8	20 Gb/s	16 Gb/s

The Virtex-5 Integrated Endpoint Block for PCI Express Designs

The Virtex-5 Integrated Endpoint block contains the functionality defined in the specifications maintained by the PCI-SIG (www.pcisig.com):

- Compliant with the *PCI Express Base 1.1 Specification*
- Endpoint block or Legacy Endpoint block for PCI Express designs
- x8, x4, x2, or x1 lane width
- RocketIO™ GTP transceivers implement a fully compliant PHY
- Block RAMs used for buffering
- Fully buffered Transmit and Receive
- Management interface to access configuration space and internal configuration
- Full range of maximum payload size (128 to 4096 bytes) supported
- Up to two virtual channels (VCs)
- Round robin, weighted round robin, or strict priority VC arbitration
- Up to 6 x 32 bit or 3 x 64 bit BARs (or a combination of 32 bit and 64 bit)
- BARs configurable for memory or I/O
- One function
- Signals to the fabric for statistics and monitoring

The Endpoint block is configurable by using a combination of attributes and port tieoffs, as part of the standard FPGA configuration. Configuration uses the LogiCORE™ GUI briefly described in Chapter 3, "Designing with LogiCORE IP for the Endpoint Block." Descriptions of the block pins can be found in "Virtex-5 Endpoint Block Interface Descriptions" in Chapter 2, and descriptions of the attributes are in Appendix A, "Endpoint Block Attributes."

There are several interfaces to the Endpoint block, including:

- Clock and Reset interface, as described in "Clock and Reset Interface," page 19.
- Transaction Layer interface, as described in "Transaction Layer Interface," page 25.
- Management interface, as described in "Management Interface," page 34.
- Memory interface, as described in "Block RAM Interface," page 37.
- Transceiver interface, as described in "Transceiver Interface," page 39.
- Configuration and Status interface, as described in "Configuration and Status Interface," page 43.

The Transceiver interface, the Memory interface, and the Clock and Reset interface are automatically connected in the CORE Generator wrappers. These interfaces are not visible outside of the wrappers. The Transaction Layer interface must interface with the user design in fabric. The rest of the interfaces are optional; the user can choose whether to access them, and which pins to access.

Memory Requirements

There are three buffers that require block RAM: the Retry buffer, the Receive (Rx) buffer, and the Transmit (Tx) buffer. Each buffer has its own interface for independent access. The amount of block RAM needed can vary greatly, depending on the user requirements. For example, more block RAM is needed for the Tx and Rx buffers when there is a larger maximum payload size or more VCs. The amount of block RAM needed for the Retry buffer can increase with multilane designs because the bandwidth is larger.

Table 1-2 shows the number of 36-kbit block RAM buffers required for several different representative usages. The number varies from 3 to 40, depending on user requirements. The typical column assumes one VC and 128- or 256-byte maximum payload size. More information on buffer sizing can be found in "Block RAM Interface," page 37.

Table 1-2: **Number of 36-kbit Block RAMs Required**

	Number of 36-kbit Block RAMs		
	Minimum	**Typical**	**Maximum**
Receive Buffer	1	1	16
Transmit Buffer	1	1	16
Retry Buffer	1	1	8
Total	3	3	40

Use Models

The example topology shown in Figure 1-1 illustrates the major components in a PCIe system. Endpoint blocks and Legacy Endpoint blocks, both upstream-facing ports, are supported by the Endpoint block.

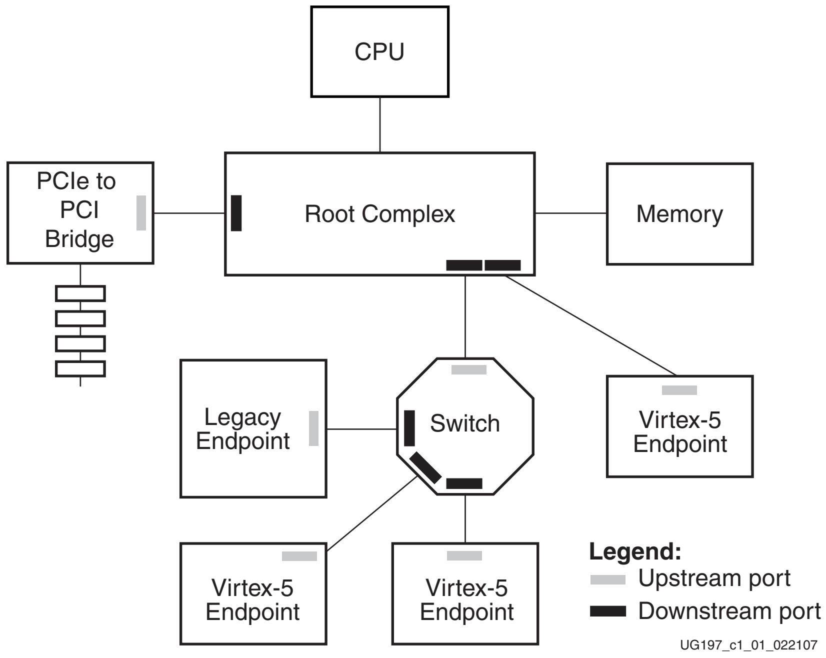

Figure 1-1: **Topology of a PCIe System**

Integrated Endpoint Block Functionality

Summary

This chapter presents information on the architecture and functionality of the Virtex-5 Endpoint block. The sections include:

- "Architecture Overview"
- "Virtex-5 Endpoint Block Interface Descriptions"
- "Registers"

Architecture Overview

The PCI Express protocol is divided into three layers: the Transaction Layer, the Data Link Layer, and the Physical Layer. These three layers interact with the Configuration Space. The Virtex-5 Endpoint block (Figure 2-1) provides the full functionality of the Transaction Layer, the Data Link Layer, the Physical Layer, and the Configuration Space as per the *PCI Express Base 1.1 Specification*.

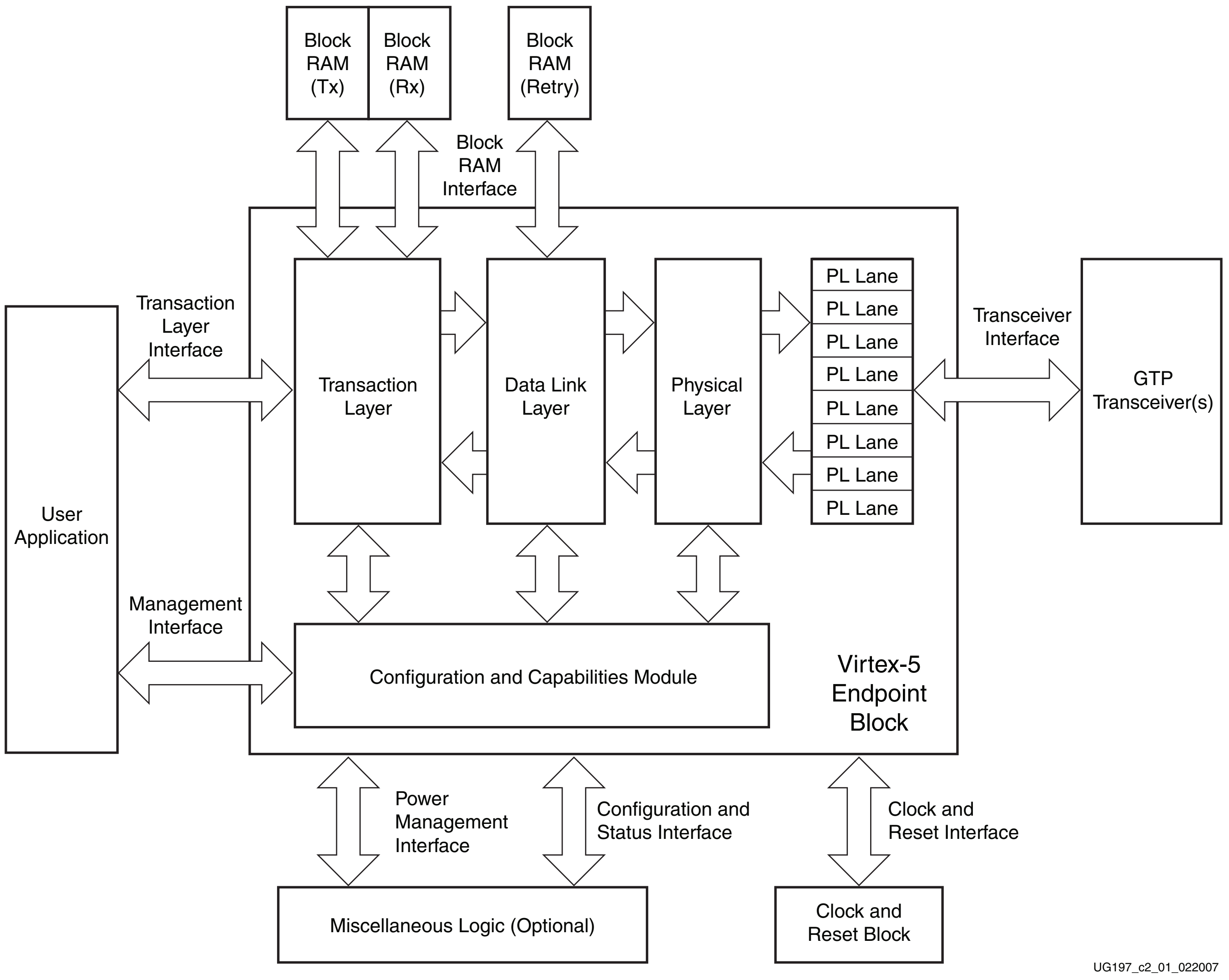

Figure 2-1: **Virtex-5 Integrated Endpoint Block Diagram**

Transaction Layer

The Transaction Layer (TL) is the upper layer in the architecture. It takes Transaction Layer Packets (TLPs) presented by user logic at the Transaction Layer interface and schedules them for transmission over the appropriate virtual channel (VC) for the specified traffic class. The module also advises the user application when TLPs are received.

TLPs can both make requests and complete requests from another device. They can also communicate certain types of events.

A TLP is composed of a header, data payload (for most packets), and optional end-to-end CRC (ECRC), as shown in Figure 2-2. The Endpoint block does not support the optional ECRC generation and checking; however, the block does pass through the ECRC untouched.

The Endpoint block's Transaction Layer implements one or two separate VCs, each with its own buffers and flow control logic. The VCs are implemented in both Tx and Rx directions for each of the traffic types supported by Endpoint blocks (posted requests, non-posted requests, and completions). The buffers are implemented separately for each VC so that the presence of a blockage on one VC does not cause the other VC to stall.

The Virtex-5 Endpoint block can be configured to have a maximum of two VCs. It has options for round robin, weighted round robin, or strict priority arbitration between these VCs.

The Transaction Layer also manages the credit-based flow control. The flow control mechanism ensures that a packet is not transmitted unless the receiving device has sufficient buffer space to accept it.

The PCI Express protocol supports four types of transactions: memory (read and write), I/O (read and write), configuration (read and write), and message.

Transactions are divided into three categories: posted, non-posted, and completion transactions. Memory writes and message transactions are posted transactions. The requester sends a packet, but the receiver does not return a completion. Non-posted transactions (memory reads, I/O reads and writes, and configuration reads and writes) require a response and are implemented as split transactions.

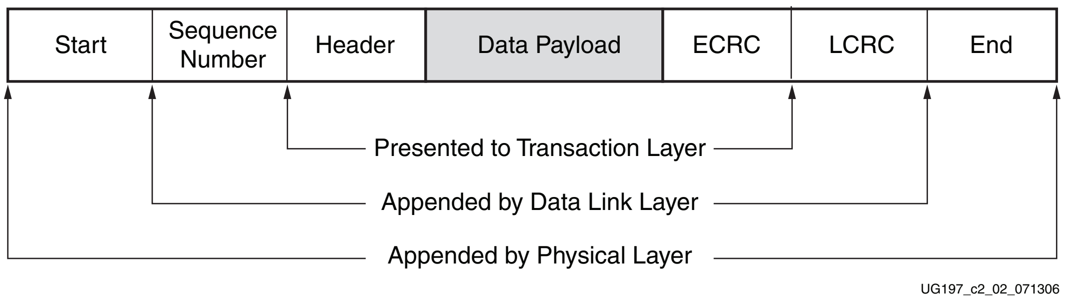
Start
Sequence Number
Header
Data Payload
ECRC
LCRC
End
Presented to Transaction Layer
Appended by Data Link Layer
Appended by Physical Layer
UG197_c2_02_071306

Figure 2-2: **PCIe Packet**

Data Link Layer

The Data Link Layer (DLL) resides between the Transaction Layer and the Physical Layer. Its primary responsibilities are link management and data integrity, including error detection and correction.

The transmission portion of the DLL accepts TLPs from the Transaction Layer and generates the appropriate TLP sequence number and Link CRC (LCRC), then passes the packet to the Physical Layer. It also places a copy of the packet in a retry buffer, making it available if the packet needs to be resent. Nullified packets are automatically purged from the retry buffer.

The DLL also generates and consumes special packets called Data Link Layer packets (DLLPs) that do not pass to the Transaction Layer. Types of DLLPs include acknowledgment (ACK/NAK), flow control, and power management. When the DLL detects errors in a packet, it requests retransmission of the packet until it is correctly received or until the link is determined to have failed.

The reception portion of the DLL checks the integrity of received TLPs. It also orders retransmission when the received TLP is found to be corrupt.

The reception portion of the DLL simply handles whatever is received, but the transmission portion also controls the order of release of the different types of packets. A prioritizer is included to sort the different sources of transmission into order of priority and schedule them for transmission according to the priority order recommended in the *PCI Express Base 1.1 Specification*.

Physical Layer

The Physical Layer module carries out the following functions:

- Packet framing and deframing
- Byte striping and unstriping; that is, distributing Tx packets across multiple lanes and reassembling Rx packets received over multiple lanes
- Generation and reception of ordered sets
- Link initialization and training, including the Link Training and Status State Machine (LTSSM)
- Generating scramble and descramble codes

Physical Layer Lane Module

There are eight Physical Layer lane modules, one for each lane that the Endpoint block supports.

On the transmission side of its operation, the PL lane module applies the scramble codes generated by the Physical Layer module to the transmit data, multiplexes this with ordered set data received from the Physical Layer module, and then passes the packet to the transceiver interface for transmission.

On the receive side, the Physical Layer lane module receives TLP bytes from the Transceiver interface, decodes ordered sets from this data, and descrambles DLLP and TLP data from the resulting datastream.

This module also detects the receipt of electrical idle characters. The remaining Physical Layer functionality, including lane-to-lane deskew and 8B/10B encoding and decoding, is included in the GTP transceivers.

Configuration and Capabilities Module

The Configuration and Capabilities module principally provides the repository for the different registers within the Configuration Space, including:

- Legacy PCI V3.0 Type 0 Configuration Space Header
- Legacy Capabilities
  - PCI Express
  - Power Management
  - Message Signaled Interrupts (MSIs)
- PCI Express Extended Capabilities
  - Virtual Channel
  - Device Serial Number

The Endpoint block does not support the Advanced Error Reporting Capability.

The module also includes a packet decoder and a packet generator for handling configuration and message packets.

Virtex-5 Endpoint Block Interface Descriptions

Clock and Reset Interface

Clocks

The Endpoint block has two synchronous clock domains: core_clk and user_clk. The user_clk domain allows user logic in the fabric to run at a slower speed than the Endpoint block in x1, x2, or x4 modes. Each clock domain has several clock ports to improve timing. All clock ports on the same clock domain must be tied to the same BUFG.

The user_clk domain is controlled by the CRMUSERCLK, CRMUSERCLKRXO, and CRMUSERCLKTXO ports (see Table 2-3). The user_clk domain clocks the following:

- The Management interface
- The Transaction Layer interface
- The write port of the Tx buffer
- The read port of the Rx buffer
- User logic in the fabric connected to the above interfaces

The core_clk domain is controlled by the CRMCORECLK, CRMCORECLKRXO, CRMCORECLKTXO, and CRMCORECLKDLO signals (see Table 2-3). The core_clk domain clocks the following:

- The rest of the Endpoint block
- The read port of the Tx buffer
- The write port of the Rx buffer
- The Retry buffer
- The Transceiver Interface
- Portions of the GTP transceiver (TXUSRCLK2, RXUSRCLK2)

Clock Frequency

The core_clk always runs at 250 MHz. The user_clk must also run at 250 MHz for x8 configurations to maintain full bandwidth. The user_clk can be run at lower frequencies for x1, x2, or x4, while still maintaining full bandwidth, lowering power, and simplifying timing closure. Table 2-1 shows the allowed clock frequencies.

Table 2-1: **Clock Frequency Versus Lane Width**

Configured Lane Width	core_clk Frequency (MHz)	user_clk Frequency (MHz)(1)
x1	250	62.5, 125, or 250
x2	250	62.5, 125, or 250
x4	250	125 or 250
x8	250	250

Notes:

1. The user_clk frequency is based on the configured lane width. It cannot be reduced, even when the negotiated lane width is smaller.

When the frequency of the user_clk domain is 250 MHz, there is no need to provide two separate clocks to the Endpoint block. In this case, the 250 MHz clock is tied to all the core_clk ports and the user_clk ports must be tied High. This gives a very simple timing model for the system: all signals on the Endpoint block and all signals on other blocks on the FPGA that directly interface with the Endpoint block are clocked by the same clock. These clock connections are included in the CORE Generator wrappers.

The core_clk and user_clk are obtained by using a Clock Management Tile (CMT). The reference clock is brought on the device through the CLKP and CLKN differential reference clock pins to the GTP transceiver. The reference clock should be forwarded from the GTP transceiver to the CMT. The CMT PLL must be used to derive the 250 MHz core_clk from the reference clock (unless a 250 MHz reference clock is used). See Figure 2-3 and Figure 2-4. The CMT PLL, BUFGs, and clocking connections are included in the CORE Generator wrappers.

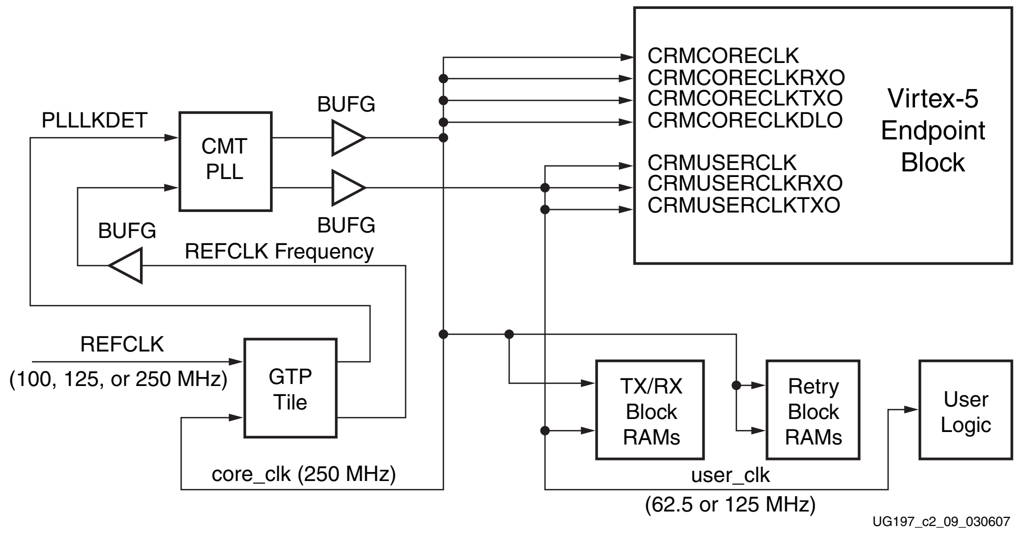

Figure 2-3: **Clocking for Applications with CLKDIVIDED = TRUE**

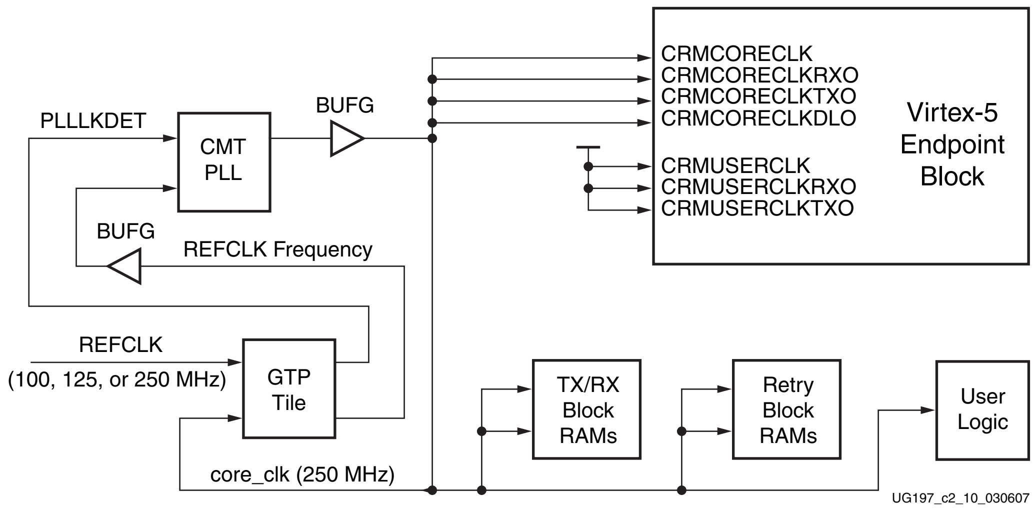

Figure 2-4: **Clocking for Applications with CLKDIVIDED = FALSE**

Resets

The Endpoint block supports three types of resets, as defined by the *PCI Express Base Specification*:

- *Cold reset*, a fundamental reset that occurs following the application of power.
- *Warm reset*, a fundamental reset that is triggered by hardware without the removal and reapplication of power.
- *Hot reset*, an in-band mechanism for propagating reset across a PCIe link.

The registers in the Endpoint block are divided into six reset domains:

- mgmt_rst: Management interface reset.
- nv_rst: Sticky (or non-volatile) registers reset. A sticky register retains its state through a hot reset.
- user_cfg_rst: Endpoint Configuration Space reset. All registers in the Endpoint Configuration Space, except the sticky registers, are affected.
- u_rst: Backend interface to the Transaction Layer (user_clk domain) reset.
- mac_rst: Physical Layer, including PL Lane reset.
- link_rst: Transaction Layer (core_clk domain), Data Link Layer, and part of the Configuration and Capabilities module reset. This affects all registers in the block that are not included in the other five reset domains.

There are six reset ports (see Table 2-3). The domain(s) that are reset by each port depend on the RESETMODE attribute (see Table 2-2).

- When RESETMODE = FALSE, most of the ports reset more than one domain; thus, only one of these signals should be asserted at a time. Two of the signals, CRMMACRSTN and CRMLINKRSTN, are not used in this mode.
- When RESETMODE = TRUE, each port resets just one domain (except for CRMMGMTRSTN, which resets the entire block); multiple reset signals can be asserted as needed.

Table 2-2: **The Effect of the RESETMODE Attribute on Reset Signal Functionality**

Port	RESETMODE	Reset Domain					
		user_cfg_rst	mac_rst	link_rst	u_rst	nv_rst	mgmt_rst
CRMUSERCFGRSTN	FALSE	•					
CRMMACRSTN(1)	FALSE						
CRMLINKRSTN(1)	FALSE						
CRMURSTN	FALSE	•	•	•	•		
CRMNVRSTN	FALSE	•	•	•	•	•	
CRMMGMTRSTN	FALSE	•	•	•	•	•	•
CRMUSERCFGRSTN	TRUE	•					
CRMMACRSTN	TRUE		•				
CRMLINKRSTN	TRUE			•			
CRMURSTN	TRUE				•		
CRMNVRSTN	TRUE					•	

Table 2-2: **The Effect of the RESETMODE Attribute on Reset Signal Functionality *(Continued)***

Port	RESETMODE	Reset Domain					
		user_cfg_rst	mac_rst	link_rst	u_rst	nv_rst	mgmt_rst
CRMMGMTRSTN	TRUE	•	•	•	•	•	•

Notes:

1. These ports are not used in this mode.

During FPGA configuration, the entire Endpoint block is reset, including the sticky register block, the PCI Configuration Space, and the Management Interface registers. All other resets of the block are controlled by the user through the six reset ports. These signals are asynchronous, but there is logic in the Endpoint block to guarantee synchronous deassertion with respect to the core_clk. The Endpoint block must be clocked while its reset port(s) are asserted in order for the appropriate portion(s) of the block to be reset.

The Endpoint block asserts the PIPERESETL*n* signals to all lanes when the MAC_RST domain is reset. PIPERESETL*n* is only deasserted for the active lanes (based on the ACTIVELANESIN attribute setting) and remains asserted for the unused lanes. The PIPERESETL*n* ports are connected to the RXCDRRESET port on the GTP transceivers. See Table 2-14, page 40 for details.

The user reset design in fabric for the PCIe system must assert the appropriate reset signals for warm reset, hot reset, DL_Down, etc. The user should also ensure that the Endpoint block is held in reset until the PLL is locked. This reset design is included in the CORE Generator wrapper.

The falling edge of the L0DLUPDOWN[0] output of the Endpoint block indicates when the link goes down (DL_Down status). The CRMDOHOTRESETN output is asserted when a hot reset is received from upstream. An Endpoint user design must use these outputs to reset a portion of the Endpoint block. This is done in the CORE Generator wrappers. The sticky registers and management interface registers should not be reset on DL_Down status or hot reset. The LTSSM does not need to be reset, but it can be reset after it transitions from Disabled (`1011`), Loopback (`1001`), Hot Reset (`1010`), Recovery (`1100`), or Configuration (`0011`) to Detect (`0001`). This transition can be seen by decoding the L0LTSSMSTATE outputs of the Endpoint block.

The CRMPWRSOFTRESETN output indicates when the Endpoint block transitions from the $D3_{hot}$ power state to the $D0_{uninitialized}$ state. This transition must be used to trigger the assertion of the CRMUSERCFGRSTN port on the Endpoint block. This is done in the CORE Generator wrappers.

Ports

Table 2-3 shows the Clock and Reset interface ports.

Table 2-3: **Clock and Reset Ports**

Port	Direction	Clock Domain	Description
CRMCORECLK	Input	core_clk	250 MHz clock from the FPGA, also drives Tx buffer read clock port, Rx buffer write clock ports, both Retry buffer clock ports, and the GTP RX/TXUSRCLK2 ports. Should be tied Low if the Endpoint block is not used. CRMCORECLK, CRMCORECLKRXO, CRMCORECLKTXO, and CRMCORECLKDLO must be tied to the output of the same BUFG.
CRMCORECLKDLO	Input	core_clk	250 MHz clock from the FPGA. Clocks the outputs of both Retry buffer ports. Should be tied Low if the Endpoint block is not used. CRMCORECLK, CRMCORECLKRXO, CRMCORECLKTXO, and CRMCORECLKDLO must be tied to the output of the same BUFG.
CRMCORECLKTXO	Input	core_clk	250 MHz clock from the FPGA. Clocks the Tx buffer read port outputs. Should be tied Low if the Endpoint block is not used. CRMCORECLK, CRMCORECLKRXO, CRMCORECLKTXO, and CRMCORECLKDLO must be tied to the output of the same BUFG.
CRMCORECLKRXO	Input	core_clk	250 MHz clock from the FPGA. Clocks the Rx buffer write port outputs. Should be tied Low if the Endpoint block is not used. CRMCORECLK, CRMCORECLKRXO, CRMCORECLKTXO, and CRMCORECLKDLO must be tied to the output of the same BUFG.
CRMUSERCLK	Input	user_clk	User clock. Should be tied Low if the Endpoint block is not used. CRMUSERCLK, CRMUSERCLKRXO, and CRMUSERCLKTXO must be tied to the output of the same BUFG when they are at a lower frequency than CRMCORECLK. Must be tied High when frequency is the same as CRMCORECLK (250 MHz).
CRMUSERCLKTXO	Input	user_clk	User clock. Clocks Tx buffer write port outputs. Should be tied Low if the Endpoint block is not used. CRMUSERCLK, CRMUSERCLKRXO, and CRMUSERCLKTXO must be tied to the output of the same BUFG when they are at a lower frequency than CRMCORECLK. Must be tied High when frequency is the same as CRMCORECLK (250 MHz).
CRMUSERCLKRXO	Input	user_clk	User clock. Clocks Rx buffer read ports outputs. Should be tied Low if the Endpoint block is not used. CRMUSERCLK, CRMUSERCLKRXO, and CRMUSERCLKTXO must be tied to the output of the same BUFG when they are at a lower frequency than CRMCORECLK. Must be tied High when frequency is the same as CRMCORECLK (250 MHz).
CRMURSTN	Input	core_clk	User reset (active Low). When the RESETMODE attribute is set to FALSE, resets all the registers in the Endpoint block, except the sticky registers and the Management Interface registers. When the RESETMODE attribute is set to TRUE, resets the backend interface to the Transaction Layer (user_clk domain). Asynchronous, but the Endpoint block ensures internal synchronous deassertion with respect to core_clk. Should be tied High if not used in the user design or if the block is not used.

Table 2-3: **Clock and Reset Ports *(Continued)***

Port	Direction	Clock Domain	Description
CRMNVRSTN	Input	core_clk	Non-volatile reset (active Low). When the RESETMODE attribute is set to FALSE, resets the sticky registers, and everything else in the block except for the Management Interface registers. When the RESETMODE attribute is set to TRUE, resets the sticky registers only. Asynchronous, but the Endpoint block ensures internal synchronous deassertion with respect to core_clk. Should be tied High if not used in the user design or if the block is not used.
CRMMGMTRSTN	Input	core_clk	Management interface reset (active Low). Resets the registers in the block, including the management interface registers. The function of this signal does not depend on the RESETMODE attribute setting. Asynchronous, but the Endpoint block ensures internal synchronous deassertion with respect to core_clk. Should be tied High if not used in the user design or if the block is not used.
CRMUSERCFGRSTN	Input	core_clk	User configuration reset (active Low). Resets all the registers in the PCI Express Configuration Space except the sticky registers. The function of this signal does not depend on the RESETMODE attribute setting. Asynchronous, but the Endpoint block ensures internal synchronous deassertion with respect to core_clk. Should be tied High if not used in the user design or if the block is not used.
CRMMACRSTN	Input	core_clk	MAC reset (active Low). When the RESETMODE attribute is set to FALSE, CRMMACRSTN is not used and should be tied High. When the RESETMODE attribute is set to TRUE, CRMMACRSTN resets the MAC link and MAC lane logic (Physical Layer). Asynchronous, but the Endpoint block ensures internal synchronous deassertion with respect to core_clk. Should be tied High if not used in the user design or if the block is not used.
CRMLINKRSTN	Input	core_clk	Link reset (active Low). When the RESETMODE attribute is set to FALSE, CRMLINKRSTN is not used and should be tied High. When the RESETMODE attribute is set to TRUE, CRMLINKRSTN resets the core_clk domain of the Transaction Layer, part of the Configuration module, and the Data Link Layer. Asynchronous, but the Endpoint block ensures internal synchronous deassertion with respect to core_clk. Should be tied High if not used in the user design or if the block is not used.
CRMDOHOTRESETN	Output	core_clk	Hot reset (active Low). Asserted on completion of hot reset handshake as a prompt for user logic to be reset. See "Resets," page 21.
CRMPWRSOFTRESETN	Output	core_clk	Soft reset (active Low). Asserted when the block makes the transition from $D3_{hot}$ to $D0_{uninitialized}$, as a prompt for user logic to be reset (with CRMUSERCFGRSTN).

Transaction Layer Interface

Packets are presented to and received from the Endpoint block's Transaction Layer through the Transaction Layer interface. On this interface, a *beat* is a clock cycle where both the source and destination are ready. The main Transaction Layer interface framing signals indicate the start of frame, the end of frame, destination ready, and source ready.

Transmit

The transmit portion of the interface accepts the data from the user application that is to be transmitted to the link partner. Transaction Layer Packets (TLPs) for transmission need to be created in accordance with the *PCI Express Base Specification*, then presented to the Endpoint block's Transaction Layer interface.

Data

The data bus contains data for the packet header, payload, and digest, if present. The header must be written before the data. The digest is treated as the last word of the data. The presence of a TLP digest (ECRC) is indicated by setting the TD bit in the header to '1'. For more information on creating the TLP digest, see Chapter 2 of the *PCI Express Base Specification*.

Packets must be formed by the user in accordance with the *PCI Express Base Specification*, and presented on the LLKTXDATA ports as shown in Table 2-4 and Table 2-5. The header and data must be presented in the order shown, although they need not be presented on consecutive clock cycles, as shown in the timing diagram in Figure 2-5.

The first header DW (32-bit DWORD) of a packet must always appear on LLKTXDATA[63:32], and cannot appear in the same clock cycle as the final DW of the previous packet (but can appear in the next clock cycle, if all other signaling requirements are met).

Table 2-4: **Byte Ordering on LLKTXDATA for 3 DW Header, 4 DW Payload**

	63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
	byte0	byte1	byte2	byte3	byte4	byte5	byte6	byte7
LLKTXDATA	Header DW0				Header DW1			
	Header DW2				Payload DW0			
	Payload DW1				Payload DW2			
	Payload DW3				don't care			

Table 2-5: **Byte Ordering on LLKTXDATA for 4 DW Header, 4 DW Payload**

	63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
	byte0	byte1	byte2	byte3	byte4	byte5	byte6	byte7
LLKTXDATA	Header DW0				Header DW1			
	Header DW2				Header DW3			
	Payload DW0				Payload DW1			
	Payload DW2				Payload DW3			

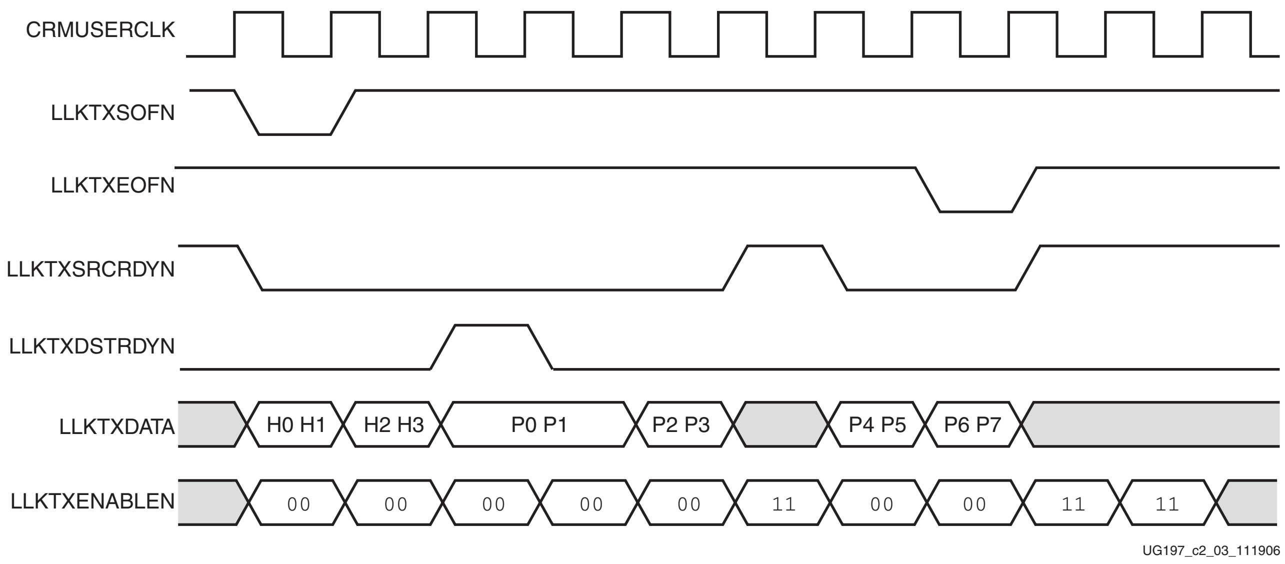

Figure 2-5: **Transaction Layer Interface Transmit Timing Diagram Showing a 4 DW Header**

Channels

The Transaction Layer interface allows for the generic concept of channels to deal with multiple logical channels for one physical interface. The Endpoint block supports eight traffic classes each having three traffic types: posted, non-posted, and completion.

Channel Ready

When the Endpoint block is ready to accept a packet into one of the buffers associated with a particular VC and traffic type, it asserts the appropriate channel ready signal(s) of the traffic class(es) that have been mapped to that VC. There is one channel ready signal per traffic class, according to the type of packet (LLKTXCHPOSTEDREADYN[7:0], LLKTXCHNONPOSTEDREADYN[7:0], and LLKTXCHCOMPLETIONREADYN[7:0]). More than one channel can be ready on any clock cycle.

Channel Select

The user application sets LLKTXCHTC[2:0] to select the traffic class and LLKTXCHFIFO[1:0] to indicate in which Tx FIFO the data is to be placed: posted (`00`), non-posted (`01`), or completion (`10`).

LLKTXDSTRDYN is asserted when the selected channel has space available, and LLKTXCHANSPACE reports the amount of free space. Pipelining causes a delay of one cycle between a change in a channel select and an output based on a selected channel (LLKTXDSTRDYN or LLKTXCHANSPACE). Also, it takes four clock cycles to update LLKTXCHANSPACE after a write transaction.

Transmit Framing

The user application uses the framing signals to indicate the start and end of frames as well as the position of the header and digest (if present). The framing signals also indicate how many 32-bit DWORDs are valid at the end of the header and the end of the frame.

The LLKTXSOFN and LLKTXEOFN signals delineate the frame boundaries.

Framing Errors

The following conditions are framing errors and are not allowed:

- Two SOFs without an intervening EOF
- Two EOFs without an intervening SOF
- An SOF and EOF in the same cycle

DWORD Enables

The Transaction Layer interface data bus, LLKTXDATA, is 64 bits wide, allowing the user to transfer one QWORD of data into the Endpoint block per clock cycle. Since the PCIe protocol allows DWORD (32-bit) alignment of header and data, certain QWORDs contain only one valid DWORD. The LLKTXENABLEN[1:0] bus indicates which DWORD(s) contain valid header or data information. Bit 1 of LLKTXENABLEN[1:0] refers to LLKTXDATA[63:32], and bit 0 refers to LLKTXDATA[31:0]. A value of `0` indicates that the corresponding DWORD is valid.

All 64 bits of LLKTXDATA must be enabled, except on the last cycle of a TLP transfer, when LLKTXEOFN = `0`. For the last QWORD of a packet, it is possible that only LLKTXDATA[63:32] is valid because of DWORD alignment. This is denoted by LLKTXENABLEN[1:0] = `01` during LLKTXEOFN. Otherwise, LLKTXENABLEN[1:0] = `00` is used for all other cycles of a TLP transfer.

Transmit Handshake

The handshake signals control the flow of data between the user application and the Endpoint block.

When the user application has a packet ready for transmission, it asserts LLKTXCHTC and LLKTXCHFIFO to select the channel, and asserts LLKTXSRCRDYN to indicate that the data bus and framing signals are set to transfer data. The user application does not need to wait for the Endpoint block to assert LLKTXDSTRDYN. Either ready signal can be asserted first. See Figure 2-6, page 28.

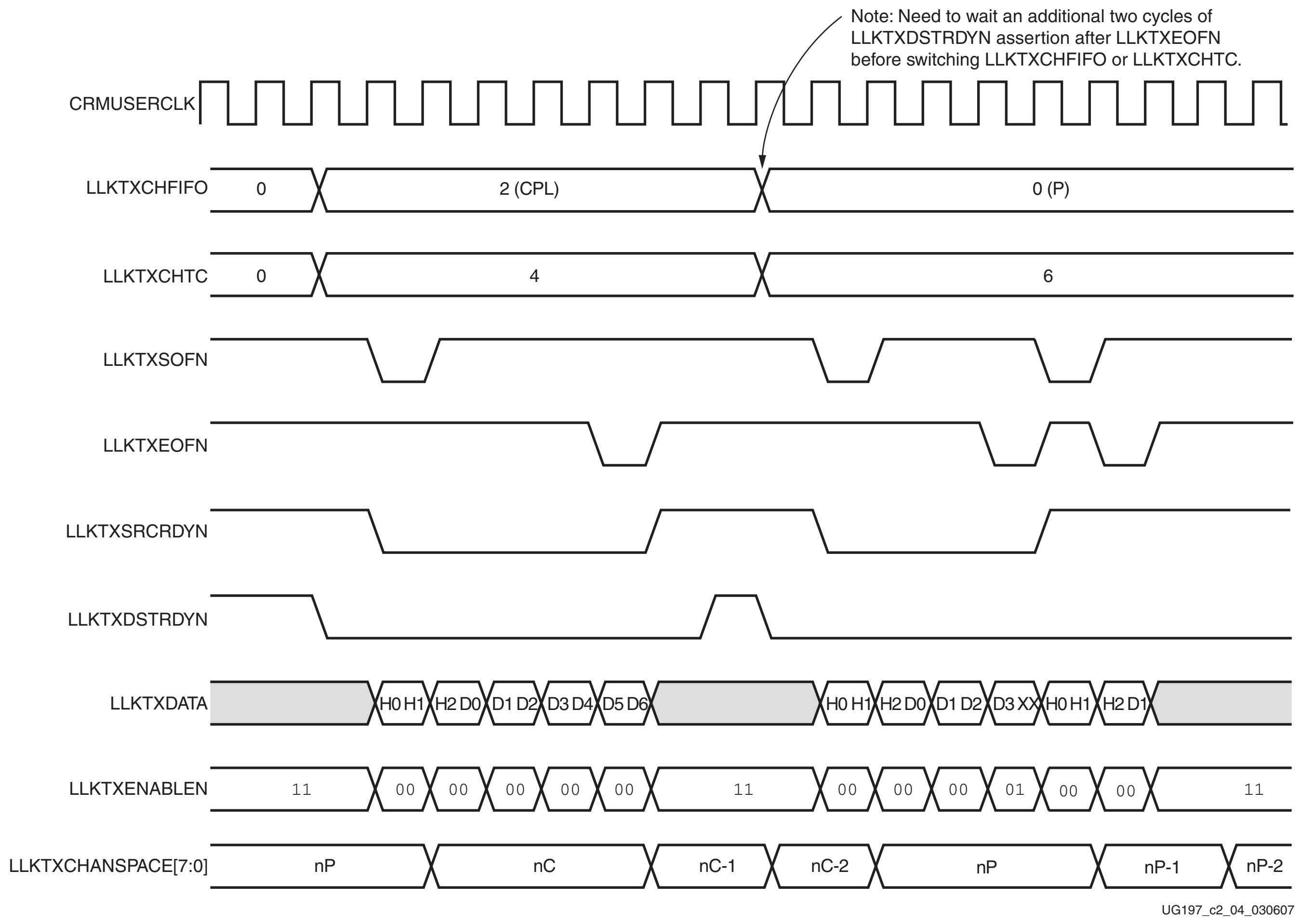

Figure 2-6: **Transaction Layer Interface Transmit Channel Switching Timing Diagram**

After transmission of a packet, a subsequent packet on the same channel (FIFO and TC) can be sent immediately. For packets on a different channel (FIFO or TC), the user logic must pause until LLKTXDSTRDYN is asserted for two cycles after LLKTXEOFN before changing the LLKTXCHFIFO or LLKTXCHTC signal.

If the transmit buffer becomes full during packet transfer, it deasserts LLKTXDSTRDYN and stalls data transfers on the Transaction Layer interface until space becomes available again as the packet is sent over the serial interface. The user can optionally check the amount of space in a channel using LLKTXCHANSPACE and decide to not begin sending the packet over the Transaction Layer interface if insufficient space is available to send the packet without stalling. The requirement on channel switching timing as shown in Figure 2-6 must also be observed.

LLKTXENABLEN is ignored unless LLKTXSRCRDYN and LLKTXDSTRDYN are asserted.

Receive

The receive portion of the interface passes the data received from the link partner to the user application in fabric.

Receive Framing

The receive framing signals are similar to the transmit framing signals. In receive packets, the header is always before the data. LLKRXVALIDN = 00 on all valid cycles except the last one. If the total number of 32-bit DWORDs (header plus payload) is odd, LLKRXVALIDN is 01 on the last beat.

Receive Handshake

When a packet has been received into the Rx buffer and confirmed as valid, the Endpoint block asserts the appropriate LLKRXCHPOSTEDAVAILABLEN, LLKRXCHNONPOSTEDAVAILABLEN, or LLKRXCHCOMPLETIONAVAILABLEN signal to indicate the type of packet that has been received. In some cases, requesting a packet that has been received can violate PCIe transaction ordering rules. The user application must monitor the LLKRXPREFERREDTYPE signal and follow the rules specified in "Ordering at Reception," page 67 before requesting a packet from the Endpoint block.

The user application selects the traffic class and the traffic type to read by setting LLKRXCHTC to select the traffic class, and LLKRXCHFIFO to select posted, non-posted, or completion.

The user asserts the LLKRXDSTREQN signal to request data from the Endpoint block. For each clock where LLKRXDSTREQN is asserted, the Endpoint block asserts LLKRXSRCRDYN for one clock after a minimum delay of 3 + TLRAMREADLATENCY. The value of the TLRAMREADLATENCY attribute is in the range [2 .. 6].

The receive interfaces provides a LLKRXSRCRDYN signal when data is valid on LLKRXDATA.

The Endpoint block asserts LLKRXSRCLASTREQN three user_clk cycles after it has received the second-to-last (penultimate) request for the current Rx packet via LLKRXDSTREQN. A single assertion of LLKRXDSTREQN during the three user_clk cycles is sufficient for the block to receive its final request for the current Rx packet. Other assertions of LLKRXDSTREQN are ignored, provided LLKRXDSTCONTREQN is deasserted. If the block has received the final request when LLKRXSRCLASTREQN is asserted, no further requests should be issued on subsequent cycles (via LLKRXDSTREQN) unless there are further packets available on the selected channel as indicated by the corresponding LLKRX*AVAILABLEN signal (where * is POSTED, NONPOSTED, or COMPLETION). When configuration packets are being processed, the LLKRX*AVAILABLEN signals are deasserted until processing of the configuration packet is complete.

When there is no more data to receive, LLKRXDSTREQN must be deasserted in the cycle after LLKRXSRCLASTREQN is first asserted. See Figure 2-7. Failure to do this causes the block to enter an undefined state; as a result, subsequent packets can be corrupted.

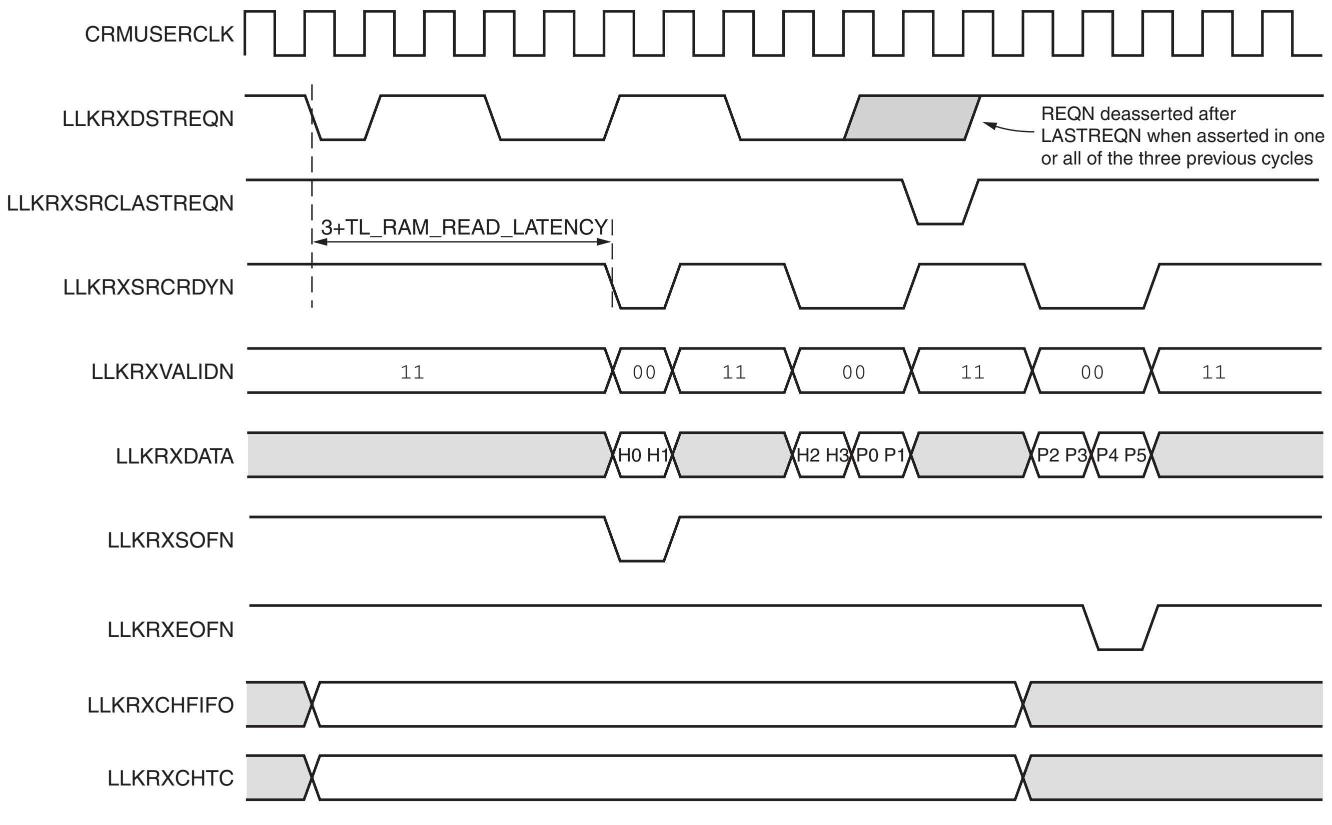

Figure 2-7: **Transaction Layer Interface Receive Timing Diagram Showing a 4 DW Header and 6 DW Data Payload**

If LLKRXDSTREQN is deasserted in each of the three cycles, including initial assertion of LLKRXSRCLASTREQN, then one more request is required to complete reception of the current packet. To complete reception, the user must assert LLKRXDSTREQN for one subsequent cycle. See Figure 2-8. LLKRXSRCLASTREQN remains asserted until three cycles after the Endpoint block has received the final request for the current Rx packet.

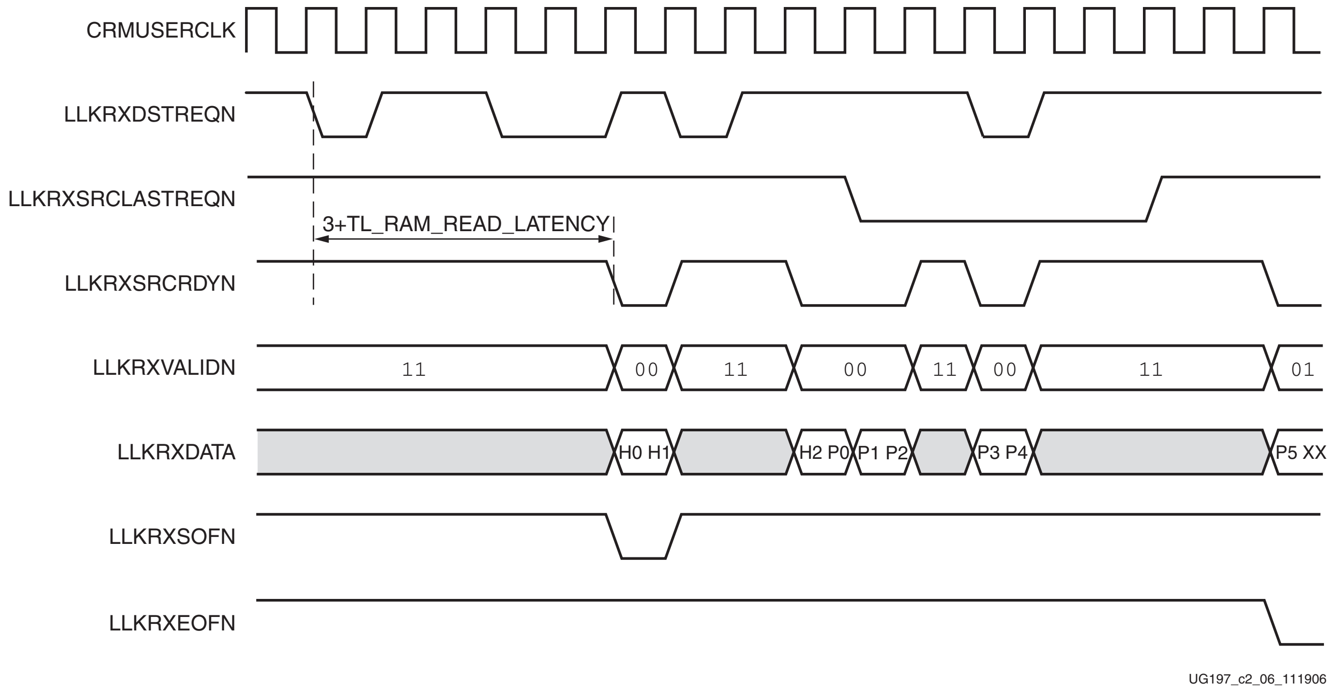

Figure 2-8: **Transaction Layer Interface Receive Timing Diagram Showing a 3 DW Header and 6 DW Data Payload**

Ports

Table 2-6 shows the ports of the Transaction Layer interface.

Table 2-6: **Transaction Layer Interface Ports**

Port	Direction	Clock Domain	Description
LLKTCSTATUS[7:0]	Output	user_clk	Report the status of the eight traffic classes: 1 implies initialized; 0 implies uninitialized.
LLKTXDATA[63:0]	Input	user_clk	Transaction Layer interface transmit data.
LLKTXSRCRDYN	Input	user_clk	Asserted (active Low) if the transmit source has data available.
LLKTXDSTRDYN	Output	user_clk	Asserted (active Low) if the transmit destination has space available on the selected channel.
LLKTXSRCDSCN	Input	user_clk	Transmit source Frame Discard (active Low). *Not supported. Must be tied High.*

Table 2-6: **Transaction Layer Interface Ports *(Continued)***

Port	Direction	Clock Domain	Description
LLKTXCHANSPACE[9:0]	Output	user_clk	Amount of free space in the Tx FIFO as selected by LLKTXCHTC and LLKTXCHFIFO. Bit [9] indicates if space is available for header: 1: Space for one header 0: No space for header Bit [8] indicates if space is available for data: 1: Space for data 0: No space for data Bits [7:0] indicate the number of data credits available: 1 .. 255: Number of credits available 0: Meaning depends on bit [8] setting: – If bit [8] = 0, no credits are available. – If bit [8] = 1, at least 256 credits are available.
LLKTXSOFN	Input	user_clk	Transaction Layer interface Tx Start of Frame (active Low).
LLKTXEOFN	Input	user_clk	Transaction Layer interface Tx End of Frame (active Low).
LLKTXSOPN	Input	user_clk	*Not supported. Must be tied High.*
LLKTXEOPN	Input	user_clk	*Not supported. Must be tied High.*
LLKTXENABLEN[1:0]	Input	user_clk	Word enable for Transaction Layer interface Transmit bus (active Low).
LLKTXCHTC[2:0]	Input	user_clk	Traffic class portion of Channel Select.
LLKTXCHFIFO[1:0]	Input	user_clk	FIFO portion of Channel Select. 00: Posted 01: Non-posted 10: Completion 11: *Reserved*
LLKTXCHPOSTEDREADYN[7:0]	Output	user_clk	Channel ready for posted packets TC7 – TC0 (active Low).
LLKTXCHNONPOSTEDREADYN[7:0]	Output	user_clk	Channel ready for non-posted packets TC7 – TC0 (active Low).
LLKTXCHCOMPLETIONREADYN[7:0]	Output	user_clk	Channel ready for completion packets TC7 – TC0 (active Low).
LLKRXDATA[63:0]	Output	user_clk	Transaction Layer interface receive data.

Table 2-6: **Transaction Layer Interface Ports *(Continued)***

Port	Direction	Clock Domain	Description
LLKRXSRCRDYN	Output	user_clk	Asserted (active Low) for one cycle if the receive source has data available on LLKRXDATA in response to an earlier LLKRXDSTREQN. Data must be captured by the user design during the cycle LLKRXSRCRDYN is asserted. LLKRXSRCRDYN reflects the value of LLKRXVALID[1:0]. LLKRXSRCRDYN = 1 when LLKRXVALIDN = 11. LLKRXSRCRDYN = 0 when either bit of LLKRXVALIDN = 0.
LLKRXDSTREQN	Input	user_clk	Receive data destination request (active Low). See LLKRXSRCLASTREQN.
LLKRXSRCLASTREQN	Output	user_clk	Asserted three cycles after the block has received the penultimate request for the current Rx packet. If LLKRXDSTREQN was asserted in one of the three cycles, then the block has received the final request for the current Rx packet. No further requests should be issued (with assertion of LLKRXDSTREQN) unless there are further packets available on the selected channel as indicated by the corresponding LLKRXCH*AVAILABLEN signal, where * is POSTED, NONPOSTED, or COMPLETION (active Low).
LLKRXDSTCONTREQN	Input	user_clk	When this signal is asserted, every assertion of LLKRXDSTREQN requests data from the selected channel, which allows continuous requests while receiving back-to-back packets. Should only be asserted in cases where there is a further packet(s) of the same type to be received after the current one. See "Performance Considerations," page 63.
LLKRXSOFN	Output	user_clk	Transaction Layer interface Rx Start of Frame (active Low).
LLKRXEOFN	Output	user_clk	Transaction Layer interface Rx End of Frame (active Low).
LLKRXSOPN	Output	user_clk	*Not supported. Must be tied High.*
LLKRXEOPN	Output	user_clk	*Not supported. Must be tied High.*
LLKRXVALIDN[1:0]	Output	user_clk	Word enable for Transaction Layer interface receive bus (active Low).
LLKRXCHTC[2:0]	Input	user_clk	Traffic class portion of Channel Select.
LLKRXCHFIFO[1:0]	Input	user_clk	FIFO portion of Channel Select. 00: Posted 01: Non-posted 10: Completion 11: *Reserved*
LLKRXCHPOSTEDAVAILABLEN[7:0]	Output	user_clk	Traffic classes with complete posted packets available (active Low).

Table 2-6: **Transaction Layer Interface Ports *(Continued)***

Port	Direction	Clock Domain	Description
LLKRXCHNONPOSTEDAVAILABLEN[7:0]	Output	user_clk	Traffic classes with complete non-posted packets available (active Low).
LLKRXCHCOMPLETIONAVAILABLEN[7:0]	Output	user_clk	Traffic classes with complete completion packets available (active Low).
LLKRXPREFERREDTYPE[15:0]	Output	user_clk	Used with LLKRXCH*AVAILABLEN to determine which queues have packets that can be read from the associated FIFO in accordance with PCIe transaction ordering rules. The bits are interpreted in pairs with bits [1:0] allocated to TC0, bits [3:2] to TC1, and so on. Within those two bits: `00`: Posted `01`: Non-posted `10`: Completion `11`: *Reserved*

Management Interface

The Management interface is used to access various registers and signals in the Endpoint block, including the PCI Express Configuration Space, and various control and status registers. The Management Interface also contains output signals for statistics and monitoring and an interface to read flow control credit outputs.

The interface has separate 32-bit data read and write buses. Separate read and write enables control the type of access on the interface. For writes, byte-write enables determine which byte of the 32-bit data (DWORD) is written. Figure 2-9 shows the read timing for the Management interface.

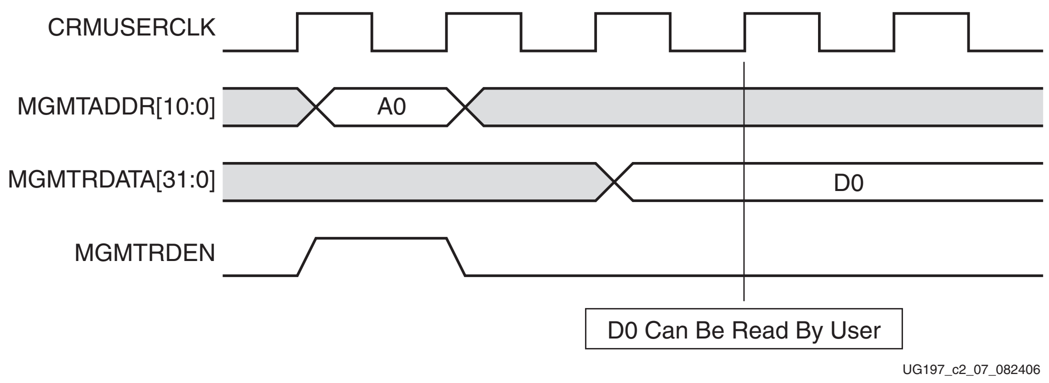

Figure 2-9: **Management Interface Read Timing**

The Management Interface address bus has DWORD addressing. A typical processor bus has byte addressing, where the lower two bits of the address bus indicate which byte in a DWORD is accessed. To connect the Management Interface to a processor bus, the lower two bits of the processor address are decoded with user logic to generate the byte write enables for the Management Interface.

To use the Management interface to override attributes (for example, DEVICEID), the Endpoint block must be held in reset during and for at least four cycles after performing the final Management write to the attribute address. This must be done to allow the new values to propagate within the Endpoint block. The reset port(s) should be asserted for an additional four CRMUSERCLK cycles after the final assertion of MGMTWREN (see Figure 2-10). The exact port(s) that must be asserted to reset the block (indicated by "reset" in the timing diagram) depends on the reset mode. When RESETMODE = FALSE, the CRMNVRSTN port should be asserted as shown in the timing diagram (Figure 2-10). When RESETMODE = TRUE, all reset ports other than CRMMGMTRSTN should be asserted.

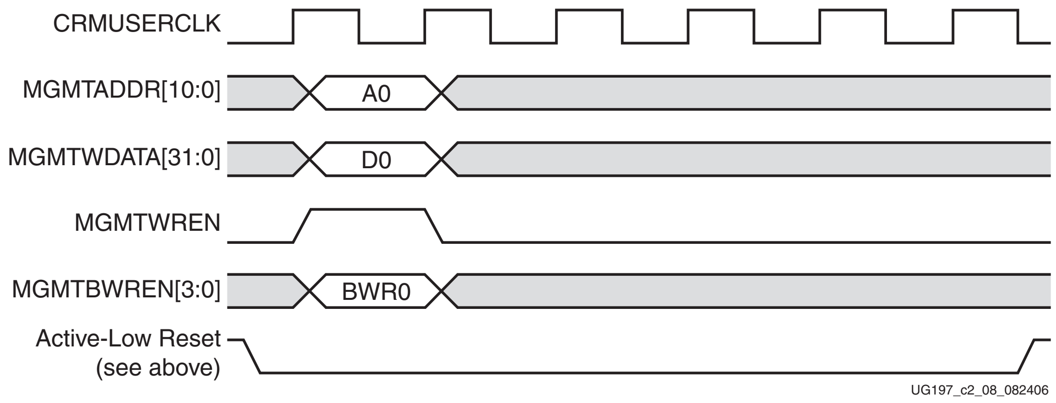

Figure 2-10: **Management Interface Write Timing**

Reset

All the flip-flops in the Management Interface have their asynchronous reset input driven by the CRMMGMTRSTN port. Endpoint block logic ensures that this reset is synchronously deasserted with respect to core_clk. Separating the management reset allows the user to reset the rest of the block without resetting the management interface. Attribute values such as DEVICEID or VENDORID that were previously written through the management interface are retained, even when other parts of the Endpoint block are reset (because of link down, soft reset, etc.). If the user desires to override the attribute definitions for attributes such as DEVICEID and VENDORID, they only need to be written once at power-on.

Ports

Table 2-7 shows the ports of the Management interface.

Table 2-7: **Management Interface Ports**

Port	Direction	Clock Domain	Description
MGMTRDATA[31:0]	Output	user_clk	Management Interface read data.
MGMTWDATA[31:0]	Input	user_clk	Management Interface write data.
MGMTBWREN[3:0]	Input	user_clk	Management Interface byte write enables.
MGMTWREN	Input	user_clk	Management Interface write enable.
MGMTADDR[10:0]	Input	user_clk	Management Interface address.
MGMTRDEN	Input	user_clk	Management Interface read enable.

Table 2-7: **Management Interface Ports *(Continued)***

Port	Direction	Clock Domain	Description
MGMTSTATSCREDIT[11:0]	Output	user_clk	Credit information as selected by MGMTSTATSCREDITSEL[6:0]. MGMTSTATSCREDIT[11:0] is updated two cycles after MGMTSTATSCREDITSEL[6:0] is switched.
MGMTSTATSCREDITSEL[6:0]	Input	user_clk	Channel select for credit information output to MGMTSTATSCREDIT[11:0]. Bits [1:0] select the VC: `00`: VC0 `01`: VC1 `10`: *Reserved* `11`: *Reserved* Bits [4:2] select the channel. `000`: Posted header (PH) `001`: Non-posted header (NPH) `010`: Completion header (CH) `011`: Posted data (PD) `100`: Non-posted data (NPD) `101`: Completion data (CD) `110`: *Reserved* `111`: *Reserved* Bits [6:5] select the type of credit information. `00`: credits consumed - Tx credits used by transmitted packets `01`: credit limit - Tx credits received from link partner `10`: credits allocated - Rx credits issued to link partner `11`: credits received - Rx credits consumed by received packets
MGMTPSO[16]	Output	user_clk	Master Data parity error. Bit 8 of the Status register.
MGMTPSO[15]	Output	user_clk	Signaled Target abort. Bit 11 of the Status register.
MGMTPSO[14]	Output	user_clk	Received Target abort. Bit 12 of the Status register.
MGMTPSO[13]	Output	user_clk	Received Master abort. Bit 13 of the Status register.
MGMTPSO[12]	Output	user_clk	Signaled system error. Bit 14 of the Status register.
MGMTPSO[11]	Output	user_clk	Detected parity error (poisoned TLP). Bit 15 of the Status register.
MGMTPSO[10]	Output	user_clk	Correctable error detected. Bit 0 of the Device Status register.
MGMTPSO[9]	Output	user_clk	Nonfatal error detected. Bit 1 of the Device Status register.
MGMTPSO[8]	Output	user_clk	Fatal error detected. Bit 2 of the Device Status register.
MGMTPSO[7]	Output	user_clk	Unsupported request detected. Bit 3 of the Device Status register.
MGMTPSO[6]	Output	user_clk	Transactions Pending. Bit 5 of the Device Status register.
MGMTPSO[5:0]	Output	user_clk	*Reserved.*

Block RAM Interface

The Transmit (Tx), Receive (Rx), and Retry buffers are implemented with block RAM. Each buffer has separate read and write interfaces. The sizes of the buffers can vary based on the application's needs.

- *Transmit buffer*. Buffers transmitted packets. It is divided into separate areas for the different VCs, and each area is further divided into separate FIFOs for posted, non-posted, and completion transactions (see Figure A-1, page 94 and Table A-7, page 99).
- *Receive buffer*. Buffers received packets. It is divided into separate areas for the different VCs, and each area is further divided into separate FIFOs for posted, non-posted, and completion transactions (see Figure A-1, page 94 and Table A-7, page 99).
- *Retry buffer*. Holds a copy of each TLP that is currently in the process of being transmitted until the information has been received correctly (or it becomes clear that the link has failed).

These three buffers are instantiated and configured in the CORE Generator wrappers, based on selections made in the CORE Generator GUI. Users implementing their design with the CORE Generator wrappers do not need to explicitly set any of the attributes or connect any pins described in this section. The block RAM datapaths are 64 bits wide.

All three block RAM interfaces operate synchronously to the rest of the Endpoint block. Each interface has separate read and write addresses, data, and control signals.

Table 2-8 shows the recommended aspect ratio for both ports of each dual-port block RAM. The block RAM can be connected directly to the Endpoint block using fabric interconnect, without additional fabric logic.

Table 2-8: **Block RAM Sizing**

RAM Requirement (KB) per Buffer	Number of 36K Block RAMs	Address Bits Used	Aspect Ratio of Each Block RAM (no ECC[1])	Block RAM Mode
4	1	[8:0]	512 x 64	Simple Dual Port
8	2	[9:0]	1k x 32	True Dual Port
16	4	[10:0]	2k x 16	True Dual Port
32	8	[11:0]	4k x 8	True Dual Port
64[2]	16	[12:0]	8k x 4	True Dual Port

Notes:

1. If ECC is required, each block RAM must be 512 x 64. This block RAM requirement uses additional fabric logic for all cases except when one 36K block RAM is used for a buffer. Fabric logic is always required if the user wants ECC error counting or logging.
2. Not available for the Retry buffer.

Since the total amount of RAM used for the Retry buffer must always be a power of two, the next size up is chosen when necessary. This restriction does not apply to the Tx and Rx buffers, but choosing a size that is not a power of two might require additional fabric logic. The CORE Generator tool always instantiates the Tx and Rx buffers as a power of two.

Block RAM output registers should generally be used but are not necessary if the design can meet timing without them. The CORE Generator tool always uses the block RAM output registers.

A pipeline stage can be added in the fabric between the Endpoint and block RAM blocks, if necessary, to meet timing. A pipeline stage can be added for either read or write, or both.

It is possible to pipeline address/control or data, or both. See "Buffer Latency" in Appendix A for more information. These additional pipeline stages are not automatically added in the CORE Generator wrapper because whether or not they are needed depends on the placement of the rest of the user's design.

Rx and Tx Buffer Capacity

Although the Tx, Rx, and Retry buffers are implemented using fully configurable block RAM, there are hard limits on how many packets can be buffered in the Tx and Rx buffers. Table 2-9 shows the maximum number of packets that can be buffered for each of the Rx and Tx buffers when the Endpoint block is configured for one or two VCs. The number of packets that can be buffered can be further limited by the various FIFO sizes and the negotiated maximum payload size set when the link is initialized.

The size of posted or completion packets is the size of the payload plus the size of the header. The posted header size should be 24 bytes, due to rounding up because of the 64-bit buffer width; 16 bytes should be used for the completion header size. Non-posted packets should be allocated 24 bytes each, due to rounding up.

Table 2-9: **Maximum Rx and Tx Buffer Capacity**

	Maximum Number of Packets (VC0)	Maximum Number of Packets (VC1)	Maximum Size (VC0)	Maximum Size (VC1)
Posted Packets	8	8	32 Kbytes	32 Kbytes
Non-Posted Packets	8	8	512 bytes	512 bytes
Completion Packets	8	8	32 Kbytes	32 Kbytes

Retry Buffer Size

The optimal size of the Retry buffer depends on the level of traffic that is expected, because the buffer needs to be large enough not to throttle the traffic flow. At a minimum, the buffer should be able to hold at least two TLPs of the size exchanged between the Endpoint block and the device to which it is attached (the negotiated maximum payload size set when the link is initialized). See Table 2-10.

The minimum size can be calculated by using the XPMAXPAYLOAD attribute (128, 256, 512, 1024, 2048, or 4096 bytes), and adding overhead for sequence number, redundancy checks, and header information. The overhead is 16 bytes for PCIe packets without ECRC. For example, if XPMAXPAYLOAD is 2048 bytes and ECRC is not used, the minimum Retry buffer size is $2 \times (2048 + 16) = 4128$ bytes. The calculated value is then rounded up to the next available Retry buffer size, which is 8192 bytes in this example.

Table 2-10: **Recommended Minimum Retry Buffer Sizes**

XPMAXPAYLOAD	Minimum Retry Buffer Size
128	4096 bytes
256	4096 bytes
512	4096 bytes
1024	4096 bytes
2048	8192 bytes
4096	16384 bytes

Ports

Table 2-11, Table 2-12, and Table 2-13 define the transmit buffer, receive buffer, and retry buffer ports for the Block RAM interface, respectively.

Table 2-11: **Transmit Buffer Ports**

Port	Direction	Clock Domain	Description
MIMTXBWDATA[63:0]	Output	user_clk	Tx Buffer Write data
MIMTXBWADD[12:0]	Output	user_clk	Tx Buffer Write address
MIMTXBRADD[12:0]	Output	core_clk	Tx Buffer Read address
MIMTXBWEN	Output	user_clk	Tx Buffer Write enable
MIMTXBRDATA[63:0]	Input	core_clk	Tx Buffer Read data
MIMTXBREN	Output	core_clk	Tx Buffer Read enable

Table 2-12: **Receive Buffer Ports**

Port	Direction	Clock Domain	Description
MIMRXBWDATA[63:0]	Output	core_clk	Rx Buffer Write data
MIMRXBWADD[12:0]	Output	core_clk	Rx Buffer Write address
MIMRXBRADD[12:0]	Output	user_clk	Rx Buffer Read address
MIMRXBWEN	Output	core_clk	Rx Buffer Write enable
MIMRXBRDATA[63:0]	Input	user_clk	Rx Buffer Read data
MIMRXBREN	Output	user_clk	Rx Buffer Read enable

Table 2-13: **Retry Buffer Ports**

Port	Direction	Clock Domain	Description
MIMDLLBWDATA[63:0]	Output	core_clk	DLL Retry buffer Write data
MIMDLLBWADD[11:0]	Output	core_clk	DLL Retry buffer Write address
MIMDLLBRADD[11:0]	Output	core_clk	DLL Retry buffer Read address
MIMDLLBWEN	Output	core_clk	DLL Retry buffer Write enable
MIMDLLBRDATA[63:0]	Input	core_clk	DLL Retry buffer Read data
MIMDLLBREN	Output	core_clk	DLL Retry buffer Read enable

Transceiver Interface

Connections between the Transceiver Interface and the GTP transceivers are included in the CORE Generator wrappers. There are eight copies of each of the signals in Table 2-14, one for each lane ($n = 0 .. 7$). If less than eight lanes are used, the lower numbered lanes should be connected to GTP transceivers, starting with lane 0, and the unused input signals should be tied off as indicated. There are two copies of each of the GTP transceiver ports in each GTP_DUAL tile. The 0/1 designation is omitted from Table 2-14 for simplicity.

For multilane designs, lane 0 should be connected to the GTP transceiver channel bonding master.

Table 2-14: **RocketIO GTP Transceiver Interface Ports**

Port	Direction	Clock Domain	Description
PIPERXELECIDLEL*n*	Input	core_clk	Electrical idle detected on receive channel of selected lane. Connect to the RXELECIDLE port on the GTP transceiver or tie High for unused lanes.
PIPERXSTATUSL*n*[2:0]	Input	core_clk	Encodes receiver status and error codes for the received data stream and receiver detection on selected lane. `000`: Data received OK `001`: One skip symbol (SKP) added `010`: One SKP removed `011`: Receiver detected `100`: 8B/10B decode error `101`: Elastic Buffer overflow `110`: Elastic Buffer underflow `111`: Receive disparity error Connect to the RXSTATUS[2:0] ports on the GTP transceiver or tie Low for unused lanes.
PIPERXDATAL*n*[7:0]	Input	core_clk	Receive data. Connect to the RXDATA[7:0] ports on the GTP transceiver or tie Low for unused lanes.
PIPERXDATAKL*n*	Input	core_clk	Control bit(s) for receive data. `0`: Data byte `1`: Control byte Connect to the RXCHARISK[0] port on the GTP transceiver or tie Low for unused lanes.
PIPEPHYSTATUSL*n*	Input	core_clk	Communicates completion of GTP transceiver functions like power management, state transitions, and receiver detection on lane. For state transitions between P0, P0s, and P1, the GTP transceiver indicates a successful transition by a single cycle assertion of PIPEPHYSTATUSL*n*. Connect to the PHYSTATUS port on the GTP transceiver or tie Low for unused lanes.
PIPERXVALIDL*n*	Input	core_clk	Symbol lock and valid data on PIPERXDATAL*n* and PIPERXDATAKL*n*. Connect to the RXVALID port on the GTP transceiver or tie Low for unused lanes.
PIPERXCHANISALIGNEDL*n*	Input	core_clk	Signal from the GTP transceiver elastic buffer. Stays High to denote that the channel is properly aligned with the master transceiver according to observed channel bonding sequences in the data stream. Connect to the RXCHANISALIGNED port on the GTP transceiver or tie Low for unused lanes.
PIPETXDATAL*n*[7:0]	Output	core_clk	Transmit data for selected lane. Connect to the TXDATA[7:0] ports on the GTP transceiver.

Table 2-14: **RocketIO GTP Transceiver Interface Ports *(Continued)***

Port	Direction	Clock Domain	Description
PIPETXDATAKL*n*	Output	core_clk	Control bits for the transmit data. 0: Data byte 1: Control byte Connect to the TXCHARISK[0] port on the GTP transceiver.
PIPETXELECIDLEL*n*	Output	core_clk	Electrical idle requested on transmit channel of selected lane. When 1, selects electrical idle on Transmit channel of selected lane. When 0, indicates that there is valid data on PIPETXDATAL*n*. Connect to the TXELECIDLE port on the GTP transceiver.
PIPETXDETECTRXLOOPBACKL*n*	Output	core_clk	Causes the GTP transceiver on the selected lane to begin receiver detection operation (PIPEPOWERDOWNL*n*=P1) or to begin loopback (PIPEPOWERDOWNL*n*=P0). Connect to the TXDETECTRX port on the GTP transceiver.
PIPETXCOMPLIANCEL*n*	Output	core_clk	When 1, sets the running disparity for the selected lane to negative. (Used when transmitting the compliance pattern). Connect to the TXCHARDISPMODE[0] port on the GTP transceiver. GTP port TXCHARDISPVAL[0] should be tied to 0.
PIPERXPOLARITYL*n*	Output	core_clk	When 1, tells the GTP transceiver on selected lane to do a polarity inversion (on the received data). Connect to the RXPOLARITY port on the GTP transceiver.
PIPEPOWERDOWNL*n*[1:0]	Output	core_clk	Power up/down signal for the transmitter for lane. 00:P0 - Normal Operation 01:P0s - Low recovery time power saving state 10:P1 - Longer recovery time power state 11:P2 - Lowest Power State Connect to the TXPOWERDOWN[1:0] and RXPOWERDOWN[1:0] ports on the GTP transceiver.
PIPEDESKEWLANESL*n*	Output	core_clk	Enable Channel bonding. Not connected to the GTP transceiver.
PIPERESETL*n*	Output	core_clk	Active-High GTP reset. Connect to the RXCDRRESET port on the GTP transceiver.

Power Management Interface

This interface includes ports related to Power Management. Most ports in this interface are tied off by the CORE Generator wrapper.

Table 2-15: **Power Management Ports**

Port	Direction	Clock Domain	Description
L0PWRSTATE0[1:0]	Output	user_clk	Indicates the current device power state as follows: `00`: D0 `01`: D1 `10`: D2 `11`: D3(1) Can be used to inhibit transfers while the block is in the D1 or D2 state.
L0PWRL1STATE	Output	user_clk	Asserted when the link is in the L1 power state.
L0PWRL23READYSTATE	Output	user_clk	Asserted when the link is in the L2/L3 power state.
L0PWRTXL0SSTATE	Output	user_clk	Asserted when Tx Link is in the L0s power state.
L0PWRTURNOFFREQ	Output	user_clk	Asserted when port has received a PMETURNOFF message. The Endpoint block immediately sends a PME_TO_Ack message in response. After this signal is asserted, the user application is guaranteed power for a minimum of 250 ns to prepare and do maintenance tasks before the power is turned off. Afterwards, the power can be turned off by the Root at any time.
L0MACNEWSTATEACK	Output	core_clk	Acknowledgment that the link has transitioned to the requested new link power state.
L0MACRXL0SSTATE	Output	core_clk	Asserted when receiver has gone into the RxL0s state.
L0MACENTEREDL0	Output	core_clk	Pulsed when the MAC transitions back into the L0 link power state.
L0PMEREQIN	Input	core_clk	*Not supported. Must be tied Low.*
L0PMEACK	Output	user_clk	*Not supported.*
L0PMEREQOUT	Output	user_clk	*Not supported.*
L0PMEEN	Output	user_clk	*Not supported.*
L0RXDLLPM	Output	core_clk	Not used. Driven to `0`.
L0RXDLLPMTYPE[2:0]	Output	core_clk	Not used. Driven to `0`.
L0DLLRXACKOUTSTANDING	Output	core_clk	Not used. Driven to `0`.
L0DLLTXOUTSTANDING	Output	core_clk	Not used. Driven to `0`.
L0DLLTXNONFCOUTSTANDING	Output	core_clk	Not used. Driven to `0`.

Notes:

1. When an upstream component programs the integrated block to the D3hot power state, the integrated block transitions into an L1 state. While the integrated block is in the D3hot state, if the upstream component sends a TLP, then the block initiates entry into the L0 state in order to process the incoming TLP and send completions, if necessary. After processing the TLP and sending any relevant completions, the integrated block does not return to the L1 state and remains in L0 state, which is not compliant. To avoid this scenario, the upstream component needs to initiate a D0 transition before sending a TLP and initiate a D3hot transition after receiving any expected completions to send the integrated block back into the D3hot power state.

Configuration and Status Interface

This interface includes control and status, error, backend interface configuration, and interrupt ports. More information on error reporting and user application design considerations can be found in Chapter 4, "Designing with the Endpoint Block." The ports are listed in Table 2-16.

Table 2-16: **Configuration and Status Ports**

Port	Direction	Clock Domain	Description
COMPLIANCEAVOID	Input	core_clk	Modifies the rules for entering POLLING. COMPLIANCE from POLLING.ACTIVE (see Section 4.2.6.2.1 of the *PCI Express Base Specification*). When 0, the block enters POLLING.COMPLIANCE if *any* lane that detected a receiver during Detect has not detected an exit from Electrical Idle since entering POLLING.ACTIVE. When 1, the block enters POLLING.COMPLIANCE if *all* the lanes that detected receivers have not detected exit from Electrical Idle since entering POLLING.ACTIVE. This option is provided to cope with broken lanes in the receive path.
L0FIRSTCFGWRITEOCCURRED	Output	user_clk	Asserted following the completion of the first configuration write after reset.
L0CFGLOOPBACKMASTER	Input	core_clk	Remote device loopback control, used to check the physical connectivity of a link. When asserted, causes the MAC to send training sequences, which put the device at the other end of the link into loopback mode. The remote device then loops back all packets sent by this device until L0CFGLOOPBACKMASTER is deasserted, causing the link to retrain.
L0CFGLOOPBACKACK	Output	core_clk	Asserted after the block has entered the Loopback master state.
L0RXMACLINKERROR[1:0]	Output	core_clk	Used to report link errors. Bit 1 asserted indicates a receiver error. Bit 0 asserted indicates a link training error.
L0MACLINKUP	Output	core_clk	Driven High when link training has completed and the link is operational.

Table 2-16: **Configuration and Status Ports *(Continued)***

Port	Direction	Clock Domain	Description
L0MACNEGOTIATEDLINKWIDTH[3:0]	Output	core_clk	Link width selected after negotiation, as follows: `0001`: One lane `0010`: Two lanes `0100`: Four lanes `1000`: Eight lanes
L0MACLINKTRAINING	Output	core_clk	Indicates that link training is in progress. Reset to logic `1`. Goes Low when the link reaches the L0 state at the end of link training. If link training fails, the signal is pulsed Low for one clock cycle before the block reenters the detect state.
L0LTSSMSTATE[3:0]	Output	core_clk	The state of the Link Training and Status State Machine, encoded as follows: `0000`: Initial `0001`: Detect `0010`: Polling `0011`: Configuration `0100`: L0 `0101`: L0sTx `0110`: L1 `0111`: L2 `1000`: Testmode Wait `1001`: Loopback `1010`: Hot Reset `1011`: Disabled `1100`: Recovery `1101`: L0 to Recovery `1110`: L0 to L0sTx `1111`: L0 to L1/L2 *Note:* The encodings `1101`, `1110`, and `1111`, corresponding to L0 transition states, identify where the device is still nominally in the L0 state but cannot transmit data.
L0DLLVCSTATUS[7:0]	Output	core_clk	Indicates the flow control initialization process for the corresponding VC is complete. A `1` indicates the VC is initialized. Bit [0]: VC0 status Bit [1]: VC1 status Bits [7:2]: Reserved

Table 2-16: **Configuration and Status Ports *(Continued)***

Port	Direction	Clock Domain	Description
L0DLUPDOWN[7:0]	Output	core_clk	When operating as a Endpoint in a PCIe design, a rising edge on this signal flags the transition from the InitFC1 phase of the initialization procedure to the InitFC2 phase. A falling edge indicates that the link has been broken and is used as a trigger for the link to be reset. There is a bit for each implemented VC. Bits [7:2] are never used.
L0CFGDISABLESCRAMBLE	Input	core_clk	When asserted at power-up, disables the MAC scrambler. Provided for use in debugging communication issues between linked devices after physical connectivity has been confirmed using loopback (see L0CFGLOOPBACKMASTER).
L0DLLERRORVECTOR[6:0]	Output	core_clk	Error signals relating to DLL data. Errors detected within the DLL result in the relevant bit in the vector being asserted for one clock period: bit 0: DLLBADTLP bit 1: DLLBADDLLP bit 2: DLLREPLAYTIMEOUT bit 3: DLLREPLAYROLLOVER bit 4: *Reserved* bit 5: RXTLPMISSING bit 6: DLLPROTOCOLERROR A missing TLP triggers both bit 5 and bit 0. The same error in consecutive DLLPs results in the associated bit being asserted for more than one clock period where the DLLPs are presented back-to-back to the Data Link Layer by the MAC.
L0COMPLETERID[12:0]	Output	user_clk	Bus number and device number components of the Completer ID. Append the 3-bit Function number, 0, to form the full 16-bit Completer ID. Also used as the Requester ID for Request TLPs.
L0TRANSACTIONSPENDING	Input	user_clk	Assert when there are outstanding transactions pending. Setting reflected in setting of the Transaction Pending bit in the Device Status register.

Table 2-16: **Configuration and Status Ports *(Continued)***

Port	Direction	Clock Domain	Description
L0SETCOMPLETERABORTERROR	Input	user_clk	When asserted, causes the relevant Completer Abort status bit(s) to be set to `1`.
L0SETDETECTEDCORRERROR	Input	user_clk	When asserted, causes the relevant Correctable Error status bit(s) to be set to `1`. If bit 0 of the Device Control Register is set (Correctable Error Reporting Enable), then a Correctable Error Message is also sent.
L0SETDETECTEDFATALERROR	Input	user_clk	When asserted, causes the relevant Fatal Error status bit(s) to be set to `1`. If bit 2 of the Device Control Register is set (Fatal Error Reporting Enable) or bit 8 of the Command Register is set (SERR Enable), then a Fatal Error Message is also sent.
L0SETDETECTEDNONFATALERROR	Input	user_clk	When asserted, causes the relevant Nonfatal Error status bit(s) to be set to `1`. If bit 1 of the Device Control Register is set (Non-Fatal Error Reporting Enable) or bit 8 of the Command Register is set (SERR Enable), then a Non-Fatal Error Message is also sent.
L0SETUSERDETECTEDPARITYERROR	Input	user_clk	When asserted, causes the relevant Parity Error status bit(s) to be set to `1`.
L0SETUSERMASTERDATAPARITY	Input	user_clk	When asserted, causes the relevant Master Data Parity status bit(s) to be set to `1`.
L0SETUSERRECEIVEDMASTERABORT	Input	user_clk	When asserted, causes the relevant Master Abort status bit(s) to be set to `1`.
L0SETUSERRECEIVEDTARGETABORT	Input	user_clk	When asserted, causes the relevant Target Abort status bit(s) to be set to `1`.
L0SETUSERSYSTEMERROR	Input	user_clk	When asserted, causes the relevant System Error status bit(s) to be set to `1`.
L0SETUSERSIGNALLEDTARGETABORT	Input	user_clk	When asserted, causes the relevant Target Abort status bit(s) to be set to `1`.
L0SETCOMPLETIONTIMEOUTUNCORRERROR	Input	user_clk	Asserted to indicate that a requester has not seen a completion and has handled this as an Uncorrectable Error. Causes the relevant "Completion Timeout" status bit(s) to be set to `1`.

Table 2-16: **Configuration and Status Ports *(Continued)***

Port	Direction	Clock Domain	Description
L0SETCOMPLETIONTIMEOUTCORRERROR	Input	user_clk	Asserted to indicate that a requester has not seen a completion and has handled this as a Correctable Error. Causes the relevant "Completion Timeout" status bit(s) to be set to 1.
L0SETUNEXPECTEDCOMPLETIONUNCORRERROR	Input	user_clk	Asserted to indicate that a receiver has received an unexpected completion and has handled this as an Uncorrectable Error. Causes the relevant Unexpected Completion status bit(s) to be set to 1.
L0SETUNEXPECTEDCOMPLETIONCORRERROR	Input	user_clk	Asserted to indicate that a receiver has received an unexpected completion and has handled this as a Correctable Error. Causes the relevant Unexpected Completion status bit(s) to be set to 1.
L0SETUNSUPPORTEDREQUESTNONPOSTEDERROR	Input	user_clk	Asserted to indicate that a completer has received an unsupported non-posted request. Causes the relevant unsupported request status bit(s) to be set to 1.
L0SETUNSUPPORTEDREQUESTOTHERERROR	Input	user_clk	Asserted to indicate that a completer has received some other kind of unsupported request (other than a non-posted request). Causes the relevant unsupported request status bit(s) to be set to 1.
L0LEGACYINTFUNCT0	Input	user_clk	Drive High to request Legacy Interrupt on Function 0.
L0MSIREQUEST0[3:0]	Input	user_clk	*Not supported. Must be tied Low.*
L0MSIENABLE0	Output	user_clk	Asserted when MSI is enabled for Function 0.
L0MULTIMSGEN0[2:0]	Output	user_clk	Asserted when MSI multiple messages are enabled for Function 0.
L0STATSDLLPRECEIVED	Output	core_clk	Asserted for a single clock cycle when a DLLP is received.
L0STATSDLLPTRANSMITTED	Output	core_clk	Asserted for a single clock cycle when a DLLP is transmitted.
L0STATSOSRECEIVED	Output	core_clk	Asserted for a single clock cycle when an ordered set is received.
L0STATSOSTRANSMITTED	Output	core_clk	Asserted for a single clock cycle when an ordered set is transmitted.
L0STATSTLPRECEIVED	Output	core_clk	Asserted for a single clock cycle when a TLP is received.

Table 2-16: **Configuration and Status Ports *(Continued)***

Port	Direction	Clock Domain	Description
L0STATSTLPTRANSMITTED	Output	core_clk	Asserted for a single clock cycle when a TLP is transmitted.
L0STATSCFGRECEIVED	Output	user_clk	Asserted for a single cycle of CRMUSERCLK when a configuration packet is received by the configuration block.
L0STATSCFGTRANSMITTED	Output	user_clk	Asserted for a single cycle of CRMUSERCLK when a configuration packet is transmitted by the configuration block.
L0STATSCFGOTHERRECEIVED	Output	user_clk	Asserted for a single cycle of CRMUSERCLK when a packet of any other type (e.g., a message packet or a posted memory write packet relating to MSI) is received by the configuration block.
L0STATSCFGOTHERTRANSMITTED	Output	user_clk	Asserted for a single cycle of CRMUSERCLK when one of these other types of packet is transmitted by the configuration block.
IOSPACEENABLE	Output	user_clk	I/O space enable. When `1`, shows that response to I/O request packets has been enabled.
MEMSPACEENABLE	Output	user_clk	Memory space enable. When `1`, response to memory request packets has been enabled.
MAXPAYLOADSIZE[2:0]	Output	user_clk	Negotiated Max Payload size, as follows: `000:` 128 bytes `001:` 256 bytes `010:` 512 bytes `011:` 1024 bytes `100:` 2048 bytes `101:` 4096 bytes `110:` *Reserved* `111:` *Reserved*
MAXREADREQUESTSIZE[2:0]	Output	user_clk	Negotiated Read request size, as follows: `000:` 128 bytes `001:` 256 bytes `010:` 512 bytes `011:` 1024 bytes `100:` 2048 bytes `101:` 4096 bytes `110:` *Reserved* `111:` *Reserved*

Table 2-16: **Configuration and Status Ports *(Continued)***

Port	Direction	Clock Domain	Description
BUSMASTERENABLE	Output	user_clk	Bus Master Enable. When 0, Endpoint is prevented from issuing any memory or I/O requests.
PARITYERRORRESPONSE	Output	user_clk	Parity Error Response. When 1, response to Master Data parity errors has been enabled.
SERRENABLE	Output	user_clk	SERR Enable. When 1, reporting of fatal and nonfatal errors has been enabled.
INTERRUPTDISABLE	Output	user_clk	Interrupt Disable. When 1, device is prevented from generating INTx interrupt messages.
URREPORTINGENABLE	Output	user_clk	Unsupported request reporting enable. When 1, reporting of unsupported requests has been enabled.
AUXPOWER	Input	user_clk	*Not supported. Must be tied Low.*
DLLTXPMDLLPOUTSTANDING	Output	core_clk	Not used. Driven to 0.
L0UNLOCKRECEIVED	Output	user_clk	*Not supported.*
L0PACKETHEADERFROMUSER[127:0]	Input	user_clk	*Not supported. Must be tied Low.*

Registers

The tables in this section describe the registers in the Endpoint block. All registers can be read through the Management interface, and those designated read/write (RW) can also be written. Note that the addresses given in the following tables refer to the Management interface address (MGMTADDR[10:0]). The addresses used when accessing the configuration registers through configuration read and write packets are different, and can be found in the PCI-SIG specifications.

Legacy Configuration Registers (Type 0)

Further documentation on each of the registers in the following tables can be found in the appropriate specifications on the PCI-SIG website (www.pcisig.com). The registers are read on MGMTRDATA[31:0] or written to MGMTWDATA[31:0].

Table 2-17: **Legacy Configuration Registers**

Management Address (Hex) MGMTADDR[10:0]	Register Name(1)	Read Only or Read Write
0	Device ID; Vendor ID	RW; RW
1	Status; Command	RO; RO
2	Class Code; Revision ID	RW; RW
3	Header Type; Cache Line Size	RO; RO
4	Base Address Registers (BAR0)	RO
5	Base Address Registers (BAR1)	RO
6	Base Address Registers (BAR2)	RO
7	Base Address Registers (BAR3)	RO
8	Base Address Registers (BAR4)	RO
9	Base Address Registers (BAR5)	RO
A	Cardbus CIS Pointer	RW
B	Subsystem ID; Subsystem Vendor ID	RW; RW
C	Expansion ROM Base Address	RO
D	Interrupt Pin; Interrupt Line; Capabilities Pointer	RW; RO; RW
E	base_addr0_mask(2)	RO
F	base_addr1_mask(2)	RO
10	rom_base_addr_mask	RO
11	base_addr2_mask(2)	RO
12	base_addr3_mask(2)	RO
13	base_addr4_mask(2)	RO
14	base_addr5_mask(2)	RO
15	Reserved	RO

Table 2-17: **Legacy Configuration Registers *(Continued)***

Management Address (Hex) MGMTADDR[10:0]	Register Name[1]	Read Only or Read Write
`16`	Reserved	RO
`17`	Reserved	RO
`18`	Reserved	RO
`19`	Reserved	RO
`1A`	Reserved	RO
`1B`	Reserved	RO
`1C`	Reserved	RO

Notes:

1. The register names are listed as they are read on MGMTRDATA[31:0] or written to MGMTWDATA[31:0].
2. The number of Base Address registers implemented depends on the BARnEXIST attribute settings, while the width of the Address range allocated by the host depends on the Base Address Register Mask, set from the BARnMASKWIDTH attributes.

Power Management Capability Registers

Table 2-18 summarizes the Power Management Capability Structure registers.

Table 2-18: **Power Management Capability Structure**

Management Address (Hex) MGMTADDR[10:0]	Register Name[1]	Read Only or Read Write
`1D`	Power Management Capabilities (PMC)[2]; Next Capability Pointer; Capability ID	RW; RW; RO
`1E`	Reserved (8 bits); Reserved (8 bits); Power Management Control/Status (PMCSR)	N/A; N/A; RO
`1F`	Reserved	N/A
`20`	Reserved	N/A
`21`	Reserved	N/A

Notes:

1. The register names are listed as they are read on MGMTRDATA[31:0] or written to MGMTWDATA[31:0].
2. The PM version correctly has a value of 3 when read through the PCI Express link, but returns a value of 2 when read through the Management interface.

Message Signaled Interrupt (MSI) Capability Structure

Table 2-19 summarizes the MSI registers.

Table 2-19: **MSI Registers**

Management Address (Hex) MGMTADDR[10:0]	Register Name(1)	Read Only or Read Write
22	Message Control; Next Pointer; Capability ID	RW; RW; RO
23	Message Address	RO
24	Message Upper Address	RO
25	Reserved (16 bits); Message Data (16 bits)	RO; RO
26	Mask Bits	RO
27	Pending Bits	RO

Notes:

1. The register names are listed as they are read on MGMTRDATA[31:0] or written to MGMTWDATA[31:0].

PCI Express Capability Structure

Table 2-20 summarizes the PCI Express Capability registers.

Table 2-20: **PCI Express Capability Registers**

Management Address (Hex) MGMTADDR[10:0]	Register Name(1)	Read Only or Read Write
28	PCIe Capabilities Register; Next Cap Pointer; PCIe Cap ID	RW; RW; RO
29	Device Capabilities	RW
2A	Device Status; Device Control	RO; RO
2B	Link Capabilities(2)	RW
2C	Link Status(2); Link Control	RW; RO
2D	Reserved	N/A
2E	Reserved	N/A
2F	Reserved	N/A
30	Reserved	N/A

Notes:

1. The register names are listed as they are read on MGMTRDATA[31:0] or written to MGMTWDATA[31:0].
2. Bit 20 of the Link Capabilities register (Data Link Layer Active Reporting Capable) and bit 13 of the Link Status register (Data Link Layer Link Active) both have a correct value of 0 when read through the PCI Express serial link, but return a value of 1 when read from the Management interface.

Reserved Registers

Table 2-21 summarizes the reserved register range.

Table 2-21: **Reserved Registers**

Management Address (Hex) MGMTADDR[10:0]	Register Name[1]	Read Only or Read Write
31 - 45	Address Range is Reserved	N/A

Notes:
1. The register names are listed as they are read on MGMTRDATA[31:0] or written to MGMTWDATA[31:0].

Device Serial Number Capability Structure

Table 2-22 summarizes the Device Serial Number registers.

Table 2-22: **Device Serial Number Registers**

Management Address (Hex) MGMTADDR[10:0]	Register Name[1]	Read Only or Read Write
46	PCIe Enhanced Capability Header	RW
47	Serial Number Register (Lower DW)	RW
48	Serial Number Register (Upper DW)	RW

Notes:
1. The register names are listed as they are read on MGMTRDATA[31:0] or written to MGMTWDATA[31:0].

PCI Express Virtual Channel Capability Structure

Table 2-23 summarizes the Virtual Channel Capability Structure registers.

Table 2-23: **Virtual Channel Capabilities Registers**

Management Address (Hex) MGMTADDR[10:0]	Register Name[1]	Read Only or Read Write
49	PCIe Enhanced Capability Header	RW
4A	Port VC Capability Register 1	RW
4B	Port VC Capability Register 2	RW
4C	Port VC Status Register; Port VC Control Register	RO; RO
4D	VC Resource Capability Register (0)	RO
4E	VC Resource Control Register (0)	RO
4F	VC Resource Status Register (0); RsvdP	RO
50	VC Resource Capability Register (1)	RO
51	VC Resource Control Register (1)	RO
52	VC Resource Status Register (1); RsvdP	RO

Table 2-23: **Virtual Channel Capabilities Registers *(Continued)***

Management Address (Hex) MGMTADDR[10:0]	Register Name[1]	Read Only or Read Write
53	VC Arbitration Table (1)	RO
54	VC Arbitration Table (2)	RO
55	VC Arbitration Table (3)	RO
56	VC Arbitration Table (4)	RO
57-3FF	Reserved	

Notes:

1. The register names are listed as they are read on MGMTRDATA[31:0] or written to MGMTWDATA[31:0].

Management Control and Status Registers

The Management Control and Status registers are loaded with the attribute settings at power-on reset and can be read or overridden through the Management Interface. Attribute registers that can be written through address `0x400` and above correspond to registers that are either read only in the PCI Express Configuration Space or are unrelated to the PCI Express Configuration Space. See Appendix A, "Endpoint Block Attributes" for attribute details and "Management Interface" in Chapter 2 for details of operation of the Management Interface.

Table 2-24: **Management Control and Status Registers**

Management Address (Hex) MGMTADDR[10:0]	Bit Position	Attribute Name	Read Only or Read Write
400	10:0	Reserved	
	11	Reserved	
401	2:0	RETRYRAMWRITELATENCY	RW
	5:3	RETRYRAMREADLATENCY	RW
	17:6	RETRYRAMSIZE	RW
	18	Reserved	
	19	Reserved	
	20	Reserved	
	21	Reserved	
	24:22	TLRAMWRITELATENCY	RW
	27:25	TLRAMREADLATENCY	RW

Table 2-24: **Management Control and Status Registers *(Continued)***

Management Address (Hex) MGMTADDR[10:0]	Bit Position	Attribute Name	Read Only or Read Write
402	0	Reserved	
	8:1	TXTSNFTSCOMCLK	RW
	16:9	TXTSNFTS	RW
	17	Reserved	
	20:18	L1EXITLATENCYCOMCLK	RW
	23:21	L1EXITLATENCY	RW
	26:24	L0SEXITLATENCYCOMCLK	RW
	29:27	L0SEXITLATENCY	RW
403	11:0	Reserved	
	19:12	Reserved	
	22:20	Reserved	
	23	Reserved	
	26:24	LOWPRIORITYVCCOUNT	RW
404	0	Reserved	
	3:1	XPMAXPAYLOAD	RW
	11:4	ACTIVELANESIN	RW
	12	INFINITECOMPLETIONS	RW
	20:13	Reserved	
	24:21	XPDEVICEPORTTYPE	RW
	25	Reserved	
	26	Reserved	
	27	Reserved	
	28	Reserved	
	29	Reserved	
405	5:0	BAR0MASKWIDTH	RW
	11:6	BAR1MASKWIDTH	RW
	17:12	BAR2MASKWIDTH	RW
406	5:0	BAR3MASKWIDTH	RW
	11:6	BAR4MASKWIDTH	RW
	17:12	BAR5MASKWIDTH	RW

Table 2-24: **Management Control and Status Registers *(Continued)***

Management Address (Hex) MGMTADDR[10:0]	Bit Position	Attribute Name	Read Only or Read Write
`407`	0	BAR0IOMEMN	RW
	1	BAR1IOMEMN	RW
	2	BAR2IOMEMN	RW
	3	BAR3IOMEMN	RW
	4	BAR4IOMEMN	RW
	5	BAR5IOMEMN	RW
`408`	0	BAR0PREFETCHABLE	RW
	1	BAR1PREFETCHABLE	RW
	2	BAR2PREFETCHABLE	RW
	3	BAR3PREFETCHABLE	RW
	4	BAR4PREFETCHABLE	RW
	5	BAR5PREFETCHABLE	RW
`409`	0	BAR0ADDRWIDTH	RW
	1	BAR1ADDRWIDTH	RW
	2	BAR2ADDRWIDTH	RW
	3	BAR3ADDRWIDTH	RW
	4	BAR4ADDRWIDTH	RW
	5	Reserved	
`40A`	0	BAR0EXIST	RW
	1	BAR1EXIST	RW
	2	BAR2EXIST	RW
	3	BAR3EXIST	RW
	4	BAR4EXIST	RW
	5	BAR5EXIST	RW
`40B`	12:0	VC1RXFIFOLIMITC	RW
	13	Reserved	
	26:14	VC1RXFIFOLIMITNP	RW
	27	Reserved	
`40C`	12:0	VC1RXFIFOLIMITP	RW
	13	Reserved	
	26:14	VC1RXFIFOBASEC	RW
	27	Reserved	

Table 2-24: **Management Control and Status Registers *(Continued)***

Management Address (Hex) MGMTADDR[10:0]	Bit Position	Attribute Name	Read Only or Read Write
40D	12:0	VC1RXFIFOBASENP	RW
	13	Reserved	
	26:14	VC1RXFIFOBASEP	RW
	27	Reserved	
40E	10:0	VC1TOTALCREDITSCD	RW
	21:11	VC1TOTALCREDITSPD	RW
40F	6:0	VC1TOTALCREDITSCH	RW
	13:7	VC1TOTALCREDITSNPH	RW
	20:14	VC1TOTALCREDITSPH	RW
410	12:0	VC1TXFIFOLIMITC	RW
	13	Reserved	
	26:14	VC1TXFIFOLIMITNP	RW
	27	Reserved	
411	12:0	VC1TXFIFOLIMITP	RW
	13	Reserved	
	26:14	VC1TXFIFOBASEC	RW
	27	Reserved	
412	12:0	VC1TXFIFOBASENP	RW
	13	Reserved	
	26:14	VC1TXFIFOBASEP	RW
	27	Reserved	
413	12:0	VC0RXFIFOLIMITC	RW
	13	Reserved	
	26:14	VC0RXFIFOLIMITNP	RW
	27	Reserved	
414	12:0	VC0RXFIFOLIMITP	RW
	13	Reserved	
	26:14	VC0RXFIFOBASEC	RW
	27	Reserved	

Table 2-24: **Management Control and Status Registers *(Continued)***

Management Address (Hex) MGMTADDR[10:0]	Bit Position	Attribute Name	Read Only or Read Write
415	12:0	VC0RXFIFOBASENP	RW
	13	Reserved	
	26:14	VC0RXFIFOBASEP	RW
	27	Reserved	
416	10:0	VC0TOTALCREDITSCD	RW
	21:11	VC0TOTALCREDITSPD	RW
417	6:0	VC0TOTALCREDITSCH	RW
	13:7	VC0TOTALCREDITSNPH	RW
	20:14	VC0TOTALCREDITSPH	RW
418	12:0	VC0TXFIFOLIMITC	RW
	13	Reserved	
	26:14	VC0TXFIFOLIMITNP	RW
	27	Reserved	
419	12:0	VC0TXFIFOLIMITP	RW
	13	Reserved	
	26:14	VC0TXFIFOBASEC	RW
	27	Reserved	
41A	12:0	VC0TXFIFOBASENP	RW
	13	Reserved	
	26:14	VC0TXFIFOBASEP	RW
	27	Reserved	
41B	7:0	XPBASEPTR	RW
	19:8	VCBASEPTR	RW
	31:20	PMBASEPTR	RW
41C	11:0	PBBASEPTR	RW
	23:12	MSIBASEPTR	RW
41D	11:0	DSNBASEPTR	RW
	23:12	AERBASEPTR	RW
41E .. 7FF		Reserved	

Chapter 3

Designing with LogiCORE IP for the Endpoint Block

Xilinx recommends using the LogiCORE™ Endpoint Block Plus for PCI Express® designs available in the CORE Generator™ tool. This provides a wrapper around the Virtex-5 FPGA's integrated Endpoint block and automatically connects the block RAMs, GTP transceivers, and reset and clock modules. The wrapper provides an easy to use interface which simplifies system design. In addition, certain features related to compliance and performance are included in the wrapper.

For advanced users using the native interface of the integrated Endpoint block, the LogiCORE Endpoint Block for PCI Express designs is available. This wrapper does not include the features or workarounds provided with the Block Plus wrapper. The user is expected to implement these workarounds in their designs. More information including data sheets and user guides are available at http://www.xilinx.com/pciexpress.

Chapter 4

Designing with the Endpoint Block

Summary

This chapter presents information related to designing with the Virtex-5 FPGA Endpoint block. The sections include:

Expansion ROM

The Endpoint block implements an expansion ROM with a fixed size of 1 MB.

Flow Control

Under the PCIe flow control system, each VC maintains its own flow control credit pool with separate credit accounts being recorded for posted requests, non-posted requests, and completions. Each VC has a separate receive FIFO for posted, non-posted, and completion packets.

The Endpoint block fully implements the rules given in the *PCI Express Base Specification*. Flow control information is passed from the receiving device to the transmitting device as further credits become available as a result of sent data being read at the receiver. This involves the receiver side of the Transaction Layer block instructing the Data Link Layer to send Update Flow Control (UpdateFC) credit value packets to the device at the other end of the link, normally after each packet is received. The flow control process is automatically handled by the Endpoint block.

Following the training sequences sent when the Integrated Endpoint port is first connected to another device, the Data Link Layer automatically initializes the process by sending the requisite flow control initialization packets for VC0 across the link. VC0 is initialized first because this is the only VC that is enabled by default. The Data Link Layer also receives flow control initialization packets advertising the number of credits available in the receive side of the device at the other end of the link. As further VCs are enabled by software, the Data Link Layer initializes these VCs with independent flow control credit values for each VC.

For the system to work, the devices at each end of the link must behave correctly and never transmit more data to the Receiver's Rx buffers than it has been told can be accepted. To ensure this does not happen in the Endpoint block, the Transaction Layer (TL) keeps a count of the header and data credits consumed by the packets of each type that it has transmitted. Every time the TL transmits a packet, it increments the count of credits consumed so far. When the transmitted data has exhausted the current credit limit held by the transmitter (the last credit value to be received from the device at the other end of the link), it halts transmission of new data of that type until it receives an UpdateFC packet.

The number of flow control credits that the Endpoint block initially advertises for each of the receive FIFOs must be set through various attributes, depending on how much buffer space is allocated to the receiver. These attributes need to be set to record the number of packets of the selected type that can be handled by the corresponding FIFO, considering both the size of the FIFO and the overriding maximum of eight packets that can be buffered by any FIFO. The initial flow control attribute values are automatically calculated and set in the CORE Generator wrappers based on buffer sizes chosen through the CORE Generator GUI. See Appendix A for more information.

When working as the receiver, the Endpoint block counts the number of credits of each type consumed as it routes each incoming packet to the appropriate Rx buffer. The credit count is used for error checking since an overflow error will occur if the number of credits consumed ever exceeds the number of credits advertised to the device at the other end of the link. If this happens, the Endpoint block flags the error to the configuration block, which then sends an error message packet to the host.

Note: The transmission of UpdateFC packets is monitored via a timer in each VC that is reset each time an UpdateFC packet is sent. If this timer expires, transmission of new Transaction Layer packets is halted in the Data Link Layer while the UpdateFC packet can be sent.

The four flow control registers detailed in the *PCI Express Base Specification* (CREDITS_CONSUMED, CREDIT_LIMIT, CREDITS_ALLOCATED, and the optional CREDITS_RECEIVED) are all implemented by the Endpoint block. The values of these registers, for every VC, can be read using the MGMTSTATSCREDITSEL and MGMTSTATSCREDIT buses in the Management Interface.

Note: The optional timeout mechanism detailed in Section 2.6.1.2 of the *PCI Express Base Specification* is implemented inside the Transaction Layer.

Handling Inbound Completion Packets

Inbound completion packets can arrive at any time after the corresponding request within the completion timeout. The *PCI Express Base Specification* states that for Endpoint devices, inbound completion packets must be given infinite flow control credits – which effectively means that there is no flow control on these items. This is reflected in the setting of the INFINITECOMPLETIONS attribute.

There is not, however, infinite memory available into which completion packets are received. The intention behind the *PCI Express Base Specification* is that the requester only issues requests for completions that it can handle. Therefore, the user must ensure that further requests requiring completions are issued only when there is sufficient buffer space to accept these new completions along with any outstanding completions. If completions are received without sufficient buffer space, the receiver overflows and requires a reset.

Two strategies are useful here. One is to keep careful account of the space required by existing completions (since split completions are allowed, a single non-posted request could result in several completion packets). The other is to ensure that completions and posted packets are removed from the buffer as quickly as they arrive. If posted packets are not removed as quickly as they arrive, an older posted packet could block removal of completion packets because of PCIe transaction ordering rules. The second method could be difficult in many applications, particularly for x8 applications, x4 applications with a 125 MHz user_clk, or x2 applications with a 62.5 MHz user_clk.

The Rx completion FIFO must be sized as defined in the descriptions for the VC0RXFIFOLIMITC and VC1RXFIFOLIMITC attributes. The FIFO can hold a maximum of eight packets, regardless of the size of the packets or the size of the FIFO. When the Rx completion FIFO is sized as required, it can hold eight packets with a payload up to the MAXPAYLOAD attribute setting.

The amount of space to allocate for the completion data associated with any packet can be calculated as follows:

- I/O Writes: A single completion packet, consisting of a 3 DW header. Each DW is 32 bits, but packets must be aligned to the 64-bit wide buffers. The buffer space required is rounded up to 4 DW.
- I/O Reads: A single completion packet, requiring three 32-bit DWs of space for the header plus one 32-bit DW for the data for a total of 4 DWs.
- Memory Reads: Multiple completion packets, each requiring sufficient buffer space for the header and data, with the total rounded up to a multiple of 64 bits (8 bytes). A completer can split a completion along the Read Completion Boundary (RCB), which can be either 64 or 128 bytes.

The Endpoint block can use completion flow control, but this choice should only be made if the user knows that the link partner is capable of receiving completion flow control updates from an Endpoint.

Performance Considerations

To obtain maximum throughput performance on the TLI for completions, the user application must drain completion packets at the same rate as they arrive in the Rx FIFO. As described earlier, this task is difficult when the line rate of the TLI matches the rate of completion packets arriving at the Rx FIFO. The user should use the LLKRXDSTCONTREQN pin and implement fabric logic to monitor the state of the Rx FIFO in order to drain packets at the line rate. The user should have prior knowledge of the traffic pattern of TLPs arriving at the RX FIFO to accomplish this. If a long burst of small completions arrives while a large TLP is being read from the RX FIFO, overflow can still occur, requiring a reset. The user should guarantee that a large TLP is not followed by a burst of completions whose size is less than one-fourth the size of the large TLP to prevent overflow. The usage of the LLKRXDSTCONTREQN pin and the supporting fabric logic is implemented in v1.2 of LogiCORE Endpoint Block Plus for PCI Express designs available in the Xilinx CORE Generator tool. Refer to the *LogiCORE Block Plus User Guide for PCI Express* for further details.

Traffic Class to Virtual Channel Mapping

Transactions are conducted through the VCs implemented within the Endpoint block. However, user applications using the Endpoint block simply specify the traffic class of each transaction they conduct.

Each TLP transmitted over a PCIe network is assigned to one of the eight possible traffic classes (TC0 through TC7). Within each device in the network, if the VC capability is enabled, each traffic class can be assigned to one of the VCs implemented within that device. Where the device just includes one VC, all eight traffic classes can be assigned to that one channel. If two VCs are enabled, each VC could have one or more traffic classes assigned to it. When the VC capability is not enabled in the device, it only transmits requests for one traffic class (TC0) and one virtual channel (VC0). It must transmit completions using the TC of the received request, however.

The scheme used in assigning traffic classes to VCs is up to the Root Complex. The only restriction on it is that TC0 is mapped to VC0 (and no other VC), though it is conventional to keep the traffic classes in the order TC0 to TC7 across the VC0 and VC1 channels to which they are mapped. It should also be noted that while more than one traffic class can be mapped to a VC, each traffic class can only be mapped to a single VC.

There is no requirement for all eight traffic classes to be assigned to a specific VC. If an Endpoint, for example, should only receive TLPs belonging to a limited range of traffic classes, just the required traffic classes should be mapped to this Endpoint's VCs. Then any TLP that is received that has been assigned to a different traffic class is treated as a malformed TLP, triggering the appropriate error handling procedure.

The mapping chosen (along with the selected VC arbitration method; see "Virtual Channel Arbitration," page 69) affects the quality of service that is achieved for each traffic class. Where more than one traffic class is mapped to the same VC, the speed of delivery of TLPs assigned to these traffic classes is the same for each of these traffic classes. Any delay on this VC affects all of these traffic classes. Where a traffic class is mapped to a separate channel, a higher quality of service can be achieved as a result of bypassing delays affecting TLPs in other traffic classes.

The required mapping is recorded in the VC Resource Control registers within the registers of the VC Capability structure. This structure is required to be included among the extended configuration registers in any PCIe device that offers more than one VC or more than one traffic class. A separate copy of this register is included for each VC that is implemented in the design. Each copy of this register records which of the traffic classes use that selected channel. This mapping is set through configuration writes to the appropriate registers (all of which are required to be sent using TC0, which is automatically assigned to VC0).

Within the Endpoint block, all packets are handled on their VCs, but they are taken from and presented to the user according to their traffic class. Thus, there is no need for the user application to process packets in terms of anything other than their traffic class.

Operation as a Transaction Requester

Two types of transactions are executed when the Endpoint block is acting as a transaction requester:

- Reads
- Posted writes
- Non-posted writes

Reads can be made from memory space. Posted writes are used for normal memory data and messages. Legacy Endpoints can issue I/O write requests, which are non-posted. Endpoints for PCI Express cannot issue non-posted write requests (I/O or configuration writes).

In each case, the user logic or software needs to:

- Generate the appropriate request
- If a response is required, assign a unique tag to the request (for tracking purposes – see below)
- Then download the data through the block's Transaction Layer (TL) interface

When directed by the user logic, the TL first stores the Transaction Layer Packet (TLP) in the appropriate Tx buffer then schedules it for transmission in relation to the other TLPs that are waiting. When its turn comes, the TL passes the TLP to the Data Link Layer via the appropriate VC for the selected traffic class.

The Tx buffer to which the request must be written depends on the type of transaction. Read requests are written to the Tx non-posted buffer. Messages or posted writes are written to the Tx posted buffer.

When a completion packet is received in response to the transaction request, the Endpoint block places this packet in the Rx completion buffer and then signals to the user logic that this packet is received. When the user logic reads the contents of this packet, it matches the tag in its header against the tags of the outstanding requests to identify which request it represents the response to.

A peer device may issue multiple completions in response to a single memory read request. A single request can be split into multiple completions at address boundaries that are integer multiples of the peer's read completion boundary (RCB) setting. The RCB setting can be either 64 or 128 bytes. All completions split in this way are returned to the requestor in increasing address order.

When a requestor issues multiple separate requests, the completions for these requests might not return in the same order as the requests were issued. The requestor must use the tag of the incoming completion to determine the matching request.

The action of reading automatically clears the buffer and releases flow control credits. The release of these flow control credits is automatically communicated to the device at the other end of the link.

The PCI Express specification requires that transaction requestors implement a completion time-out mechanism to prevent deadlock if a completion fails to return. A timer must be started for each request that requires one or more completions. The timer should be started when the request is issued. The timer is normally cleared when all completions associated with the request are received. If the timer reaches a preset value before being cleared then a Completion Timeout has occurred. The specification requires that the value used to trigger a completion time-out is between 50 µs and 50 ms. It is strongly recommended that this value is greater then 10 ms.

If a completion time-out occurs the requestor can optionally re-issue the request a finite number of times. There must be a limit to the number of times a request is re-issued. When this limit is reached and the device no longer re-issues the request, it must signal an uncorrectable error by asserting L0SETCOMPLETIONTIMEOUTUNCORRERROR.

It is acceptable for a requestor not to attempt a re-issue in which case it should assert L0SETCOMPLETIONTIMEOUTUNCORRERROR when the initial Completion Timeout occurs.

Operation as a Transaction Completer

The types of transactions that the Endpoint block might need to service when acting as a transaction completer are:

- Reads
- Posted writes
- Non-posted writes

In each case, the Endpoint block:

- Receives the packet over the serial link
- Reads the header to determine the packet type
- Places the packet contents in the appropriate Rx buffer for the type of packet
- Passes the messages and configuration requests claimed internally by the Endpoint block to the Configuration and Capabilities module

The user logic is then advised that a packet has been placed in this buffer. The Rx buffer to which the request is written depends on the type of transaction. Read requests and non-posted writes (traffic class 0 only) are written to the Rx non-posted buffer. Messages and other posted writes are written to the Rx posted buffer.

The user logic then needs to read the contents of the specified Rx buffer. If the read is made from the Rx non-posted buffer, the user logic also needs to prepare the appropriate completion data (complete with the Requester's identifying tag), place this data in the Tx completion buffer and signal to the Endpoint block that this packet is available to be sent. See "Transaction Layer Interface" in Chapter 2.

The action of reading automatically clears the buffer and liberates flow control credits. The release of these flow control credits is automatically communicated to the device at the other end of the link.

Handling Configuration Requests

Endpoints receive configuration request packets addressing their internal configuration registers from the PCIe link.

Configuration read and write request packets are included in the traffic received by the Transaction Layer from the Data Link Layer. Type 0 configuration requests are automatically filtered out and passed to the Endpoint block's Configuration and Capabilities module without any need for intervention from the user logic.

Responses from the Configuration and Capabilities module are automatically constructed into Transaction Layer completion packets and placed in the Tx completion buffer for transmission back to the configuration requester over the PCIe link.

Completions for Configuration requests compete with User Application generated TLPs in the Transmit direction. Configuration completions can be stalled if there is a continuous stream of three to four DW TLPs being transmitted by the User Application. In such a scenario, the user should ensure there are frequent gaps in transmission to allow Configuration completions to be transmitted.

Transaction Ordering

The *PCI Express Base Specification* has rules about the type of traffic that can overtake other types of traffic to avoid blockages.

The PCI Express ordering rules apply at transmission of Transaction Layer packets, and also at reception and transfer through the Transaction Layer interface to the user application. The details are given in "Ordering at Transmission," page 67 and "Ordering at Reception," page 67. The Endpoint block also arbitrates over the order in which data is taken from and returned to the different VCs, as explained in "Virtual Channel Arbitration," page 69.

Ordering at Transmission

The Transaction Layer arbitrates between each Transmit buffer to give each buffer access to the Data Link Layer transmit logic, based upon the PCI Express ordering rules, which are outlined in Table 4-1.

Table 4-1: **Summary of Ordering Rules Applied: Can "Row" pass "Column?"**

	Posted Request	Non-Posted Request	Completion
Posted Request	No	Yes	Yes
Non-Posted Request	No	No	Yes
Completion	No	Yes	No

The Endpoint block primarily chooses between the three transmit streams based on how long the packets have been waiting. The general rule is that the oldest is sent first. However, should there be a delay in processing a particular type of traffic (e.g., due to a lack of flow control credits), then the block could allow packets of other traffic types to be sent instead – based on the above ordering rules.

The Endpoint block never allows two transactions of the same type to be transmitted out of order, because both transactions are placed into the same FIFO pipeline and it is impossible for one transaction to leapfrog another inside the same buffer.

Ordering at Reception

The LLKRXPREFERREDTYPE signals indicate a recommended packet type that could be received in compliance with ordering rules, although there could be other packet types that qualify to be received in compliance. The LLKRXCH*AVAILABLEN signals indicate when a packet is available in the Rx buffer, but in some cases receiving the packet could violate ordering rules.

The incoming stream of transactions is placed into the Rx buffers, and each transaction header is given an order code by the Endpoint block as it is placed into the Rx posted, Rx non-posted, or Rx completion buffer.

The outputs provided to the user logic then become active in accordance with the ordering rules. For example, if a posted request is waiting for processing and its order code is earlier than a non-posted request also waiting in the receive buffers, LLKRXPREFERREDTYPE indicates the posted queue, while the LLKRXCH*AVAILABLE signals indicate that packets are available in both the posted and non-posted queues.

The LLKRXPREFERREDTYPE and LLKRXCH*AVAILABLEN signals together indicate when incoming packets are available and legal to receive according to the PCIe transaction ordering rules. These signals have separate bits to indicate the preferred type and packet availability for each traffic class. Within a given traffic class, the following rules must be applied to determine which packet(s) can be legally read in accordance with the PCI strongly ordered model:

- A posted packet can be read when LLKRXCHPOSTEDAVAILABLEN is asserted, which allows a posted packet to pass a non-posted or completion packet.
- A non-posted packet can be read when LLKRXCHNONPOSTEDAVAILABLEN is asserted and LLKRXPREFERREDTYPE indicates the non-posted channel, which prevents a non-posted packet from passing a posted or completion packet.
- A completion packet can be read when LLKRXCHCOMPLETIONAVAILABLEN is asserted and LLKRXPREFERREDTYPE indicates completion.
- A completion packet can be read when LLKRXCHCOMPLETIONAVAILABLEN is asserted and LLKRXCHPOSTEDAVAILABLEN is not asserted, which allows a completion packet to pass a non-posted packet, but not a posted packet.

The user logic can choose to modify the completion rule when the relaxed ordering bit is set on the packet. (If a request is made with the relaxed ordering bit set, the received completion should have the relaxed ordering bit set.) In this case:

- A completion packet can be read when LLKRXCHCOMPLETIONAVAILABLEN is asserted, regardless of LLKRXPREFERREDTYPE, which allows a completion packet to pass either a non-posted packet or a posted packet.

When packets are available in more than one traffic class, the user can choose to service the traffic classes in any order. For more information on PCI Express transaction ordering rules, see section 2.4.1 of the *PCI Express Base 1.1 Specification*.

In certain cases, the ordering rules allows transactions of different types to be made known to the user logic simultaneously. The *PCI Express Base Specification* states that in such cases, it is up to the user logic to arbitrate between the different transaction type streams (though it does recommend certain strategies).

As with transmission, transactions with the same type and VC cannot be passed to the user out of sequence because they are placed into the same received FIFO pipeline and therefore cannot pass each other.

Performance Considerations

The user application should avoid situations where non-posted or completion packets are stalled in the transmit buffers due to lack of flow control credits. Many systems advertise infinite completions, so this is primarily an issue for non-posted packets. This can be accomplished by monitoring the CREDIT_LIMIT and CREDIT_CONSUMED values in the Management interface to ensure that there are sufficient credits before pushing a packet through the Transaction Layer interface.

On the receive side, the user application should make sure packets are received through the Transaction Layer interface as quickly as possible and that forward progress is always made.

Virtual Channel Arbitration

The administration of data flow between devices in a PCIe network can necessitate arbitration between packets handled by different VCs. This is referred to as VC arbitration. It is carried out in accordance with the *PCI Express Base Specification*.

The overall aim of this arbitration is to ensure forward progress for each traffic class, with a differentiated service given (where appropriate) to higher priority traffic. A range of strategies is supported for VC arbitration.

For VC arbitration, the supported VCs are divided into a High Priority group and a Low Priority group. The channels in the High Priority group are processed according to their inherent priority, VC0 (lowest) – VC1 (highest).

The channels in the Low Priority group are processed when there are no packets waiting to be processed on any of the High Priority channels. The individual channels within this group can be processed either in a simple Round Robin order or by following a Weighted Round Robin ordering with 32 phases. This choice is made in the CORE Generator GUI and used to set the PORTVCCAPABILITYVCARBCAP attribute.

By default, VC0 is the sole member of the Low Priority group while VC1, if implemented, is placed in the High Priority group. This causes the channels to be processed according to their inherent priority. VC1 can be included in the Low Priority group through a selection in the CORE Generator GUI (LOWPRIORITYVCCOUNT attribute). See "Designing with LogiCORE IP for the Endpoint Block" in Chapter 3.

Interrupt Handling

The Endpoint block supports sending interrupt requests as either Legacy interrupts or Message Signaled Interrupts (MSI). The mode is programmed using the MSI Enable bit in the Message Control register of the MSI Capability structure. If the MSI Enable bit is set to 1, then the user application can generate MSI requests by creating and sending memory write TLPs on the transmit Transaction Layer interface. If the MSI Enable bit is reset to 0, the block generates Legacy interrupt messages as long as the Interrupt Disable bit in the PCI Command register is set to 0. This is reflected on the INTERRUPTDISABLE output:

- INTERRUPTDISABLE = 0: interrupts enabled
- INTERRUPTDISABLE = 1: interrupts disabled (requests are blocked)

The MSI Enable bits in the MSI Control register and the Interrupt Disable bit in the PCI Command register are programmed by the Root Complex. The user application has no direct control over these bits. The user application must either poll the MSI Enable bit or check that the L0MSIENABLE0 output is asserted to ensure that the Root Complex has enabled the device to send MSI packets.

Message Signaled Interrupts

Under the MSI scheme, a device requests service for a particular interrupt event by writing a system-specified MSI request (message) to a system-specified address. The interrupt events are edge-triggered.

The Endpoint block supports up to four multiple MSI messages. The user can enable one, two, or four messages through the CORE Generator GUI. This value is recorded as the MSICAPABILITYMULTIMSGCAP attribute and used to set the Multiple Message Capable field of the Message Control register included in the MSI capability. After the system powers up, the Root Complex optionally enables up to four messages, depending on the value of the MSI capability structure's Message Control register.

The Endpoint block supports the 64-bit Message Capability Structure described in section 6.8 of the *PCI Local Bus Specification, v3.0*. It offers the option of masking individual MSI vectors, thereby prohibiting the block from sending the associated messages. The capability also includes a register of pending bits to record any messages that would have been sent if the associated MSI vector was not masked. If software unmasks the corresponding mask bit, the message is sent.

To send an MSI packet, the user application must form the packet and present it at the Transaction Layer interface for transmission. MSI packets are formed as either memory write 32-bit addressable packets or memory write 64-bit addressable packets. The user must choose either the 32-bit or 64-bit addressable packet, depending on the contents of the Message Upper Address register of the MSI capability structure. If this register has a non-zero value, then a 64-bit addressable memory write is used. Otherwise a 32-bit addressable memory write is used. The data for the packet is the 16-bit data message specified by the system found in the Message Data register for the MSI Capability structure. Modifications are made according to the number of different messages that are enabled. If two messages are enabled, the required messages are formed by modifying the last bit; if four messages are enabled, the required messages are formed by modifying the last two bits.

MSI packets are identical to normal memory write packets except the *PCI Express Base Specification* specifies that the attribute bits of the transaction description field must be set to 00b. This imposes default ordering and hardware enforced cache coherency on the packet. See Sections 2.2.6.4, 2.2.6.5, and 6.1 of the *PCI Express Base 1.1 Specification* for more information on MSI packets. To send the MSI packet, the user must poll the MSI capability structure using the Management interface to obtain the necessary information for the packet.

Legacy Interrupts

The Endpoint block supports Legacy interrupt mode if MSI mode is not being used. Legacy interrupts use in-band messages to assert and deassert virtual interrupt lines on the link to emulate the Legacy PCI interrupt pins. See Section 6.1 of the *PCI Express Base 1.1 Specification* for more details. To assert an interrupt, the ASSERT_INTx message is sent; to deassert the interrupt, the DEASSERT_INTx message is sent.

The Endpoint block supports only Legacy interrupt pin A or INTA, which is common practice in Endpoint applications. The L0LEGACYINTFUNCT0 input is used to send the ASSERT_INTA and DEASSERT_INTA messages. To send the ASSERT_INTA message, the user drives L0LEGACYINTFUNCT0 from Low to High for at least one clock cycle. Once the user application is ready to send the DEASSERT_INTA message, it drives the L0LEGACYINTFUNCT0 signal from High to Low.

The Legacy interrupt system is enabled through the interrupt pin register included among the Legacy configuration registers. A value of 1 associates the Legacy interrupt pin with INTA. When Legacy interrupt messages are not used, the value is 0. The required value is set using the CORE Generator GUI and recorded as the INTERRUPTPIN attribute.

Error Detection

The *PCI Express Base Specification* identifies a number of errors a PCIe port should check for, together with a number of additional optional checks.

Most of the required checks (including several of the optional checks) are carried out by the Endpoint block. Some, however, need to be implemented by the user. The Endpoint block performs checks on received TLPs only. The user must perform all checks on transmit TLPs. Details of checks made by the Endpoint block or the user are shown in Table 4-2. This table is organized broadly in line with the sections of the *PCI Express Base Specification* describing how these checks should be made.

Table 4-2: **Error Checking Summary**

	PCI Express Specification Section	Check is Required or Optional	Where Check is Implemented
Checks Made Regarding TLPs with Data Payloads			
That the data payload of a TLP does not exceed Max_Payload_Size. Any TLP that violates this rule is a Malformed TLP.	2.2.2	Required	Endpoint block
That where a TLP includes data, the actual amount of data matches the value in the length field. Any TLP that violates this rule is a Malformed TLP.	2.2.2	Required	Endpoint block
Checks Made Regarding TLP Digests			
That the presence (or absence) of a digest correctly reflects the setting of the TD field. Any TLP that violates this rule is a Malformed TLP.	2.2.3	Required	Endpoint block
Checks Made Regarding First/Last DW Byte Enable (1DW = 32 bits)			
• That if length > 1DW, then the first DW BE is not `0000` • That if length = 1DW, then the last DW BE is `0000` • That if length > 1DW, then the last DW BE is not `0000` • That the BEs are not non-contiguous for packets ≥ 3DW in length or 2DW packets that are not QWORD aligned Any TLP that violates these rules is a Malformed TLP.	2.2.5	Optional	Endpoint block
Checks Made Regarding Memory, I/O, and Configuration Requests			
That the tag field is the correct length for the current configuration. The tag field for received and transmitted memory and I/O requests must be checked by the user.	2.2.6.2	Optional	Endpoint block
That MWr requests do not specify an Address/Length combination that causes a Memory Space access to cross a 4 kB boundary. Any MWr request that violates this rule is treated as a Malformed TLP. For MRd requests, this optional check should be implemented in the fabric, if desired.	2.2.7	Optional	Endpoint block
That I/O requests obey the following restrictions: • TC[2:0] must be `000b` • Attr[1:0] must be `00b` • Length[9:0] must be `00 0000 0001b` • The last DW BE[3:0] must be `000b` Any I/O request that violates this rule is treated as a Malformed TLP.	2.2.7	Optional	Endpoint block

Table 4-2: **Error Checking Summary *(Continued)***

	PCI Express Specification Section	Check is Required or Optional	Where Check is Implemented
That configuration requests obey the following restrictions: • TC[2:0] must be `000b` • Attr[1:0] must be `00b` • Length[9:0] must be `00 0000 0001b` • The last DW BE[3:0] must be `000b` Any configuration request that violates this rule is treated as a Malformed TLP.	2.2.7	Optional	Endpoint block
That configuration requests address a valid function number field.	7.3.2	Required	Endpoint block
Checks Made Regarding Message Requests			
That Assert_INTx/Deassert_INTx Messages are only issued by upstream Ports. Any Assert_INTx/Deassert_INTx Message that violates this rule is treated as a Malformed TLP.	2.2.8.1	Optional	Endpoint block
That Assert_INTx/Deassert_INTx Messages use TC0. Any Assert_INTx/Deassert_INTx Message that violates this rule is treated as a Malformed TLP.	2.2.8.1	Required	Endpoint block
That Power Management Messages use TC0. Any PM Message that violates this rule is treated as a Malformed TLP.	2.2.8.2	Required	Endpoint block
That Error Signaling Messages use TC0. Any Error Signaling Message that violates this rule is treated as a Malformed TLP.	2.2.8.3	Required	Endpoint block
That Unlock Messages use TC0. Any Unlock Message that violates this rule is treated as a Malformed TLP.	2.2.8.4	Required	Endpoint block
That Set_Slot_Power_Limit Messages use TC0. Any Set_Slot_Power_Limit message that violates this rule is treated as a Malformed TLP.	2.2.8.5	Required	Endpoint block
Unsupported Type 0 Vendor-Defined Messages. Reported as unsupported requests. **Note:** Type 1 Vendor-Defined Messages should be ignored.	2.2.8.6	Required	User
Unsupported messages, i.e., all messages other than: • Supported Type 0 Vendor-Defined Messages (message code `01111110`) • Type 1 Vendor-Defined Messages (message code `01111111`) • Ignored Messages (messages codes `01000000`, `01000001`, `01000011`, `01000100`, `01000101`, `01000111`, `01001000`) Reported as unsupported requests.	2.2.8.6, 2.2.8.7	Required	User
Checks Made Regarding Handling of TLPs			
That any received TLP passes the required and implemented optional checks on TLP formation. Any TLP that violates this rule is a malformed TLP. The user must generate the appropriate completion TLP.	2.3	Required	Endpoint block

Table 4-2: **Error Checking Summary *(Continued)***

	PCI Express Specification Section	Check is Required or Optional	Where Check is Implemented
That Memory Read Request-Locked (MRdLk) requests do not include a payload. Any MRdLk requests with payload must be discarded by the user and a malformed TLP must be signaled.	2.3	Required	User
That a Completion with Data (CplD) has a 3DW header. Any CplD with a 4 DW header must be discarded by the user and a malformed TLP must be signaled.	2.3	Required	User
That an I/O request has a 3DW header. Any I/O request with a 4DW header must be discarded by the user and a malformed TLP must be signaled.	2.2.7	Required	User
That the byte enable rules for received memory reads are followed. If not, TLP must be discarded by the user and a malformed TLP must be signaled.	2.2.5	Required	User
Checks Made Regarding Request Handling			
Unsupported request types. Reported as an unsupported request. The user must generate the appropriate completion TLP. **Note:** While type 0 configuration requests are routed to the Endpoint block's configuration completer, type 1 configuration requests are routed to the receive Transaction Layer interface and should be handled by the user as an unsupported request.	2.3.1	Required	User
Requests that violate the programming model of the device. Reported as a completer abort. The user must generate the appropriate completion TLP.	2.3.1	Optional	User
Requests that cannot be processed due to a device-specific error condition. Reported as a completer abort. The user must generate the appropriate completion TLP.	2.3.1	Required	User
That completions do not include more data than permitted by the MAX_PAYLOAD_SIZE. Any completion that violates this rule is treated as a Malformed TLP.	2.3.1.1	Required	Endpoint block
Violations of RCB. Any completion that violates the RCB rules is treated as a Malformed TLP.	2.3.1.1	Optional	User
Checks Made Regarding Completion Handling			
Unexpected completions.	2.3.2	Required	User
Completions with a status of request retry for requests other than configuration requests. Treated as a malformed TLP.	2.3.2	Optional	User
Completions with a completion status of unsupported request or completer abort. Reported via conventional PCI reporting mechanisms.	2.3.2	Required	User
Checks Made Regarding Virtual Channel Mechanism			
That requesters that do not support the VC capability structure only operate on TC0. Received requests on TC1-TC7 must be handled normally (without error) and completions must be returned on the same TC in which the request was received.	2.5	Optional	User

Table 4-2: **Error Checking Summary *(Continued)***

	PCI Express Specification Section	Check is Required or Optional	Where Check is Implemented
That the TC associated with each TLP is mapped to an enabled VC at an Ingress Port. Any TLP that violates this rule is treated as a Malformed TLP.	2.5.3	Required	Endpoint block
Checks Made Regarding Flow Control			
That the initial FC value is greater than or equal to the minimum advertisement. Reported as a flow control protocol error. Requires knowledge of the device and the Max Payload Size setting at the far end of the link.	2.6.1	Optional	User
That no receiver ever cumulatively issues more than 2047 outstanding unused data credits or 127 outstanding unused header credits. Reported as a flow control protocol error.	2.6.1	Optional	Endpoint block
That if infinite credits are advertised during initialization, all updates must also be infinite. Reported as a flow control protocol error. This also applies where just a header or just the data has been advertised as infinite.	2.6.1	Optional	Endpoint block
That the VC used by a TLP has been enabled. Any TLP that violates this rule is treated as a Malformed TLP.	2.6.1	Required	Endpoint block
Receiver Overflow. The *PCI Express Base Specification* defines this as happening where the number of TLPs exceeds CREDITS_ALLOCATED. However, the Endpoint block reports where FIFO actually overflows.	2.6.1.2	Optional	Endpoint block
That Update FCPs are scheduled for transmission at the specified interval. The Endpoint block takes the option of employing a 200 µs watchdog timer.	2.6.1.2	Optional	Endpoint block
Checks Made Regarding Data Integrity			
Integrity of TD bit in messages received and forwarded by switches. Any failed ECRC checks are reported.	2.7.1	Required	Endpoint block
Receipt of a Poisoned TLP.	2.7.2.2	Required	User
Checks Made Regarding Completion Timeout			
That the completion timeout timer does not expire in less than 50 µs but must expire if a request is not completed in 50 ms.	2.8	Required	User
Checks Made Regarding LCRC and Sequence Number (TLP Transmitter)			
REPLAY_NUM rolling over from `11b` to `00b`. Causes the Transmitter to: (a) report an error; (b) signal the Physical Layer to retrain the Link.	3.5.2.1	Required	Endpoint block
Retry buffer containing TLPs for which no Ack or Nak DLLP has been received for a period exceeding specified maximum time. Causes the Transmitter to: (a) report an error; (b) initiate a replay.	3.5.2.1	Required	Endpoint block
Value in the CRC field of all received DLLPs compared with calculated result. If not equal: (a) the DLLP is discarded as corrupt; (b) an error is reported.	3.5.2.1	Required	Endpoint block

Table 4-2: **Error Checking Summary *(Continued)***

	PCI Express Specification Section	Check is Required or Optional	Where Check is Implemented
Sequence Number specified by the AckNak_Seq_Num compared with that of unacknowledged TLPs and value in ACKD_SEQ. If no match found: (a) the DLLP is discarded; (b) a DLLP error is reported.	3.5.2.1	Required	Endpoint block
Checks Made Regarding LCRC and Sequence Number (TLP Receiver)			
LCRC field of the received TLP compared with calculated result. If not equal: (a) the TLP is discarded as corrupt; (b) an error is reported.	3.5.3.1	Required	Endpoint block
LCRC field of the received TLP compared with logical NOT of calculated result if TLP end framing symbol is EDB. LCRC does not match logical NOT of the calculated value: (a) the TLP is discarded as corrupt; (b) an error is reported.	3.5.3.1	Required	Endpoint block
TLP Sequence Number compared with expected value stored in NEXT_RCV_SEQ. If not equal, an error is reported.	3.5.3.1	Required	Endpoint block
Checks Resulting in Receiver Errors			
Validity of received 8B/10B symbols bearing in mind the running disparity. Errors reported as Receiver Errors.	4.2.1.3	Required	Endpoint block
Framing Errors, Loss of Symbol Lock, Lane Deskew Errors, and Elasticity Buffer Overflow/Underflow. Errors reported as Receiver Errors.	4.2.2.1	Optional	User

Error Reporting

While failed requests are reported through the completion status field of the completion packet sent in response to the request, the occurrence of other error conditions is required by the *PCI Express Base Specification* to be recorded in the appropriate configuration registers. In addition, a message advising that an error condition has occurred can optionally be sent upstream towards the Root Complex. Details of the error condition are available by reading the relevant fields of the device configuration registers.

As shown in Table 4-3, the Endpoint block performs the error reporting for the errors shown as being checked. For error checking done by the user, the Endpoint block offers a range of inputs that can update relevant registers in the event of an error condition occurring. The error bit(s) set as a result of asserting these signals depends on both the type of error and the range of extended capabilities configured for the Endpoint block.

As well as setting the appropriate registers, the assertion of these error signals can also cause the appropriate in-band ERR_CORR, ERR_NONFATAL, or ERR_FATAL message to be sent upstream towards the Root Complex.

The Endpoint block must be reset by the Root Complex after a fatal error is detected in order to return to normal operation.

Table 4-3 summarizes how different types of errors are reported and handled by the Endpoint block.

Table 4-3: **Error Reporting with Endpoint Block Action**

Error Detected		Action by Endpoint Block	
Errors Flagged by Transaction Layer			
Flow Control Protocol Error	Message generated	Action when Receiver	ERR_FATAL
	Standard PCIe Error Reporting		Fatal Error Detected bit of the Device Status register is set.
	Legacy Error Reporting		Signaled System Error bit of Status register set (but only if an ERR_FATAL message is sent).
Malformed TLP	Message generated	Action when Receiver	ERR_FATAL
	Standard PCIe Error Reporting		Fatal Error Detected bit of Device Status Register set.
	Legacy Error Reporting		Signaled System Error bit of Status register set (but only if an ERR_FATAL message is sent).
Poisoned TLP	Message generated	Action when Transmitter	(None)
	Standard PCIe Error Reporting		(None)
	Legacy Error Reporting		Master Data Parity Error bit of Status register set (but only if PERR_EN is logic 1)
Receiver Overflow	Message generated	Action when Receiver	ERR_FATAL
	Standard PCIe Error Reporting		Fatal Error Detected bit of Device Status Register set.
	Legacy Error Reporting		Signaled System Error bit of Status register set (but only if an ERR_FATAL message is sent).
Errors Flagged by Data Link Layer			
Data Link Protocol Error	Message generated	Action when Transmitter or Receiver	ERR_FATAL
	Standard PCIe Error Reporting		Fatal Error Detected bit of Device Status register set.
	Legacy Error Reporting		Signaled System Error bit (but only if an ERR_FATAL message is sent).

Table 4-4 summarizes how different types of errors are reported and the actions taken by the user.

Table 4-4: **Error Reporting with User Action**

Error Detected	Action by User	
Errors Flagged by Transaction Layer		
Completer Abort	Action when Transmitter (i.e., the Completer)	Assert L0SETUSERSIGNALLEDTARGETABORT port.
	Action when Receiver (i.e., receiver of the completion)	Assert L0SETUSERRECEIVEDTARGETABORT port.
Malformed TLP	Action when Receiver	Assert L0SETDETECTEDFATALERROR port.
Poisoned TLP	Action when Receiver	Assert L0SETUSERMASTERDATAPARITY port only if it is a completion TLP. If it is not an advisory non-fatal error, assert L0SETDETECTEDNONFATALERROR port, otherwise do not assert it.
Completion with Unsupported Request	Action when Receiver	Assert L0SETUSERRECEIVEDMASTERABORT port.

Message Tags

The Endpoint block supports the use of either 5-bit or 8-bit (extended) Message Tags. The Extended Tag Field Supported bit is permanently enabled in the Device Capabilities register and the choice of tag-length depends on the Extended Tag Field Enable bit in the Device Control register. If 5-bit tags are used, the remaining three bits of the tag field should be set to zero. Any nonzero bits within the remaining three bits of the tag field can cause the Endpoint block to report an unsupported request.

Phantom Function Support

The Endpoint block supports the use of Phantom Functions. The Phantom Functions Supported bits of the Device Capabilities register are set to `01`, indicating single bit support. Functions 0, 1, 2, and 3 can claim functions 4, 5, 6, and 7 as Phantom Functions, respectively. If the Phantom Function Number Enable bit is set, the maximum possible number of outstanding requests requiring completion can be increased beyond 256 by using Function Numbers not assigned to implemented functions to logically extend the tag identifier.

Lane Width

The maximum number of lanes supported by a design using the Endpoint block can be specified through the ACTIVELANESIN and LINKCAPABILITYMAXLINKWIDTH attributes. These attributes should specify the same number of lanes, and a GTP transceiver should be connected to the Endpoint block through the Transceiver interface for each lane specified. This is automatically configured and connected based on choices made in the CORE Generator GUI.

If a design using the Endpoint block is plugged into a slot having fewer lanes than the configuration of the Endpoint block, or if lane(s) are broken, the Endpoint block auto-negotiates a smaller lane width with the link partner. The following lane width auto-negotiations are supported:

- x8 to x4, x2 or x1
- x4 to x2 or x1
- x2 to x1

The negotiated lane width is indicated by the L0MACNEGOTIATEDLINKWIDTH output once the link has entered L0.

Once a link has been retrained to a lower than maximum supported link width, it is unable to retrain back up to a higher link width through recovery. A complete receiver detect sequence is required to configure the design to a higher link width. This can be done by resetting the Endpoint block.

Lane Reversal

The integrated Endpoint block supports limited lane reversal capabilities and therefore provides flexibility in the design of the board for the link partner. The link partner can choose to layout the board with reversed lane numbers and the Endpoint block will continue to link train successfully and operate normally. The configurations that have lane reversal support are x8 and x4 (excluding downshift modes). Downshift refers to the link width negotiation process that occurs when link partners have different lane width capabilities advertised. As a result of lane width negotiation, the link partners negotiate down to the smaller of the two advertised lane widths. Table 4-5 describes the several possible combinations including downshift modes and availability of lane reversal support.

Table 4-5: **Lane Reversal Support**

Endpoint Block Advertised Lane Width	Negotiated Lane Width	Lane Number Mapping (Endpoint → Link Partner)		Lane Reversal Supported
		Endpoint	Link Partner	
x8	x8	Lane 0 … Lane 7	Lane 7 … Lane 0	Yes
x8	x4	Lane 0 … Lane 3	Lane 7 … Lane 4	No(1)
x8	x2	Lane 0 … Lane 3	Lane 7… Lane 6	No(1)
x4	x4	Lane 0 … Lane 3	Lane 3 … Lane 0	Yes
x4	x2	Lane 0 … Lane 1	Lane 3 … Lane 2	No(1)

Notes:

1. When the lanes are reversed in the board layout and a downshift adapter card is inserted between the Endpoint and link partner, Lane 0 of the link partner remains unconnected (as shown by the lane mapping in Table 4-5) and therefore does not link train.

Known Restrictions

This section describes several restrictions and anomalies in the functionality of the integrated Endpoint block for PCI Express® designs. Designers must understand each restriction and the potential impact on their application. This chapter also clearly describes the user action required to workaround the restrictions and anomalies. In some cases there are no workarounds available. Wherever applicable, the availability of the workaround in the LogiCORE™ wrappers is indicated. Designers must read the descriptions and workarounds carefully before proceeding to design.

TX Transmission Issues Due to Lack of Data Credits

Whenever the transmission of a minimum size packet (1DW posted or completion, non-posted) causes the transmit buffers to run out of data credits (while header credits are still available), then one of the following symptoms can result in x8 designs:

- A nullified TLP is transmitted. This occurs when the transmit path incorrectly starts transmission of a TLP when the buffer is empty and subsequently nullifies it.
- TLPs could be sent out of order. This will result in non-posted packets potentially passing posted packets.
- A valid TLP is transmitted when there are no credits available. This could result in the TLP being dropped by the partner device due to lack of buffer space to accept the TLP.

In addition, if the partner device is configured so that the advertised flow control credits follow the guidelines shown in Table 4-6, the symptoms described in this section will not occur. Completion packets need to satisfy one of the two guidelines (row 3 OR row 4 in Table 4-6).

For Table 4-6, all credits are in units of four DWORDs = 16 bytes and n = MPS/16 or MTU/16 (where the maximum payload size (MPS) and maximum transfer unit (MTU) are expressed in bytes).

Table 4-6: **Advertised Flow Control Credits Guidelines (from Partner Device)**

Packet Type	Header Credits	Data Credits
Non Posted	x	0 or ≥ x
Posted	x	≥ x * n
Completions	0	0
Completions	x	≥ x * n

Workaround

The user can perform flow control in the FPGA and prevent a minimum size packet from being presented to the Endpoint Block if it is in danger of running out of data credits. This requires monitoring the advertised credit information, the consumed credit information from the Endpoint Block, and the occupancy levels of the transmit buffers. This workaround is implemented in LogiCORE Endpoint Block Plus for PCI Express Designs v1.6.1 or later. This workaround has a potential performance impact of up to 12% (worst case scenario). Actual numbers will vary across applications and systems, and could be much lower. No workaround is implemented in LogiCORE Endpoint Block for PCI Express Designs.

64-Packet Threshold for Completion Streaming on RX Interface

The LLKPREFERREDTYPE and LLKRXCH*AVAILABLE signals together allow the user to implement both strict-ordering and relaxed-ordering rules. Relaxed ordering allows completion packets to bypass older posted or non-posted packets available in the receive buffers. For more information on relaxed ordering, refer to Section 2.4 of the *PCI Express Base Specification*. LogiCORE Endpoint Block Plus for PCI Express uses these signals to implement relaxed ordering when used in Completion Streaming mode to achieve high performance on completions.

When older posted and non-posted packets are bypassed by completion packets, the user must ensure that any given completion packet is allowed to pass any given non-posted packet only if it is within a 64-packet window from the non-posted packet. If this requirement is not met, several undesirable effects can result including older non-posted packets, passing older posted packets, complete blocking of posted, or non-posted packets. This requirement must be met when completions to bypass posted and non-posted packets are allowed while draining packets from the receive buffers.

Workaround

Certain precautions must be taken when allowing completion packets to bypass older posted or non-posted packets. When older posted and non-posted packets are bypassed by completion packets, any given completion packet is only allowed to pass any given non-posted packet when it is within a 64-packet window from the non-posted packet. Monitoring when a completion packet arrives relative to a non-posted packet and waiting for it to be drained will ensure a 64-packet window from the non-posted packet before allowing it to pass. Using the management interface, monitor the LLKRXNONPOSTEDAVAILABLE signal, LLKRXPREFERREDTYPE signal, and the credit status information. By selecting one of three options in the GUI, LogiCORE Endpoint Block Plus for PCI Express v1.4 or later, when used in Completion Streaming Mode implements the workaround logic. No workarounds are implemented in LogiCORE Endpoint Block for PCI Express.

The next step is to switch from draining completions to draining posted or non-posted packets whenever the 64-packet window is required. In this example, the 64-packet window requirement workaround uses a traffic pattern where posted and non-posted packets are scattered inside a stream of completions:

1P, 10C, 2NP, 50C, 1P, 10C, 1NP, 90C, 2NP …

In this illustration, each packet is described with the type of packet followed by the sequence number assigned to it by the receive logic. The example sequence is converted to:

P-1, C-2, ..., C-11, NP-12, NP-13, C-14, ..., C-63, P-64, C-65, ..., C-74, NP-75,
C-76, ..., C-165, NP-166, NP-167

and the following statements are true:

- C-77 (12+65) cannot pass NP-12, C-78 (13+65) cannot pass NP-13 and so forth.
- If one of the above conditions is violated, and C-130 is allowed to pass P-1(1+129), then NP-12 will look younger than P-1 and could be read out ahead of P-1.

Using the previous example, the packets can be drained in the following sequence without violating the 64-packet window requirement:

C-1,…, C-76, P-1, NP-12, C-77, NP-13, C-78,…, C-139, P-64, NP-75, C-140,…, C-165, NP-166, NP-167 ...

However, if the posted packets are large and completions are very short, the completion buffer is at risk of overflow when draining the posted packet. The risk is higher if multiple posted or non-posted packets must be drained before switching back to draining completions. Since the risk of completion buffer overflow depends on the traffic pattern, there are three potential workarounds:

1. When a predictable traffic pattern with uniform packet sizes is used with Completion Streaming mode, the user should switch from draining completions to draining posted/non-posted packets whenever there is danger of build up of posted/non-posted packets (to avoid completion buffer overflow while draining posted/non-posted packets) or when there is a danger of completion passing a non-posted packet outside the 64-packet window
2. In all other cases which use Completion Streaming mode, the flow control credits for posted packets and non-posted packets should be restricted to one header each. In addition, preference should be given to draining posted and non-posted packets whenever there is a packet of that type available in the receive buffer. This preference controls the transmission of posted and non-posted packets from the partner device and will always guarantee safe operation independent of traffic pattern.
3. This solution is an alternative to solution #2 and can be used with and without Completion Streaming mode. The user should turn off infinite completion credits by setting the attribute INFINITECOMPLETIONS in the integrated block to FALSE. The user should then switch to draining posted/non-posted packets whenever there is danger of a completion passing a non-posted packet outside the 64-packet window. However, turning off infinite completion credits will result in a non-compliant solution.

Users should choose the option that best suits their applications. LogiCORE Endpoint Block Plus for PCI Express implements all three solutions and provides them as a user selectable option.

Reset Considerations in LTSSM Polling State

When the integrated block's LTSSM is in the polling state and Lane 0 breaks electrical idle (transitions from 1 to 0), the block is not reset.

Workaround

The user must monitor the PIPERXELECIDLE0 and L0LTSSMSTATE signals and generate an additional reset to the block when this LTSSM condition occurs. The additional reset must be applied to all registers except the sticky and management registers. Sample pseudocode is provided:

```
additional reset = (PIPERXLECIDLE0 == 1 → 0)
& (L0LTSSMSTATE == 4'b0010)
```

This workaround is implemented in LogiCORE Endpoint Block Plus for PCI Express designs v1.3 or later and LogiCORE Endpoint Block for PCI Express Designs v1.4 or later.

Invalid Cycles in LLKRXPREFERREDTYPE Signal

Due to the way the integrated block updates LLKRXPREFERREDTYPE and the LLKRXCH*AVAILABLE signals, there will be some cycles in which LLKRXPREFERREDTYPE is invalid. Sampling the signal during the invalid cycles can result in incorrect operation.

Workaround

These invalid cycles can be detected in a deterministic fashion and depend on whether the arbiter in the receive path has granted the request to the TLI or the internal configuration completion. Whenever the user asserts LLKRXDSTREQN, the grant of the arbiter to the user and subsequently the determination of the invalid cycles of LLKRXPREFERREDTYPE can be implemented in a state machine and additional logic.

This workaround is implemented in LogiCORE Endpoint Block Plus for PCI Express Designs v1.3 or later and LogiCORE Endpoint Block for PCI Express v1.4 or later.

Continuous Deassertion of LLKTXCONFIGREADYN Signal

Receiving an undefined MsgD with a payload greater than two or receiving a malformed configuration request with format `2'b11` causes the internal configuration block to hang, resulting in continuous deassertion of LLKTXCONFIGREADYN.

Workaround

The user should monitor the LLKTXCONFIGREADYN signal to detect any fatal failures in the internal configuration block. If the LLKTXCONFIGREADYN signal is deasserted for more than several thousands of user clock cycles, the user should transmit a fatal error message by toggling L0SETDETECTEDFATALERROR. The user should wait a minimum of 2500 user clock cycles of continuous deassertion before transmitting a fatal error message.

No workarounds are implemented in LogiCORE Endpoint Block Plus or LogiCORE Endpoint Block for PCI Express Designs.

Transmitting Completion TLP with Completer Abort Status

Whenever the user application sends a completion packet with the Completer Abort status bit set, the integrated block transmits a non-fatal error message that could result in a blue screen on the host.

Workaround

The user can alternatively send a completion packet with the unsupported request status bit set to prevent an error message from being transmitted. This solution, however, is non-compliant.

No workarounds are implemented in LogiCORE Endpoint Block Plus or LogiCORE Endpoint Block for PCI Express Designs.

Link Retrain Due to an Absence of UpdateFC DLLPs

When the partner device advertises infinite header and data credits for all packet types for a given virtual channel, the integrated block might not receive any UpdateFC DLLPs. When the integrated block does not receive any UpdateFC DLLPs, it initiates a link retrain because an internal timer used to track the receipt of UpdateFC DLLPs has expired. This behavior is non-compliant.

Workaround

The partner device should be configured to have at least one packet type per virtual channel advertising the finite header and data credits.

No workarounds are implemented in LogiCORE Endpoint Block Plus or LogiCORE Endpoint Block for PCI Express Designs.

Automatic Transmission of PME_TO_Ack Message

The integrated block automatically sends a PME_TO_Ack message in response to a received PME_Turn_Off message instead of allowing the user to control the transmission of the PME_Turn_Off message.

Workaround

There are no workarounds to prevent the transmission of the PME_Turn_Off. Any required housekeeping must be completed within 250 ns from the receipt of PME_Turn_Off message in preparation for power removal. The receipt of PME_Turn_Off message is indicated by a transition to 1 on the L0PWRTURNOFFREQ port. This is described in Table 2-15, page 42.

No workarounds are implemented in LogiCORE Endpoint Block Plus or LogiCORE Endpoint Block for PCI Express Designs.

64-Packet Threshold on Posted Packets Passing Non-posted and Completion Packets in the TX Direction

If non-posted packets and completion packets are stalled inside the integrated block's transmit buffer and more than 64 packets pass a given stalled packet, then the following scenarios can occur:

- Subsequent posted packets might get stalled until the stalled non-posted or completion packet is transmitted.
- Older non-posted and completion packets could become younger and will be incorrectly blocked by a posted packet that arrives later. These non-posted and completion packets are transmitted if all posted packets that were in the transmit buffer when the blocking occurred are eventually transmitted.
- A nullified TLP can be transmitted.

Workaround

To avoid the issues listed, the user needs to prevent non-posted packets and completion packets from being stalled inside the transmit buffer of the integrated block. The user needs to monitor the credit status through the management interface and send non-posted and completion packets on the TLI only if sufficient credits are available for transmission. The user determines if sufficient credits are available by monitoring "credits consumed" and "credit limit" for non-posted and completion packets in the transmit direction. The usage of the Management Interface ports for monitoring credit information is described in Table 2-7, page 35.

No workarounds are implemented in LogiCORE Endpoint Block Plus or LogiCORE Endpoint Block for PCI Express Designs.

REPLAY_NUM Rollover in LTSSM State TX.L0s

If a given packet is replayed several times, it causes REPLAY_NUM to rollover. According to the PCI Express Base Specification 1.1, this should always trigger link training. However, the integrated block will not initiate link training due to REPLAY_NUM rollover if the LTSSM state is *Tx.L0s*. As a result, the block returns to L0 state instead of Recovery state, and will not replay any packets. The block will continue to remain in this state until link training is initiated.

Workaround

To avoid this scenario, the user can inject TS1 training sets into the receive path of the block when the LTSSM returns to L0 state. Insert training sets by adding FPGA logic at the Transceiver interface. Monitor signals L0DLLERRVECTOR[3] and L0LTSSMSTATE to detect when a rollover occurs and the state of the LTSSM.

No workarounds are implemented in LogiCORE Endpoint Block Plus or LogiCORE Endpoint Block for PCI Express Designs.

ACK Ignored When Followed by IDLE Ordered Set

When the host sends an ACK followed by an IDLE ordered set to initiate *L0s.Entry*, the integrated block never sees the ACK and instead replays the packet. If this scenario repeats multiple times, REPLAY_NUM rolls over, causing the block to initiate link training.

Workaround

To avoid this scenario, the user can intercept the IDLE ordered set and delay it in the FPGA logic by adding logic at the Transceiver interface.

No workarounds are implemented in LogiCORE Endpoint Block Plus or LogiCORE Endpoint Block for PCI Express Designs.

Access to Unimplemented Configuration Space

According to PCI Express Specification 1.1, an Endpoint should treat access to an unimplemented configuration space as an unsupported request. The integrated block responds with a successful completion that is non-compliant.

Workaround

There are no workarounds for this issue. However, as an upstream component is not expected to access an unimplemented configuration space, this has no impact on safe operation.

No workarounds are implemented in LogiCORE Endpoint Block Plus or LogiCORE Endpoint Block for PCI Express Designs.

Receive TLPs with Illegal Payload Length

According to PCI Express Specification 1.1, any TLP with a payload length that is not a multiple of 1DW is illegal. The integrated block does not send an ERR_FATAL message when it receives a TLP with an illegal payload length. Instead, the block detects this TLP as a bad LCRC and sends back a NAK.

Workaround

There are no workarounds for this issue. However, such an occurrence is very rare.

No workarounds are implemented in LogiCORE Endpoint Block Plus or LogiCORE Endpoint Block for PCI Express Designs.

Receiving PM_PME or PME_TO_Ack Messages

According to PCI Express Specification 1.1, an Endpoint should not receive a PM_PME or a PME_TO_Ack message. If it receives such a message, it should respond by sending an ERR_NON_FATAL message. The integrated block does not respond with any error message and silently drops the received messages.

Workaround

There are no workarounds for this issue. However, this issue is expected to have minimal or no impact on safe operation.

No workarounds are implemented in LogiCORE Endpoint Block Plus or LogiCORE Endpoint Block for PCI Express Designs.

Loopback Slave Mode Considerations

The integrated block supports Loopback Slave mode operation as described in the PCI Express Base Specification 1.1. When the integrated block is operating as a Loopback slave, all data received is looped back to the upstream component. The upstream component can initiate an exit from Loopback by transmitting an Electrical Idle ordered set followed by transitioning the serial lines to Electrical Idle. The integrated block is expected to loopback all data including the Electrical Idle ordered set before transitioning its TX serial line to Electrical Idle. The block does not loopback all data.

Workaround

The user can workaround this issue by introducing a delay of 160 ns (equal to 40 CRMCORECLK cycles) in the FPGA logic on the RXELECIDLE 0/1 signals in the interface between the GTP transceiver and the integrated block. The user can build a single FPGA logic design that turns on the delay whenever L0LTSSMSTATE = Loopback, thus preventing delay during normal operation.

No workarounds are implemented in LogiCORE Endpoint Block Plus or LogiCORE Endpoint Block for PCI Express Designs.

Link Upconfigure Bit on TS2 Training Sequence

The integrated block is designed for forward compatibility with PCI Express Base Specification 2.0 and successful interoperability. However, there is one exception.

According to the PCI Express Specification 2.0, a bit in the TS2 sequence is used as a Link Upconfigure bit. This bit is reserved in the PCI Express Specification 1.1. The integrated block is expected to transmit a 1 on this bit and ignore the value on the RX side. The integrated block does not ignore this bit and fails to link train if it is set to 1.

Workaround

The user should force this bit to 0 in each lane by inserting FPGA logic in the interface between the GTP transceiver and the integrated block.

This workaround is implemented in LogiCORE Endpoint Block Plus for PCI Express Designs v1.5 or later. No workarounds are implemented in LogiCORE Endpoint Block for PCI Express Designs.

Returning to L1 from L0 in D3hot State

When an upstream component programs the integrated block to the D3hot power state, the integrated block transitions into an L1 state. While the integrated block is in the D3hot state, if the upstream component sends a TLP, then the block initiates entry into the L0 state in order to process the incoming TLP and send completions, if necessary. After processing the TLP and sending any relevant completions, the integrated block does not return to the L1 state and remains in L0 state, which is not compliant.

Workaround

To avoid this scenario, the upstream component needs to initiate a D0 transition before sending a TLP and initiate a D3hot transition after receiving any expected completions to send the integrated block back into the D3hot power state.

No workarounds are implemented in LogiCORE Endpoint Block Plus or LogiCORE Endpoint Block for PCI Express Designs.

Credit Leak When Transmitting Completion TLPs

Whenever a minimum size completion TLP (1DW) entering the TX completion buffer causes it to become full and there is a pending configuration completion at the same time, then the configuration completion is incorrectly entered into the TX posted buffer. This results in a reduction of advertised posted credits and no reduction in advertised completion credits, which are both incorrect. This could potentially lead to two symptoms:

- A completion will be transmitted when the partner device does not have credits to accept it causing flow control protocol error.
- Posted packets will be stalled even though the partner device has enough credits to accept the packet.

Workaround

User can perform flow control in the FPGA and prevent a minimum size completion from being presented to the Endpoint Block, if it is in danger of causing the transmit completion buffer to become full. This requires monitoring several statistics signals to accurately measure the occupancy level of the transmit completion buffer. This workaround is implemented in LogiCORE Endpoint Block Plus for PCI Express Designs v1.6.1 or later. No workaround is implemented in LogiCORE Endpoint Block for PCI Express Designs.

Receipt of Ignored Messages

Whenever an ignored message is received, the integrated block does not perform any action on it and the message is passed to the user logic.

Workaround

The user should monitor the user interface for receipt of an ignored message and perform appropriate user action as per the PCI Express Base Specification 1.1, section 2.2.8.7.

No workarounds are implemented in LogiCORE Endpoint Block Plus or LogiCORE Endpoint Block for PCI Express Designs.

Receipt of Unsupported Configuration Requests and Poisoned Configuration Writes

Whenever an unsupported configuration request is received (for example, a configuration request to functions 1 to 7) *OR* a poisoned configuration request is received, the integrated block incorrectly sets the correctable error detected and unsupported request detected bits.

Workaround

The user should implement a separate version (set correctly) of the correctable error detected and unsupported request detected bits in user logic. These registers should be used to overwrite the internal bits when host reads the Device Control register and Status register.

This workaround is implemented in LogiCORE Endpoint Block Plus for PCI Express Designs v1.3 or later. No workarounds are implemented in LogiCORE Endpoint Block for PCI Express Designs.

Receipt of Back-to-Back ACK DLLPs

Whenever ACKs are received in consecutive cycles for x8 designs, the TX path of the block can lock up.

Workaround

User can workaround this issue by monitoring the interface between the Endpoint Block and the GTP or GTX transceivers and nullify the second ACK by zeroing out all the bits in the ACK DLLP.

This workaound is implemented in LogiCORE Endpoint Block Plus for PCI Express Designs v1.8 or later. No workarounds are implemented in LogiCORE Endpoint Block for PCI Express Designs.

Simulating with the Endpoint Block

Summary

This chapter describes how to simulate the Endpoint block embedded in Virtex-5 devices and provides some examples. The sections include:

- "Overview"
- "Power-up and Reset"
- "Clocking"
- "Examples"

Overview

The simulation of designs containing the Endpoint block has specific prerequisites that the simulation environment and the testbench must fulfill.

The Synthesis and Simulation Design Guide explains how to setup the simulation environment for supported simulators depending on the used Hardware Description Language (HDL). This design guide can be downloaded from the Xilinx website at: http://www.xilinx.com/support/sw_manuals/xilinx8/download/

Prerequisites for the simulation environment used for designs containing the Endpoint block are:

1. Simulator with a SWIFT interface to support SmartModels
2. Installed SmartModel for the Endpoint block
3. Installed SmartModel for the GTP_DUAL tile
4. Correct setting of the environment variable that points to the SmartModel installation directory
5. Correct setup of the simulator for SmartModel use (initialization file and environment variable(s))
6. Compilation of the SmartModel wrapper files into the UNISIM and SIMPRIM libraries
7. Compilation of the Integrated Endpoint and GTP_DUAL SmartModels into a simulation library
8. Correct simulator resolution (Verilog)
9. Correct compilation order of simulation libraries (VHDL)

Before proceeding, make sure all the prerequisites are met. The user guide of the simulator and the Synthesis and Simulation Design Guide provide a detailed list of settings for SmartModel support. The `compxlib` tool with `sl_admin` facilitates the setup of the supported simulator.

SmartModel Description

The behavior of the Endpoint block is modeled using a SmartModel, which is an encrypted version of the HDL used to implement the modeled block. SmartModel allows designs containing a Endpoint block to be simulated in the following design phases:

- Register Transfer Level (RTL)/Pre-Synthesis Simulation
- Post-Synthesis Simulation/Pre-NGDBuild Simulation
- Post-NGDBuild/Pre-Map Simulation
- Post-Map/Partial Timing Simulation
- Post-Place and Route/Timing Simulation

Power-up and Reset

Simulating in Verilog

The GSR signal is a global routing of nets in the FPGA that provide a means of setting or resetting applicable components in the device during configuration. The simulation behavior of this signal is modeled using the `glbl` module:

```
$XILINX/verilog/src/glbl.v
```

The `glbl.v` module connects the global signals to the design, making it necessary to compile this module with the other design files and load it along with the design and testbench files for simulation.

GSR does not need to be defined in the testbench. The `glbl.v` file declares the GSR signal and automatically pulses GSR for 100 ns. This handling is sufficient for backend simulations and functional simulations.

Simulating in VHDL

There are no special steps necessary for driving GSR in VHDL.

Further details about simulating in either VHDL or Verilog can be found in the *Synthesis and Simulation Design Guide*, which can be downloaded from the Xilinx website at http://www.xilinx.com/support/sw_manuals/xilinx9/download/

Clocking

The user should generate the clocks in the testbench to supply the user_clk and core_clk domains described in Chapter 2, "Integrated Endpoint Block Functionality." The LogiCORE Endpoint block simplifies this task by requiring the user to only supply a reference clock. A clocking module provided in the top-level wrappers generated by the LogiCORE block generates and connects the clocks appropriately. The details of the clocking module can be found in UG350, *LogiCORE Endpoint Block User Guide for PCI Express*.

Examples

Simulation Setup (ModelSim SE 6.1e on Linux)

This section provides an example of how to set up a simulation for SmartModel support. This is a prerequisite for simulating designs containing the Endpoint block and GTP_DUAL tile.

This example uses ModelSim SE 6.1e, the HDL simulator from Mentor Graphics, Linux as the operating system, and version 9.2i of the Xilinx ISE® Development System. The Synthesis and Simulation Guide provides guidelines and examples for a different HDL simulator or Xilinx ISE Development System.

The following environment variables are set using `setenv`:

• XILINX	Location of the installed Xilinx ISE system
• LMC_HOME	`$XILINX/smartmodel/lin/installed_lin`
• LD_LIBRARY_PATH	`$LMC_HOME/lib/linux.lib:$LD_LIBRARY_PATH`

The initialization file (`modelsim.ini`) will need some modifications. Use the Synthesis and Simulation design guide to determine the appropriate settings.

The location of SmartModel source in the ISE directory tree is automatically set to:

```
$XILINX/smartmodel/lin/image
```

The next step in the setup process is to compile the simulation libraries including SmartModel binaries. Xilinx provides a COMPXLIB tool in ISE to perform the compilation. For the specific example being described, the selected options for the COMPXLIB tool are:

```
compxlib -s mti_se -l all -arch virtex5 -smartmodel_setup
```

These options use the COMPXLIB tool to compile all libraries for all languages for the ModelSim SE 6.1e HDL simulator. The above example also generates the SmartModel. By default, the output directory created for storing the libraries is `$XILINX/language/target_simulator`. VHDL libraries are stored in `$XILINX/vhdl/mti_se`, and Verilog libraries are stored in `$XILINX/verilog/mti_se`.

Running a Simulation

To run a simulation, first compile the files located in the `comp_xilinx.all` file in the `simulation/functional` directory of the output from the LogiCORE Endpoint block. For ModelSim SE, use the provided `simulation_mti.do` file, which executes the following commands to compile the necessary files:

```
(Verilog) vlog -f comp_xilinx.all
```

```
    (VHDL)    vcom -f comp_xilinx.all
```

Next, a downstream port model needs to be specified to emulate the partner design. Since there are a few different products on the market, instructions are not provided on how to do this. Refer to the instructions for your particular model. The recommendations on the partner designs and associated models are not within the scope of this document.

To run the simulation, the Integrated Endpoint wrapper needs the following settings for `vsim`:

```
    (Verilog) vsim -t ps -L UNISIMS_VER ${top_level_tb} glbl
    (VHDL)    vsim -t ps ${top_level_tb}
```

where {`top_level_tb`} is the user's top-level testbench. For Verilog, the `glbl` module needs to be specified for the global reset to operate correctly. The downstream port model can have requirements to add to this call. Refer to the simulation instructions for your downstream port model.

Endpoint Block Attributes

Summary

This appendix lists the attributes that must be set for the Virtex-5 Integrated Endpoint block. All attributes are set in the CORE Generator wrapper, based on choices made in the CORE Generator GUI, and are documented here for reference.

- "Tx and Rx Buffer Layout"
- "Buffer Latency"
- "Initial Flow Control Credits"
- "Extended Capabilities"
- "Endpoint Block Attributes"

Tx and Rx Buffer Layout

Each buffer is divided into separate areas for the VCs, and further divided into separate areas for posted, non-posted, and completion packets. No gaps can be in the FIFO base and limit attribute settings, and they must be in the order shown in Figure A-1.

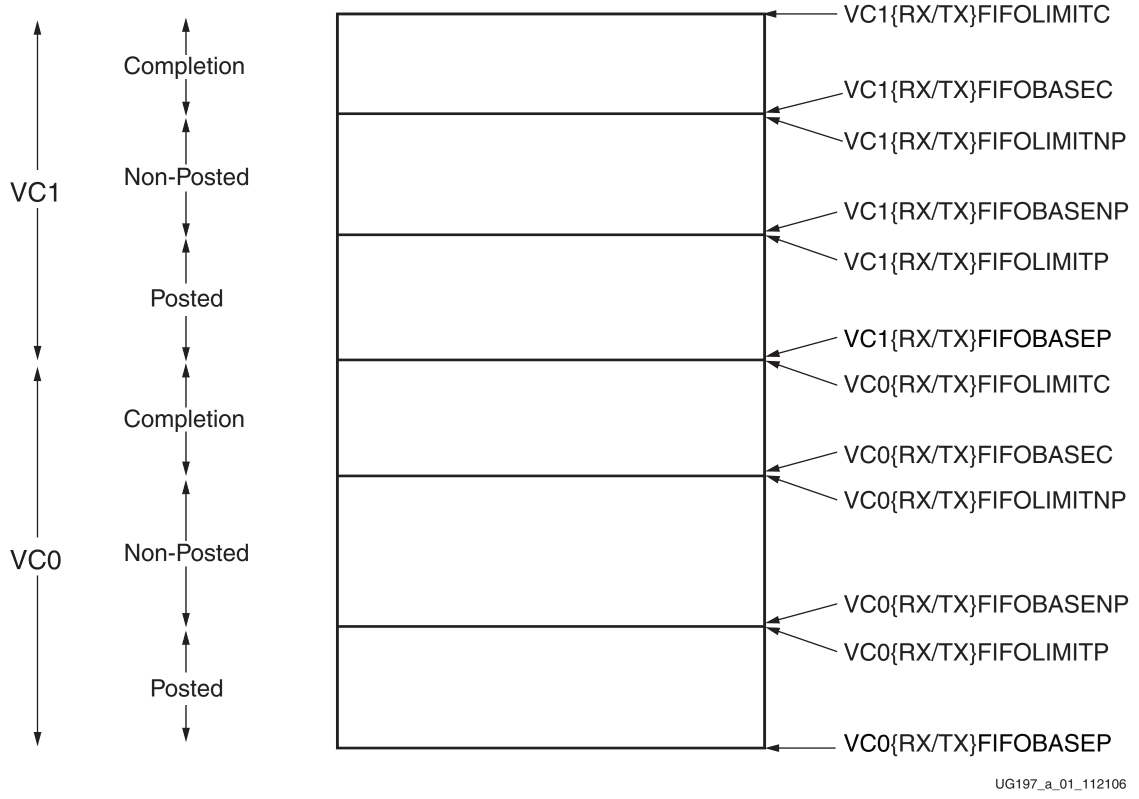

Figure A-1: **Layout for Tx and Rx Buffers**

The selection of the FIFO base and FIFO limit attribute values for any particular VC and packet type can be determined by the amount of RAM available to allocate, the number of packets to be to accommodated in the FIFO at any given time, and the size of those packets. One way of determining the necessary FIFO size is to multiply the maximum packet size by the number of packets. A smaller FIFO could be used, or a larger number of packets accommodated, if the pattern of traffic is such that maximum-sized packets are interspersed with smaller packets. Care should be taken in opting to specify a larger number of packets than the FIFO technically has room for, to ensure that allowance has been made for effects that ordering rules can have on the packets needing to be handled. This applies in particular to VC0 FIFOs, the use of which is shared by the internal configuration block. The other factor that needs to be considered is the absolute limit of eight packets that can be handled by any FIFO.

Note: The number of packets that the FIFO can handle determines the maximum number of flow control credits that can be offered for any VC and for any particular type of packet. The **TOTALCREDITS** attributes are used to initialize the flow control credits. More information on the setting of these attributes is given in Table A-7, page 99.

Because the block RAM interface is 64 bits wide, the VC*FIFOBASE* and VC*FIFOLIMIT* pointers should be defined accordingly. For example, VC0TXFIFOBASEP will be set to 0, so VC0TXFIFOLIMITP = [Posted Tx FIFO size (in bytes)] / 8 - 1. More information on the buffer sizing is given in "Block RAM Interface," page 37.

When VC1 is not used, all the VC1RXFIFOBASE* and VC1RXFIFOLIMIT* pointers must be set to the same value, which is VC0RXFIFOLIMITC + 1. Similarly, all VC1TXFIFOBASE* and VC1TXFIFOLIMIT* pointers must be set to the same value, which is VC0TXFIFOLIMITC + 1.

Buffer Latency

Allowable buffer latency is established by setting the appropriate attributes, shown in Table A-1.

- The TLRAMREADLATENCY attribute applies to both Tx and Rx buffer READs.
- The TLRAMWRITELATENCY attribute applies to both the Tx and Rx buffer WRITEs.
- The Retry buffer latencies (RETRYRAMREADLATENCY and RETRYRAMWRITELATENCY) can be controlled independently.

Table A-1: **Allowed RAM Latency Settings**

Attribute	Applicable Buffer(s)	Allowed Latencies
TLRAMREADLATENCY	Tx, Rx	2 – 6
TLRAMWRITELATENCY	Tx, Rx	1 – 2
RETRYRAMREADLATENCY	Retry	2 – 6
RETRYRAMWRITELATENCY	Retry	1 – 2

The read latency attribute settings are calculated as:

```
  # block RAM output registers (0 or 1)
+ # of fabric read pipeline stages (data)
+ # of fabric read pipeline stages (address/control)
+ 2
  ____________________
  Read Latency Setting
```

Typical settings are given Table A-2. Other cases are possible, but probably not very useful. For the purposes of this calculation, TRUE = `1` and FALSE = `0`.

> ***Note:*** The difference between the TLRAMREADLATENCY and RETRYRAMREADLATENCY settings must be no more than two.

Table A-2: **Memory Interface Read Latency Settings**

Block RAM Output Registers Used	Number of Fabric Pipeline Stages (Data)	Number of Fabric Pipeline Stages (Address/Control)	TLRAMREADLATENCY or RETRYRAMREADLATENCY Setting	Notes
1	0	0	`011b`	Typical setting
1	1	1	`101b`	For very large buffer sizes implemented in slow speed grade parts

The write latency attribute settings are calculated as:

```
  # of fabric write pipeline stages (address and data)
+ 1
  _____________________
  Write Latency Setting
```

The allowed settings are given Table A-3.

Table A-3: **Memory Interface Write Latency Settings**

Number of Fabric Pipeline Stages (Address and Data)	TLRAMWRITELATENCY or RETRYRAMWRITELATENCY Setting	Notes
0	`001b`	Typical setting.
1	`010b`	For very large buffer sizes implemented in slower speed grade devices.

Initial Flow Control Credits

Initial flow control credits attributes should be set according to Table A-4.

Table A-4: **Flow Control Attribute Settings**

Attribute	Value
VC0TOTALCREDITSPH	Maximum of 8
VC0TOTALCREDITSNPH	Maximum of 8
VC0TOTALCREDITSCH	0 if INFINITECOMPLETIONS = TRUE. Maximum of 8 if INFINITECOMPLETIONS = FALSE.
VC0TOTALCREDITSPD	(((VC0RXFIFOLIMITP – VC0RXFIFOBASEP + 1) × 8) – (VC0TOTALCREDITSPH × 24))/16
VC0TOTALCREDITSCD	0 if INFINITECOMPLETIONS = TRUE, otherwise (((VC0RXFIFOLIMITC – VC0RXFIFOBASEC + 1) × 8) – (VC0TOTALCREDITSCH × 16))/16
VC1TOTALCREDITSPH	Maximum of 8
VC1TOTALCREDITSNPH	Maximum of 8
VC1TOTALCREDITSCH	0 if INFINITECOMPLETIONS = TRUE. Maximum of 8 if INFINITECOMPLETIONS = FALSE.
VC1TOTALCREDITSPD	(((VC1RXFIFOLIMITP – VC1RXFIFOBASEP + 1) × 8) – (VC1TOTALCREDITSPH × 24))/16
VC1TOTALCREDITSCD	0 if INFINITECOMPLETIONS = TRUE, otherwise (((VC0RXFIFOLIMITC – VC0RXFIFOBASEC + 1) × 8) – (VC0TOTALCREDITSCH × 16))/16
INFINITECOMPLETIONS	TRUE or FALSE

Note:

- Endpoints are required to advertise infinite completions. The Endpoint block can use completion flow control, but this choice should only be made if the user knows that the link partner is capable of receiving completion flow control updates from an Endpoint.
- Infinite Completions are indicated by setting the flow control credit attribute to 0, and setting the INFINITECOMPLETIONS attribute to TRUE.
- Each posted data credit is 16 bytes.
- The maximum number of packets that can be buffered by each FIFO is eight. Thus, the maximum number of posted data or completion data credits that should be advertised is 8 × XPMAXPAYLOAD/16.

Extended Capabilities

The Endpoint block supports several Extended Capabilities:

- Power Management (PM)
- Message Signaled Interrupt (MSI)
- PCI Express (XP or PCIe)
- Device Serial Number (DSN)
- Virtual Channel (VC)

Attributes are defined for pointers to these capability structures. In addition, there are attributes for pointers to capabilities that are included in the PCI Express specification but not supported by the Virtex-5 Endpoint block:

- Advanced Error Reporting (AER)
- Power Budgeting (PB)

The PCI Express and Power Management capabilities should be enabled in PCI Express compliant implementations. The MSI, DSN, and VC capabilities can be enabled or disabled, depending on the application.

A base pointer and a next pointer must be set for each extended capability, depending on which capabilities are chosen. Each pointer has a default value, which is used if all available capabilities are enabled. If one or more capabilities are disabled, then the appropriate pointers must be changed.

Table A-5: **Default Pointer Attribute Settings**

Attribute	Value	Notes
PMBASEPTR	40h	Cannot be changed.
MSIBASEPTR	48h	Cannot be changed.
XPBASEPTR	60h	Cannot be changed.
AERBASEPTR	10Ch	See Table A-6 for other legal values.
PBBASEPTR	144h	See Table A-6 for other legal values.
DSNBASEPTR	100h	See Table A-6 for other legal values.
VCBASEPTR	154h	See Table A-6 for other legal values.
CAPABILITIESPOINTER	PMBASEPTR	Should not be changed for PCIe compliant systems.
PMCAPABILITYNEXTPTR	MSIBASEPTR	
MSICAPABILITYNEXTPTR	XPBASEPTR	Should not be changed for PCIe compliant systems.
PCIECAPABILITYNEXTPTR	0	Cannot be changed.
AERCAPABILITYNEXTPTR	PBBASEPTR	Cannot be changed.
PBCAPABILITYNEXTPTR	DSNBASEPTR	Cannot be changed.
DSNCAPABILITYNEXTPTR	VCBASEPTR	
VCCAPABILITYNEXTPTR	0	Cannot be changed.

Two linked lists are defined. The method used to disable capabilities depends on which list includes the capability. The capabilities in the first list are defined in the following order: PM, MSI, XP (PCIE).

The start of the first linked list is defined by CAPABILITIESPOINTER, which should be set to the base pointer of the first enabled capability in the above list. The next pointer for the first enabled capability is set to the base pointer of the second enabled capability (if there is one), and so on. The next pointer for the last enabled capability is set to 0, as is done with PCIECAPABILITYNEXTPTR in the default settings. The default values of the base pointers for the capabilities in this first list are always used.

The second list includes the following capabilities in order: AER, PB, DSN, VC.

The base pointer for the first enabled capability in the second list is set to `100h`. If the first enabled capability is DSN, then all the capabilities will use the default base pointers shown in Table A-5. If the VC capability is set to `100h`, then the other capabilities must be moved to occupy the leftover address space.

Table A-6 lists the possible base pointer settings based on the first enabled capability.

Table A-6: **Possible Combinations of Base Pointer Settings**

DSN Enabled	VC Enabled (DSN Not Enabled)	None Enabled (DSN, VC Not Enabled)
DSNBASEPTR = `100h`	VCBASEPTR = `100h`	AERBASEPTR = `110h`
AERBASEPTR = `10Ch`	AERBASEPTR = `12Ch`	PBBASEPTR = `138h`
PBBASEPTR = `144h`	PBBASEPTR = `164h`	DSNBASEPTR = `148h`
VCBASEPTR = `154h`	DSNBASEPTR = `174h`	VCBASEPTR = `154h`

The next pointer for each enabled capability is set to the base pointer of the next enabled capability. The next pointer for the last enabled capability is set to 0.

The next pointer for an unused capability (in either list) should be left at its default value, since the next pointer is not used.

Endpoint Block Attributes

Table A-7 summarizes the Endpoint block attributes.

Table A-7: **Endpoint Block Attributes**

Attribute Name	Type	Description
VC0TXFIFOBASEP	13-bit Hex	Base of address area used for header and data of transmitted posted packets associated with VC0. Must be set to 0.
VC0TXFIFOBASENP	13-bit Hex	Base of address area used for header and data of transmitted non-posted packets associated with VC0.
VC0TXFIFOBASEC	13-bit Hex	Base of address area used for header and data of transmitted completion packets associated with VC0.
VC0TXFIFOLIMITP	13-bit Hex	Top of address area used for header and data of transmitted posted packets associated with VC0. The maximum allowed FIFO size is 32,768 bytes.
VC0TXFIFOLIMITNP	13-bit Hex	Top of address area used for header and data of transmitted non-posted packets associated with VC0. The maximum allowed FIFO size is 512 bytes.
VC0TXFIFOLIMITC	13-bit Hex	Top of address area used for header and data of transmitted completion packets associated with VC0. The maximum allowed FIFO size is 32,768 bytes.
VC0TOTALCREDITSPH	7-bit Hex	Number of credits that should be advertised for posted headers received on VC0. Can be limited by the FIFO size. The maximum supported value is 8.
VC0TOTALCREDITSNPH	7-bit Hex	Number of credits that should be advertised for non-posted headers received on VC0. Can be limited by the FIFO size. The maximum supported value is 8. There is no corresponding VC0TOTALCREDITSNPD attribute as this is always advertised as infinite.
VC0TOTALCREDITSCH	7-bit Hex	Number of credits that should be advertised for completion headers received on VC0. Can be limited by the FIFO size. The maximum supported value is 8. Must be set to 0 when INFINITECOMPLETIONS = TRUE.
VC0TOTALCREDITSPD	11-bit Hex	Number of credits that should be advertised for posted data received on VC0. Limited by the FIFO size and/or the overriding maximum of 8 posted data packets that can be buffered on VC0.
VC0TOTALCREDITSCD	11-bit Hex	Number of credits that should be advertised for Completion data received on VC0. Limited by the FIFO size and/or the overriding maximum of 8 completion data packets that can be buffered on VC0. Must be set to 0 when INFINITECOMPLETIONS = TRUE.

Table A-7: **Endpoint Block Attributes *(Continued)***

Attribute Name	Type	Description
VC0RXFIFOBASEP	13-bit Hex	Base of address area used for header and data of received posted packets associated with VC0. Must be set to 0.
VC0RXFIFOBASENP	13-bit Hex	Base of address area used for header and data of received non-posted packets associated with VC0.
VC0RXFIFOBASEC	13-bit Hex	Base of address area used for header and data of received completion packets associated with VC0.
VC0RXFIFOLIMITP	13-bit Hex	Top of address area used for header and data of received posted packets associated with VC0. The maximum allowed FIFO size is 32,768 bytes.
VC0RXFIFOLIMITNP	13-bit Hex	Top of address area used for header and data of received non-posted packets associated with VC0. The maximum allowed FIFO size is 512 bytes.
VC0RXFIFOLIMITC	13-bit Hex	Top of address area used for header and data of received completion packets associated with VC0. Must be set to VC0RXFIFOLIMITNP + (216 + (9 * XPMAXPAYLOAD)) / 8 -1 when the INFINITECOMPLETIONS attribute is set to TRUE. The maximum allowed FIFO size is 32,768 bytes.
VC1TXFIFOBASEP	13-bit Hex	Base of address area used for header and data of transmitted posted packets associated with VC1.
VC1TXFIFOBASENP	13-bit Hex	Base of address area used for header and data of transmitted non-posted packets associated with VC1.
VC1TXFIFOBASEC	13-bit Hex	Base of address area used for header and data of transmitted completion packets associated with VC1.
VC1TXFIFOLIMITP	13-bit Hex	Top of address area used for header and data of transmitted posted packets associated with VC1. The maximum allowed FIFO size is 32,768 bytes.
VC1TXFIFOLIMITNP	13-bit Hex	Top of address area used for header and data of transmitted non-posted packets associated with VC1. The maximum allowed FIFO size is 512 bytes.
VC1TXFIFOLIMITC	13-bit Hex	Top of address area used for header and data of transmitted completion packets associated with VC1. The maximum allowed FIFO size is 32,768 bytes.
VC1TOTALCREDITSPH	7-bit Hex	Number of credits that should be advertised for posted headers received on VC1. Can be limited by the FIFO size. The maximum supported value is 8.

Table A-7: **Endpoint Block Attributes *(Continued)***

Attribute Name	Type	Description
VC1TOTALCREDITSNPH	7-bit Hex	Number of credits that should be advertised for non-posted headers received on VC1. Can be limited by the FIFO size. The maximum supported value is 8. There is no corresponding VC1TOTALCREDITSNPD attribute as this is always advertised as infinite.
VC1TOTALCREDITSCH	7-bit Hex	Number of credits that should be advertised for Completion headers received on VC1. Can be limited by the FIFO size. The maximum supported value is 8. Must be set to 0 when INFINITECOMPLETIONS = TRUE.
VC1TOTALCREDITSPD	11-bit Hex	Number of credits that should be advertised for posted data received on VC1. Limited by the FIFO size and/or the overriding maximum of 8 posted data packets that can be buffered on VC1.
VC1TOTALCREDITSCD	11-bit Hex	Number of credits that should be advertised for Completion data received on VC1. Limited by the FIFO size and/or the overriding maximum of 8 completion data packets that can be buffered on VC1. Must be set to 0 when INFINITECOMPLETIONS = TRUE.
VC1RXFIFOBASEP	13-bit Hex	Base of address area used for header and data of received posted packets associated with VC1.
VC1RXFIFOBASENP	13-bit Hex	Base of address area used for header and data of received non-posted packets associated with VC1.
VC1RXFIFOBASEC	13-bit Hex	Base of address area used for header and data of received completion packets associated with VC1.
VC1RXFIFOLIMITP	13-bit Hex	Top of address area used for header and data of received posted packets associated with VC1. The maximum allowed FIFO size is 32,768 bytes.
VC1RXFIFOLIMITNP	13-bit Hex	Top of address area used for header and data of received non-posted packets associated with VC1. The maximum allowed FIFO size is 512 bytes.
VC1RXFIFOLIMITC	13-bit Hex	Top of address area used for header and data of received completion packets associated with VC1. Must be set to VC1RXFIFOLIMITNP + (216 + (9 * XPMAXPAYLOAD)) / 8 -1 when the INFINITECOMPLETIONS attribute is set to TRUE and VC1 is used. The maximum allowed FIFO size is 32,768 bytes.
ACTIVELANESIN	8-bit Hex	Bit mask of available active lanes. Valid settings are: `01h`: x1 `03h`: x2 `0Fh`: x4 `FFh`: x8

Table A-7: **Endpoint Block Attributes *(Continued)***

Attribute Name	Type	Description
TXTSNFTS	Integer	Sets the number of FTS sequences generally advertised in the TS1 Ordered Sets (used for all lanes).
TXTSNFTSCOMCLK	Integer	Sets the number of FTS sequences advertised in the TS1 Ordered Sets when the Link Configuration register shows that a common clock source is selected (used for all lanes).
RETRYRAMREADLATENCY	Integer	Specifies the Retry buffer read latency. Valid range is 2 .. 6.
RETRYRAMWRITELATENCY	Integer	Specifies the Retry buffer write latency. Valid settings are 1 or 2.
RETRYRAMSIZE	12-bit Hex	Specifies width of Retry buffer address.
INFINITECOMPLETIONS	Boolean	FALSE specifies the block does not advertise infinite completion credits. TRUE specifies the block does advertise infinite completion flow control credits. Must be set to TRUE.
TLRAMREADLATENCY	Integer	Specifies the read latency for the Tx and Rx buffers in terms of cycles of core_clk for Tx or user_clk for Rx. Valid range is 2 .. 6.
TLRAMWRITELATENCY	Integer	Specifies the write latency for Tx and Rx buffers in terms of cycles of user_clk for Tx or core_clk for Rx. Valid settings are 1 or 2.
L0SEXITLATENCY	Integer	Sets the exit latency from L0s state to be applied where separate clocks are used. Transferred to the Link Capabilities register. Possible values are: 0: less than 64 ns 1: 64 ns to less than 128 ns 2: 128 ns to less than 256 ns 3: 256 ns to less than 512 ns 4: 512 ns to less than 1 µs 5: 1 µs to less than 2 µs 6: 2 µs to 4 µs 7: more than 4 µs
L0SEXITLATENCYCOMCLK	Integer	Sets the exit latency from L0s state to be applied where a common clock is used. Transferred to the Link Capabilities register. Possible values are: 0: less than 64 ns 1: 64 ns to less than 128 ns 2: 128 ns to less than 256 ns 3: 256 ns to less than 512 ns 4: 512 ns to less than 1 µs 5: 1 µs to less than 2 µs 6: 2 µs to 4 µs 7: more than 4 µs

Table A-7: **Endpoint Block Attributes *(Continued)***

Attribute Name	Type	Description
L1EXITLATENCY	Integer	Sets the exit latency from L1 state to be applied where separate clocks are used. Transferred to the Link Capabilities register. Possible values are: 0: less than 1 µs 1: 1 µs to less than 2 µs 2: 2 µs to less than 4 µs 3: 4 µs to less than 8 µs 4: 8 µs to less than 16 µs 5: 16 µs to less than 32 µs 6: 32 µs to 64 µs 7: more than 64 µs
L1EXITLATENCYCOMCLK	Integer	Sets the exit latency from L1 state to be applied where a common clock is used. Transferred to the Link Capabilities register. Possible values are: 0: less than 1 µs 1: 1 µs to less than 2 µs 2: 2 µs to less than 4 µs 3: 4 µs to less than 8 µs 4: 8 µs to less than 16 µs 5: 16 µs to less than 32 µs 6: 32 µs to 64 µs 7: more than 64 µs
BAR0EXIST	Boolean	TRUE specifies that Base Address Register 0 exists.
BAR1EXIST	Boolean	TRUE specifies that Base Address Register 1 exists.
BAR2EXIST	Boolean	TRUE specifies that Base Address Register 2 exists.
BAR3EXIST	Boolean	TRUE specifies that Base Address Register 3 exists.
BAR4EXIST	Boolean	TRUE specifies that Base Address Register 4 exists.
BAR5EXIST	Boolean	TRUE specifies that Base Address Register 5 exists.
BAR0ADDRWIDTH	Integer	Specifies BAR 0 address width. Valid settings are: 0: 32 bits wide 1: 64 bits wide When 64-bit addressing is selected, the BAR occupies both the BAR0 and the BAR1 registers.
BAR1ADDRWIDTH	Integer	Specifies BAR 1 address width. Valid settings are: 0: 32 bits wide 1: 64 bits wide When 64-bit addressing is selected, the BAR occupies both the BAR1 and BAR2 registers.
BAR2ADDRWIDTH	Integer	Specifies BAR 2 address width. Valid settings are: 0: 32 bits wide 1: 64 bits wide When 64-bit addressing is selected, the BAR occupies both the BAR2 and BAR3 registers.

Table A-7: **Endpoint Block Attributes *(Continued)***

Attribute Name	Type	Description
BAR3ADDRWIDTH	Integer	Specifies BAR 3 address width. Valid settings are: `0`: 32 bits wide `1`: 64 bits wide When 64-bit addressing is selected, the BAR occupies both the BAR3 and BAR4 registers.
BAR4ADDRWIDTH	Integer	Specifies BAR 4 address width. Valid settings are: `0`: 32 bits wide `1`: 64 bits wide When 64-bit addressing is selected, the BAR occupies both the BAR4 and BAR5 registers. Because BAR5 must always be 32-bits wide, there is no BAR5ADDRWIDTH attribute.
BAR0PREFETCHABLE	Boolean	Specifies BAR 0 memory region is prefetchable. Valid settings are: TRUE: prefetchable FALSE: not prefetchable
BAR1PREFETCHABLE	Boolean	Specifies BAR 1 memory region is prefetchable. Valid settings are: TRUE: prefetchable FALSE: not prefetchable
BAR2PREFETCHABLE	Boolean	Specifies BAR 2 memory region is prefetchable. Valid settings are: TRUE: prefetchable FALSE: not prefetchable
BAR3PREFETCHABLE	Boolean	Specifies BAR 3 memory region is prefetchable. Valid settings are: TRUE: prefetchable FALSE: not prefetchable
BAR4PREFETCHABLE	Boolean	Specifies BAR 4 memory region is prefetchable. Valid settings are: TRUE: prefetchable FALSE: not prefetchable
BAR5PREFETCHABLE	Boolean	Specifies BAR 5 memory region is prefetchable. Valid settings are: TRUE: prefetchable FALSE: not prefetchable
BAR0IOMEMN	Integer	Selects Memory or I/O Space for BAR 0. Valid settings are: `0`: Memory Space `1`: I/O Space
BAR1IOMEMN	Integer	Selects Memory or I/O Space for BAR 1. Valid settings are: `0`: Memory Space `1`: I/O Space

Table A-7: **Endpoint Block Attributes *(Continued)***

Attribute Name	Type	Description
BAR2IOMEMN	Integer	Selects Memory or I/O Space for BAR 2. Valid settings are: `0`: Memory Space `1`: I/O Space
BAR3IOMEMN	Integer	Selects Memory or I/O Space for BAR 3. Valid settings are: `0`: Memory Space `1`: I/O Space
BAR4IOMEMN	Integer	Selects Memory or I/O Space for BAR 4. Valid settings are: `0`: Memory Space `1`: I/O Space
BAR5IOMEMN	Integer	Selects Memory or I/O Space for BAR 5. Valid settings are: `0`: Memory Space `1`: I/O Space
BAR0MASKWIDTH	6-bit Hex	Specifies top bit of address range for BAR 0. Valid settings are in the range `04h` to `3Fh`.
BAR1MASKWIDTH	6-bit Hex	Specifies top bit of address range for BAR 1. Valid settings are in the range `04h` to `3Fh`.
BAR2MASKWIDTH	6-bit Hex	Specifies top bit of address range for BAR 2. Valid settings are in the range `04h` to `3Fh`.
BAR3MASKWIDTH	6-bit Hex	Specifies top bit of address range for BAR 3. Valid settings are in the range `04h` to `3Fh`.
BAR4MASKWIDTH	6-bit Hex	Specifies top bit of address range for BAR 4. Valid settings are in the range `04h` to `3Fh`.
BAR5MASKWIDTH	6-bit Hex	Specifies top bit of address range for BAR 5. Valid settings are in the range `04h` to `3Fh`.
XPDEVICEPORTTYPE	4-bit Hex	Identifies the type of device/port as follows: `0h`: Endpoint device for PCI Express designs `1h`: Legacy Endpoint device for PCI Express designs Transferred to PCI Express Capabilities register (see Table 2-20, page 52).
XPMAXPAYLOAD	Integer	Specifies maximum payload supported. Valid settings are: 0: 128 bytes 1: 256 bytes 2: 512 bytes 3: 1024 bytes 4: 2048 bytes 5: 4096 bytes Transferred to the Device Capabilities register.

Table A-7: **Endpoint Block Attributes *(Continued)***

Attribute Name	Type	Description
LOWPRIORITYVCCOUNT	Integer	Sets the number of VCs in addition to VC0 that are to be included in the Low Priority VC group. Valid settings are 0 and 1.
VENDORID	16-bit Hex	Unique Manufacturer ID. Transferred to the Vendor ID register.
DEVICEID	16-bit Hex	Unique Device ID. Transferred to the Device ID register.
REVISIONID	8-bit Hex	ID identifying revision of device. Transferred to the Revision ID register.
CLASSCODE	24-bit Hex	Code identifying basic function, subclass and applicable programming interface. Transferred to the Class Code register.
CARDBUSCISPOINTER	32-bit Hex	Pointer to Cardbus data structure. Transferred to the Cardbus CIS Pointer register.
SUBSYSTEMVENDORID	16-bit Hex	ID that can be used to provide additional vendor information to that provided by Vendor ID. Transferred to the Subsystem Vendor ID register.
SUBSYSTEMID	16-bit Hex	ID that can be used to provide additional device information to that provided by Device ID. Transferred to the Subsystem ID register.
CAPABILITIESPOINTER	8-bit Hex	Points to the first capabilities structure
INTERRUPTPIN	8-bit Hex	Indicates mapping for legacy interrupt messages. Valid values are: `0h`: No legacy interrupt messages used. `1h`: INTA
PMCAPABILITYNEXTPTR	8-bit Hex	The offset to the next PCI Capability Structure or `00h` if no further capability structures are available at higher addresses.
PMCAPABILITYDSI	Boolean	Device Specific Initialization (DSI). TRUE: `1` FALSE: `0` Transferred to the PM Capabilities register.
PMCAPABILITYAUXCURRENT	3-bit Hex	Reserved. Must be set to `0h`.
PMCAPABILITYD1SUPPORT	Boolean	D1 Support. Transferred to the PM Capabilities register. Must be set to FALSE.
PMCAPABILITYD2SUPPORT	Boolean	D2 Support. Transferred to the PM Capabilities register. Must be set to FALSE.

Table A-7: **Endpoint Block Attributes *(Continued)***

Attribute Name	Type	Description
PMCAPABILITYPMESUPPORT	5-bit Hex	PME Support. These five bits indicate support for PME generation within D3$_{COLD}$, D3$_{HOT}$, D2, D1, and D0, respectively. Transferred to the PM Capabilities register. Must be set to `0h`.
PMDATA0	8-bit Hex	Reserved. Must be set to `0h`.
PMDATA1	8-bit Hex	Reserved. Must be set to `0h`.
PMDATA2	8-bit Hex	Reserved. Must be set to `0h`.
PMDATA3	8-bit Hex	Reserved. Must be set to `0h`.
PMDATA4	8-bit Hex	Reserved. Must be set to `0h`.
PMDATA5	8-bit Hex	Reserved. Must be set to `0h`.
PMDATA6	8-bit Hex	Reserved. Must be set to `0h`.
PMDATA7	8-bit Hex	Reserved. Must be set to `0h`.
PMDATASCALE0	Integer	Reserved. Must be set to 0.
PMDATASCALE1	Integer	Reserved. Must be set to 0.
PMDATASCALE2	Integer	Reserved. Must be set to 0.
PMDATASCALE3	Integer	Reserved. Must be set to 0.
PMDATASCALE4	Integer	Reserved. Must be set to 0.
PMDATASCALE5	Integer	Reserved. Must be set to 0.
PMDATASCALE6	Integer	Reserved. Must be set to 0.
PMDATASCALE7	Integer	Reserved. Must be set to 0.
PMDATASCALE8	Integer	Reserved. Must be set to 0.
MSICAPABILITYNEXTPTR	8-bit Hex	Pointer to the next item in the capabilities list or `00h` if no further capability structures are available at higher addresses.
MSICAPABILITYMULTIMSGCAP	3-bit Hex	Multiple Message Capable. Each MSI function can request up to four unique messages. System software can read this field to determine the number of messages requested. Number of messages requested are encoded as follows: `0h`: 1 `1h`: 2 `2h`: 4
PCIECAPABILITYNEXTPTR	8-bit Hex	The offset to the next PCI Capability Structure or `00h` if no further capability structures are available at higher addresses. Must be set to `0h`.

Table A-7: **Endpoint Block Attributes *(Continued)***

Attribute Name	Type	Description
DEVICECAPABILITYENDPOINTL0SLATENCY	3-bit Hex	Endpoint L0s Acceptable Latency. Records the latency the Endpoint can withstand on transitions from the L0s state to L0. Valid settings are: `0h`: Maximum of 64 ns `1h`: Maximum of 128 ns `2h`: Maximum of 256 ns `3h`: Maximum of 512 ns `4h`: Maximum of 1 µs `5h`: Maximum of 2 µs `6h`: Maximum of 4 µs `7h`: No limit
DEVICECAPABILITYENDPOINTL1LATENCY	3-bit Hex	Endpoint L1 Acceptable Latency. Records the latency that the Endpoint can withstand on transitions from the L1 state to L0 (if the L1 state is supported). Valid settings are: `0h`: Maximum of 1 µs `1h`: Maximum of 2 µs `2h`: Maximum of 4 µs `3h`: Maximum of 8 µs `4h`: Maximum of 16 µs `5h`: Maximum of 32 µs `6h`: Maximum of 64 µs `7h`: No limit
LINKCAPABILITYMAXLINKWIDTH	6-bit Hex	Maximum Link Width. Valid settings are: `1h`: x1 `2h`: x2 `4h`: x4 `8h`: x8
LINKCAPABILITYASPMSUPPORT	2-bit Hex	Active State PM Support. Indicates the level of active state power management supported by the selected PCI Express Link, encoded as follows: `0h`: Reserved `1h`: L0s entry supported `2h`: Reserved `3h`: L0s and L1 entry supported Must be set to `1h`.
LINKSTATUSSLOTCLOCKCONFIG	Boolean	Slot Clock Configuration. Indicates where the component uses the same physical reference clock that the platform provides on the connector. For a port that connects to the slot, indicates that it uses a clock with a common source to that used by the slot. For an adaptor inserted in the slot, indicates that it uses the same clock source as the slot, not a locally-derived clock source. Transferred to the Link Status register (see Table 2-20, page 52).

Table A-7: **Endpoint Block Attributes *(Continued)***

Attribute Name	Type	Description
AERCAPABILITYNEXTPTR	12-bit Hex	Next Capability Offset. The offset to the next PCI Express capability structure or `000h` if no further capability structures are available at higher addresses. Should be set to PBBASEPTR.
VCCAPABILITYNEXTPTR	12-bit Hex	Next Capability Offset. The offset to the next PCI Express capability structure or `000h` if no further capability structures are available at higher addresses. Must be set to `000h`.
PORTVCCAPABILITYEXTENDEDVCCOUNT	3-bit Hex	Extended VC Count. Indicates the number of (extended) VCs in addition to the default VC supported by the device. Valid settings are `0h` or `1h`.
PORTVCCAPABILITYVCARBCAP	8-bit Hex	VC Arbitration Capability. Indicates the types of VC Arbitration supported for VCs in the LPVC group. Valid settings are: Bit 0: Hardware fixed arbitration scheme (for example, Round Robin) Bit 1: Weighted Round Robin (WRR) arbitration with 32 phases Bits 7-2: Reserved. Valid only when PORTVCCAPABILITYEXTENDEDVCCOUNT > `0h`. Should be set to `0h` when PORTVCCAPABILITYEXTENDEDVCCOUNT = `0h`.
PORTVCCAPABILITYVCARBTABLEOFFSET	8-bit Hex	VC Arbitration Table Offset. Contains the offset of the base of the VC Arbitration Table from the base address of the VC Capability structure, expressed in DQWords (16 bytes). A value of `0h` indicates that the table is not present. Should be set to `0h` when PORTVCCAPABILITYEXTENDEDVCCOUNT = `0h` and set to `8h` when PORTVCCAPABILITYEXTENDEDVCCOUNT = `1h`.
DSNCAPABILITYNEXTPTR	12-bit Hex	Next Capability Offset. The offset to the next PCI Express capability structure above DSN (Device Serial Number) or `000h` if no further capability structures are available at higher addresses.
DEVICESERIALNUMBER	64-bit Hex	PCI Express Device Serial Number. IEEE-defined EUI-64 64-bit extended unique identifier. This identifier includes a 24-bit company ID value assigned by IEEE registration authority and a 40-bit extension assigned by the manufacturer to identify the particular device.

Table A-7: **Endpoint Block Attributes *(Continued)***

Attribute Name	Type	Description
PBCAPABILITYNEXTPTR	12-bit Hex	Next Capability Offset. The offset to the next PCI Express capability structure above power budgeting or `000h` if no further capability structures are available at higher addresses. Should be set to DSNBASEPTR.
PBCAPABILITYDW0BASEPOWER	8-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW0DATASCALE	2-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW0PMSUBSTATE	3-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW0PMSTATE	2-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW0TYPE	3-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW0POWERRAIL	3-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW1BASEPOWER	8-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW1DATASCALE	2-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW1PMSUBSTATE	3-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW1PMSTATE	2-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW1TYPE	3-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW1POWERRAIL	3-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW2BASEPOWER	8-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW2DATASCALE	2-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW2PMSUBSTATE	3-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW2PMSTATE	2-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW2TYPE	3-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW2POWERRAIL	3-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW3BASEPOWER	8-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW3DATASCALE	2-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW3PMSUBSTATE	3-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW3PMSTATE	2-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW3TYPE	3-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYDW3POWERRAIL	3-bit Hex	Reserved. Must be set to `0h`.
PBCAPABILITYSYSTEMALLOCATED	Boolean	Reserved. Must be set to FALSE.

Table A-7: **Endpoint Block Attributes *(Continued)***

Attribute Name	Type	Description
RESETMODE	Boolean	A value of FALSE selects a hierarchical reset scheme that uses four reset domains. A value of TRUE selects a 6-domain reset scheme where each signal resets a separate domain, except for CRMMGMTRSTN, which resets the entire block for either RESETMODE setting. See Table 2-3, page 23 for more information.
CLKDIVIDED	Boolean	Specifies whether the user_clk domain frequency is a divided version of the core_clk domain frequency. Set to FALSE when user_clk frequency is the same as core_clk. Set to TRUE when the user_clk frequency is one half or one quarter the frequency of the core_clk.
AERBASEPTR	12-bit Hex	Location of the base of the Advanced Error Reporting Capability Structure (Table 2-21, page 53).
DSNBASEPTR	12-bit Hex	Location of the base of the Device Serial Number Capability Structure (Table 2-22, page 53).
MSIBASEPTR	12-bit Hex	Location of the base of the Message Signaled Interrupt Capability Structure (Table 2-19, page 52).
PBBASEPTR	12-bit Hex	Location of the base of the Power Budgeting Capability Structure (Table 2-22, page 53).
PMBASEPTR	12-bit Hex	Location of the base of the Power Management Capability Structure (Table 2-18, page 51).
VCBASEPTR	12-bit Hex	Location of the base of the Virtual Channel Capability Structure (Table 2-23, page 53).
XPBASEPTR	8-bit Hex	Location of the base of the PCI Express Capability Structure (Table 2-20, page 52).

Glossary

Click on a letter, or scroll down to view the entire glossary. The *PCI Express Base 1.1 Specification* also includes a Terms and Acronyms section.

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

8

8B/10B Encoding

An encoding method used for encoding 8-bit data bytes as 10-bit Transmission Characters to acheive the following: bit synchronization, DC balance, a simplified design for receivers and transmitters, improved error detection, and easy identification of control characters. An example is shown below. For illustration purposes, the 10-bit code is shown only for RD+ (positive running disparity).

	Data Byte	8B/10B
00	0000 0000	011000 1011
01	0000 0001	100010 1011
02	0000 0010	010010 1011
04	0000 0101	001010 1011
07	0000 0111	000111 0100
08	0000 1000	000110 1011
0F	0000 1111	101000 1011
F0	1111 0000	100100 1110
FF	1111 1111	010100 1110

A

AER

Advanced Error Reporting.

B

BAR

Base Address Register.

Beat

A clock cycle where both the source and destination are ready.

C

Completer

The device addressed by a request. It executes the completer transaction.

Completion

A Packet used to terminate or partially terminate a transaction sequence.

Configuration Space

One of the four address spaces within the PCI Express architecture (the others are I/O, memory and message). Packets with a Configuration Space address are used to configure a device.

CplD

Completion with Data. Used for memory, I/O, and configuration read completions.

CRC

Cyclic redundancy check. A method of error detection. A number is calculated from the data being transmitted, and is sent along with the data. The number is re-calculated at the destination and compared to the transmitted value.

D

Data Link Layer

The middle layer of the PCI Express architecture, that is between the Transaction Layer and the Physical Layer. See "Data Link Layer" on page 17.

Digest

A single 32-bit DW at the end of a TLP, containing an ECRC value.

DL_Down

DL_Down status indicates that there is no connection with another component on the Link, or that the connection with the other component has been lost and is not recoverable by the Physical or Data Link Layer.

DLLP

Data Link Layer Packet. A packet generated in the Data Link Layer to support Link management functions.

DSN

Device Serial Number.

DWORD, DW

Four bytes.

E

ECRC

End-to-end CRC. The ECRC is generated by the source component on the header and data of the TLP and checked by the ultimate destination component. A switch forwards the ECRC untouched, unless the packet is destined for the switch itself. ECRC is optional, and requires Advanced Error Reporting support.

F

Flow control

The protocol that determines how transactions flow between the various ports in a PCI Express fabric. It is a method for communicating receive buffer status from a receiver to a transmitter to prevent receive buffer overflow and allow transmitter compliance with ordering rules.

Function

A logical function corresponding to a PCI function configuration space. Can be used to refer to one function of a multi-function device, or to the only function in a single-function device.

G

H

Hot plug

The ability to swap cards in a powered up system using software control.

Hot swap

The ability to swap cards in a powered up system without software control.

I

In-band

Communications that use the differential wire pairs of the PCI Express lanes for signaling.

Isochronous data transfer

A time sensitive data transfer, such as video. It relies on a guaranteed latency and bandwidth.

J

K

L

Lane

A set of differentially driven signal lines, one for each direction of data flow. A 1-lane PCI Express implementation is sometimes referred to as "x1" (by-1), a 4-lane implementation "x4" (by-4), and so on.

LCRC

Link CRC. A CRC added by the Data Link Layer, that covers the entire TLP and the sequence number. It is checked by the neighboring receiver device.

Legacy Interrupt

PCI interrupt delivery using active Low INTA signal.

Link

A communication path between two PCI Express components. A link consists of one or more lanes.

M

MRd32

Memory Read Request (32-bit).

MRd64

Memory Read Request (64-bit).

MWr32

Memory Write Request (32-bit).

MWr64

Memory Write Request (64-bit).

MSI

Message Signaled Interrupt.

N

O

Ordered set

The sequences of multiples of four characters starting with a comma (COM) character. These special sequences of characters are used during link training, clock compensation, electrical idle, and L0s exit.

P

Packet

A unit of data transferred across a PCI Express Link. The three types of packets are TLPs, DLLPs, and PLPs.

PB

Power Budgeting.

PM

Power Management.

Port

The interface between a PCI Express component and the Link, consisting of the transmitters and receivers on a chip associated with the Link.

Physical Layer

The lowest of the three layers in the PCI Express architecture. See "Physical Layer" on page 18.

Q

QWORD, QW

Eight bytes.

R

Requester

The device that first initiates a PCI Express transaction sequence by executing a requester transaction.

S

Sideband

A signal that is implemented with its own wire. Communication using a sideband signal is not "in-band". Required by certain PCI Express form factors.

Sticky register

A register that retains its state through a hot reset.

Switching fabric

The combination of hardware and software that moves data coming in to a network node out the correct port to the next node in the network. The switching fabric includes data buffers and the use of shared memory, the switching units in a node, the integrated circuits that they contain, and the programming that allows switching paths to be controlled. The switching fabric is independent of the bus technology and infrastructure used to move data between nodes and also separate from the router. The term is sometimes used to mean collectively all switching hardware and software in a network.

T

TL

Transaction Layer.

TLP

Transaction Layer Packet. A Packet generated in the Transaction Layer to convey a Request or Completion.

Transaction Layer

The uppermost of the three layers of the PCI Express architecture. See "Physical Layer" on page 18.

U

V

VC

Virtual channel.

Virtual channel

Virtual connections between two PCI Express devices based on priority-based servicing. A PCI Express Link can support one or multiple virtual channels. There is no correspondence between the number of virtual channels and the number of Lanes.

W

X

x1, x2, x4, x8, etc.

A notation for designating how many lanes are included in the PCI Express link (1, 2, 4, 8 in this example). Pronounced by 1, by 2, by 4, by 8, etc.

Y

Z

Index