LogiCORE IP Endpoint PIPE v1.8 for PCI Express

DS321 July 23, 2010 **Product Specification**

Introduction

The LogiCORE™ IP Endpoint PIPE (PHY Interface) for PCI Express® 1-lane core is a high-bandwidth scalable and reliable serial interconnect intellectual property building block for use with the Spartan®-3, Spartan-3E, and Spartan-3A FPGAs in conjunction with an external PHY device. This solution, compliant with the *PCI Express Base Specification v1.1*, is a flexible low-cost chipset that can be used in a wide variety of high-volume applications including add-in cards, host bus adapters, and high-end server and graphics cards.

PCI Express offers a serial architecture that alleviates some of the limitations of parallel bus architectures by using clock data recovery (CDR) and differential signaling. Using CDR (as opposed to source synchronous clocking) lowers pin count, enables superior frequency scalability, and makes data synchronization easier. The layered architecture of PCI Express provides for future attachment to copper, optical, or emerging physical signaling media. PCI Express technology, adopted by the PCI-SIG as the next generation PCI, is backward-compatible to the existing PCI software model.

The Xilinx solutions for PCI Express set the industry standard for a high-performance and cost-efficient third-generation I/O solution by providing higher bandwidth per pin, low overhead, low latency, reduced signal integrity issues, and CDR architecture.

Xilinx Endpoint solutions for PCI Express are compatible with industry standard application form factors such as *PCI Express Card Electromechanical (CEM) v1.1* and *PCI Industrial Computer Manufacturers Group (PICMG) 3.4* specifications.

LogiCORE IP Facts Table			
Core Specifics			
Supported Device Family(1)	Spartan-3, Spartan-3E, Spartan-3A(2)		
Minimum Device Requirement	Spartan-3	XC3S1000-4	
	Spartan-3E	XC3S500E-4	
	Spartan-3A	XC3S700A-4	
	Resources(3)		
Configuration	**LUTs**	**FFs**	**Block RAMs(4)**
1-Lane Endpoint PIPE	5880-6150(5)	4650-4790(5)	8
Supported PHY	NXP PX1011B-EL1		
Provided with Core			
Documentation	Product Specification Getting Started Guide User Guide Instantiation Template		
Design Files	Netlist		
Example Design	Verilog		
Test Bench	Verilog		
Constraints File	Specify Xilinx Constraints File		
Simulation Model	Verilog and VHDL		
Tested Design Tools			
Design Entry Tools	Mentor Graphics® ModelSim® PE/SE v6.5c and above Synopsys® VCS and VCS MX 2009.12 and above Cadence® Incisive Enterprise Simulator (IES) v9.2 and above		
Simulation	Version of Simulator Tools Tested		
Synthesis Tools (Verilog only)	Synplicity® Synplify®, Xilinx XST		
Support			
Provided by Xilinx, Inc.			

1. For a complete listing of supported devices, see the release notes for this core.
2. Spartan-3AN is not a supported device family.
3. The precise number of slices depends on the user configuration of the interface and the level of resource sharing with adjacent logic.
4. Based on 18K block RAMs (or 36K - select appropriate size).
5. This range indicates resources used for a 2BAR–7BAR implementation.

Features

- High-performance, highly flexible, scalable, reliable, and general purpose I/O core
  - Compliant with the *PCI Express Base Specification v1.1*
  - Compatible with conventional PCI software model
- Fully compliant with PCI Express transaction ordering rules
- Six individually programmable/configurable BARs and expansion ROM BAR
- Supports MSI and INTX emulation
- 32-bit internal data path
- Supports removal of corrupt packets for error detection and recovery
- Compatible with PCI/PCI Express power management functions
  - Active state power management (ASPM)
  - Programmed power management (PPM)
- Used in conjunction with NXP PX1011B PCI Express standalone PHY to achieve high transceiver capability
  - 2.5 Gbps line speed
  - Elastic buffers and clock compensation
  - Automatic clock and data recovery
  - 8b/10b encode and decode
- Offers standardized easy-to-use Xilinx LocalLink interface
  - Packet-based full-duplex communication
  - Back-to-back transactions enable greater link bandwidth utilization
  - Enables flow control of data and discontinuance of an in-process transaction in the transmit direction
  - Enables flow control of data in the receive direction
  - Automatically decodes and removes error forwarding packet indicator from received data
- Supports a maximum transaction payload of up to 512 bytes
- Fully configurable using the Xilinx CORE Generator™ v12.2
- Design verified using a Xilinx proprietary test bench

Applications

The Endpoint PIPE for PCI Express core architecture enables a broad range of computing and consumer communications target applications, emphasizing performance, cost, scalability, feature extensibility and mission-critical reliability. Typical applications include:

- Test equipment
- Consumer graphics boards
- Medical imaging equipment
- Data communications networks
- Telecommunications networks
- Broadband deployments
- Cross-connects
- Workstation and mainframe backbones
- Network interface cards

- Chip-to-chip and backplane interconnect
- Crossbar switches
- Wireless base stations
- High-bandwidth digital video
- High-bandwidth server applications

Functional Description

The Endpoint PIPE for PCI Express is organized into four main modules based on the three discrete logical layers defined by the *PCI Express Base Specification v1.1*. The logic modules, which manage all the system-level functions, include the following:

- Physical Layer Module (PLM)
- Data Link Layer Module (LLM)
- Transaction Layer Module (TLM)
- Configuration Management Module (CMM)

Each modules is further partitioned into the Receive and the Transmit sections. The Receive section processes the inbound information, and the Transmit section processes the outbound information. Figure 1 illustrates the main modules interfacing with one another and with the User Application using a set of four interfaces:

- System (SYS) interface
- PIPE (PXPIPE) interface
- Configuration (CFG) interface
- Transaction (TRN) interface

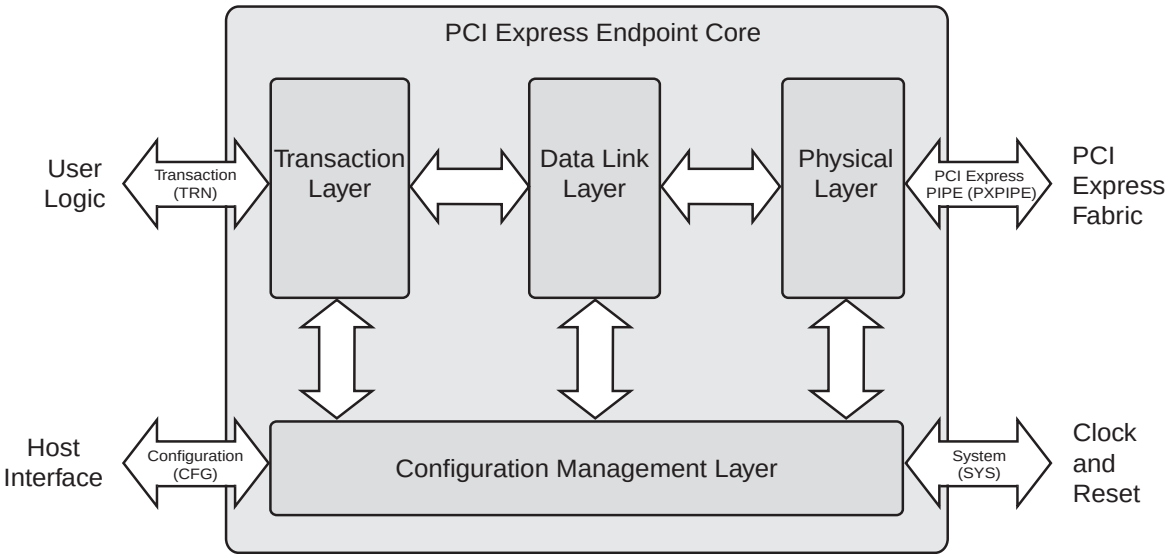

Figure 1: **Top-level Functional Blocks and Interfaces**

The core uses packets to exchange information between the various modules. Packets are formed in the Transaction and Data Link Layers to carry information from the transmitting component to the receiving component. Necessary information is added to the packet being transmitted, which is required to handle the packet at those layers. At the receiving end, each layer of the receiving element processes the incoming packet, strips the relevant information

and forwards the packet to the next layer. As a result, the received packets get transformed from their Physical Layer representation to their Data Link Layer representation and the Transaction Layer representation. The main logic modules comprising the Endpoint PIPE for PCI Express and their interfaces are described in the following sections.

Logic Modules

The logic modules are responsible for handling the functionality related to each of the layers defined by the *PCI Express Base Specification v1.1*. The functions of these modules include generation and processing of Transaction Layer Packets (TLPs), flow control management, initialization and power management functions, data protection, error checking and retry functions, physical link interface initialization, maintenance and status tracking, serialization, de-serialization and other circuitry for interface operation. These modules and their functionality are described below.

Physical Layer Module

The Physical Layer exchanges information with the Data Link Layer in an implementation-specific format. This layer is responsible for converting information received from the Data Link Layer into an appropriate format and transmitting it across the PXPIPE interface at a frequency and width compatible with the external PHY.

PXPIPE is the *NXP PHY Specification*, an extended version of the PIPE specification.

Data Link Layer Module

The Data Link Layer acts as an intermediate stage between the Transaction Layer and the Physical Layer. Its primary responsibility is to provide a reliable mechanism for the exchange of TLPs between the two Components on a Link.

Services provided by the Data Link Layer include data exchange (TLPs), error detection and recovery, initialization services and the generation and consumption of Data Link Layer Packets (DLLPs). DLLPs are the mechanism used to transfer information between Data Link Layers of two directly connected Components on the Link. They are used for conveying information such as Flow Control and TLP acknowledgments.

Transaction Layer Module

The upper layer of the PCI Express architecture is the Transaction Layer. The primary function of the Transaction Layer is to accept, buffer and disseminate TLPs. Packets are formed in the Transaction and Data Link Layers to carry the information from the transmitting component to the receiving component. TLPs are used to communicate transactions, such as read and write, as well as certain types of events. To maximize the efficiency of communication between devices, the Transaction Layer implements a pipelined, full split-transaction protocol; manages credit-based flow control of TLP which eliminates wasted Link bandwidth due to retries; and offers optional support for data poisoning.

Configuration Management Module

This module supports generation and reception of System Management Messages by communicating with the other modules and the user application.

The Configuration Management module does the following:

- Implements Configuration Space registers which support:
  - Legacy PCI2.3 configuration space
  - PCI Express Capabilities (minimum required set)
- Supports Configuration Space accesses

- Power Management Functions
  - Programmed Power Management (PPM)
  - Active State Power Management (ASPM)
- Implements error reporting and status functionality
  - Error tracking
  - Error reporting
- Implements packet processing functions
  - Receive
    - Configuration Reads and Writes
  - Transmit:
    - Completions with or without data
    - TLM Error Messaging
    - User Error Messaging
    - Power Management Messaging/Handshake
- Implements MSI and INTx interrupt emulation with support for Multi-Vector MSI

PCI Configuration Space

This block provides a standard Type 0 configuration space. The Configuration Space consists of a 64 byte, Type 0 configuration space header, with an additional 192 bytes used for extended capabilities.

The following extended capabilities are provided in the interface:

- Express Capability Item
- Power Management Capability Item
- Message Signaled Interrupt Capability Item
- Device serial number extended capability structure

These capabilities, together with the standard Type 0 header shown in Table 1, support software driven *Plug and Play* initialization and configuration.

Table 1: **Configuration Space Header for PCI Express**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Device ID																Vendor ID																000h
Status																Command																004h
Class Code																								Rev ID								008h
BIST								Header								Lat Timer								Cache Ln								00Ch
Base Address Register 0																																010h
Base Address Register 1																																014h
Base Address Register 2																																018h
Base Address Register 3																																01Ch
Base Address Register 4																																020h
Base Address Register 5																																024h
Cardbus CIS Pointer																																028h
Subsystem ID																Subsystem Vendor ID																02Ch
Expansion ROM Base Address																																030h
Reserved																								CapPtr								034h
Reserved																																038h
Max Lat								Min Gnt								Intr Pin								Intr Line								03Ch
PM Capability																NxtCap								PM Cap								040h
Data								BSE								PMCSR																044h
MSI Control																NxtCap								MSI Cap								048h
Message Address (Lower)																																04Ch
Message Address (Upper)																																050h
Reserved																Message Data																054h
PE Capability																NxtCap								PE Cap								058h
PCI Express Device Capabilities																																05Ch
Device Status																Device Control																060h
PCI Express Link Capabilities																																064h
Link Status																Link Control																068h
Reserved Legacy Configuration Space (Read Access: Returns successful CplD w/ Data = 0x0000_0000 Write Access: Returns successful Cpl.)																																06Ch-0FFh
Next Cap								Capability								PCI Exp. Capability																100h
PCI Express Device Serial Number (1st)																																104h
PCI Express Device Serial Number (2nd)																																108h
Reserved Extended Configuration Space (Returns Completion with UR)																																10Ch-FFFh

Core Interfaces

The Endpoint PIPE for PCI Express includes top-level signal interfaces that have sub-groups for the receive direction, transmit direction, and signals common to both directions.

System Interface

The system interface (SYS) consists of the system reset signal, as described in Table 2.

Table 2: **System Interface Signals**

Signal Name	Direction	Description
sys_reset_n	Input	**System Reset:** An asynchronous, active low, reset signal from the root complex/system that puts the endpoint in a known initial state.

The system reset signal is active low. The assertion of this signal causes a hard reset of the entire core, including the NXP PHY. This signal is an asynchronous input. In typical endpoint applications, a sideband reset signal is present and should be connection to `sys_reset_n`. For endpoint applications that do not have a sideband system reset signal, the initial hardware reset should be generated locally. In either case, subsequent resets may be communicated in-band using the PCI Express protocol.

Note that systems designed to the PCI Express electro-mechanical specification provide a sideband reset signal that uses 3.3V signaling levels—carefully read the FPGA device data sheet to understand the requirements for interfacing to such signals.

NXP Standalone PHY

The Endpoint PIPE for PCIe uses an external NXP PX1011B-EL1. The PIPE architecture is defined by Intel Corporation and is used as the interconnect between the Endpoint for PCI Express core and the NXP PHY. The NXP PHY (PXPIPE) is an extended version of the PIPE specification. Using another PHY with PIPE interface is incompatible. See the NXP PX1011B data sheet for more information and errata items related to the NXP PHY. For additional information, go to ics.nxp.com/products/pcie/phys/.

PHY Pin Description

The PXPIPE input and output signals are described in Table 3 through Table 7. Note that input/output is defined from the perspective of the FPGA—Output signals are *driven* by the FPGA, and Input signals are *received* by the FPGA.

Table 3: **PXPIPE Transmit Data Interface Signals**

Symbol	Direction	Description
TXDATA[7:0]	Output	8-bit transmit data from the FPGA to the NXP PHY.
TXDATAK	Output	Data/Control for the symbols of transmit data. A value of 0 indicates a data byte, a value of 1 indicates a control byte.

Table 4: **PXPIPE Receive Data Interface Signals**

Symbol	Direction	Description
RXDATA[7:0]	Input	8-bit receive data from the NXP PHY to the FPGA.
RXDATAK	Input	Data/Control for the symbols of receive data. A value of 0 indicates a data byte, a value of 1 indicates a control byte.

Table 5: **Clock and Reference Signals**

Symbol	Direction	Description
TXCLK	Output	Source synchronous 250 MHz clock (from FPGA) for transmit clock from MAC input. All the data and the input signals to the PHY are synchronized to this clock.
RXCLK	Input	Source synchronous 250 MHz clock (to FPGA) for received data bound for the MAC output.
fast_train_simulation_only	Input	Should only be asserted for simulation. Training counters are lowered when this input is asserted (set to “1”) to allow the simulation to train faster. Do not assert this input when using the core in hardware. Doing so causes the core to fail link training.
two_plm_auto_config	Input	Used only for simulation; forces device to act as a down stream device for link training purposes. This signal should only be set to “1,” if in simulation two Endpoint PIPE cores are connected back-to-back. When using the provided downstream port model for simulation, this signal should be set to “0” on the endpoint core under test. This signal should be set to “0” when using the core in hardware as an endpoint device. Any other setting in hardware is not supported.

Table 6: **PXPIPE Command Interface Signals**

Symbol	Direction	Description
TXDETECTRX_ LOOPBACK	Output	Enable the NXP PHY to begin a receiver detection operation or to begin loopback.
TXELECIDLE	Output	Forces NXP PHY Tx output to electrical idle when asserted in all power states. When deasserted while in P0 (as indicated by the PowerDown signals), indicates that there is valid data present on the TXDATA[7:0] and TXDATAK pins and that the data should be transmitted. When deasserted in P2 the signal has no function. It would be used to indicate that the PHY should begin transmitting beacon signaling however beacon is not supported in this version of the PHY. TXELECIDLE must always be asserted while in power states P0s and P1.
TXCOMPLIANCE	Output	When high, sets the running disparity to negative. Used when transmitting the compliance pattern.
RXPOLARITY	Output	Active high, signals the PHY to perform a polarity inversion on the receive data.
RESETN	Output	Active low PHY reset from FPGA.
POWERDOWN[1:0]	Output	Power up or down the transceiver. Power states: • 00 - P0, normal operation • 01 - P0s, low recovery time latency, power saving state • 10 - P1, longer recovery time (64us max) latency, lower power state • 11 - Reserved for P2, lowest power state

Table 7: **PXPIPE Status Interface Signals**

Symbol	Direction	Description
RXVALID	Input	Indicates symbol lock and valid data on RxData and RxDataK.
PHYSTATUS	Input	Used to communicate completion of several NXP PHY functions including power management state transitions, and receiver detection.

Table 7: **PXPIPE Status Interface Signals *(Cont'd)***

Symbol	Direction	Description
RXELECIDLE	Input	Indicates receiver detection of an electrical idle. This is an asynchronous signal.
RXSTATUS[2:0]	Input	Encodes receiver status and error codes for the received data stream and receiver detection. • 000 - Received data OK • 001 - 1 SKP added • 010 - 1 SKP removed • 011 - Receiver detected • 100 - 8B/10B decode error • 101 - Elastic Buffer overflow • 110 - Elastic Buffer underflow • 111 - Receive disparity error

Transaction Interface

The Transaction (TRN) interface provides a mechanism for the user design to generate and consume TLPs. The signal names and signal descriptions for this interface are shown in Table 8, Table 9, and Table 11.

Common TRN Interface

Table 8 lists the common TRN interface signal names and descriptions.

Table 8: **Common Transaction Interface Signals**

Name	Direction	Description
trn_clk	Output	**Transaction Clock**: 62.50 MHz. Transaction and Configuration interface operations are referenced-to and synchronous-with the rising edge of this clock. trn_clk is unavailable when the core sys_reset_n is held asserted. trn_clk is guaranteed to be stable at the nominal operating frequency once the core deasserts trn_reset_n.
trn_reset_n	Output	**Transaction Reset**: Active low. User logic interacting with the Transaction and Configuration interfaces must use trn_reset_n to return to their quiescent states. trn_reset_n is deasserted synchronously with respect to trn_clk, sys_reset_n is deasserted and is asserted asynchronously with sys_reset_n assertion. Note that trn_reset_n is not asserted for the core in-band reset events like *Hot Reset* or *Link Disable*. trn_reset_n is asserted due to loss of input receive clock from the external PHY device, which is used by the core's internal DCM, to generate trn_clk.
trn_lnk_up_n	Output	**Transaction Link Up**: Active low. Transaction link-up is asserted when the core and the connected upstream link partner port are ready and able to exchange data packets. Transaction link-up is deasserted when the core and link partner are attempting to establish communication, and when communication with the link partner is lost due to errors on the transmission channel. When the core is driven to *Hot Reset* and *Link Disable* states by the link partner, trn_lnk_up_n is deasserted and all TLPs stored in the core are lost.

Transmit TRN Interface

Table 9 lists the transmit (Tx) TRN interface signal names and descriptions.

Table 9: **Transmit Transaction Interface Signals**

Name	Direction	Description
trn_tsof_n	Input	**Transmit Start-of-Frame** (SOF): Signals the start of a packet. Active low.
trn_teof_n	Input	**Transmit End-of-Frame** (EOF): Signals the end of a packet. Active low.
trn_td[31:0]	Input	**Transmit Data:** Packet data to be transmitted.

Table 9: **Transmit Transaction Interface Signals**

Name	Direction	Description
trn_terrfwd_n	Input	**Transmit Error Forward**: Marks the current packet in progress as error-poisoned. May be asserted any time between SOF and EOF, inclusive. Active low.
trn_tsrc_rdy_n	Input	**Transmit Source Ready**: Indicates that the User Application is presenting valid data on **trn_td[31:0]**. Active low.
trn_tdst_rdy_n	Output	**Transmit Destination Ready**: Indicates that the core is ready to accept data on trn_td[31:0]. Active low. The simultaneous assertion of trn_tsrc_rdy_n and trn_tdst_rdy_n marks the successful transfer of one DWORD of data on trn_td[31:0].
trn_tsrc_dsc_n	Input	**Transmit Source Discontinue**: Indicates that the User Application is aborting the current packet. Active low.
trn_tdst_dsc_n	Output	**Transmit Destination Discontinue**: Indicates that the core is aborting the current packet. Asserted when the physical link is going into reset. Active low.
trn_tbuf_av[4:0]	Output	**Transmit Buffers Available**: Number of transmit buffers available in the core. The maximum number is 8. Each transmit buffer can hold one packet with up to 512 bytes of payload.

Figure 2 illustrates the transfer on the TRN interface of two TLPs to be transmitted on the PCI Express Link. Every valid transfer can be up to one double word (DWORD) of data.

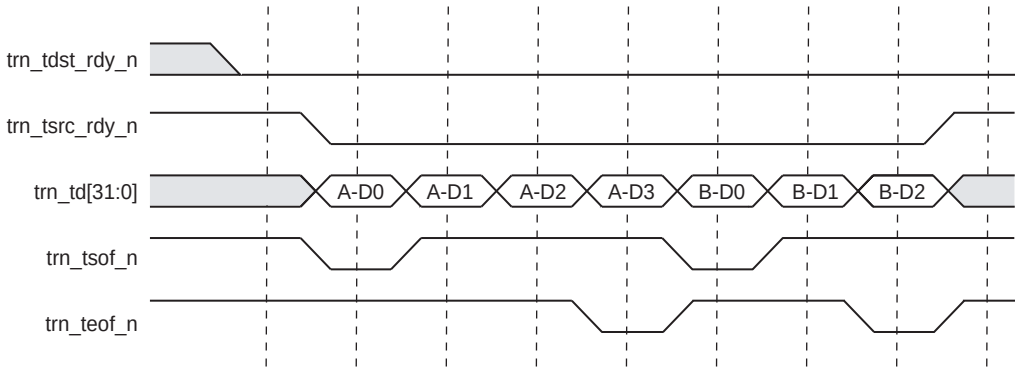

Figure 2: **Tx TRN Interface**

Table 10 lists the transmit path clock cycle signals and event descriptions.

Table 10: **Transmit Path Signaling Events**

Clock Cycle	Event Description
1	The assertion of trn_tdst_rdy_n tells the user application that the core is ready to accept data.
2	The user application initiates the transfer with the assertion of trn_tsrc_rdy_n and trn_tsof_n. The combined assertion of trn_tsrc_rdy_n and trn_tdst_rdy_n marks a data transfer. Frame A DWORD D0 is transferred.
3	Frame A DWORD D1 is transferred.
4	Frame A DWORD D2 is transferred.
5	The end of Frame A is signaled with trn_teof_n. Frame A DWORD D3 is transferred.
6	The user application signals the start of a new TLP with trn_tsof_n. Frame B DWORD D0 is transferred.
7	Frame B DWORD D1 is transferred.

Table 10: **Transmit Path Signaling Events**

Clock Cycle	Event Description
8	The end of Frame B is signaled with trn_teof_n. Frame B DWORD D2 is transferred.
9	The core keeps trn_tdst_rdy_n asserted to offer the user application the opportunity to start the transmission of the next TLP.

Receive TRN Interface

Table 11 lists the receive (Rx) TRN interface signal names and descriptions.

Table 11: **Receive Transaction Interface Signals**

Name	Direction	Description
trn_rsof_n	Output	**Receive Start-of-Frame** (**SOF**): Signals the start of a packet. Active low.
trn_reof_n	Output	**Receive End-of-Frame** (**EOF**): Signals the end of a packet. Active low.
trn_rd[31:0]	Output	**Receive Data:** Packet data being received.
trn_rerrfwd_n	Output	**Receive Error Forward:** Marks the current packet in progress as error-poisoned. Asserted by the core at EOF. Active low.
trn_rsrc_rdy_n	Output	**Receive Source Ready:** Indicates that the core is presenting valid data on trn_rd[31:0]. Active low.
trn_rdst_rdy_n	Input	**Receive Destination Ready:** Indicates that the User Application is ready to accept data on trn_rd[31:0]. Active low. The simultaneous assertion of trn_rsrc_rdy_n and trn_rdst_rdy_n marks the successful transfer of one DWORD of data on trn_rd[31:0].
trn_rsrc_dsc_n	Output	**Receive Source Discontinue:** Indicates that the core is aborting the current packet. Asserted when the physical link is going into reset. Active low.
trn_rnp_ok_n	Input	**Receive Non-Posted OK:** The User Application asserts this whenever it is ready to accept a Non-Posted Request packet. This allows Posted and Completion packets to bypass Non-Posted packets in the inbound queue if necessitated by the User Application. Active low. When the User Application approaches a state where it is unable to service Non-Posted Requests, it must deassert trn_rnp_ok_n after SOF of the second-to-last Non-Posted packet it can accept.
trn_rbar_hit_n[6:0]	Output	**Receive BAR Hit:** Indicates BAR(s) targeted by the current receive transaction. Active low. trn_rbar_hit_n[0] => BAR0 trn_rbar_hit_n[1] => BAR1 trn_rbar_hit_n[2] => BAR2 trn_rbar_hit_n[3] => BAR3 trn_rbar_hit_n[4] => BAR4 trn_rbar_hit_n[5] => BAR5 trn_rbar_hit_n[6] => Expansion ROM Address Note that if two BARs are configured into a single 64-bit address, both corresponding trn_rbar_hit_n bits are asserted.
trn_rfc_ph_av[7:0]	Output	**Receive Posted Header Flow Control Credits Available:** The number of Posted Header FC credits available to the remote link partner.(1)
trn_rfc_pd_av[11:0]	Output	**Receive Posted Data Flow Control Credits Available:** The number of Posted Data FC credits available to the remote link partner.()
trn_rfc_nph_av[7:0]	Output	**Receive Non-Posted Header Flow Control Credits Available:** The number of Non-Posted Header FC credits available to the remote link partner.()
trn_rfc_npd_av[11:0]	Output	**Receive Non-Posted Data Flow Control Credits Available:** The number of Non-Posted Data FC credits available to the remote link partner.()

Table 11: **Receive Transaction Interface Signals *(Cont'd)***

Name	Direction	Description
trn_rfc_cplh_av[7:0]	Output	**Receive Completion Header Flow Control Credits Available:** The number of Completion Header FC credits available to the remote link partner.(2) Note that this value and trn_rfc_cpld_av[11:0] are hypothetical quantities reflecting credit availability that would be advertised to the remote link partner if the core were not required to advertise infinite Completion credits.
trn_rfc_cpld_av[11:0]	Output	**Receive Completion Data Flow Control Credits Available:** The number of Completion Data FC credits available to the remote link partner.()

1. Credit values given to the user are instantaneous quantities, not the cumulative (from time zero) values seen by the remote link partner.
2. Completion credit values given to the user reflect the actual state of the core's receive FIFO, even though the *Base Specification v1.1* requires an endpoint to advertise infinite completion credits to its link partner regardless of actual receive capacity.

Figure 3 illustrates the transfer on the TRN interface of two TLPs received on the PCI Express Link.

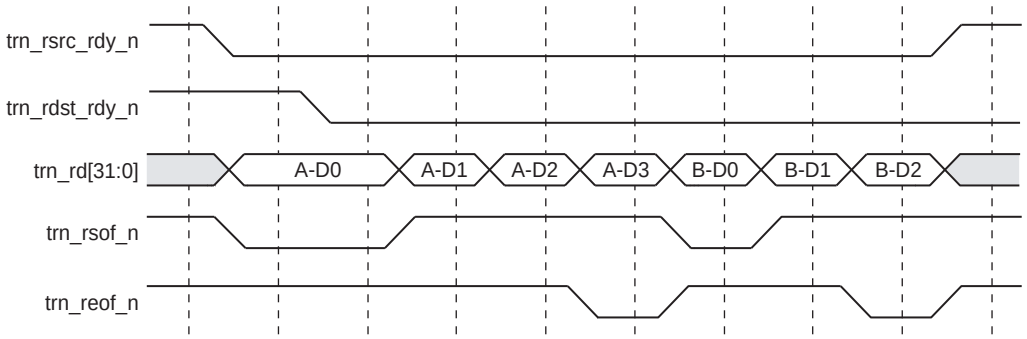

Figure 3: **Rx TRN Interface**

Table 12 lists the receive path clock cycle signals and descriptions.

Table 12: **Receive Path Signaling Events**

Clock Cycle	Event Description
1	The core signals by the assertion of trn_rsrc_rdy_n that a valid TLP has been entirely received from the link. The core provides the first DWORD of the TLP and asserts trn_rsof_n to mark the start of the frame.
2	The assertion of trn_rdst_rdy_n alerts the core that the user application is ready to accept data. The combined assertion of trn_rsrc_rdy_n and trn_rdst_rdy_n marks a data transfer. Frame A DWORD D0 is transferred.
3	Frame A DWORD D1 is transferred.
4	Frame A DWORD D2 is transferred.
5	The end of Frame A is signaled with trn_reof_n. Frame A DWORD D3 is transferred.
6	The core signals the start of the second frame with trn_rsof_n. Frame B DWORD D0 is transferred.
7	Frame B DWORD D1 is transferred.
8	The end of Frame B is signaled with trn_reof_n. Frame B DWORD D2 is transferred.
9	The core deasserts its trn_rsrc_rdy_n signal to indicate that there are no more pending TLPs to transfer.

Configuration Interface

The configuration (CFG) interface enables the user design to inspect the state of the Endpoint configuration space. The user provides a 10-bit configuration address which selects one of the 1024 configuration space double word

(DWORD) registers. The endpoint returns the state of the selected register over the 32-bit data output port. Table 13 describes the configuration interface signals.

Table 13: **Configuration Interface Signals**

Name	Direction	Description
cfg_do[31:0]	Output	**Configuration Data Out**: This is a 32-bit data output port used to obtain read data from the configuration space inside the endpoint core.
cfg_rd_wr_done_n	Output	**Configuration Read Write Done:** This active-low read-write done signal indicates a successful completion of the user configuration register access operation. For a user configuration register read operation, the signal validates the cfg_do[31:0] data-bus value. For a user configuration register write operation, the assertion signals completion of a successful write operation. Not supported for write operations.
cfg_di[31:0]	Input	**Configuration Data In**: This is a 32-bit data input port used to provide write data to the configuration space inside the core. Not supported.
cfg_dwaddr[9:0]	Input	**Configuration DWORD Address**: This is a 10-bit address input port used to provide a configuration register DWORD address during configuration register accesses.
cfg_wr_en_n	Input	**Configuration Write Enable**: This is the active low write enable for configuration register access. Not supported.
cfg_rd_en_n	Input	**Input Description:** Configuration Read Enable: This is the active low read enable for configuration register access.
cfg_interrupt_n	Input	**Configuration Interrupt**: Active-low interrupt-request signal. The User Application may assert this to cause selected interrupt message-type to be transmitted by the core. The signal should be held low until cfg_interrupt_rdy_n is asserted.
cfg_interrupt_rdy_n	Output	**Configuration Interrupt Ready:** Active-low interrupt grant signal. The simultaneous assertion of cfg_interrupt_rdy_n and cfg_interrupt_n indicates that the core has successfully transmitted the requested interrupt message.
cfg_interrupt_mmenable[2:0]	Output	**Configuration Interrupt Multiple Message Enable:** This is the value of the Multiple Message Enable field. Values range from 000b to 101b. A value of 000b indicates that single vector MSI is enabled, while other values indicate the number of bits that may be used for multi-vector MSI.
cfg_interrupt_msienable	Output	**Configuration Interrupt MSI Enabled:** Indicates that the Message Signaling Interrupt (MSI) messaging is enabled. If 0, then only Legacy (INTx) interrupts may be sent.
cfg_interrupt_di[7:0]	Input	**Configuration Interrupt Data In**: For Message Signaling Interrupts (MSI), the portion of the Message Data that the Endpoint must drive to indicate MSI vector number, if Multi-Vector Interrupts are enabled. The value indicated by cfg_interrupt_mmenable[2:0] determines the number of lower-order bits of Message Data that the Endpoint provides; the remaining upper bits of cfg_interrupt_di[7:0] are not used. For Single-Vector Interrupts, cfg_interrupt_di[7:0] is not used. For Legacy interrupt messages (Assert_INTx, Deassert_INTx), this indicates the message type to send, where: **Value** — **Legacy Interrupt** 00h — INTA 01h — INTB 02h — INTC 03h — INTD

Table 13: **Configuration Interface Signals *(Cont'd)***

Name	Direction	Description
cfg_interrupt_do[7:0]	Output	**Configuration Interrupt Data Out:** This is the value of the lowest 8 bits of the Message Data field in the Endpoint's MSI capability structure. This value is used in conjunction with cfg_interrupt_mmenable[2:0] to drive cfg_interrupt_di[7:0].
cfg_interrupt_assert_n	Input	**Configuration Legacy Interrupt Assert/Deassert Select:** Selects between Assert and Deassert messages for Legacy interrupts when cfg_interrupt_n is asserted. Not used for MSI interrupts. **ValueMessage Type** 0Assert 1Deassert
cfg_turnoff_ok_n	Input	**Configuration Turnoff OK**: This is the active low power turn-off ready signal. The User Application may assert this to notify the core that it is safe for power to be removed.
cfg_to_turnoff_n	Output	**Configuration To Turnoff**: This output signal notifies the user that a PME_TURN_Off message has been received and the CMM starts polling the cfg_turnoff_ok_n input coming in from the user. Once cfg_turnoff_ok_n is asserted, CMM sends a PME_To_Ack message to the upstream device.
cfg_byte_en_n[3:0]	Input	**Configuration Byte Enable**: This is the active low byte enables for configuration register access signal. Not supported.
cfg_err_ecrc_n	Input	**ECRC Error Report**: The user can assert this signal to report an ECRC error (end-to-end CRC).
cfg_err_cpl_timeout_n	Input	**Configuration Error Completion Timeout**: The user can assert this signal to report a completion timed out.
cfg_err_cpl_abort_n	Input	**Configuration Error Completion Aborted**: The user can assert this signal to report that a completion was aborted.
cfg_err_cpl_unexpect_n	Input	**Configuration Error Completion Unexpected**: The user can assert this signal to report that an unexpected completion was received.
cfg_err_posted_n	Input	**Configuration Error Posted**: This signal is used to further qualify any of the cfg_err_* input signals. When this input is asserted concurrently with one of the other signals, it indicates that the transaction which caused the error was a posted transaction.
cfg_err_cor_n	Input	**Configuration Error Correctable Error**: The user can assert this signal to report that a correctable error was detected.
cfg_err_ur_n	Input	**Configuration Error Unsupported Request**: The user can assert this signal to report that an unsupported request was received.
cfg_err_tlp_cpl_header[47:0]	Input	**Configuration Error TLP Completion Header**: This input to the core accepts the header information from the user when an error is signaled. This information is required so that the core can issue a correct completion, if required.
cfg_bus_number[7:0]	Output	**Configuration Bus Number**: This output provides the assigned bus number for the device. The User Application must use this information in the Bus Number field of outgoing TLP requests. Default value after reset is 00h. Refreshed whenever a Type 0 Configuration packet is received.
cfg_device_number[4:0]	Output	**Configuration Device Number**: This output provides the assigned device number for the device. The User Application must use this information in the Device Number field of outgoing TLP requests. Default value after reset is 00000b. Refreshed whenever a Type 0 Configuration packet is received.

Table 13: **Configuration Interface Signals *(Cont'd)***

Name	Direction	Description
cfg_function_number[2:0]	Output	**Configuration Function Number**: Provides the function number for the device. The User Application must use this information in the Function Number field of outgoing TLP request. Function number is hard-wired to 000b.
cfg_status[15:0]	Output	**Configuration Status**: Status register from the Configuration Space Header.
cfg_command[15:0]	Output	**Configuration Command**: Command register from the Configuration Space Header.
cfg_dstatus[15:0]	Output	**Configuration Device Status**: Device status register from the PCI Express Extended Capability Structure.
cfg_dcommand[15:0]	Output	**Configuration Device Command**: Device control register from the PCI Express Extended Capability Structure.
cfg_lstatus[15:0]	Output	**Configuration Link Status**: Link status register from the PCI Express Extended Capability Structure.
cfg_lcommand[15:0]	Output	**Configuration Link Command**: Link control register from the PCI Express Extended Capability Structure.
cfg_pm_wake_n	Input	**Configuration Power Management Wake**: A one-clock cycle active low assertion on this signal enables the core to generate and send a Power Management Wake event to the upstream link partner. **NOTE**: The user is required to assert this input only under stable link conditions as reported on the cfg_pcie_link_state[2:0] bus. Assertion of this signal when the PCI Express Link is under transition results in incorrect behavior on the PCI Express Link.
cfg_trn_pending_n	Input	**User Transaction Pending**: When asserted, sets the Transaction Pending bit in the Device Status Register. User is required to assert this input if the user application has not received a completion to an upstream request. Active low.
cfg_dsn[63:0]	Input	**Configuration Device Serial Number**: Serial Number Register fields of the PCI Express Device Serial Number extended capability.
cfg_pcie_link_state_n[2:0]	Output	**PCI Express Link State**: This one-hot encoded bus reports the PCI Express Link State Information to the user. 110b - PCI Express Link State is "L0" 101b - PCI Express Link State is "L0s" 011b - PCI Express Link State is "L1" 111b - PCI Express Link State is "under transition"

Core Verification

The Endpoint PIPE for PCI Express is verified using a Xilinx proprietary test bench.

Ordering Information

This core is provided under the SignOnce IP Site License and can be generated using the Xilinx CORE Generator system v12.2 or higher. The CORE Generator system is shipped with Xilinx ISE® Foundation Series Development software.

A simulation evaluation license for the core is shipped with the CORE Generator system. To access the full functionality of the core, including FPGA bitstream generation, a full license must be obtained from Xilinx. For more information, please visit the Endpoint PIPE for PCI Express.

Please contact your local Xilinx sales representative for pricing and availability of additional Xilinx LogiCORE blocks and software. Information about additional Xilinx LogiCORE modules is available on the Xilinx IP Center.

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
4/11/05	1.1	Initial Xilinx release.
6/10/05	1.1.5	Updated data sheet with editorial changes.
8/31/05	1.2	Updated for delivery through CORE Generator v7.1i SP3.
1/18/06	1.3	Updated for delivery through CORE Generator v8.1i.
4/1/06	1.3.5	Updated Facts table: Resource usage and Block RAM.
7/13/06	1.4	Updated core to version 1.4; Xilinx tools 8.2i.
9/21/06	1.5	Updated core to version 1.5.
2/15/07	1.6	Updated core version to 1.6; Xilinx tools 9.1i.
5/17/07	1.7	Updated core version to 1.7; updated for PCI-SIG compliance.
7/23/10	2.0	Updated core version to 1.8 and ISE to 12.2.

Notice of Disclaimer