LogiCORE IP Endpoint v3.7 for PCI Express®

DS506 April 19, 2010 **Product Specification**

Introduction

The LogiCORE™ Endpoint for PCI Express® offers high-bandwidth, scalable, and reliable serial interconnect intellectual property building blocks for use with Virtex®-4, and Virtex-5 devices. All cores in the solution (1-lane, 4-lane, and 8-lane) are protocol-compliant and electrically compatible with the *PCI Express Base Specification v1.1*.

PCI Express (PCIe®) 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 PCIe provides for future attachment to copper, optical, or emerging physical signaling media. PCIe technology, adopted by PCI-SIG as the next generation PCI, is backward-compatible to the existing PCI software model.

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

The Xilinx Endpoint solution for PCIe is compatible with industry-standard application form factors such as the *PCI Express Card Electromechanical (CEM) v1.1* and the *PCI Industrial Computer Manufacturers Group (PICMG) 3.4* specifications.

The Endpoint solutions for PCIe are defined in the following table.

Product	FPGA Support	Data Path Width
1-lane 64-bit Endpoint	Virtex-4 FX	64
4-lane 64-bit Endpoint	Virtex-4 FX	64
8-lane 64-bit Endpoint	Virtex-5 LX, Virtex-4 FX	64
1-lane 32-bit Endpoint	Virtex-5 LX, Virtex-4 FX	32
4-lane 32-bit Endpoint	Virtex-5 LX, Virtex-4 FX	32

LogiCORE Facts

Core Specifics

Device Families[1]	Virtex-4 FX[2], Virtex-5 LX[2],[3]			
Minimum Device Requirement	1-lane 64-bit Endpoint	XC4VFX20 -10		
	4-lane 64-bit Endpoint	XC4VFX20 -10		
	8-lane 64-bit Endpoint	XC4VFX60 -11/XC5VLX50T-1		
	1-lane 32-bit Endpoint	XC4VFX20 -10/XC5VLX50T-1		
	4-lane 32-bit Endpoint	XC4VFX20 -11/XC5VLX50T-1		
Resources Used	**Endpoint (EP) Product**	**I/O[4]**	**LUT**	**FF**
	1-lane 64-bit Endpoint	1[5]	7800	7000
	4-lane 64-bit Endpoint	4	10550	9200
	8-lane 64-bit EP Virtex-4 8-lane 64-bit EP Virtex-5	8	13500 11400	12000 11200
	1-lane 32-bit EP Virtex-4 1-lane 32-bit EP Virtex-5	1[5]	6300 5100	5100 4700
	4-lane 32-bit EP Virtex-4 4-lane 32-bit EP Virtex-5	4	8700 7200	7000 6700
		Block RAM	**CMPS[6] # Tx Buffers**	**CMPS**
	1-lane 64-bit Endpoint	12	16	512
	4-lane 64-bit Endpoint	12	16	512
	8-lane 64-bit EP Virtex-4 8-lane 64-bit EP Virtex-5	12 6	32	256
	1-lane 32-bit EP Virtex-4 1-lane 32-bit EP Virtex-5	8 4	8	512
	4-lane 32-bit EP Virtex-4 4-lane 32-bit EP Virtex-5	12 6	16	512
Special Features	Rocket IO™ Transceivers[4], Digital Clock Manager, block RAM			

Provided with Core

Documentation	Product Specification, Getting Started Guide User Guide, Instantiation Template
Design Files	Verilog® and VHDL Simulation Models Xilinx Generic Netlist Format (ngo netlist) Verilog Example Test Bench & Example Design
Constraints File	User Constraints File (UCF)

Design Tool Support

HDL Synthesis Tool	Synplicity® Synplify®, Xilinx XST
Implementation Tools	Xilinx ISE® v12.1
Verification Tools (SWIFT-compliant simulator required)	Cadence™ Incisive Enterprise Simulator (IES) v9.2 and above, Synopsys® VCS and VCS MX 2009.12 and above , Mentor Graphics® ModelSim® v6.5c and above

Support

Provided by Xilinx, Inc. @ **www.xilinx.com/support**

1. For the complete list of supported devices, see the 12.1 release notes for this core.
2. Virtex-4 and Virtex-5 solutions require the latest production silicon stepping and are pending hardware validation. The Xilinx LogiCORE warranty does not include production usage with engineering sample silicon (ES).
3. XC5VLX50T, XC5VLX110T, and XC5VLX330T are supported.
4. RocketIO Multi-Gigabit Transceiver (MGT) for Virtex-4, RocketIO GTP Transceiver for Virtex-5.
5. In Virtex-4 and Virtex-5 devices, the 1-lane Endpoint core consumes an entire RocketIO transceiver pair. One RocketIO transceiver tile is used for lane 0; the other is unused and tied off inside the core.
6. CMPS: Capability Maximum Payload Size.

Features

- High-performance, highly flexible, scalable, and reliable, general purpose I/O solution
  - Compliant with the *PCI Express Base Specification v1.1*
  - Compatible with conventional PCI software model
- Incorporates Xilinx Smart-IP™ technology to guarantee critical timing
- Uses embedded RocketIO™ transceivers to achieve high-transceiver capability
  - 2.5 Gbps line speed
  - Supports 1-lane, 4-lane, and 8-lane operation
  - Elastic buffers and clock compensation
  - Automatic clock data recovery
- 8B/10B encode and decode
- Offers standardized user interface
  - Easy-to-use packet-based protocol
  - Full-duplex communication
  - Back-to-back transactions enable greater link bandwidth utilization
  - Supports flow control of data and discontinuation of an in-process transaction in transmit direction
  - Supports flow control of data in receive direction
  - Support for automatic handling of error forwarded packets
- Supports removal of corrupted packets for error detection and recovery
- Compliant with PCI/PCI-Express power management functions
- Supports a maximum transaction payload of up to 512 bytes
- Bandwidth scalability with frequency and/or interconnect width
- Fully compliant with PCI Express transaction ordering rules
- Design verified using a Xilinx proprietary test bench

Applications

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

- Data communications networks
- Telecommunications networks
- Broadband wired and wireless applications
- Cross-connects
- Network interface cards
- Chip-to-chip and backplane interconnect
- Crossbar switches
- Wireless base stations

Functional Description

The Endpoint cores for PCI Express are organized into four main modules based on the three discrete logical layers defined by the *PCI Express Base Specification*. The four 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 module 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 the user application using the following set of four interfaces:

- System interface (SYS)
- PCI Express interface (PCI EXP)
- Configuration interface (CFG)
- Transaction interface (TRN)

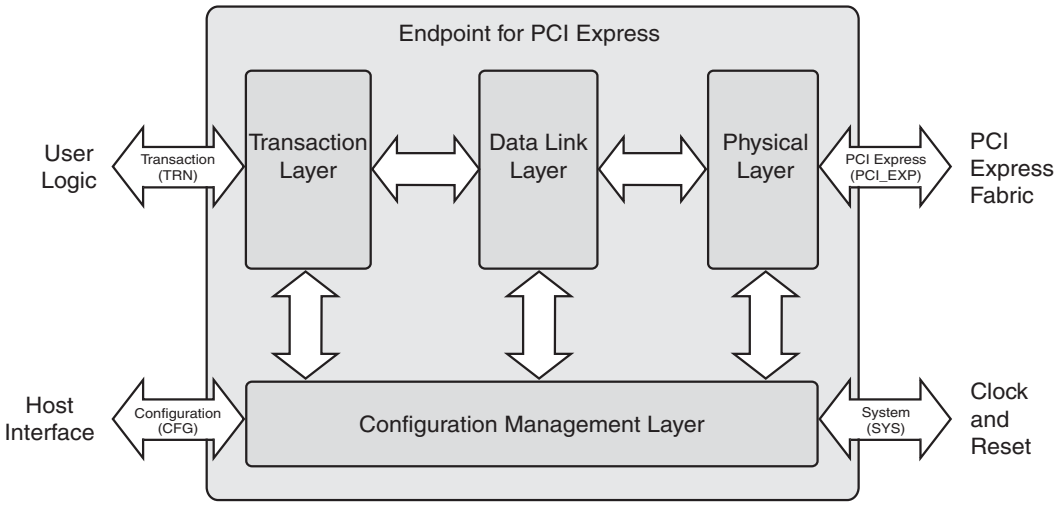

Figure 1: **Endpoint Top-level Functional Blocks and Interfaces**

The core allows the use of packets to exchange information between 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 specific 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 are transformed from their Physical Layer representation to their Data Link Layer representation and Transaction Layer representation.

The primary logic modules comprising the Endpoint core for PCI Express and their interfaces are described in the sections that follow.

Logic Modules

The logic modules handle the functionality related to each of the layers defined by the *PCI Express Base Specification*. The functions of these modules include

- Generating and processing of 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

Each of the logic models, introduced in the "Functional Description," page 3, are defined in the following sections.

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 serialized format and transmitting it across the PCI Express Link at a frequency and width compatible with the remote device.

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 Transaction Layer Packets (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 goal of the Transaction Layer is the assembly and disassembly of Transaction Layer Packets (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 TLPs, and offers optional support for data poisoning.

Configuration Management Module

The Configuration Management Module supports generation and reception of System Management Messages by communicating with the other modules and the user application. This module contains the device configuration space and other system functions. The CMM implements PCI/PCI-Express power management capabilities, and facilitates exchange of power management messages, including

support for PME event generation. Also implemented are user-triggered error message generation and user-read access to the device configuration space.

PCI Configuration Space

The PCI Configuration Space block provides a standard Type 0 configuration space, consisting of a 64-byte, Type 0 configuration space header with an additional 192 bytes used for extended capabilities. Four extended capabilities are provided in the interface:

- Express capability structure
- Power management capability structure
- Message signaled interrupt capability structure
- 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: **PCI Configuration Space Header**

31	24	23	16	15	8	7	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 (Returns 0x00000000)								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

Endpoint Interfaces

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

System Interface

Table 2 defines the System (SYS) interface signals. The system reset (sys_reset_n) signal is an asynchronous input (active low). The assertion of this signal causes a hard reset of the entire endpoint, including the Rocket IO transceiver. In the CEM add-in card form factor, the PERST# signal should be connected to the sys_reset_n signal. For form factors where no sideband reset is available, it must be generated locally.

The system clock signal (sys_clk) is used to clock the entire endpoint, including the Rocket IO transceiver. The system clock is used to clock logic that coordinates the hardware reset process. This clock must be a free-running clock that is not a DCM output. See the "*Digital Design Considerations*" chapter of the *Rocket I/O Transceiver User Guide* for more information. Additional information about core clocking consideration can be found in Answer Record 20600.

Table 2: **System Interface Signals**

Name	Direction	Description
sys_reset_n	Input	**System Reset**: An asynchronous input (active low) signal reset from the root complex/system that puts the endpoint in a known initial state.
sys_clk	Input	**Reference Clock**: The reference clock for the PCI Express Endpoint solutions. **Product** / **Virtex-4** / **Virtex-5** 1-lane 64-bit Endpoint / 250 MHz / N/A 4-lane 64-bit Endpoint / 250 MHz / N/A 8-lane 32-bit Endpoint / 250 MHz / 100 MHz 1-lane 32-bit Endpoint / 250 MHz / 100 MHz 4-lane 32-bit Endpoint / 250 MHz / 100 MHz

PCI Express Interface

The PCI Express (PCI_EXP) interface consists of differential transmit and receive pairs organized in multiple lanes. A PCI Express lane consists of a pair of transmit differential signals (pci_exp_txp, pci_exp_txn) and a pair of receive differential signals (pci_exp_rxp, pci_exp_rxn). The 1-lane core supports only lane 0, while the 4-lane core supports lanes 0-3, and the 8-lane core support lanes 0-7. Table 3 and Table 4 define the transmit and receive signals of the PCI_EXP interface signals for 1- and 4-lane cores, respectively. Table 5 describes the signals for the 8-lane core.

Table 3: **PCI Express Interface Signals for the 1-lane Endpoint Core**

Lane Number	Name	Direction	Description
0	pci_exp_txp0	Output	**PCI Express Transmit Positive**: Serial Differential Output 0 (+)
0	pci_exp_txn0	Output	**PCI Express Transmit Negative**: Serial Differential Output 0 (–)
0	pci_exp_rxp0	Input	**PCI Express Receive Positive**: Serial Differential Input 0 (+)
0	pci_exp_rxn0	Input	**PCI Express Receive Negative**: Serial Differential Input 0 (–)

Table 4: **PCI Express Interface Signals for the 4-lane Endpoint Core**

Lane Number	Name	Direction	Description
0	pci_exp_txp0	Output	**PCI Express Transmit Positive**: Serial Differential Output 0 (+)
0	pci_exp_txn0	Output	**PCI Express Transmit Negative**: Serial Differential Output 0 (–)
0	pci_exp_rxp0	Input	**PCI Express Receive Positive**: Serial Differential Input 0 (+)
0	pci_exp_rxn0	Input	**PCI Express Receive Negative**: Serial Differential Input 0 (–)
1	pci_exp_txp1	Output	**PCI Express Transmit Positive**: Serial Differential Output 1 (+)
1	pci_exp_txn1	Output	**PCI Express Transmit Negative**: Serial Differential Output 1 (–)
1	pci_exp_rxp1	Input	**PCI Express Receive Positive**: Serial Differential Input 1 (+)
1	pci_exp_rxn1	Input	**PCI Express Receive Negative**: Serial Differential Input 1 (–)
2	pci_exp_txp2	Output	**PCI Express Transmit Positive**: Serial Differential Output 2 (+)
2	pci_exp_txn2	Output	**PCI Express Transmit Negative**: Serial Differential Output 2 (–)
2	pci_exp_rxp2	Input	**PCI Express Receive Positive**: Serial Differential Input 2 (+)
2	pci_exp_rxn2	Input	**PCI Express Receive Negative**: Serial Differential Input 2 (–)
3	pci_exp_txp3	Output	**PCI Express Transmit Positive**: Serial Differential Output 3 (+)
3	pci_exp_txn3	Output	**PCI Express Transmit Negative**: Serial Differential Output 3 (–)
3	pci_exp_rxp3	Input	**PCI Express Receive Positive**: Serial Differential Input 3 (+)
3	pci_exp_rxn3	Input	**PCI Express Receive Negative**: Serial Differential Input 3 (–)

Table 5: **PCI Express Interface Signals for the 8-lane Endpoint Core**

Lane Number	Name	Direction	Description
0	pci_exp_txp0	Output	**PCI Express Transmit Positive**: Serial Differential Output 0 (+)
0	pci_exp_txn0	Output	**PCI Express Transmit Negative**: Serial Differential Output 0 (–)
0	pci_exp_rxp0	Input	**PCI Express Receive Positive**: Serial Differential Input 0 (+)
0	pci_exp_rxn0	Input	**PCI Express Receive Negative**: Serial Differential Input 0 (–)
1	pci_exp_txp1	Output	**PCI Express Transmit Positive**: Serial Differential Output 1 (+)
1	pci_exp_txn1	Output	**PCI Express Transmit Negative**: Serial Differential Output 1 (–)
1	pci_exp_rxp1	Input	**PCI Express Receive Positive**: Serial Differential Input 1 (+)
1	pci_exp_rxn1	Input	**PCI Express Receive Negative**: Serial Differential Input 1 (–)
2	pci_exp_txp2	Output	**PCI Express Transmit Positive**: Serial Differential Output 2 (+)
2	pci_exp_txn2	Output	**PCI Express Transmit Negative**: Serial Differential Output 2 (–)
2	pci_exp_rxp2	Input	**PCI Express Receive Positive**: Serial Differential Input 2 (+)
2	pci_exp_rxn2	Input	**PCI Express Receive Negative**: Serial Differential Input 2 (–)
3	pci_exp_txp3	Output	**PCI Express Transmit Positive**: Serial Differential Output 3 (+)
3	pci_exp_txn3	Output	**PCI Express Transmit Negative**: Serial Differential Output 3 (–)
3	pci_exp_rxp3	Input	**PCI Express Receive Positive**: Serial Differential Input 3 (+)
3	pci_exp_rxn3	Input	**PCI Express Receive Negative**: Serial Differential Input 3 (–)
4	pci_exp_txp4	Output	**PCI Express Transmit Positive**: Serial Differential Output 4 (+)
4	pci_exp_txn4	Output	**PCI Express Transmit Negative**: Serial Differential Output 4 (–)
4	pci_exp_rxp4	Input	**PCI Express Receive Positive**: Serial Differential Input 4 (+)
4	pci_exp_rxn4	Input	**PCI Express Receive Negative**: Serial Differential Input 4 (–)
5	pci_exp_txp5	Output	**PCI Express Transmit Positive**: Serial Differential Output 5 (+)
5	pci_exp_txn5	Output	**PCI Express Transmit Negative**: Serial Differential Output 5 (–)
5	pci_exp_rxp5	Input	**PCI Express Receive Positive**: Serial Differential Input 5 (+)
5	pci_exp_rxn5	Input	**PCI Express Receive Negative**: Serial Differential Input 5 (–)
6	pci_exp_txp6	Output	**PCI Express Transmit Positive**: Serial Differential Output 6 (+)
6	pci_exp_txn6	Output	**PCI Express Transmit Negative**: Serial Differential Output 6 (–)
6	pci_exp_rxp6	Input	**PCI Express Receive Positive**: Serial Differential Input 6 (+)
6	pci_exp_rxn6	Input	**PCI Express Receive Negative**: Serial Differential Input 6 (–)
7	pci_exp_txp7	Output	**PCI Express Transmit Positive**: Serial Differential Output 7 (+)
7	pci_exp_txn7	Output	**PCI Express Transmit Negative**: Serial Differential Output 7 (–)
7	pci_exp_rxp7	Input	**PCI Express Receive Positive**: Serial Differential Input 7 (+)
7	pci_exp_rxn7	Input	**PCI Express Receive Negative**: Serial Differential Input 7 (–)

Configuration Interface

The Configuration (CFG) interface provides a mechanism for the user design to inspect the state of the Endpoint core's PCI Express configuration space. The user provides a 10-bit configuration address that selects one of the 1024 configuration space double word (DWORD) registers. The endpoint then returns

the state of the selected register over the 32-bit data output port. Table 6 describes the configuration interface signals.

Table 6: **Configuration Interface Signals**

Name	Direction	Description
cfg_do[31:0]	Output	**Configuration Data Out**: A 32-bit data output port used to obtain read data from the configuration space inside the core.
cfg_rd_wr_done_n	Output	**Configuration Read Write Done**: 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**: A 32-bit data input port used to provide write data to the configuration space inside the core. Unimplemented, reserved for future use.
cfg_dwaddr[9:0]	Input	**Configuration DWORD Address**: A 10-bit address input port used to provide a configuration register DWORD address during configuration register accesses.
cfg_interrupt_n	Input	**Configuration Interrupt**: Active-low interrupt-request signal. The User Application may assert this to cause the 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), the following list defines the type of message to be sent: **Value** — **Legacy Interrupt** 00h — INTA 01h — INTB 02h — INTC 03h — INTD
cfg_interrupt_do[7:0]	Output	**Configuration Interrupt Data Out:** 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].

Table 6: **Configuration Interface Signals *(Cont'd)***

Name	Direction	Description
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. **Value** — **Message Type** 0 — Assert 1 — Deassert
cfg_turnoff_ok_n	Input	**Configuration Turnoff OK**: The active low power turn-off ready signal. The user application can assert this to notify the endpoint that it is safe for power to be turned off.
cfg_wr_en_n	Input	**Configuration Write Enable**: The active low write enable for configuration register access signal. Unimplemented, it is reserved for future use.
cfg_byte_en_n[3:0]	Input	**Configuration Byte Enable**: The active low byte enables for configuration register access signal. Unimplemented, is reserved for future use.
cfg_err_ecrc_n	Input	**ECRC Error Report**: The user can assert this signal to report an end-to-end CRC (ECRC) error. Active low.
cfg_err_cpl_timeout_n	Input	**Configuration Error Completion Timeout**: The user can assert this signal to report a completion timed out. Active low.
cfg_err_cpl_abort_n	Input	**Configuration Error Completion Aborted**: The user can assert this signal to report that a completion was aborted. Active low.
cfg_err_cpl_unexpect_n	Input	**Configuration Error Completion Unexpected**: The user can assert this signal to report that an unexpected completion was received. Active low.
cfg_err_posted_n	Input	**Configuration Error Posted**: 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. Active low.
cfg_err_tlp_cpl_header[47:0]	Input	**Configuration Error TLP Completion Header**: Input that 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.
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.

Table 6: **Configuration Interface Signals *(Cont'd)***

Name	Direction	Description
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_cfg[1023:0]	Input	**Configuration Configure**: This 1024-bit input port is used to configure core options. This port must by driven by a1024-bit constant valued binary pattern by the user.
cfg_err_ur_n	Input	**Configuration Error Unsupported Request**: The user can assert this signal to report that an unsupported request was received. Active low.
cfg_err_cor_n	Input	**Configuration Error Correctable Error**: The user can assert this signal to report that a correctable error was detected. Active low.
cfg_to_turnoff_n	Output	**Configuration To Turnoff**: Output that notifies the user that a PME_TURN_Off message has been received and the CMM will start 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. Active low.
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 PME Message TLP 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 in transition will result in incorrect behavior on the PCI Express Link.
cfg_pcie_link_state_n[2:0][1]	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 in transition
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.

Table 6: **Configuration Interface Signals *(Cont'd)***

Name	Direction	Description
fast_train_simulation_only	Input	**Fast Train**: 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 will cause the core to fail link training.
two_plm_auto_config	Input	**Two-PLM Auto-Config**: Used only for simulation; forces device to act as a down stream device for link training purposes. This signal should only be set to "11," if in simulation, two endpoint cores are connected back-to-back. When using the provided downstream port model for simulation, this signal should be set to "00" on the endpoint core under test. This signal should be set to "00" when using the core in hardware as an endpoint device. Any other setting in hardware is not supported.

Transaction Interface

The Transaction (TRN) interface provides a mechanism for the user design to generate and consume TLPs. The signal names and signal descriptions, as well as the clock cycles and event descriptions for both interfaces, are described in Table 7 through Table 10 and illustrated in Figure 2 and Figure 3.

Transmit TRN Interface

Table 7 defines the transmit (Tx) TRN interface signals.

Table 7: **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[W-1:0]	Input	**Transmit Data:** Packet data to be transmitted. **Product Name** / **Data Bus Width (W)** 1-lane 64-bit Endpoint: 64 4-lane 64-bit Endpoint: 64 8-lane 64-bit Endpoint: 64 1-lane 32-bit Endpoint: 32 4-lane 32-bit Endpoint: 32
trn_trem_n[7:0](1)	Input	**Transmit Data Remainder**: Valid only if both trn_teof_n and trn_tdst_rdy_n are asserted. Legal values are: 0000_0000b = packet data on all of trn_td[63:0], 0000_1111b = packet data only on trn_td[63:32]
trn_terrfwd_n	Input	**Transmit Error Forward**: Marks the current packet in progress as error-poisoned. Can 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[W-1:0]. Active low.
trn_tdst_rdy_n	Output	**Transmit Destination Ready**: Indicates that the core is ready to accept data on trn_td[W-1:0]. Active low. Simultaneous assertion of trn_tsrc_rdy_n and trn_tdst_rdy_n marks successful transfer of data on trn_td[W-1:0].
trn_tsrc_dsc_n	Input	**Transmit Source Discontinue**: Indicates that the user application is aborting the current packet. Active low.

Table 7: **Transmit Transaction Interface Signals *(Cont'd)***

Name	Direction	Description
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 [N-1:0]	Output	**Transmit Buffers Available**: Number of transmit buffers available in the core. The maximum number is 32. Each transmit buffer can accommodate one TLP up to the supported Max_Payload_Size. **Product Name** / **Tx Buffers Available** / **Width (N)** 1-lane 64-bit Endpoint: 16: 5 4-lane 64-bit Endpoint: 16: 5 8-lane 64-bit Endpoint: 32: 6 1-lane 32-bit Endpoint: 8: 5 4-lane 32-bit Endpoint: 16: 5

1. trn_trem[7:0] is not supported or necessary for the 1-lane and 4-lane 32-bit endpoint products.

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 a Quad Word (QWORD) of data.

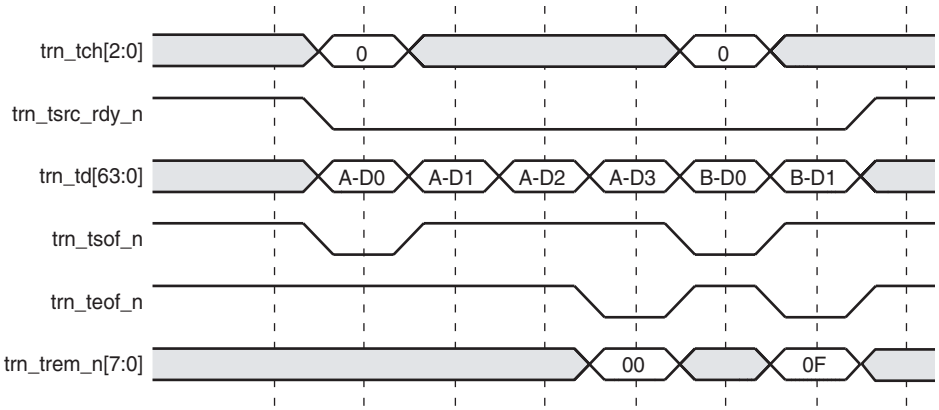

Figure 2: **Tx TRN Interface (64-bit Transaction Interface shown)**

Table 8 defines and describes the transmit path clock cycle signals.

Table 8: **Transmit Path Clock Cycle Signals**

Clock Cycle	Event Description
1	The Endpoint core for PCI Express signals that it can accept the transfer of a TLP, with the assertion of trn_tdst_rdy_n.
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.
3	Frame A QWORD D1 is transferred.
4	Frame A QWORD D2 is transferred.
5	The trn_trem_n[7:0] bus specifies the valid bytes on the last QWORD.
6	Frame B QWORD D0 is transferred.

Table 8: **Transmit Path Clock Cycle Signals**

Clock Cycle	Event Description
7	Frame B QWORD D1 is transferred. The trn_trem_n[7:0] bus specifies the valid bytes on the last QWORD.
8	Note that trn_tdst_rdy_n is not deasserted to offer the user application the option to start the transmission of the next TLP, even though it can be stalled before being completed until a buffer becomes available.

Receive TRN Interface

Table 9 defines the receive (Rx) TRN interface signals.

Table 9: **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[W-1:0]	Output	**Receive Data:** Packet data being received. **Product Name** — **Data Bus Width (W)** 1-lane 64-bit Endpoint — 64 4-lane 64-bit Endpoint — 64 8-lane 64-bit Endpoint — 64 1-lane 32-bit Endpoint — 32 4-lane 32-bit Endpoint — 32
trn_rrem_n[7:0](1)	Output	**Receive Data Remainder**: Valid only if both trn_reof_n and trn_rdst_rdy_n are asserted. Legal values are: 0000_0000b = packet data on all of trn_rd[63:0], 0000_1111b = packet data only on trn_rd[63:32]
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[W-1:0]. Active low.
trn_rdst_rdy_n	Input	**Receive Destination Ready:** Indicates that the user application is ready to accept data on trn_rd[W1:0]. Active low. Simultaneous assertion of trn_rsrc_rdy_n and trn_rdst_rdy_n marks the successful transfer of data on trn_rd[W-1:0].
trn_rsrc_dsc_n	Output	**Receive Source Discontinue:** Indicates that the core is aborting the current packet transfer. 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 one clock cycle before the core presents EOF of the last Non-Posted TLP the user application can accept.

Table 9: **Receive Transaction Interface Signals *(Cont'd)***

Name	Direction	Description
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 will be asserted.
trn_rfc_ph_av[7:0](2)	Output	**Receive Posted Header Flow Control Credits Available:** The number of Posted Header FC credits available to the remote link partner.
trn_rfc_pd_av[11:0]2	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]2	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]2	Output	**Receive Non-Posted Data Flow Control Credits Available:** The number of Non-Posted Data FC credits available to the remote link partner.
trn_rfc_cplh_av[7:0]2	Output	**Receive Completion Header Flow Control Credits Available:** The number of Completion Header FC credits available to the remote link partner. 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]2	Output	**Receive Completion Data Flow Control Credits Available:** The number of Completion Data FC credits available to the remote link partner.

1. trn_rrem[7:0] is not supported or needed for the 1-lane and 4-lane 32-bit endpoint products.
2. Credit values given to the user are instantaneous quantities, not the cumulative (from time zero) values seen by the remote link partner.

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

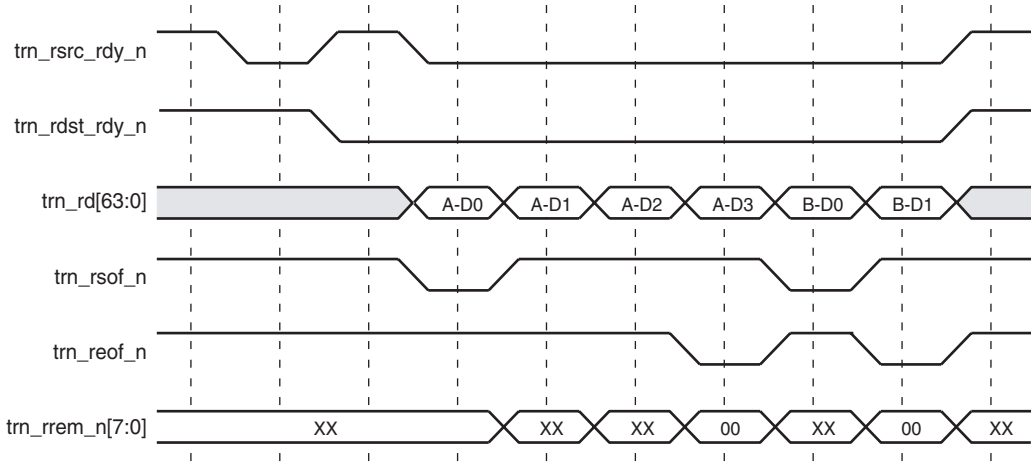

Figure 3: **Rx TRN Interface (64-bit Transaction Interface shown)**

Table 10 defines and describes the receive path clock cycle signals.

Table 10: **Receive Path Clock Cycle Signals**

Clock Cycle	Event Description
1	The core signals by the assertion of trn_rsrc_rdy_n and trn_rsof_n that a valid TLP has been entirely received from the link.
2	The user application asserts trn_rdst_rdy_n to signal that it is ready to receive the TLP.
3	The combined assertion of trn_rsrc_rdy_n and trn_rdst_rdy_n marks a data transfer.
5	The end of the frame is signaled with trn_reof_n.
6	The first QWORD of the second frame is transferred. The core asserts trn_rsof_n to mark the start of the frame.
7	The end of the current frame is marked with the assertion of trn_reof_n.
8	The core deasserts its trn_rsrc_rdy_n signal because there are no more pending TLPs to transfer.

Common TRN Interface

Table 11 defines and describes the common TRN interface signals.

Table 11: **Common Transaction Interface Signals**

Name	Direction	Description
trn_clk	Output	**Transaction Clock**: 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. **Product** — **Frequency (MHz)** 1-lane 64-bit Endpoint — 31.25 MHz 4-lane 64-bit Endpoint — 125 MHz 8-lane 64-bit Endpoint — 250 MHz 1-lane 32-bit Endpoint — 62.5 MHz 4-lane 32-bit Endpoint — 250 MHz
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 core in-band reset events like Hot Reset or Link Disable.
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 endpoint core are lost.

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*.

Ordering Information

The Endpoint for PCIe core is provided under the SignOnce IP Site License and can be generated using the Xilinx CORE Generator system v12.1or 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 product page for this core.

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

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
7/31/02	1.0	Initial Xilinx release.
2/14/03	1.1	Updated data sheet for Xilinx release v1.1 of the product.
6/30/03	1.2	Updated data sheet for Xilinx release v1.2 of the product.
9/26/03	2.0	Updated data sheet for Xilinx release v2.0 of the product.
1/15/05	2.1	Updated document to show core support for Xilinx software v6.3i SP3. Also updated trademark and copyright information in the document.
5/9/05	2.1.1	Updated to support PCI Express Endpoint cores v2.1 and Xilinx software v7.1i SP2.
6/6/05	2.1.2	Update to core version 2.1.1.
8/2/05	3.0	Early access document updates.
8/31/05	3.1	Updated to ISE 7.1i SP4; other minor edits.
10/17/05	3.2	Early access phase II.
11/21/05	3.3	Updated data sheet for initial Xilinx release v3.0.
12/19/05	3.4	Updated ISE tools version, release data and version number.
12/27/05	3.5	Minor edits.
1/18/06	3.6	Updated to Xilinx tools 8.1i.
4/1/06	3.7	Updated Facts table: LUT, FF, BlockRAM, CMPS
5/15/06	3.8	Update to System Interface signals (Table 2) for Virtex-4 reference clock speeds.
7/13/06	4.0	Updated core version to 3.2 and Xilinx tools to 8.2i.
9/21/06	5.0	Updated core version to 3.3.
2/15/07	6.0	Updated core version to 3.4; Xilinx ISE tools to 9.1i.
5/17/07	7.0	Updated core version to 3.5.
4/19/10	8.0	Updated core version to v3.7 and Xilinx ISE tools to v12.1. Removed support for Virtex-II Pro.

Notice of Disclaimer