Endpoint Block Plus v1.9 for PCI Express

DS551 September 19, 2008 **Product Specification**

Introduction

The LogiCORE™ IP Endpoint Block Plus for PCI Express® core is a high-bandwidth, scalable, and reliable serial interconnect building block for use with Virtex®-5 LXT/SXT/FXT/TXT FPGAs. The Endpoint Block Plus for PCI Express (PCIe®) solution supports 1-lane, 4-lane, and 8-lane configurations, all of which are protocol-compliant and electrically compatible with the *PCI Express Base Specification v1.1*.

PCI Express offers a serial architecture that alleviates many 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.

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

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

Product	FPGA Support	Data Path Width
1-lane Endpoint Block Plus	Virtex-5 FPGA LXT/SXT/FXT/TXT	64
4-lane Endpoint Block Plus		64
8-lane Endpoint Block Plus		64

LogiCORE IP Facts				
Core Specifics				
Supported FPGA Device Families	Virtex-5 LXT/SXT/FXT/TXT(1)			
Minimum Device Requirements	1-lane Endpoint Block Plus	XC5VLX20T-1(2) XC5VSX35T-1 XC5VFX30T-1 XC5VTX150T-1		
	4-lane Endpoint Block Plus			
	8-lane Endpoint Block Plus			
Resources Used	**Product**	**I/O(3)**	**LUT(4)**	**FF4**
	1-lane Endpoint Block Plus	1(5)	2350	2800
	4-lane Endpoint Block Plus	4	2400	2900
	8-lane Endpoint Block Plus	8	2450	3100
		Block RAM	**CMPS(6) # Tx Buffers**	**CMPS**
	1-lane Endpoint Block Plus	6	14(7)	512
	4-lane Endpoint Block Plus			
	8-lane Endpoint Block Plus			
Special Features	RocketIO™ GTP and GTX Transceivers Virtex-5 FPGA Integrated Block for PCI Express Phased Lock Loop 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 and VHDL Test Bench Verilog and VHDL Example Design			
Constraints File	User Constraints File (UCF)			
Design Tool Support				
HDL Synthesis Tool	Synplicity® Synplify®, Xilinx XST			
Xilinx Implementation Tools	Xilinx ISE® v10.1			
Simulation Tools(8)	Cadence® IUS v6.1 Synopsys® vcs_mxY-2006.06-SP1 Mentor Graphics® ModelSim® v6.3c			
Support				
Provided by Xilinx, Inc. @ www.xilinx.com/support				

1. Virtex-5 FPGA solutions require the latest production silicon stepping and are pending hardware validation; the LogiCORE warranty does not include production usage with engineering sample silicon (ES).
2. XC5VLX20T does not support 8-lane product.
3. RocketIO GTP or GTX transceivers.
4. Numbers are for the default core configuration; actual LUT and FF utilization values vary based on specific configurations.
5. In Virtex-5 devices, 1-lane Endpoint core uses 1 GTP/GTX tile (2 RocketIO GTP or GTX transceivers).
6. Capability Maximum Payload Size (CMPS).
7. Supports 14 TLPs at CMPS (512 bytes payload):
   - 8 Non-posted, 3 Posted, 3 Completion.
   - Supports 22 TLPs at 256 bytes payload: 8 Non-posted, 7 Posted, 7 Completion.
   - Supports 24 TLPs at 128 bytes payload or less: 8 Non-posted, 8 Posted, 8 Completion.
8. Virtex-5 device designs require either a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator or a SWIFT-compliant simulator. For a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator, ModelSim is currently supported; for a SWIFT-compliant simulator, Cadence IUS and Synopsys are currently supported. See the Facts table for supported versions.

Features

- High-performance, highly flexible, scalable, and reliable, general purpose I/O core
  - 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 RocketIO GTP transceivers for Virtex-5 LXT and SXT devices and RocketIO GTX transceivers for Virtex-5 FXT and TXT devices
  - 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
- Supports Lane Reversal and Lane Polarity Inversion per PCI Express specification requirements
- 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
- 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
- Supports Multi-Vector MSI for up to 32 vectors
- Supports only TC0 / VC0 configuration per PCI Express specification requirement
- Bandwidth scalability with interconnect width
- Fully compliant with PCI Express transaction ordering rules

Applications

The Endpoint Block Plus for PCI Express 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 Block Plus for PCIe internally instances the Virtex-5 FPGA Integrated Block for PCI Express. For information about the internal architecture of the Virtex-5 FPGA Endpoint block, see UG 197, *Virtex-5 FPGA Integrated Endpoint Block for PCI Express Designs User Guide*. Figure 1 illustrates the interfaces to the core.

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

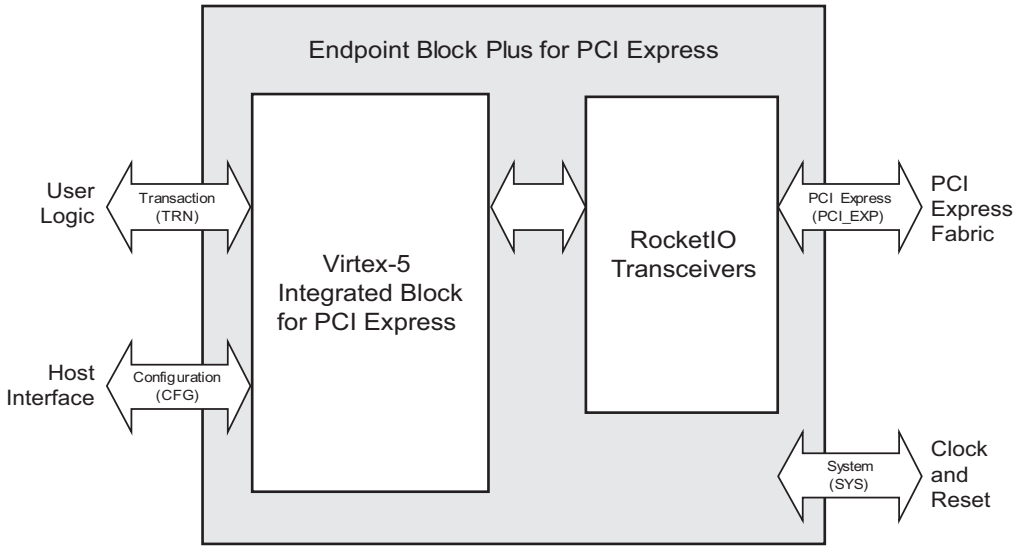

Figure 1: **Endpoint Block Plus for PCI Express Top-level Functional Blocks and Interfaces**

Protocol Layers

The Endpoint Block Plus for PCIe follows the *PCI Express Base Specification* layering model, which consists of the Physical, Data Link, and Transaction Layers. The protocol uses packets to exchange information between layers. 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 functions of the protocol layers 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 protocol layers are defined in the sections that follow.

Physical Layer

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

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. DLLPs are used for conveying information such as Flow Control and TLP acknowledgments.

Transaction Layer

The upper layer of the PCI Express architecture is the Transaction Layer. The primary function 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 and manages credit-based flow control of TLPs.

Configuration Management

The Configuration Management Layer supports generation and reception of System Management Messages by communicating with the other layers and the user application. This layer contains the device configuration space and other system functions. The Configuration layer 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

This integrated block provides a standard Type 0 configuration space. The configuration space consists of a Type 0 configuration space header and 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	16	15	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
Reserved Legacy Configuration Space (Returns 0x00000000)				058h-05Ch
PE Capability		NxtCap	PE Cap	060h
PCI Express Device Capabilities				064h
Device Status		Device Control		068h
PCI Express Link Capabilities				06Ch
Link Status		Link Control		070h
Reserved Legacy Configuration Space (Returns 0x00000000)				074h-0FFh
Next Cap	Cap. Ver.	PCI Exp. Capability		100h
PCI Express Device Serial Number (1st)				104h
PCI Express Device Serial Number (2nd)				108h
Reserved Extended Configuration Space (Returns 0x00000000)				10Ch-FFFh

Endpoint Interfaces

The Endpoint Block Plus for PCIe 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 RocketIO GTP or RocketIO GTX transceivers. 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. sys_reset_n can be tied deasserted in certain form factors where a global reset signal is unavailable. In this case, the core sees a *warm reset* only on device power-on and can be subsequently reset by the connected downstream port utilizing the in-band *hot reset* mechanism.

The system clock signal (sys_clk) is used to clock the entire endpoint, including the RocketIO transceivers. 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.

For reference clock guidelines for RocketIO GTP and GTX Transceivers, see one of the following:

- For Virtex-5 FPGA RocketIO GTP Transceivers, see "*Chapter 10, GTP-to-Board Interface*," in the *Virtex-5 FPGA RocketIO GTP Transceiver User Guide*.
- For Virtex-5 FPGA RocketIO GTX Transceivers, see "*Chapter 10, GTX-to-Board Interface*," in the *Virtex-5 FPGA RocketIO GTX Transceiver User Guide*.

Additional information about core clocking considerations can be found in Answer Record 18329.

The reference clock output signal (refclkout) is a free running reference clock output based on the Reference Clock Frequency selected (sys_clk). This clock signal is derived from the PCIe Lane 0 GTP/GTX REFCLK output, and is available on the FPGA global clock network (BUFG output).

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 places the endpoint in a known initial state. **Note**: sys_reset_n can be tied deasserted in certain form factors where a global reset signal is unavailable. In this case, the core sees a warm reset only on device power-on and can be subsequently reset by the connected downstream port utilizing the in-band hot reset mechanism.
sys_clk	Input	**Reference Clock**: The reference clock for the Endpoint Block Plus solutions. **Product** / **Reference Clock** 1-lane Endpoint Block Plus: 100 or 250 MHz 4-lane Endpoint Block Plus: 100 or 250 MHz 8-lane Endpoint Block Plus: 100 or 250 MHz
refclkout	Output	**Reference Clock Out**: A free running clock output - 100 or 250 MHz, based on Reference Clock.

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 endpoint core supports only lane 0, the 4-lane endpoint core supports lanes 0-3, and the 8-lane endpoint core supports lanes 0-7. Transmit and receive signals of the PCI_EXP interface signals for 1- and 4-lane Endpoint cores are described in Tables 3 and 4, respectively. Table 5 shows the PCI Express signals for the 8-lane Endpoint core.

Table 3: **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: **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: **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's PCI Express configuration space. The user provides a 10-bit configuration address which selects one of the 1024 configuration space double word (DWORD) registers. The Endpoint Block Plus for PCIe core 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. The 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. Currently, writes to the configuration space through the configuration port are not supported.(1)
cfg_di[31:0]	Input	**Configuration Data In**: 32-bit data input port used to provide write data to the configuration space inside the core. Not supported.(1)
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_wr_en_n	Input	**Configuration Write Enable**: Active-low write-enable for configuration register access. Not supported.(1)
cfg_rd_en_n	Input	**Configuration Read Enable**: Active low read-enable for configuration register access. Note: Parking this signal asserted will block updates to internal registers. Only assert when needed.
cfg_interrupt_n	Input	**Configuration Interrupt**: Active-low interrupt-request signal. The User Application may assert this to cause appropriate interrupt messages to be transmitted by the core.
cfg_interrupt_rdy_n	Output	**Configuration Interrupt Ready:** Active-low interrupt grant signal. Assertion on this signal indicates that the core has successfully transmitted the appropriate 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.

Table 6: **Configuration Interface Signals *(Continued)***

Name	Direction	Description
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].
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_to_turnoff_n	Output	**Configuration To Turnoff**: Notifies the user that a PME_TURN_Off message has been received and the main power will soon be removed.
cfg_byte_en_n[3:0]	Input	**Configuration Byte Enable**: Active-low byte enables for configuration register access signal. Not supported.(1)
cfg_bus_number[7:0]	Output	**Configuration Bus Number**: 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**: 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.
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.

Table 6: **Configuration Interface Signals *(Continued)***

Name	Direction	Description
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 signals the core to generate and send a Power Management Wake Event (PM_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]. Assertion of this signal when the PCIe link is in transition results in incorrect behavior on the PCIe link.
cfg_pcie_link_state_n[2:0]	Output	**PCI Express Link State**: One-hot encoded bus that reports the PCIe 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**: If asserted, sets the Transactions Pending bit in the Device Status Register. **Note**: The user is required to assert this input if the User Application has not received a completion to an upstream request.
cfg_dsn[63:0]	Input	**Configuration Device Serial Number**: Serial Number Register fields of the Device Serial Number extended capability.

1. Currently, writing to the configuration space through the user configuration port is not supported; these ports are on the Endpoint core to allow for future feature enhancement.

Error Reporting Signals

Table 7 defines the User Application error-reporting signals.

Table 7: **User Application Error-Reporting Signals**

Port Name	Direction	Description
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_ur_n	Input	**Configuration Error Unsupported Request**: The user can assert this signal to report that an unsupported request was received.
cfg_err_cpl_timeout_n	Input	**Configuration Error Completion Timeout**: The user can assert this signal to report a completion timed out. **Note**: The user should assert this signal only if the device power state is D0. Asserting this signal in non-D0 device power states might result in an incorrect operation on the PCIe link. For additional information, see the *PCI Express Base Specification*, Rev.1.1, Section 5.3.1.2.
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_cpl_abort_n	Input	**Configuration Error Completion Aborted**: The user can assert this signal to report that a completion was aborted.
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_tlp_cpl_header[47:0]	Input	**Configuration Error TLP Completion Header**: 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. The following information should be extracted from the received error TLP and presented in the format below: [47:41] Lower Address [40:29] Byte Count [28:26] TC [25:24] Attr [23:8] Requester ID [7:0] Tag

Table 7: **User Application Error-Reporting Signals *(Continued)***

Port Name	Direction	Description
cfg_err_cpl_rdy_n	Output	**Configuration Error Completion Ready**: When asserted, this signal indicates that the core can accept assertions on cfg_err_ur_n and cfg_err_cpl_abort_n for Non-Posted Transactions. Assertions on cfg_err_ur_n and cfg_err_cpl_abort_n are ignored when cfg_err_cpl_rdy_n is deasserted.
cfg_err_locked_n	Input	**Configuration Error Locked**: 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 that caused the error was a locked transaction. This signal is intended to be used in Legacy mode. If the user needs to signal an unsupported request or an aborted completion for a locked transaction, this signal can be used to return a Completion Locked with UR or CA status. **Note**: When not in Legacy mode, the core will automatically return a Completion Locked, if appropriate.

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 shown in Tables 8 through 12, and in Figures 2 and 3.

Transmit TRN Interface

Table 8 defines the transmit (Tx) Transaction interface signals.

Table 8: **Transmit Transaction Interface Signals**

Name	Direction	Description
trn_tsof_n	Input	**Transmit Start-of-Frame** (SOF): Active low. Signals the start of a packet. Valid only along with assertion of trn_tsrc_rdy_n.
trn_teof_n	Input	**Transmit End-of-Frame** (EOF): Active low. Signals the end of a packet. Valid only along with assertion of trn_tsrc_rdy_n.
trn_td[63:0]	Input	**Transmit Data:** Packet data to be transmitted.
trn_trem_n[7:0]	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_tsrc_rdy_n	Input	**Transmit Source Ready**: Active low. Indicates that the User Application is presenting valid data on trn_td[63:0].
trn_tdst_rdy_n	Output	**Transmit Destination Ready**: Active low. Indicates that the core is ready to accept data on trn_td[63:0]. The simultaneous assertion of trn_tsrc_rdy_n and trn_tdst_rdy_n marks the successful transfer of one data beat on trn_td[63:0].
trn_tsrc_dsc_n	Input	**Transmit Source Discontinue**: May be asserted any time starting on the first cycle after SOF to EOF, inclusive.
trn_tdst_dsc_n	Output	**Transmit Destination Discontinue**: Active low. Indicates that the core is aborting the current packet. Asserted when the physical link is going into reset. Not supported; signal is tied high.
trn_tsrc_dsc_n	Input	**Transmit Source Discontinue**: May be asserted any time starting on the first cycle after SOF to EOF, inclusive.
trn_tbuf_av[3:0]	Output	**Transmit Buffers Available**: Indicates transmit buffer availability in the core. Each bit of trn_tbuf_av corresponds to one of the following credit queues: • trn_tbuf_av[0] ≥ Non Posted Queue • trn_tbuf_av[1] ≥ Posted Queue • trn_tbuf_av[2] ≥ Completion Queue • trn_tbuf_av[3] ≥ Look-Ahead Completion Queue A value of 1 indicates that the core can accept at least 1 TLP of that particular credit class. A value of 0 indicates no buffer availability in the particular queue. trn_tbuf_av[3] indicates that the core may be about to run out of Completion Queue buffers. If this is deasserted, performance can be optimized by sending a Posted or Non-Posted TLP instead of a Completion TLP. If a Completion TLP is sent when this signal is de-asserted, the core may be forced to stall the TRN interface for all TLP types, until new Completion Queue buffers become available.

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.

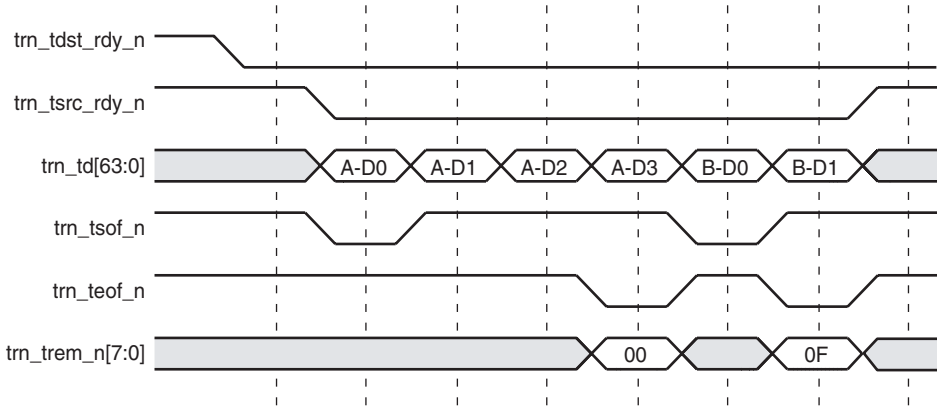

Figure 2: **Tx TRN Interface**

Table 9 defines and describes the transmit path clock cycle signals.

Table 9: **Transmit Path Clock Cycle Signals**

Clock Cycle	Event Description
1	The Block Plus core 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. Frame A QWORD 0 is transferred in conjunction with trn_tsof_n.
3	Frame A QWORD D1 is transferred.
4	Frame A QWORD D2 is transferred.
5	Frame A QWORD D3 is transferred. The trn_trem_n[7:0] bus specifies that all 8 bytes are valid on the last QWORD.
6	Frame B QWORD D0 is transferred.
7	Frame B QWORD D1 is transferred. The trn_trem_n[7:0] bus specifies that the upper 4 bytes are valid (trn_td[63:32]) on the last QWORD.
8	Note that trn_tdst_rdy_n remains asserted to offer the user application the option to start the transmission of the next TLP.

Receive TRN Interface

Table 10 defines the receive (Rx) TRN interface signals.

Table 10: **Receive Transaction Interface Signals**

Name	Direction	Description
trn_rsof_n	Output	**Receive Start-of-Frame** (**SOF**): Active low. Signals the start of a packet. Valid only if trn_rsrc_rdy_n is also asserted.
trn_reof_n	Output	**Receive End-of-Frame** (**EOF**): Active low. Signals the end of a packet. Valid only if trn_rsrc_rdy_n is also asserted.
trn_rd[63:0]	Output	**Receive Data:** Packet data being received. Valid only if trn_rsrc_rdy_n is also asserted.
trn_rrem_n[7:0]	Output	**Receive Data Remainder**: Valid only if all of the following signals are asserted: trn_reof_n, trn_rsrc_rdy_n, and trn_rdst_rdy_n. 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:** Active low. Marks the packet in progress as error poisoned. Asserted by the core for the entire length of the packet.
trn_rsrc_rdy_n	Output	**Receive Source Ready:** Active low. Indicates the core is presenting valid data on trn_rd[63:0].
trn_rdst_rdy_n	Input	**Receive Destination Ready:** Active low. Indicates the User Application is ready to accept data on trn_rd[63:0]. The simultaneous assertion of trn_rsrc_rdy_n and trn_rdst_rdy_n marks the successful transfer of one data beat on trn_td[63:0].
trn_rsrc_dsc_n	Output	**Receive Source Discontinue:** Active low. Indicates the core is aborting the current packet. Asserted when the physical link is going into reset. Not supported; signal is tied high.
trn_rnp_ok_n	Input	**Receive Non-Posted OK:** Active low. The User Application asserts trn_rnp_ok_n when it is ready to accept a Non-Posted Request packet. When asserted, packets are presented to the user application in the order they are received unless trn_rcpl_streaming_n is asserted. 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 next-to-last Non-Posted TLP the User Application can accept, allowing Posted and Completion packets to bypass Non-Posted packets in the inbound queue.
trn_rcpl_streaming_n	Input	**Receive Completion Streaming:** Active low. Asserted to enable Upstream Memory Read transmission without the need for throttling.

Table 10: **Receive Transaction Interface Signals *(Continued)***

Name	Direction	Description
trn_rbar_hit_n[6:0]	Output	**Receive BAR Hit:** Active low. Indicates BAR(s) targeted by the current receive transaction. 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 BAR 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](1)	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](1)	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](1)	Output	**Receive Non-Posted Header Flow Control Credits Available:** Number of Non-Posted Header FC credits available to the remote link partner.
trn_rfc_npd_av[11:0](1)	Output	**Receive Non-Posted Data Flow Control Credits Available:** Number of Non-Posted Data FC credits available to the remote link partner. Always 0 as a result of advertising infinite initial data credits.

1. 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 from the PCI Express Link on the TRN interface.

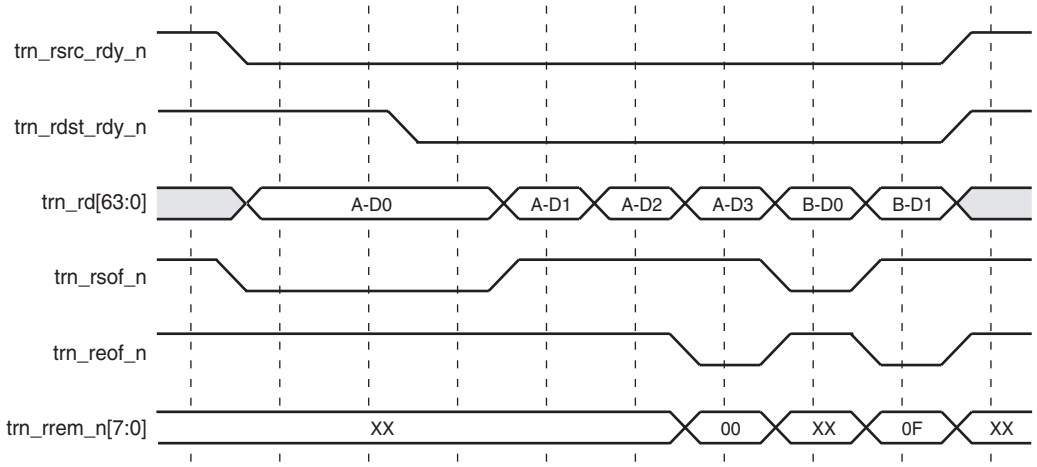

Figure 3: **Rx TRN Interface**

Table 11 defines the receive path clock cycle signals.

Table 11: **Receive Path Clock Cycle Signals**

Clock Cycle	Event Description
1	The Block Plus 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.
3	The user application asserts trn_rdst_rdy_n to signal that it is ready to receive the TLP. The combined assertion of trn_rsrc_rdy_n and trn_rdst_rdy_n marks a data transfer.
6	The end of the frame is signaled with trn_reof_n. The trn_rrem_n[7:0] bus specifies all bytes are valid on the last QWORD.
7	The first QWORD of the second frame is transferred. The core asserts trn_rsof_n to mark the start of the frame.
8	The end of the current frame is marked with the assertion of trn_reof_n. The trn_rrem_n[7:0] bus specifies that the upper 4 bytes (trn_rd[63:32]) are valid on the last QWORD.
9	The Block Plus core deasserts its trn_rsrc_rdy_n signal because there are no more pending TLPs to transfer.

Common TRN Interface

Table 12 defines and describes the common TRN interface signals.

Table 12: **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 after the core deasserts trn_reset_n. The trn_clk clock output is a fixed frequency configured in the CORE Generator GUI. trn_clk does not shift frequencies in case of link recovery or training down. **Product** / **Recommended Frequency (MHz)** / **Optional Frequency (MHz)** 1-lane Endpoint Block Plus: 62.5 / 125.0 4-lane Endpoint Block Plus: 125.0 / 250.0 8-lane Endpoint Block Plus: 250.0 / 125.0
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. trn_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.

Ordering and Support Information

The Endpoint Block Plus for PCI Express provides two free licensing options. The Simulation Only license lets you assess the core functionality and demonstrates the various interfaces to the core in simulation, and is provided with the Xilinx CORE Generator v10.1 and higher. The Full license lets you assess the core functionality and demonstrates the various interfaces to the core in both simulation and hardware, and is available after registering for access to the product lounge, a secure area of the Endpoint Block Plus for PCI Express product page.

For information about other Xilinx LogiCORE IP modules, contact your local Xilinx sales representative or visit the Xilinx IP Center. Xilinx provides technical support for this product when used as described in product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices not listed above, or if customized beyond that allowed in the product documentation, or if any changes are made in sections of design marked DO NOT MODIFY.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
9/21/06	1.0	Initial Xilinx release.
2/15/07	2.0	Update core to version 1.2; Xilinx tools v9.1i.
5/17/07	3.0	Update core to version 1.3; updated for PCI-SIG compliance.
8/8/07	4.0	Update core to version 1.4; ISE tools v9.2i, Cadence IUS v5.8.
10/10/07	5.0	Update core to version 1.5, Cadence IUS to v6.1.
3/24/08	6.0	Update core to version 1.6, Xilinx tools 10.1.
4/25/08	7.0	Update core to version 1.7.
6/27/08	8.0	Update core to version 1.8.
9/18/08	9.0	Update core to version 1.9.

Notice of Disclaimer