PCI Express x16 Data Link and Logical Physical Core Databook

SA15-5767-09

Core version 4.02 or later

February 23, 2006 - IBM Confidential

© Copyright International Business Machines Corporation 2003, 2006

All Rights Reserved
Printed in the United States of America February, 2006

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both.
IBM IBM Logo

Other company, product, and service names may be trademarks or service marks of others.

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in applications such as implantation, life support, or other hazardous uses where malfunction could result in death, bodily injury, or catastrophic property damage. The information contained in this document does not affect or change IBM product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN “AS IS” BASIS. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

This document is VALID ONLY on the date printed.

This document can be printed. The responsibility for using the latest level of this document and for destroying down-level versions lies with the user of the document. To request the most recent version of this document, contact your IBM representative.

IBM Systems and Technology Group
2070 Route 52, Bldg. 330
Hopewell Junction, NY 12533-6351

The IBM home page can be found at **ibm.com**

The IBM Semiconductor solutions home page can be found at **ibm.com**/chips

SA15-5767-09
February 23, 2006 - IBM Confidential

Contents

Preface

About this Book

This book describes the functionality, registers, interfaces, signals, timing specifications, core integration, clocking guidelines, testing and physical design requirements, and the performance and operating environment of the PCI Express x16 Data Link and Logical Physical core.

This document is available to anyone with access to the Design Kit for the core. If you need access, contact your IBM representative.

Who Should Use this Book

This book is for hardware, software, and application developers who need to understand the PCI Express x16 Data Link and Logical Physical core.

Revision Log

Note: Changes from previous versions of this document are marked with blue change bars throughout the text. Italicized text and page numbers in this table are hypertext links. Change bars apply only to the most recent version of the document.

Date	Version	Revision	Page
February 23, 2006	Document Revision 9 This version of the document corresponds to core version 4.02 or later.	Updated descriptions of the supported link negotiation features in *Section 1.2 Features*, *Section 2.1 Supported Link Widths*, and *Table 5 Configuration Interface Signal Descriptions*.	9,14, 24
		Updated terminology so that references to *Section 5.1.1 Physical Link Width* are consistent.	25, 71
October, 2005	Document Revision 8 This version of the document corresponds to core version 4.02 or later.	Core version 4.02 adds the feature described in the new section *Section 5.1.5 Use of Synchronous or Asynchronous Set/Reset Flip-Flops*.	73
		Modified *Section 5.1 Configurability* to describe the new feature.	71
		Modified *Section 5.2.3 Flip/Flop Selection* to describe the new feature.	74
April, 2005	Document Revision 7 This version of the document corresponds to core version 4.00 or later.	Updated *Section 1.4 References*.	11
		Updated *Table 7 Training Control Interface Signal Descriptions*.	28
		Updated *Section 4.4.1.3 Training Control Loopback (Master)*.	30

Date	Version	Revision	Page
January, 2005	Document Revision 6 This version of the document corresponds to core version 4.00 or later.	Changed compliance level to *PCI Express Base Specification 1.1.*	Throughout
		Added Surprise Down functionality. Updated *Section 1.2 Features* and *Section 4.3 Configuration Interface*. Added *Section 4.3.3.7 DL_EC08_SURPRISEDOWN*.	9, 22, 27
		Added configurability to PHY interface width of x1. Updated *Section 1.2 Features* and *Section 5.1.1 Physical Link Width*. Also updated various references through the document.	9, 71
		Corrected RAM address width vs. physical link width specification in *Section 15 Max_Payload_Size versus Replay Buffer Configuration Settings*.	71
		Re-worded some text in *Section 5.2.1.2 Timing* to clarify the timing requirements.	73
September, 2004	Document Revision 5 This version of the document corresponds to core version 3.02 or later.	Updated *Section 1.2 Features* to highlight link width configurability.	9
		Updated *Section 2.1 Supported Link Widths*	14
		Changed signal SYS_EC0C_MAXLINKWIDTH to SYS_MAXLINKWIDTH in *Section 4.3 Configuration Interface* and *Table 5 Configuration Interface Signal Descriptions*..	22, 23
		Added new output signal DL_LANESREVERSED to *Figure 7 Configuration Interface I/O Signals Diagram* and *Table 5 Configuration Interface Signal Descriptions*.	22, 23
		Changed the configuration item “Local NFTS” (described previously in Section 5.1.4) to an input port. See *Section 4.3.1 Use of the SYS_MAXLINKWIDTH and SYS_LOCALNFTS Inputs*.	24
		Updated *Section 4.4.1.2 Training Control Reset* to describe the expanded behavior of the SYS_TCTX_RESET input when the core is in the L2 link state.	30
		Added *Section 4.9.1 PHY Requirements to Support Exit from the L2/3 Ready Link State* to describe the core requirements if a link state transition from L2 back to Detect is required.	64
		Updated *Section 4.10.2 Reading TLP Data from a Replay Buffer Configured for One-Cycle Access*	66
		Added additional configuration parameter: 2 Cycle Replay Access. See changes in *Section 4.10.3 Reading TLP Data from a Replay Buffer Configured for Two-Cycle Access* and *Section 5.1.3 Replay Buffer Read Access Time*.	67, 72
		Update *Section 5 Core Integration*	71
		Update *Section 5.1.3 Replay Buffer Read Access Time*	72
		Modified allowed range of the PCIEXDLP_LOSTIMEOUT configuration parameter and corrected typo in *Section 5.1.4 L0s Entry Timeout Setting*.	73
		Added additional entry to *Table 16 Asynchronous Capture Flip-Flop Pairs in the Design*.	74

Date	Version	Revision	Page
June, 2004	Document Revision 4 This version of the document corresponds to core version 3.00 or later.	**NOTE: This version of the databook is Preliminary. Customers should NOT tape out with this version of the core.**	
		Added DL_INBANDPRESENCE core output to *Figure 7* and *Table 5*. Revised *Section 4.3.2 DL_INBANDPRESENCE and DL_LINKUP* and *Figure 8* to include a description of this new signal.	22, 23
		Corrected text in *Section 4.7.2 Initializing Flow Control for a Virtual Channel* which describes operation of the flow control interfaces while in link power management states.	46
		Fixed typo in *Table 12 PHY Interface Signal Descriptions*; PHY_RXVALID should be RXVALID.	60
		Inserted a new *Section 5.1.1 Physical Link Width* before the existing 5.1.1 to describe a new configuration option. Also modified the following to accommodate the new configuration option: *Section 5.1.2 Replay Buffer Size* *Section 4.5 TLP Transmit Interface* *Section 4.6 TLP Receive Interface* *Section 4.9 PHY Interface* *Section 4.10 TLP Replay Interface* *Section 4.11 Loopback Interface*	71
March, 2004	Document Revision 3 This version of the document corresponds to core version 2.00 or later.	Updated *Figure 1 PCI Express Core Application*.	11
		Corrected signal count in *Figure 3 I/O Signal Diagram* and *Figure 43 PHY Interface Signal Diagram*.	18, 59
		Added *Section 4.5.4 Transmit TLP Length Considerations*.	37
		Updated text in *Section 4.6.2 TLP Receive Interface Protocol* to describe behavior when 1 and 2 DWord TLPs are received.	40
		Updated *Section 4.7 Flow Control Interface* to reflect a change to the Flow Control Initialization interface.	43
		Added a note to the end of *Section 4.7.2 Initializing Flow Control for a Virtual Channel*.	46
		Corrected reference to TL_FCU_REQUEST in *Section 4.7.3 Transmitting Flow Control Update Information*.	47
		Added description of ASPM and PCI-PM L1 exit conditions to *Section 4.8.2.3 Exiting the ASPM L1 Link State* and *Section 4.8.3.3 Exiting the PCI-PM L1 Link State*.	54, 55
		Updated section *Section 5.1 Configurability* to reflect an enhancement made to the Replay Buffer size selection controls.	71
		Added text and *Table 16 Asynchronous Capture Flip-Flop Pairs in the Design* to *Section 5.2.1.2 Timing*.	74
November, 2003	Document Revision 2 This version of the document is released with the Full Design Kit (FDK) and corresponds to core version 1.00 or later.	Full Design Kit publication (Previous document versions were for preliminary publication only.)	–
September, 2003	Preliminary Copy - Revision 1	Updated entire book.	–
September, 2003	Preliminary Copy	Initial release	–

1. Overview

1.1 Introduction

The PCI Express x16 Data Link and Logical Physical (PCIEX16DLP) core provides the majority of the functionality of the PCI Express data link and logical physical layers. It can be used to implement all types of PCI Express devices: root ports, switch ports, bridge PCI Express ports, and endpoints.

1.2 Features

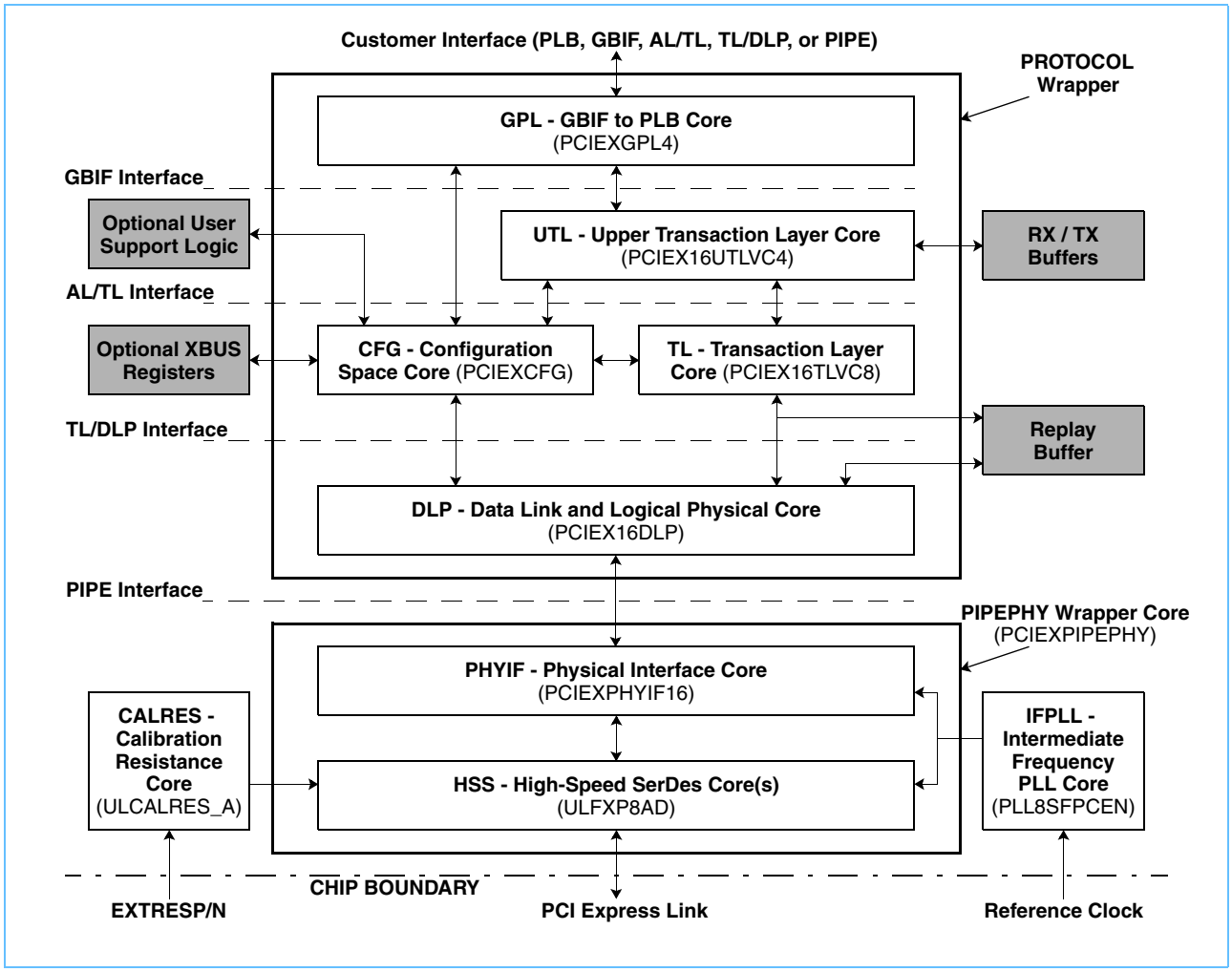
- Supports the *PCI Express Base Specification,* Revision 1.1
- Compatible with the industry standard *PHY Interface for PCI Express Architecture* (PIPE) specification, version 1.00
- Can be configured to support PHY interface widths of x1, x4, x8, or x16
- Supports x1, x4, x8, and x16 negotiated link widths (limited by configured physical interface width)
- Supports all defined TLP packet sizes
- Provides the following data link and logical physical layer functions:
  - Link width negotiation to x1, x4, x8, and x16 link widths (limited by the configured physical link width)
  - Automatic receiver polarity reversal
  - Multi-lane receive data deskew
  - Data scrambling
  - Upstream or downstream port configurability
  - Link training features supported as a downstream facing port:
    - Generation of link reset, link disable, loopback, and scrambling disable
    - Reception of loopback and scrambling disable
  - Link training features supported as an upstream facing port:
    - Generation of loopback and scrambling disable
    - Reception of link reset, disable, loopback, and scrambling disable
  - Advanced error checking and reporting
  - Active state power management: Link states L0s and L1
  - PCI power management: Link states L1 and L2/3 ready
  - DLLP assembly/disassembly with CRC generation and checking
  - TLP sequence number and LCRC generation and checking
  - Surprise Down error detection

- Contains the following interfaces:
  - TLP transmit (data bus width depends on selected PHY interface width)
  - TLP replay buffer (external 1-port RAM required, data bus width depends on selected PHY interface width)
  - TLP receive (data bus width depends on selected PHY interface width)
  - Flow control initialization and update
  - Power management
  - PCI Express configuration space - supports advanced error reporting
- Projected target libraries: 0.13 μm

1.3 Typical Application

Figure 1 on page 11 shows the total available IBM PCI Express solution. The PCI Express Protocol Wrapper core can be configured to include the protocol layers and functionality required for your application, and the PIPEPHY wrapper core can be configured to link widths of x1, x4, x8, and x16. Contact your IBM representative for additional information about configurations available for your application.

Figure 1. PCI Express Core Application



1.4 References

- *PCI Express Base Specification*, Revision 1.1, March 28, 2005
- *PHY Interface for PCI Express Architecture*, Version 1.00, June 19, 2003
- *PCI Express x16 PHY Interface*, IBM Document Number SA15-5766-00
- *PCI Express x16 Transaction Layer*, IBM Document Number SA15-5770-00

1.5 Additional Required Elements

The following additional elements are required to use this core:

- 1-port synchronous RAM (or equivalent)
- PIPE version 1.00 compliant PHY

1.6 Related IBM Cores

The IBM PCI Express PIPE PHY Wrapper core can be used to provide PIPE version 1.0 compliant PHY functionality.

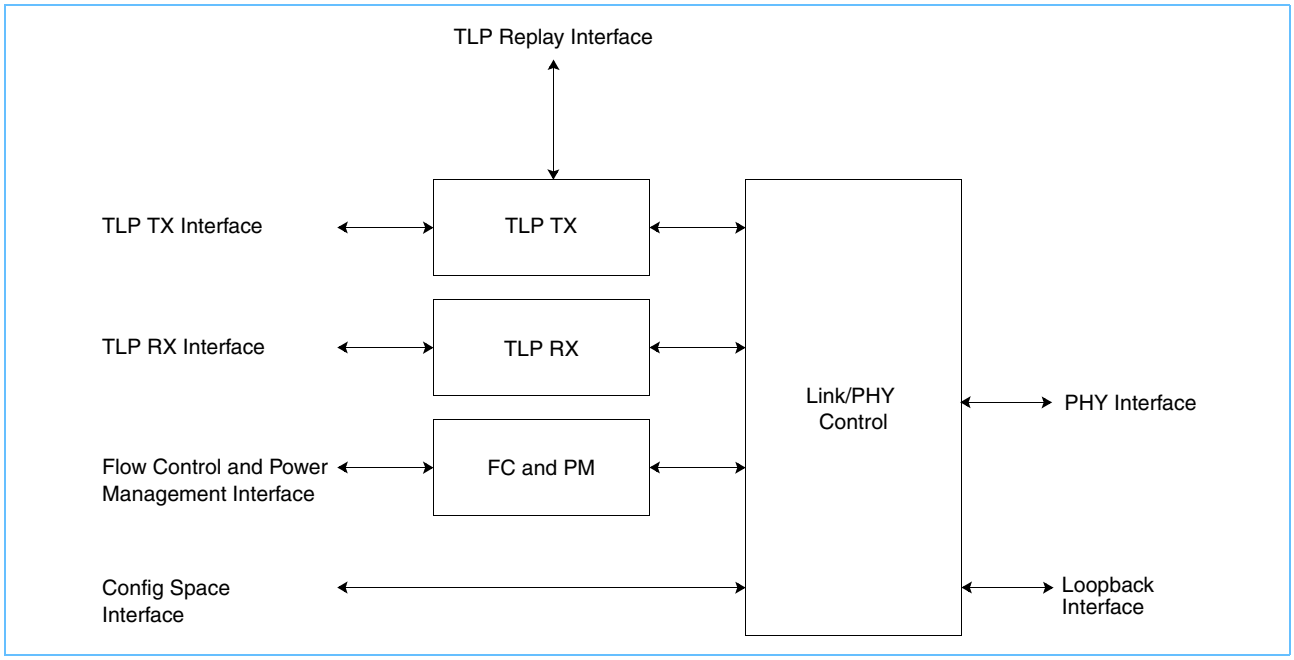
Contact your IBM representative for additional information on this core and other IBM products.

1.7 Conventions

- All data and register content in this document is referred to in little endian format to match the notation used in the *PCI Express Base Specification*.
- All data sizes in this document are defined as:
  - Byte – 8 bits
  - Word – 16 bits
  - Double-word – 32 bits
- Signal timings are calculated from the rising edge of the appropriate clock.

2. Functional Description

Figure 2. Core Block Diagram



The PCIEX16DLP core enables transaction layer packet (TLP) communication across a PCI Express link and control and maintenance of the link. A block diagram of the core is shown in *Figure 2*.

Each of the defined interfaces is explained in detail in *Section 4 Hardware Interface* on page 17.

2.1 Supported Link Widths

The core supports negotiated link widths of x1, x4, x8, and x16 (limited by the core's configured physical link width). In addition, it supports automatic lane reversal when the negotiated link width is the same as the core's configured physical link width. See *Section 5.1.1 Physical Link Width* on page 71 for more information on configuring the physical link width.

2.2 TLP Transmit and Receive Latency

Table 1 shows the best-case latency for the TLP transmit and receive paths through this core. For a more detailed explanation, see *Section 4.5.1 , TLP Transmit Latency, on page 33* and *Section 4.6.1, TLP Receive Latency, on page 39*.

Table 1. Best Case TLP Latency

Direction	Latency (Clock Cycles)
Transmit	3
Receive	5

3. Software Interface

The core does not contain any user accessible registers, and therefore does not have a defined software interface.

4. Hardware Interface

The core uses several interfaces to perform its functions. A top-level interface block diagram in shown in *Figure 3 on page 18*. Each interface is described in detail in subsequent sections.

Port names have a prefix which designates the source of the signal. The convention is:

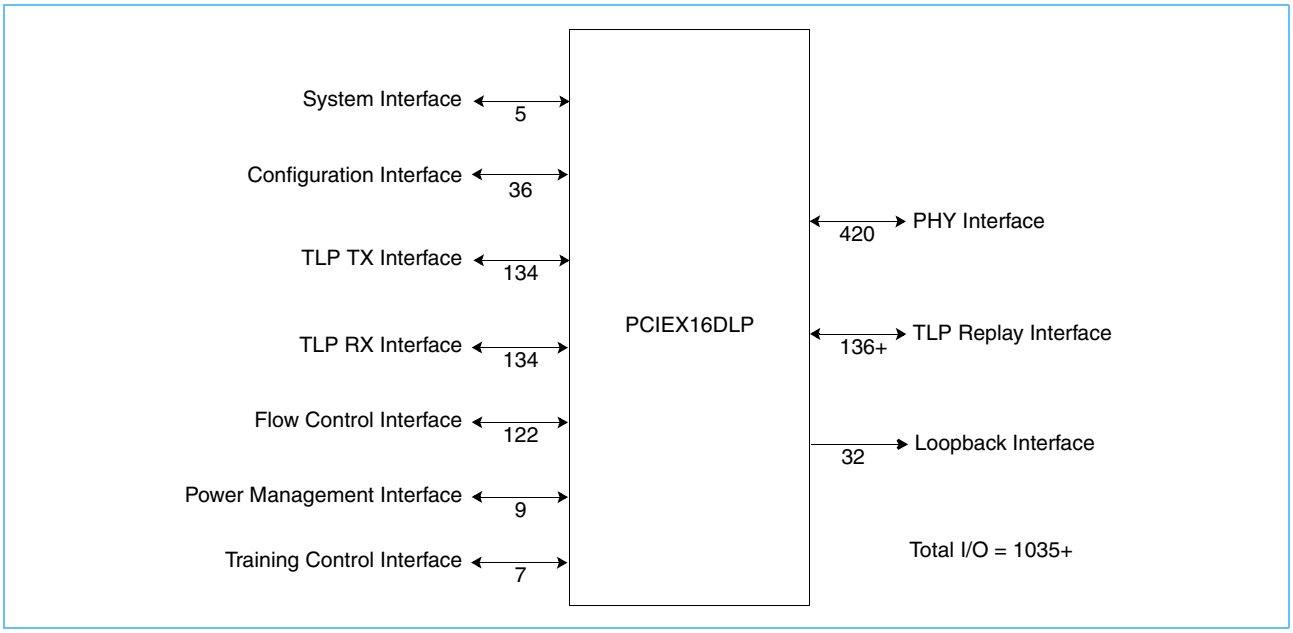
PCLK250	250 MHz system clock.
SYS_*name*	Signals supplied by user logic. Signals of this type which may be sourced by VDD or GND are noted.
DL_*name*	Signals driven by this core. Most connect to transaction layer logic or configuration logic, but some can be used by user logic.
TL_*name*	Signals supplied by transaction layer logic.
CFG_*name*	Signals supplied by configuration logic.
TIE0_*name*	Signals which *must* be sourced by GND.

Each signal also has a timing associated with it. The meaning of each is as follows:

Async	The signal is asynchronous (has no defined timing relationship to any clock).
Clock	The signal is a clock. The frequency will be specified.
Begin	Inputs to the core: The setup time is 85% of the referenced clock cycle.
	Outputs: The valid time is 15% of the referenced clock cycle.
Early	Inputs to the core: The setup time is 75% of the referenced clock cycle.
	Outputs: The valid time is 25% of the referenced clock cycle.
Middle	Inputs to the core: The setup time is 50% of the referenced clock cycle.
	Outputs: The valid time is 50% of the referenced clock cycle.
Late	Inputs to the core: The setup time is 25% of the referenced clock cycle.
	Outputs: The valid time is 75% of the referenced clock cycle.

4.1 Signal Diagram

Figure 3. I/O Signal Diagram



4.2 System Interface

The system interface consists of signals that control the clocking, reset, and simulation speed of the core. The timing of the signals and their asserted level (if any) are noted in their descriptions.

Figure 4. System Interface I/O Signals Diagram

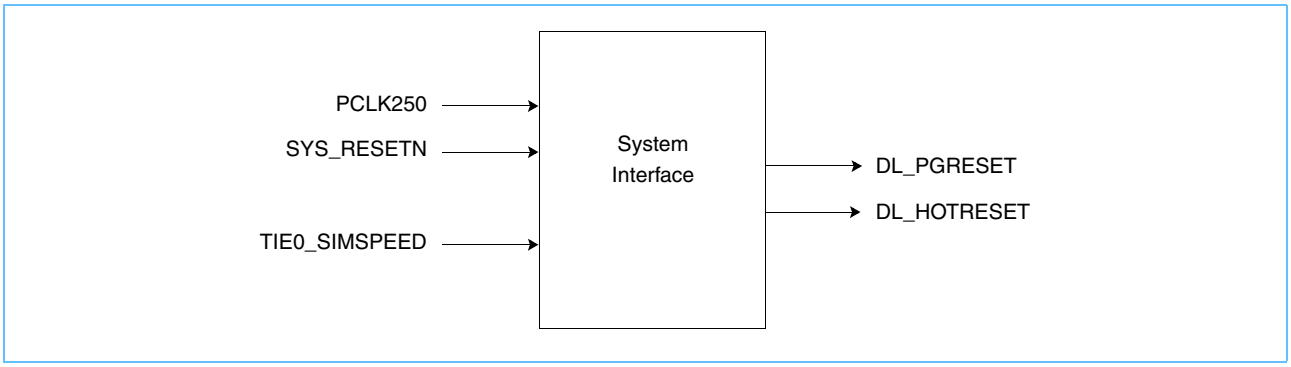


Table 2. System Interface Signal Descriptions

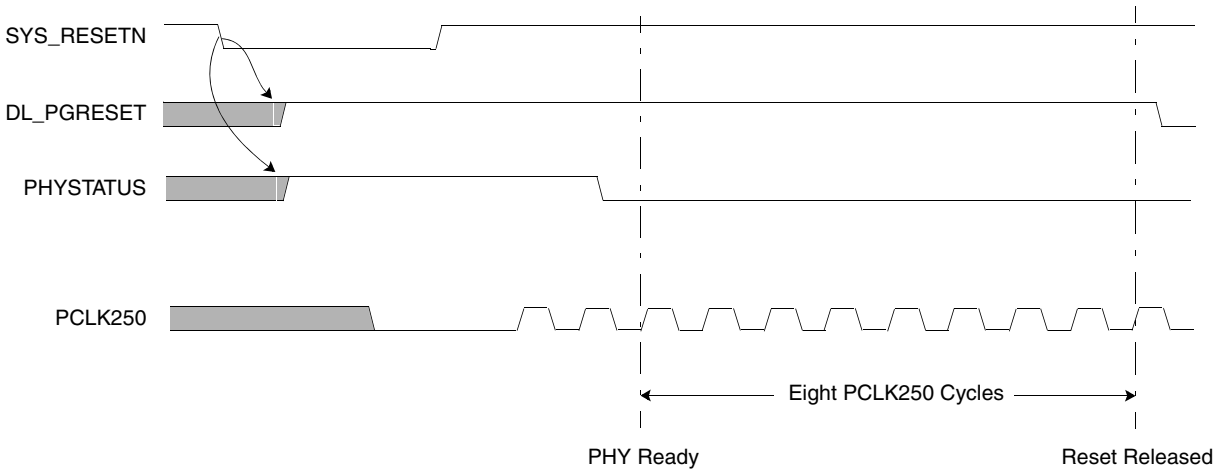
Signal Name	Type	Timing	Description
PCLK250	Input	Clock	250 MHz clock. Sourced from the PHY layer (the PIPE PCLK signal).
SYS_RESETN	Input	Async	This signal is active low. It should be the same signal connected to the RESET# input of the PHY. Asserting this signal causes a core reset to occur. Typically this signal is asserted during system power-up. It may be asserted and deasserted asynchronously.
DL_PGRESET	Output	Early	This signal is active high. It is a partially synchronized version of the SYS_RESETN signal which must be used by the Transaction Layer and Configuration Register logic to return all logic to a Power-On state. It is asserted asynchronously with the assertion of SYS_RESETN, but deasserted synchronously after SYS_RESETN has been deasserted and the PHY has reported that it is operational.
DL_HOTRESET	Output	Early	This signal is active high. It is a synchronous signal that is asserted when the core is configured as an upstream facing port and is receiving “Hot Reset” signalling from upstream. It is not asserted when the core is configured as a downstream facing port.
TIE0_SIMSPEED	Input	N/A	This signal must be tied to GND. It is provided to reduce simulation time spent during link training. See *Section 4.2.2, Hot Reset (DL_HOTRESET), on page 20* for more information.

4.2.1 Power-Good Reset (DL_PGRESET)

A power-good reset proceeds as follows (see *Figure 5 on page 20*):

1. The device is powered on.
2. SYS_RESETN is asserted.
   - The minimum assertion time of SYS_RESETN is determined by the minimum assertion time of the asynchronous set/reset pins of the targeted flip-flops in the library used for synthesis. Typically, the minimum time is less than 10 ns, but note that the PHY may require a much longer time. It is the user’s responsibility to ensure that the requirements are met.
   - This signal should also drive the RESET# input of the attached PHY.
   - The DLP core asserts DL_PGRESET asynchronously with the assertion of SYS_RESETN.
   - The attached PHY asserts its PHYSTATUS output to indicate that it is initializing.
3. SYS_RESETN is deasserted.
   - Before SYS_RESETN is deasserted, the attached PHY must drive its PCLK output (this core’s PCLK250 input) to either a ‘0’ or ‘1’ level. PCLK does not, however, have to be running at this time.
4. The attached PHY completes its initialization process and informs the core that it is ready for normal function by deasserting PHYSTATUS.
   - Before deasserting PHYSTATUS, the attached PHY must be driving its PCLK output (this core’s PCLK250 input) at 250 MHz +/- 300 ppm, not including any spread spectrum clocking effects.
5. The core deasserts DL_PGRESET synchronously with a rising edge of PCLK250 eight clock cycles after PHYSTATUS was sampled as deasserted.
   - All PCI Express logic, including transaction layer logic, configuration register logic, and any user logic that uses the PCLK250 clock source, must reset when DL_PGRESET is asserted. This ensures that all logic using the PCLK250 clock is reset at the same time.

Figure 5. Timing Diagram for Power Good Reset

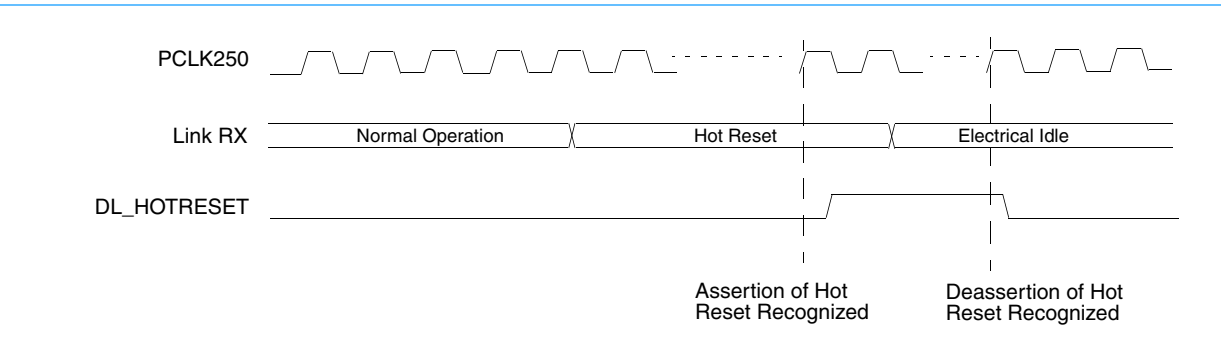


4.2.2 Hot Reset (DL_HOTRESET)

A hot reset proceeds as follows (see *Figure 6*):

1. The port is configured as a downstream component (upstream facing port) and has negotiated a link.
2. The upstream component asserts hot reset signalling on the link.
   - Hot reset is signalled with the training control feature of TS1/2 ordered sets. See the *PCI Express Base Specification* for more information.
   - The core asserts DL_HOTRESET when it recognizes the assertion of hot reset signalling.
3. The upstream component deasserts hot reset signalling on the link.
   - The core deasserts DL_HOTRESET when it recognizes the deassertion of hot reset signalling.
   - The core will attempt to re-establish a link after hot reset signalling is deasserted.

Figure 6. Timing Diagram for Hot Reset



4.2.3 Use of TIE0_SIMSPEED to Improve Simulation Throughput

TIE0_SIMSPEED must be tied to GND. It is provided as a simulation aid only. It may be forced to VDD by a simulator to reduce timeouts and the number of training ordered sets transmitted by the core in the various link training states.

Table 3 describes the timeouts for link training states as specified by the *PCI Express Base Specification* and the resulting timeout when TIE0_SIMSPEED has been forced to VDD. Link training states which do not appear in the table do not have a timeout associated with them and therefore are not affected by TIE0_SIMSPEED.

Table 3. Timeouts Controlled by TIE0_SIMSPEED

Link Training State	Timeout Specified by PCI Express	Timeout when TIE0_SIMSPEED is Forced to VDD
Detect.Quiet	12 ms	2 µs
Detect.Active	12 ms	2 µs
Polling.Active	24 ms	4 µs
Polling.Configuration	48 ms	4 µs
Polling.Speed	2 ms	2 µs
Configuration.Linkwidth.Start	24 ms	4 µs
Configuration.Linkwidth.Accept	2 ms	2 µs
Configuration.Lanenum.Wait	2 ms	2 µs
Configuration.Complete	2 ms	2 µs
Configuration.Idle	2 ms	2 µs
Recovery.RcvrLock	24 ms	4 µs
Recovery.RcvrCfg	48 ms	4 µs
Recovery.Idle	2 ms	2 µs
RX_L0s.FTS	< 32 *[N_FTS+3] ns > 16*[N_FTS+3] ns	32*[N_FTS+2] ns
RX_L0s.FTS (Extended Sync)	< 65 µs > 32 µs	2 µs
Disabled	2 ms	2 µs
Loopback.Entry	2 ms	2 µs
Loopback.Exit	2 ms	2 µs
HotReset	2 ms	2 µs

Table 4 describes the minimum number of TS1 or TS2 ordered sets transmitted in the link training states as specified by the *PCI Express Base Specification* and the resulting number when TIE0_SIMSPEED has been forced to VDD. Link training states that do not appear in the table, either do not cause TS1 or TS2 ordered sets to be transmitted or do not have a specified minimum and therefore are not affected by TIE0_SIMSPEED.

Table 4. Number of Ordered Sets Controlled by TIE0_SIMSPEED (Page 1 of 2)

Link Training State	Number of Ordered Sets Specified by PCI Express	Number of Ordered Sets When TIE0_SIMSPEED is Forced to VDD
Polling.Active	1024	48
Polling.Configuration	16	16
Configuration.Complete	16	16
Configuration.Idle	16	16

Table 4. Number of Ordered Sets Controlled by TIE0_SIMSPEED (Page 2 of 2)

Link Training State	Number of Ordered Sets Specified by PCI Express	Number of Ordered Sets When TIE0_SIMSPEED is Forced to VDD
Recovery.RcvrLock (Extended Sync)	1024	48
Recovery.RcvrCfg	16	16
TX_L0s.FTS	N_FTS	N_FTS
Disabled	16	16

4.3 Configuration Interface

The configuration interface defines the configuration of the PCI Express device using this core. It is also used to report the state of the link.

All input signals, except as noted, must be synchronous to the rising edge of PCLK250. All output signals are synchronous to the rising edge of PCLK250. All signals that are defined to have an asserted state are considered positive-active; meaning that their asserted level is a logical ‘1’.

Figure 7. Configuration Interface I/O Signals Diagram

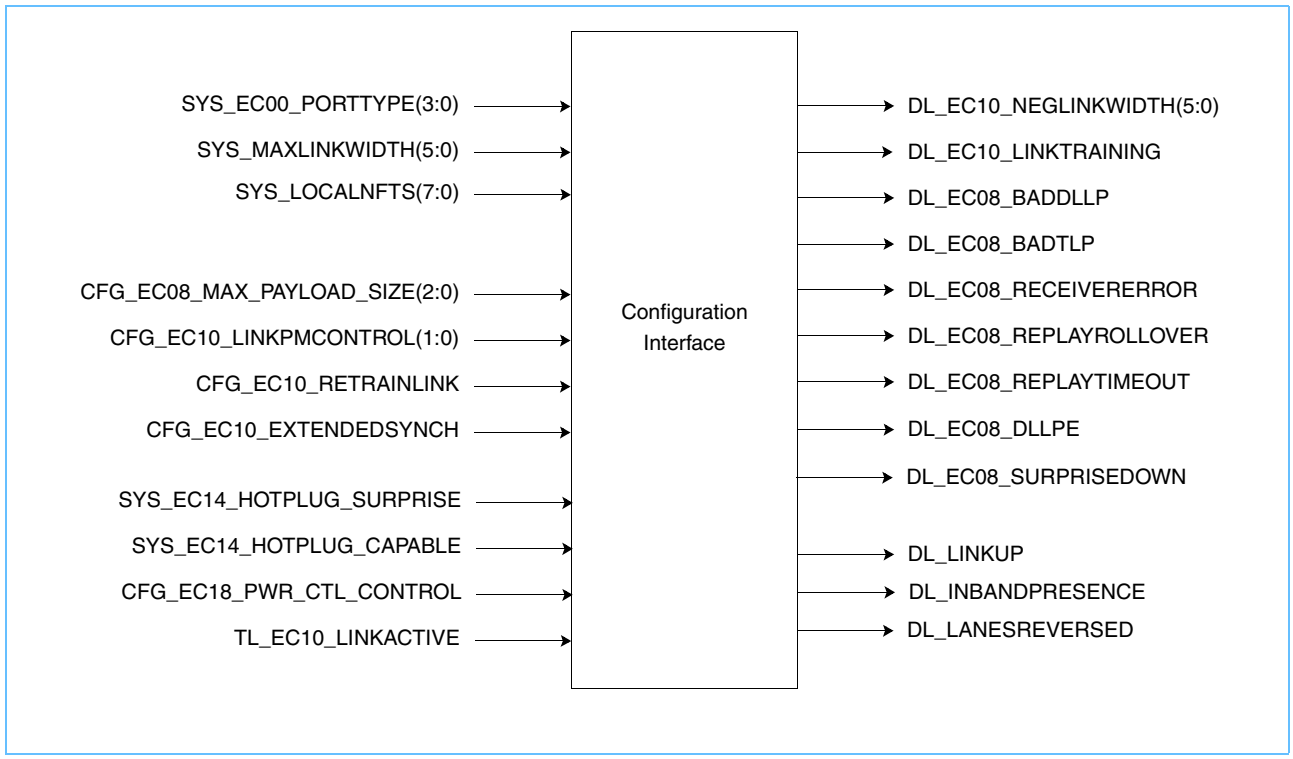


Table 5. Configuration Interface Signal Descriptions (Page 1 of 2)

Signal Name	I/O Type	Timing	Description
SYS_EC00_PORTTYPE(3:0)	Input	Begin	This signal is the Device/Port Type field of the PCI Express Capabilities register. It may be driven by VDD/GND. If not driven by VDD/GND, it must be stable after SYS_RESETN is deasserted. Its setting must be equivalent to that of the Device/Port Type field reported by the configuration logic of the port.
SYS_MAXLINKWIDTH(5:0)	Input	Begin	This signal defines the maximum link width that the core attempts to negotiate. It may be driven by VDD/GND. If not driven by VDD/GND, it must not be changed after SYS_RESETN is deasserted. Its setting must be less than or equal to the Maximum Link Width defined in the Link Capabilities Register of the port. See *Section 4.3.1 Use of the SYS_MAXLINKWIDTH and SYS_LOCALNFTS Inputs* on page 24 for more information. **Note:** This signal does not exist when the core is configured for a x1 Physical Link Width.
SYS_LOCALNFTS(7:0)	Input	Begin	This signal defines the number of FTS ordered sets required by the local device's receiver to exit the L0s state. It may be driven by VDD/GND. If not driven by VDD/GND, it must not be changed after SYS_RESETN is deasserted. See *Section 4.3.1 Use of the SYS_MAXLINKWIDTH and SYS_LOCALNFTS Inputs* on page 24 for more information.
CFG_EC08_MAX_PAYLOAD_SIZE(2:0)	Input	Early	This signal is the Maximum Payload Size field in the Device Control register. It may be driven by VDD/GND. Its setting must be equivalent to that of the Maximum Payload Size field reported by the configuration logic of the port.
CFG_EC10_LINKPMCONTROL(1:0)	Input	Early	This signal is the Active State Link PM Control field of the Link Control register. This signal must be driven by registers which reflect the Active State Link PM Control field of the Link Control register.
CFG_EC10_RETRAINLINK	Input	Early	A link retrain may be initiated by driving this signal to '1' for one cycle. It may be generated from the Retrain Link bit in the Link Control register. **Note:** The Retrain Link bit of the Link Control register is reserved for upstream facing port types, and so this input must be driven by GND in that case.
CFG_EC10_EXTENDEDSYNCH	Input	Early	This signal is the Extended Synch bit of the Link Control Register. It must be driven by a register which reflects the Extended Synch bit of the Link Control register.
SYS_EC14_HOTPLUG_SURPRISE	Input	Early	This signal is the Hot-Plug Surprise bit of the Slot Capabilities Register. It must be driven by VDD/GND or a register which reflects the Hot-Plug Surprise bit of the Slot Capabilities Register.
SYS_EC14_HOTPLUG_CAPABLE	Input	Early	This signal is the Hot-Plug Capable bit of the Slot Capabilities Register. It must be driven by VDD/GND or a register which reflects the Hot-Plug Capable bit of the Slot Capabilities Register.
CFG_EC18_PWR_CTL_CONTROL	Input	Early	This signal is the Power Controller Control bit of the Slot Control Register. It must be driven by a register which reflects the Power Controller Control bit of the Slot Control Register.
TL_EC10_LINKACTIVE	Input	Early	This signal from transaction layer logic must be driven to '1' when VC0 flow control negotiation is complete. It indicates that the link is capable of transmitting and receiving TLPs.
DL_EC10_NEGLINKWIDTH(5:0)	Output	Early	This signal is the link width established by the core when link negotiation has completed. It may be used as the Negotiated Link Width field of the Link Status register.

Table 5. Configuration Interface Signal Descriptions (Page 2 of 2)

Signal Name	I/O Type	Timing	Description
DL_EC10_LINKTRAINING	Output	Early	This signal is asserted when link training is in progress. It may be used as the Link Training bit of the Link Status register.
DL_EC08_BADDLLP	Output	Early	This signal is asserted for one cycle when the core has detected a bad DLLP. It may be used to set the Bad DLLP Status bit of the Correctable Error Status register.
DL_EC08_BADTLP	Output	Early	This signal is asserted for one cycle when the core has detected a bad TLP. It may be used to set the Bad TLP Status bit of the Correctable Error Status register.
DL_EC08_RECEIVERERROR	Output	Early	This signal is asserted for one cycle when the core has detected a Receiver Error. It may be used to set the Receiver Error Status bit of the Correctable Error Status register.
DL_EC08_REPLAYROLLOVER	Output	Early	This signal is asserted for one cycle when the REPLAY_NUM counter rolls over and a link retrain is initiated. It may be used to set the REPLAY_NUM Rollover Status bit of the Correctable Error Status register.
DL_EC08_REPLAYTIMEOUT	Output	Early	This signal is asserted for one cycle when the REPLAY_TIMER times out and a replay is initiated. It may be used to set the Replay Timer Timeout Status bit of the Correctable Error Status register.
DL_EC08_DLLPE	Output	Early	This signal is asserted for one cycle when a data link protocol error has been detected. It may be used to set the Data Link Protocol Error Status bit of the Uncorrectable Error Status register.
DL_EC08_SURPRISEDOWN	Output	Early	This signal is asserted for one cycle when a link surprise down event has been detected. It may be used to set the Surprise Down Error Status bit of the Uncorrectable Error Status register.
DL_LINKUP	Output	Early	This signal is asserted when the core has established a link with a remote component.
DL_INBANDPRESENCE	Output	Early	This signal is asserted when the core has detected that a remote component is connected to the port.
DL_LANESREVERSED	Output	Early	This signal is asserted when the core has negotiated a link with lane reversal. When not asserted, the link's Lane 0 is mapped to PIPE lane 0. When asserted, Lane 0 is mapped to the highest numbered PIPE lane: • PIPE Lane 3 when the core is configured for a physical link width of x4 • PIPE Lane 7 when the core is configured for a physical link width of x8 • PIPE Lane 15 when the core is configured for a physical link width of x16

4.3.1 Use of the SYS_MAXLINKWIDTH and SYS_LOCALNFTS Inputs

SYS_MAXLINKWIDTH and SYS_LOCALNFTS are used to define link configuration parameters.

4.3.1.1 SYS_MAXLINKWIDTH

The SYS_MAXLINKWIDTH input defines the maximum link width that the core attempts to configure during link negotiation. The possible settings vary based on the Physical Link Width configuration parameter (see *Section 5.1.1 Physical Link Width* on page 71) and are shown in *Table 6* on page 25.

Table 6. Possible Settings of SYS_MAXLINKWIDTH

Configured Physical Link Width	Possible Settings of SYS_MAXLINKWIDTH
x1 - PCIEXD_LINKWIDTH_X1	N/A
x4 - PCIEXD_LINKWIDTH_X4	6'b000001 - x1 6'b000100 - x4
x8 - PCIEXD_LINKWIDTH_X8	6'b000000 - x1 6'b000100 - x4 6'b001000 - x8
x16 - PCIEXD_LINKWIDTH_X16	6'b000001 - x1 6'b000100 - x4 6'b001000 - x8 6'b010000 - x16

Note: The Maximum Link Width field of the Link Capabilities Register of the port must always report a value equal to or larger than the largest SYS_MAXLINKWIDTH setting to be used.

4.3.1.2 SYS_LOCALNFTS

SYS_LOCALNFTS defines the number of FTS ordered sets required by the local device's receiver to exit the L0s link state. This value is advertised to the port at the other end of the link during link configuration and affects the operation of this core's Replay Timeout logic and Receiver L0s exit timeout logic.

The value of SYS_LOCALNFTS must account for the number of FTS ordered sets required by this core (one) and the number of FTS ordered sets required by the PHY attached to this core. As an example, if the PHY requires six FTS ordered sets, SYS_LOCALNFTS should be set to at least seven. Using a value too small to allow the PHY and this core to synchronize, results in the entire link being retrained (which takes more time to complete than the transmission of additional FTS ordered sets).

The equivalent time required to transmit the FTS ordered sets specified by SYS_LOCALNFTS should also be reflected in the L0s exit latency field of the link capabilities register of the local device. The time is (SYS_LOCALNFTS + 1) * 16 ns.

The SYS_LOCALNFTS input is an 8-bit signal and must be set to a value of 1 to 255.

4.3.2 DL_INBANDPRESENCE and DL_LINKUP

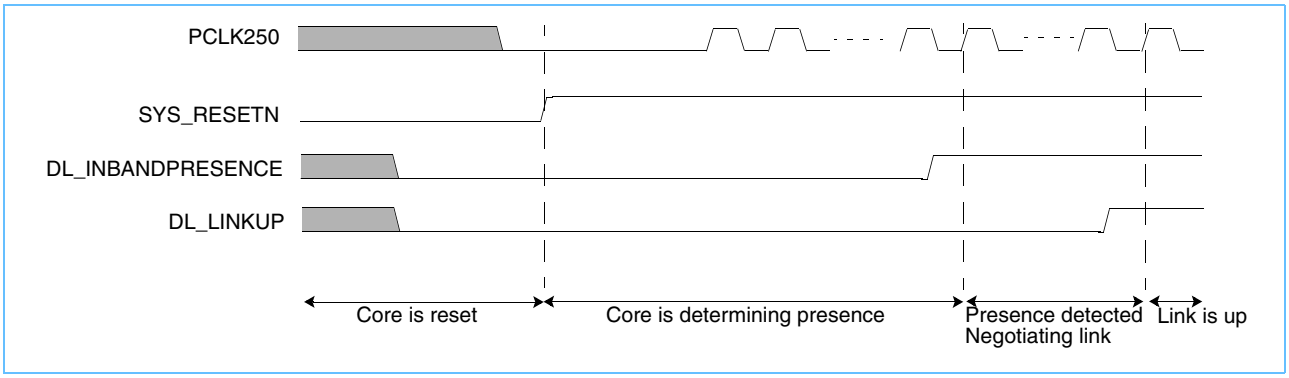
The DL_INBANDPRESENCE and DL_LINKUP signals correspond to the "In-Band Presence" and "LinkUp" columns of Table 4-4 in the *PCI Express Base Specification*.

The DL_INBANDPRESENCE signal describes the presence detect status of the port. When asserted, the port is connected to a remote port. Note that having a connection does not mean that a link has been established or can be established. This signal is intended for use in downstream facing ports (although it exists and will be driven when this core is configured as either a downstream or upstream facing port) as a way of indicating the presence of an adapter in a slot. Also see the description of the Presence Detect State bit in the Slot Status Register in the *PCI Express Base Specification* for more information on usage of this signal.

The DL_LINKUP signal describes the state of the link. When deasserted this signal indicates that there is currently no link established with a remote port. All TLP and DLLP interfaces are non-operational. This means that transaction layer logic must not attempt to transmit any TLPs or DLLPs and no TLPs or DLLPs will be received. When DL_LINKUP changes from a deasserted state to an asserted state, the transaction layer logic must initialize flow control for VC0 before attempting to transmit any TLPs or DLLPs (other than those required to initialize VC0). The core does not contain an interlock to prevent such actions from taking place.

DL_INBANDPRESENCE and DL_LINKUP are deasserted when SYS_RESETN is asserted. Their normal behavior after reset is shown in *Figure 8*:

Figure 8. DL_INBANDPRESENCE and DL_LINKUP After Reset



4.3.3 Error Reporting

The core provides output signals that can be used to implement standard and/or advanced error reporting for the port. The following sections define the output signals and the detected errors reported by each.

4.3.3.1 DL_EC08_RECEIVERERROR

This signal is asserted for one PCLK250 cycle when a receiver error is detected. A receiver error is defined as one or more of the following conditions:

- Any error condition detected in the data stream of any enabled lane and reported by the attached PHY. This includes 8b/10b decode/disparity errors, elastic FIFO overflow/underflow errors, and loss of symbol lock.
- A framing error during reception of a TLP or DLLP. This includes at least:
  - STP/SDP/END/EDB symbols received on lanes of a multi-lane link which are not defined to be valid lanes on which to transmit those symbols
  - END symbol received without a corresponding STP/SDP symbol
  - EDB symbol received without a corresponding STP symbol
  - Two SDP symbols received in one symbol time on a multi-lane link
  - Two STP symbols received in one symbol time on a multi-lane link
- A lane-to-lane deskew error on a multi-lane link. This includes detection of a COM symbol not received on all lanes in the same symbol time, and detected skew larger than five symbol times.

4.3.3.2 DL_EC08_BADDLLP

This signal is asserted for one PCLK250 cycle when a bad DLLP is received. A bad DLLP is defined as a DLLP that was received without a receiver error but which fails the CRC check.

4.3.3.3 DL_EC08_BADTLP

This signal is asserted for one PCLK250 cycle when a bad TLP is received. A bad TLP is defined as a TLP that was received without a receiver error, but which fails the CRC check; or, a TLP which was received out-of-order (the sequence number does not correspond to the next expected one or one previously received).

4.3.3.4 DL_EC08_REPLAYROLLOVER

This signal is asserted for one PCLK250 cycle for every four times that the contents of the replay buffer has been retransmitted.

4.3.3.5 DL_EC08_REPLAYTIMEOUT

This signal is asserted for one PCLK250 cycle when the core has not received an Ack or Nak DLLP for the REPLAY_TIMER timeout period defined in the *PCI Express Base Specification*. The timeout period varies with the negotiated link width and the value of the CFG_EC08_MAX_PAYLOAD_SIZE input.

4.3.3.6 DL_EC08_DLLPE

This signal is asserted for one PCLK250 cycle when an Ack or Nak DLLP is received, but the specified sequence number does not match the last acknowledged sequence number or a currently outstanding sequence number.

4.3.3.7 DL_EC08_SURPRISEDOWN

This signal is asserted for one PCLK250 cycle when the link is dropped unexpectedly. It is asserted when the core is operating as a downstream facing port , the TL_EC10_LINKACTIVE input is asserted, and DL_LINKUP is de-asserted (indicating the link has been dropped) except in the following cases:

- The link was directed to the Disabled state. See *Section 4.4.1.1 Training Control Disable* on page 29.
- The link was directed to the Hot Reset state. See *Section 4.4.1.2 Training Control Reset* on page 30.
- The link was directed to the L2 state.
- The Hot-Plug Surprise bit of the Slot Capabilities Register is ‘1’.
- The Hot-Plug Capable bit of the Slot Capabilities Register is ‘1’ and the Power Controller Control bit of the Slot Control Register is ‘1’ (indicating that power is off to the slot).

.This signal can be used to set the Surprise Down Error Status bit of the Uncorrectable Error Status Register.

4.4 Training Control Interface

The Training Control Interface of the core may be used to initiate PCI Express training control features. It is also used to report training control status.

When configured as a downstream facing port, the following features may be initiated:

- Hot Reset
- Disable Link
- Disable Scrambling
- Loopback

and the following conditions are reported:

- Link Disabled
- Loopback
- Scrambling Disabled

When configured as an upstream facing port, the following features may be initiated:

- Disable Scrambling
- Loopback

and the following conditions are reported:

- Link Disabled
- Loopback
- Scrambling Disabled

Note: When configured as an upstream facing port, reception of the 'Hot Reset' training control feature is reported with the DL_HOTRESET core output described in *Section 4.2, System Interface, on page 18*. As a result, it does not appear in this interface.

All input signals, except as noted, must be synchronous to the rising edge of PCLK250. All output signals are synchronous to the rising edge of PCLK250. All signals that are defined to have an asserted state are considered positive-active; meaning that their asserted level is a logical '1'.

Figure 9. Training Control Interface I/O Signals Diagram

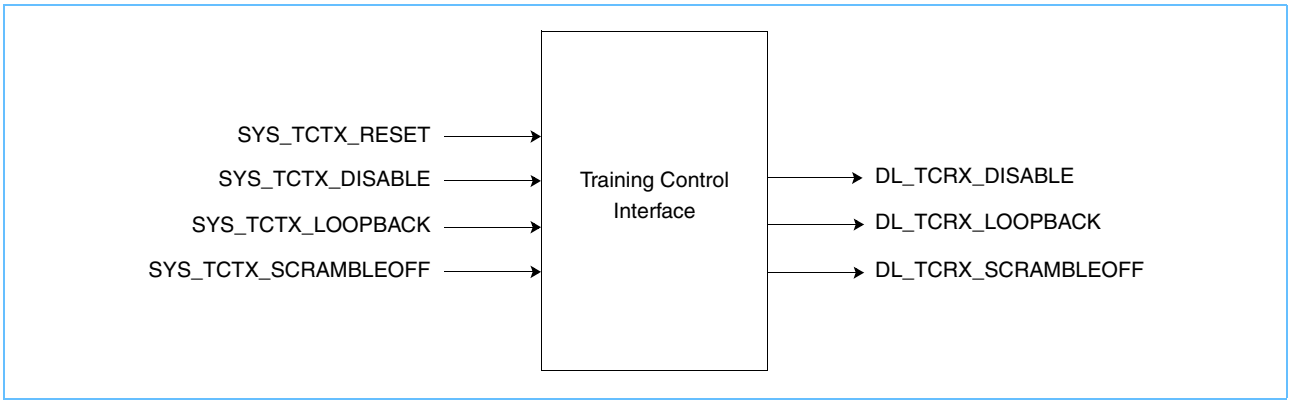


Table 7. Training Control Interface Signal Descriptions

Signal Name	I/O Type	Timing	Description
SYS_TCTX_RESET	Input	Early	Assert this signal when the core is configured as a downstream facing port (root port, etc...) and the hierarchy below the port is to be reset. Normally, this signal should be driven by the Secondary Bus Reset bit of the Bridge Control register.
SYS_TCTX_DISABLE	Input	Early	Assert this signal when the core is configured as a downstream facing port (root port, etc...) and the device attached to the port is to be disabled. Normally, this signal should be driven by the Link Disable bit of the Link Control register.
SYS_TCTX_LOOPBACK	Input	Early	Assert this signal when the core is configured as either an upstream or downstream facing port and loopback mode is desired. The source of this signal is application specific. See *Section 4.4.1.3 Training Control Loopback (Master)* on page 30 for more information.
SYS_TCTX_SCRAMBLEOFF	Input	Early	Assert this signal when the core is configured as either an upstream or downstream facing port and scrambling should be disabled on the link. The source of this signal is application specific. It must be asserted before the link is configured to be effective.
DL_TCRX_DISABLE	Output	Early	Status output. Asserted when the link is in the Disabled state.
DL_TCRX_LOOPBACK	Output	Early	Status output. Asserted when the link is in the Loopback state.
DL_TCRX_SCRAMBLEOFF	Output	Early	Status output. Asserted when a link has been established with scrambling disabled.

4.4.1 Training Control Priorities

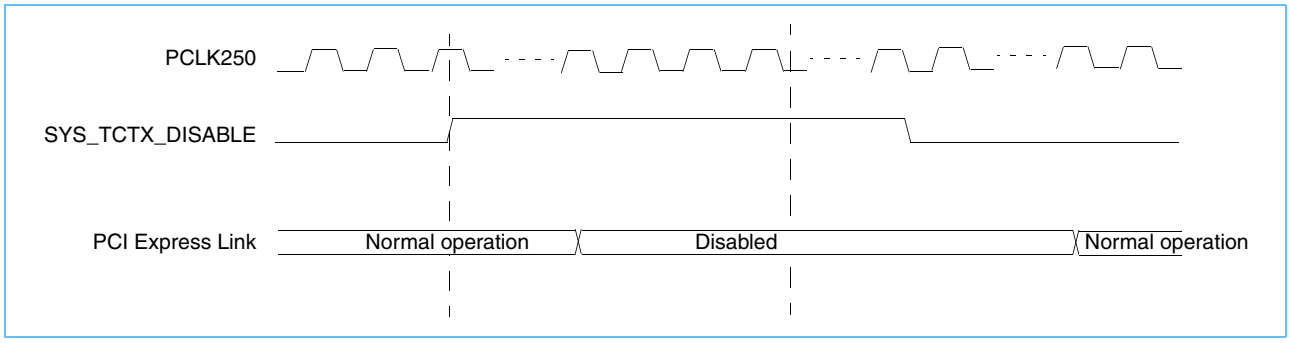
SYS_TCTX_RESET, SYS_TCTX_DISABLE, SYS_TCTX_LOOPBACK, and SYS_TCTX_SCRAMBLEOFF may be asserted at any time. However, the core conforms to the *PCI Express Base Specification* and will not assert more than one training control bit at any given time. The signalling priority is:

1. SYS_TCTX_DISABLE
2. SYS_TCTX_RESET
3. SYS_TCTX_LOOPBACK
4. SYS_TCTX_SCRAMBLEOFF

4.4.1.1 Training Control Disable

Asserting SYS_TCTX_DISABLE causes a downstream-facing port to disable the link if a link is being negotiated or is currently operational. The link remains disabled until SYS_TCTX_DISABLE is deasserted. An attached device cannot be reset with SYS_TCTX_RESET while the link is disabled.

Figure 10. Training Control Disable



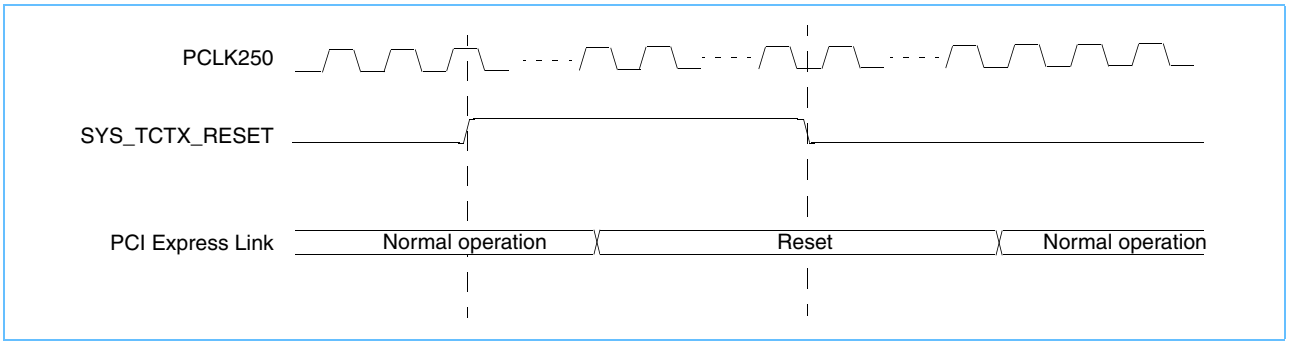
4.4.1.2 Training Control Reset

Asserting SYS_TCTX_RESET causes a downstream facing port to initiate the Hot Reset link protocol and resets the hierarchy attached to the port.

If both SYS_TCTX_RESET and SYS_TCTX_DISABLE are asserted, the link is disabled.

Note: Asserting SYS_TCTX_RESET when the core is in the PCI-PM L2/L3 ready link state (signalled with DL_PM_INL23) causes the core to exit that link state and attempt to negotiate a link. See *Section 4.9.1 PHY Requirements to Support Exit from the L2/3 Ready Link State* on page 64 for additional requirements on PCLK250 and PHYSTATUS.

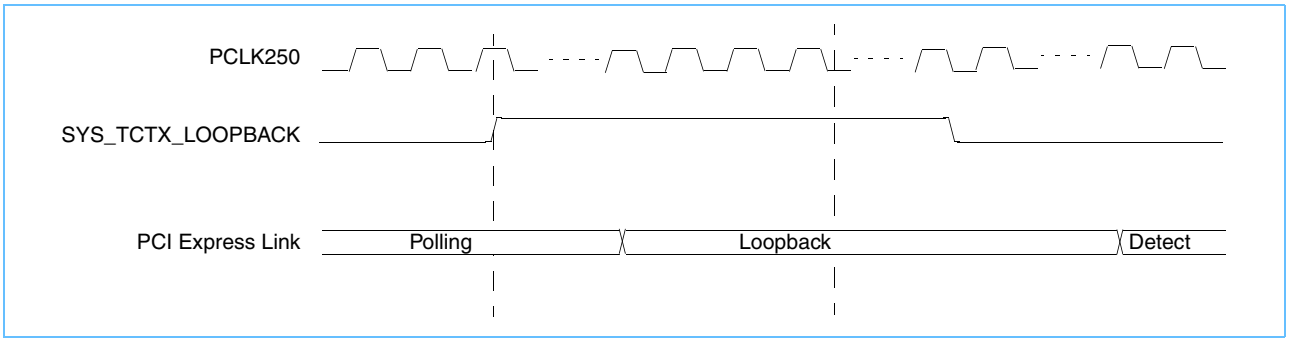
Figure 11. Training Control Reset (Hot Reset)



4.4.1.3 Training Control Loopback (Master)

Asserting SYS_TCTX_LOOPBACK causes the port (both upstream and downstream facing configurations) to enter the loopback master mode if at least one lane has detected a receiver when a link is being negotiated. It is recommended that SYS_TCTX_LOOPBACK be asserted before SYS_RESETN is deasserted. SYS_TCTX_LOOPBACK can be asserted after SYS_RESETN is deasserted, but it is not possible to predict when loopback will be entered and undesirable system effects could occur. For example, if a link has been established and transactions are in-progress when SYS_TCTX_LOOPBACK is asserted, loopback mode will be entered (a link down condition) and the transactions will not be completed. The link remains in loopback mode until SYS_TCTX_LOOPBACK is deasserted at which time it is renegotiated. See *Section 4.11, Loopback Interface, on page 68* for more information.

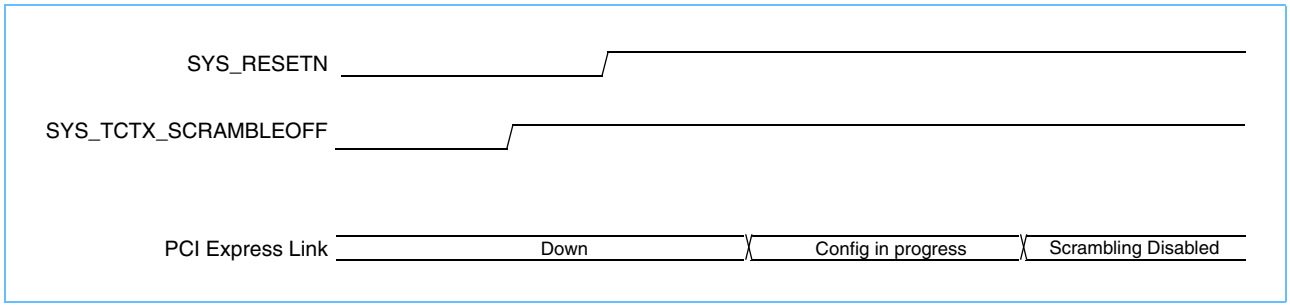
Figure 12. Training Control Loopback (Master)



4.4.1.4 Training Control Disable Scrambling

Asserting SYS_TCTX_SCRAMBLEOFF causes either a downstream- or upstream-facing port to disable scrambling on the link during link configuration as long as SYS_TCTX_RESET, SYS_TCTX_DISABLE, and SYS_TCTX_LOOPBACK are deasserted. It has no effect once a link has been configured. To ensure that scrambling will be turned off, it is recommended that it be asserted before SYS_RESETN is deasserted when performing a full reset, or before DL_HOTRESET is deasserted when configured as an upstream-facing port and a Hot Reset is being received.

Figure 13. Disabling Scrambling During Reset



4.5 TLP Transmit Interface

The TLP transmit interface is used to transmit TLPs on the link.

Note: Some of this interface's signals are affected by the Physical Link Width configuration option. See *Section 5.1.1 Physical Link Width* on page 71 for more information.

The core assigns each TLP a sequence number and while transmitting it also saves it in the replay buffer. When a TLP is acknowledged by the remote component, it is removed from the replay buffer. When the core receives a negative acknowledgement, all TLPs in the replay buffer are re-transmitted as soon as any in-progress TLP or DLLP has been transmitted.

The core can store up to eight unacknowledged TLPs in the replay buffer at any one time.

If the link goes down (indicated by the deassertion of DL_LINKUP), all TLPs which were accepted for transmission by the core but not acknowledged by the remote component (i.e. all TLPs in the replay buffer), are considered lost.

All input signals must be synchronous to the rising edge of PCLK250. All output signals are synchronous to the rising edge of PCLK250. All signals that are defined to have an asserted state are considered positive-active; meaning that their asserted level is a logical '1'.

Figure 14. TLP Transmit interface Signal Diagram

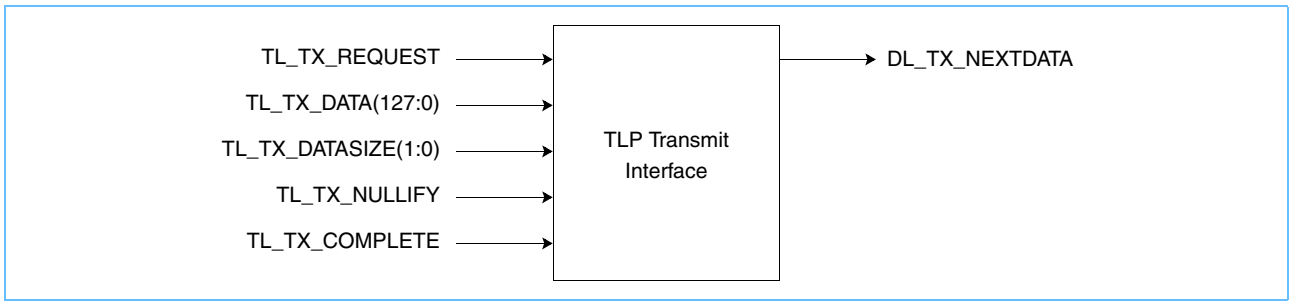


Table 8. TLP Transmit Interface Signal Descriptions

Signal Name	I/O Type	Timing	Description
TL_TX_REQUEST	Input	Begin	When asserted, indicates an active TLP transmit request. This signal is expected to be driven directly from a latch (which results in reduced latency through the core). TL_TX_REQUEST may be deasserted one cycle after DL_TX_NEXTDATA is initially asserted for the current TLP request. It may also be held asserted to indicate a new request, but such a new request is not recognized until one cycle after the current TLP transmission is completed (TL_TX_COMPLETE asserted and DL_TX_NEXTDATA asserted).
TL_TX_DATA(127:0) TL_TX_DATA(63:0) TL_TX_DATA(31:0)	Input	Middle	Data for the TLP. The least significant byte (bits (7:0)) is transmitted first in the lowest numbered lane of the link and the most significant byte (bits (127:120)) is transmitted last. The core does not distinguish between TLP header and data information, so care must be taken when supplying data on this bus to ensure correct transmission order over the link. See *Section 4.5.3, TL_TX_DATA Mapping to PCI Express Link Lanes, on page 36* for more information. **Note:** The width of this signal depends on the configuration of the core. It is (31:0) when the core is configured for a x1 or x4 Physical Link Width, (63:0) when configured for a x8 Physical Link Width, and (127:0) when configured for a x16 Physical Link Width.
TL_TX_DATASIZE(1:0) TL_TX_DATASIZE (Does not exist for x1 or x4)	Input	Middle	This signal is used to indicate how many of the 32-bit double-words of the TL_TX_DATA data bus have valid data at the end of the packet. **Note:** The width and existence of this signal depends on the configuration of the core. It does not exist when the core is configured for a x1 or x4 Physical Link Width, it is 1 bit when configured for a x8 Physical Link Width, and (1:0) when configured for a x16 Physical Link Width. The encodings are as follows when configured for a x8 Physical Link Width: ‘0’ indicates that two double-words are valid. ‘1’ indicates that only the double-word in bits (31:0) is valid. The encodings are as follows when configured for a x16 Physical Link Width: ‘00’ indicates that four double-words are valid. ‘01’ indicates that only the double-word in bits (31:0) is valid. ‘10’ indicates that the two double-words in bits (63:0) are valid. ‘11’ indicates that the three double-words in bits (95:0) are valid.
TL_TX_NULLIFY	Input	Middle	This signal may be asserted when TL_TX_COMPLETE is asserted to cause the TLP to be nullified.
TL_TX_COMPLETE	Input	Middle	This signal must be asserted when the data driven on the TL_TX_DATA bus is the last data for the TLP being transmitted.
DL_TX_NEXTDATA	Output	Middle	When asserted, indicates that the core has captured the data on the TL_TX_DATA bus and new data must be presented the following cycle. If asserted when TL_TX_COMPLETE is asserted, it indicates that the TLP transmission has been completed and the core is ready to transmit another TLP. Depending on the negotiated link width and the current traffic flow, DL_TX_NEXTDATA may be asserted in the same cycle that TL_TX_REQUEST is asserted or at any time afterward.

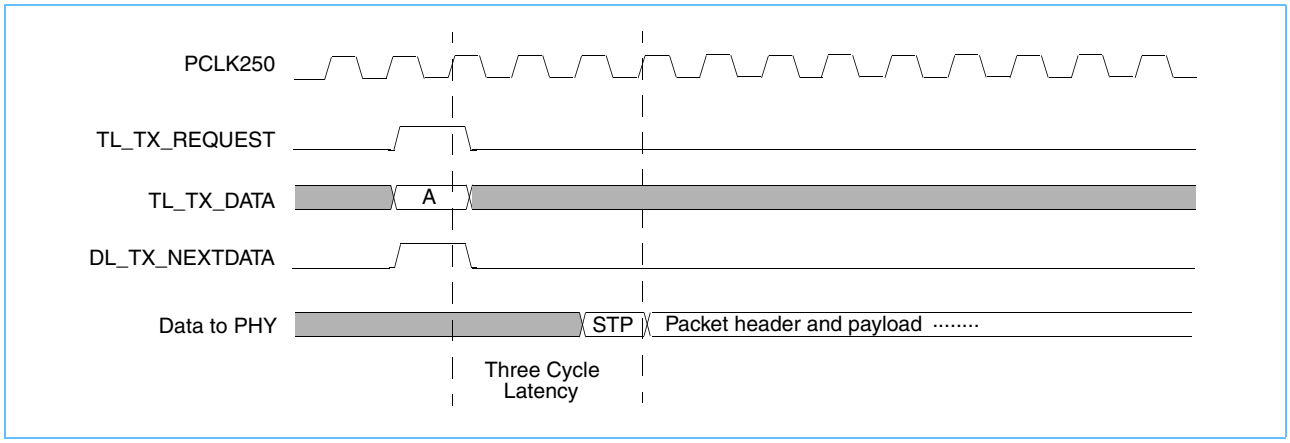
4.5.1 TLP Transmit Latency

The best-case TLP transmit latency occurs under the following conditions:

- The link is currently in the L0 link state
- No DLLP transmit requests are currently pending
- No skip-ordered sets are currently pending

If these conditions are true, then three PCLK250 clock cycles will elapse from the time TL_TX_REQUEST is asserted to when the STP symbol of the framed TLP is driven on the PHY interface. *Figure 15* shows an example timing diagram. Note that this latency is the same for all negotiated link widths.

Figure 15. Timing Diagram Showing TLP Transmit Latency



4.5.2 TLP Transmit Interface Protocol

Transmitting a TLP proceeds according to the following steps. See *Figure 16 on page 34*, *Figure 17 on page 35*, and *Figure 18 on page 35* for example timing diagrams showing operation of this interface when various link widths have been negotiated.

1. A new TLP transmit request is signalled by asserting TL_TX_REQUEST.
   - TL_TX_DATA bus must be driven with the first data of the TLP.
   - If the TLP is three double-words in length, TL_TX_COMPLETE must be asserted and TL_TX_DATASIZE must be driven to ‘11’ at the same time TL_TX_REQUEST is asserted.
   - If the TLP is four double-words in length, TL_TX_COMPLETE must be asserted and TL_TX_DATASIZE must be driven to ‘00’ at the same time TL_TX_REQUEST is asserted.
   - TLPs of one or two double-words are illegal, and if such a transfer is requested by driving TL_TX_DATASIZE to ‘01’ or ‘10’ and asserting TL_TX_COMPLETE with TL_TX_REQUEST, the request will be treated as a three double-word request.
   - When the TLP transmit request is the highest priority (which could be the same cycle TL_TX_REQUEST is asserted or an undefined amount of time later), the core initiates a TLP transmit.
2. When the data currently driven on the TL_TX_DATA bus has been stored by the core, DL_TX_NEXTDATA is asserted.
   - TL_TX_DATA must be updated with the next data to be transmitted one cycle after DL_TX_NEXTDATA is asserted.

- The core continues to transmit data and assert DL_TX_NEXTDATA until TL_TX_COMPLETE is asserted.
- Note from the example timing diagrams that the number of cycles between assertions of DL_TX_NEXTDATA depends on the negotiated link width. It is asserted in consecutive cycles when the negotiated link width is x16. When the negotiated link size is smaller than x16, DL_TX_NEXTDATA is asserted less frequently: every two cycles in the case of a x8 link, every 4 cycles in the case of a x4 link, and every 16 cycles in the case of a x1 link. Transaction layer logic should not assume any specific interval between assertions of DL_TX_NEXTDATA.
- After the first assertion of DL_TX_NEXTDATA for a TLP request, TL_TX_REQUEST may be deasserted without affecting the TLP in-progress. TL_TX_REQUEST is ignored until one cycle after DL_TX_NEXTDATA and TL_TX_COMPLETE are asserted.

3. When the last data of the TLP is presented on the TL_TX_DATA bus, TL_TX_COMPLETE must be asserted.

- At this time, TL_TX_DATASIZE must be driven to a value indicating the final number of double-words to be transmitted. TL_TX_DATASIZE must not change while TL_TX_COMPLETE is asserted. TL_TX_DATASIZE only needs to be valid while TL_TX_COMPLETE is asserted.
- Also at this time, TL_TX_NULLIFY may be asserted to indicate that the TLP is to be nullified. TL_TX_NULLIFY must not change while TL_TX_COMPLETE is asserted. TL_TX_NULLIFY only needs to be valid while TL_TX_COMPLETE is asserted.

Figure 16. Timing Diagram for TLP Transmit – Negotiated Link Size of x16

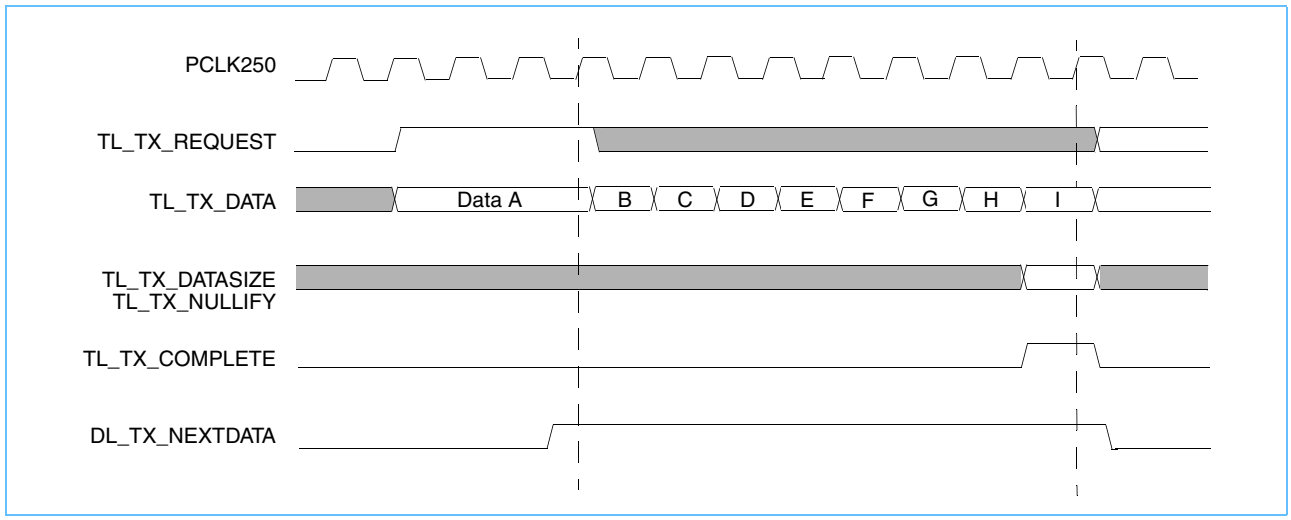


Figure 17. Timing Diagram for TLP Transmit – Negotiated Link Sized of x8

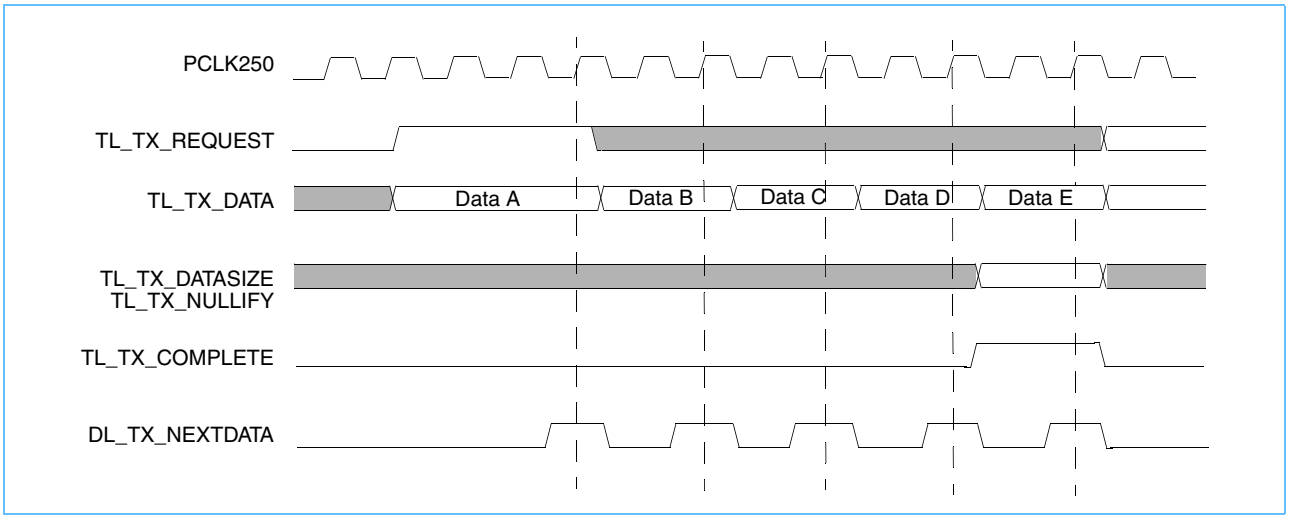


Figure 18. Timing Diagram for TLP Transmit – Negotiated Link Size of x4

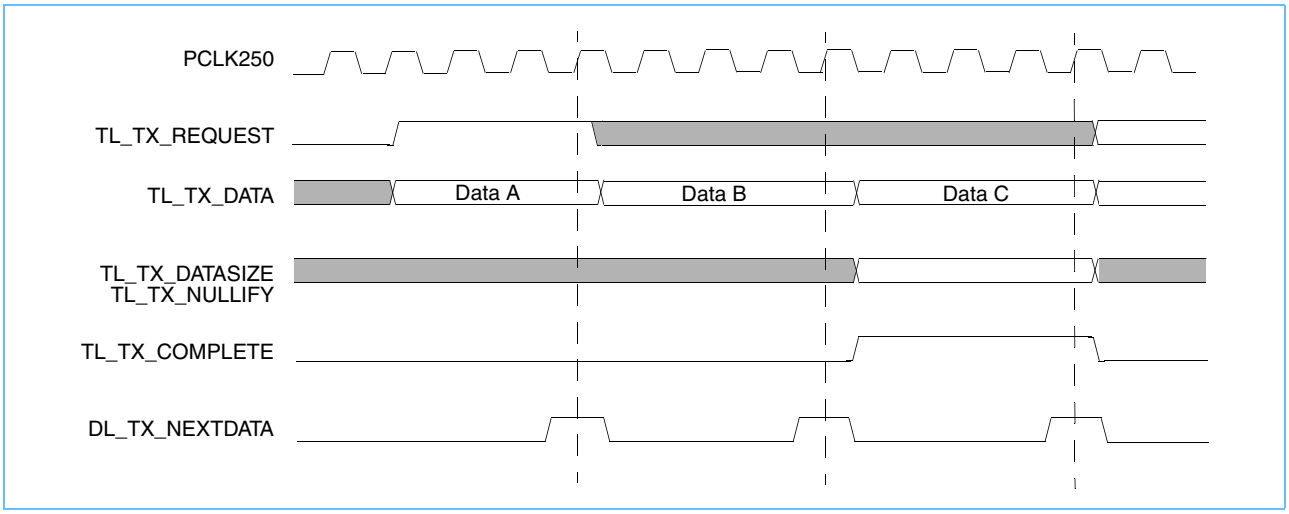
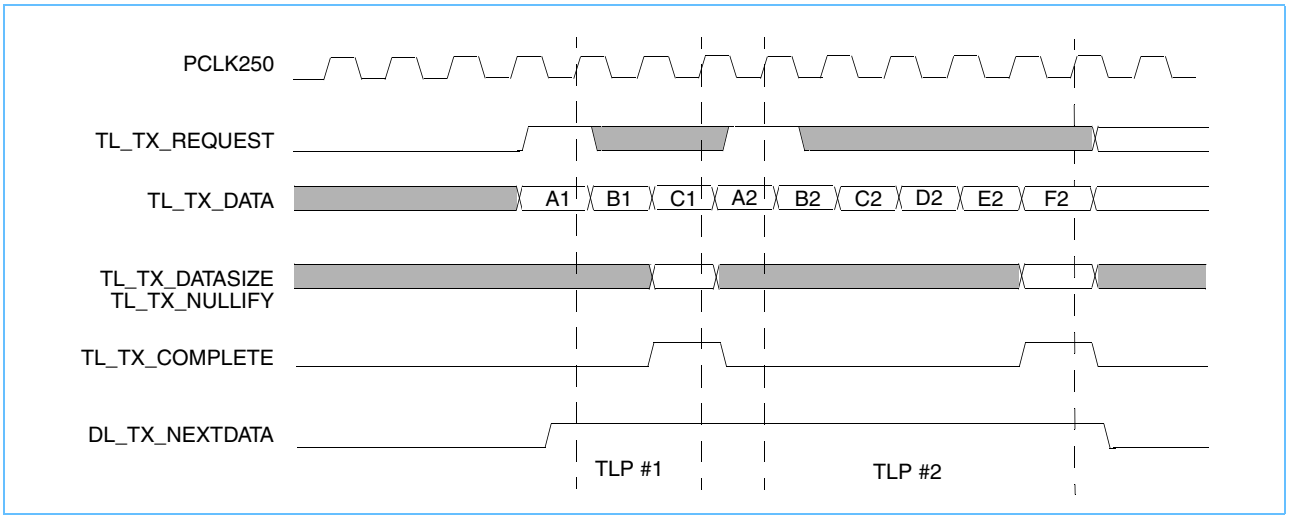


Figure 19 shows the maximum rate of TLP transmission using this interface. The interval between two consecutive TLP transmit requests may be zero cycles (as shown in *Figure 19 on page 36*), but may be extended by many factors including, but not limited to:

- The negotiated link width
- The number of TLPs currently outstanding on the link
- The priority of any DLLPs currently scheduled for transmission
- Skip ordered set insertion

Figure 19. Timing Diagram for Back-to-Back TLP Transmit – Negotiated Link Size of x16

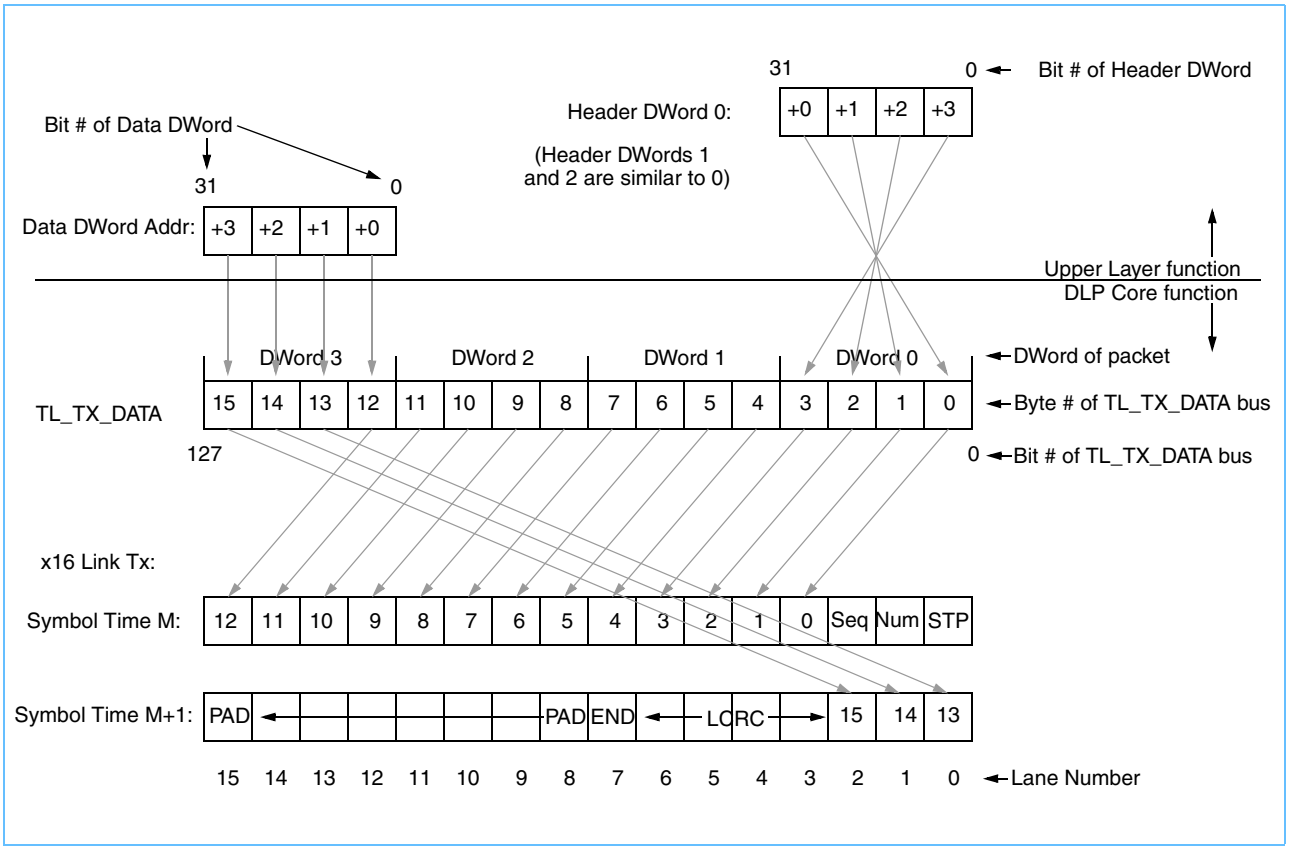


4.5.3 TL_TX_DATA Mapping to PCI Express Link Lanes

The data presented to the core on the TL_TX_DATA bus is mapped to the PCI Express link such that the least significant byte (bits (7:0)) is transmitted first in the lowest numbered lane of the link and the most significant byte (bits (127:120)) is transmitted last. An example mapping of a TLP to a x16 link is shown graphically in *Figure 20 on page 37*. In the example, the TLP consists of a 3 DWord header and a 1 DWord data payload.

Note that byte +0 of header DWord 0 must be transmitted first, and so the header DWords are byte-reversed when they are mapped to the TL_TX_DATA bus (see section 2.1.2 Packet Format Overview in the *PCI Express Base Specification* for more information). Also note that the lowest addressed byte of the data payload must be transmitted first (see section 2.2.2 TLPs with Data Payloads - Rules in the *PCI Express Base Specification* for more information). Data DWords may or may not need to be byte reversed depending on the source of the data and the endianness of the system. The figure shows an example of how a little endian based system might operate - the data DWord is NOT byte-reversed.

Figure 20. Example Mapping of a 4 DWord TLP on TL_TX_DATA to a x16 Link



4.5.4 Transmit TLP Length Considerations

As described in *Section 4.5.2, TLP Transmit Interface Protocol*, TLPs must be a minimum of 3 DWords in length and the interface protocol ensures this.

Maximum TLP length, however, is not ensured by the interface protocol. Instead, the transaction layer logic must ensure that TLPs presented to this core for transmission must not exceed the number of 16-byte transfers on the TL_TX_DATA bus required to transfer a TLP that contains a 4 DWord header, a payload of the configured maximum payload size supported (see *Section 5.1.2, Replay Buffer Size* for more information), and a TLP digest field. For example, suppose the configured maximum payload size supported is 256 bytes. In this case, a 4 DWord header requires 1 transfer on the TL_TX_DATA bus (16 bytes), a maximum size payload requires 16 transfers (256 bytes), and the digest requires 1 additional transfer since the header and payload are both multiples of 16 bytes, for a total of 18 transfers.

If the transaction layer exceeds the maximum number of transfers for a TLP, the behavior of this core is undefined and a reset may be required to return to normal operation. The core does, however, allow for one transfer over the maximum as long as the TLP is nullified.

4.6 TLP Receive Interface

The TLP receive interface is used to transfer received TLPs to the transaction layer logic.

Note: Some of this interface's signals are affected by the Physical Link Width configuration option. See *Section 5.1.1 Physical Link Width* on page 71 for more information.

All output signals are synchronous to the rising edge of PCLK250. All signals that are defined to have an asserted state are considered positive-active; meaning that their asserted level is a logical '1'.

Figure 21. TLP Receive Interface Signal Diagram

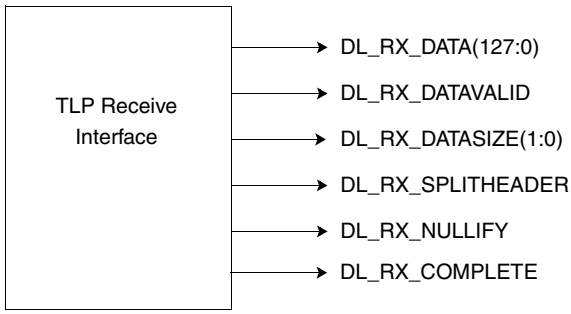


Table 9. TLP Receive Interface Signal Descriptions (Page 1 of 2)

Signal Name	I/O Type	Timing	Description
DL_RX_DATA(127:0) DL_RX_DATA(63:0) DL_RX_DATA(31:0)	Output	Early	TLP data. The least significant byte (bits (7:0)) contains the data byte which was received first (in the lowest numbered lane of the link) and the most significant byte (bits (127:120)) is the byte received last. The core does not distinguish between TLP header and data information, so care must be taken when decoding this data to ensure proper interpretation of the packet. **Note:** The width of this signal depends on the configuration of the core. It is (31:0) when the core is configured for a x1 or x4 Physical Link Width, (63:0) when configured for a x8 Physical Link Width, and (127:0) when configured for a x16 Physical Link Width.
DL_RX_DATAVALID	Output	Early	When asserted, this signal indicates that a TLP is being received and the DL_RX_DATA bus must be sampled by the transaction layer logic. The first assertion of this signal, after either a core reset or an assertion of DL_RX_COMPLETE, indicates the start of a TLP.

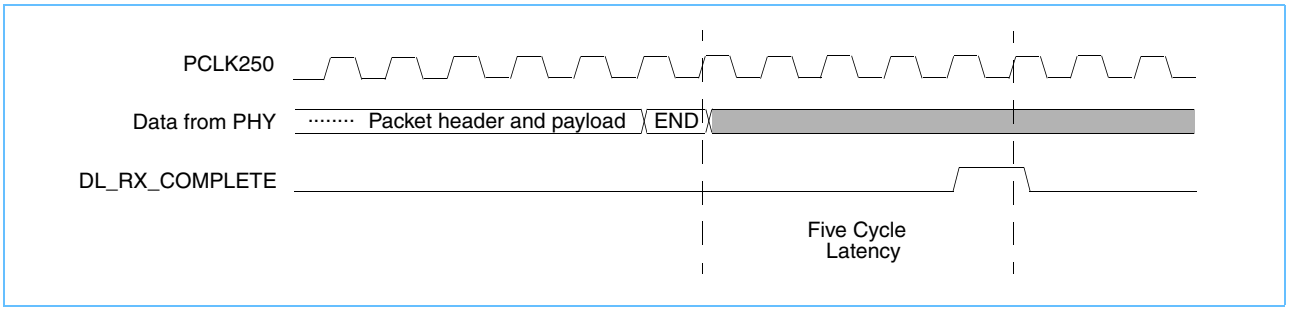
Table 9. TLP Receive Interface Signal Descriptions (Page 2 of 2)

Signal Name	I/O Type	Timing	Description
DL_RX_DATASIZE(1:0) DL_RX_DATASIZE (Does not exist for x1 or x4)	Output	Early	This signal is used to indicate how many of the 32-bit double-words of the data bus have valid data at the end of the packet. **Note:** The width and existence of this signal depends on the configuration of the core. It does not exist when the core is configured for a x1 or x4 Physical Link Width, it is 1 bit when configured for a x8 Physical Link Width, and (1:0) when configured for a x16 Physical Link Width. The encodings are as follows when configured for a x8 Physical Link Width: '0' indicates that two double-words are valid. '1' indicates that only the one double-word in bits (31:0) is valid. This signal is used only when both DL_RX_DATAVALID and DL_RX_COMPLETE are asserted. If only DL_RX_COMPLETE is asserted, all data for the packet was transferred in a previous cycle and the TLP was a multiple of two double-words in length. The encodings are as follows when configured for a x8 Physical Link Width: '00' indicates that four double-words are valid. '01' indicates that only the one double-word in bits (31:0) is valid. '10' indicates that the two double-words in bits (63:0) are valid. '11' indicates that the three double-words in bits (95:0) are valid. This signal is used only when both DL_RX_DATAVALID and DL_RX_COMPLETE are asserted. If only DL_RX_COMPLETE is asserted, all data for the packet was transferred in a previous cycle and the TLP was a multiple of four double-words in length. This signal will be '01' when DL_RX_SPLITHEADER is asserted.
DL_RX_SPLITHEADER	Output	Early	When asserted, this signal indicates that DL_RX_DATA contains the following information: the last double-word from the TLP in-progress on bits (31:0) and one double-word for the next TLP being received on bits (127:96). **Note:** This signal only exists when the core is configured for a x16 Physical Link Width.
DL_RX_NULLIFY	Output	Middle	When asserted with DL_RX_COMPLETE, this signal indicates that the TLP was nullified by the sender or an error occurred while receiving the TLP. All data for the TLP must be discarded. If this signal is asserted in the same cycle DL_RX_SPLITHEADER is asserted, it indicates that the TLP in-progress (not the new TLP) was nullified.
DL_RX_COMPLETE	Output	Early	This signal is asserted for one cycle when all data for the current TLP has been transferred. This signal may be asserted with DL_RX_DATAVALID, which indicates the last data for the TLP and completion of the TLP, or it may be asserted after DL_RX_DATAVALID is asserted indicating that all data was transferred previously. This signal is asserted for one cycle for each TLP completed, and may be asserted in consecutive cycles. This signal will be asserted if DL_RX_SPLITHEADER is asserted.

4.6.1 TLP Receive Latency

If the link is currently in the L0 link state, five PCLK250 clock cycles will elapse from the time the END or EDB symbol of the framed TLP is received from the PHY interface to when DL_RX_COMPLETE is asserted on the TLP Receive interface. *Figure 22* on page 40 shows an example timing diagram. Note that this latency is the same for all negotiated link widths.

Figure 22. Timing Diagram Showing TLP Receive Latency



4.6.2 TLP Receive Interface Protocol

The TLP receive interface is active only when DL_LINKUP is asserted. See *Figure 23 on page 41*, *Figure 24 on page 41*, *Figure 25 on page 41*, *Figure 26 on page 42*, and *Figure 27 on page 42* for example timing diagrams showing the operation of this interface when various link widths have been negotiated.

Operation of this interface is as follows:

1. The core drives DL_RX_DATA and asserts DL_RX_DATAVALID for one clock cycle to indicate data was received by the link.
2. The end of a TLP and the last data for the TLP may be signalled in three different ways depending on the negotiated link size and the alignment of the traffic on the link.
   - Scenario A: The core drives DL_RX_DATA and asserts DL_RX_DATAVALID, and at the same time asserts DL_RX_COMPLETE and drives DL_RX_DATASIZE to indicate the number of double-words valid at the end of the TLP. An example of this scenario is shown in *Figure 23 on page 41*.
   - Scenario B: The core drives DL_RX_DATA and asserts DL_RX_DATAVALID. Then, at least one cycle later, it asserts DL_RX_COMPLETE *without* asserting DL_RX_DATAVALID. This indicates that the TLP's length was a multiple of four double-words and that all data was transferred with the previous assertion of DL_RX_DATAVALID. No data is transferred in this case and DL_RX_DATASIZE must be ignored. Examples of this scenario are shown in *Figure 24 on page 41*, *Figure 25 on page 41*, and *Figure 26 on page 42*.
   - Scenario C: The core drives DL_RX_DATA, asserts DL_RX_DATAVALID, DL_RX_SPLITHEADER, DL_RX_COMPLETE, and drives DL_RX_DATASIZE to '01'. This indicates the completion of one TLP and the start of another TLP. Bits (31:0) of DL_RX_DATA are the last double-word of the current TLP and bits (127:96) of DL_RX_DATA are the first double-word of the next TLP. An example of this scenario is shown in *Figure 27 on page 42*.
   - Note that DL_RX_COMPLETE may be asserted the first time DL_RX_DATAVALID is asserted for a packet. This can occur when the TLP is four double-words or less in length. One and two double-word TLPs are considered malformed according to the *PCI Express Base Specification*: they may be passed to the transaction layer, so transaction layer logic should check for this condition.
   - Note that DL_RX_NULLIFY may be asserted when DL_RX_COMPLETE is asserted (in any of the three scenarios described above) to indicate that the packet should be discarded with no further action.

Figure 23. Timing Diagram for TLP Receive – Negotiated Link Size of x16 and DL_RX_COMPLETE Asserted with DL_RX_DATAVALID

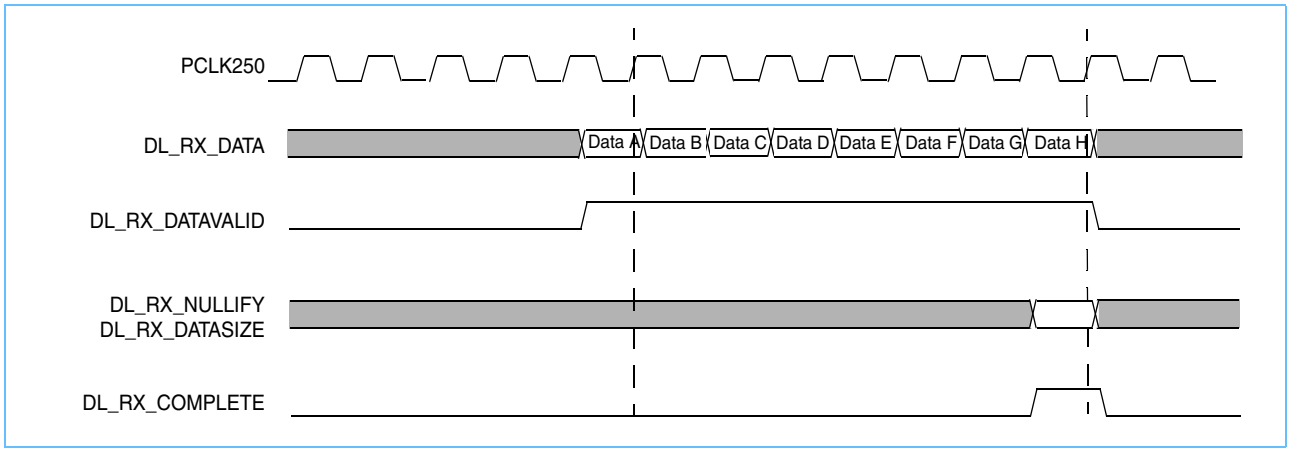


Figure 24. Timing Diagram for TLP Receive – Negotiated Link Size of x16 and DL_RX_COMPLETE Asserted after DL_RX_DATAVALID

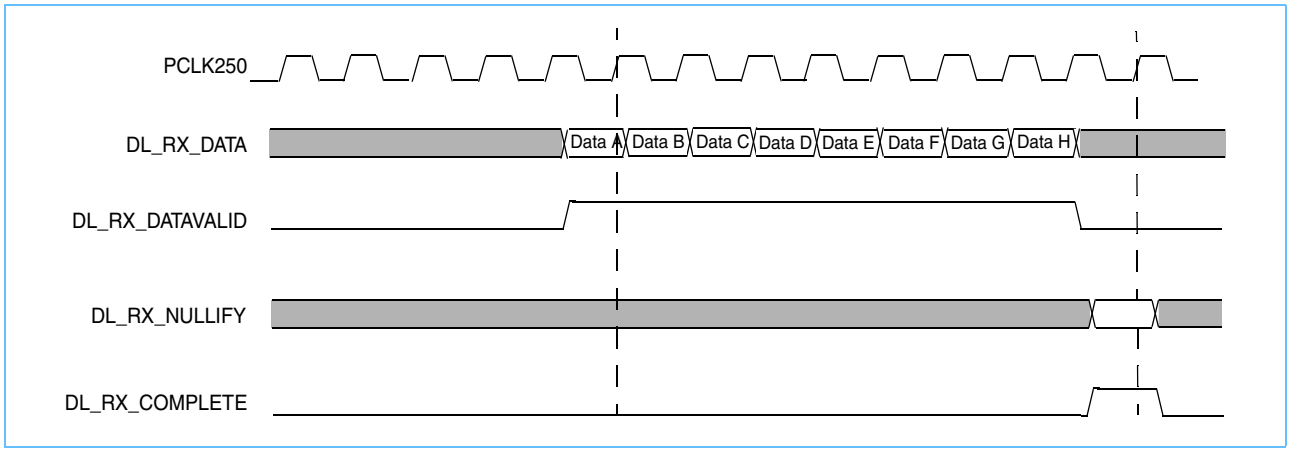


Figure 25. Timing Diagram for TLP Receive – Negotiated Link Size of x8 and DL_RX_COMPLETE Asserted after DL_RX_DATAVALID

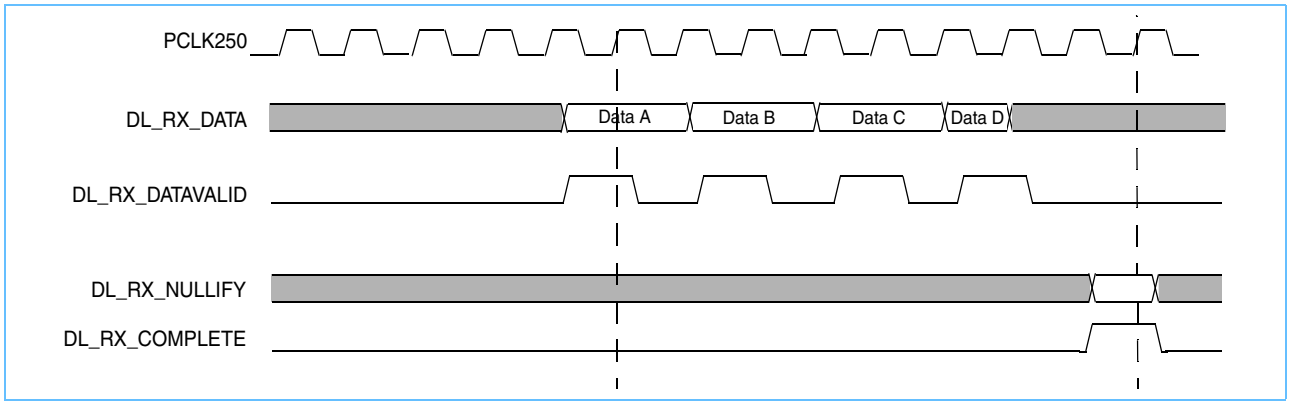


Figure 26. Timing Diagram for TLP Receive – Negotiated Link Size of x4 and DL_RX_COMPLETE Asserted after DL_RX_DATAVALID

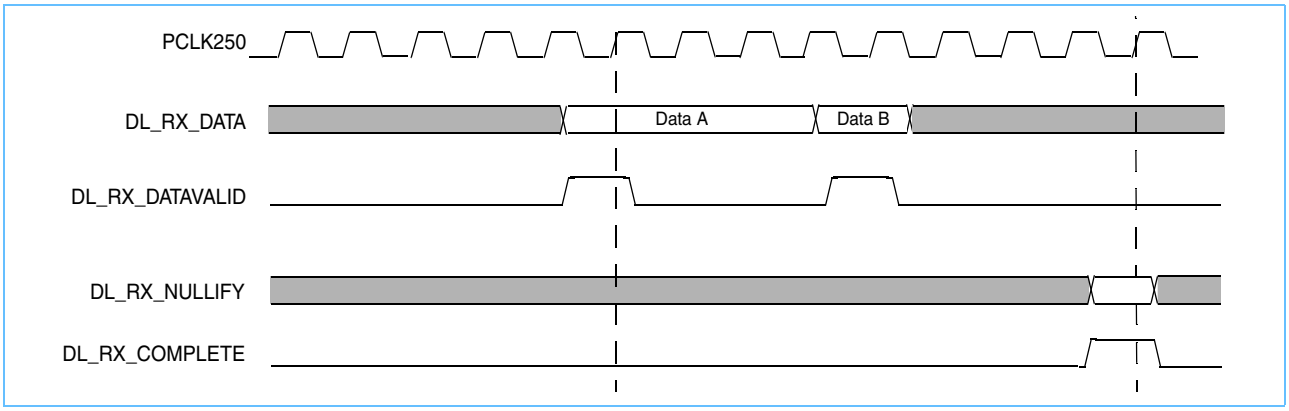
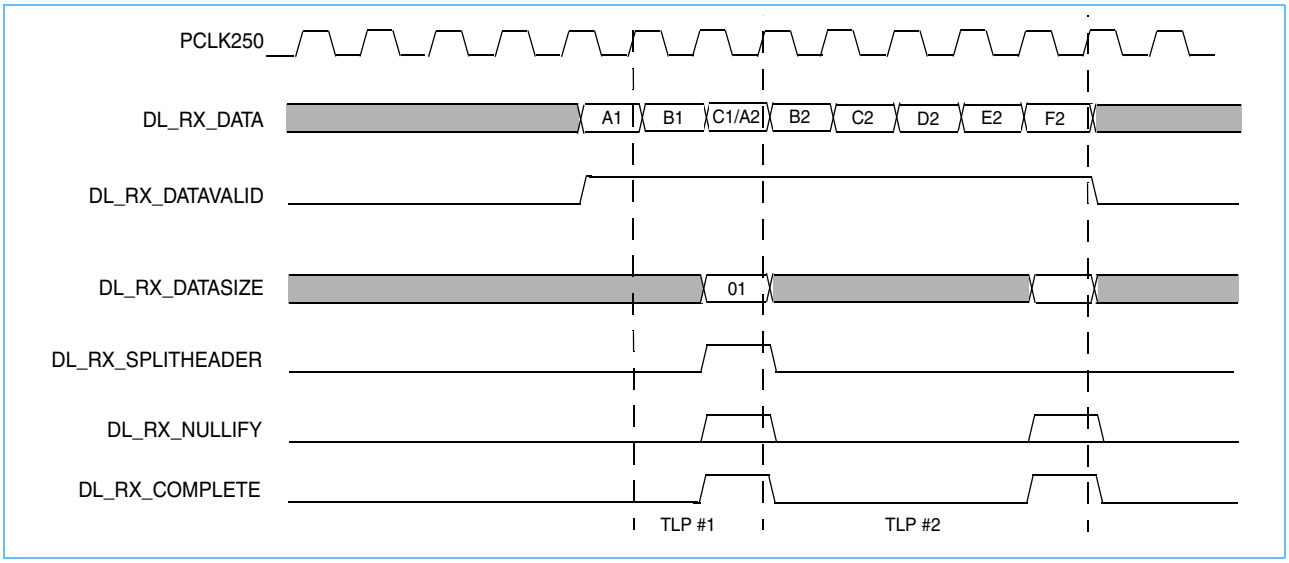


Figure 27. Timing Diagram for a Split Header TLP Receive – Negotiated Link Size of x16

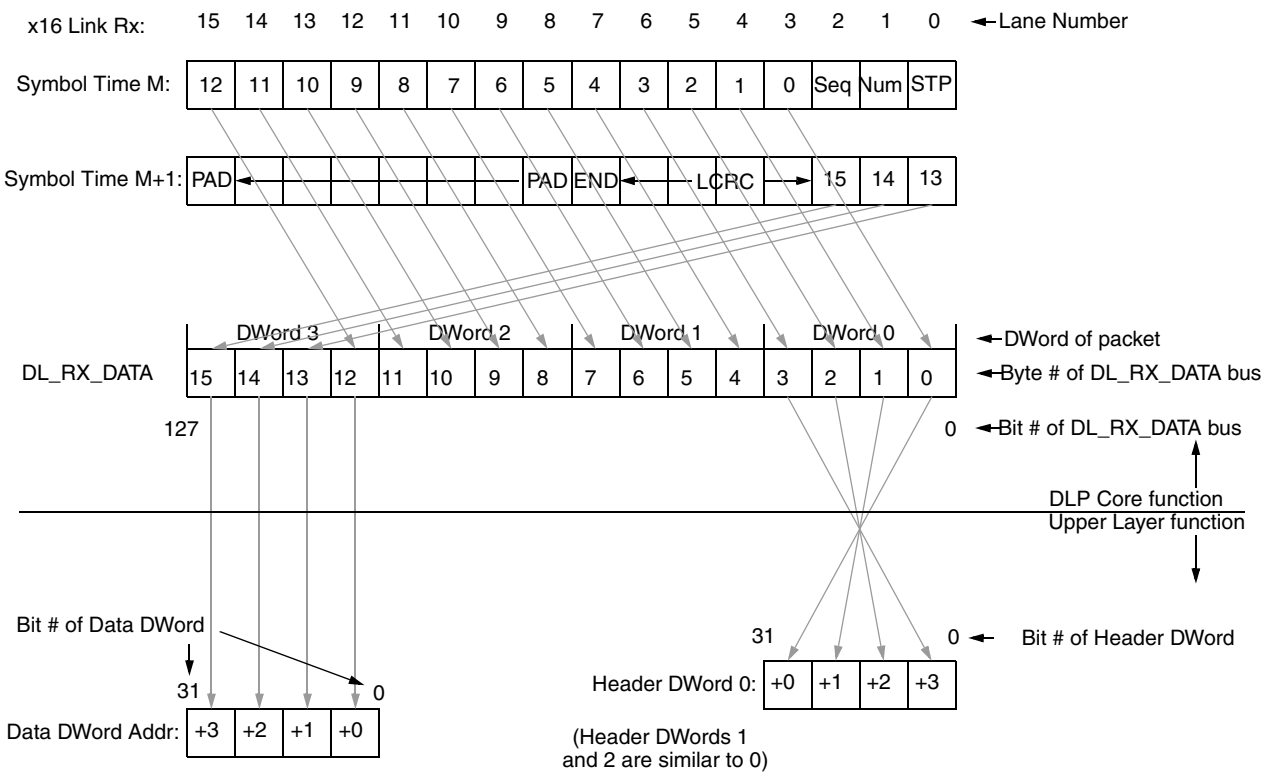


4.6.3 PCI Express Link Lane Mapping to DL_RX_DATA

The data received by the core on the PCI Express link is mapped to the DL_RX_DATA bus such that the byte received on the lowest numbered lane of the link is mapped to the least significant byte (bits (7:0)). This is exactly the reverse of how the core transmits data, see *Section 4.5.3, TL_TX_DATA Mapping to PCI Express Link Lanes, on page 36* for more information. An example mapping of a TLP received on a x16 link is shown graphically in *Figure 28 on page 43*. In the example, the TLP consists of a 3 DWord header and a 1 DWord data payload.

Note that byte +0 of header DWord 0 is received first, and so bits (31:0) of DL_RX_DATA are byte-reversed header DWord 0. The data payload is mapped in the same manner, however, the data may or may not need to be byte-reversed depending on the source of the data and the endianness of the system.

Figure 28. Example Mapping of a 4 DWord TLP Received on a x16 Link to DL_RX_DATA



4.7 Flow Control Interface

The flow control interface is used to transmit flow control information for the local component and to receive flow control information from the remote component. The interface consists of three sub-interfaces: initialization transmit, update transmit, and flow control receive.

All input signals must be synchronous to the rising edge of PCLK250. All output signals are synchronous to the rising edge of PCLK250. All signals which are defined to have an asserted state are considered positive-active; meaning that their asserted level is a logical ‘1’.

Figure 29. Flow Control Interface Signal Diagram

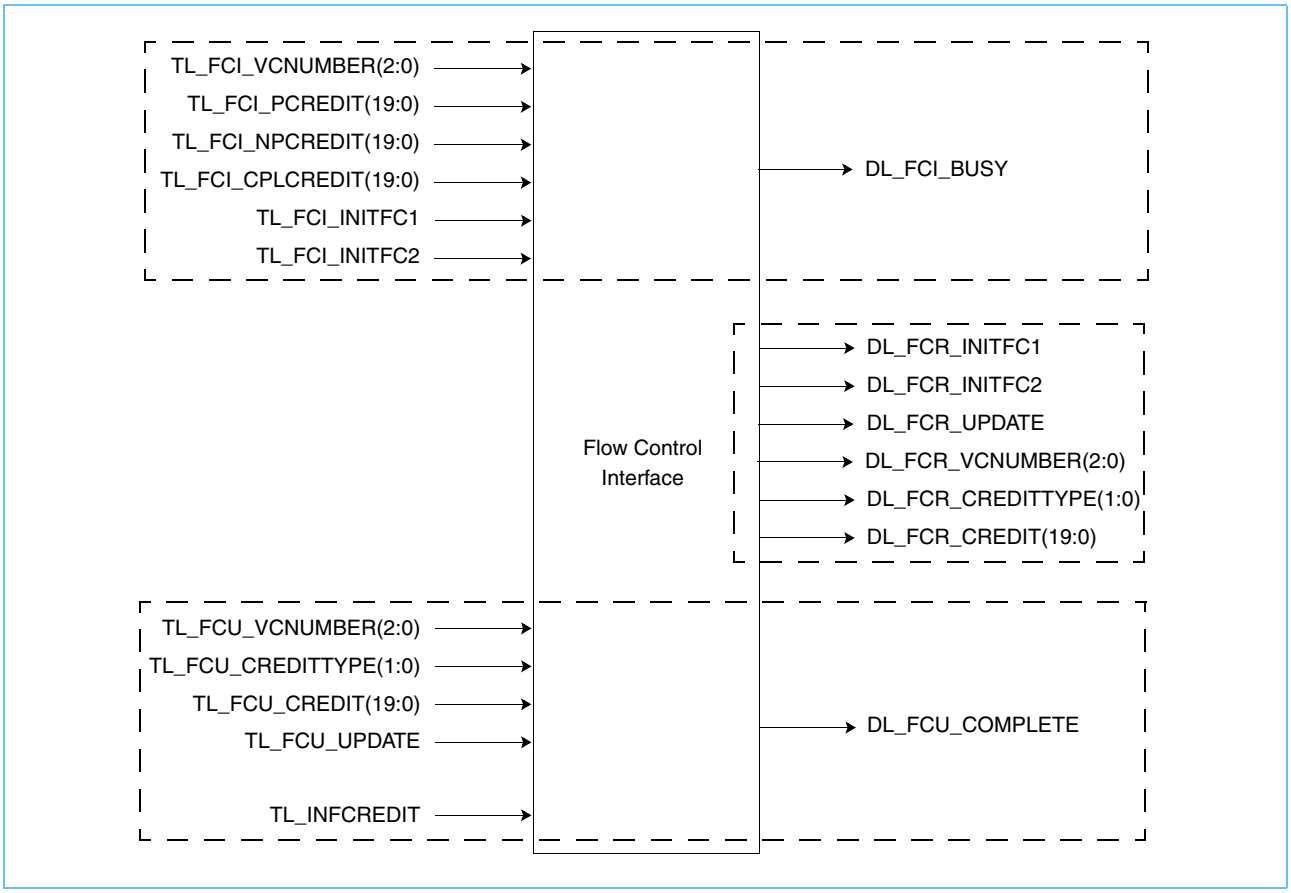


Table 10. Flow Control Interface Signal Descriptions (Page 1 of 2)

Signal Name	I/O Type	Timing	Description
TL_FCI_VCNUMBER(2:0)	Input	Middle	Used for flow control initialization. The number of the VC to be initialized.
TL_FCI_PCREDIT(19:0)	Input	Middle	Used for flow control initialization. Posted request credit information for the VC to be initialized. Bits (11:0) are the data credit information. Bits (19:12) are the header credit information.
TL_FCI_NPCREDIT(19:0)	Input	Middle	Used for flow control initialization. Non-posted request credit information for the VC to be initialized. Bits (11:0) are the data credit information. Bits (19:12) are the header credit information.
TL_FCI_CPLCREDIT(19:0)	Input	Middle	Used for flow control initialization. Completion credit information for the VC to be initialized. Bits (11:0) are the data credit information. Bits (19:12) are the header credit information.
TL_FCI_INITFC1	Input	Middle	Used for flow control initialization. When asserted, the TL_FCI_INITFC1 signal causes the DL to send InitFC1 DLLPs for the VC specified by TL_FCI_VCNUMBER. When de-asserted, this signal must not be asserted when DL_FCI_BUSY is asserted.
TL_FCI_INITFC2	Input	Middle	Used for flow control initialization. When asserted, the TL_FCI__INITFC2 signal causes the DL to send InitFC2 DLLPs for the VC specified by TL_FCI_VCNUMBER. When de-asserted, this signal must not be asserted when DL_FCI_BUSY is asserted.

Table 10. Flow Control Interface Signal Descriptions (Page 2 of 2)

Signal Name	I/O Type	Timing	Description
DL_FCI_BUSY	Output	Begin	Used for flow control initialization. Asserted when InitFC1 or InitFC2 DLLPs are being transmitted.
TL_FCU_VCNUMBER(2:0)	Input	Middle	Used to transmit flow control updates. The number of the VC to be updated.
TL_FCU_CREDITTYPE(1:0)	Input	Middle	Used to transmit flow control updates. The credit type of the update request. The encodings are: '00' = Posted '01' = Non-posted '10' = Completion The '11' type is reserved. If it is used, the request will be consumed by the core and no DLLP is transmitted.
TL_FCU_CREDIT(19:0)	Input	Middle	Used to transmit flow control updates. The credit information. Bits (11:0) are the data credit information. Bits (19:12) are the header credit information.
TL_FCU_UPDATE	Input	Middle	Used to transmit flow control updates. When asserted, the TL_FCU_UPDATE signal causes an UpdateFC DLLP to be transmitted for the VC specified by TL_FCU_VCNUMBER.
TL_INFCREDIT	Input	Middle	Used to indicate that the remote component has infinite credits available for all enabled VC types. Note: See *Section 4.7.1, Use of the TL_INFCREDIT Input, on page 46* for more information.
DL_FCU_COMPLETE	Output	Early	Used to transmit flow control updates. Asserted for one cycle when an UpdateFC DLLP for the current TL_FCU_UPDATE request has been queued for transmission.
DL_FCR_INITFC1	Output	Early	Used to receive flow control updates. When asserted, this signal indicates that an InitFC1 DLLP was received with the information described by the DL_FCR_VCNUMBER, DL_FCR_CREDITTYPE, and DL_FCR_CREDIT signals. This signal is asserted for one cycle for each DLLP of this type received. Only one of DL_FCR_INITFC1, DL_FCR_INITFC2, and DL_FCR_UPDATE is asserted in any one cycle, but any combination of the three may occur in back-to-back cycles.
DL_FCR_INITFC2	Output	Early	Used to receive flow control updates. When asserted, this signal indicates that an InitFC2 DLLP was received with the information described by the DL_FCR_VCNUMBER, DL_FCR_CREDITTYPE, and DL_FCR_CREDIT signals. This signal is asserted for one cycle for each DLLP of this type received.
DL_FCR_UPDATE	Output	Early	Used to receive flow control updates. When asserted, this signal indicates that an UpdateFC DLLP was received with the information described by the DL_FCR_VCNUMBER, DL_FCR_CREDITTYPE, and DL_FCR_CREDIT signals. This signal is asserted for one cycle for each DLLP of this type received.
DL_FCR_VCNUMBER(2:0)	Output	Early	Used to receive flow control updates. The VC number of the received flow control DLLP.
DL_FCR_CREDITTYPE(1:0)	Output	Early	Used to receive flow control updates. The credit type of the received DLLP. The encodings are: '00' = Posted '01' = Non-posted '10' = Completion An encoding of '11' will *not* be driven on this interface.
DL_FCR_CREDIT(19:0)	Output	Early	Used to receive flow control updates. Credit information. Bits (11:0) are the data credit information. Bits (19:12) are the header credit information.

4.7.1 Use of the TL_INFCREDIT Input

The *PCI Express Base Specification* defines a timer that tracks when the last flow control update was received from the remote component. Normally, flow control information is updated every 30 μs, so if the timer reaches 200 μs, it is assumed that flow control updates have been lost and the link is retrained. However, if a credit type has been initialized with infinite credit, then no flow control updates are required. TL_INFCREDIT should be asserted if all types for all enabled virtual channels have been initialized with infinite credit from the remote component to prevent the timer from timing out and retraining the link when no retrain is necessary.

4.7.2 Initializing Flow Control for a Virtual Channel

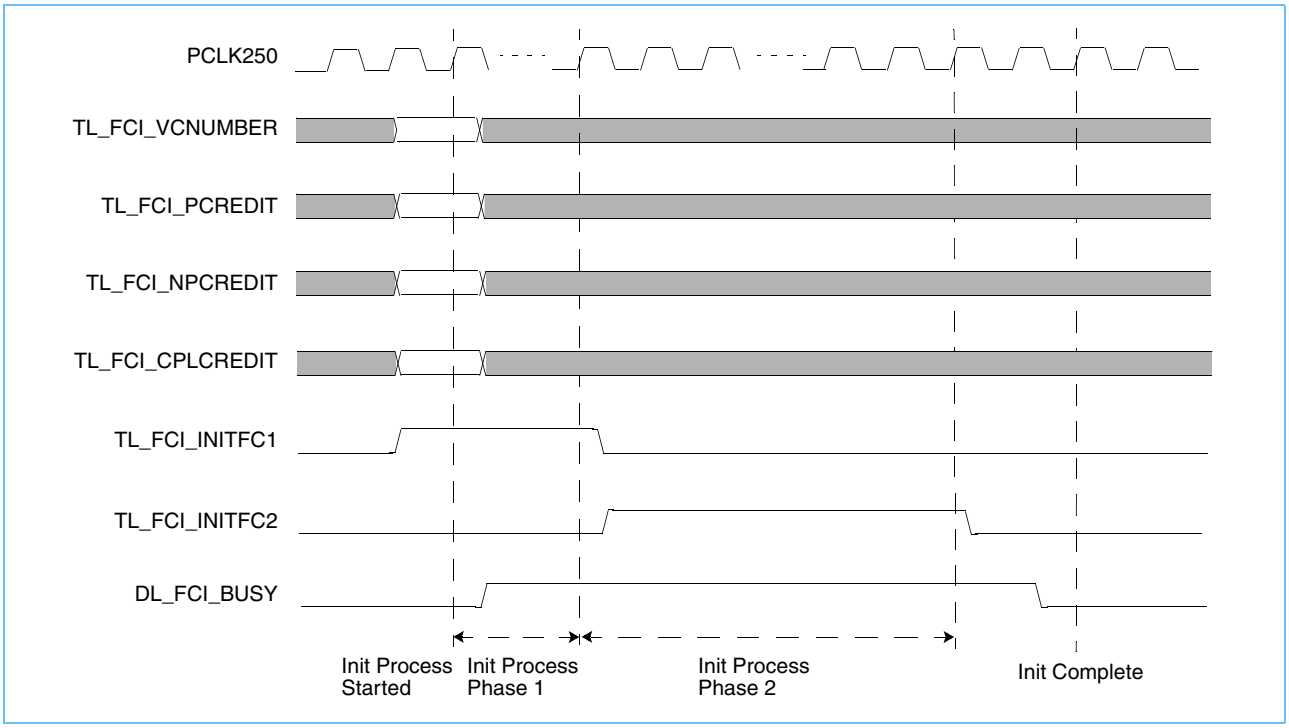
The initialization sub-interface may be used only when DL_LINKUP is asserted. If DL_LINKUP is deasserted during a flow control initialization sequence, TL_FCI_INITFC1, and TL_FCI_INITFC2 must be deasserted within one clock cycle and the initialization that was in progress is considered cancelled.

A flow control initialization proceeds as follows (see *Figure 30 on page 47*):

1. TL_FCI_VCNUMBER, TL_FCI_PCREDIT, TL_FCI_NPCREDIT, and TL_FCI_CPLCREDIT are driven to the desired values and TL_FCI_INITFC1 is asserted.
   - The transaction layer logic must ensure that DL_FCI_BUSY is deasserted before asserting TL_FCI_INITFC1.
   - The assertion of TL_FCI_INITFC1 causes the core to sample the VC number and credit information and begin to transmit InitFC1 DLLPs with the information supplied. The DLLPs are transmitted repeatedly and are guaranteed to be transmitted at an interval less than 34 μs. If the link is in the L0s power management state, it will be returned to the L0 state and then the DLLPs will be transmitted. If the link is in the L1 power management state, transaction layer logic must return the link to the L0 state in order for DLLPs to be transmitted.
   - Transaction layer logic must monitor the flow control receive interface to look for flow control initialization information from the remote component. TL_FCI_INITFC1 must be held asserted until Flag FI1 is set, as described in the *PCI Express Base Specification*.
   - The core will assert DL_FCI_BUSY one cycle after TL_FCI_INITFC1 is asserted to indicate to the transaction layer logic that the flow control initialization request is active. The VC number and credit information may be changed once DL_FCI_BUSY is asserted.
2. TL_FCI_INITFC1 is deasserted and TL_FCI_INITFC2 is asserted.
   - This causes the core to switch to transmitting InitFC2 DLLPs with the credit and VC information sampled in step 1. The DLLPs are transmitted repeatedly and are guaranteed to be transmitted at an interval less than 34 μs. If the link is in the L0s power management state, it will be returned to the L0 state and then the DLLPs will be transmitted. If the link is in the L1 power management state, transaction layer logic must return the link to the L0 state in order for DLLPs to be transmitted.
   - Transaction layer logic must monitor the flow control receive interface to look for flow control initialization information from the remote component. TL_FCI_INITFC2 must be held asserted until Flag FI2 is set, as described in the *PCI Express Base Specification*.
3. TL_FCI_INITFC2 is deasserted.
   - The core will complete the current DLLP transmission (either InitFC1 or InitFC2) and deassert DL_FCI_BUSY. There may be a significant delay between the deassertion of TL_FCI_INITFC2 and the deassertion of DL_FCI_BUSY.

- The transaction layer logic must wait for DL_FCI_BUSY to be deasserted before attempting to initialize another virtual channel.

Figure 30. Timing Diagram for Flow Control Initialization



Note: When flow control initialization is in-progress for any virtual channel, software should not initiate any device power state changes in either component of the link that would initiate an L1 or L2/3 link condition. In addition, Active State Power Management L1 should not be enabled during flow control initialization. Low power link states will significantly increase the time required to initialize flow control.

4.7.3 Transmitting Flow Control Update Information

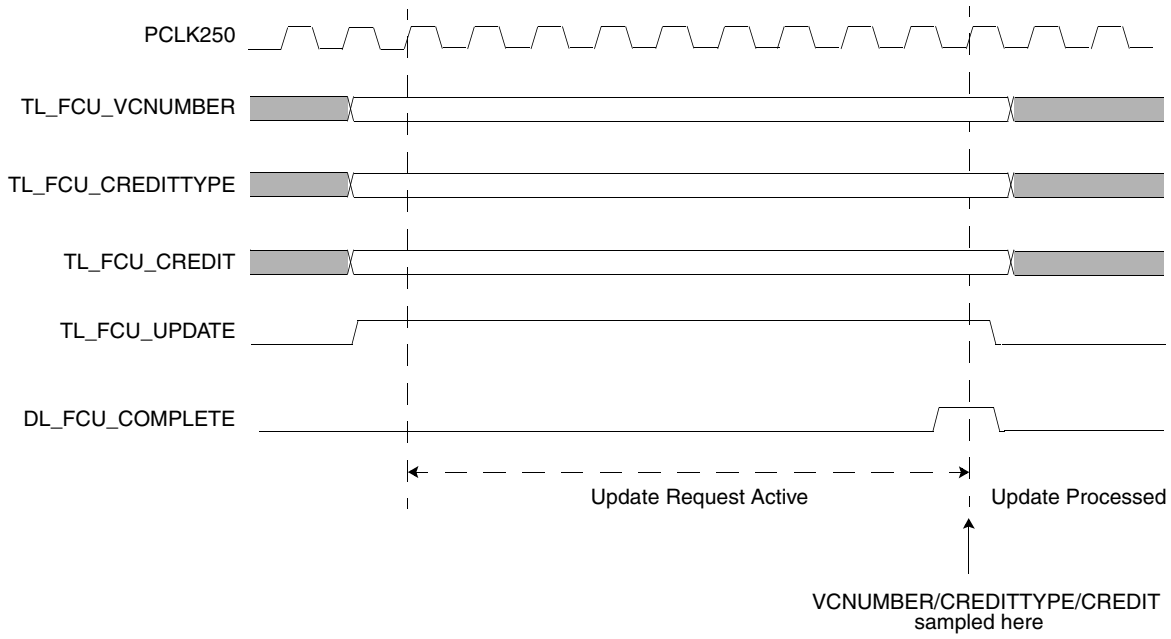
The update transmit sub-interface may be used only when DL_LINKUP is asserted. If DL_LINKUP is deasserted during a flow control update request, TL_FCU_UPDATE must be deasserted within one clock cycle and the initialization that was in progress is considered cancelled.

A flow control update proceeds as follows (refer to *Figure 31*):

1. TL_FCU_VCNUMBER, TL_FCU_CREDITTYPE, and TL_FCU_CREDIT are driven to the desired values and TL_FCU_UPDATE is asserted.
   - TL_FCU_VCNUMBER, TL_FCU_CREDITTYPE, and TL_FCU_CREDIT may change while TL_FCU_UPDATE is asserted.
   - TL_FCU_UPDATE must remain asserted until DL_FCU_COMPLETE is asserted.
2. Once the request has been recognized, the core asserts DL_FCU_COMPLETE for one cycle.
   - TL_FCU_VCNUMBER, TL_FCU_CREDITTYPE, and TL_FCU_CREDIT are sampled when DL_FCU_COMPLETE is asserted and those values are used to transmit the update to the remote component.

- If the transaction layer logic wants to drive another flow control update, TL_FCU_UPDATE may be held asserted and TL_FCU_VCNUMBER, TL_FCU_CREDITTYPE, and TL_FCU_CREDIT are driven to new values the cycle after DL_FCU_COMPLETE is asserted.
- If the transaction layer logic does not want to transmit another flow control update, TL_FCU_UPDATE should be deasserted in the cycle following the assertion of DL_FCU_COMPLETE.

Figure 31. Timing Diagram for Transmitted Flow Control Updates



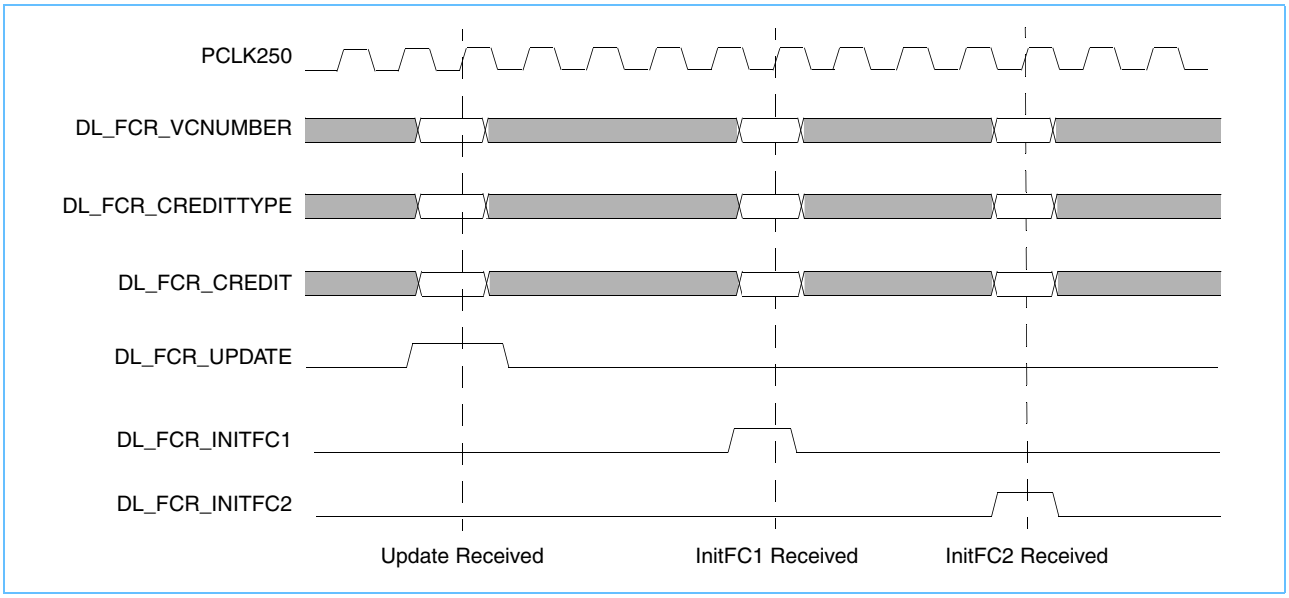
4.7.4 Receiving Flow Control Information

The receive sub-interface is only active when DL_LINKUP is asserted.

The reception of a valid flow control DLLP is signalled by driving DL_FCR_VCNUMBER, DL_FCR_CREDITTYPE, and DL_FCR_CREDIT to the received values and asserting one of DL_FCR_INTFC1, DL_FCR_INITFC2, or DL_FCR_UPDATE (refer to *Figure 32*). Only one of DL_FCR_INITFC1, DL_FCR_INITFC2, and DL_FCR_UPDATE will be asserted in any one cycle - they are mutually exclusive.

Note: Flow control receive information must be handled immediately by transaction layer logic. When a link width of x8 or x16 is negotiated, a flow control DLLP may be received by the core every symbol time, and if so, DL_FCR_INITFC1, DL_FCR_INITFC2, or DL_FCR_UPDATE may be asserted in consecutive cycles. For each cycle a signal is asserted, flow control information must be processed.

Figure 32. Timing Diagram for Received Flow Control



4.8 Power Management Interface

The power management interface is used to control and signal entry to and exit from the L0, Active State Power Management L1, PCI Power Management L1, and PCI Power Management L2/L3 Ready link states.

All input signals must be synchronous to the rising edge of PCLK250. All output signals are synchronous to the rising edge of PCLK250. All signals which are defined to have an asserted state are considered positive-active; meaning that their asserted level is a logical ‘1’.

Figure 33. Power Management Interface Signal Diagram

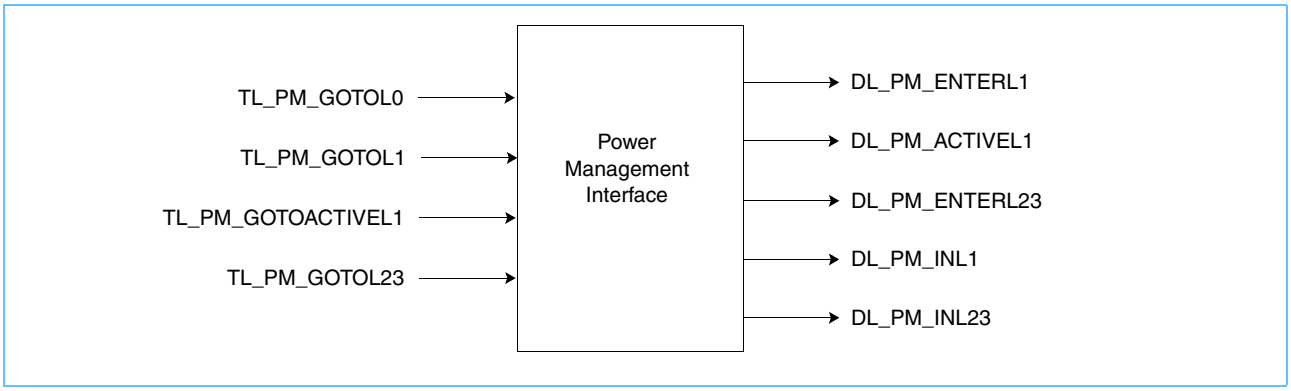


Table 11. Power Management Interface Signal Descriptions

Signal Name	I/O Type	Timing	Description
TL_PM_GOTOL0	Input	Middle	This signal should be asserted for one PCLK250 cycle to initiate the desired action. When asserted in an upstream facing port configuration, this signal cancels a pending active state L1 request or initiates an exit from the L1 state. When asserted in a downstream facing port configuration, this signal initiates an exit from the L1 state.
TL_PM_GOTOL1	Input	Middle	This signal should be asserted for one PCLK250 cycle to initiate the desired action. When asserted in an upstream facing port configuration, this signal initiates the L1 protocol. When asserted in a downstream facing port configuration, this signal acknowledges an L1 request. Note: The transaction layer must not attempt to transmit a TLP after asserting this signal and until the L1 state has been entered and subsequently exited.
TL_PM_GOTOACTIVEL1	Input	Middle	This signal should be asserted for one PCLK250 cycle to initiate the desired action. When asserted in an upstream facing port configuration, this signal initiates the active state L1 protocol. This signal is not used in a downstream facing port configuration and may be set to any value. Note: The transaction layer must not attempt to transmit a TLP after asserting this signal and until the L1 state has been entered and subsequently exited or until the active state L1 request is cancelled by an assertion of TL_PM_GOTOL0.
TL_PM_GOTOL23	Input	Middle	This signal should be asserted for one PCLK250 cycle to initiate the desired action. When asserted in an upstream facing port configuration, this signal initiates the L23 protocol. When asserted in a downstream facing port configuration, this signal acknowledges an L23 request. Note: The transaction layer must not attempt to transmit a TLP after asserting this signal.
DL_PM_ENTERL1	Output	Early	This signal is asserted for one PCLK250 clock cycle when the core is in a downstream facing port configuration and a PM_Enter_L1 DLLP is received.
DL_PM_ACTIVEL1	Output	Early	This signal is asserted for one PCLK250 clock cycle when the core is in a downstream facing port configuration and a PM_Active_State_Request_L1 DLLP is received.
DL_PM_ENTERL23	Output	Early	This signal is asserted for one PCLK250 clock cycle when the core is in a downstream facing port configuration and a PM_Enter_L23 DLLP is received.
DL_PM_INL1	Output	Early	This signal is asserted when the link is in the L1 state (whether due to an active state L1 request or a standard L1 request). It is asserted in both downstream and upstream facing port configurations. Note: The transaction layer must not attempt to transmit a TLP when this signal is asserted.
DL_PM_INL23	Output	Early	This signal is asserted when the link is in the L2/L3 ready state. It is asserted in both downstream and upstream facing port configurations. Note: The transaction layer must not attempt to transmit a TLP when this signal is asserted.

4.8.1 Active State Power Management – L0s Link State

This interface is *not* used to control entry to and exit from the L0s link state. The transmitter enters L0s automatically when L0s is enabled in the link control register and no DLLP or TLP requests have been made for the number of symbol times (PCLK250 clock cycles) for which the core has been configured (see *Section 5.1.4, L0s Entry Timeout Setting, on page 73* for more information). The transmitter exits L0s automatically when a DLLP or TLP request is made. The receiver enters and exits L0s automatically based on the state of the link.

4.8.2 Active State Power Management – L1 Link State

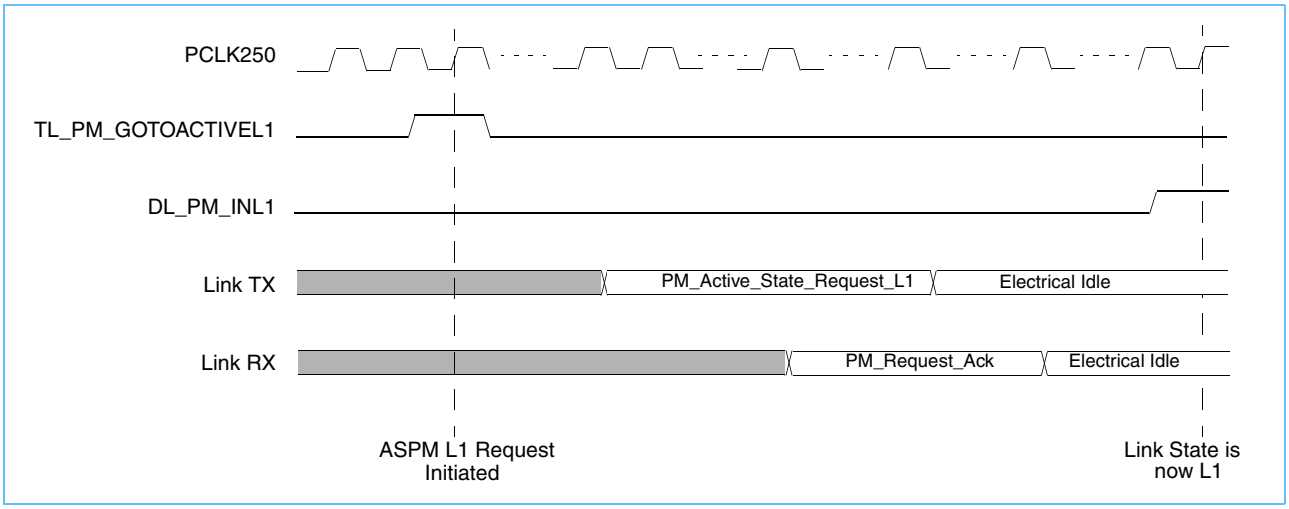
The core supports the ASPM L1 link state as both an upstream and downstream facing port. Upstream facing ports may request this link state, and downstream facing ports may accept or reject the request.

4.8.2.1 ASPM L1 for an Upstream Facing Port

When an upstream facing port requests the ASPM L1 link state and the request is accepted, the sequence of events is as follows (see *Figure 34 on page 51* for an example timing diagram):

- The transaction layer logic asserts TL_PM_GOTOACTIVEL1 for one cycle.
  - This causes PM_Active_State_Request_L1 DLLPs to be transmitted upstream.
- This core receives at least one PM_Request_Ack DLLP from upstream.
  - The remaining stages of the protocol are completed by this core automatically.
- DL_PM_INL1 is asserted when the link is in the ASPM L1 link state.

Figure 34. ASPM L1 Transition – Upstream Facing Port

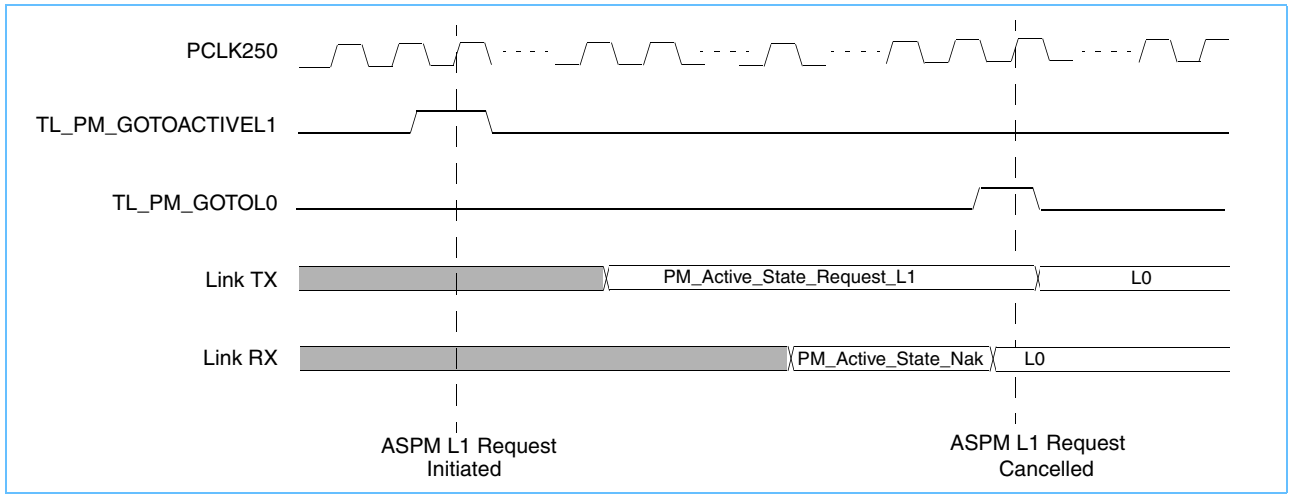


When an upstream facing port requests the ASPM L1 link state and the request is rejected, the sequence of events is as follows (see *Figure 35 on page 52* for an example timing diagram):

- The transaction layer logic asserts TL_PM_GOTOACTIVEL1 for one cycle.
  - This causes PM_Active_State_Request_L1 DLLPs to be transmitted upstream.
- This core does not receive any PM_Request_Ack DLLPs from upstream, but will receive and pass on to the transaction layer a PM_Active_State_Nak TLP.

- The transaction layer logic asserts DL_PM_GOTOL0 for one cycle to cancel the ASPM L1 request.
  - This core then stops transmitting PM_Active_State_Request_L1 DLLPs upstream and the link remains in the L0 link state.
  - A subsequent ASPM L1 request may be initiated one cycle after TL_PM_GOTOL0 is asserted. However, the core contains a timer that ensures that at least 11 μs of time will pass before such a request will initiate PM_Active_State_Request_L1 DLLPs on the link again. This ensures that the upstream component can identify when one ASPM L1 request ends and another begins.

Figure 35. ASPM L1 Rejection – Upstream Facing Port

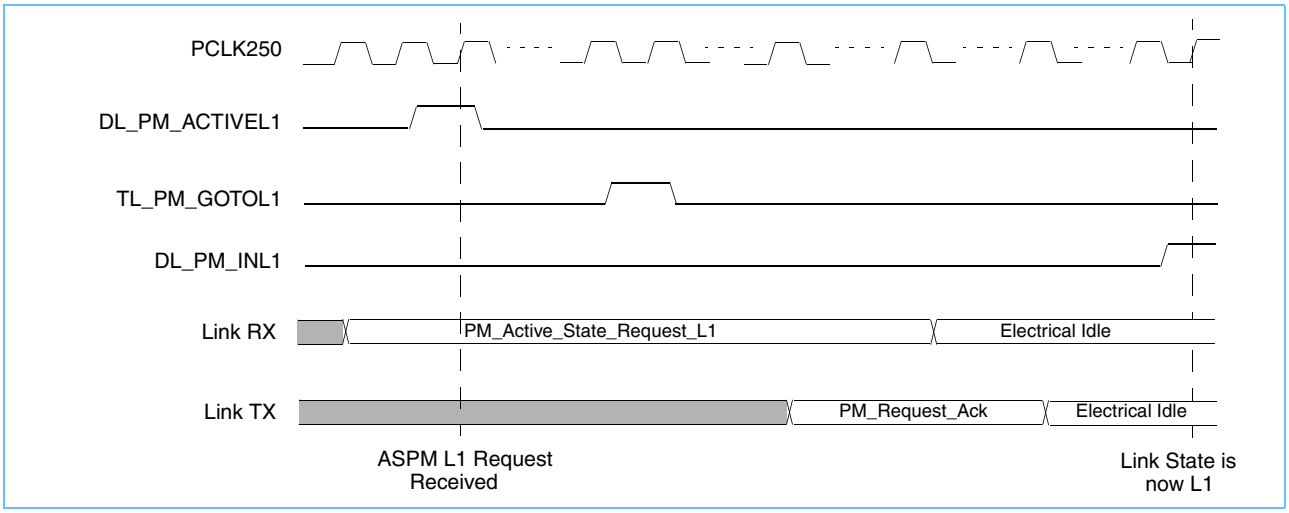


4.8.2.2 ASPM L1 for a Downstream Facing Port

When a downstream facing port receives an ASPM L1 link state request and accepts it, the sequence of events is as follows (see *Figure 36* for an example timing diagram):

- This core receives at least one PM_Active_State_Request_L1 DLLP from downstream.
  - DL_PM_ACTIVEL1 is asserted for one cycle for the first PM_Active_State_Request_L1 DLLP received. Subsequent PM_Active_State_Request_L1 DLLPs do not cause an assertion of DL_PM_ACTIVEL1 until the current request ends. The end of the current request is defined as either Link state L0s on the RX side of the link; or 9.5 μs of time while in the L0 link state (time spent in the Recovery state does not count toward the 9.5 μs) during which no PM_Active_State_Request_L1 DLLPs are received.
- The transaction layer logic asserts TL_PM_GOTOL1 for one cycle.
  - This core transmits PM_Request_Ack DLLPs downstream and completes the remaining stages of the protocol automatically.
- DL_PM_INL1 is asserted when the link is in the ASPM L1 link state.

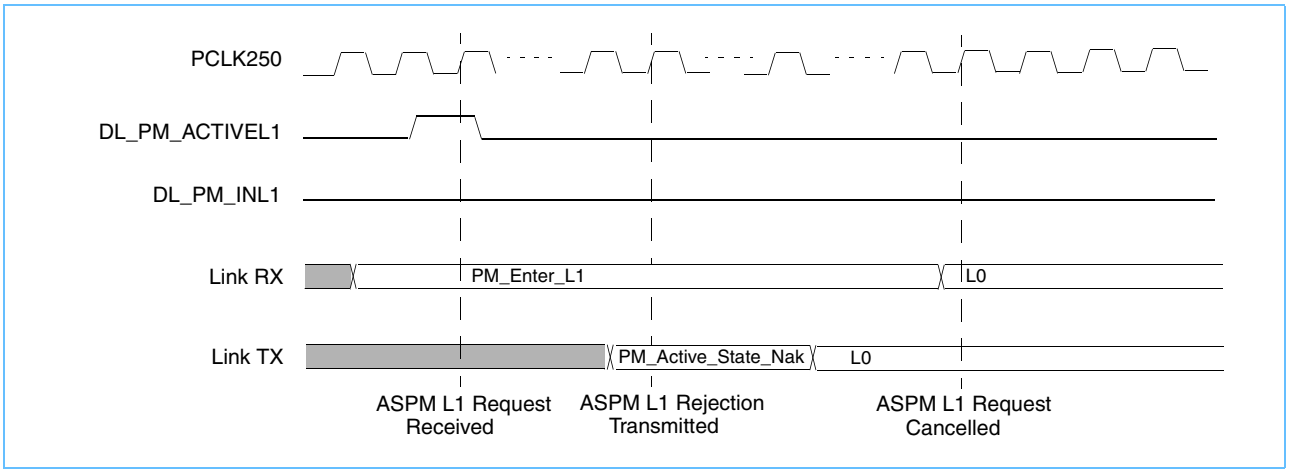
Figure 36. ASPM L1 Transition – Downstream Facing Port



When an downstream facing port receives an ASPM L1 link state request and rejects it, the sequence of events is as follows (see *Figure 37* for an example timing diagram):

- This core receives at least one PM_Active_State_Request_L1 DLLP from downstream.
  - DL_PM_ACTIVEL1 is asserted for one cycle for each PM_Active_State_Request_L1 DLLP received.
- The transaction layer logic transmits a PM_Active_State_Nak TLP downstream.
  - Once the downstream component receives the TLP, the ASPM L1 request is cancelled and no further action is necessary.

Figure 37. ASPM L1 Rejection – Downstream Facing Port



4.8.2.3 Exiting the ASPM L1 Link State

The ASPM L1 link state may be exited as follows:

- The transaction layer asserts TL_PM_GOTOL0 for one cycle.
  - While the link is being recovered, DL_PM_INL1 will be deasserted.
- A receive lane detects a non-electrical idle condition on the link (RXELECIDLE is de-asserted).
  - While the link is being recovered, DL_PM_INL1 will be deasserted.
- When configured as a downstream facing port and SYS_TCTX_DISABLE is asserted.
  - While the link is being recovered, DL_PM_INL1 will be deasserted. The link will then proceed to the Disabled state.
- When configured as a downstream facing port and SYS_TCTX_RESET is asserted.
  - While the link is being recovered, DL_PM_INL1 will be deasserted. The link will then proceed to the Reset state.

4.8.3 PCI Power Management – L1 Link State

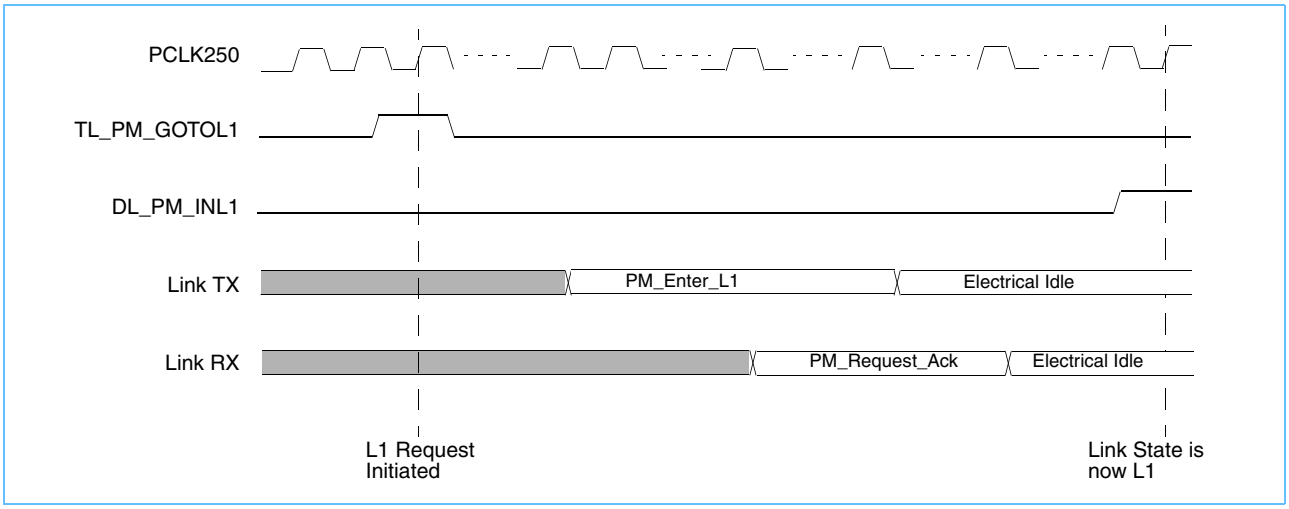
The core supports the PCI-PM L1 link state as both an upstream and downstream facing port. Upstream facing ports initiate this link state, and downstream facing ports must complete the protocol.

4.8.3.1 PCI-PM L1 for an Upstream Facing Port

When an upstream facing port initiates the PCI-PM L1 link state, the sequence of events is as follows (see *Figure 38 on page 54* for an example timing diagram):

- The transaction layer logic asserts TL_PM_GOTOL1 for one cycle.
  - This causes PM_Enter_L1 DLLPs to be transmitted upstream.
- This core receives at least one PM_Request_Ack DLLP from upstream.
  - The remaining stages of the protocol are completed by this core automatically.
- DL_PM_INL1 is asserted when the link is in the PCI-PM L1 link state.

Figure 38. PCI-PM L1 Transition – Upstream Facing Port

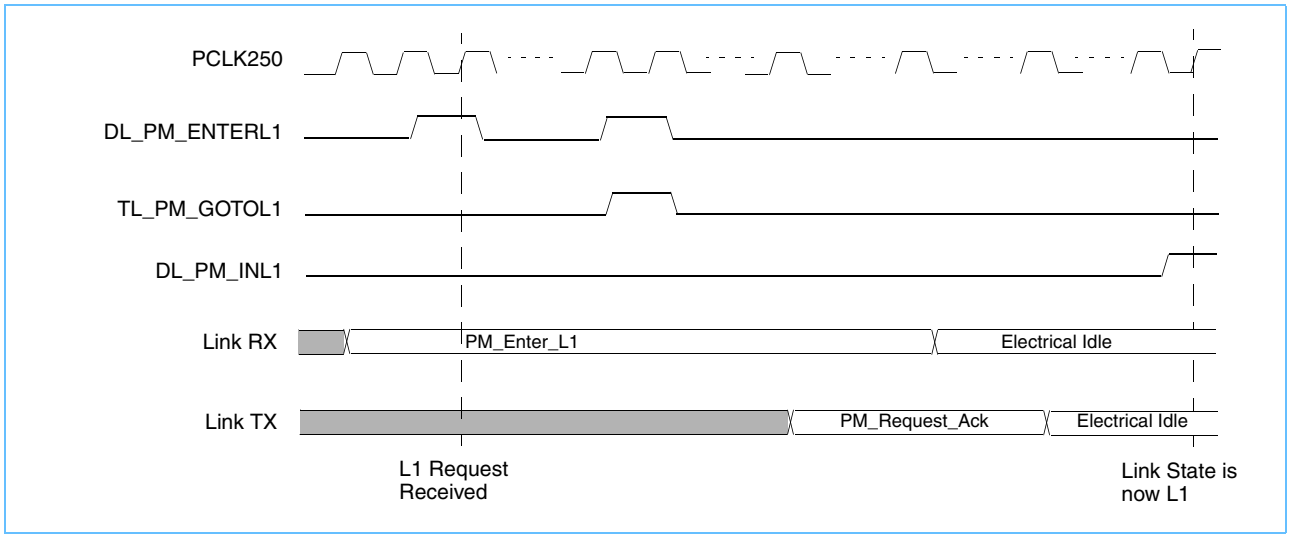


4.8.3.2 PCI-PM L1 for a Downstream Facing Port

When a downstream facing port receives a PCI-PM L1 link state request, the sequence of events is as follows (see *Figure 39* for an example timing diagram):

- This core receives at least one PM_Enter_L1 DLLP from downstream.
  - DL_PM_ENTERL1 is asserted for one cycle for each PM_Enter_L1 DLLP received.
- The transaction layer logic asserts TL_PM_GOTOL1 for one cycle.
  - This core transmits PM_Request_Ack DLLPs downstream and completes the remaining stages of the protocol automatically.
- DL_PM_INL1 is asserted when the link is in the PCI-PM L1 link state.

Figure 39. PCI-PM L1 Transition – Downstream Facing Port

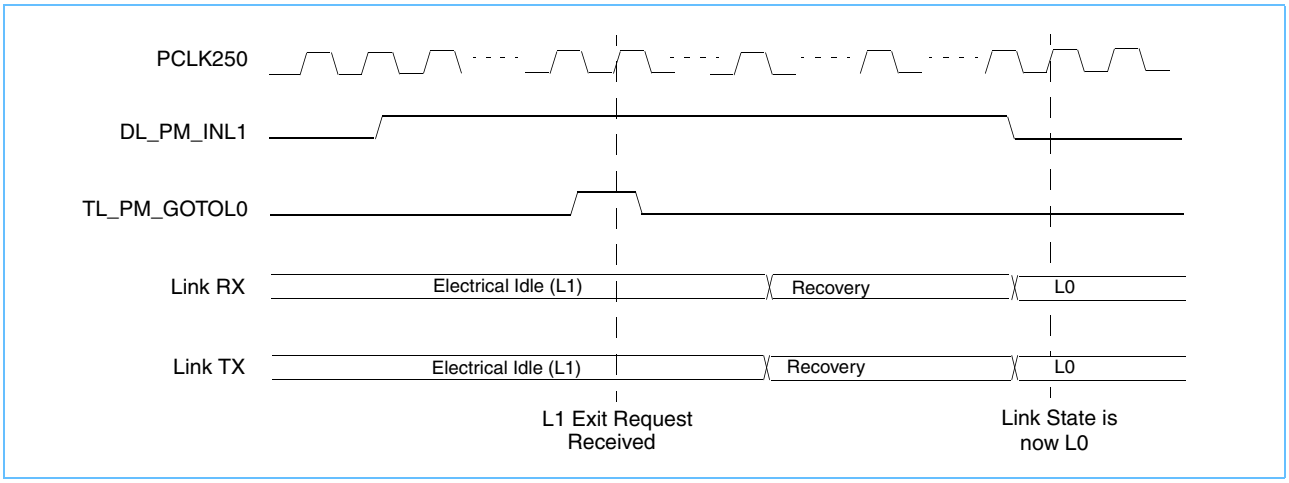


4.8.3.3 Exiting the PCI-PM L1 Link State

The PCI-PM L1 link state may be exited by upstream and downstream facing ports as follows (see *Figure 40 on page 56* for an example timing diagram):

- The transaction layer asserts TL_PM_GOTOL0 for one cycle.
  - While the link is being recovered, DL_PM_INL1 will be deasserted.
- A receive lane detects a non-electrical idle condition on the link (RXELECIDLE is de-asserted).
  - While the link is being recovered, DL_PM_INL1 will be deasserted.
- When configured as a downstream facing port and SYS_TCTX_DISABLE is asserted.
  - While the link is being recovered, DL_PM_INL1 will be deasserted. The link will then proceed to the Disabled state.
- When configured as a downstream facing port and SYS_TCTX_RESET is asserted.
  - While the link is being recovered, DL_PM_INL1 will be deasserted. The link will then proceed to the Reset state.

Figure 40. PCI-PM L1 Exit Transition – Upstream and Downstream Facing Port



4.8.4 PCI Power Management – L2/L3 Ready Link State

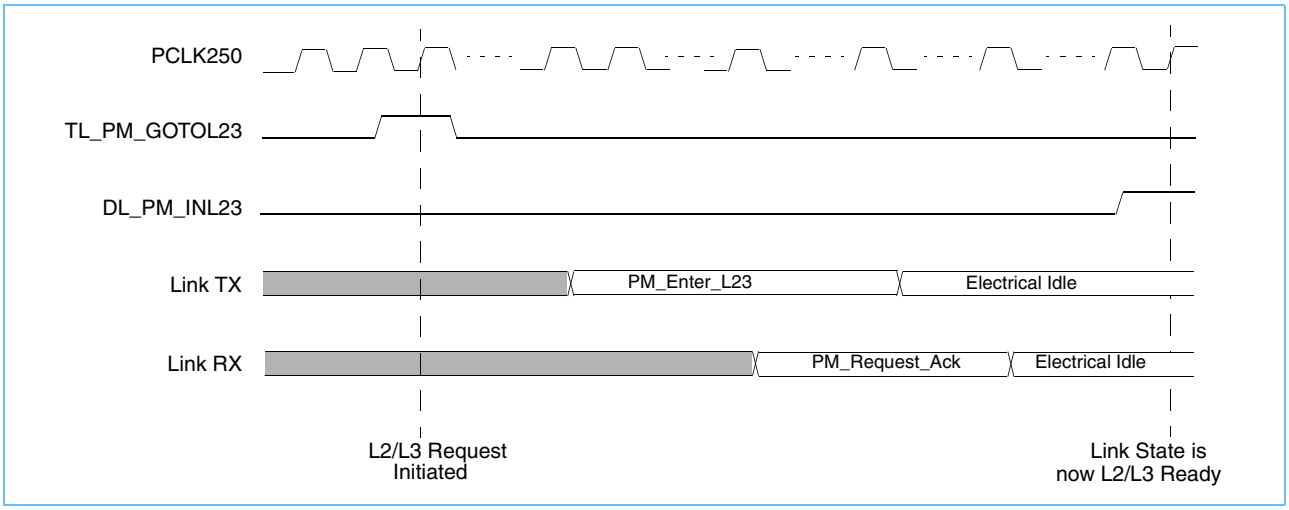
The core supports the PCI-PM L2/L3 ready link state as both an upstream and downstream facing port. Upstream facing ports initiate this link state, and downstream facing ports must complete the protocol.

4.8.4.1 PCI-PM L2/L3 Ready for an Upstream Facing Port

When an upstream facing port initiates the PCI-PM L2/L3 ready link state, the sequence of events is as follows (see *Figure 41 on page 57* for an example timing diagram):

- The transaction layer logic asserts TL_PM_GOTOL23 for one cycle.
  - This causes PM_Enter_L23 DLLPs to be transmitted upstream.
- This core receives at least one PM_Request_Ack DLLP from upstream.
  - The remaining stages of the protocol are completed by this core automatically.
- DL_PM_INL23 is asserted when the link is in the PCI-PM L2/L3 ready link state.

Figure 41. PCI-PM L2/L3 Ready Transition – Upstream Facing Port

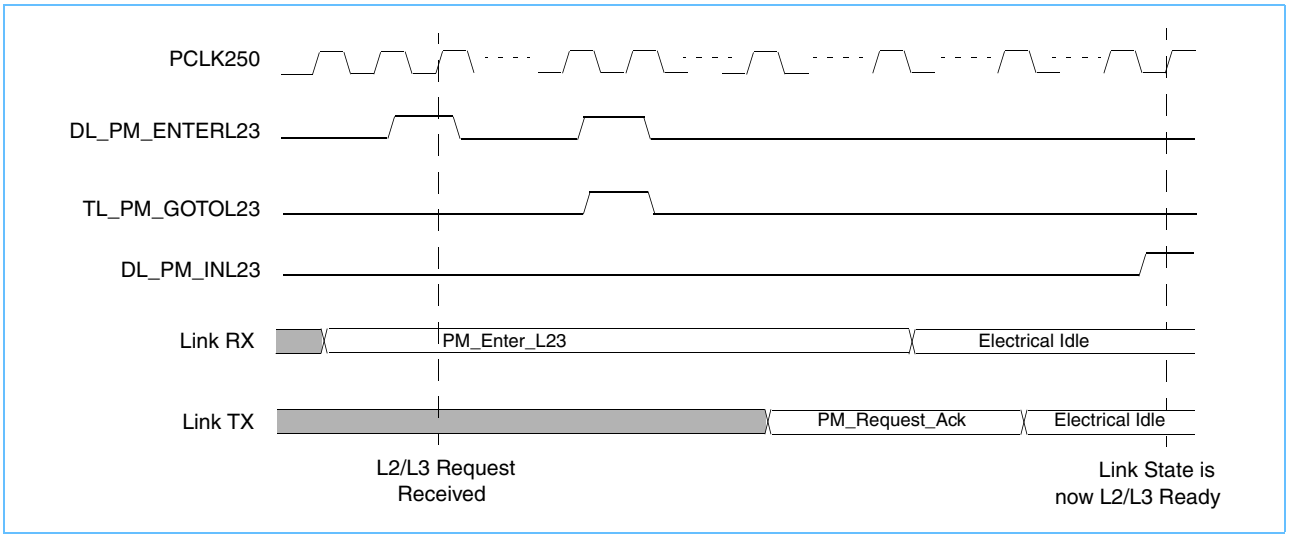


4.8.4.2 PCI-PM L2/L3 for a Downstream Facing Port

When a downstream facing port receives a PCI-PM L2/L3 ready link state request, the sequence of events is as follows (see *Figure 42* for an example timing diagram):

- This core receives at least one PM_Enter_L23 DLLP from downstream.
  - DL_PM_ENTERL23 is asserted for one cycle for each PM_Enter_L23 DLLP received.
- The transaction layer logic asserts TL_PM_GOTOL23 for one cycle.
  - This core transmits PM_Request_Ack DLLPs downstream and completes the remaining stages of the protocol automatically.
- DL_PM_INL23 is asserted when the link is in the PCI-PM L2/L3 ready link state.

Figure 42. PCI-PM L2/L3 Ready Transition – Downstream Facing Port



4.8.4.3 Exiting the PCI-PM L2/L3 Ready Link State

The PCI-PM L2/L3 ready link state can only be exited by asserting SYS_RESETN. This is due to the fact that the attached PHY has been directed to its P2 power state in which PCLK250 is stopped.

4.8.4.4 Use of the DL_PM_INL23 Signal and Clocking Issues

The DL_PM_INL23 signal is asserted when the link is in the L2/L3 ready link state and the PHY has been directed to its P2 power state. Note that assertion of DL_PM_INL23 *does not* guarantee that the PHY is ready for removal of its reference clock. See the *PHY Interface for PCI Express Architecture* specification and the PHY's databook for more information on how and when the PHY will indicate that its reference clock may be stopped.

4.9 PHY Interface

The PHY interface of the core complies with the industry standard *PHY Interface for PCI Express Architecture* (PIPE) specification, version 1.00. The core may not operate correctly with PHYs designed to other versions of the PIPE specification.

For more information on the definition, behavior, and usage of each PHY interface signal, refer to the *PHY Interface for PCI Express Architecture* specification.

Note: Some of this interface's signals are affected by the Physical Link Width configuration option. See *Section 5.1.1 Physical Link Width* on page 71 for more information.

All signals, except as noted, are synchronous to the rising edge of PCLK250. All signals which are defined to have an asserted state are considered positive-active, meaning that their asserted level is a logical '1'.

Figure 43. PHY Interface Signal Diagram

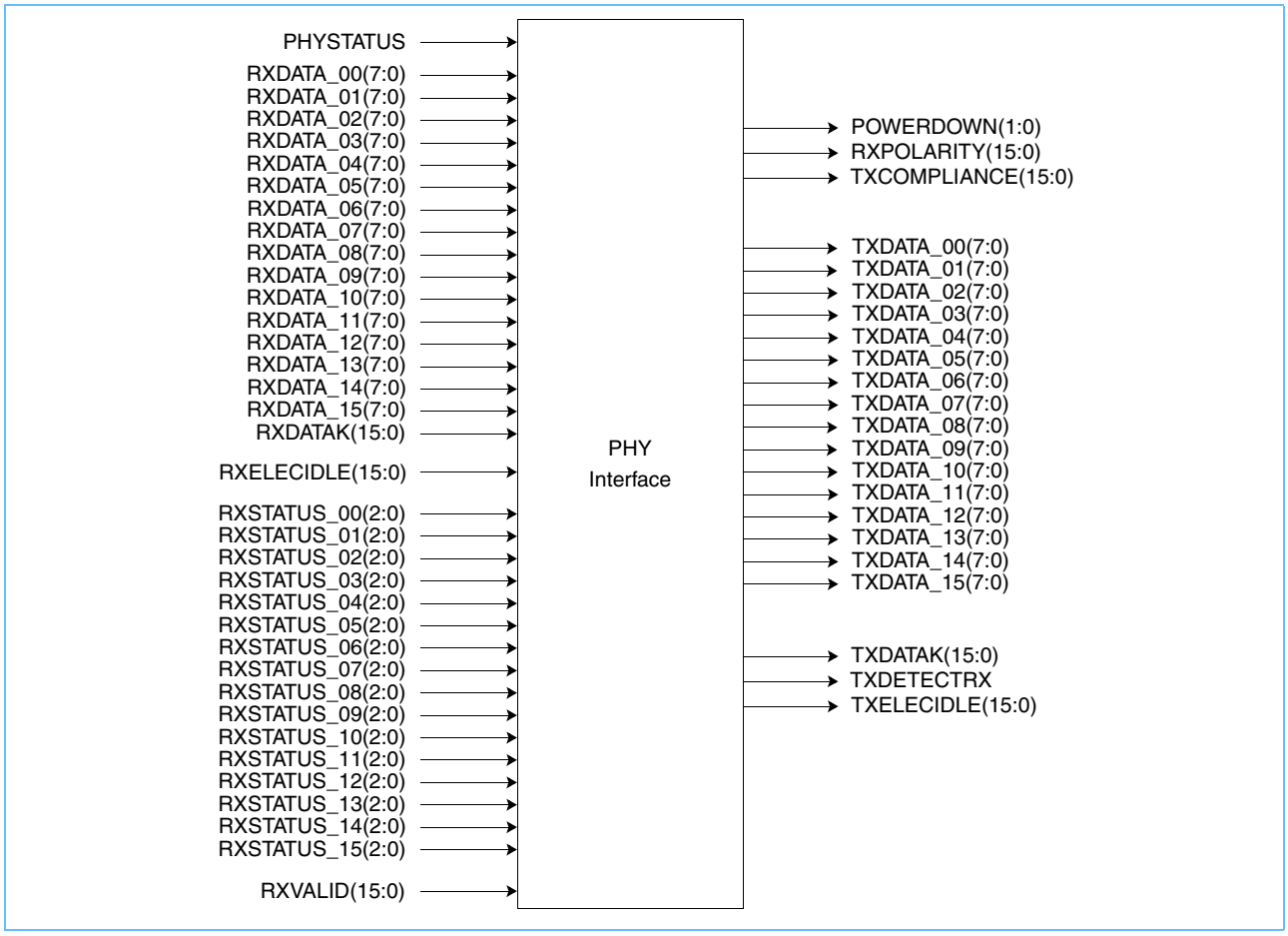


Table 12. PHY Interface Signal Descriptions (Page 1 of 5)

Signal Name	I/O Type	Timing	Description
PHYSTATUS	Input	Middle	This signal is used by the PHY to indicate the completion of one of several operations that it may be directed to perform. When the PHY is starting up or shutting down, PHYSTATUS is an asynchronous signal. For all other operations, however, it is a synchronous signal.
RXDATA_00(7:0)	Input	Middle	Data received on the link's physical lane 0.
RXDATA_01(7:0)	Input	Middle	Data received on the link's physical lane 1. **Note:** This port does not exist when the core is configured for a x1 Physical Link Width.
RXDATA_02(7:0)	Input	Middle	Data received on the link's physical lane 2. **Note:** This port does not exist when the core is configured for a x1 Physical Link Width.
RXDATA_03(7:0)	Input	Middle	Data received on the link's physical lane 3. **Note:** This port does not exist when the core is configured for a x1 Physical Link Width.
RXDATA_04(7:0)	Input	Middle	Data received on the link's physical lane 4. **Note:** This port does not exist when the core is configured for a x1 or x4 Physical Link Width.
RXDATA_05(7:0)	Input	Middle	Data received on the link's physical lane 5. **Note:** This port does not exist when the core is configured for a x1 or x4 Physical Link Width.
RXDATA_06(7:0)	Input	Middle	Data received on the link's physical lane 6. **Note:** This port does not exist when the core is configured for a x1 or x4 Physical Link Width.
RXDATA_07(7:0)	Input	Middle	Data received on the link's physical lane 7. **Note:** This port does not exist when the core is configured for a x1 or x4 Physical Link Width.
RXDATA_08(7:0)	Input	Middle	Data received on the link's physical lane 8. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXDATA_09(7:0)	Input	Middle	Data received on the link's physical lane 9. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXDATA_10(7:0)	Input	Middle	Data received on the link's physical lane 10. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXDATA_11(7:0)	Input	Middle	Data received on the link's physical lane 11. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXDATA_12(7:0)	Input	Middle	Data received on the link's physical lane 12. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXDATA_13(7:0)	Input	Middle	Data received on the link's physical lane 13. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXDATA_14(7:0)	Input	Middle	Data received on the link's physical lane 14. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.

Table 12. PHY Interface Signal Descriptions (Page 2 of 5)

Signal Name	I/O Type	Timing	Description
RXDATA_15(7:0)	Input	Middle	Data received on the link's physical lane 15. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXDATAK(15:0)	Input	Middle	The K-code indicator bit for each lane's RXDATA. Bit 0 corresponds to lane 0 and bit 15 corresponds to lane 15. **Note:** This signal is one bit when the core is configured for a x1 Physical Link Width, (3:0) when the core is configured for a x4 Physical Link Width, (7:0) when the core is configured for a x8 Physical Link Width, and (15:0) when the core is configured for a x16 Physical Link Width.
RXELECIDLE(15:0)	Input	Async	The Electrical Idle indicator bit for each lane. Bit 0 corresponds to lane 0 and bit 15 corresponds to lane 15. **Note:** This signal is one bit when the core is configured for a x1 Physical Link Width, (3:0) when the core is configured for a x4 Physical Link Width, (7:0) when the core is configured for a x8 Physical Link Width, and (15:0) when the core is configured for a x16 Physical Link Width.
RXSTATUS_00(2:0)	Input	Middle	Receiver status indicator for lane 0.
RXSTATUS_01(2:0)	Input	Middle	Receiver status indicator for lane 1. **Note:** This port does not exist when the core is configured for a x1 Physical Link Width.
RXSTATUS_02(2:0)	Input	Middle	Receiver status indicator for lane 2. **Note:** This port does not exist when the core is configured for a x1 Physical Link Width.
RXSTATUS_03(2:0)	Input	Middle	Receiver status indicator for lane 3. **Note:** This port does not exist when the core is configured for a x1 Physical Link Width.
RXSTATUS_04(2:0)	Input	Middle	Receiver status indicator for lane 4. **Note:** This port does not exist when the core is configured for a x1 or x4 Physical Link Width.
RXSTATUS_05(2:0)	Input	Middle	Receiver status indicator for lane 5. **Note:** This port does not exist when the core is configured for a x1 or x4 Physical Link Width.
RXSTATUS_06(2:0)	Input	Middle	Receiver status indicator for lane 6. **Note:** This port does not exist when the core is configured for a x1 or x4 Physical Link Width.
RXSTATUS_07(2:0)	Input	Middle	Receiver status indicator for lane 7. **Note:** This port does not exist when the core is configured for a x1 or x4 Physical Link Width.
RXSTATUS_08(2:0)	Input	Middle	Receiver status indicator for lane 8. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXSTATUS_09(2:0)	Input	Middle	Receiver status indicator for lane 9. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXSTATUS_10(2:0)	Input	Middle	Receiver status indicator for lane 10. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXSTATUS_11(2:0)	Input	Middle	Receiver status indicator for lane 11. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.

Table 12. PHY Interface Signal Descriptions (Page 3 of 5)

Signal Name	I/O Type	Timing	Description
RXSTATUS_12(2:0)	Input	Middle	Receiver status indicator for lane 12. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXSTATUS_13(2:0)	Input	Middle	Receiver status indicator for lane 13. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXSTATUS_14(2:0)	Input	Middle	Receiver status indicator for lane 14. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXSTATUS_15(2:0)	Input	Middle	Receiver status indicator for lane 15. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
RXVALID(15:0)	Input	Middle	The data valid indicator for each lane's RXDATA. Bit 0 corresponds to lane 0 and bit 15 corresponds to lane 15. **Note:** This signal is one bit when the core is configured for a x1 Physical Link Width, (3:0) when the core is configured for a x4 Physical Link Width, (7:0) when the core is configured for a x8 Physical Link Width, and (15:0) when the core is configured for a x16 Physical Link Width.
POWERDOWN(1:0)	Output	Early	The power state of the PHY. The encodings are as follows: '00' = P0, normal operation. '01' = P0s, low power state, not transmitting, receive may be active. '10' = P1, low power state, not transmitting or receiving. '11' = P2, lowest power state, all clocks and PLL shut down.
RXPOLARITY(15:0)	Output	Early	Controls receiver polarity reversal for each lane's RXDATA. Bit 0 corresponds to lane 0 and bit 15 corresponds to lane 15. **Note:** This signal is one bit when the core is configured for a x1 Physical Link Width, (3:0) when the core is configured for a x4 Physical Link Width, (7:0) when the core is configured for a x8 Physical Link Width, and (15:0) when the core is configured for a x16 Physical Link Width.
TXCOMPLIANCE(15:0)	Output	Early	Controls transmitter encoding disparity of each lane's TXDATA. Bit 0 corresponds to lane 0 and bit 15 corresponds to lane 15. **Note:** This signal is one bit when the core is configured for a x1 Physical Link Width, (3:0) when the core is configured for a x4 Physical Link Width, (7:0) when the core is configured for a x8 Physical Link Width, and (15:0) when the core is configured for a x16 Physical Link Width.
TXDATA_00(7:0)	Output	Early	Transmit data for the link's physical lane 0.
TXDATA_01(7:0)	Output	Early	Transmit data for the link's physical lane 1. **Note:** This port does not exist when the core is configured for a x1 Physical Link Width.
TXDATA_02(7:0)	Output	Early	Transmit data for the link's physical lane 2. **Note:** This port does not exist when the core is configured for a x1 Physical Link Width.
TXDATA_03(7:0)	Output	Early	Transmit data for the link's physical lane 3. **Note:** This port does not exist when the core is configured for a x1 Physical Link Width.

Table 12. PHY Interface Signal Descriptions (Page 4 of 5)

Signal Name	I/O Type	Timing	Description
TXDATA_04(7:0)	Output	Early	Transmit data for the link's physical lane 4. **Note:** This port does not exist when the core is configured for a x1 or x4 Physical Link Width.
TXDATA_05(7:0)	Output	Early	Transmit data for the link's physical lane 5. **Note:** This port does not exist when the core is configured for a x1 or x4 Physical Link Width.
TXDATA_06(7:0)	Output	Early	Transmit data for the link's physical lane 6. **Note:** This port does not exist when the core is configured for a x1 or x4 Physical Link Width.
TXDATA_07(7:0)	Output	Early	Transmit data for the link's physical lane 7. **Note:** This port does not exist when the core is configured for a x1 or x4 Physical Link Width.
TXDATA_08(7:0)	Output	Early	Transmit data for the link's physical lane 8. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
TXDATA_09(7:0)	Output	Early	Transmit data for the link's physical lane 9. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
TXDATA_10(7:0)	Output	Early	Transmit data for the link's physical lane 10. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
TXDATA_11(7:0)	Output	Early	Transmit data for the link's physical lane 11. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
TXDATA_12(7:0)	Output	Early	Transmit data for the link's physical lane 12. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
TXDATA_13(7:0)	Output	Early	Transmit data for the link's physical lane 13. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
TXDATA_14(7:0)	Output	Early	Transmit data for the link's physical lane 14. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
TXDATA_15(7:0)	Output	Early	Transmit data for the link's physical lane 15. **Note:** This port does not exist when the core is configured for a x1, x4, or x8 Physical Link Width.
TXDATAK(15:0)	Output	Early	K-code indicator bit for each lane's TXDATA. Bit 0 corresponds to lane 0 and bit 15 corresponds to lane 15. **Note:** This signal is one bit when the core is configured for a x1 Physical Link Width, (3:0) when the core is configured for a x4 Physical Link Width, (7:0) when the core is configured for a x8 Physical Link Width, and (15:0) when the core is configured for a x16 Physical Link Width.
TXDETECTRX	Output	Early	When asserted, instructs the PHY to execute a receiver detection sequence on all lanes.

Table 12. PHY Interface Signal Descriptions (Page 5 of 5)

Signal Name	I/O Type	Timing	Description
TXELECIDLE(15:0)	Output	Early	Electrical idle indicator bit for each lane's transmitter. Bit 0 corresponds to lane 0 and bit 15 corresponds to lane 15. **Note:** This signal is one bit when the core is configured for a x1 Physical Link Width, (3:0) when the core is configured for a x4 Physical Link Width, (7:0) when the core is configured for a x8 Physical Link Width, and (15:0) when the core is configured for a x16 Physical Link Width.

4.9.1 PHY Requirements to Support Exit from the L2/3 Ready Link State

When the core enters the L2/3 Ready link state, it directs the PHY to the P2 power state which is defined to stop the PCLK250 clock from the PHY. However, PCLK250 must be running (not necessarily at 250 MHz) in order for the core to exit from the L2/3 Ready link state. If it is running, then the core exits to the Detect link state if the following conditions occur:

- As an upstream facing port: De-assertion of any RXELECIDLE signal of lanes that are part of the configured link.
- As a downstream facing port: De-assertion of the RXELECIDLE signal corresponding to lane 0 of the configured link, or assertion of SYS_TCTX_RESET (see *Section 4.4.1.2 Training Control Reset* on page 30 for more information).

If an exit condition is detected and the state transition is executed, the core directs the PHY to the P1 power state with the POWERDOWN signal. The PHY asserts PHYSTATUS asynchronously when it recognizes the change of POWERDOWN and eventually de-asserts PHYSTATUS synchronously when it completes the power state transition. During this period, the core requires PHYSTATUS to be asserted for at least three rising edges of PCLK250 and PCLK250 must be running at 250 MHz before PHYSTATUS is de-asserted.

4.10 TLP Replay Interface

The TLP Replay Interface connects to a synchronous one-port RAM (or equivalent function) and stores TLPs for future replay (if needed) by the link. The RAM must have the following features:

- Synchronous read and write initiated by the rising edge of the clock.
- After a read access takes place, the read data must remain on the RAM outputs until another read access is initiated.
- The RAM's write data input must be connected to TL_TX_DATA external to this core.
- The RAM must be clocked with PCLK250—the same clock connected to this core—and therefore must be capable of running at 250 MHz.

Note: Some of this interface's signals are affected by the Physical Link Width configuration option. See *Section 5.1.1 Physical Link Width* on page 71 for more information.

Note: The replay buffer is described as a single entity, but because it may have a data bus width of 128 bits, multiple RAMs may be needed to create the proper configuration.

All inputs and outputs are synchronous to the rising edge of PCLK250. All signals which are defined to have an asserted state are considered positive-active; meaning that their asserted level is a logical '1'.

Figure 44. TLP Replay Interface Signal Diagram

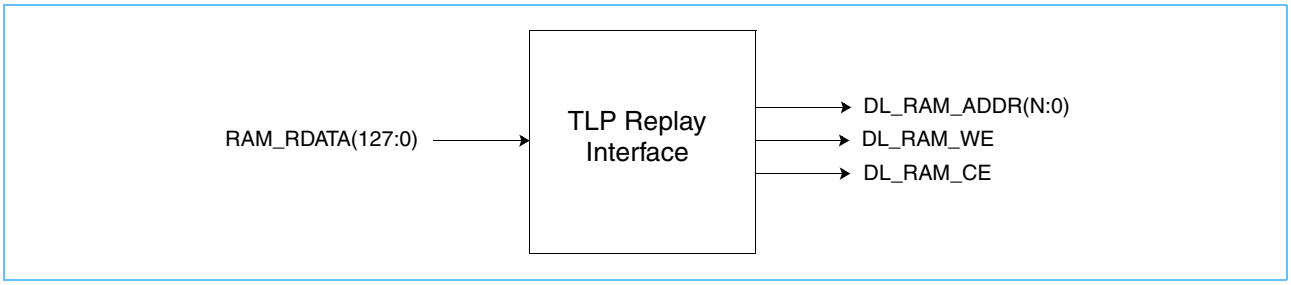


Table 13. TLP Replay Interface Signal Descriptions

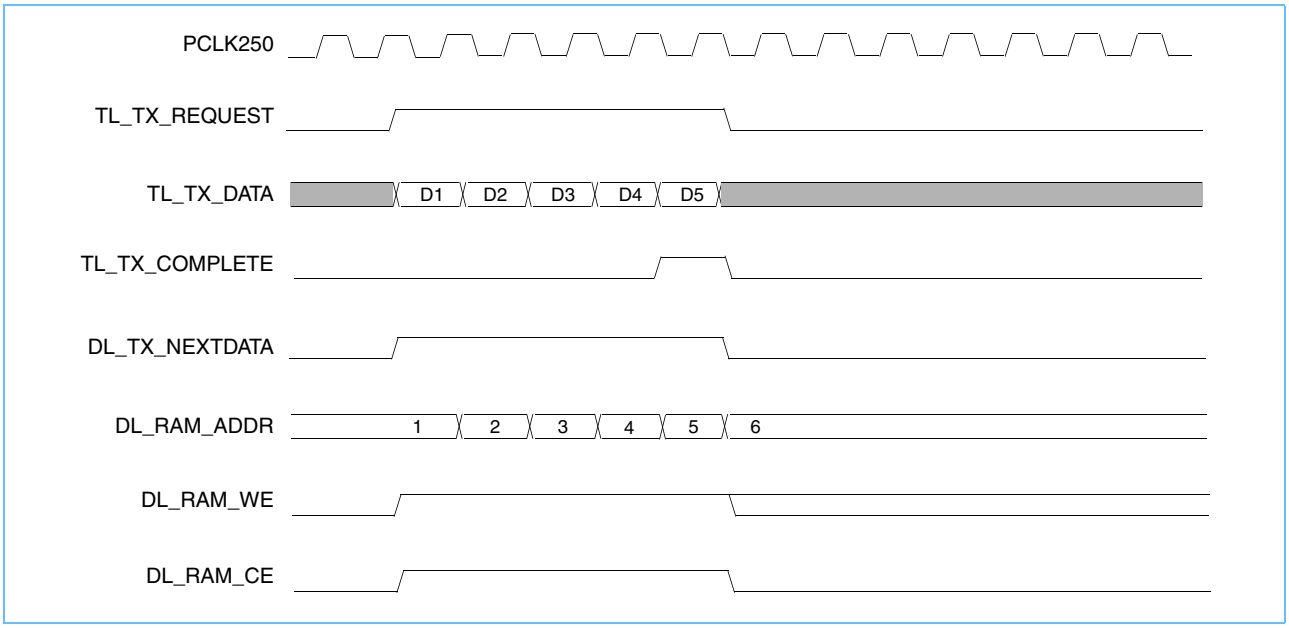
Signal Name	I/O Type	Timing	Description
RAM_RDATA(127:0) RAM_RDATA(63:0) RAM_RDATA(31:0)	Input	Late	This is the RAM's read data output. Data read from the RAM must remain stable until another read is initiated. **Note:** The width of this signal depends on the configuration of the core. It is (31:0) when the core is configured for a x1 or x4 Physical Link Width, (63:0) when configured for a x8 Physical Link Width, and (127:0) when configured for a x16 Physical Link Width.
DL_RAM_ADDR(N:0)	Output	Early	The address to read data from or write data to. The width of this bus is determined by the Max Payload Size Supported configuration option. See *Section 5.1.2, Replay Buffer Size, on page 71* for more information.
DL_RAM_WE	Output	Early	The write enable signal for the RAM.
DL_RAM_CE	Output	Middle	The chip enable signal for the RAM. It is asserted when either a read or a write cycle is initiated. The determination of read versus write is made with the DL_RAM_WE signal.

4.10.1 Writing TLP Data to the Replay Buffer

Data is written to the replay buffer during a transaction layer TLP transmit request. See *Figure 45 on page 66* for an example timing diagram of a write sequence.

1. A TLP transmission is requested by asserting TL_TX_REQUEST and driving TL_TX_DATA with the data to transmit.
2. When the core is ready to transmit the TLP, it signals the acceptance of the data by asserting DL_TX_NEXTDATA. At same time, it initiates a write to the RAM attached to the TLP Replay interface by driving DL_RAM_ADDR and asserting DL_RAM_WE and DL_RAM_CE.
   - Depending on the negotiated link width and the length of the TLP request, DL_RAM_ADDR may change every cycle and DL_RAM_WE and DL_RAM_CE may be asserted in consecutive cycles. When writing, DL_RAM_CE is asserted the same time DL_TX_NEXTDATA is asserted on the TLP transmit Interface.

Figure 45. Timing Diagram for TLP Replay Write – Negotiated Link Size of x16

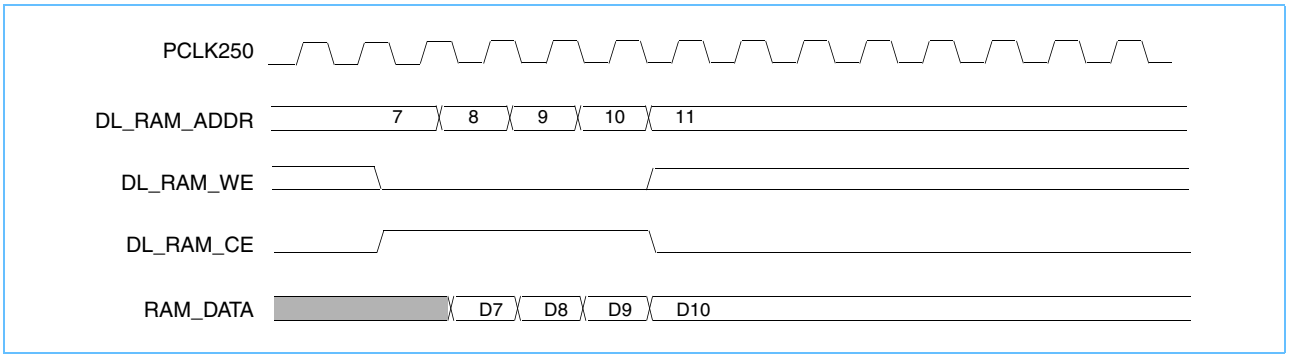


4.10.2 Reading TLP Data from a Replay Buffer Configured for One-Cycle Access

Data is read from the replay buffer when a replay is initiated. Refer to *Figure 46 on page 67* for an example timing diagram for a read sequence.

1. When a replay is initiated, the core drives DL_RAM_ADDR to the starting address of the TLP(s) to be replayed. DL_RAM_WE is driven low to indicate a read operation, and DL_RAM_CE is asserted for one cycle for each address to be read from.
   - RAM_DATA must hold its value until a new read is initiated. Depending on the negotiated link width, reads may take place every cycle or may have idle cycles between reads.
2. When DL_RAM_CE is asserted, a rising edge of PCLK250 initiates a read from the RAM. When configured for a one-cycle read access time, the read must take place and RAM_RDATA must be valid before the next rising edge of PCLK250.
   - RAM_RDATA must hold its value until a new read is initiated. Depending on the negotiated link width, reads may take place every cycle or may have idle cycles between reads.

Figure 46. Timing Diagram for TLP Replay Read – One-Cycle Access

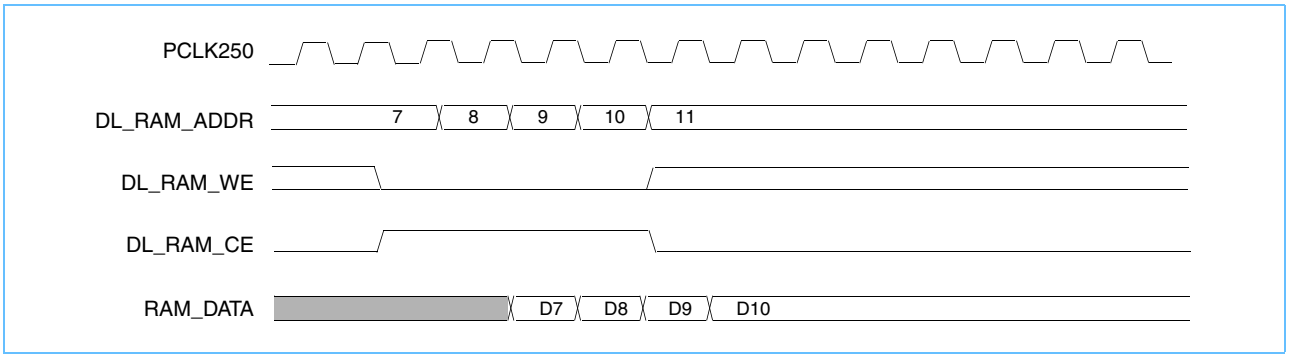


4.10.3 Reading TLP Data from a Replay Buffer Configured for Two-Cycle Access

Data is read from the replay buffer when a replay is initiated. Refer to *Figure 47 on page 67* for an example timing diagram for a read sequence.

1. When a replay is initiated, the core drives DL_RAM_ADDR to the starting address of the TLP(s) to be replayed. DL_RAM_WE is driven low to indicate a read operation, and DL_RAM_CE is asserted for one cycle for each address to be read from.
   - RAM_DATA must hold its value until a new read is initiated. Depending on the negotiated link width, reads may take place every cycle or may have idle cycles between reads.
2. When DL_RAM_CE is asserted, a rising edge of PCLK250 initiates a read from the RAM. When configured for a two-cycle read access time, the read must take place and RAM_RDATA must be valid before the second rising edge of PCLK250 after DL_RAM_CE is asserted. The RAM must still be capable of producing read data every cycle.
   - RAM_RDATA must hold its value until a new read is initiated. Depending on the negotiated link width, reads may take place every cycle or may have idle cycles between reads.

Figure 47. Timing Diagram for TLP Replay Read – Two-Cycle Access



4.10.4 RAM W/R and R/W Turn-around Time

When switching from write mode (new TLP from the transaction layer) to read mode (replay of previously transmitted TLP), the DL_RAM_WE and DL_RAM_CE signals are deasserted for at least one PCLK250 cycle before a read is initiated.

When switching from read mode (replay of previously transmitted TLP) to write mode (new TLP from the transaction layer), the DL_RAM_CE signal is deasserted for at least one PCLK250 cycle before a write is initiated. The DL_RAM_WE signal may be asserted the same cycle DL_RAM_CE is asserted to initiate the next write.

4.11 Loopback Interface

The loopback interface is active when the core is in master loopback mode. It can be used for link diagnostic purposes. There are two signals per lane: one to indicate whether the lane is in loopback, and one to indicate when an error has been detected. The pattern transmitted by the core is as follows:

- 64 symbols of the compliance pattern (used to provide interference between adjacent lanes)
- One skip-ordered set
- Repeat until directed to exit loopback mode

The following errors will be detected:

- 8b/10b disparity and decode
- Receiver underflow or overflow
- Unexpected electrical idle at the receiver
- Loss of symbol lock at the receiver

Note: Some of this interface's signals are affected by the Physical Link Width configuration option. See *Section 5.1.1 Physical Link Width* on page 71 for more information.

All inputs and outputs are synchronous to the rising edge of PCLK250. All signals which are defined to have an asserted state are considered positive-active; meaning that their asserted level is a logical '1'.

Figure 48. Loopback Interface Signal Diagram

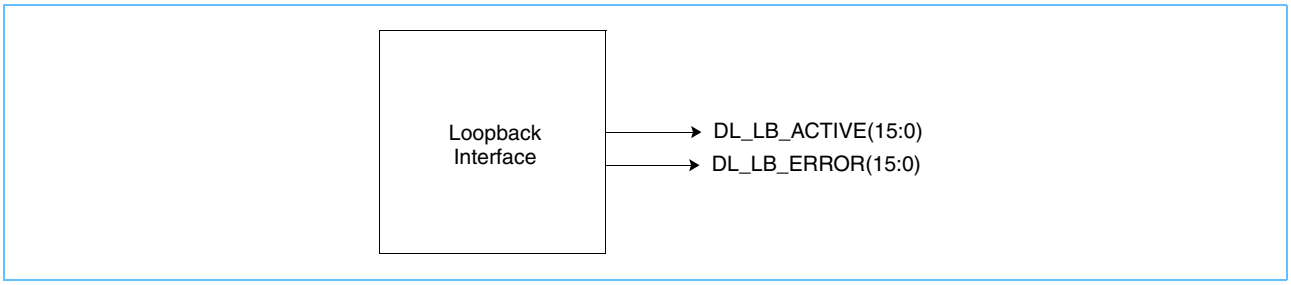


Table 14. Loopback Interface Signal Descriptions

Signal Name	I/O Type	Timing	Description
DL_LB_ACTIVE(15:0)	Output	Middle	This signal is asserted per lane when the lane is in loopback. **Note:** This signal is one bit when the core is configured for a x1 Physical Link Width, (3:0) when the core is configured for a x4 Physical Link Width, (7:0) when the core is configured for a x8 Physical Link Width, and (15:0) when the core is configured for a x16 Physical Link Width.
DL_LB_ERROR(15:0)	Output	Middle	This signal is asserted per lane for one PCLK250 cycle each time an error is detected in the received loopback data stream. **Note:** This signal is one bit when the core is configured for a x1 Physical Link Width, (3:0) when the core is configured for a x4 Physical Link Width, (7:0) when the core is configured for a x8 Physical Link Width, and (15:0) when the core is configured for a x16 Physical Link Width.

4.11.1 Entering Master Loopback Mode

See *Section 4.4, Training Control Interface, on page 27* for information on how to direct the core to attempt to enter master loopback mode. Once directed, the following sequence is performed:

1. If at least one lane has been determined to be operable, the core enters loopback negotiation which consists of transmitting TS1 ordered sets with the Loopback bit set in the training control field and waiting for them to be looped back. If at least one lane completes loopback negotiation, the core proceeds to checking. DL_TCRX_LOOPBACK is asserted at this time.
2. The first checking step is to look for a sequence of two bytes on each lane: K28.5 and D21.5. For each lane that receives that sequence without errors, checking is enabled and that lane's DL_LB_ACTIVE signal is asserted. From that time on, that lane's DL_LB_ERROR is asserted when an error is detected.

The SYS_TCRX_LOOPBACK, DL_LB_ACTIVE, and DL_LB_ERROR signals may be used in the following manner:

- If SYS_TCRX_LOOPBACK is asserted, but no DL_LB_ACTIVE signals are asserted, this indicates that no lanes have successfully completed loopback negotiation. The link returns to the detect state after a 2 ms timeout.
- If SYS_TCRX_LOOPBACK is asserted and at least one DL_LB_ACTIVE signal is asserted, this indicates that all lanes for which DL_LB_ACTIVE is asserted have completed the loopback negotiation process and are being monitored for errors. DL_LB_ERROR is asserted for each of those lanes when an error is detected.
- DL_LB_ERROR is not asserted for lanes that have not completed the loopback negotiation process.

4.11.2 Exiting Master Loopback Mode

See *Section 4.4, Training Control Interface, on page 27* for information on how to direct the core to exit master loopback mode.

4.11.3 Entering Slave Loopback Mode

The core enters slave loopback mode automatically when requested by the remote component.

4.11.4 Exiting Slave Loopback Mode

The core exits slave loopback mode automatically when it receives an Electrical Idle ordered set from the remote component.

Note: The core continues to direct the attached PHY to loopback incoming data for 19 PCLK250 cycles after receiving the Electrical Idle ordered set. This should allow the PHY sufficient time to loopback the ordered set, but the PHY specification should be checked for compatibility with this behavior.

5. Core Integration

5.1 Configurability

The core contains five configurable items: the physical link width, the replay buffer size, the replay buffer read access time, the L0s entry timeout value, and use of asynchronously set/reset flip-flops.

Configurability is controlled with the PCIEX16DLP.defines.v file.

5.1.1 Physical Link Width

The physical link width supported by the core is controlled with three parameters:

- PCIEXDLP_LINKWIDTH_X1
- PCIEXDLP_LINKWIDTH_X4
- PCIEXDLP_LINKWIDTH_X8
- PCIEXDLP_LINKWIDTH_X16

Only one of these parameters may be defined at any one time.

This configuration item is intended to reduce the size of the synthesized core when implementing x1, x4, or x8 PCI Express ports. Note that several signal interfaces on the core are modified depending on the physical link width selected. For example, the TL_TX_DATA and DL_RX_DATA buses are 32 bits wide when either a x1 or x4 physical link width is chosen. The interface and signal descriptions indicate all signals which are modified with this configuration item.

Note: These parameters correspond to the LINKWIDTH parameters which control the link width of the IBM PIPE PHY core. The parameters of both cores should be set in the same manner.

5.1.2 Replay Buffer Size

The replay buffer size is determined by three items:

1. The value which the device will advertise to the system in the Max Payload Size Supported field of the Device Capabilities register.
2. A Replay Buffer Size Multiplication Factor of either 4 or 8 times the Max Payload Size Supported value.
3. The setting of the Physical Link Width parameter (see *Section 5.1.1 Physical Link Width* on page 71). This parameter affects the data bus width of the TL_TX_DATA bus and therefore the replay buffer's data and address bus widths.

Results of combining the items are shown in *Table 15*, which shows the replay buffer size and the width of the address bus (for x1, x4, x8 and x6 Physical Link Width configurations) to the RAM used as the replay buffer.

Table 15. Max_Payload_Size versus Replay Buffer Configuration Settings (Page 1 of 2)

Max Payload Size Supported (Bytes)	Multiplication Factor	Replay Buffer Size (Bytes)	RAM Address Bus Width Physical Link Width of x16/x8/x4/x1
128	4	512	5/6/7/7
256	4	1024	6/7/8/8
512	4	2048	7/8/9/9
1024	4	4096	8/9/10/10

Table 15. Max_Payload_Size versus Replay Buffer Configuration Settings (Page 2 of 2)

Max Payload Size Supported (Bytes)	Multiplication Factor	Replay Buffer Size (Bytes)	RAM Address Bus Width Physical Link Width of x16/x8/x4/x1
2048	4	8192	9/10/11/11
4096	4	16,384	10/11/12/12
128	8	1024	6/7/8/8
256	8	2048	7/8/9/9
512	8	4096	8/9/10/10
1024	8	8192	9/10/11/11
2048	8	16,384	10/11/12/12
4096	8	32,768	11/12/13/13

The parameters that control the Maximum Payload Size Supported setting are:

- PCIEXDLP_MPSS_128
- PCIEXDLP_MPSS_256
- PCIEXDLP_MPSS_512
- PCIEXDLP_MPSS_1024
- PCIEXDLP_MPSS_2048
- PCIEXDLP_MPSS_4096

Only one of these parameters may be defined at any one time.

The Replay Buffer Size Multiplication Factor is controlled by one parameter:

- PCIEXDLP_REPLAYSIZE_4X

If the parameter is defined, a multiplication factor of 4 is used. If it is ***NOT*** defined, a multiplication factor of 8 is used.

Note that the maximum number of TLPs stored in the replay buffer will be either:

- 3 TLPs which contains a payload of the maximum size supported (if a multiplication factor of 4 is selected).
- 7 TLPs which contains a payload of the maximum size supported (if a multiplication factor of 8 is selected).
- 8 TLPs if their combined size (including header, payload, and digest) does not exceed the replay buffer size.

5.1.3 Replay Buffer Read Access Time

The replay buffer read access time may be set to one cycle (default) or two cycles with the parameter:

PCIEXDLP_2_CYCLE_REPLAY_ACCESS

If the parameter is defined, the read access time of the replay buffer is set to two cycles. Defining this parameter does not change the data throughput requirement of the replay buffer. See section *Section 4.10 TLP Replay Interface* on page 65 for a full description of the interface requirements.

5.1.4 L0s Entry Timeout Setting

The L0s entry policy is based on the number of symbol times elapsed with no TLPs or DLLPs pending for transmission. The number of symbol times may be set to a value of 10 to 255 symbol times, yielding a timeout of 40 ns to 1020 ns.

The PCIEXDLP_LOSTIMEOUT parameter controls this setting. It must be set to an 8-bit integer value of 10 to 255.

5.1.5 Use of Synchronous or Asynchronous Set/Reset Flip-Flops

This core's default configuration requires the use of flip-flops that can be set/reset when the PCLK250 clock is not running to ensure correct initialization of the logic while SYS_RESETN is asserted. Such flip-flops are commonly referred to as asynchronous set/reset flip-flops, and use of them allows the core to support any PIPE compliant PHY.

It is possible, however, that PCLK250 is running when SYS_RESETN is asserted and therefore asynchronous set/reset flip-flops are not necessary. The core configuration can be changed so that flip-flops without asynchronous set/reset are sufficient for synthesis. The parameter that controls the configuration is:

- PCIEXDLP_SYNC_RESET

If the parameter is defined, the core does not require asynchronous set/reset flip-flops.

5.2 Synthesis Guidelines

5.2.1 User Defined Synthesis and Timing Options

5.2.1.1 Synthesis

The Synopsys Design Compiler synthesis script supplied has one user defined setup option: the wire load model used for net loading estimation. The script should be edited and the wire load model name changed to a wire load that exists in the target library.

5.2.1.2 Timing

The Synopsys SDC timing constraints file supplied has one user defined setup option: the amount of clock uncertainty applied to the PCLK250 clock source. This parameter may be used to model clock jitter and/or used to provide additional pre-layout timing margin if necessary. The script should be edited and the parameter changed to a number that is appropriate for the synthesis, timing, and layout methodology used.

The core also contains flip-flop pairs which are used to capture asynchronous input signals. *Table 16* lists the captured signal, the standard synthesized instance names of the capture flip-flops, and the capture clock's name. Correct operation of these flip-flop pairs may be ensured in one of two ways:

1. Ensure that the delay from the output of the first flip-flop to the input of the second flip-flop is at most 25% of the capture clock's period.
2. Ensure that the two flip-flops are placed next to each other, their clock inputs are connected to the same net with minimum wiring length between them, and there is minimum wiring length between the output of the first and the input of the second.

Table 16. Asynchronous Capture Flip-Flop Pairs in the Design

Signal to be Captured	Flip-Flop Instance Names	Capture Clock
RXELECIDLE[15:0] - core input	U_RXA_LANE_yy/RxElecIdleM1_S0_reg U_RXA_LANE_yy/RxElecIdle_S0_reg yy = lane number	PCLK250
PHYSTATUS - core input	U_TXS/LP/PHYSTATUS1_S0_reg U_TXS/LP/PHYSTATUS2_S0_reg	PCLK250

5.2.2 Clocking

This core operates entirely on a 250 MHz clock (PCLK250) supplied by the PHY. The frequency is 250 MHz +/- 300 ppm, and spread spectrum clocking may be used if desired.

Clock gating may be employed during synthesis of the core if desired.

5.2.3 Flip/Flop Selection

If the PCIEXDLP_SYNC_RESET configuration parameter is **not** defined (see *Section 5.1.5 Use of Synchronous or Asynchronous Set/Reset Flip-Flops* on page 73), this core requires flip-flops with the following characteristics:

- Rising-edge triggered.
- At least one flip-flop with an asynchronous reset control input.
- At least one flip-flop with an asynchronous set control input.
- If it exists in the library, at least one flip-flop without asynchronous set or reset control pins.
- All flip/flops must be capable of running with a clock frequency of 250 MHz.

The following flip-flops in the IBM CU11 ASIC library satisfy the above requirements:

- D_L_LPH4041_LPC (asynchronous reset)
- D_F_LPH2021_LPC (asynchronous set)
- D_F_LPH0001_LPC and/or D_F_LPH0001 (without asynchronous set or reset)

If the PCIEXDLP_SYNC_RESET configuration parameter is defined (see *Section 5.1.5 Use of Synchronous or Asynchronous Set/Reset Flip-Flops* on page 73), this core requires flip-flops with the following characteristics:

- Rising-edge triggered.
- At least one flip-flop without asynchronous set or reset control pins.
- All flip/flops must be capable of running with a clock frequency of 250 MHz.

The following flip-flops in the IBM CU11 ASIC library satisfy the above requirements:

- D_F_LPH0001_LPC and/or D_F_LPH0001 (without asynchronous set or reset)

5.3 Test Interface Requirements

This core contains no test signals or test logic.

5.4 Physical Design Floorplanning Guidelines

It is likely that the physical area required by the attached PHY will be significantly larger than the area required by this core. As a result, placement of this core will generally be determined by the location of the PHY's PIPE interface pin locations. A typical floor plan might look like the one shown in *Figure 49*.

Figure 49. Typical Physical Floor Plan