Spartan-6 FPGA Integrated Endpoint Block v1.2 for PCI Express

DS718 September 16, 2009 **Product Specification**

Introduction

The LogiCORE™ IP Spartan®-6 FPGA Integrated Endpoint Block for PCI Express® core is a high-bandwidth, scalable, and reliable serial interconnect building block for use with Spartan-6 FPGA devices. The Spartan-6 FPGA Integrated Endpoint Block for PCI Express (PCIe®) solution supports a 1-lane configuration that is 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 Integrated Endpoint Block sets the industry standard for a high-performance, cost-efficient third-generation I/O solution.

The Integrated Endpoint Block solution 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 Integrated Endpoint Block solution is defined in the following table.

Product Name	1-lane Integrated Endpoint Block
FPGA Architecture	Spartan-6
User Interface Width	32
Lane Widths Supported	x1
Link Speeds Supported	2.5 GT/s
PCI Express Base Specification Compliance	v1.1

LogiCORE IP Facts			
Core Specifics			
Supported FPGA Device Families	Spartan-6(1)		
Minimum Device Requirements	XC6SLX25T-CSG324-2		
Resources Used	**I/O(2)**	**LUT(3)**	**FF(3)**
	1(4)	3	0
	Block RAM	**CMPS(5) # Tx Buffers**	**CMPS**
	2-18	30(6)	512
Special Features	GTP Transceivers Spartan-6 FPGA Integrated Block for PCI Express Phased Lock Loop Block RAM		
Provided with Core			
Documentation	Product Specification, User Guide, Instantiation Template		
Design Files	Verilog and VHDL Unencrypted RTL source files for Simulation and Synthesis, 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® v11.3		
Simulation Tools(7)	Cadence® IUS v8.1 -s009 and above Synopsys® VCS-MX 2008.09 and above Mentor Graphics® ModelSim® v6.4b and above		
Support			
Provided by Xilinx, Inc. @ www.xilinx.com/support			

1. Spartan-6 FPGA solutions require the latest production silicon stepping and are pending hardware validation; the LogiCORE IP warranty does not include production usage with engineering sample silicon (ES).
2. GTP transceivers.
3. Numbers are for the default core configuration; actual LUT and FF utilization values vary based on specific configurations.
4. In Spartan-6 devices, 1-lane Endpoint core uses 1 GTP tile (2 GTP transceivers). It is possible to use the other GTP transceiver for user designs with some limitations.
5. Capability Maximum Payload Size (CMPS).
6. Supports 30 TLPs at CMPS (512 bytes payload): No restrictions based on type.
   - Supports 29 TLPs at 256 bytes payload: No restrictions based on type.
   - Supports 27 TLPs at 128 bytes payload or less: No restrictions based on type.
7. Requires a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator. For VHDL simulation, a mixed HDL license is required.

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 GTP transceivers for Spartan-6 LXT
  - 2.5 Gbps line speed
  - Supports 1-lane operation
  - Elastic buffers and clock compensation
  - Automatic clock data recovery
- 8b/10b encode and decode
- Supports 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
  - Transmit streaming, cut-through mode on TX interface for decreased latency
  - 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
- Fully compliant with PCI Express transaction ordering rules

Applications

The Spartan-6 FPGA Integrated Endpoint Block 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

For information about the internal architecture of the Spartan-6 FPGA Endpoint block, see UG 654, *Spartan-6 FPGA Integrated Endpoint Block for PCI Express User Guide*. Figure 1 illustrates the interfaces to the core.

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

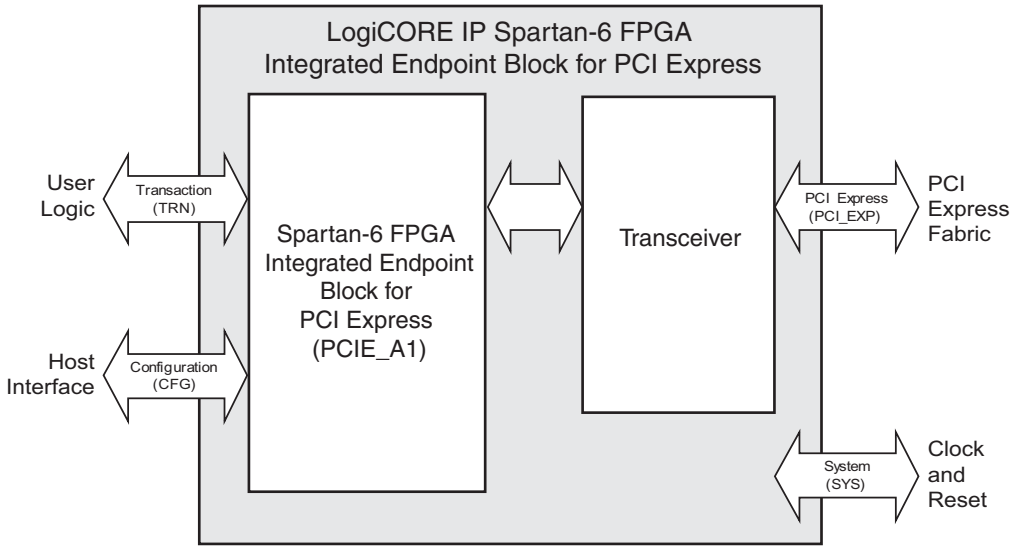

Figure 1: **Integrated Endpoint Block for PCI Express Top-level Functional Blocks and Interfaces**

Protocol Layers

The Integrated Endpoint Block 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 (optional)

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
	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
Optional	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 Integrated Endpoint Block 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 GTP 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 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 GTP Transceivers, see the *Spartan-6 FPGA GTP Transceiver User Guide*.

Additional information about core clocking considerations can be found in UG654, *LogiCORE IP Spartan-6 FPGA Integrated Endpoint Block for PCI Express User Guide*.

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 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 Integrated Endpoint Block solutions. **Product** — **Reference Clock** 1-lane Integrated Endpoint Block — 125 or 250 MHz

PCI Express Interface

The PCI Express (PCI_EXP) interface consists of differential transmit and receive pairs. 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. Transmit and receive signals of the PCI_EXP interface signals for 1-lane Endpoint cores are described in Table 3.

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 (–)

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 Integrated Endpoint Block core returns the state of the selected register over the 32-bit data output port. Table 4 describes the configuration interface signals.

Table 4: **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. • Writes to the configuration space is not supported.
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_rd_en_n	Input	**Configuration Read Enable**: Active low read-enable for configuration register access. ***Note:*** cfg_rd_en_n must be asserted for no more than 1 trn_clk cycle at a time.
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.

Table 4: **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_interrupt_di[7:0]	Input	**Configuration Interrupt Data In**: For Message Signaled 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 not used and is provided for informational purposes and backwards compatibility.
cfg_interrupt_mmenable[2:0]	Output	**Configuration Interrupt Multiple Message Enable:** This is the value of the Multiple Message Enable field and defines the number of vectors the system allows for multi-vector MSI. Values range from 000b to 101b. A value of 000b indicates that single vector MSI is enabled, while other values indicate the number of lower-order bits that may be used for cfg_interrupt_di[7:0]. cfg_interrupt_mmenable[2:0] values: • 000b, 0 bits • 001b, 1 bit • 010b, 2 bits • 011b, 3 bits • 100b, 4 bits • 101b, 5 bits
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. If 1, then only MSI interrupts may be sent.
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.

Table 4: **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_to_turnoff_n	Output	**Configuration To Turnoff**: Active low. 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. After cfg_turnoff_ok_n is asserted, CMM sends a PME_To_Ack message to the upstream device.
cfg_turnoff_ok_n	Input	**Configuration Turnoff OK**: Active low. The user application can assert this to notify the Integrated Endpoint core that it is safe to turn the power off.
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] bus. Assertion of this signal when the PCI Express Link is in transition results in incorrect behavior on the PCI Express Link.
cfg_pcie_link_state_n[2:0]	Output	**PCI Express Link State**: This encoded bus 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. Active low.
cfg_dsn[63:0]	Input	**Configuration Device Serial Number**: Serial Number Register fields of the Device Serial Number extended capability. Not used if DSN capability is disabled.

Error Reporting Signals

Table 5 defines the User Application error-reporting signals.

Table 5: **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). Active low.
cfg_err_ur_n	Input	**Configuration Error Unsupported Request**: Active low. The user can assert this signal to report that an unsupported request was received. This signal is ignored if cfg_err_cpl_rdy_n is deasserted.
cfg_err_cpl_timeout_n	Input	**Configuration Error Completion Timeout**: Active low. 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**: Active low. The user can assert this signal to report that an unexpected completion was received.
cfg_err_cpl_abort_n	Input	**Configuration Error Completion Aborted**: Active low. The user can assert this signal to report that a completion was aborted. This signal is ignored if cfg_err_cpl_rdy_n is deasserted.
cfg_err_posted_n	Input	**Configuration Error Posted**: Active low. 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**: Active low. 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 5: **User Application Error-Reporting Signals *(Cont'd)***

Port Name	Direction	Description
cfg_err_cpl_rdy_n	Output	**Configuration Error Completion Ready**: Active low. 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**: Active low. 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 Table 6, Table 7, Table 8, Table 9, and Table 10, and in Figure 2 and Figure 3.

Transmit TRN Interface

Table 6 defines the transmit (Tx) Transaction interface signals.

Table 6: **Transaction Transmit 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[31:0]	Input	**Transmit Data:** Packet data to be transmitted.
trn_tsrc_rdy_n	Input	**Transmit Source Ready**: Active low. Indicates that the User Application is presenting valid data on trn_td[31:0]. The simultaneous assertion of trn_tsrc_rdy_n and trn_tdst_rdy_n marks the successful transfer of one data beat on trn_td[31:0].
trn_tdst_rdy_n	Output	**Transmit Destination Ready**: Active low. Indicates that the core is ready to accept data on trn_td[31:0]. The simultaneous assertion of trn_tsrc_rdy_n and trn_tdst_rdy_n marks the successful transfer of one data beat on trn_td[31:0].
trn_tsrc_dsc_n	Input	**Transmit Source Discontinue**: Active low. Can be asserted any time starting on the first cycle after SOF to EOF, inclusive.
trn_tbuf_av[5:0]	Output	**Transmit Buffers Available**: Indicates the number of transmit buffers available in the core. Each free transmit buffer can accommodate one TLP up to the supported Maximum Payload Size. Maximum number of Transmit buffers is determined by the Supported Maximum Payload Size and block RAM configuration selected.
trn_terr_drop_n	Output	**Transmit Error Drop**: Active Low. Indicates that the core discarded a packet because of a length violation or, when streaming, data was not presented on consecutive clock cycles. Length violations include packets longer than supported.
trn_tstr_n	Input	**Transmit Streamed**: Active low. Indicates a packet will be presented on consecutive clock cycles and transmission on the link may begin before the entire packet has been written to the core. Commonly referred to as transmit cut-through mode.
trn_tcfg_req_n	Output	**Transmit Configuration Request**: Active low. Asserted when the core is ready to transmit a Configuration Completion or other internally-generated TLP.

Table 6: **Transaction Transmit Interface Signals *(Cont'd)***

Name	Direction	Description
trn_tcfg_gnt_n	Input	**Transmit Configuration Grant**: Active low. Asserted by the user application in response to trn_tcfg_req_n, to allow the core to transmit an internally-generated TLP. Holding trn_tcfg_gnt_n deasserted after trn_tcfg_req_n allows user-initiated TLPs to be given higher priority of transmission over core generated TLPs. trn_tcfg_req_n will be asserted once for each internally-generated packet. It may not deassert immediately following trn_cfg_gnt_n if there are no transmit buffers available. If the user does not want to control the transmission of internally-generated TLPs, this signal may be continuously asserted.
trn_terrfwd_n	Input	**Transmit Error Forward:** Active low. This input marks the current packet in progress as error-poisoned. It can be asserted any time between SOF and EOF, inclusive. trn_terrfwd_n must not be asserted if trn_tstr_n is asserted.

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 Double Word (DWORD) of data.

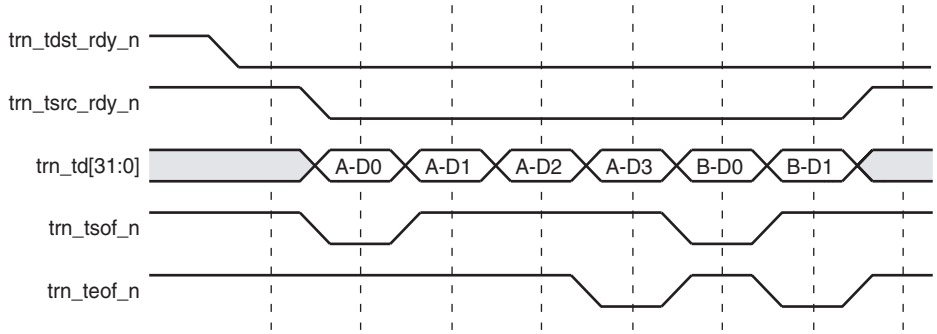

Figure 2: **Tx TRN Interface**

Table 7 defines and describes the transmit path clock cycle signals.

Table 7: **Transmit Path Clock Cycle Signals**

Clock Cycle	Event Description
1	The Integrated Endpoint Block 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 DWORD 0 is transferred in conjunction with trn_tsof_n.
3	Frame A DWORD D1 is transferred.
4	Frame A DWORD D2 is transferred.
5	Frame A DWORD D3 is transferred.
6	Frame B DWORD D0 is transferred.
7	Frame B DWORD D1 is transferred.
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 8 defines the receive (Rx) TRN interface signals.

Table 8: **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[31:0]	Output	**Receive Data:** Packet data being received. Valid only if trn_rsrc_rdy_n is also asserted.
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[31:0]
trn_rdst_rdy_n	Input	**Receive Destination Ready:** Active low. Indicates the User Application is ready to accept data on trn_rd[31:0]. The simultaneous assertion of trn_rsrc_rdy_n and trn_rdst_rdy_n marks the successful transfer of one data beat on trn_td[31:0].
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. Active low.
trn_rnp_ok_n	Input	**Receive Non-Posted OK:** Active low. The user application asserts this signal when it is ready to accept a Non-Posted Request TLPs. trn_rnp_ok_n must be deasserted when the user application cannot process received Non-Posted TLPs, so that these may be buffered within the core's receive queue. In this case, Posted and Completion TLPs received after the Non-Posted TLPs will bypass the blocked TLPs. 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.
trn_rbar_hit_n[6:0]	Output	**Receive BAR Hit:** Active low. Indicates BAR(s) targeted by the current receive transaction. Asserted throughout the packet, from trn_rsof_n to trn_reof_n. • 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.

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

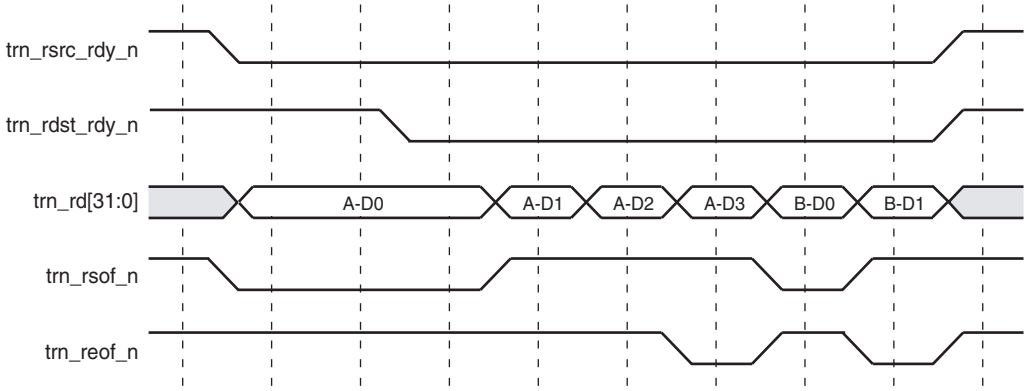

Figure 3: **Rx TRN Interface**

Table 9 defines the receive path clock cycle signals.

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

Clock Cycle	Event Description
1	The Integrated Endpoint Block 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.
7	The first DWORD 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.
9	The Integrated Endpoint Block core deasserts its trn_rsrc_rdy_n signal because there are no more pending TLPs to transfer.

Common TRN Interface

Table 10 defines and describes the common TRN interface signals.

Table 10: **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 only after trn_reset_n is de-asserted. The trn_clk clock output is a fixed frequency clock output. trn_clk does not change frequencies in case of link recovery. • 1-lane Integrated Endpoint Block Frequency: 62.5 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 its quiescent state. trn_reset_n is deasserted synchronously with respect to trn_clk, trn_reset_n is asserted asynchronously with sys_reset_n assertion. Note that trn_reset_n is asserted for core in-band reset events like Hot Reset or Link Disable.

Table 10: **Common Transaction Interface Signals *(Cont'd)***

Name	Direction	Description
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, or when communication with the link partner is lost due to errors on the transmission channel. trn_lnk_up_n is also deasserted when the core is driven to Hot Reset or Link Disable states by the link partner, and all TLPs stored in the core are lost.
trn_fc_sel[2:0]	Input	**Flow Control Informational Select**: Selects the type of flow control information presented on the trn_fc_* signals. Possible values: • 000 == receive buffer available space • 001 == receive credits granted to the link partner • 010 == receive credits consumed • 100 == transmit user credits available • 101 == transmit credit limit • 110 == transmit credits consumed
trn_fc_ph[7:0]	Output	**Posted Header Flow Control Credits:** The number of Posted Header FC credits for the selected flow control type
trn_fc_pd[11:0]	Output	**Posted Data Flow Control Credits:** The number of Posted Data FC credits for the selected flow control type.
trn_fc_nph[7:0]	Output	**Non-Posted Header Flow Control Credits:** The number of Non-Posted Header FC credits for the selected flow control type.
trn_fc_npd[11:0]	Output	**Non-Posted Data Flow Control Credits:** The number of Non-Posted Data FC credits for the selected flow control type.
trn_fc_cplh[7:0]	Output	**Completion Header Flow Control Credits:** The number of Completion Header FC credits for the selected flow control type.
trn_fc_cpld[11:0]	Output	**Completion Data Flow Control Credits:** The number of Completion Data FC credits for the selected flow control type.

Support

Xilinx provides technical support for this LogiCORE IP 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 Spartan-6 FPGA Integrated Block for PCI Express is included with the ISE CORE Generator v11.2 and higher. To activate full access to this core, you must install a Full license key which you can generate at the Xilinx.com Product Download and Licensing Site. For detailed instructions, please refer to the "Access Core" link on the Spartan-6 FPGA Integrated Endpoint Block for PCI Express product page.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/24/09	1.0	Initial Xilinx draft.
09/16/09	2.0	Updated core to v1.2 and ISE to v11.3. Added support for VHDL.

Notice of Disclaimer