LATTICE SEMICONDUCTOR

LatticeCORE

PCI Express 2.0 x1, x4 Endpoint IP Core User's Guide

October 2014
IPUG75_2.2

Table of Contents

Introduction

PCI Express is a high performance, fully scalable, well defined standard for a wide variety of computing and communications platforms. It has been defined to provide software cPCI Express 2.0 x1, x4 IP Coreompatibility with existing PCI drivers and operating systems. Being a packet based serial technology, PCI Express greatly reduces the number of required pins and simplifies board routing and manufacturing. PCI Express is a point-to-point technology, as opposed to the multidrop bus in PCI. Each PCI Express device has the advantage of full duplex communication with its link partner to greatly increase overall system bandwith. The basic data rate for a single lane is double that of the 32 bit/33 MHz PCI bus. A four lane link has eight times the data rate in each direction of a conventional bus.

Lattice's PCI Express core provides a x1 or x4 endpoint solution from the electrical SERDES interface to the transaction layer. This solution supports the LatticeECP3™, LatticeECP2M™ and LatticeSCM™ FPGA device families. The LatticeSCM PCI Express core utilizes Lattice's unique MACO™ technology to support the data link layer using the flexiMAC™ MACO core and the portions of the PHY layer using the LTSSM MACO core. The Lattice MACO technology is only available on the LatticeSCM family of the devices. When used with the LatticeECP3™ and LatticeECP2M families, the PCI Express core is implemented using an extremely economical and high value FPGA platform.

This user's guide covers three versions of the Lattice PCI Express core:

- The **Native PCI Express x4 Core** targets the LatticeECP3, LatticeECP2M and LatticeSCM families of devices.
- The **x4 Downgraded x1 Core** also targets the LatticeECP3, LatticeECP2M and LatticeSCM families. The x4 Downgraded x1 core is a x4 core that has three channels powered down to create a x1 core. This is designed for users who wish to use a 64-bit datapath with a x1 link width.
- The **Native PCI Express x1 Core** is only available in the LatticeECP3 and LatticeECP2M families. This is a reduced LUT count x1 core with a 16-bit datapath.

Refer to Lattice's PCIe Solutions web site at:

http://www.latticesemi.com/Solutions.aspx

for links to the following documents, solutions, and IP related to the PCI Express IP core:

- LatticeECP3 PCI Express Development Kit and LatticeECP2M PCI Express Development Kit
- LatticeECP2M PCI Express Solutions Board and LatticeECP3 PCI Express Solutions Board (contained in the Development Kits listed above)
- LatticeSC PCI Express x4 Evaluation Board
- PCI Express Endpoint IP Core Demo for LatticeECP3, LatticeECP2M and LatticeSCM
- Lattice Scatter-Gather Direct Memory Access (DMA) Controller IP Core

Quick Facts

Table 1-1 gives quick facts about the Lattice PCI Express IP core.

Table 1-1. PCI Express IP Core Quick Facts

		PCI Express IP Configuration					
		x1 Endpoint		**x4 Endpoint**[1]		**x4 Endpoint, Soft LTSSM**[1, 2, 3]	**x4 Endpoint, Hard LTSSM**[1, 2, 3]
Core Requirements	FPGA Families Supported	LatticeECP2M and LatticeECP3				LatticeSCM	
	Minimal Device Needed [4]	LFE3-17E-7FN484C	LFE2M-20E-6F484C	LFE3-70E-7FN672C	LFE2M-50E-6F672C	LFSC3GA15E-6F900C	LFSC3GA15E-6F900C
Typical Resource Utilization	Targeted Device	LFE3-17E-7FN484C	LFE2M-20E-6F484C	LFE3-70E-7FN672C	LFE2M-50E-6F672C	LFSC3GA25E-6FC1020C	LFSC3GA80E-6FC1704C
	Data Path Width	16	16	64	64	64	64
	LUTs	6100	6000	12100	12200	8200	4700
	sysMEM EBRs	4	4	11	11	13	13
	Registers	4000	4100	9800	9600	6100	4700
Design Tool Support [5]	Lattice Implementation	Lattice Diamond™ 1.1 or ispLEVER® 8.1 SP1					
	Synthesis	Synopsys® Synplify® Pro for Lattice D-2009.12L-1					
		Mentor Graphics® Precision® RTL					
	Simulation	Aldec Active-HDL® 8.2 (Windows only, Verilog and VHDL)					
		Mentor Graphics ModelSim® SE 6.5F (Verilog Only)					
		Cadence® NC-Verilog® (Linux only)					

1. When the x4 cores downgrades to x1 mode, utilization and performance results for x1 are identical to x4 mode for LatticeECP2M, LatticeECP3, and LatticeSCM families.
2. LTSSM MACO is available in LFSC3GA15/40/80/115 devices. LTSSM MACO not available in LFSC3GA25; soft LTSSM is required.
3. Resource utilization and performance results for x1 and x4 mode are identical in LatticeSCM devices.
4. The packages specified in the Minimal Device Needed row relate to the many user interface signals implemented as I/Os in the evaluation design. Depending on the application, it might be possible to implement a design in a package with fewer I/O pins since the majority of the user interface signals are terminated inside the FPGA.
5. Design tool support for IP core version 4.1.

Features

The Lattice PCI Express IP core supports the following features.

PHY Layer

- 2.5 Gbps CML electrical interface
- PCI Express 1.1 electrical compliance for the LatticeECP2M and LatticeSCM families, and 2.0 electrical compliance for the LatticeECP3 family
- Many options for signal integrity including differential output voltage, transmit pre-emphasis and receiver equalization
- Serialization and de-serialization
- 8b10b symbol encoding/decoding
- Data scrambling and de-scrambling
- Link state machine for symbol alignment
- Clock tolerance compensation supports +/- 300 ppm
- Framing and application of symbols to lanes

- Data scrambling
- Lane-to-lane de-skew
- Link Training and Status State Machine (LTSSM)
  - Electrical idle generation
  - Receiver detection
  - TS1/TS2 generation/detection
  - Lane polarity inversion
  - Link width negotiation
  - Higher layer control to jump to defined states
  - MACO-based LTSSM available in the following LatticeSCM devices: LFSCM15, LFSCM40, LFSCM80, and LFSCM115

Data Link Layer

- Data link control and management state machine
- Flow control initialization
- Ack/Nak DLLP generation/termination
- Power management DLLP generation/termination through simple user interface
- LCRC generation/checking
- Sequence number appending/checking/removing
- Retry buffer and management

Transaction Layer

- Supports all types of TLPs (memory, I/O, configuration and message)
- Power management user interface to easily send and receive power messages
- Optional ECRC generation/checking
- 128, 256, 512, 1 k, 2 k, or 4 kbyte maximum payload size

Configuration Space Support

- PCI-compatible Type 0 Configuration Space Registers contained inside the core (0x0-0x3C)
- PCI Express Capability Structure Registers contained inside the core
- Power Management Capability Structure Registers contained inside the core
- MSI Capability Structure Registers contained inside the core
- Device Serial Number Capability Structure contained inside the core
- Advanced Error Reporting Capability Structure contained inside the core

Top Level IP Support

- 125 MHz user interface
  - Native x4 and Downgraded x1 support a 64-bit datapath
  - Native x1 supports a 16-bit datapath
- In transmit, user creates TLPs without ECRC, LCRC, or sequence number
- In receive, user receives valid TLPs without ECRC, LCRC, or sequence number
- Credit interface for transmit and receive for PH, PD, NPH, NPD, CPLH, CPLD credit types

- Upstream/downstream, single function endpoint topology
- Higher layer control of LTSSM via ports
- Access to select configuration space information via ports

This chapter provides a functional description of the Lattice PCI Express Endpoint IP core.

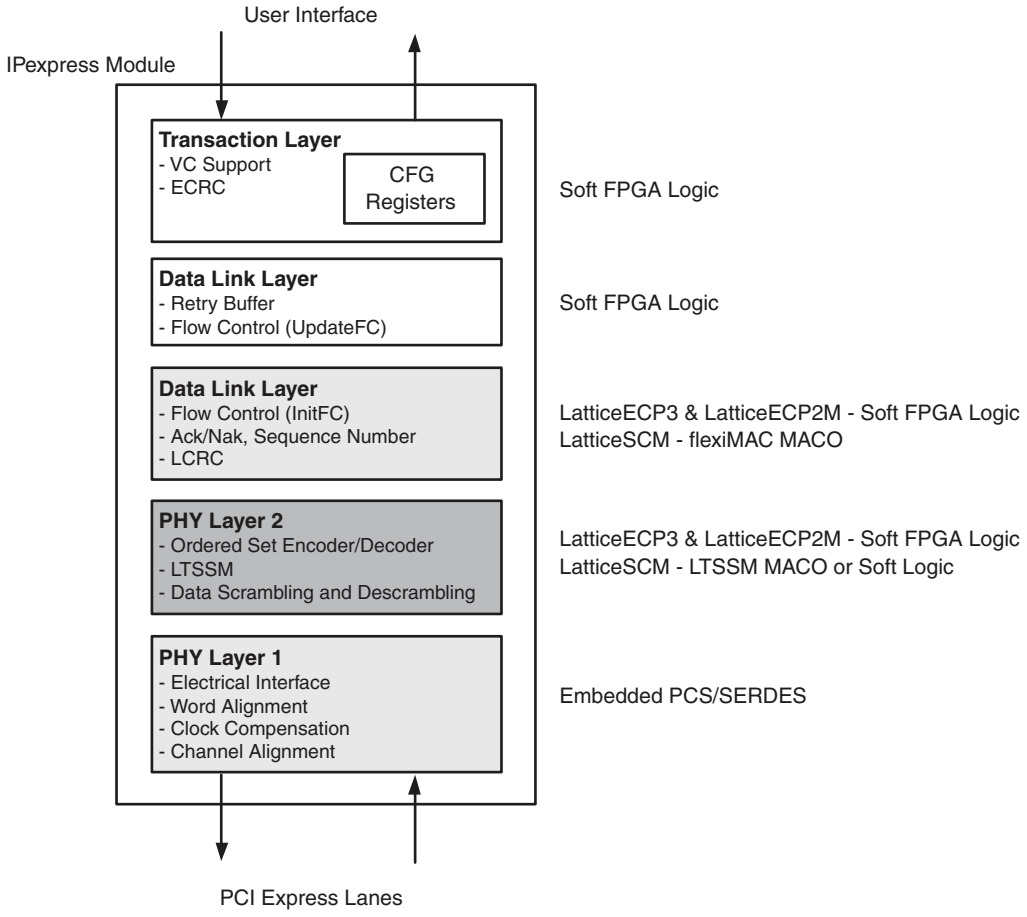
Overview

The PCI Express core is implemented in several different FPGA technologies. These technologies include soft FPGA fabric elements such as LUTs, registers, embedded block RAMs (EBRs), embedded hard elements with the PCS/SERDES, and Lattice's unique MACO technology (LatticeSCM family only).

The IPexpress™ design tool is used to customize and create a complete IP module for the user to instantiate in a design. Inside the module created by the IPexpress tool are several blocks implemented in heterogeneous technologies. All of the connectivity is provided, allowing the user to interact at the top level of the IP core.

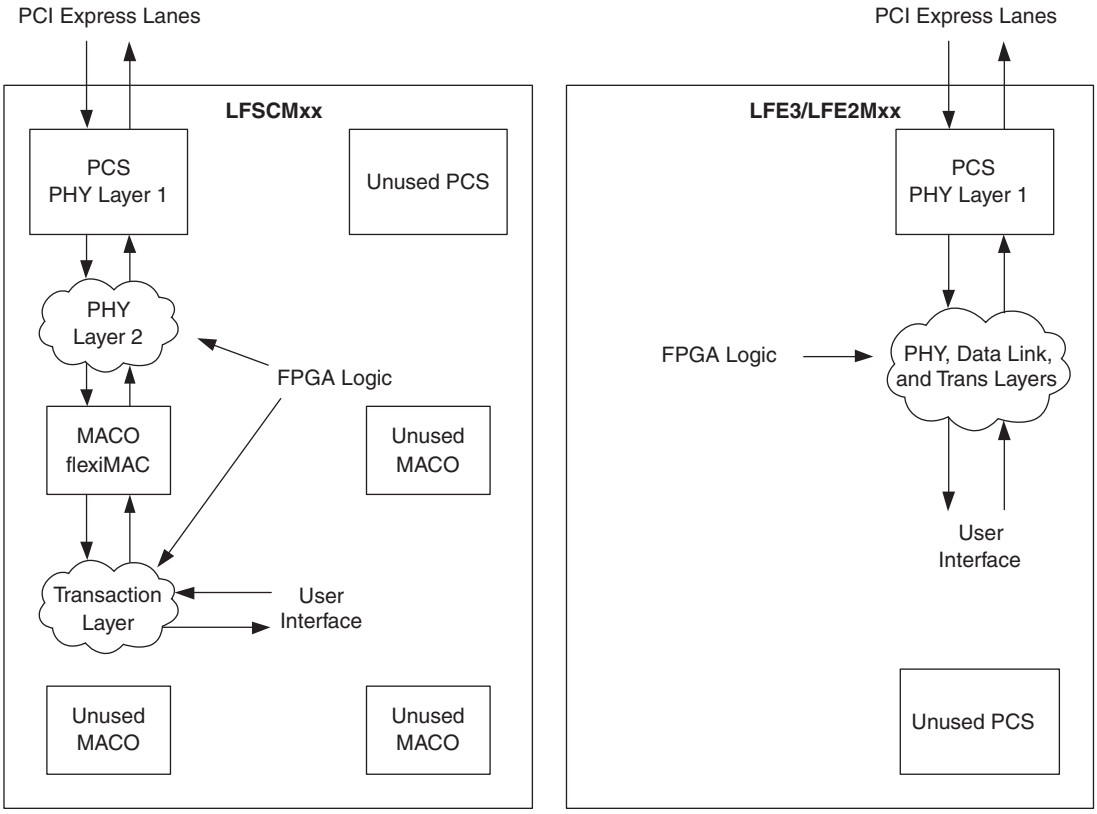
Figure 2-1 provides a high-level block diagram to illustrate the main functional blocks and the technology used to implement PCI Express functions.

Figure 2-1. PCI Express IP Core Technology and Functions



As the PCI Express core proceeds through the Diamond or ispLEVER software design flow specific technologies are targeted to their specific locations on the device. Figure 2-2 provides implementation representations of the LFSCMxx and LFE3/LFE2Mxx devices with a PCI Express core.

Figure 2-2. PCI Express Core Implementation in the LatticeSCM, LatticeECP3 and LatticeECP2M



As shown, the data flow moves in and out of the heterogeneous FPGA technology. The user is responsible for selecting the location of the hard blocks (this topic will be discussed later in this document). The FPGA logic placement and routing is the job of the Diamond or ispLEVER design tools to select regions nearby the hard blocks to achieve the timing goals.

Figure 2-3 provides a high-level interface representation.

Figure 2-3. PCI Express Interfaces

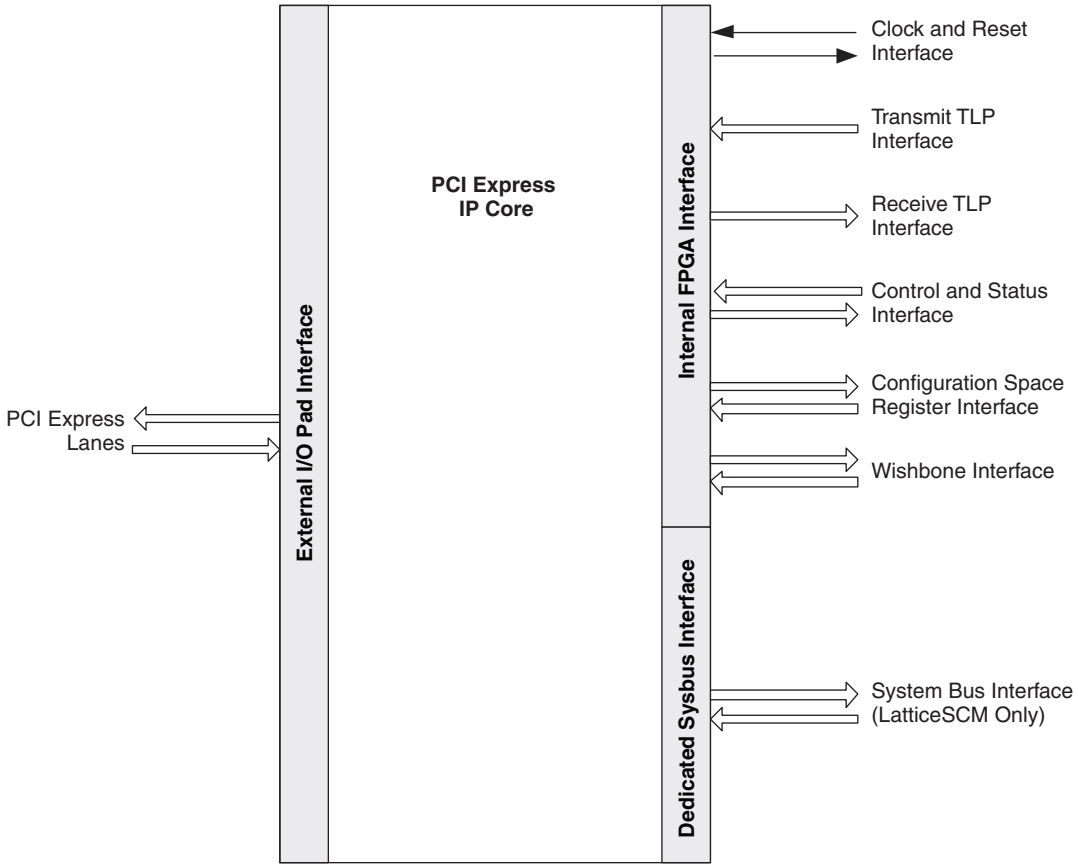


Table 2-1 provides the list of ports and descriptions for the PCI Express IP core.

Table 2-1. PCI Express IP Core Port List

Port Name	Direction	Clock	Description
Clock and Reset Interface			
refclk[p,n] (LatticeECP3 and LatticeECP2M only)	Input		100 MHz PCI Express differential reference clock used to generate the 2.5 Gbps data.
refclk_250 (LatticeSCM only)	Input		250 MHz reference clock to the Physical Layer. This clock input must use a primary clock route to reduce SERDES transmit jitter. For the LatticeSCM device, a FPGA-based PLL is used to synthesize the PCI Express reference clock to the 250 MHz required for this port. This topic is covered in the Clocking Scheme section of this document.
sys_clk_125	Output		125 MHz clock derived from refclk to be used in the user application.
rst_n	Input		Active-low asynchronous datapath and state machine reset.

Table 2-1. PCI Express IP Core Port List (Continued)

Port Name	Direction	Clock	Description
serdes_rst (Lattice SCM only)	Input		Active-high asynchronous SERDES PLL reset. This resets the SERDES. The serdes_rst can be tied low. This is not required unless the refclk changes or stops. This is typically not the case in most systems.
rx_rst (LatticeSCM only)	Input		Active-high asynchronous receive PHY layer reset. This reset only resets the PCS/SERDES receive datapath. The rx_rst must be toggled after the PCS multi-channel aligner settings have changed. The uML reference design provided and documented in the Resource Utilization section produces this signal.
clk_50 (LatticeSCM only)	Input		Control plane clock used by the flexiMAC. This clock can run up to 50MHz.
PCI Express Lanes			
hdin[p,n]_[0,1,2,3]	Input		PCI Express 2.5 Gbps CML inputs for lanes 0,1,2, and 3. The port “flip_lanes” is used to define the connection of PCS/SERDES channel to PCI Express lane. flip_lanes=0 hdin[p,n]_0 - PCI Express Lane 0 hdin[p,n]_1 - PCI Express Lane 1 hdin[p,n]_2 - PCI Express Lane 2 hdin[p,n]_3 - PCI Express Lane 3 flip_lanes=1 hdin[p,n]_0 - PCI Express Lane 3 hdin[p,n]_1 - PCI Express Lane 2 hdin[p,n]_2 - PCI Express Lane 1 hdin[p,n]_3 - PCI Express Lane 0
hdout[p,n]_[0,1,2,3]	Output		PCI Express 2.5 Gbps CML outputs for lanes 0,1,2, and 3. The port “flip_lanes” is used to define the connection of PCS/SERDES channel to PCI Express lane. flip_lanes=0 hdout[p,n]_0 - PCI Express Lane 0 hdout[p,n]_1 - PCI Express Lane 1 hdout[p,n]_2 - PCI Express Lane 2 hdout[p,n]_3 - PCI Express Lane 3 flip_lanes=1 hdout[p,n]_0 - PCI Express Lane 3 hdout[p,n]_1 - PCI Express Lane 2 hdout[p,n]_2 - PCI Express Lane 1 hdout[p,n]_3 - PCI Express Lane 0
Transmit TLP Interface			
tx_data_vc0[n:0]	Input	sys_clk_125	Native x4 and Downgraded x1 Transmit data bus [63:56] Byte N [55:48] Byte N+1 [47:40] Byte N+2 [39:32] Byte N+3 [31:24] Byte N+4 [23:16] Byte N+5 [15: 8] Byte N+6 [7: 0] Byte N+7 Native x1 Transmit data bus [15:8] Byte N [7:0] Byte N+1

Table 2-1. PCI Express IP Core Port List (Continued)

Port Name	Direction	Clock	Description
tx_req_vc0	Input	sys_clk_125	Active high transmit request. This port is asserted when the user wants to send a TLP. If several TLPs will be provided in a burst, this port can remain high until all TLPs have been sent.
tx_rdy_vc0	Output	sys_clk_125	Active high transmit ready indicator. Tx_st should be provided next clock cycle after tx_rdy is high.This port will go low between TLPs.
tx_st_vc0	Input	sys_clk_125	Active high transmit start of TLP indicator.
tx_end_vc0	Input	sys_clk_125	Active high transmit end of TLP indicator. This signal must go low at the end of the TLP.
tx_nlfy_vc0	Input	sys_clk_125	Active high transmit nullify TLP. Can occur anywhere during the TLP. If tx_nlfy_vc0 is asserted to nullify a TLP the tx_end_vc0 port should not be asserted. The tx_nlfy_vc0 terminates the TLP.
tx_dwen_vc0	Input	sys_clk_125	Active high transmit 32-bit word indicator. Used if only bits [63:32] provide valid data. This port is available only on the Native x4 and x4 Downgraded x1 cores.
tx_val	Output	sys_clk_125	Active high transmit clock enable. When a x4 is downgraded to a x1, this signal is used as the clock enable to downshift the transmit bandwidth. This port is available only on the Native x4 and x4 Downgraded x1 cores.
tx_ca_[ph,nph,cplh]_vc0[8:0]	Output	sys_clk_125	Transmit Interface credit available bus. This port will decrement as TLPs are sent and increment as UpdateFCs are received. Ph - Posted header Nph - Non-posted header Cplh - Completion header This credit interface is only updated when an UpdateFC DLLP is received from the PCI Express line. [8] - This bit indicates the receiver has infinite credits. If this bit is high then bits. [7:0] should be ignored. [7:0] - The amount of credits available at the receiver.
tx_ca_[pd,npd,cpld]_vc0[12:0]	Output	sys_clk_125	Transmit Interface credit available bus. This port will decrement as TLPs are sent and increment as UpdateFCs are received. pd - posted data npd - non-posted data cpld - completion data [12] - This bit indicates the receiver has infinite credits. If this bit is high, then bits [11:0] should be ignored. [11:0] - The amount of credits available at the receiver.
tx_ca_p_recheck_vc0	Output	sys_clk_125	Active high signal that indicates the core sent a Posted TLP which changed the tx_ca_p[h,d]_vc0 port. This might require a recheck of the credits available if the user has asserted tx_req_vc0 and is waiting for tx_rdy_vc0 to send a Posted TLP.
tx_ca_cpl_recheck_vc0	Output	sys_clk_125	Active high signal that indicates the core sent a Completion TLP which changed the tx_ca_cpl[h,d]_vc0 port. This might require a recheck of the credits available if the user has asserted tx_req_vc0 and is waiting for tx_rdy_vc0 to send a Completion TLP.

Table 2-1. PCI Express IP Core Port List (Continued)

Port Name	Direction	Clock	Description
Receive TLP Interface			
rx_data_vc0[n:0]	Output	sys_clk_125	Native x4 and Downgraded x1 Receive data bus [63:56] Byte N [55:48] Byte N+1 [47:40] Byte N+2 [39:32] Byte N+3 [31:24] Byte N+4 [23:16] Byte N+5 [15: 8] Byte N+6 [7: 0] Byte N+7 Native x1 Receive data bus [15:8] Byte N [7:0] Byte N+1
rx_st_vc0	Output	sys_clk_125	Active high receive start of TLP indicator.
rx_end_vc0	Output	sys_clk_125	Active high receive end of TLP indicator.
rx_dwen_vc0	Output	sys_clk_125	Active high 32-bit word indicator. Used if only bits [63:32] contain valid data. This port is available only on the Native x4 and x4 Downgraded x1 cores.
rx_ecrc_err_vc0	Output	sys_clk_125	Active high ECRC error indicator. Indicates a ECRC error in the current TLP. Only available if ECRC is enabled in the IPexpress tool.
rx_us_req_vc0	Output	sys_clk_125	Active high unsupported request indicator. Asserted if any of the following TLP types are received: - Memory Read Request-Locked - The TLP is still passed to the user where the user will need to terminate the TLP with an Unsupported Request Completion.
rx_malf_tlp_vc0	Output	sys_clk_125	Active high malformed TLP indicator. Indicates a problem with the current TLPs length or format.
rx_bar_hit[6:0]	Output	sys_clk_125	Active high BAR indicator for the current TLP. If this bit is high the current TLP on the receive interface is in the address range of the defined BAR. [6] - Expansion ROM [5] - BAR5 [4] - BAR4 [3] - BAR3 [2] - BAR2 [1] - BAR1 [0] - BAR0 For 64-bit BARs, a BAR hit will be indicated on the lower BAR number. The rx_bar_hit changes along with the rx_st_vc0 signal.
ur_np_ext	Input	sys_clk_125	Active high indicator for unsupported non-posted request reception.
ur_p_ext	Input	sys_clk_125	Active high indicator for unsupported posted request reception.
[ph,pd, nph,npd] _buf_status_vc0	Input	sys_clk_125	Active high user buffer full status indicator. When asserted, an UpdateFC will be sent for the type specified as soon as possible without waiting for the UpdateFC timer to expire.
[ph,nph]_processed_vc0	Input	sys_clk_125	Active high indicator to inform the IP core of how many credits have been processed. Each clock cycle high counts as one credit processed. The core will generate the required UpdateFC DLLP when either the UpdateFC timer expires or enough credits have been processed.

Table 2-1. PCI Express IP Core Port List (Continued)

Port Name	Direction	Clock	Description
[pd,npd]_processed_vc0	Input	sys_clk_125	Active high enable for [pd, npd]_num_vc0 port. The user should place the number of data credits processed on the [pd, npd]_num_vc0 port and then assert [pd, npd]_processed_vc0 for one clock cycle. The core will generate the required UpdateFC DLLP when either the UpdateFC timer expires or enough credits have been processed.
[pd,npd]_num_vc0[7:0]	Input	sys_clk_125	This port provides the number of PD or NPD credits processed. It is enabled by the [pd, npd]_processed_vc0 port.
Control and Status			
PHYSICAL LAYER			
no_pcie_train	Input	Async	Active high signal disables LTSSM training and forces the LTSSM to L0 as a x4 configuration. This is intended to be used in simulation only to force the LTSSM into the L0 state.
force_lsm_active	Input	Async	Forces the Link State Machine for all of the channels to the linked state.
force_rec_ei	Input	Async	Forces the detection of a received electrical idle.
force_phy_status	Input	Async	Forces the detection of a receiver during the LTSSM Detect state on all of the channels.
force_disable_scr	Input	Async	Disables the PCI Express TLP scrambler.
hl_snd_beacon	Input	sys_clk_125	Active high request to send a beacon.
hl_disable_scr	Input	Async	Active high to set the disable scrambling bit in the TS1/TS2 sequence.
hl_gto_dis	Input	Async	Active high request to go to Disable state when LTSSM is in Config or Recovery.
hl_gto_det	Input	sys_clk_125	Active high request to go to Detect state when LTSSM is in L2 or Disable.
hl_gto_hrst	Input	Async	Active high request to go to Hot Reset when LTSSM is in Recovery.
hl_gto_l0stx	Input	sys_clk_125	Active high request to go to L0s when LTSSM is in L0.
hl_gto_l0stxfts	Input	sys_clk_125	Active high request to go to L0s and transmit FTS when LTSSM is in L0s.
hl_gto_l1	Input	sys_clk_125	Active high request to go to L1 when LTSSM is in L0.
hl_gto_l2	Input	sys_clk_125	Active high request to go to L2 when LTSSM is in L0.
hl_gto_lbk[3:0]	Input	sys_clk_125	Active high request to go to Loopback when LTSSM is in Config or Recovery.
hl_gto_rcvry	Input	sys_clk_125	Active high request to go to Recovery when LTSSM is in L0, L0s or L1.
hl_gto_cfg	Input	sys_clk_125	Active high request to go to Config when LTSSM is in Recovery.
phy_ltssm_state[3:0]	Output	sys_clk_125	PHY Layer LTSSM current state 0000 - Detect 0001 - Polling 0010 - Config 0011 - L0 0100 - L0s 0101 - L1 0110 - L2 0111 - Recovery 1000 - Loopback 1001 - Hot Reset 1010 - Disabled

Table 2-1. PCI Express IP Core Port List (Continued)

Port Name	Direction	Clock	Description
phy_ltssm_substate[2:0]	Output	sys_clk_125	PHY Layer LTSSM current substate. Each major LTSSM state has a series of substates. When phy_ltssm_state=DETECT 000 - DET_WAIT 001 - DET_QUIET 010 - DET_GODET1 011 - DET_ACTIVE1 100 - DET_WAIT12MS 101 - DET_GODET2 110 - DET_ACTIVE2 111 - DET_EXIT When phy_ltssm_state=POLLING 000 - POL_WAIT 001 - POL_ACTIVE 010 - POL_COMPLIANCE 011 - POL_CONFIG 100 - POL_EXIT When phy_ltssm_state=CONFIG 000 - CFG_WAIT 001 - CFG_LINK_WIDTH_ST 010 - CFG_LINK_WIDTH_ACC 011 - CFG_LANE_NUM_WAIT 100 - CFG_LANE_NUM_ACC 101 - CFG_COMPLETE 110 - CFG_IDLE 111 - CFG_EXIT When phy_ltssm_state=L0 000 - L0_WAIT 001 - L0_L0 010 - L0_L0RX 011 - L0_L0TX 100 - L0_EIDLE_0 101 - L0_EIDLE_1 110 - L0_EXIT When phy_ltssm_state=L0s 000 - L0s_RX_WAIT 001 - L0s_RX_ENTRY 010 - L0s_RX_IDLE 011 - L0s_RX_FTS 100 - L0s_RX_EXIT When phy_ltssm_state=L1 000 - L1_WAIT 001 - L1_ENTRY 010 - L1_IDLE 011 - L1_EXIT When phy_ltssm_state=L2 000 - L2_WAIT 001 - L2_IDLE 001 - L2_EXIT When phy_ltssm_state=Recovery 000 - RCVRY_WAIT 001 - RCVRY_RCVRLK 010 - RCVRY_RCVRCFG 011 - RCVRY_IDLE 100 – RCVRY_EXIT

Table 2-1. PCI Express IP Core Port List (Continued)

Port Name	Direction	Clock	Description
phy_cfgln[3:0]	Output	sys_clk_125	Active high LTSSM Config state link status. An active bit indicates the channel is included in the configuration link width negotiation. [0] - PCI Express Lane 3 [1] - PCI Express Lane 2 [2] - PCI Express Lane 1 [3] - PCI Express Lane 0
phy_cfgln_sum[2:0]	Output	sys_clk_125	Link Width 000 - No link defined 001 - Link width = 1 100 - Link width = 4
phy_pol_compliance	Output	sys_clk_125	Active high indicator that the LTSSM is in the Polling.Compliance state.
phy_mloopback (LatticeSCM only)	Output	sys_clk_125	PHY Layer LTSSM is in the Master Loopback mode.
phy_sloopback (LatticeSCM only)	Output	sys_clk_125	PHY Layer LTSSM is in the Slave Loopback mode.
phy_l0s_tx_state[2:0] (LatticeSCM only)	Output	sys_clk_125	PHY Layer LTSSM L0s state 000 - Wait 001 - Electrical Idle 010 - Entry 011 - Idle 100 - FTS 101 - Exit
phy_l1_state[1:0] (LatticeSCM only)	Output	sys_clk_125	PHY Layer LTSSM L1 state 00 - Wait 01 - Entry 10 - Idle 11 - Exit
phy_l2_state[1:0] (LatticeSCM only)	Output	sys_clk_125	PHY Layer LTSSM L2 state 00 - Wait 01 - Idle 10 - Transmit Wake 11 - Exit
phy_realign_req (LatticeSCM only)	Output	sys_clk_125	Active high signal to user to issue a PCS multi channel aligner realignment. This is used by the reference design uML in the Appendix. This signal should be gated with the phy_ltssm_state[3:0] being in the Config state.
phy_snd_beacon (LatticeSCM only)	Output	sys_clk_125	PHY Layer is sending a beacon.
lsm_status_[0,1,2,3] (LatticeSCM only)	Output	sys_clk_125	Active high Link State Machine link status: Lsm_status_0 - PCI Express Lane 0 Lsm_status_1 - PCI Express Lane 1 Lsm_status_2 - PCI Express Lane 2 Lsm_status_3 - PCI Express Lane 3
flip_lanes	Input	Async	Reverses the lane connections to the PCS/SERDES. This function is used to provide flexibility for the PCB layout. The “Locating” section later in this document describes how this function can be used. 0 - Lane 0 connects to SERDES Channel 0, etc. 1 - Lane 0 connects to SERDES Channel 3, etc.
sys_clk_250 (LatticeSCM only)	Output		This output clock is used to transmit and receive data using the Master Loopback datapath. This port is only available if the Master Loopback feature is enabled in the IPexpress tool.
tx_lbk_rdy	Output	sys_clk_250	This output port is used to enable the transmit master loopback data. This port is only available if the Master Loopback feature is enabled in the IPexpress tool.

Table 2-1. PCI Express IP Core Port List (Continued)

Port Name	Direction	Clock	Description
tx_lbk_kcntl[3:0]	Input	sys_clk_250	This input port is used to indicate a K control word is being sent on tx_lbk_data port. This port is only available if the Master Loopback feature is enabled in the IPexpress tool. [3] - K control on tx_lbk_data[31:24] [2] - K control on tx_lbk_data[23:16] [1] - K control on tx_lbk_data[15:8] [0] - K control on tx_lbk_data[7:0]
tx_lbk_data[31:0]	Input	sys_clk_250	This input port is used to send 32-bit data for the master loopback. This port is only available if the Master Loopback feature is enabled in the IPexpress tool. [31:24] - Lane 3 data [23:16] - Lane 2 data [15:8] - Lane 1 data [7:0] - Lane 0 data
rx_lbk_kcntl[3:0]	Output	sys_clk_250	This output port is used to indicate a K control word is being received on rx_lbk_data port. This port is only available if the Master Loopback feature is enabled in the IPexpress tool. [3] - K control on rx_lbk_data[31:24] [2] - K control on rx_lbk_data[23:16] [1] - K control on rx_lbk_data[15:8] [0] - K control on rx_lbk_data[7:0]
rx_lbk_data[31:0]	Output	sys_clk_250	This output port is used to receive 32-bit data for the master loopback. This port is only available if the Master Loopback feature is enabled in the IPexpress tool. [31:24] - Lane 3 data [23:16] - Lane 2 data [15:8] - Lane 1 data [7:0] - Lane 0 data
prog_done (LatticeSCM only)	Input	sys_clk_125	Active high input to indicate the config state can continue to L0. This input signal is used to facilitate dynamic link width. The LTSSM will wait in the Configuration Complete state until this port is asserted. This allows the PCS Multi-Channel Aligner settings to be set to match the configured link width. Once the MCA has been programmed, the prod_done port should be asserted high. The uML reference design documented in Appendix A provides the control for this port.
DATA LINK LAYER			
dl_inactive	Output	Async	Data Link Layer is the DL_Inactive state.
dl_init	Output	Async	Data Link Layer is in the DL_Init state.
dl_active	Output	Async	Data Link Layer is in the DL_Active state.
dl_up	Output	Async	Data Link Layer is in the DL_Active state and is now providing TLPs to the Transaction Layer.
tx_dllp_val	Input	sys_clk_125	Active high power message send command. 00 - Nothing to send 01 - Send DLLP using tx_pmtype DLLP 10 - Send DLLP using tx_vsd_data Vendor Defined DLLP 11 - Not used
tx_pmtype[2:0]	Input	sys_clk_125	Transmit power message type 000 - PM Enter L1 001 - PM Enter L2 011 - PM Active State Request L1 100 - PM Request Ack
tx_vsd_data[23:0]	Input	sys_clk_125	Vendor-defined data to send in DLLP.
tx_dllp_sent	Output	sys_clk_125	Requested DLLP was sent.

Table 2-1. PCI Express IP Core Port List (Continued)

Port Name	Direction	Clock	Description
rxdp_pmd_type[2:0]	Output	sys_clk_125	Receive power message type 000 - PM Enter L1 001 - PM Enter L2 011 - PM Active State Request L1 100 - PM Request Ack
rxdp_vsd_data[23:0]	Output	sys_clk_125	Vendor-defined DLLP data received.
rxdp_dllp_val	Output	sys_clk_125	Active high power message received
tx_rbuf_empty	Output	sys_clk_125	Transmit retry buffer is empty. (Used for ASPM implementation outside the core.)
tx_dllp_pend	Output	sys_clk_125	A DLLP is pending to be transmitted. (Used for ASPM implementation outside the core.)
rx_tlp_rcvd	Output	sys_clk_125	A TLP was received. (Used for ASPM implementation outside the core.)
TRANSACTION LAYER			
ecrc_gen_enb	Input or Output	Async	If AER and ECRC are enabled then this port is an output and indicates when ECRC generation is enabled by the PCI Express IP core. If ECRC is enabled, but AER is not enabled then this port in an input and is used to turn on ECRC generation. If ECRC generation is turned on then the TD bit in the transmit TLP header must be set to provide room in the TLP for the insertion of the ECRC.
ecrc_chk_enb	Input or Output	Async	If AER and ECRC are enabled then this port is an output and indicates when ECRC checking is enabled by the PCI Express IP core. If ECRC is enabled, but AER is not enabled then this port in an input and is used to turn on ECRC checking.
cmpln_tout	Input	sys_clk_125	Completion Timeout Indicator for posted request. Used to force non-fatal error message generation and also set appropriate bit in AER.
cmpltr_abort_np	Input	sys_clk_125	Complete or Abort Indicator for non-posed request. Used to force non-fatal error message generation and also set appropriate bit in AER.
cmpltr_abort_p	Input	sys_clk_125	Complete or Abort Indicator. Used to force non-fatal error message generation and also set appropriate bit in AER.
unexp_cmpln	Input	sys_clk_125	Unexpected Completion Indicator. Used to force non-fatal error message generation and also set appropriate bit in AER.
np_req_pend	Input	sys_clk_125	Sets device Status[5] indicating that a Non-Posted transaction is pending.
err_tlp_header[127:0]	Input	Async	Advanced Error Reporting errored TLP header. This port is used to provide the TLP header for the TLP associated with a unexp_cmpln or cmpltr_abort_np/cmpltr_abort_p. The header data should be provided on the same clock cycle as the unexp_cmpln or cmpltr_abort_np/cmpltr_abort_p.
CONFIGURATION REGISTERS			
bus_num[7:0]	Output	sys_clk_125	Bus Number supplied with configuration write.
dev_num[4:0]	Output	sys_clk_125	Device Number supplied with configuration write.
func_num[2:0]	Output	sys_clk_125	Function Number supplied with configuration write.

Table 2-1. PCI Express IP Core Port List (Continued)

Port Name	Direction	Clock	Description
cmd_reg_out[5:0]	Output	sys_clk_125	PCI Type0 Command Register bits [5] - Interrupt Disable [4] - SERR# Enable [3] - Parity Error Response [2] - Bus Master [1] - Memory Space [0] - IO Space
dev_cntl_out[14:0]	Output	sys_clk_125	PCI Express Capabilities Device Control Register bits [14:0].
lnk_cntl_out[7:0]	Output	sys_clk_125	PCI Express Capabilities Link Control Register bits [7:0].
inta_n	Input	sys_clk_125	Legacy INTA interrupt request. Falling edge will produce a ASSERT_INTA message and set the Interrupt Status bit to a 1. Rising edge will produce a DEASSERT_INTA message and clear the Interrupt Status bit. The Interrupt Disable bit will disable the message to be sent, but the status bit will operate as normal. The inta_n port has a requirement for how close an assert or deassert event can be to the previous assert or deassert event. For the x4 and x1 downgraded cores, this is two sys_clk_125 clock cycles. For the x1 core this is eight sys_clk_125 clock cycles. If the inta_n port is low indicating an ASSERT_INTA and the Interrupt Disable bit is driven low by the system, then the inta_n port needs to be pulled high to send a DEASSERT_INTA message. This can be automatically performed by using a logic OR between the inta_n and cmd_reg_out[5] port. Note: Only one Legacy interrupt INTA is supported.
msi[7:0]	Input	sys_clk_125	MSI interrupt request. Rising edge on a bit will produce a MemWr TLP for a MSI interrupt for the provided address and data by the root complex. [7] - MSI 8 [6] - MSI 7 [5] - MSI 6 [4] - MSI 5 [3] - MSI 4 [2] - MSI 3 [1] - MSI 2 [0] - MSI 1
flr_rdy_in	Input	sys_clk_125	Ready from user logic to perform Functional Level Reset
initiate_flr	Output	sys_clk_125	Initiate Functional Level Reset for user logic
dev_cntl_2_out	Output	sys_clk_125	PCI Express Capabilities Device Control 2 Register Bits [4:0]
mm_enable[2:0]	Output	Async	Multiple MSI interrupts are supported by the root complex. This indicates how many messages the root complex will accept.
msi_enable	Output	Async	MSI interrupts are enabled by the root complex. When this port is high MSI interrupts are to be used. The inta_n port is disabled.
pme_status	Input	Async	Active high input to the Power Management Capability Structure PME_Status bit. Indicates that a Power Management Event has occurred on the endpoint.
pme_en	Output	Async	PME_En bit in the Power Management Capability Structure. Active high signal to allow the endpoint to send PME messages.

Table 2-1. PCI Express IP Core Port List (Continued)

Port Name	Direction	Clock	Description
pm_power_state[1:0]	Output	Async	Power State in the Power Management Capability Structure. Software sets this state to place the endpoint in a particular state. 00 - D0 01 - D1 10 - D2 11 - D3
load_id	Input	Async	This port is only present when the “Load IDs from Ports” checkbox is enabled in the IPexpress tool. When this port is low, the core will send Configuration Request Retry Status for all Configuration Requests. When this port is high, the core will send normal Successful Completions for Configuration Requests. On the rising edge of load_id the vectors on vendor_id[15:0], device_id[15:0], rev_id[7:0], class_code[23:0], subsys_ven_id[15:0], and subsys_id[15:0] will be loaded into the proper configuration registers.
vendor_id[15:0]	Input	Async	This port is only present when the “Load IDs from Ports” checkbox is enabled in the IPexpress tool. This port will load the vendor ID for the core on the rising edge of load_id.
device_id[15:0]	Input	Async	This port is only present when the “Load IDs from Ports” checkbox is enabled in the IPexpress tool. This port will load the device ID for the core on the rising edge of load_id.
rev_id[7:0]	Input	Async	This port is only present when the “Load IDs from Ports” checkbox is enabled in the IPexpress tool. This port will load the revision ID for the core on the rising edge of load_id.
class_code[23:0]	Input	Async	This port is only present when the “Load IDs from Ports” checkbox is enabled in the IPexpress tool. This port will load the class code for the core on the rising edge of load_id.
subsys_ven_id[15:0]	Input	Async	This port is only present when the “Load IDs from Ports” checkbox is enabled in the IPexpress tool. This port will load the subsystem vendor ID for the core on the rising edge of load_id.
subsys_id[15:0]	Input	Async	This port is only present when the “Load IDs from Ports” checkbox is enabled in the IPexpress tool. This port will load the subsystem device ID for the core on the rising edge of load_id.
Wishbone Interface[1]			
CLK_I	Input		Wishbone interface clock.
RST_I	Input	CLK_I	Asynchronous reset.
SEL_I [3:0]	Input	CLK_I	Data valid indicator [3] - DAT_I[31:24] [2] - DAT_I[23:16] [1] - DAT_I[15:8] [0] - DAT_i[7:0]
WE_I	Input	CLK_I	Write enable 1 - write 0 - read
STB_I	Input	CLK_I	Strobe input.
CYC_I	Input	CLK_I	Cycle input.
DAT_I[31:0]	Input	CLK_I	Data input.
ADR_I[12:0]	Input	CLK_I	Address input.

Table 2-1. PCI Express IP Core Port List (Continued)

Port Name	Direction	Clock	Description
CHAIN_RDAT_in[31:0]	Input	CLK_I	Daisy chain read data. If using a read chain for the wishbone interface, this would be the read data from the previous slave. If not using a chain, then this port should be tied low.
CHAIN_ACK_in	Input	CLK_I	Daisy chain ack. If using a read chain for the wishbone interface, this would be the ack from the previous slave. If not using a chain, then this port should be tied low.
ACK_O	Output	CLK_I	Ack output.
IRQ_O	Output	CLK_I	Interrupt output. This port is not used (always 0).
DAT_O[31:0]	Output	CLK_I	Data output.
LatticeSCM System Bus Interface (These optional ports are required if run time access of the PCS/SERDES is required)			
sysbus_in[44:0]	Input		System bus interface for the PCS/SERDES block inside the IP core.
sysbus_out[16:0]	Output		System bus interface for the PCS/SERDES block inside the IP core.

1. Complete information on the Wishbone interface specification can be found at www.opencores.org in the WISHBONE System-on-Chip (SOC) Interconnection Architecture for Portable IP Cores specification.

Interface Description

This section describes the datapath user interfaces of the IP core. Both the transmit and receive interfaces use the TLP as the data structure. The lower layers attach the start, end, sequence number and crc.

Transmit TLP Interface

In the transmit direction, the user must first check the credits available on the far end before sending the TLP. This information is found on the tx_ca_[ph,pd,nph,npd]_vc0 bus. There must be enough credits available for the entire TLP to be sent.

The user must then check that the core is ready to send the TLP. This is done by asserting the tx_req_vc0 port and waiting for the assertion of tx_rdy_vc0. While waiting for tx_rdy_vc0, if tx_ca_p/cpl_recheck is asserted, then the user must check available credit again. If there is enough credit, the user can proceed with the sending data based on tx_rdy_vc0. If the credit becomes insufficient, tx_req_vc0 must be deasserted on the next clock until enough credit is available. When tx_rdy_vc0 is asserted the next clock cycle will provide the first 64-bit word of the TLP and assert tx_st_vc0.

Tx_rdy_vc0 will remain high until one clock cycle before the last clock cycle of TLP data (based on the length field of the TLP). This allows the tx_rdy_vc0 to be used as the read enable of a non-pipelined FIFO.

Transmit TLP Interface Waveforms for Native x4 and x4 Downgraded x1 Cores

Figure 2-4 through Figure 2-11 provide timing diagrams for the tx interface signals with a 64-bit datapath.

Figure 2-4. Transmit Interface x4, 3DW Header, 1 DW Data

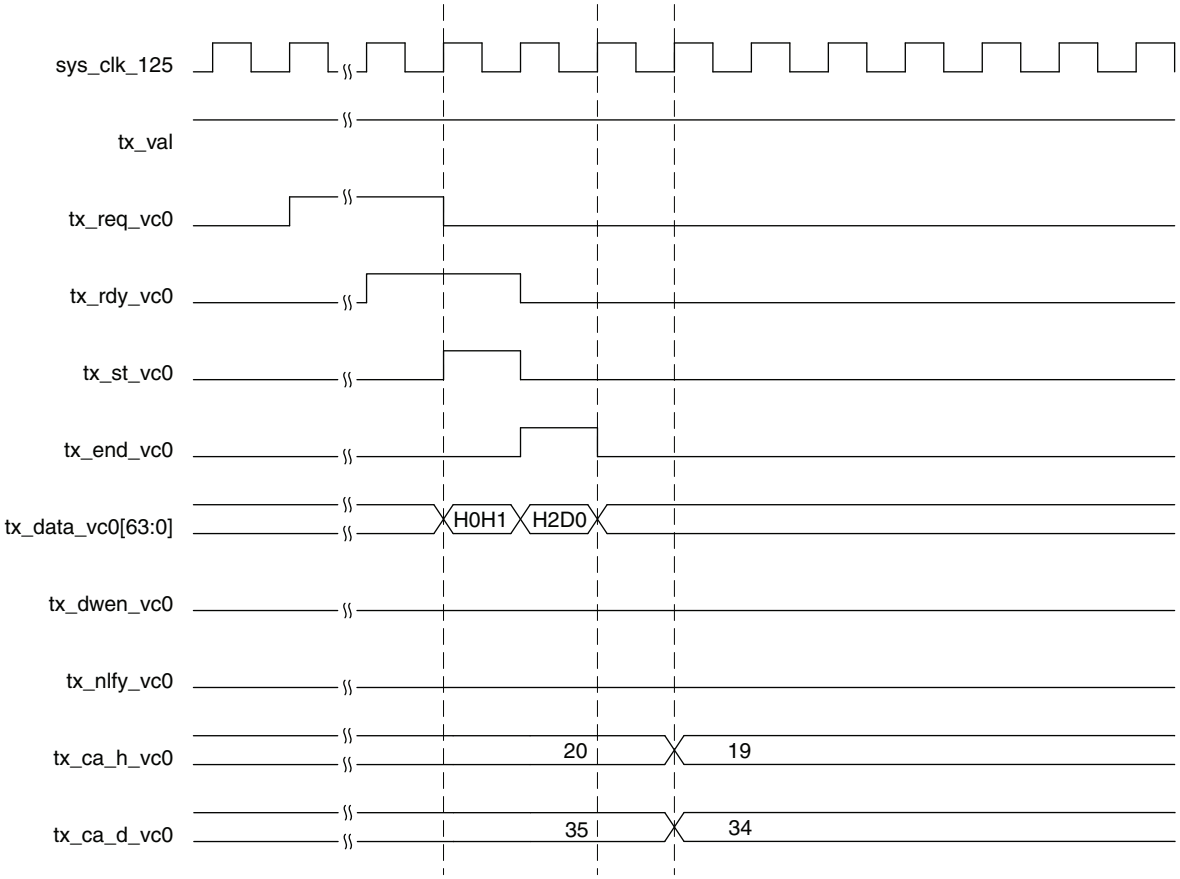


Figure 2-5. Transmit Interface x4, 3DW Header, 2 DW Data

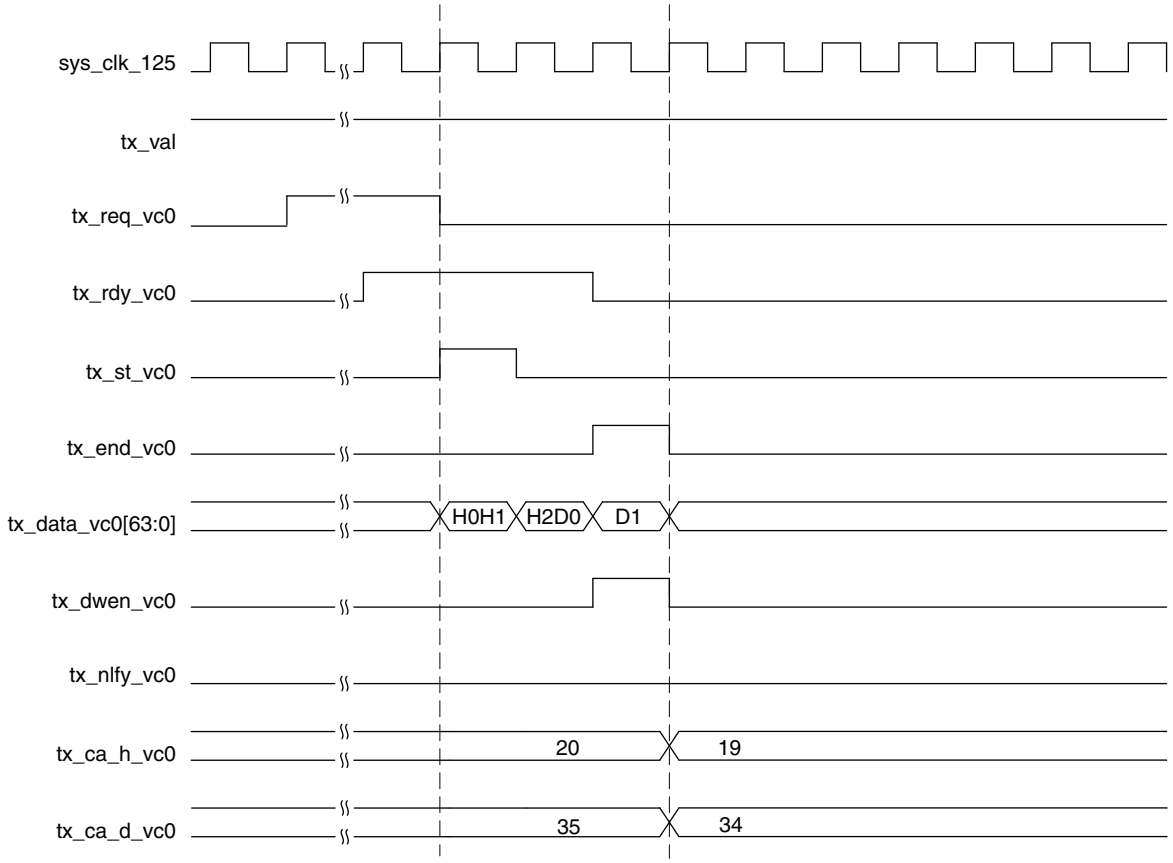


Figure 2-6. Transmit Interface x4, 4DW Header, 0 DW

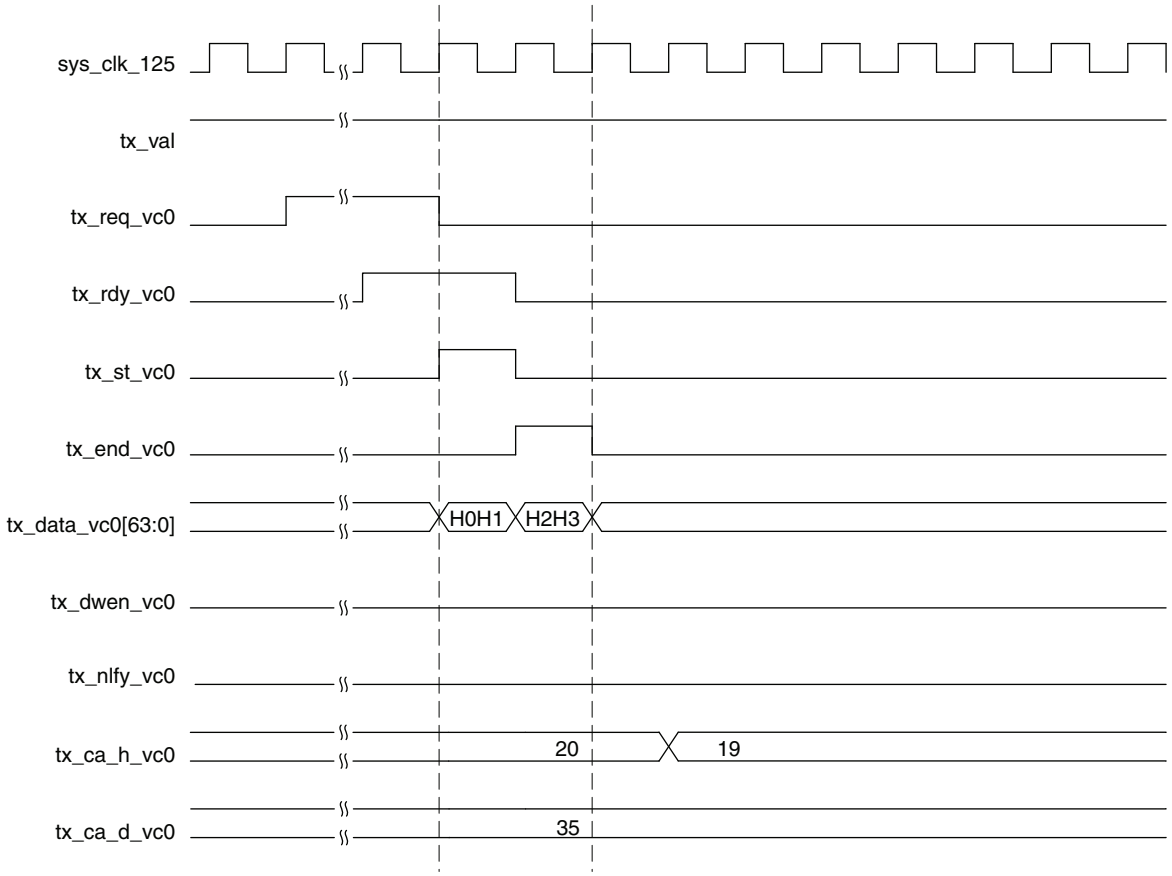


Figure 2-7. Transmit Interface x4, 4DW Header, Odd Number of DWs

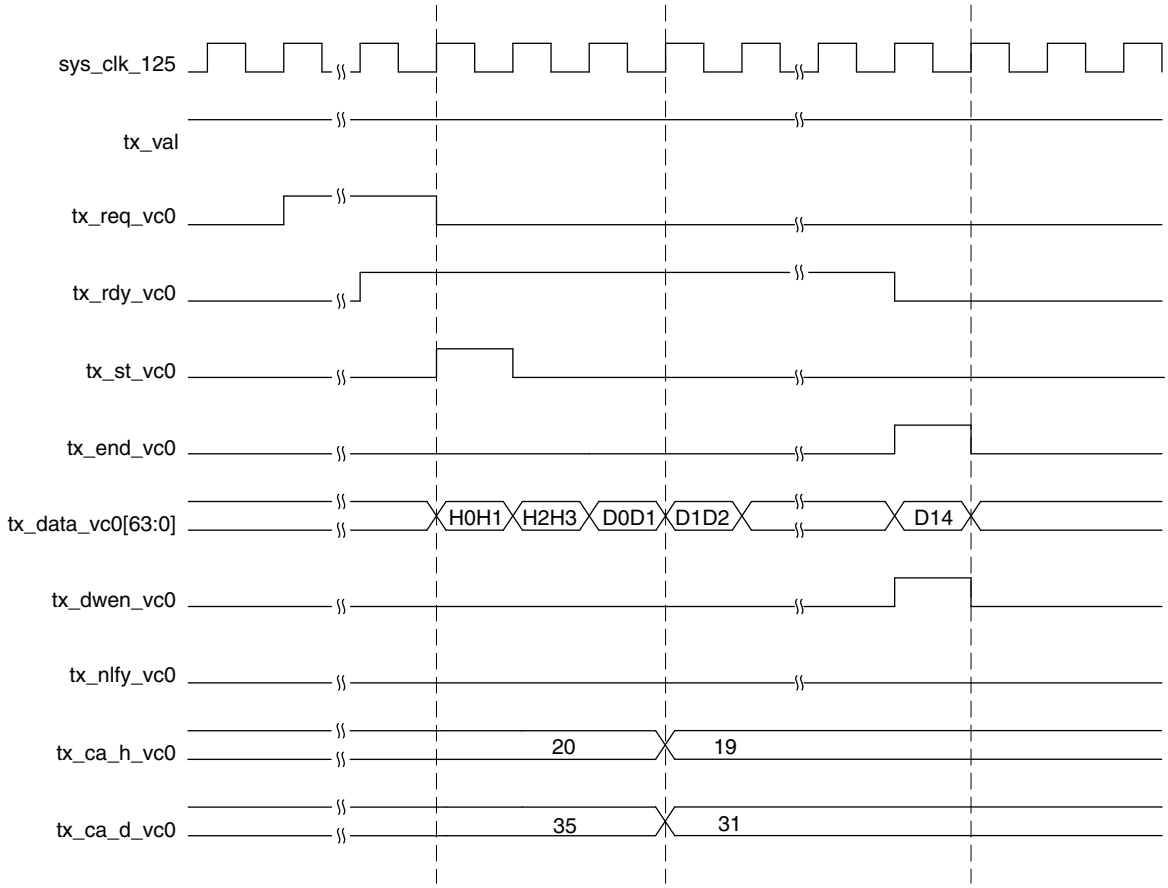


Figure 2-8. Transmit Interface x4, Burst of Two TLPs

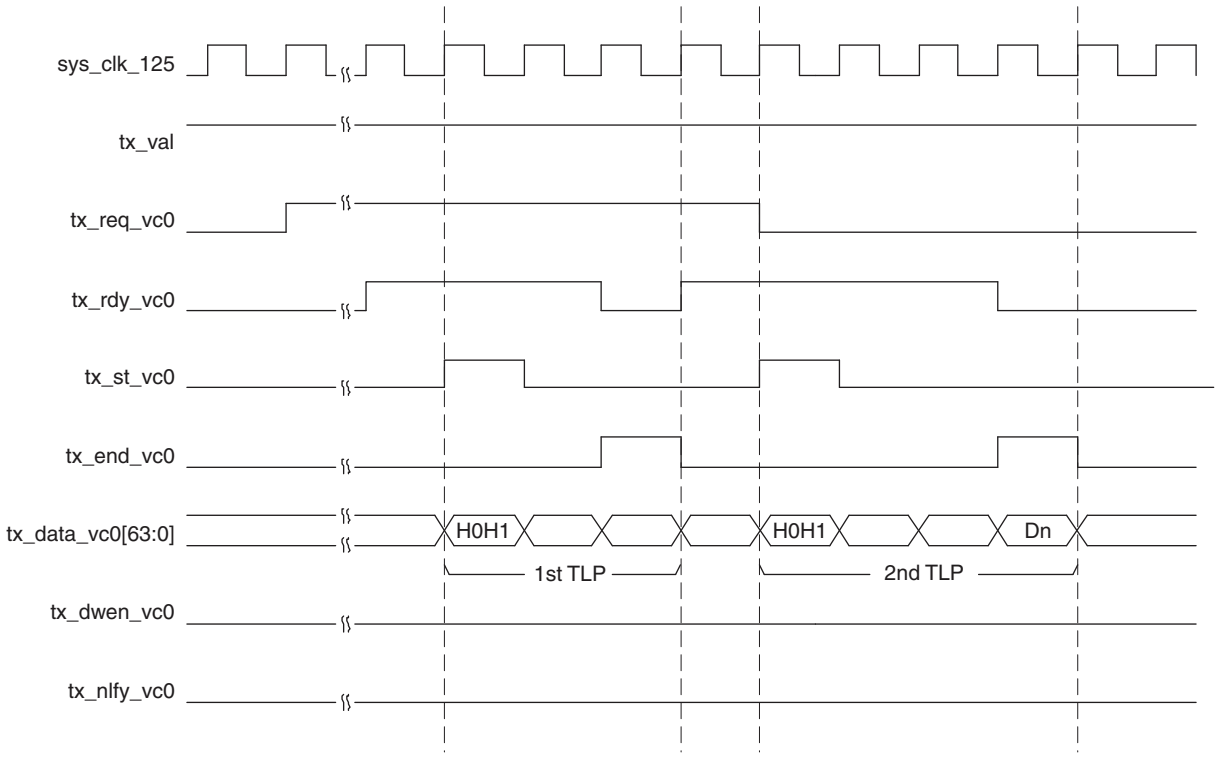


Figure 2-9. Transmit Interface x4, Nullified TLP

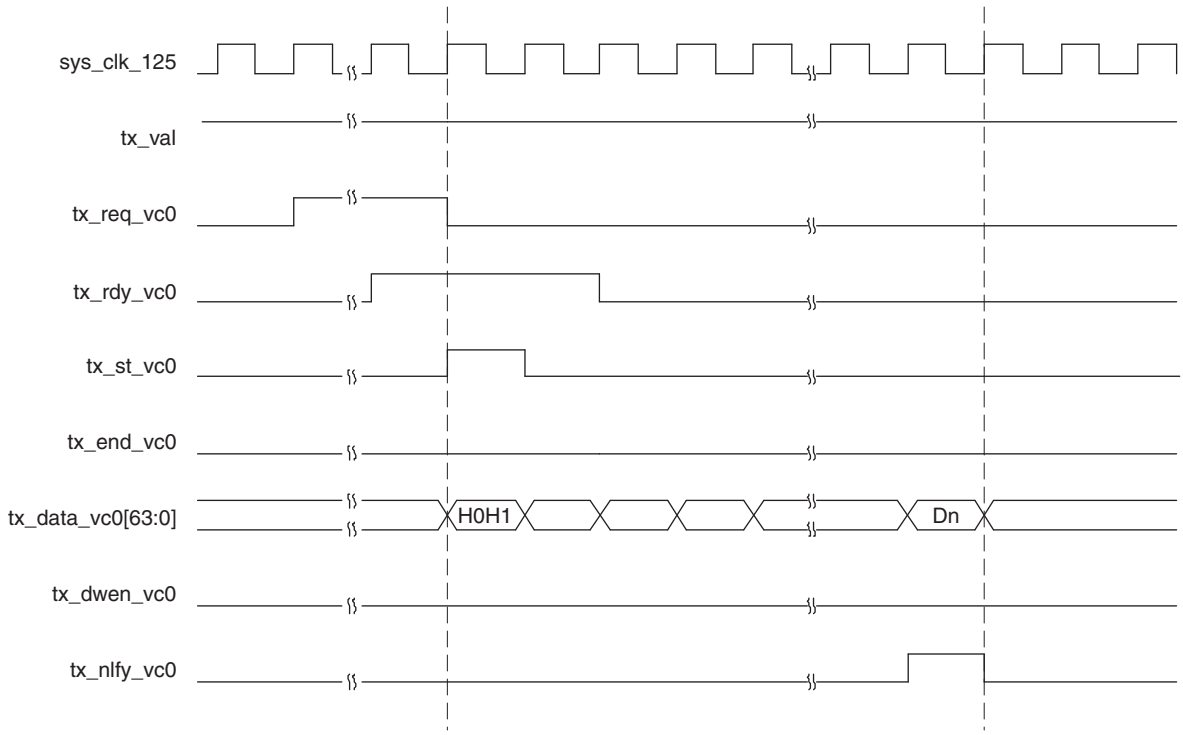


Figure 2-10. Transmit Interface x1 Downgrade Using tx_val

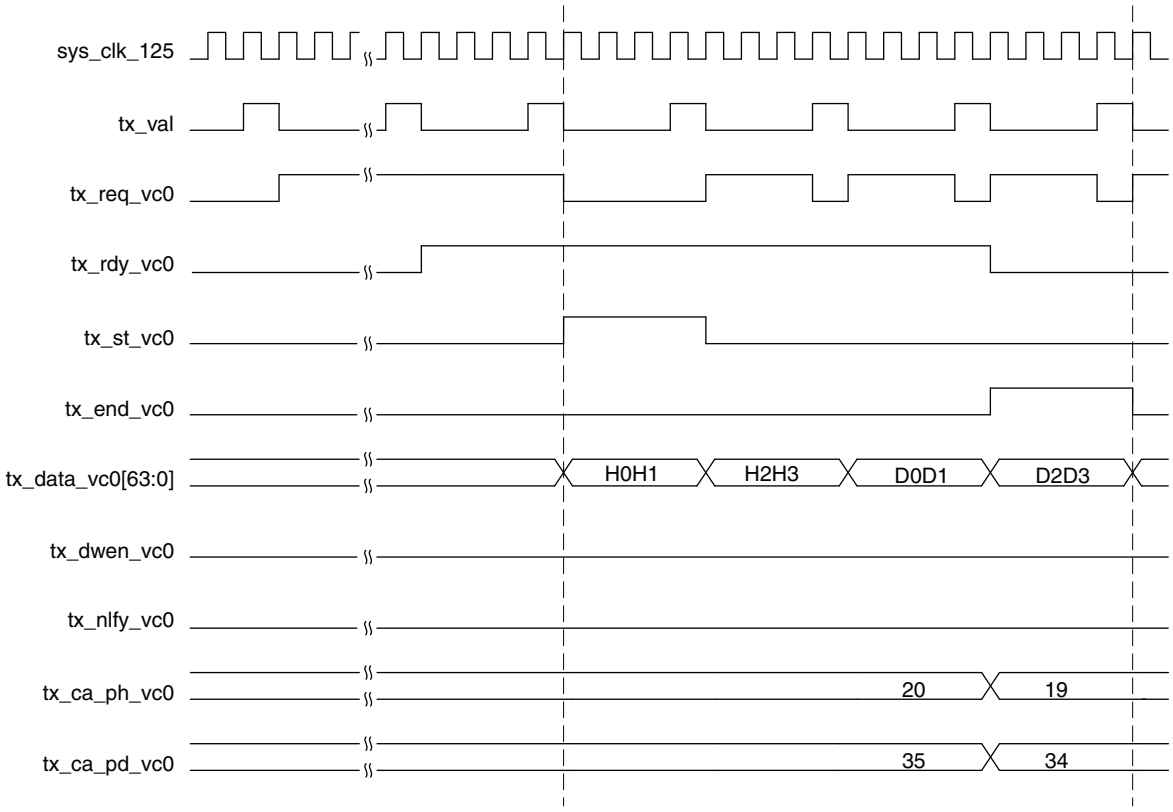
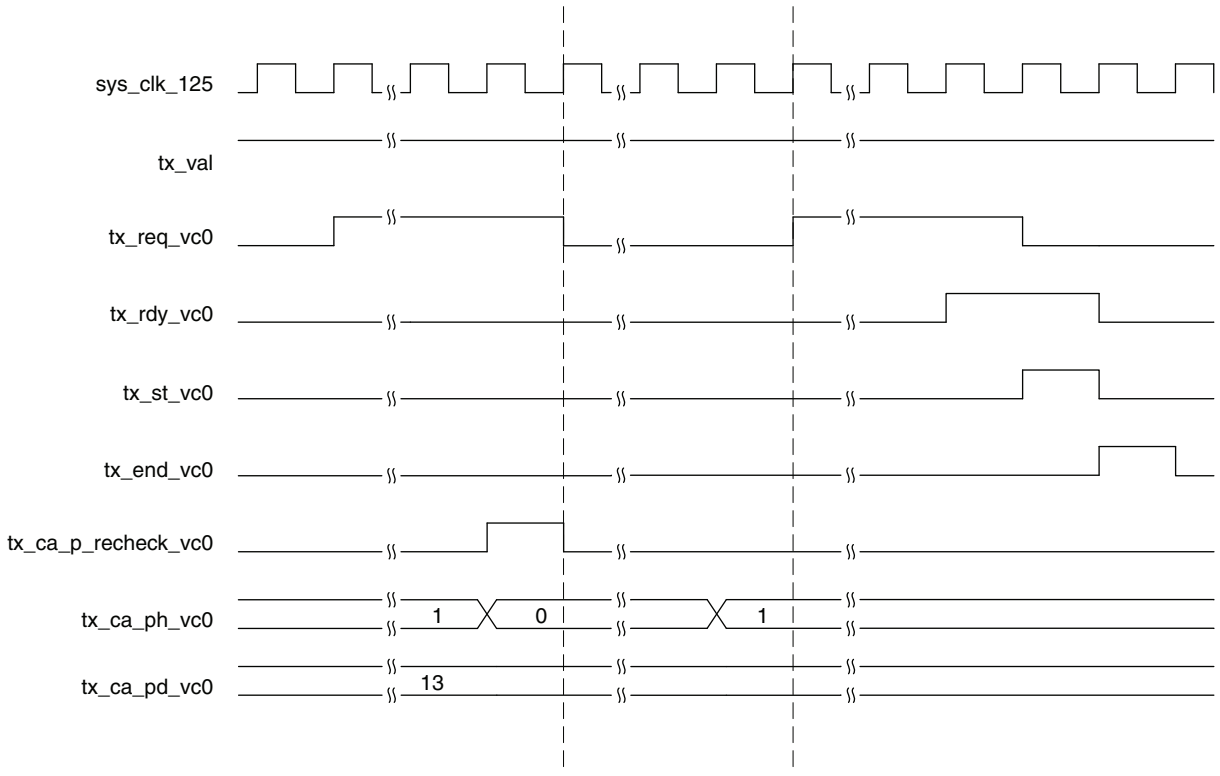


Figure 2-11. Transmit Interface x4 Posted Request with tx_ca_p-recheck Assertion



Transmit TLP Interface Waveforms for Native x1

Figure 2-12 through Figure 2-15 provide timing diagrams for the transmit interface signals with a 16-bit datapath.

Figure 2-12. Transmit Interface Native x1, 3DW Header, 1 DW Data

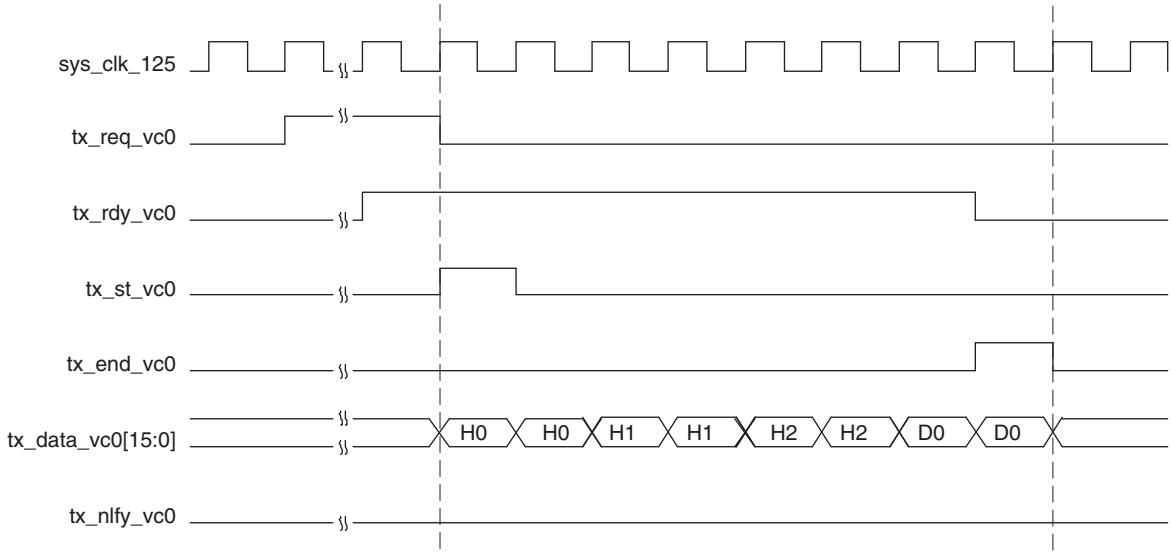


Figure 2-13. Transmit Interface Native x1, Burst of Two TLPs

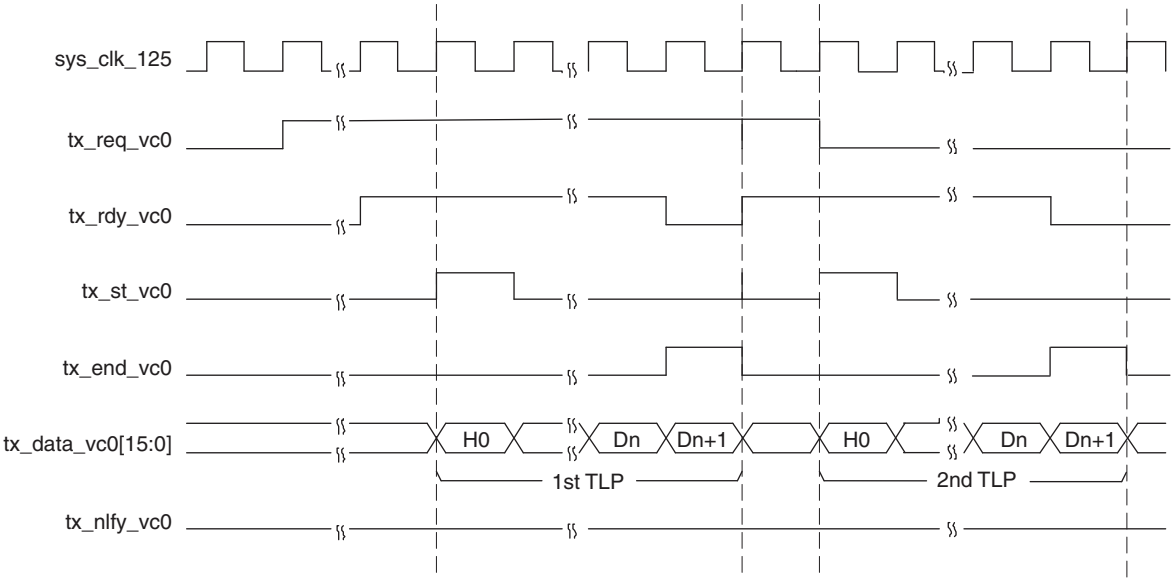


Figure 2-14. Transmit Interface Native x1, Nullified TLP

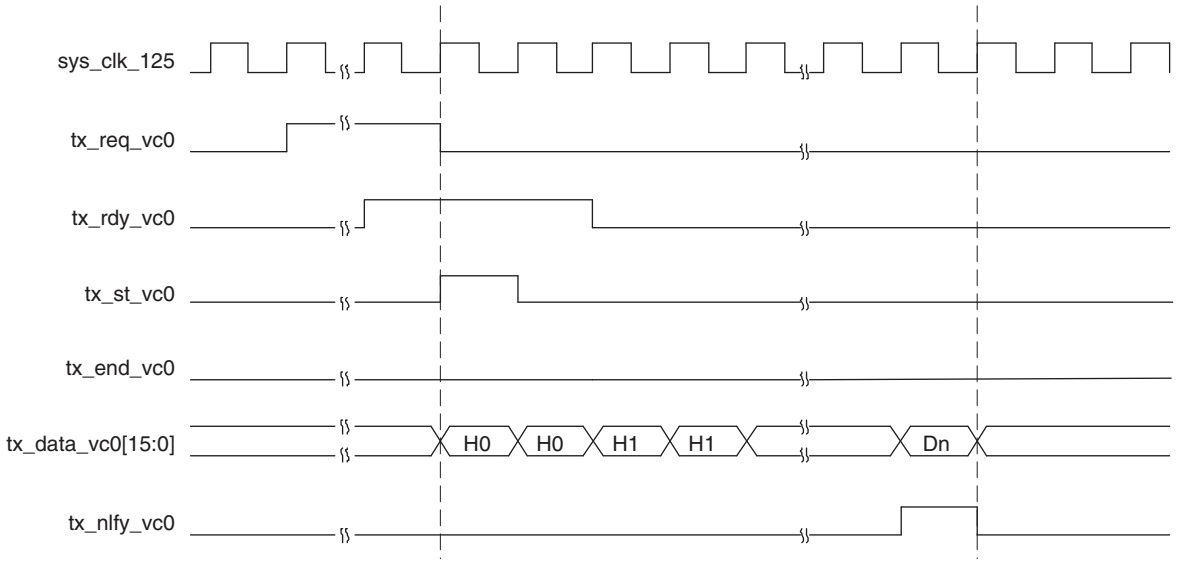
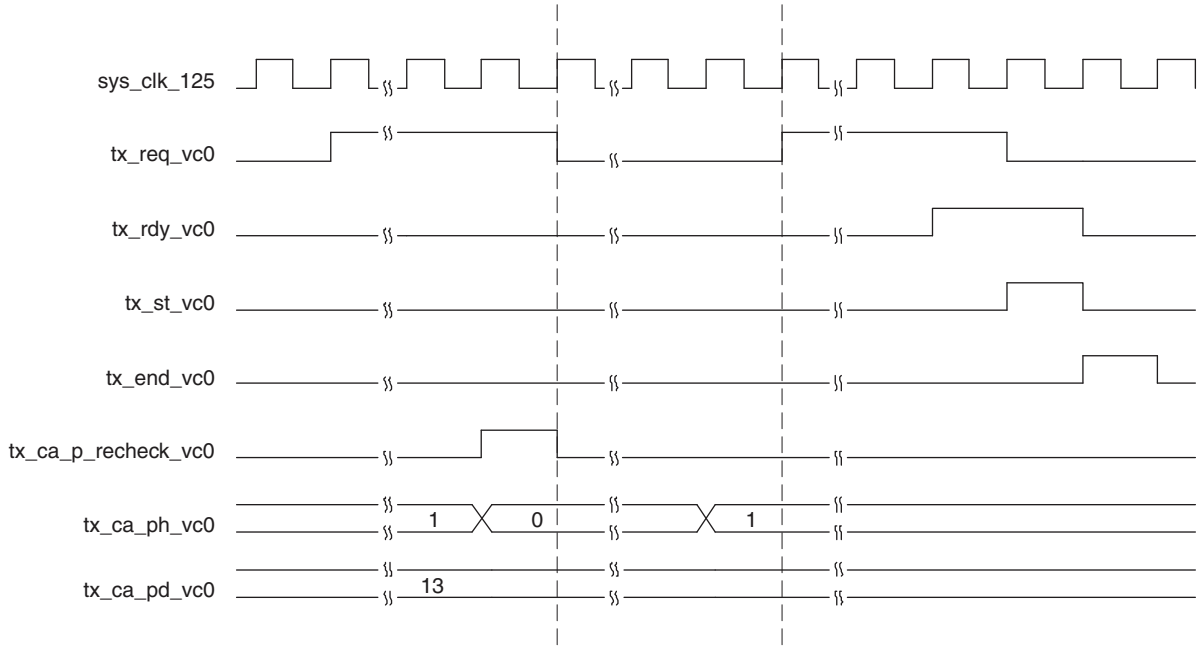


Figure 2-15. Transmit Interface Native x1 Posted Request with tx_ca_p-recheck Assertion



Receive TLP Interface

In the receive direction, TLPs will come from the core as they are received on the PCI Express lanes. Config read and config write TLPs to registers inside the core will be terminated inside the core. All other TLPs will be provided to the user. Also, if the core enables any of the BARs the TLP will go through a BAR check to make sure the TLPs address is in the range of any programmed BARs. If a BAR is accessed, the specific BAR will be indicated by the rx_bar_hit[6:0] bus.

When a TLP is sent to the user the rx_st_vc0 signal will be asserted with the first word of the TLP. The remaining TLP data will be provided on consecutive clock cycles until the last word with rx_end_vc0 asserted. If the TLP contains a ECRC error the rx_ecrc_err_vc0 signal will be asserted at the end of the TLP. If the TLP has a length prob-

lem the rx_malf_tlp_vc0 will be asserted at any time during the TLP. Figure 2-16 through Figure 2-19 provide timing diagrams of the receive interface.

TLPs come from the receive interface only as fast as they come from the PCI Express lanes. There will always be at least one clock cycle between rx_end_vc0 and the next rx_st_vc0.

Figure 2-16. Receive Interface, Clean TLP

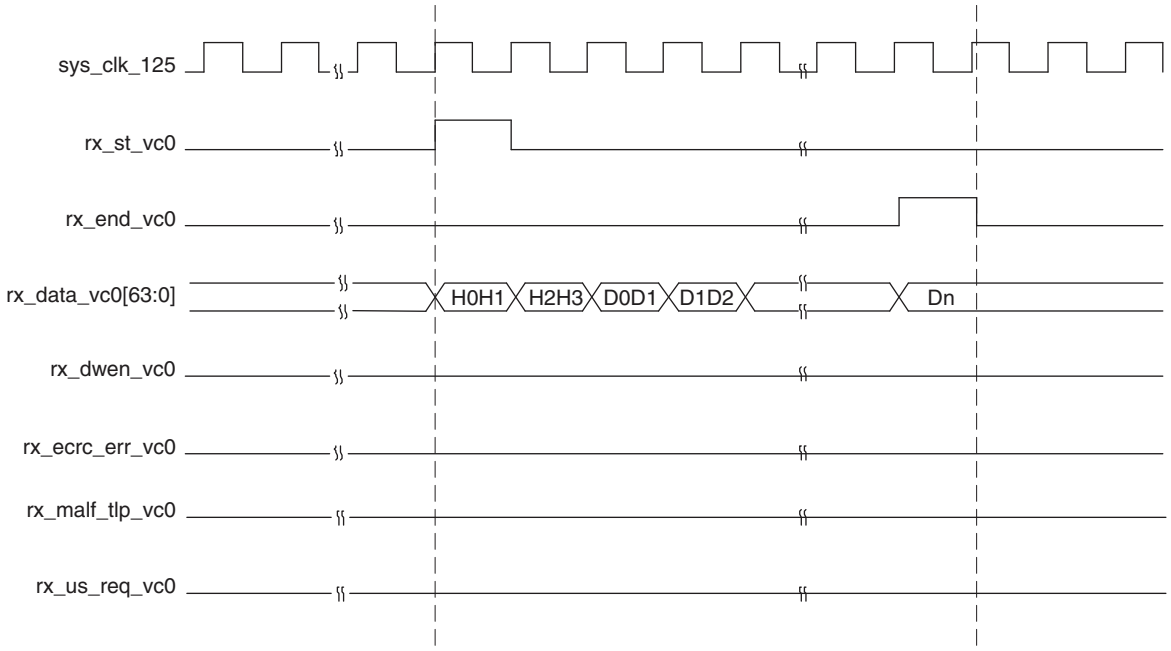


Figure 2-17. Receive Interface, ECRC Errored TLP

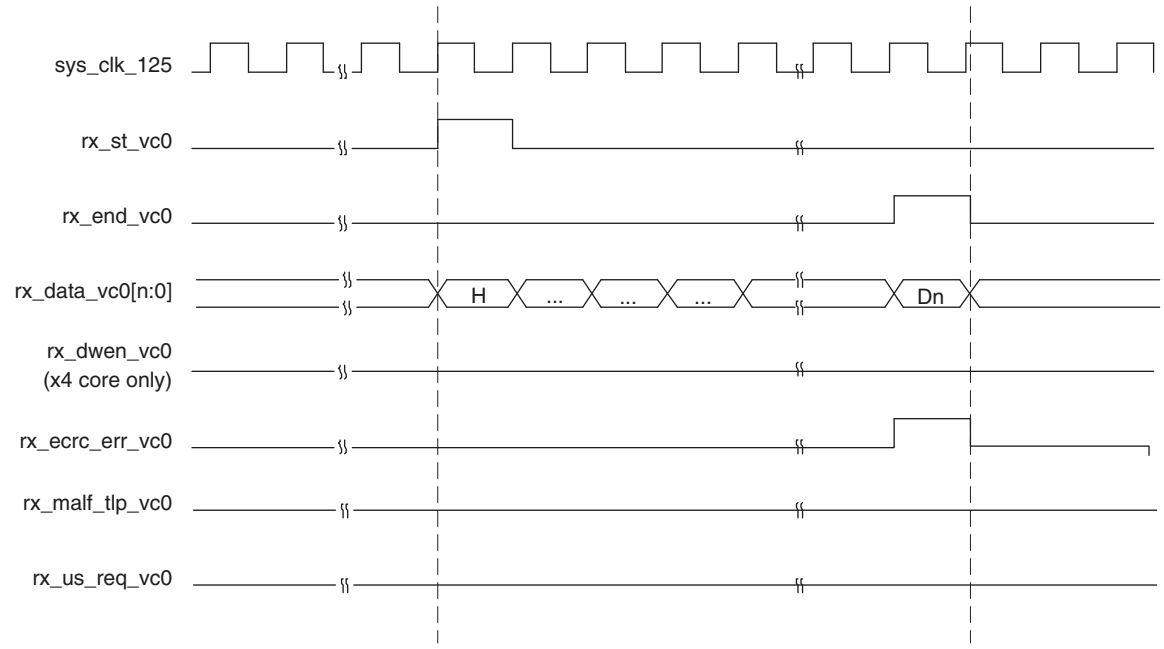


Figure 2-18. Receive Interface, Malformed TLP

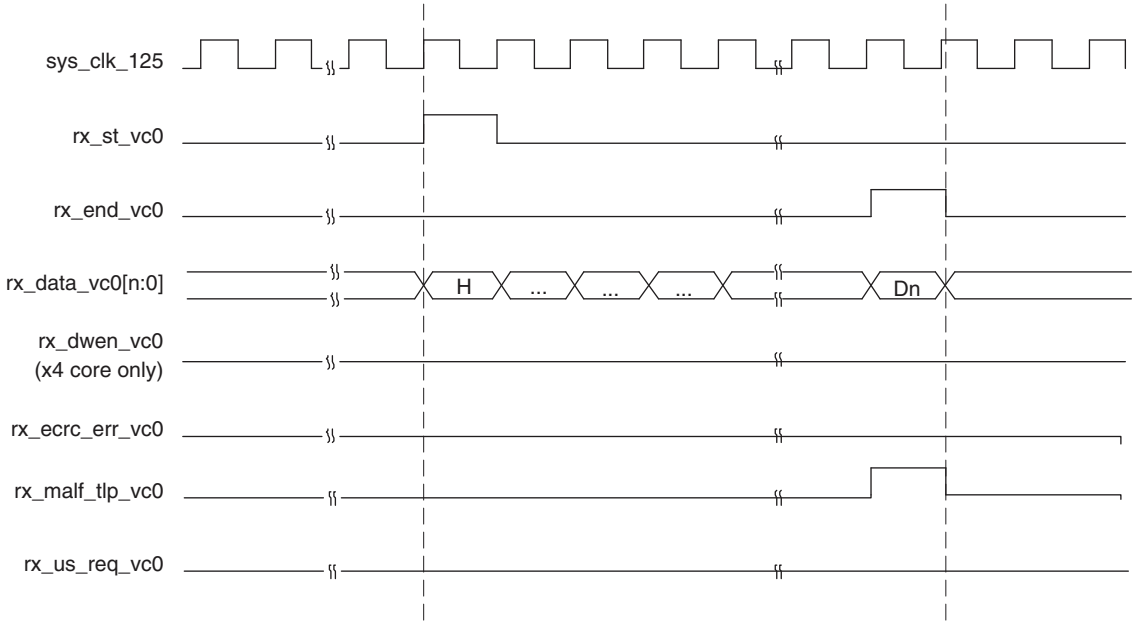
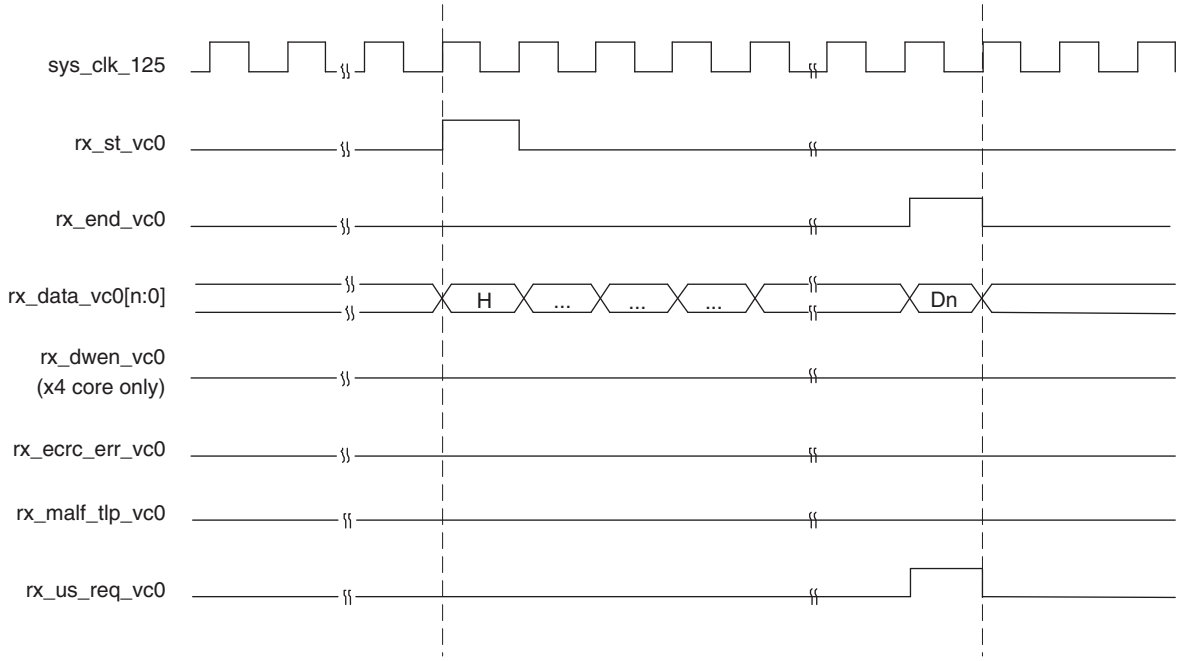


Figure 2-19. Receive Interface, Unsupported Request TLP



Using the Transmit and Receive Interfaces

There are two ways a PCI Express endpoint can interact with a root complex. As a completer, the endpoint will respond to accesses made by the root complex. As an initiator, the endpoint will perform accesses to the root complex. The following sections will discuss how to use the transmit and receive TLP interfaces for both of these types of interactions.

When the “Terminate All Config TLPs” option is checked in the IPexpress tool, the IP core will handle all configuration requests. This includes responding as well as credit handling.

As a Completer

In order to be accessed by a root complex at least one of the enabled BARs will need to be programmed. The BIOS or OS will enumerate the configuration space of the endpoint. The BARs initial value (loaded via the GUI) is read to understand the memory requirements of the endpoint. Each enabled BAR is then provided a base address to be used.

When a memory request is received the PCI Express core will perform a BAR check. The address contained in the memory request is checked against all of the enabled BARs. If the address is located in one of the BARs' address range the rx_bar_hit[6:0] port will indicate which BAR is currently being accessed.

At this time the rx_st_vc0 will be asserted with rx_data_vc0 providing the first eight bytes of the TLP. The memory request will terminate with the rx_end_vc0 port asserting. The user must now terminate the received TLP by release credit returns and completions for a non posted request. The user logic must decode release credits for all received TLPs except for errored TLP. If the core finds any errors in the current TLP, the error will be indicated on the rx_ecrc_err_vc0 or rx_malf_tlp_vc0 port. Refer to the Error Handling section for additional details on credit handling and completion rules for receive errored TLP.

If the TLP is a 32-bit MWr TLP (rx_data_vc0[63:56]= 0x40) or 64-bit MWr TLP (rx_data_vc0[63:56]=0x30) the address and data need to be extracted and written to the appropriate memory space. Once the TLP is processed the posted credits for the MWr TLP must be released to the far end. This is done using the ph_processed_vc0, pd_processed_vc0, and pd_num_vc0 ports. Each MWr TLP takes 1 header credit. There is one data credit used per four DWs of data. The length field (rx_data_vc0[41:32]) provides the number of DWs used in the TLP. If the TLP length is on 4DW boundary (rx_data_vc0[33:32]=0x0), the number of credits is the TLP length divided by 4 (rx_data_vc0[41:34]). If the TLP length is not on 4DW boundary (rx_data_vc0[33:32] >0) the number of credits is rx_data_vc0[41:34] + 1 (round up by 1). The number of credits used should then be placed on pd_num_vc0[7:0]. Assert ph_processed_vc0 and pd_processed_vc0 for 1 clock cycle to latch in the pd_num_vc0 port and release credits.

If the TLP is a 32-bit MRd TLP (rx_data_vc0[63:56]= 0x00) or 64-bit MRd TLP (rx_data_vc0[63:56]=0x20) the address needs to be read creating a completion TLP with the data. A CplD TLP (Completion with Data) will need to be created using the same Tag from the MRd. This Tag field allows the far end device to associate the completion with a read request. The completion must also not violate the read completion boundary of the far end requestor. The read completion boundary of the requestor can be found in the Link Control Register of the PCI Express capability structure. This information can be found from the IP core using the link_cntl_out[3]. If this bit is 0 then the read completion boundary is 64 bytes. If this bit is a 1 then the read completion boundary is 128 bytes. The read completion boundary tells the completer how to segment the CplDs required to terminate the read request. A completion must not cross a read completion boundary and must not exceed the maximum payload size. The Lower Address field of the CplD informs the far end the lower address of the current CplD allow the far end to piece the entire read data back together.

Once the CplD TLP is assembled the TLP needs to be sent and the credits for the MRd need to be released. To release the credits the port nph_processed_vc0 needs to be asserted for 1 clock cycle. This will release the 1 Non-Posted header credit used by a MRd.

The CplD TLP can be sent immediately without checking for completion credits. If a requestor requests data then it is necessary for the requestor to have enough credits to handle the request. If the user still wants to check for credits before sending then the credits for a completion should be checked against the tx_ca_cplh and tx_ca_cpld ports.

As a Requestor

As a requestor the endpoint will issue memory requests to the far end. In order to access memory on the far end device the physical memory address will need to be known. The physical memory address is the address used in the MWr and MRd TLP.

To send a MWr TLP the user must assemble the MWr TLP and then check to see if the credits are available to send the TLP. The credits consumed by a MWr TLP is the length field divided by 4. This value should be compared against the tx_ca_pd port value. If tx_ca_pd[12] is high, this indicates the far end has infinite credits available. The TLP can be sent regardless of the size. A MWr TLP takes 1 Posted header credit. This value can be compared against the tx_ca_ph port. Again, if tx_ca_ph[8] is high, this indicates the far end has infinite credits available.

To send a MRd TLP the user must assemble the MRd TLP and then check to see if the credits are available to send the TLP. The credits consumed by a MRd TLP is 1 Non-Posted header credit. This value should be compared against the tx_ca_nph port value. If tx_ca_nph[8] is high, this indicates the far end has infinite credits available. After a Non-Posted TLP is sent the np_req_pend port should be asserted until all Non-Posted requests are terminated.

In response to a MRd TLP the far end will send a CplD TLP. At this time the rx_st_vc0 will be asserted with rx_data_vc0 providing the first 8 bytes of the TLP. The completion will terminate with the rx_end_vc0 port asserting. The user must now terminate the received CplD. If the core found any errors in the current TLP the error will be indicated on the rx_ecrc_err_vc0 or rx_malf_tlp_vc0 port. An errored TLP does not need to be terminated or release credits. The core will not provide a NAK even if the rx_ecrc_err_vc0 or rx_malf_tlp_vc0 are asserted.

If the TLP is a CplD TLP (rx_data_vc0[63:56]= 0x4A) the data needs to be extracted stored until all CplDs associated with the current Tag are received.

For more information on credit handling, refer to the LatticeECP2M PCI Express Development Kit web page at:

http://www.latticesemi.com/Products/FPGAandCPLD/LatticeECP2M.aspx

and click on the Design Resources > Development Kit and Board for Lattice ECP2M link.

Demo designs can be found on the “Demo Applications” link. Supporting information for the evaluation board, including User Manuals, can be found on the “Solutions Board” link.

Unsupported Request Generation

The user ultimately is responsible for sending an Unsupported Request completion based on the capabilities of the user's design. For example, if the user's design only works with memory transactions and not I/O transactions, then I/O transactions are unsupported. These types of transactions require an Unsupported Request completion. There are several instances in which an Unsupported Request must be generated by the user. These conditions are listed below.

- rx_us_req port goes high with rx_st indicating a Memory Read Locked, Completion Locked, or Vendor Defined Message.
- Type of TLP is not supported by the user's design (I/O or memory request)

Table 2-2 shows the types of unsupported TLPs which can be received by the IP core and the user interaction.

Table 2-2. Unsupported TLPs Which Can be Received by the IP

		Unsupported Event Request	"rx_us_req" Port Goes High	User Logic Needs to Send UR Completion	User Needs to Release Credit
"Terminate All Config TLP" is enabled	1	Configuration Read/Write Type 1	No	No	No
	2	Memory Read Request - Locked	Yes	Yes	Yes
	3	Locked Completions	Yes	No (Because EP Ignores Locked Completions)	No
	4	Vendor Defined Message Type 0 and Type1	No	Yes	Yes
	Unsupported by User Design	TLP with invalid BAR Address	No	Yes (With UR Status)	Yes
		MRd with Inconsistent TLP Type	No	Yes (With UR Status)	Yes
		MWr with Inconsistent TLP Type	No	No (Since MWr is the Posted Req)	Yes
		I/ORd with Inconsistent TLP Type	No	Yes (With UR Status)	Yes
		I/OWr with Inconsistent TLP Type	No	Yes (With UR Status)	Yes
		Msg with Inconsistent TLP Type	No	No (Since Msg is the Posted Req)	Yes
		MsgD with Inconsistent TLP Type	No	No (Since Msg is the Posted Req)	Yes

1. For unsupported by user design events, "inconsistent TLP type" means, for example, MRd request came in for the BAR that only supports I/O, and the other way around.

Configuration Space

The PCI Express IP core includes the required PCI configuration registers and several optional capabilities. The section will define which registers are included inside the core and how they are utilized.

Base Configuration Type0 Registers

This base configuration Type0 registers are the legacy PCI configuration registers from 0x0-0x3F. The user sets appropriate bits in these registers using the IPexpress GUI. The user is provided with the cmd_reg_out[3:0] to monitor certain bits in the base configuration space.

Power Management Capability Structure

The Power Management Capability Structure is required for PCI Express. The base of this capability structure is located at 0x50. The user sets appropriate bits in these registers using the IPexpress GUI. The user is provided with the pme_status, pme_enable, and pm_power_state ports to monitor and control the Power Management of the endpoint.

MSI Capability Structure

The Message Signaled Interrupt Capability Structure is optional and is included in the IP core. The base of this capability structure is located at 0x70. The number of MSIs is selected in the IPexpress GUI. The user is provided with the msi, mm_enable, and msi_enable ports to utilize MSI.

How to Enable/Disable MSI

The user can enable or disable MSI by setting or resetting bit 16 of PCI Express configuration registers (address 70h). This bit value is also shown on "msi_enable" on IP core ports.

This status of MSI is also reflected at bit 16 of the first Dword (Dword 0) of MSI Capability Register Set (which is bit 0 of Message Control Register), and this value is also shown on "msi_enable" port.

How to issue MSI

Up to eight MSI interrupts can be issued. The user can use any bit. Assertion to any of bit 0 to 7 of MSI issues an interrupt of the corresponding MSI number. The IP issues the interrupt at the rising edge of MSI input signal.

PCI Express Capability Structure

The PCI Express Capability Structure is required for PCI Express. The base of this capability structure is located at 0x90. The user sets appropriate bits in these registers using the IPexpress GUI. The user is provided with the dev_cntl_out and lnk_cntl_out ports to monitor certain registers in the design.

Device Serial Number Capability Structure

The Device Serial Number Capability Structure is optional and is included in the IP core. The base of this capability is located at 0x100 which is in the extended register space. The user sets the 64-bit Device Serial Number in the IPexpress GUI.

Advanced Error Reporting Capability Structure

The Advanced Error Reporting Capability Structure is optional and is included in the IP core. The base of this capability is located at 0x1A0 which is in the extended register space. The user is provided the cmpln_tout, cmpltr_abort_np/cmpltr_abort_p, unexp_cmpln, and err_tlp_header ports to provide error conditions to the AER.

Handling of Configuration Requests

Table 2-3 provides the Configuration Space memory map.

Table 2-3. PCI Express Core Configuration Space Memory Map

Address	Description	Config Register
0x0 - 0x3C	Type 0	0 - F
0x40 - 0x4F	Empty	
0x50 - 0x57	Power Management CS	14 - 15
0x58 - 0x6F	Empty	
0x70 - 0x7F	MSI CS	1C - 1D
0x80 - 0x8F	Empty	
0x90 - 0xC3	PCI Express CS	24 - 30
0xA4 - 0xFF	Empty	
0x100 - 0x10B	Device Serial Number CS	Extended 0 - 2
0x10C - 0x17B	Reserved	
0x17C - 0x19F	Empty	
0x1A0 - 0x1C8	AER CS	Extended 128 - 132

The PCI Express core might optionally terminate all configuration requests registers identified in the table. By default, configuration requests to registers that are marked as empty will not be terminated by the core and passed to the user through the receive TLP interface. If the user wishes to implement further capability structures not implemented by the core, or implement PCI-SIG ECNs this could be implemented in the user design. If the user does not want to handle any configuration requests there is an option in the IPexpress GUI to have the core terminate all configuration requests. When this is selected, the user will never see a configuration request on the receive TLP interface.

Wishbone Interface

The optional wishbone interface provides the user with access into the configuration space and select status and control registers. This interface is useful for designs which require knowledge of the entire configuration space, not only those provided as port to the user. When a Wishbone access to the PCI express configuration register space occurs along with configuration access from PCI express link, the later takes the precedence and Wishbone cycle termination will be delayed.

Wishbone Byte/Bit Ordering

The write byte order for Wishbone is:

DAT_I = {upper byte of N+2, lower byte of N+2, upper byte of N, lower byte of N}

The read byte order for Wishbone is different depending on the address range.

1. For an address range of 0x0000-0x0FFF accessing the PCI Express configuration space, the read byte ordering is:
   DAT_0 = {lower byte of N, upper byte of N, lower byte of N+2, upper byte of N+2}

2. For an address range of 0x1000-101F accessing control and status registers inside the PCI Express IP core, the read byte ordering is:
   DAT_O = {upper byte of N+2, lower byte of N+2, upper byte of N, ,lower byte of N}

The bit ordering within a byte is always 7:0.

The memory map for the Wishbone interface is provided in Table 2-4.

Table 2-4. Wishbone Interface Memory Map

Type	Address (hex)	Bits		Default	Description
PCI Express Type00 CFG Space					
PCI Express Configuration Space	0-FFF	31:0	R/W	GUI	PCI Express Configuration Space. Includes Type0/1, New Capabilities, and extended.
IP Control and Status Registers					
Power Management	1000-1003	31	COW	0	Received Vendor type DLLP
		30			Received Power Message DLLP
		29:27			Reserved
		28:3			RX Vendor DLLP Data
		2:0			RX Power Message DLLP Type
	1004-1007	31	R/W	0	Transmit Vendor type DLLP, cleared when DLLP is sent
		30			Transmit Power Message DLLP, cleared when DLLP is sent
		29:27			Reserved
		26:3			TX Vendor DLLP Data
		2:0			TX Power Message DLLP Type

Table 2-4. Wishbone Interface Memory Map (Continued)

Type	Address (hex)	Bits		Default	Description
Status	1008-100B	31:24	R	0	Reserved
		23:20			PHY LSM Status. For X1 Core [23:21] are Reserved.
		19:16	COW		PHY Connection Status / Result of Receiver Detection. For X1 Core [19:17] are Reserved.
		15:12			PHY Receive/Rx Electrical Idle. For x1 Core [15:13] are Reserved.
		11:7	R		LTSSM State *
		6:3			DLL/Link Control SM Status [6] - DL Inactive State [5] - DL Init State [4] - DL Active State [3] - DL Up State
		2:0			Reserved
	100C-100F	31:22	R/W	0	Reserved
		21:18			LTSSM goto Loopback For x1 Core [21:19] are Reserved
		17			TLP Debug Mode: TLP bypasses DLL & TRNC check.
		16			PHY/LTSSM Send Beacon
		15			Force LSM Status active
		14			Force Received Electrical Idle
		13			Force PHY Connection Status

Table 2-4. Wishbone Interface Memory Map (Continued)

Type	Address (hex)	Bits		Default	Description
Status (Cont.)	100C-100F (Cont.)	12	R/W	0	Force Disable Scrambler (to PCS)
		11			Disable scrambling bit in TS1/TS2
		10			LTSSM go to Disable
		9			LTSSM go to Detect
		8			LTSSM go to HotReset
		7			LTSSM go to L0s
		6			LTSSM go to L1
		5			LTSSM go to L2
		4			LTSSM go to L0s and Tx FTS
		3			Reserved
		2			LTSSM go to Recovery
		1			LTSSM go to Config
		0			LTSSM no training
	1010-1013	31:30	R/W	GUI	Reserved
		29:16			ACK/NAK Latency Timer
		15			Reserved
		14:10			Number of FTS
		9:0			SKP Insert Counter
	1014-1017	31:18	R/W	Set in IP	Reserved
		17:11			Update Frequency for Posted Header
		10:0			Update Frequency for Posted Data
	1018-101B	31:18	R/W	Set in IP	Reserved
		17:11			Update Frequency for Non-Posted Header
		10:0			Update Frequency for Non-Posted Data
	101C-101F	31:18	R/W	Set in IP	Reserved
		17:11			Update Frequency for Completion Header
		10:0			Update Frequency for Completion Data
	1020-1023	31:12	R/W	Set in IP	Reserved
		11:0			FC Update Timer
	1023-1027	31:8	R/W	0	Reserved
		7:0			Link Number

*LTSSM State Encoding:
0 - DETECT
1 - POLLING
2 - CONFIG
3 - L0
4 - L0s
5 - L1
6 - L2
7 - RECOVERY
8 - LOOPBACK
9 - HOTRST
10 - DISABLED
R - Read Only
R/W - Read and Write
COW - Clear On Wrfatalite

Error Handling

The IP Core handles DLL errors. User logic does not need to control any of DLL errors. When the IP receives NAK, the IP does not report outside of IP and retransmit TLP in retry buffer.

Table 2-5 lists physical layer error messages, error type, and how the IP responds to the error. Table 2-5 corresponds to Table 6-2 of the *PCI Express Base Specification Version 1.1*. The IP automatically responds to all Physical Layer errors. No user logic interaction is required.

Table 2-5. Physical Layer Error List

Error Name	Error Type	IP Action	IP Handling from User Logic	References
Receiver Error	Correctable	Send ERR_COR to root complex.	Nothing is required.	Section 4.2.1.3 Section 4.2.2.1 Section 4.2.4.4 Section 4.2.6

Table 2-6 lists data link layer error messages, error type, and how the IP responds to the errors. Table 2-6 corresponds to Table 6-3 of the *PCI Express Base Specification Version 1.1*. The IP automatically responds to all Data Link Layer errors. No user logic interaction is required.

Table 2-6. Data Link Layer Error List

Error Name	Error Type	IP Action	IP Handling from User Logic	References
Bad TLP	Correctable	Send ERR_COR to root complex.	Nothing is required.	Section 3.5.2.1
Bad DLLP	Correctable	Send ERR_COR to root complex.	Nothing is required.	Section 3.5.2.1
Replay Timeout	Correctable	Send ERR_COR to root complex.	Nothing is required.	Section 3.5.2.1
REPLAY NUM Rollover	Correctable	Send ERR_COR to root complex.	Nothing is required.	Section 3.5.2.1
Data Link Layer Protocol Error	Uncorrectable (fatal)	- If the severity level is fatal, send ERR_FATAL to root complex. - If the severity level is non-fatal, send ERR_NONFATAL to root complex.	Nothing is required.	Section 3.5.2.1
Surprise Down	Uncorrectable (fatal)	Not required to implement for endpoint, per section 7.8.6, page 380	Nothing is required.	Section 3.5.2.1

Table 2-7 lists transaction layer error messages, error type, how the IP responds to the errors, and any IP handling that requires user logic interaction. The table corresponds to Table 6-4 of the *PCI Express Base Specification Version 1.1*.

Table 2-7. Transaction Layer Error List

Error Name	Error Type	IP Action	IP Handling from User Logic	References
Poisoned TLP Received	Uncorrectable (non-fatal)	**Automatically:** - If the severity level is non-fatal, send ERR_COR for the advisory non-fatal error to root croot complex. - If the severity level is fatal, send ERR_FATAL to root complex. - Log the header of the poisoned TLP. - Release credit.	This is a special case where the TLP is passed to the user, but the core handles the error message automatically. The data is passed to the user logic for handling, per section 2.7.2.2.	Section 2.7.2.2
ECRC Check Failed	Uncorrectable (non-fatal)	**Automatically (if ECRC checking is supported):** - If the severity level is non-fatal, send the non-fatal error to root complex. - If the severity level is fatal, send ERR_FATAL to root complex. - Log the header of the TLP that encountered the ECRC error. - Assert rx_ecrc_err_vc0 to userlogic side.	- Packet is passed to the user for discard. - No completion for non-posted is returned because anything can be wrong in the packet. - Release credit.	Section 2.7.1
Unsupported Request (UR) Configuration Type 1	Uncorrectable (non-fatal)	**Automatically:** - If the severity level is non-fatal, send ERR_COR for the advisory non-fatal error to root complex. - If the severity level is fatal, send ERR_FATAL to root complex. - Release credit. - Send UR completion. - Log the header of the TLP.	Nothing is required.	Section 2.2.8.6 Section 2.3.1 Section 2.3.2 Section 2.7.2.2 Section 2.9.1 Section 5.3.1 Section 6.2.3 Section 6.2.6 Section 6.2.8.1 Section 6.5.7 Section 7.3.1 Section 7.3.3 Section 7.5.1.1 Section 7.5.1.2
Unsupported Request (UR) MrdLK	Uncorrectable (non-fatal)	**Automatically:** - If the severity level is non-fatal, send ERR_COR for the advisory non-fatal error to root complex. - If the severity level is fatal, send ERR_FATAL to root complex. - Assert rx_ur_req_vc0 to userlogic side. - Log the header of the TLP.	- Release credit. - Send UR completion.	Section 2.2.8.6 Section 2.3.1 Section 2.3.2 Section 2.7.2.2 Section 2.9.1 Section 5.3.1 Section 6.2.3 Section 6.2.6 Section 6.2.8.1 Section 6.5.7 Section 7.3.1 Section 7.3.3 Section 7.5.1.1 Section 7.5.1.2

Table 2-7. Transaction Layer Error List (Continued)

Error Name	Error Type	IP Action	IP Handling from User Logic	References
Unsupported Request (UR) Vector Defined Message Type 0	Uncorrectable (non-fatal)	**Based on assertion of ur_p_ext input:** - If the severity level is non-fatal, send ERR_NONFATAL because this is an unsupported posted request which is not advisory non-fatal. - If the severity level is fatal, send ERR_FATAL to root complex. **Based on assertion of err_tlp_header[127:0] input:** - Log the header of the TLP.	- Assert ur_p_ext input. - Assert err_tlp_header[127:0] inputs. - Release credit. - Send completion with UR status for vendor define Type0 if not supported.	Section 2.2.8.6 Section 2.3.1 Section 2.3.2 Section 2.7.2.2 Section 2.9.1 Section 5.3.1 Section 6.2.3 Section 6.2.6 Section 6.2.8.1 Section 6.5.7 Section 7.3.1 Section 7.3.3 Section 7.5.1.1 Section 7.5.1.2
Unsupported Request (UR) Locked Completion	Uncorrectable (non-fatal)	**Based on assertion of ur_np_ext/ur_p_ext input:** - If the severity level is non-fatal and the request type is non-posted, send ERR_COR for the advisory non-fatal error to root complex. - If the severity level is fatal, send ERR_FATAL to root complex. **Based on assertion of err_tlp_header[127:0] input:** - Log the header of the TLP.	Assert cmpln_tout input.	Section 2.2.8.6 Section 2.3.1 Section 2.3.2 Section 2.7.2.2 Section 2.9.1 Section 5.3.1 Section 6.2.3 Section 6.2.6 Section 6.2.8.1 Section 6.5.7 Section 7.3.1 Section 7.3.3 Section 7.5.1.1 Section 7.5.1.2
Unsupported Request (UR) Request type not supported	Uncorrectable (non-fatal)	**Based on assertion of ur_np_ext/ur_p_ext input:** - If the severity level is non-fatal and request type is non-posted, send ERR_COR for the advisory non-fatal error to root complex. Send ERR_NONFATAL for posted request which is not advisory non-fatal. - If the severity level is fatal, send ERR_FATAL to root complex. **Based on assertion of err_tlp_header[127:0] input:** - Log the header of the TLP.	- Assert ur_np_ext/ur_p_ext input. - Assert err_tlp_header[127:0] inputs. - Release credit. - Send UR completion if request is non-posted.	Section 2.2.8.6 Section 2.3.1 Section 2.3.2 Section 2.7.2.2 Section 2.9.1 Section 5.3.1 Section 6.2.3 Section 6.2.6 Section 6.2.8.1 Section 6.5.7 Section 7.3.1 Section 7.3.3 Section 7.5.1.1 Section 7.5.1.2

Table 2-7. Transaction Layer Error List (Continued)

Error Name	Error Type	IP Action	IP Handling from User Logic	References
Unsupported Request (UR) Request not referencing address space mapped within the device	Uncorrectable (non-fatal)	**Based on assertion of ur_np_ext/ur_p_ext input:** - If the severity level is non-fatal and request type is non-posted, send ERR_COR for the advisory non-fatal error to root complex. Send ERR_NONFATAL for posted request which is not advisory non-fatal. - If the severity level is fatal, send ERR_FATAL to root complex. **Based on assertion of err_tlp_header[127:0] input:** - Log the header of the TLP.	- Assert ur_np_ext/ur_p_ext input. - Assert err_tlp_header[127:0] input. - Release credit. - Send UR completion if request is non-posted.	Section 2.2.8.6 Section 2.3.1 Section 2.3.2 Section 2.7.2.2 Section 2.9.1 Section 5.3.1 Section 6.2.3 Section 6.2.6 Section 6.2.8.1 Section 6.5.7 Section 7.3.1 Section 7.3.3 Section 7.5.1.1 Section 7.5.1.2
Unsupported Request (UR) Message code with unsupported or undefined message code	Uncorrectable (non-fatal)	**Based on assertion of ur_p_ext input:** - If the severity level is non-fatal, send ERR_NONFATAL (Unsupported Posted request is not advisory non-fatal). - If the severity level is fatal, send ERR_FATAL to root complex. **Based on assertion of err_tlp_header[127:0] input:** - Log the header of the TLP.	- Assert ur_p_ext input. - Assert err_tlp_header[127:0] inputs. - Release credit.	Section 2.2.8.6 Section 2.3.1 Section 2.3.2 Section 2.7.2.2 Section 2.9.1 Section 5.3.1 Section 6.2.3 Section 6.2.6 Section 6.2.8.1 Section 6.5.7 Section 7.3.1 Section 7.3.3 Section 7.5.1.1 Section 7.5.1.2
Completion Timeout	Uncorrectable (non-fatal)	**Based on assertion of cmpln_tout input:** - If the severity level is non-fatal, send ERR_COR for the advisory non-fatal error to root complex. - If the severity level is fatal, send ERR_FATAL to root complex.	Assert cmpln_tout input.	Section 2.8
Completer Abort	Uncorrectable (non-fatal)	**Based on assertion of cmpltr_abort_p/cmpltr_abort_np input:** - If the severity level is non-fatal, send ERR_COR for the advisory non-fatal error to root complex. - If the severity level is fatal, send ERR_FATAL to root complex. **Based on assertion of err_tlp_header[127:0] input:** - Log the header of the TLP.	- Assert cmpltr_abort_p/cmpltr_abort_np input. input. - Assert err_tlp_header[127:0] input.	Section 2.3.1

Table 2-7. Transaction Layer Error List (Continued)

Error Name	Error Type	IP Action	IP Handling from User Logic	References
Unexpected Completion	Uncorrectable (non-fatal)	**Based on assertion of unexp_cmpln:** - If the severity level is non-fatal, send ERR_COR for the advisory non-fatal error to root complex. - If the severity level is fatal, send ERR_FATAL to root complex. **Based on assertion of err_tlp_header[127:0] input:** - Log the header of the TLP.	- Assert unexp_cmpln input. - Assert err_tlp_header[127:0] input.	Section 2.3.2
Receiver Overflow	Uncorrectable (fatal)	**Automatically:** - If the severity level is fatal, send ERR_FATAL to root complex. - If the severity level is non-fatal,send ERR_NONFATAL to root complex.	Nothing is required.	Section 2.6.1.2
Flow Control Protocol Error	Uncorrectable (fatal)	**Automatically:** - If the severity level is fatal, send ERR_FATAL to root complex. - If the severity level is non-fatal, send ERR_NONFATAL to root complex.	Nothing is required.	Section 2.6.1
Malformed TLP	Uncorrectable (fatal)	**Automatically:** - If the severity level is fatal, send ERR_FATAL to root complex. - If the severity level is non-fatal, send ERR_NONFATAL to root complex. - Assert rx_malf_tlp_vc0. - Log the header of the TLP.	Nothing is required.	Section 2.2.8.6 Section 2.3.1 Section 2.3.2 Section 2.7.2.2 Section 2.9.1 Section 5.3.1 Section 6.2.3 Section 6.2.6 Section 6.2.8.1 Section 6.5.7 Section 7.3.1 Section 7.3.3 Section 7.5.1.1 Section 7.5.1.2

Parameter Settings

The IPexpress tool is used to create IP and architectural modules in the Diamond or ispLEVER software. Refer to the IP Core Generation and Evaluation section for a description on how to generate the IP.

Table 3-1 provides the list of user configurable parameters for the PCI Express IP core. The parameter settings are specified using the PCI Express IP core Configuration GUI in IPexpress. The numerous PCI Express parameter options are partitioned across multiple GUI tabs as shown in this chapter.

Table 3-1. IP Core Parameters

Parameters	Range/Options	Default
General		
LTSSM		
PCI Express Link Configuration	x4 Native, x1 Downgraded, x1 Native	x4 Native
Use Hard LTSSM MACO Block (LatticeSCM only)	Yes/No	Yes
Control Interface		
Include System Bus interface for PCS access (LatticeSCM only)	Yes/No	Yes
Include Wishbone interface	Yes/No	No
Endpoint Type	PCI Express Endpoint, Legacy Endpoint	PCI Express Endpoint
PCS Pipe Options (ECP2M/ECP3 only)		
Config		
Configuration	x4	x4
Data Width	8	8
Channel	Ch0	Ch0
Quad Location		
Quad Location (ECP2M)	Left Top, Right Top, Left Bottom, Right Bottom	Right Top
Quad Location (ECP3)	PCSA, PCSB, PCSD, PCSD	PCSA
Physical Layer		
Include Master Loop back datapath	Yes/No	No
Data Link Layer		
Update Flow Control Generation Control		
Number of PH credits between UpdateFC P	1-127	8
Number of PD credits between UpdateFC P	1-2047	255
Number of NPH credits between UpdateFC NP	1-127	8
Number of NPD credits between UpdateFC NP	1-2047	255
Worst case number of 125MHz clock cycles between UpdateFC	3650-4095	4095
Transaction Layer		
Include ECRC support	Yes/No	No
Initial Receive Credits		
Infinite PH Credits	Yes/No	No
Initial PH credits available	1-127	0
Infinite PD Credits	Yes/No	No
Initial PD credits available	8-255	0
Initial NPH credits available	1-32	0

Table 3-1. IP Core Parameters (Continued)

Parameters	Range/Options	Default
Initial NPD credits available	8-2047	0
Configuration Space		
Type0 Config Space		
Device ID	0000-ffff	0000
Vendor ID	0000-ffff	0000
Class Code	000000-ffffff	000000
Rev ID	00-ff	00
BIST	00-ff	00
Header Type	00-ff	00
Bar0	00000000-ffffffff	00000000
Bar0 Enable	Yes/No	No
Bar1	00000000-ffffffff	00000000
Bar1Enable	Yes/No	No
Bar2	00000000-ffffffff	00000000
Bar2 Enable	Yes/No	No
Bar3	00000000-ffffffff	00000000
Bar3 Enable	Yes/No	No
Bar4	00000000-ffffffff	00000000
Bar4 Enable	Yes/No	No
Bar5	00000000-ffffffff	00000000
Bar5 Enable	Yes/No	No
CardBus CIS Pointer	00000000-ffffffff	00000000
Subsystem ID	0000-ffff	0000
Subsystem Vendor ID	0000-ffff	0000
ExpROM Base Addr	00000000-ffffffff	00000000
Expansion ROM Enable	Yes/No	No
Load IDs from Ports	Yes/No	No
Power Management Capability Structure		
Power Management Cap Reg [31-16]	0000-ffff	0003
Date Scale Multiplier	0-3	0
Power Consumed in D0 (Watts)	00-ff	00
Power Consumed in D0 (Watts)	00-ff	00
Power Consumed in D1 (Watts)	00-ff	00
Power Consumed in D2 (Watts)	00-ff	00
Power Consumed in D3 (Watts)	00-ff	00
Power Dissipated in D0 (Watts)	00-ff	00
Power Dissipated in D1 (Watts)	00-ff	00
Power Dissipated in D2 (Watts)	00-ff	00
Power Dissipated in D3 (Watts)	00-ff	00
Power Dissipated in D4 (Watts)	00-ff	00
MSI Capability Structure		
Use Message Signaled Interrupts	Yes/No	Yes
Number of Messages Requested	1-8	1

Table 3-1. IP Core Parameters (Continued)

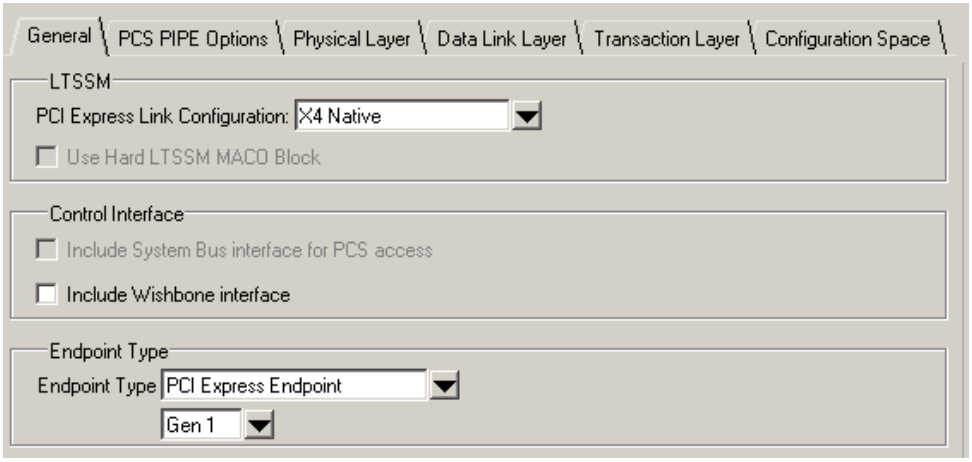
Parameters	Range/Options	Default
PCI Capability Structure		
Next Capability Pointer	00-ff	00
PCIe Capability Version	1 or 2	1
Max Payload Size Bytes	128, 256, 512, 1024, 2048, 4096	128
Device Capabilities Register [28:3]	0000000-fffffff	0000000
Enable Relaxed Ordering	Yes/No	Yes
Maximum Link Width	1, 4	4
Link Capabilities Register [17:10]	00-ff	00
Device Capabilities 2 Register [4:0]	00-1f	11
Device Serial Number		
Device Serial Number Version	1	1
Device Serial Number	0000000000000000-ffffffffffffffff	0000000000000000
Advanced Error Reporting		
Use Advanced Error Reporting	Yes/No	No
Advanced Error Reporting	1	Disabled
Terminate All Config TLPs		
Terminate All Config TLPs	Yes/No	Yes
User Extended Capability Structure	000-fff	Disabled

The default values shown in the following pages are those used for the PCI Express reference design. IP core options for each tab are discussed in further detail.

General Tab

Figure 3-1 shows the contents of the General tab.

Figure 3-1. PCI Express IP Core General Options in the IPexpress Tool



The General tab consists of the following parameters:

PCI Express Link Configuration

Specifies the link width and type of core to be used.

- x4 Native - This is a x4 link width using a 64-bit datapath. This configuration can dynamically downgrade to a x1 link width.

- x1 Downgraded - This is a x1 link width using a 64-bit datapath.
- x1 Native - This is a x1 link width only using a 16-bit datapath.

Use Hard LTSSM MACO Block (LatticeSCM Only)

This option selects the MACO based LTSSM block. This block is available in the LatticeSCM 15, 40, 80, and 115 device sizes.

Include System Bus Interface for PCS Access (LatticeSCM Only)

This option includes the system bus interface to the PCS/SERDES block. The system bus interface allows the user to interact with the memory map of the PCS/SERDES block.

The user has the option of accessing the embedded memory maps of the PCS/SERDES. For more information on the System Bus please refer to Lattice technical note TN1085, LatticeSC MPI/System Bus.

The PCS/SERDES contains an embedded memory map that can be accessed at run time via a dedicated system bus connection. This memory map provides access to properties such as SERDES buffer settings, resets, power-down, and status. The PCI Express IP core allows the user access to the system bus interface of the PCS/SERDES through top-level ports.

The system bus interface should be used if the user requires run time access to the controls provided by the memory map.

The memory map will need to be accessed if the link width is greater than a x1 since the MCA will need to be programmed to the configured link width (see Appendix A). The System Bus interface can also be used to power down unused channels, change buffers settings, and gather status information.

A complete description of the memory map of the PCS/SERDES can be found in the Memory Map section of DS1005, LatticeSC/M Family flexiPCS Data Sheet.

The Resource Utilization section provides a reference design that uses an FPGA design to interact with the IP core and the system bus to support greater than x1 dynamic lane width.

Include Wishbone Interface

This option includes a Wishbone interface to access certain features of the PCI Express core (see Table 2-1).

Endpoint Type

This option allow the user to chose between PCI Express Endpoint or Legacy Endpoint.

- Legacy Endpoint is permitted to support locked access.
- PCI Express Endpoint does not support locked access. PCI Express Endpoint must treat a MRdLk request as an unsupported request. Refer to the *PCI Express Base Specification Version 1.1, sections* 6.5.6 and 6.5.7, for more details.

PCS Pipe Options Tab

Figure 3-2 and Figure 3-3 show the contents of the PCS Pipe Options tab.

Figure 3-2. PCI Express PCS Pipe Options in the IPexpress Tool (ECP2M, ECP2MS)

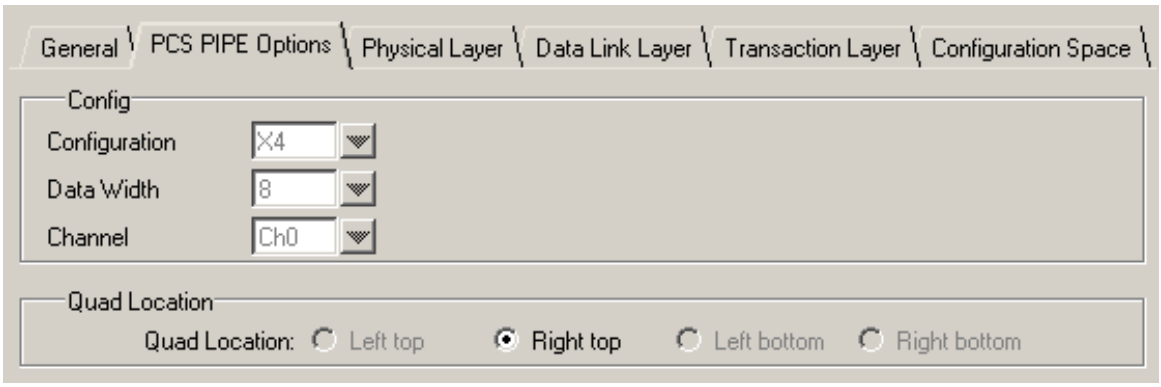
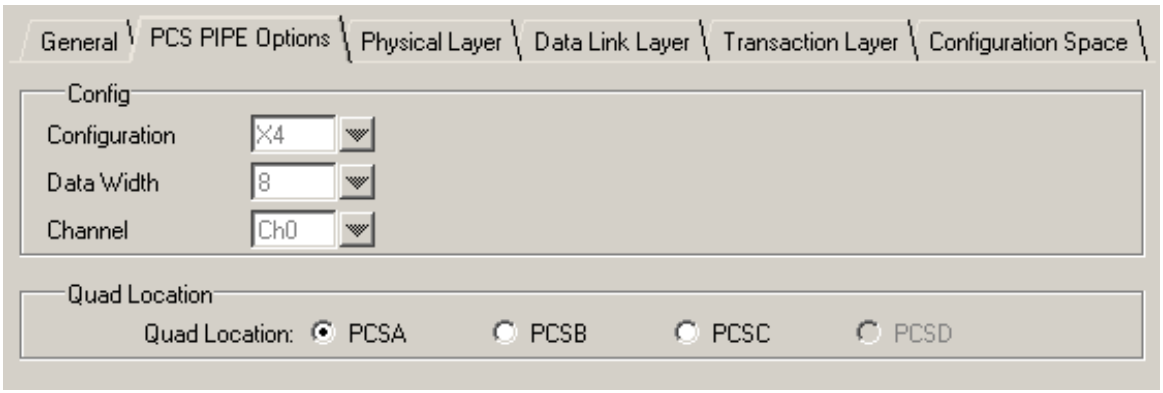


Figure 3-3. PCI Express PCS Pipe Options in the IPexpress Tool (ECP3)



Config

If the PCI Express Link Configuration drop down list in the General tab selection is X4 Native or X4 Downgraded, these options can not be modified. If 1 native is selected, Channel (Ch0/Ch1/Ch2/Ch3) of a SERDES quad can be selected.

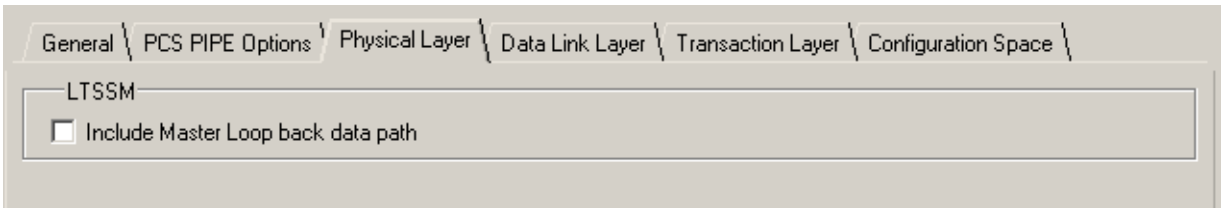
Quad Location

This option is used to select the location of the SERDES quad. See the Locating the LatticeECP3 Hard Elements section.

Physical Layer Tab

Figure 3-4 shows the contents of the Physical Layer tab.

Figure 3-4. PCI Express IP Core Physical Layer Options in the IPexpress Tool



Include Master Loopback Data Path

This option includes additional transmit and receive data path ports to the IP, if the device needs to be used as a loopback master in Loopback state of the LTSSM. In Table 2-1, refer to following I/O ports:

tx_lbk_rdy

tx_lbk_kcntl

tx_lbk_data

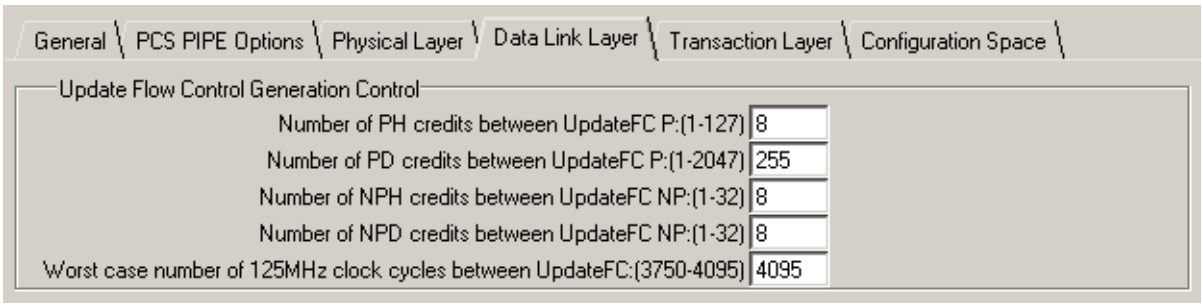
rx_lbk_kcntl

rx_lbk_data

Data Link Layer Tab

Figure 3-5 shows the contents of the Data Link Layer tab.

Figure 3-5. PCI Express IP Core Data Link Layer Options in the IPexpress Tool



Update Flow Control Generation Control

There are two times when an UpdateFC DLLP will be sent by the IP core. The first is based on the number of TLPs (header and data) that were processed. The second is based on a timer.

For both controls a larger number will reduce the amount of UpdateFC DLLPs in the transmit path resulting in more throughput for the transmit TLPs. However, a larger number will also increase the latency of releasing credits for the far end to transmit more data to the endpoint. A smaller number will increase the amount of UpdateFC DLLPs in the transmit path. But, the far end will see credits available more quickly.

Number of P TLPs Between UpdateFC

This control sets the number of Posted Header TLPs that have been processed before sending an UpdateFC-P.

Number of PD TLPs Between UpdateFC

This control sets the number of Posted Data TLPs (credits) that have been processed before sending an UpdateFC-P.

Number of NP TLPs Between UpdateFC

This control sets the number of Non-Posted Header TLPs that have been processed before sending an UpdateFC-NP.

Number of NPD TLPs Between UpdateFC

This control sets the number of Non-Posted Data TLPs (credits) that have been processed before sending an UpdateFC-NP.

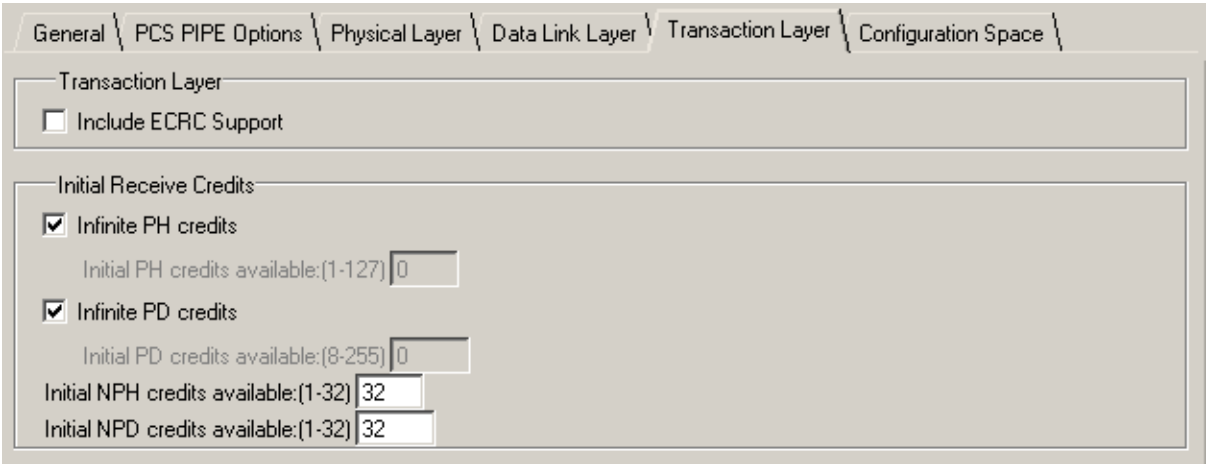
Worst Case Number of 125MHz Clock Cycles Between UpdateFC

This is the timer control that is used to send UpdateFC DLLPs. The core will send UpdateFC DLLPs for all three types when this timer expires regardless of the number of credits released.

Transaction Layer Tab

Figure 3-6 shows the contents of the Transaction Layer tab.

Figure 3-6. PCI Express IP Core Transaction Layer Options in the IPexpress Tool



Include ECRC Support

This option includes the ECRC generation and checking logic into the IP core. The ECRC logic is only utilized if the user enables this feature using the top level ports ecrc_gen_enb and ecrc_chk_enb. Not including this features saves nearly 1k LUTs from the core.

Initial Receive Credits

During the Data Link Layer Initialization InitFC1 and InitFC2 DLLPs are transmitted and received. This function is to allow both ends of the link to advertise the amount of credits available. The following controls are used to set the amount of credits available that the IP core will advertise during this process.

Infinite PH Credits

This option is used if the endpoint will have an infinite buffer for PH credits. This is typically used if the endpoint will terminate any PH TLP immediately.

Initial PH Credits Available

If PH infinite credits are not used then this control allows the user to set a initial credit value. This will be based on the receive buffering that exists in the user's design connected to the receive interface.

Infinite PD Credits

This option is used if the endpoint will have an infinite buffer for PD credits. This is typically used if the endpoint will terminate any PD TLP immediately.

Initial PD Credits Available

If PD infinite credits are not used then this control allows the user to set a initial credit value. This will be based on the receive buffering that exists in the user's design connected to the receive interface.

Initial NPH Credits Available

This option allows the user to set a initial credit value. This will be based on the receive buffering that exists in the user's design connected to the receive interface.

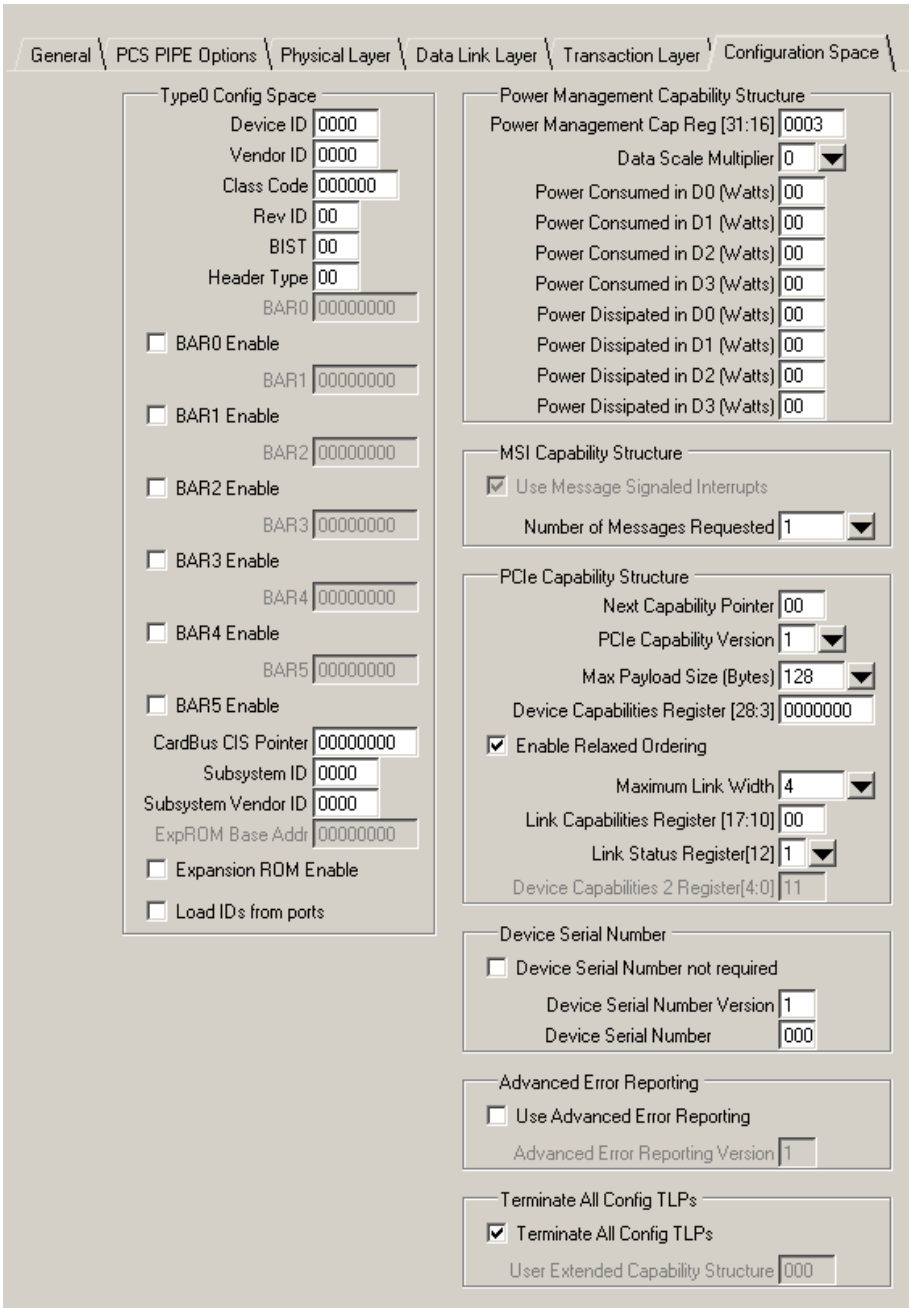
Initial NPD Credits Available

This option allows the user to set a initial credit value. This will be based on the receive buffering that exists in the user's design connected to the receive interface.

Configuration Space Tab

Figure 3-7 shows the contents of the Configuration Space tab.

Figure 3-7. PCI Express IP Core Configuration Space Options in the IPexpress Tool



Configuration Space Options

This page of the IPexpress tool allows the user to set the initial setting of the configuration space of the PCI Express IP core. The IP core contains the Type0, Power Management, PCI Express, MSI, AER, and Device Serial Number configuration structures.

Type 0 Config Space

This section provides relevant PCI Express settings for the legacy Type0 space.

Device ID

This 16-bit read only register is assigned by the manufacturer to identify the type of function of the endpoint.

Vendor ID

This 16-bit read only register assigned by the SIG to identify the manufacturer of the endpoint.

Class Code

This 24-bit read only register is used to allow the OS to identify the type of endpoint.

Revision ID

This 8-bit read only register is assigned by the manufacturer and identifies the revision number of the endpoint.

BIST

This 8-bit read only register indicates if a Built-In Self-Test is implemented by the function.

Header Type

This 8-bit read only register identifies the Header Type used by the function.

BAR Enable

This option enables the use of the particular Base Address Register (BAR).

BAR

This field allows the user to program the BAR to request a specific amount of address space from the system. If using 64-bit BARs then the BARs will be paired. BARs 0 and 1 will be paired with BAR0 for the LSBs and BAR1 for the MSBs. BARs 2 and 3 will be paired with BAR2 for the LSBs and BAR3 for the MSBs. BARs 4 and 5 will be paired with BAR4 for the LSBs and BAR5 for the MSBs.

For more details on BAR register bits, refer to the Configuration Space section of the PCI Local Bus Specification Revision 3.0.

The following section provides an example for requesting address space by setting BAR registers.

Bit [0] – 0 for memory space request. 1 for I/O space request.

Bits [2:1] – 00 for 32-bit memory address space. 10 for 64-bit memory address space.

Bit [3] – 1 for prefetchable memory. 0 for non-prefetchable memory.

Bits [31:4] – Indicate the size of required address space by resetting least significant bits.

Example 1: 32'hFFFF_F000 requests for memory space(bit[0]=0),32-bit address space(bit[2:1]=00), non-prefetchable memory(bit[3]=0) and 4KB address space (bits[31:4]=FFFF_F00)

Example 2: 32'hFFF0_0000 requests for memory space(bit[0]=0),32-bit address space(bit[2:1]=00), non-prefetchable memory(bit[3]=0) and 1MB address space (bits[31:4]=FFF0_000)

CardBus CIS Pointer

This is an optional register used in card bus system architecture and points to the Card Information Structure (CIS) on the card bus device. Refer to *PCI Local Bus Specification Revision 3.0* for further details.

Subsystem ID

This 16-bit read only register assigned by the manufacturer to identify the type of function of the endpoint.

Subsystem Vendor ID

This 16-bit read only register assigned by SIG to identify the manufacturer of the endpoint.

Expansion ROM Enable

This option enables the Expansion ROM to be used.

Expansion ROM Enable

The Expansion ROM base address if one is used in the solution.

Load IDs From Ports

This option provides ports for the user to set the Device ID, Vendor ID, Class Code, Rev ID, Subsystem ID, and Subsystem Vendor ID from the top level of the core. This is useful for designs which use the same hardware with different software drivers.

Power Management Capability Structure

This section includes options for the Power Management Capability Structure. This structure is used to pass power information from the endpoint to the system. If power management is not going to be used by the solution then all fields can remain in the default state.

Power Management Cap Reg (31:16)

This field sets the Power Management Capabilities (PMC) register bits 31:16.

Data Scale Multiplier

This control sets the Data Scale Multiplier used by system to multiplier the power numbers provided by the endpoint.

Power Consumed in D0, D1, D2, D3

These controls allow the user to specify the power consumed by the endpoint in each power state D0, D1, D2, and D3. The user specifies Watts as a 8-bit hex number.

Power Dissipated in D0, D1, D2, D3

These controls allow the user to specify the power dissipated by the endpoint in each power state D0, D1, D2, and D3. The user specifies Watts as a 8-bit hex number.

Message Signaled Interrupts Capability Structure Options

These controls allow the user to include MSI and request a certain number of interrupts.

Use Message Signaled Interrupts

This option includes MSI support in the IP core.

Number of Messages Requested

This number specifies how many MSIs will be requested by the endpoint of the system. The system will respond with how many interrupts have been provided. The number of interrupts provided can be found on the mm_enable port of the IP core.

PCI Express Capability Structure Options

These controls allow the user to control the PCI Express Capability Structure.

Next Capability Pointer

This control defines the pointer to additional non-extended capability implemented in the user application design. The default is 0 to indicate that there are no user implemented non-extended capability. If the pointer is set to a non-zero value, “Terminate All Config TLPs” must not be selected.

PCI Express Capability Version

Indicates the version of the PCI Express Capability Register. This number must be set to 2 for PCI Express version 2.0 (only for the LatticeECP3 family of devices) and 1 for PCI Express version 1.1 for most applications.

Max Payload Size

This option allows the endpoint to advertise the max payload size supported by the endpoint, and is used to size the Retry Buffer contained in the Data Link Layer. The user should select the largest size payload size that will be used in the application. The option 512B retry buffer size should also be selected for payload size size of 128B and 256B. The retry buffer uses Embedded Block RAM (EBR) and will be sized accordingly. Table 3-2 provides a total EBR count for the core based on Max Payload Size.

Table 3-2. Total EBR Count Based on Max Payload Size

Max Payload Size	LatticeECP3/LatticeECP2M Native x1 EBR Usage	LatticeECP3/LatticeECP2M Native x4 and x1 Downgraded EBR Usage	LatticeSCM EBR Usage[1, 2]
512B	4	11	11
1KB	5	11	11
2KB	9	13	13
4KB	15	20	20

1. The LatticeSCM EBR count will be reduced by 8 for each instance of the PCI Express IP core in addition to the number in the table above. This reduction is due to dedicated connections from certain EBRs to the flexiMAC MACO block, when used.
2. When using the LatticeSCM MACO LTSSM, 4 EBRs are available solely for connections to the LTSSM MACO site. The total number of EBRs available in the device will be reduced by 4 for each LTSSSM MACO block used.

Device Capabilities Register (27:3)

This 25-bit field sets the Device Capabilities Register bits 27:3.

Enable Relaxed Ordering

Relaxed ordering is the default setting for PCI Express. If the PCI Express link does not support relaxed ordering then this checkbox should be cleared. This feature does not change the behavior of the core, only the setting of this bit in the PCI Express capability structure. The user will be required to ensure strict ordering is enforced by the transmitter.

Maximum Link Width

This option sets the maximum link width advertised by the endpoint. This control should match the intended link width of the endpoint.

Link Capabilities Register (17:10)

This 8-bit field is not used in the generated IP core.

Device Capabilities 2 Register (4:0)

This 5-bit field sets the Device Capabilities Register bits 4:0.

Device Serial Number Version

Indicates the version of the Device Serial Number Capability. This number must always be set to 1 for v1.1.

Device Serial Number

This 64-bit value is provided in the IP core through the Device Serial Number Capability Structure.

Use Advanced Error Reporting

This control will include AER in the IP core. AER is used to provide detailed information on the status of the PCI Express link and errored TLPs.

Advanced Error Reporting Version

Indicates the version of the Advanced Error Reporting Capability. This number must always be set to 1 for v1.1.

Terminate All Configuration TLPs

If enabled, this control will allow the core to terminate all Configuration requests. The user will not need to handle any configuration requests in the user's design. If the user wants to implement other capabilities not contained.

User Extended Capability Structure

This control defines the pointer to additional non-extended capability implemented in the user application design. The default is 0 to indicate there is no user implemented non-extended capability. If the pointer is set to a non-zero value, “Terminate All Config TLPs” must not be selected.

IP Core Generation and Evaluation

This chapter provides information on licensing the PCI Express IP core, generating the core using the Diamond or ispLEVER software IPexpress tool, running functional simulation, and including the core in a top-level design.

The Lattice PCI Express IP core can be used in LatticeSCM, LatticeECP2M and LatticeECP3 device families. For the LatticeSCM versions, the majority of the IP core is implemented with pre-engineered and hard-wired MACO structured ASIC blocks integrated in the LatticeSCM device. Each LatticeSCM device contains a different collection of MACO IP. Refer to the Lattice web pages on LatticeSCM and MACO IP for more information.

Licensing the IP Core

An IP license is required to enable full, unrestricted use of the PCI Express IP core in a complete, top-level design. The specific PCI Express IP licensing requirements are different for targeting the LatticeECP2M and LatticeECP3 families versus targeting the LatticeSCM family.

Licensing Requirements for LatticeECP2M/LatticeECP3

An IP license that specifies the IP core (PCI Express), device family (ECP2M or ECP3) and configuration (x1 or x4) is required to enable full use of the PCI Express IP core in LatticeECP2M or LatticeECP3 devices. Instructions on how to obtain licenses for Lattice IP cores are given at:

http://www.latticesemi.com/Products/DesignSoftwareAndIP.aspx

Users may download and generate the PCI Express IP core for Lattice ECP2M and LatticeECP3 and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The PCI Express IP core for LatticeECP2M and Lattice ECP3 also supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited time (approximately four hours) without requiring an IP license (see the Hardware Evaluation section for further details). However, a license is required to enable timing simulation, to open the design in the Diamond or ispLEVER EPIC tool, and to generate bitstreams that do not include the hardware evaluation timeout limitation.

Note that there are no specific IP licensing requirements associated with an x4 core that functionally supports the ability to downgrade to an x1 configuration. Such a core is licensed as an x4 configuration.

Licensing Requirements for LatticeSCM

LatticeSCM MACO IP licenses are required to enable full use of MACO IP cores, including the PCI Express IP core for LatticeSCM. These MACO IP licenses are included as part of the standard Diamond or ispLEVER software license. There are no additional licensing requirements associated with using the PCI Express IP core in LatticeSCM devices.

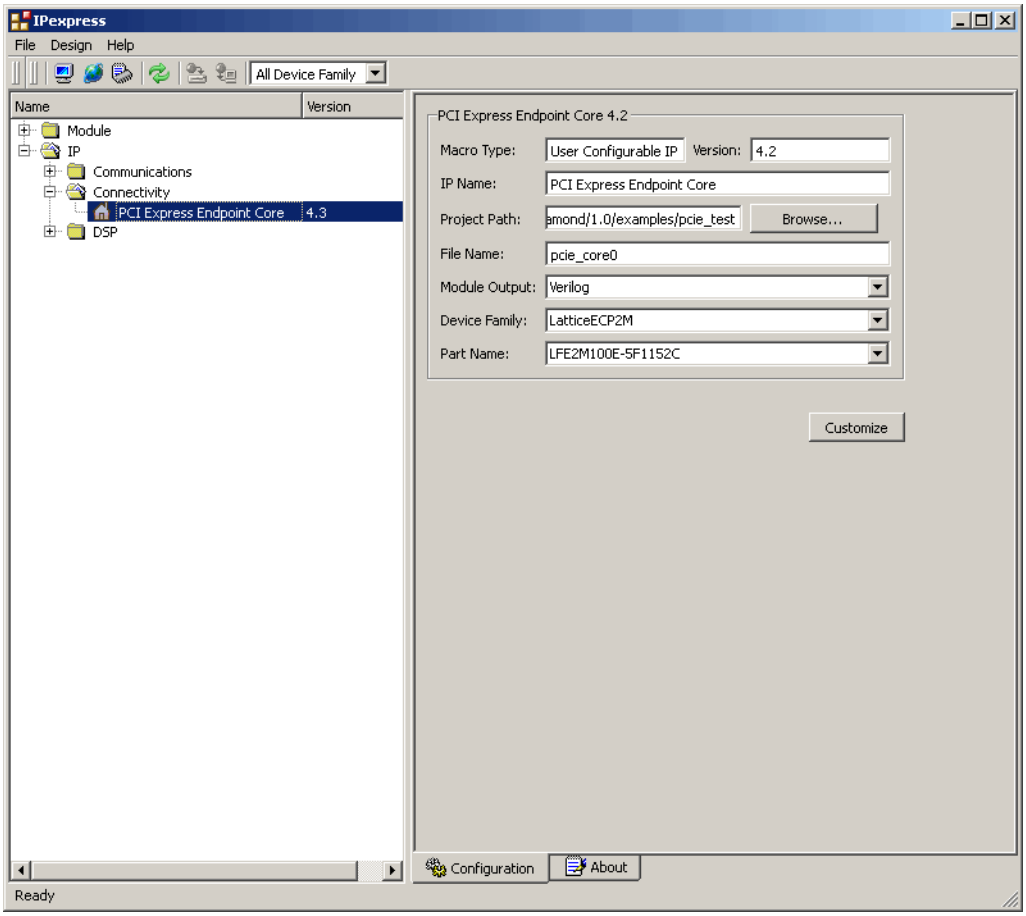
Getting Started

The PCI Express IP core is available for download from the Lattice IP server using the IPexpress tool. The IP files are automatically installed using ispUPDATE technology in any customer-specified directory. After the IP core has been installed, the IP core will be available in the IPexpress GUI dialog box shown in Figure 4-1.

The IPexpress tool GUI dialog box for the PCI Express IP core is shown in Figure 4-1. To generate a specific IP core configuration the user specifies:

- **Project Path** – Path to the directory where the generated IP files will be located.
- **File Name** – "username" designation given to the generated IP core and corresponding folders and files.
- **(Diamond) Module Output** – Verilog or VHDL.
- **(ispLEVER) Design Entry Type** – Verilog HDL or VHDL.
- **Device Family** – Device family to which IP is to be targeted (e.g. LatticeSCM, Lattice ECP2M, LatticeECP3, etc.). Only families that support the particular IP core are listed.
- **Part Name** – Specific targeted part within the selected device family.

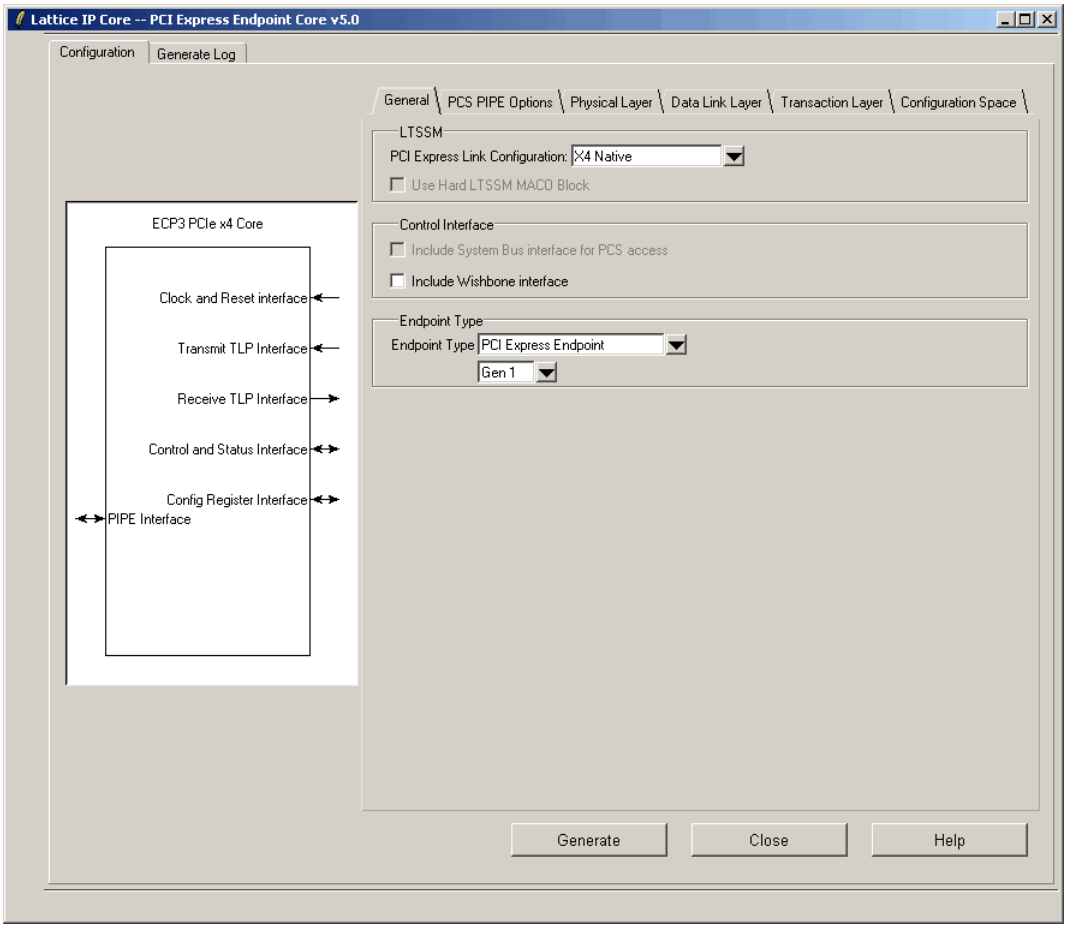
Figure 4-1. IPexpress Tool Dialog Box (Diamond Version)



Note that if the IPexpress tool is called from within an existing project, Project Path, Module Output (Design Entry in ispLEVER), Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information.

To create a custom configuration, the user clicks the **Customize** button in the IPexpress tool dialog box to display the PCI Express IP core Configuration GUI, as shown in Figure 4-2. From this dialog box, the user can select the IP parameter options specific to their application. Refer to the Parameter Settings section for more information on the PCI Express parameter settings. Additional information and known issues about the PCI Express IP core are provided in a ReadMe document that may be opened by clicking on the Help button in the Configuration GUI.

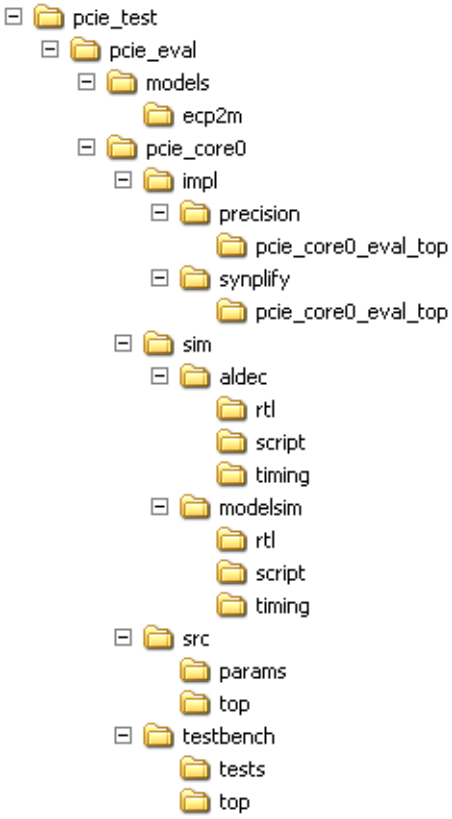
Figure 4-2. IPexpress Configuration GUI (Diamond Version)



IPexpress-Created Files and Top Level Directory Structure

When the user clicks the **Generate** button in the IP Configuration dialog box, the IP core and supporting files are generated in the specified "Project Path" directory. The directory structure of the generated files is shown in Figure 4-3.

Figure 4-3. LatticeECP2M PCI Express Core Directory Structure



The design flow for IP created with the IPexpress tool uses a post-synthesized module (NGO) for synthesis and a protected model for simulation. The post-synthesized module is customized and created during the IPexpress tool generation. The protected simulation model is not customized during the IPexpress tool process, and relies on parameters provided to customize behavior during simulation.

Table 4-1 provides a list of key files and directories created by the IPexpress tool and how they are used. The IPexpress tool creates several files that are used throughout the design cycle. The names of most of the files created are customized to the user's module name specified in the IPexpress tool.

Table 4-1. File List

File	Sim	Synthesis	Description
<*username*>_inst.v			This file provides an instance template for the IP.
<*username*>.v	Yes		This file provides the PCI Express core for simulation.
<*username*>_beh.v	Yes		This file provides the front-end simulation library for the PCI Express core.
pci_exp_params.v	Yes		This file provides the user options of the IP for the simulation model.
pci_exp_ddefines.v	Yes		This file provides parameters necessary for the simulation.
<*username*>_bb.v		Yes	This file provides the synthesis black box for the user's synthesis.

Table 4-1. File List (Continued)

File	Sim	Synthesis	Description
<*username*>.ngo		Yes	This file provides the synthesized IP core used by the Diamond or ispLEVER software. This file needs to be pointed to by the Build step by using the search path property.
PCS autoconfig file	Yes	Yes	This file contains the PCS/SERDES memory map initialization. This file must be copied in to the simulation directory as well as the Diamond or ispLEVER software project directory. For the LatticeSC this file is named pcie_pcs.txt. For the LatticeECP3/LatticeECP2M x4 Native and x1 Downgraded this file is named pcs_pipe_8b_x4.txt. For the LatticeECP3/LatticeECP2M x1 Native this file is named pcs_pipe_8b_x1.txt.
<*username*>.lpc			This file contains the IPexpress tool options used to recreate or modify the core in the IPexpress tool.
<*username*>.ipx			The IPX file holds references to all of the elements of an IP or Module after it is generated from the IPexpress tool (Diamond version only). The file is used to bring in the appropriate files during the design implementation and analysis. It is also used to re-load parameter settings into the IP/Module generation GUI when an IP/Module is being re-generated.
pmi_*.ngo			These files contain the memories used by the IP core. These files need to be pointed to by the Build step by using the search path property.
<*username*>_eval			This directory contains a sample design. This design is capable of performing a simulation and running through the Diamond or ispLEVER software.
LatticeECP3/LatticeECP2M-Specific Files			
<*username*>_top.[v,vhd]	Optional	Optional	This file provides a module which instantiates the PCI Express core and the PIPE interface. This file can be easily modified for the user's instance of the PCI Express core. This file is located in the `<username>_eval/pcie/src/top` directory.
pcs_pipe_top.v	Yes		This is the top level of the PIPE interface. This file is located in the `<username>_eval/models/[ecp3/ecp2m]` directory. All of the HDL files located in this directory are used to simulate the PIPE interface.
pcs_pipe_bb.v	Yes		This file provides the synthesis black box for the PIPE interface. This file is located in the `<username>_eval/models/[ecp3/ecp2m]` directory.
pcs_pipe_top.ngo	Yes		This file provides the synthesized PIPE interface used by the Diamond or ispLEVER software. This file needs to be pointed to by the Build step using the search path property.
LatticeSCM-Specific Files			
<*username*>_ba_beh.v	Yes		This file provides the back-end simulation library for the PCI Express core.

Most of the files required to use the PCI Express IP core in a user's design reside in the root directory created by the IPexpress tool. This includes the synthesis black box, simulation model, and example preference file.

The `\pcie_eval` and subtending directories provide files supporting PCI Express IP core evaluation. The `\pcie_eval` directory contains files/folders with content that is constant for all configurations of the PCI Express IP core. The `\<username>` subfolder (`\pcie_core0` in this example) contains files/folders with content specific to the <*username*> configuration.

The PCI Express ReadMe document is also provided in the `\pcie_eval` directory.

For example information and known issues on this core, see the Lattice PCI Express ReadMe document. This file is available when the core is installed in the Diamond or ispLEVER software. The document provides information on creating an evaluation version of the core for use in Diamond or ispLEVER and simulation.

The **\pcie_eval** directory is created by the IPexpress tool the first time the core is generated and updated each time the core is regenerated. A **\<***username***>** directory is created by the IPexpress tool each time the core is generated and regenerated each time the core with the same file name is regenerated. A separate **\<username>** directory is generated for cores with different names, e.g. **\<my_core_0>**, **\<my_core_1>**, etc.

The **\pcie_eval** directory provides an evaluation design which can be used to determine the size of the IP core and a design which can be pushed through the Diamond or ispLEVER software including front-end and timing simulations. The models directory provides the library element for the PCS (and PIPE interface for LatticeECP3 and LatticeECP2M).

The **\<***username***>** directory contains the sample design for the configuration specified by the customer. The \<*username*>**\impl** directory provides project files supporting Precision RTL and Synplify synthesis flows. The sample design pulls the user ports out to external pins. This design and associated project files can be used to determine the size of the core and to push it through the mechanics of the Diamond or ispLEVER software design flow.

The **\<***username***>\sim** directory provides project files supporting RTL and timing simulation for both the Active-HDL and ModelSim simulators. The **\<***username***>\src** directory provides the top-level source code for the evaluation design. The **\testbench** directory provides a top-level testbench and test case files.

Instantiating the Core

The generated PCI Express IP core package includes black-box (<*username*>_bb.v) and instance (<*username*>_inst.v) templates that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file that can be used as an instantiation template for the IP core is provided in **\<***project_dir***>\pcie_eval\<***username***>\src\top**. Users may also use this top-level reference as the starting template for the top-level for their complete design.

Running Functional Simulation

Simulation support for the PCI Express IP core is provided for Aldec and ModelSim simulators. The PCI Express core simulation model is generated from the IPexpress tool with the name <*username*>.v. This file calls <*username*>_beh.v which contains the obfuscated simulation model. An obfuscated simulation model is Lattice's unique IP protection technique which scrambles the Verilog HDL while maintaining logical equivalence. VHDL users will use the same Verilog model for simulation.

When compiling the PCI Express IP core the following files must be compiled with the model.

- pci_exp_params.v
- pci_exp_ddefines.v

These files provide "define constants" that are necessary for the simulation model.

The ModelSim environment is located in **\<***project_dir***>\pcie_eval\<***username***>\sim\modelsim**. Users can run the ModelSim simulation by performing the following steps:

1. Open ModelSim.
2. Under the File tab, select **Change Directory** and choose folder **\<***project_dir***>\pcie_eval\<***username***>\sim\modelsim**.
3. Under the Tools tab, select **Tcl > Execute Macro** and execute one of the ModelSim "do" scripts shown, depending on which version of ModelSim is used (ModelSim SE or the Lattice OEM version).

The Aldec Active-HDL environment is located in **\<***project_dir***>\pcie_eval\<***username***>\sim\aldec**. Users can run the Aldec evaluation simulation by performing the following steps:

1. Open Active-HDL.

2. Under the Tools tab, select **Execute Macro**.

3. Browse to the directory **\<*project_dir*>\pcie_eval\<*username*>\sim\aldec** and execute the Active-HDL "do" script shown.

Synthesizing and Implementing the Core in a Top-Level Design

The PCI Express IP core itself is synthesized and provided in NGO format when the core is generated through the IPexpress tool. You can combine the core in your own top-level design by instantiating the core in your top level file as described in the Instantiating the Core section and then synthesizing the entire design with either Synplify or Precision RTL Synthesis.

The top-level file <*username*>_eval_top.v provided in
\<*project_dir*>\pcie_eval\<*username*>\src\top
supports the ability to implement the PCI Express core in isolation. Push-button implementation of this top-level design with either Synplify or Precision RTL Synthesis is supported via the project files
<username>_eval.ldf (Diamond) or .syn (ispLEVER) located in the
\<*project_dir*>\pcie_eval\<*username*>\impl\synplify
and the
\<*project_dir*>\pcie_eval\<*username*>\impl\precision directories, respectively.

To use this project file in Diamond:

1. Choose **File > Open > Project**.

2. Browse to **\<*project_dir*>\pcie_eval\<*username*>\impl\(synplify** or **precision**) in the Open Project dialog box.

3. Select and open <*username*>.ldf. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.

4. Select the **Process** tab in the left-hand GUI window.

5. Implement the complete design via the standard Diamond GUI flow.

To use this project file in ispLEVER:

1. Choose **File > Open Project.**

2. Browse to **\<*project_dir*>\pcie_eval\<*username*>\impl\(synplify** or **precision**) in the Open Project dialog box.

3. Select and open <*username*>.syn. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.

4. Select the device top-level entry in the left-hand GUI window.

5. Implement the complete design via the standard ispLEVER GUI flow.

Hardware Evaluation

The PCI Express IP core supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of IP cores that operate in hardware for a limited period of time (approximately four hours) without requiring the purchase on an IP license. It may also be used to evaluate the core in hardware in user-defined designs.

Enabling Hardware Evaluation in Diamond

Choose **Project > Active Strategy > Translate Design Settings**. The hardware evaluation capability may be enabled/disabled in the Strategy dialog box. It is enabled by default.

Enabling Hardware Evaluation in ispLEVER

In the Processes for Current Source pane, right-click the **Build Database** process and choose **Properties** from the dropdown menu. The hardware evaluation capability may be enabled/disabled in the Properties dialog box. It is enabled by default.

Updating/Regenerating the IP Core

By regenerating an IP core with the IPexpress tool, you can modify any of its settings including: device type, design entry method, and any of the options specific to the IP core. Regenerating can be done to modify an existing IP core or to create a new but similar one.

Regenerating an IP Core in Diamond

To regenerate an IP core in Diamond:

1. In IPexpress, click the **Regenerate** button.
2. In the Regenerate view of IPexpress, choose the IPX source file of the module or IP you wish to regenerate.
3. IPexpress shows the current settings for the module or IP in the Source box. Make your new settings in the T**arget** box.
4. If you want to generate a new set of files in a new location, set the new location in the **IPX Target File** box. The base of the file name will be the base of all the new file names. The IPX Target File must end with an .ipx extension.
5. Click **Regenerate.** The module's dialog box opens showing the current option settings.
6. In the dialog box, choose the desired options. To get information about the options, click **Help**. Also, check the About tab in IPexpress for links to technical notes and user guides. IP may come with additional information. As the options change, the schematic diagram of the module changes to show the I/O and the device resources the module will need.
7. To import the module into your project, if it's not already there, select **Import IPX to Diamond Project** (not available in stand-alone mode).
8. Click **Generate**.
9. Check the Generate Log tab to check for warnings and error messages.
10. Click **Close**.

The IPexpress package file (.ipx) supported by Diamond holds references to all of the elements of the generated IP core required to support simulation, synthesis and implementation. The IP core may be included in a user's design by importing the .ipx file to the associated Diamond project. To change the option settings of a module or IP that is already in a design project, double-click the module's .ipx file in the File List view. This opens IPexpress and the module's dialog box showing the current option settings. Then go to step 6 above.

Regenerating an IP Core in ispLEVER

To regenerate an IP core in ispLEVER:

1. In the IPexpress tool, choose **Tools > Regenerate IP/Module**.
2. In the Select a Parameter File dialog box, choose the Lattice Parameter Configuration (.lpc) file of the IP core you wish to regenerate, and click **Open**.
3. The Select Target Core Version, Design Entry, and Device dialog box shows the current settings for the IP core in the Source Value box. Make your new settings in the Target Value box.

4. If you want to generate a new set of files in a new location, set the location in the LPC Target File box. The base of the .lpc file name will be the base of all the new file names. The LPC Target File must end with an .lpc extension.
5. Click **Next**. The IP core's dialog box opens showing the current option settings.
6. In the dialog box, choose desired options. To get information about the options, click **Help**. Also, check the About tab in the IPexpress tool for links to technical notes and user guides. The IP core might come with additional information. As the options change, the schematic diagram of the IP core changes to show the I/O and the device resources the IP core will need.
7. Click **Generate**.
8. Click the **Generate Log** tab to check for warnings and error messages.

This chapter provides supporting information on how to use the PCI Express IP core in complete designs. Topics discussed include IP simulation and verification, FPGA design implementation and board-level implementation.

Simulation and Verification

This section discusses strategies and alternative approaches for verifying the proper functionality of the PCI Express core through simulation.

Simulation Strategies

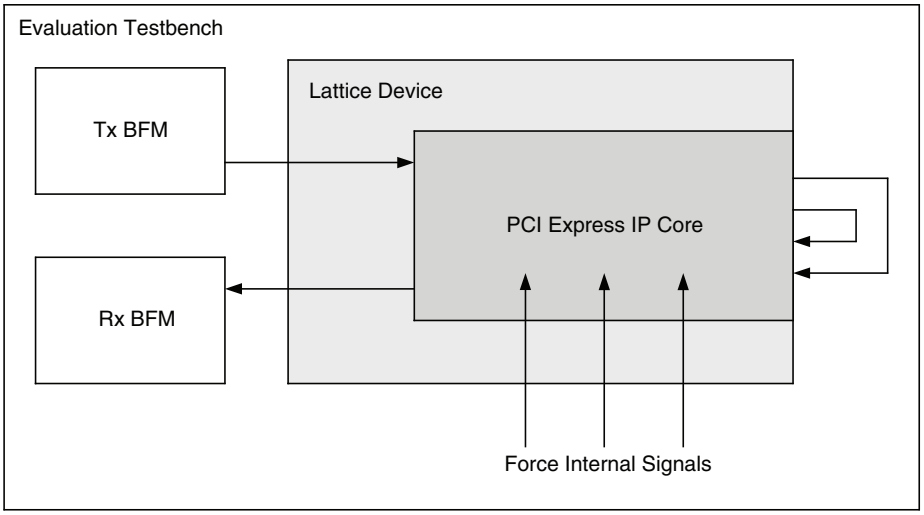
Included with the core from the IPexpress tool is the evaluation testbench located in the `<username>` directory. The intent of the evaluation testbench is to show the core performing in simulation, as well as to provide timing simulations post place and route. Many communication cores work in a loopback format to simplify the data generation process and to meet the simple objectives of this evaluation testbench. A loopback format has been used in this case as well.

In a real system, however, PCI Express requires that an upstream port connect to a downstream port. In the simple-to-use, Lattice-supplied eval testbench, a few force commands are used to force an L0 state as a x4 link. Other force commands are also used to kick off the credit processing correctly.

Once a link is established via a loopback with the core, a few TLPs are sent through the link to show the transmit and receive interface. This is the extent of the evaluation testbench.

Figure 5-1 illustrates the evaluation testbench process.

Figure 5-1. PCI Express x4 Core Evaluation Testbench Block Diagram



This testbench scheme works for its intent, but it is not easily extendible for the purposes of a Bus Functional Model (BFM) to emulate a real user system. Users can take the testbench provided and modify it to build in their own specific tests.

Sometimes the testbench is oriented differently than users anticipate. Users might wish to interface to the PCI Express core via the serial lanes. As an endpoint solution developer the verification should be performed at the endpoint of the system from the root complex device.

Refer to the Alternative Testbench Approach section for more information on setting up a testbench.

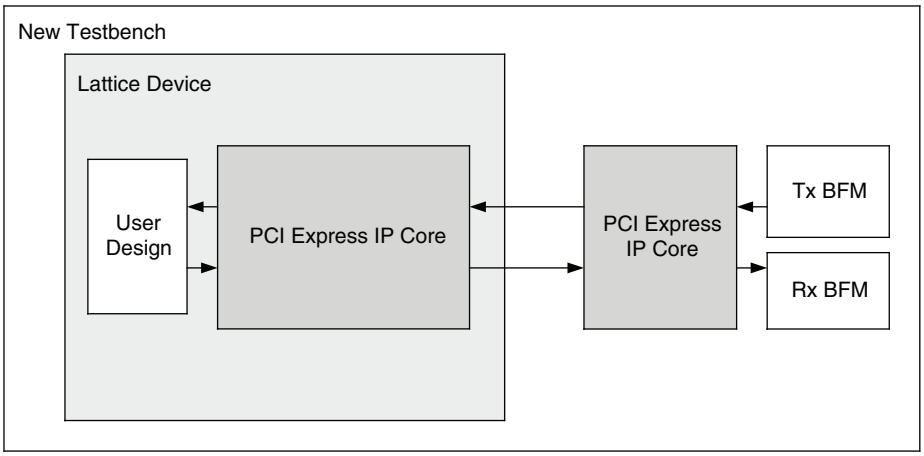
Users simulating a multi-lane core at the serial level should give consideration to lane ordering. Lane ordering is dependant on the layout of the chip on a board. Refer to the Board Layout Concerns for Add-in Cards section for further information.

Alternative Testbench Approach

In order to create a testbench which meets the user's needs, the data must be sourced across the serial PCI Express lanes. The user must also have the ability to create the source traffic that will be pushed over the PCI Express link. This solution can be created by the user using the Lattice core.

Figure 5-2 shows a block diagram that illustrates a new testbench orientation which can be created by the user.

Figure 5-2. PCI Express x4 Core Testbench Using Two Cores



Use two PCI Express cores. The first PCI Express core is used in the design and the second PCI Express core is used as a driver. The user needs to use the no_pcie_train command to force the L0 state of the LTSSM in both cores.

When IP Core does not use the Wishbone bus, the bench must force no_pcie_train port on the IP to “1” to set LTSSM to L0 status.

When the Wishbone bus is implemented, there is no no_pcie_train port on the IP Core. Therefore, the bench must set the “LTSSM no training” register to force LTSSM to L0 status.

Whether or not the Wishbone bus is implemented, the bench must force LTSSM to L0 after both LTSSM state machines of transmitter and receiver are moved to Configuration status (4’d2).

As a result, the second core can then be used as a traffic separator. The second core is created to be the opposite of the design core. Thus an upstream port will talk with a downstream port and vice versa. The second core is used as a traffic generator. User-written BFMs can be created to source and sink PCI Express TLPs to exercise the design.

An issue associated with this testbench solution is that the run time tends to be long since the testbench will now include two PCS/SERDES cores. There is a large number of functions contained in both of the IP blocks which will slow down the simulation. Another issue is that the Lattice PCI Express solution is being used to verify the Lattice PCI Express solution. This risk is mitigated by the fact that Lattice is PCI-SIG compliant (see the Integrator's list at www.pci-sig.com) and a third party verification IP was used during the development process.

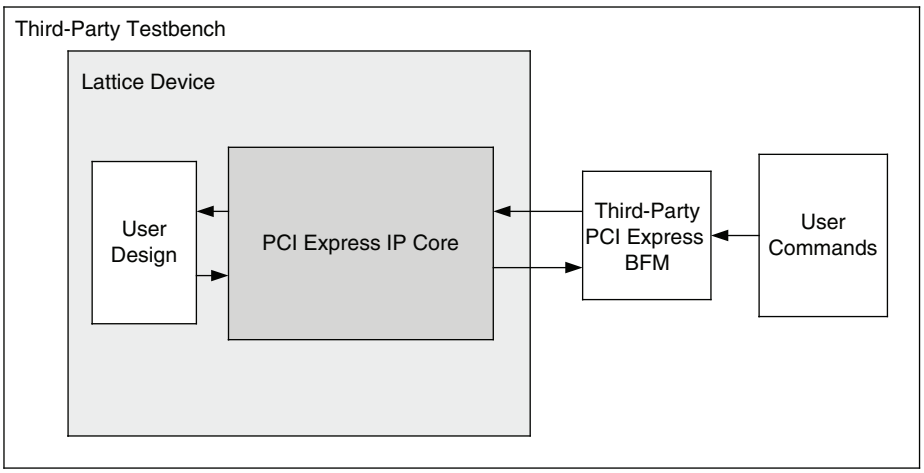
It should also be noted that this approach does not allow for PCI Express layer error insertion.

Third Party Verification IP

The ideal solution for system verification is to use a third party verification IP. These solutions are built specifically for the user's needs and supply the BFMs and provide easy to use interfaces to create TLP traffic. Also, models are behavioral, gate level, or even RTL to increase the simulation speed.

Lattice has chosen the Synopsys PCI Express verification IP for development of the PCI Express core, as shown in Figure 5-3. There are other third party vendors for PCI Express including Denali® and Cadence®.

Figure 5-3. PCI Express x4 Core Testbench with Third-Party VIP



If desired, an independent Bus Functional Model can be modified to emulate a user's environment. This option is highly recommended.

FPGA Design Implementation

This section provides information on implementing the PCI Express IP core in a complete FPGA design. Topics covered include how to set up the IP core for various link width combinations, clocking schemes and physically locating the IP core within the FPGA.

Setting Up the Core

This section describes how to set up the PCI Express core for various link width combinations. The user must provide a different PCS/SERDES autoconfig file based on the link width and the flipping of the lanes. The PCS/SERDES memory map is initially configured during bitstream loading using the autoconfig file generated with the IPexpress tool.

Note that transactions shown display data in hexadecimal format with bit 0 as the MSb.

Lane flipping is not applicable for x1 Native. The user can select which channel of the quad to be the active channel in the IPexpress tool.

Setting Up for x4 (No Flip)
This is the default condition that is created from the IPexpress tool. Simply use the autoconfig file to setup the channels. The flip_lanes port should be tied low.

Setting Up for x4 (Flipped)
LatticeECP3 and LatticeECP2M
No changes required. Simply use the pcs_pcie_8b_x4.txt file generated from the IPexpress tool.

LatticeSCM

If the design will be using flipped lanes then the PCS/SERDES channels need to understand that Lane 0 is now using Channel 3. The only change is to change Channel 3 to be the master channel for clocking. The following two lines need to be added in the pcie_pcs.txt file.

```
quad 01 FF  # Ch3 as MCA clock source
quad 02 30  # Ch3 as ref_p clock source
```

The flip_lanes port should be tied high.

Setting Up for x1 Downgraded (No Flip)

LatticeECP3 and LatticeECP2M

If the design will be using only a single channel and it is not flipped then Channels 1, 2, and 3 need to be powered down. Change the following lines from the pcs_pipe_x4.txt file.

```
CH0_MODE "GROUP1"
CH1_MODE "GROUP1"
CH2_MODE "GROUP1"
CH3_MODE "GROUP1"
```

to

```
CH0_MODE "GROUP1"
CH1_MODE "DISABLE"
CH2_MODE "DISABLE"
CH3_MODE "DISABLE"
```

The flip_lanes port should be tied low.

LatticeSCM

No changes required, simply use the pcs_pcie.txt file generated from the IPexpress tool.

Setting Up for x1 Downgraded (Flipped)

If the design will be using only a single channel and it is flipped then Channel 3 becomes the master channel and Channels 0, 1, and 2 to be powered down using the autoconfig file.

LatticeECP3 and LatticeECP2M

Change the following lines from the pcs_pcie_8b_x4.txt file.

```
CH0_MODE     "GROUP1"
CH1_MODE     "GROUP1"
CH2_MODE     "GROUP1"
CH3_MODE     "GROUP1"
```

to

```
CH0_MODE     "DISABLE"
CH1_MODE     "DISABLE"
CH2_MODE     "DISABLE"
CH3_MODE     "GROUP1"
```

The flip_lanes port should be tied high.

LatticeSCM

Remove the following lines from the pcie_pcs.txt file.

```
Ch0 13 03 # Powerup Channel
Ch0 00 01
ch1 13 03 # Powerup Channel
```

```
ch1 00 01
ch2 13 03 # Powerup Channel
ch2 00 01
quad 19 00 # MCA x4 alignment
quad 05 01 # MCA latency
quad 06 06 # MCA depth
quad 07 FF # MCA alignment mask
quad 08 BC # MCA alignment character
quad 09 BC # MCA alignment character
quad 0A 15 # MCA k control
ch0 14 93 # 16% Pre-emphasis, +12.5% output
ch1 14 93 # 16% Pre-emphasis, +12.5% output
ch2 14 93 # 16% Pre-emphasis, +12.5% output
ch0 15 10 # +6dB equalization
ch1 15 10 # +6dB equalization
ch2 15 10 # +6dB equalization
```

Now add the following lines.

```
quad 01 FF  # Ch3 as MCA clock source
quad 02 30  # Ch3 as ref_p clock source
```

The flip_lanes port should be tied high.

Setting Design Constraints

There are several design constraints that are required for the IP core. These constraints must be placed as preferences in the .lpf file. These preferences can be entered in the .lpf file through the Preference Editing View in Diamond, the Design Planner in ispLEVER, or directly in the text based .lpf file.

Several interfaces inside the core run at 250MHz. These internal clocks must be constrained for the place and route.

LatticeECP3 and LatticeECP2M

```
FREQUENCY NET "<instance name>/u1_pcs_pipe/ff_rx_fclk_0" 250 MHz ;
FREQUENCY NET "<instance name>/u1_pcs_pipe/ff_rx_fclk_1" 250 MHz ;
FREQUENCY NET "<instance name>/u1_pcs_pipe/ff_rx_fclk_2" 250 MHz ;
FREQUENCY NET "<instance name>/u1_pcs_pipe/ff_rx_fclk_3" 250 MHz ;
FREQUENCY NET "<instance name>/pclk" 250.000000 MHz ;
```

LatticeSCM

```
FREQUENCY NET "<instance name>/u1_flxmc_sys_pcie/sys_clk_250_inferred_clock"
   250 MHz ;
FREQUENCY NET "<instance name>/u1_flxmc_sys_pcie/ref_pclk" 250 MHz ;
```

There are also several places inside the PCI Express core which can be blocked from timing analysis. These paths are asynchronous control signals which are not critical. The paths can be blocked using the following preferences.

LatticeECP3 and LatticeECP2M

```
BLOCK PATH FROM CELL "*ctc_reset_chx*";
BLOCK NET "<instance name>/u1_pcs_pipe/sync_rst";
BLOCK NET "<instance name>/core_rst_n";
BLOCK NET "<instance name>/*rxp_status_ln0_2";
MULTICYCLE FROM CELL "*lbk_sloopback*" TO CELL "*cs_reqdet_sm*" 2 X;
MULTICYCLE FROM CELL "*lbk_sloopback*" TO CELL "*cnt_st*" 2 X;
MULTICYCLE FROM CELL "*lbk_sloopback*" TO CELL "*ffc_pcie_det_en*" 2 X;
```

```
MULTICYCLE FROM CELL "*lbk_sloopback*" TO CELL "*det_result*" 2 X;
MULTICYCLE FROM CELL "*nfts_rx_skp_cnt*" TO CELL "*cnt_done_nfts_rx*" 2 X;
MULTICYCLE FROM CELL "*nfts_rx_skp_cnt*" TO CELL "*ltssm_nfts_rx_skp*" 2 X;
```

LatticeSCM

```
MULTICYCLE FROM CELL "*lbk_sloopback*" TO CELL "*cs_reqdet_sm*" 2 X;
MULTICYCLE FROM CELL "*lbk_sloopback*" TO CELL "*cnt_st*" 2 X;
MULTICYCLE FROM CELL "*lbk_sloopback*" TO CELL "*ffc_pcie_det_en*" 2 X;
MULTICYCLE FROM CELL "*lbk_sloopback*" TO CELL "*det_result*" 2 X;
MULTICYCLE FROM CELL "*nfts_rx_skp_cnt*" TO CELL "*cnt_done_nfts_rx*" 2 X;
MULTICYCLE FROM CELL "*nfts_rx_skp_cnt*" TO CELL "*ltssm_nfts_rx_skp*" 2 X;
```

The user interface clock sys_clk_125 must be constrained to run at 125 MHz. Based on the connectivity of the design the name of this clock net might change.

```
FREQUENCY NET “sys_clk_125” 125 MHz;
```

LatticeSCM-Specific Preferences

The refclk_250 clock from the PLL to the PCI Express core needs to be routed on a primary clock route to provide the best signal integrity.

```
USE PRIMARY NET “refclk_250”;
```

There are several points in the design where data is transferred from the 125 MHz clock domain to the 250 MHz clock domain. These clock domain transfers are handled by the design. The timing tools need to be instructed not to include these clock domain crossings in timing analysis with the following preferences.

```
BLOCK PATH FROM CLKNET "<instance
name>/u1_flxmc_sys_pcie/sys_clk_250_inferred_clock" TO CLKNET "sys_clk_125" ;

BLOCK PATH FROM CLKNET "sys_clk_125" TO CLKNET "<instance
name>/u1_flxmc_sys_pcie/sys_clk_250_inferred_clock" ;
```

Errors and Warnings

During the process of running the Diamond or ispLEVER software there are several warning messages that will be created. This section documents the normal warning messages that will be present when using the PCI Express core.

LatticeECP3 and LatticeECP2M Errors and Warnings

Place & Route Design

The following warnings will be present in the place and route log file.

```
WARNING - par: The driver of primary clock net
          pcie/u1_pcs_pipe/ff_rx_fclk_0 is not placed on one of the PIO
          sites which are dedicated for primary clocks.  This primary clock
          will be routed to a H-spine through general routing resource or be
          routed as secondary clock and may suffer from excessive delay or
          skew.

WARNING - par: The driver of primary clock net
          pcie/u1_pcs_pipe/ff_rx_fclk_1 is not placed on one of the PIO
          sites which are dedicated for primary clocks.  This primary clock
          will be routed to a H-spine through general routing resource or be
          routed as secondary clock and may suffer from excessive delay or
          skew.
```

```
WARNING - par: The driver of primary clock net
          pcie/u1_pcs_pipe/ff_rx_fclk_2 is not placed on one of the PIO
          sites which are dedicated for primary clocks.  This primary clock
          will be routed to a H-spine through general routing resource or be
          routed as secondary clock and may suffer from excessive delay or
          skew.

WARNING - par: The driver of primary clock net
          pcie/u1_pcs_pipe/ff_rx_fclk_3 is not placed on one of the PIO
          sites which are dedicated for primary clocks.  This primary clock
          will be routed to a H-spine through general routing resource or be
          routed as secondary clock and may suffer from excessive delay or
          skew.
```

These warnings inform the user that the receive clocks from the PCS module are using general routing to connect to a primary clock connection point. This is expected for this architecture and clock connection.

LatticeSCM Errors and Warnings

Map Design

The following warnings will be present in the map log file.

```
WARNING - map: Security for MACO block
     pcie/u1_flxmc_sys_pcie/u1_flxmc_pcie_core/u1_phy_dll/u1_flxmc_top_ebr/fl
     xmc_top_mib will be checked during bitgen

WARNING - map: Security for MACO block
     pcie/u1_flxmc_sys_pcie/u1_flxmc_pcie_core/u1_phy_dll/LTSSM will be
     checked during bitgen
```

These warnings inform the user that during the Generate Bitstream process a license for the MACO cores of the flexiMAC and LTSSM will be required to create the bitstream.

Generate Bitstream

The following warnings will be present in the bitgen log file.

```
WARNING - blockcheck: SLICE
          pcie/u1_flxmc_sys_pcie/u1_flxmc_pcie_core/u1_phy_dll/maco_pmi_distr
          ibuted_dpram/SLICE_544 in SPRAM/DPRAM mode, LSR is held hi, CE is
          held hi.  Doing so enables writes continuously to the RAM.

WARNING - blockcheck: SLICE
          pcie/u1_flxmc_sys_pcie/u1_flxmc_pcie_core/u1_phy_dll/maco_pmi_distr
          ibuted_dpram/SLICE_545 in SPRAM/DPRAM mode, LSR is held hi, CE is
          held hi.  Doing so enables writes continuously to the RAM.

WARNING - blockcheck: SLICE
          pcie/u1_flxmc_sys_pcie/u1_flxmc_pcie_core/u1_phy_dll/maco_pmi_distr
          ibuted_dpram/SLICE_546 in SPRAM/DPRAM mode, LSR is held hi, CE is
          held hi.  Doing so enables writes continuously to the RAM.

WARNING - blockcheck: SLICE
          pcie/u1_flxmc_sys_pcie/u1_flxmc_pcie_core/u1_phy_dll/maco_pmi_distr
          ibuted_dpram/SLICE_547 in SPRAM/DPRAM mode, LSR is held hi, CE is
          held hi.  Doing so enables writes continuously to the RAM.

WARNING - blockcheck: No signals are connected to component osc.
```

These warnings inform the user that a SLICE is programmed in DPRAM mode which allows a constant write to the RAM. This is an expected implementation of the RAM which is used in the PCI Express design.

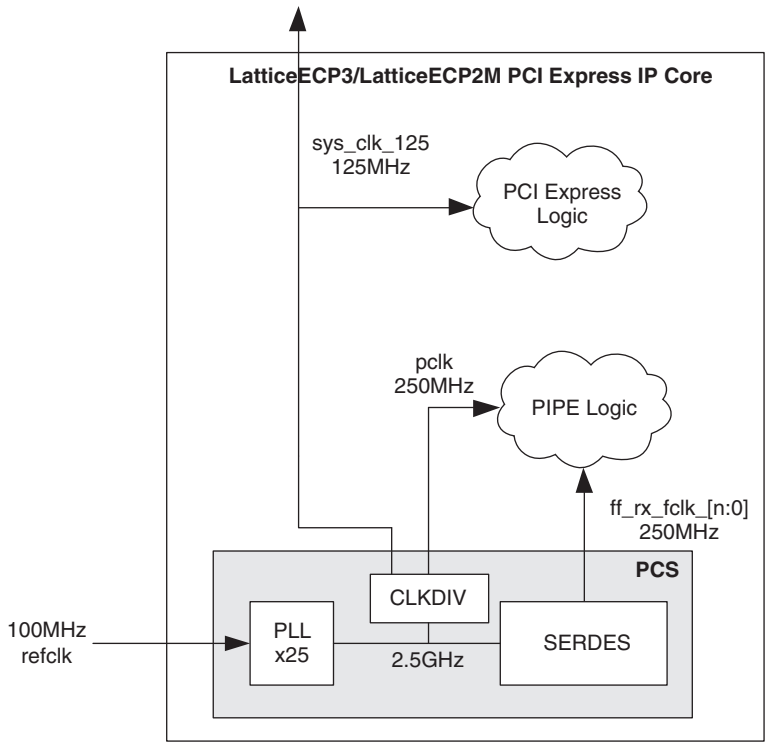
Clocking Scheme

A PCI Express link is typically provided with a 100MHz reference clock from which the 2.5Gbps data rate is achieved. The user interface for the PCI Express IP core is clocked using a 125MHz clock (sys_clk_125).

LatticeECP3 and LatticeECP2M Clocking

Figure 5-4 provides the internal clocking structures of the IP core in the LatticeECP3 and LatticeECP2M families.

Figure 5-4. LatticeECP3 and LatticeECP2M PCI Express Clocking Scheme



The LatticeECP3 and LatticeECP2M clocking solution uses the 100MHz differential refclk provided from the PCI Express link connected directly to the REFCLKP/N of the SERDES. The 100 Ω differential termination is included inside the SERDES so external resistors are not required on the board. It is recommended that both the sys_clk_125 and pclk clock nets are routed using primary clock routing.

Inside the SERDES, a PLL creates the 2.5 Gbps rate from which a transmit 250 MHz clock (pclk) and recovered clock(s) (ff_rx_fclk_[n:0]) are derived. The Lattice PCI Express core then performs a clock domain change to the sys_clk_125 125 MHz clock for the user interface.

The reset sequence logic for the SERDES/PCS present in the pcs_pipe_top module uses a CLKDIV component to derive clock from the 100 Mhz reference clock.

Table 5-1 shows clocking resources used with LatticeECP3 and LatticeECP2M.

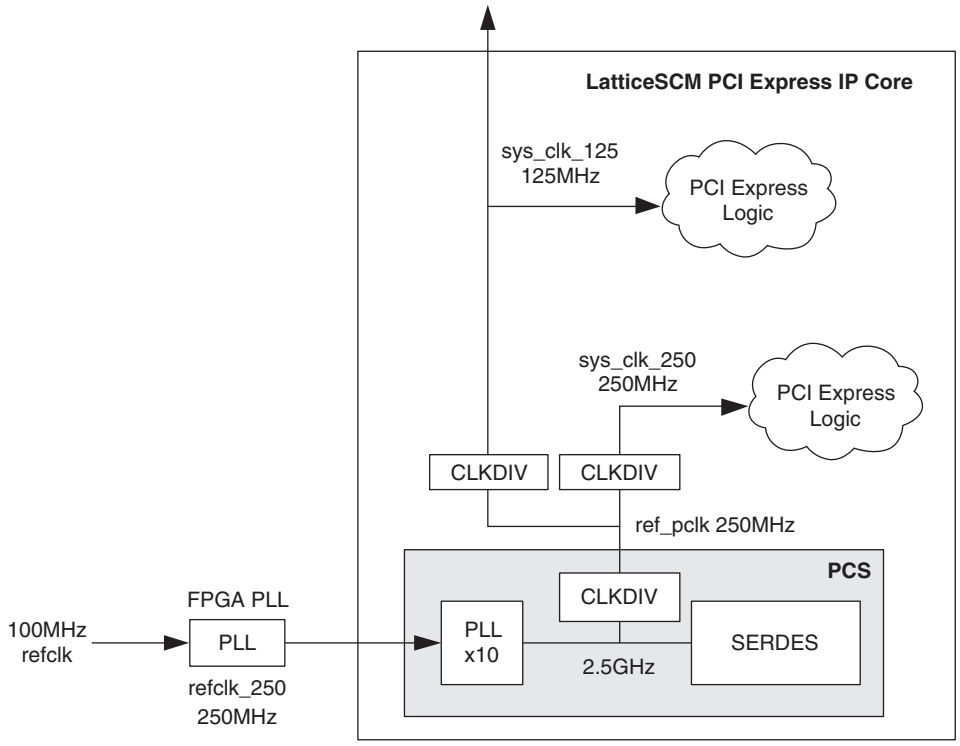
Table 5-1. LatticeECP3 and LatticeECP2M Clocking Resources Used

Type	Number	Notes
Primary Clocks	3/6	3 for x1 interface, and 6 for x4 interface.

LatticeSCM Clocking

Figure 5-5 provides the internal clocking structure of the IP core in the LatticeSCM family.

Figure 5-5. LatticeSCM PCI Express Clocking Scheme



The LatticeSCM clocking solution uses the 100 MHz differential refclk provided from the PCI Express link. This clock must be connected to an LVDS preferred pin for an FPGA PLL. This PLL is required to drive a primary clock network to the PCS/SERDES. The 100 Ω differential termination is included in the LVDS input buffer from the FPGA. The following preference can be used:

```
IOBUF PORT "pcie_clk" IO_TYPE=LVDS DIFFRESISTOR=120;
```

Once inside the device a high bandwidth FPGA PLL should be used to create a 250 MHz from the 100 MHz refclk. A high bandwidth FPGA PLL can be created using the IPexpress tool. Connect the 250 MHz output of the PLL to the refclk_250 port of the PCI Express core. The refclk_250 clock must be routed using primary clock routing to reduce transmit jitter. This can be accomplished by using the following preference.

```
USE PRIMARY NET "refclk_250";
```

Inside the core the refclk_250 port will be routed to the SERDES. Using a x10 PLL the 2.5 Gpbs rate will be created. From the SERDES a 250 MHz transmit clock (ref_pclk) will be created. The clock tolerance compensation block of the PCS is used so that the recovered clocks are not used from the PCS.

The PCI Express core then performs a clock domain change to the sys_clk_125 125 MHz clock for the user interface. The LatticeSCM PCS connection into the FPGA fabric will route ref_pclk to a primary clock network. The CLKDIVs to create the sys_clk_125 and sys_clk_250 nets will also be routed on primary clock networks since they are driven by a CLKDIV. There are no specific site locations for the CLKDIVs since they are sourced and drive primary clocks. They can be left floating for the Place & Route tools to select. This will reduce any possible collisions with other CLKDIVs that require a fixed location.

Table 5-2 shows clocking resources used with LatticeSCM.

Table 5-2. LatticeSCM Clocking Resources Used

Type	Number	Notes
Primary Clocks	2	
CLKDIV	2	Floating.
PLL	1	Floating. Must use preferred input pin.

Locating the IP

The PCI Express core uses a mixture of hard and soft IP blocks to create the full design. This mixture of hard and soft IP requires the user to locate, or place, the core in a defined location on the device array. The hard blocks' fixed locations will drive the location of the IP. Table 5-3 lists the site names for the hard blocks on the different device arrays.

Table 5-3. PCS, flexiMAC and LTSSM Location Site Names for Lattice Devices

Device	PCS	flexiMAC	LTSSM
LFE3-17	PCSA		
LFE3-35	PCSA		
LFE3-70[1]	PCSA PCSB PCSC		
LFE3-95[1]	PCSA PCSB PCSC		
LFE3-150[1]	PCSA PCSB PCSC PCSD		
LFE2M20	URPCS		
LFE2M35	URPCS		
LFE2M50	URPCS LRPCS		
LFE2M70	URPCS ULPCS LRPCS		
LFE2M100	URPCS ULPCS LRPCS LLPCS		
LFSCM15	PCS36000 PCS3E000	LUMACO0	RUMACO0
LFSCM25	PCS36000 PCS36100 PCS3E000 PCS3E100	RUMACO0 LUMACO0	
LFSCM40	PCS36000 PCS36100 PCS3E000 PCS3E100	RUMACO1 LUMACO1	RUMACO0 LUMACO0

Table 5-3. PCS, flexiMAC and LTSSM Location Site Names for Lattice Devices (Continued)

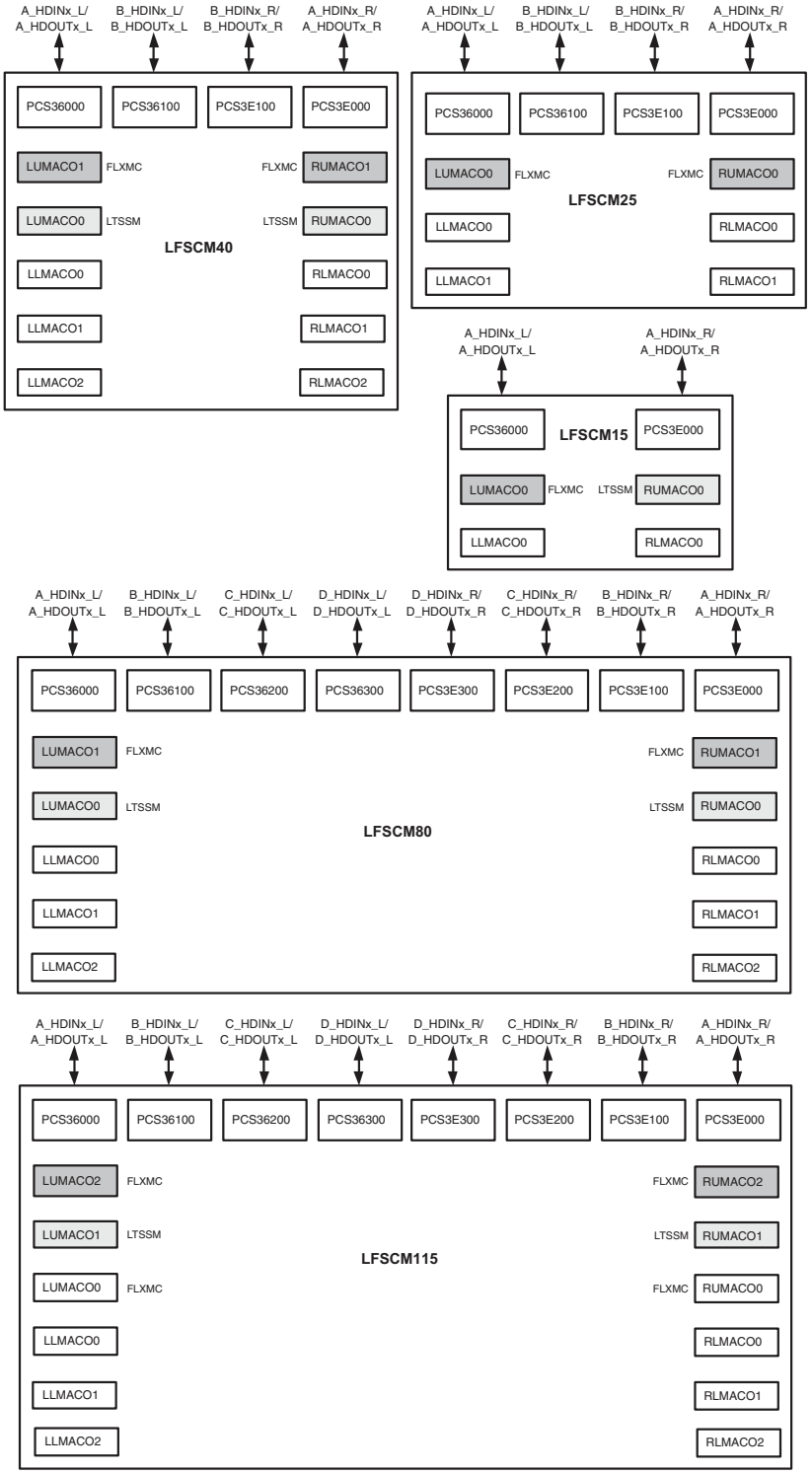
Device	PCS	flexiMAC	LTSSM
LFSCM80	PCS36000 PCS36100 PCS36200 PCS36300 PCS3E000 PCS3E100 PCS3E200 PCS3E300	RUMACO1 LUMACO1	RUMACO0 LUMACO0
LFSCM115	PCS36000 PCS36100 PCS36200 PCS36300 PCS3E000 PCS3E100 PCS3E200 PCS3E300	RUMACO2 LUMACO2 RUMACO0 LUMACO0	RUMACO1 LUMACO1

1. The number of SERDES quads on these devices depends on the package. Lower pinout devices contain fewer quads.

Locating the LatticeSCM Hard Elements

Figure 5-6 provides a diagram of the LatticeSCM15, 25, 40, 80, and 115 devices with site names for the MACO and PCS/SERDES blocks. The MACO cores shaded in gray represent valid flexiMAC and LTSSM locations. Valid PCS/SERDES locations are dependent on the package selection since all PCS/SERDES quads are not bonded out to package pins in all packages.

Figure 5-6. LatticeSCM Device Arrays with PCS/SERDES and MACO Sites



The user should select the PCS/SERDES quad location based on the package pinout and the location of the PCI Express interface on the board layout. Once a PCS/SERDES quad has been located the closest LTSSM location to the selected PCS/SERDES location should be used. Naturally, the location of the flexiMAC will follow the selection of the LTSSM. In order to locate the PCS/SERDES, LTSSM and flexiMAC, the user must use a LOCATE preference in the .lpf file of the Diamond or ispLEVER software. In some instances, the flexiMAC and PCS/SERDES block is deep inside the IP core so the complete IP hierarchy must be included in the component description.

Below is an example of locating the hard blocks.

```
LOCATE COMP “<instance name>/u1_flxmc_sys_pcie/u1_pci_exp_pcs/pcsa_inst”
SITE “PCS36000”;

LOCATE COMP “<instance
name>/u1_flxmc_sys_pcie/u1_flxmc_pcie_core/u1_phy_dll_0/u1_flxmc_
top_ebr/flxmc_top_mib”
SITE “LUMACO0”;

LOCATE COMP "<instance
name>/u1_flxmc_sys_pcie/u1_flxmc_pcie_core/u1_phy_dll_0/LTSSM"
SITE "RUMACO0";
```

If the MACO LTSSM is not used the hierarchy of the design changes slightly. Here are the locate preferences when using the LUT based LTSSM.

```
LOCATE COMP “<instance name>/u1_flxmc_sys_pcie/u1_pci_exp_pcs/pcsa_inst”
SITE “PCS36000”;

LOCATE COMP “<instance
name>/u1_flxmc_sys_pcie/u1_flxmc_pcie_core/u1_phy_dll/u1_flxmc_
top_ebr/flxmc_top_mib”
SITE “LUMACO0”;
```

Locating the LatticeECP3 Hard Elements

Figure 5-7 provides a block diagram with placement positions of the PCS/SERDES quads in the LatticeECP3 devices.

Figure 5-7. LatticeECP3 Device Arrays with PCS/SERDES

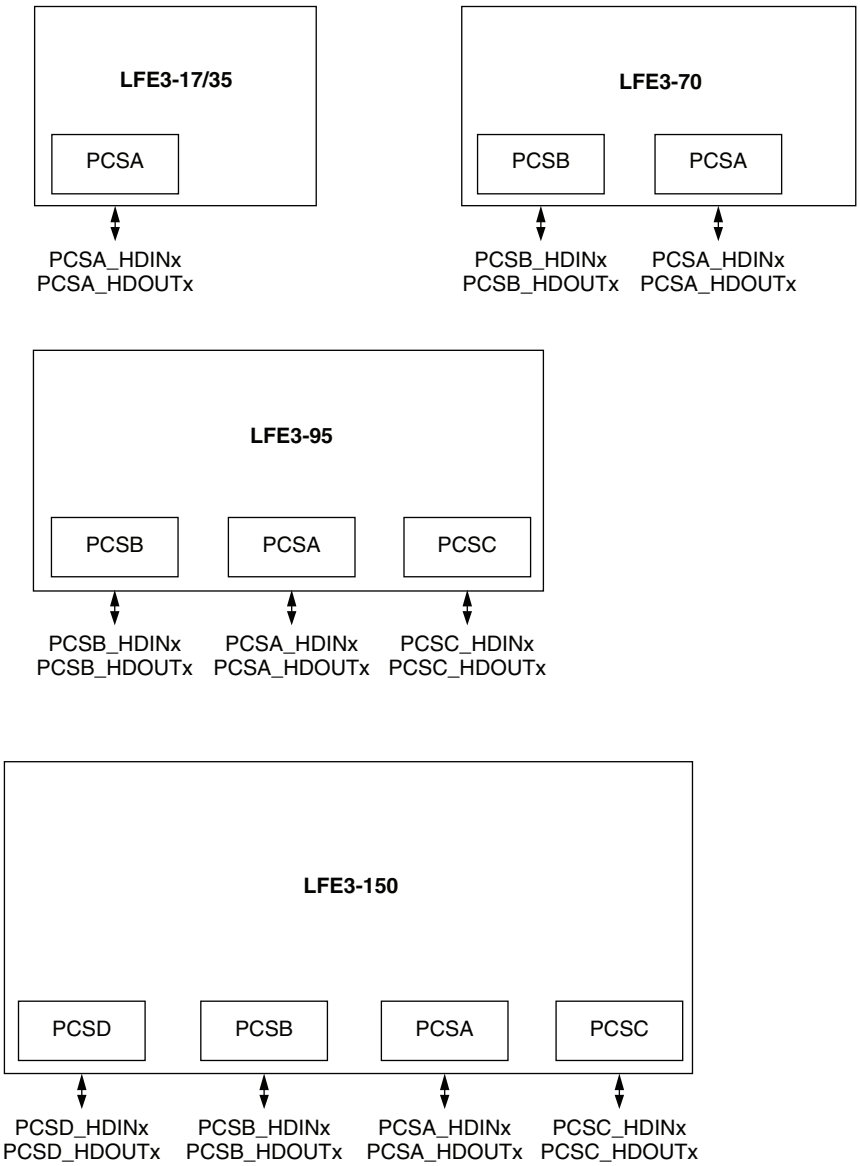
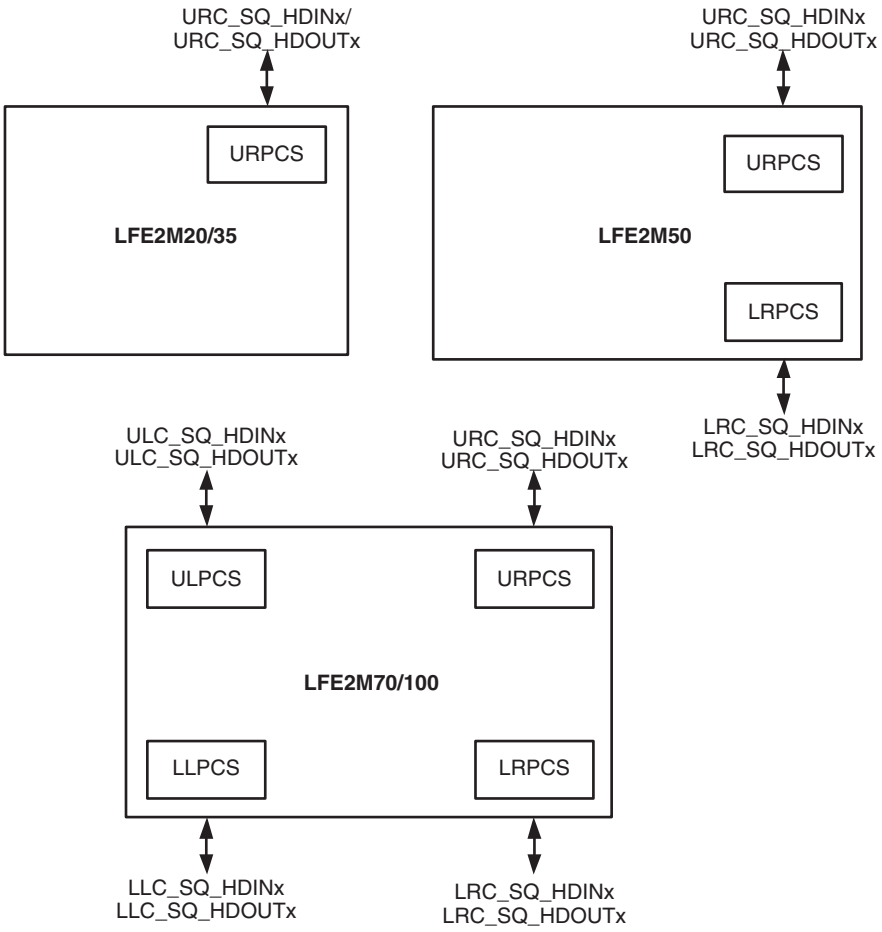


Figure 5-8 provides a block diagram with placement positions of the PCS/SERDES quads in the LatticeECP2M devices.

Figure 5-8. LatticeECP2M Device Arrays with PCS/SERDES



The user should select the PCS/SERDES quad location based on the package pinout and the location of the PCI Express interface on the board layout. Below is an example of locating the PCS in a LatticeECP2M device.

```
LOCATE COMP "<instance name>/u1_pcs_pipe/pcs_top_0/pcs_inst_0" SITE "URPCS" ;
```

Board-Level Implementation Information

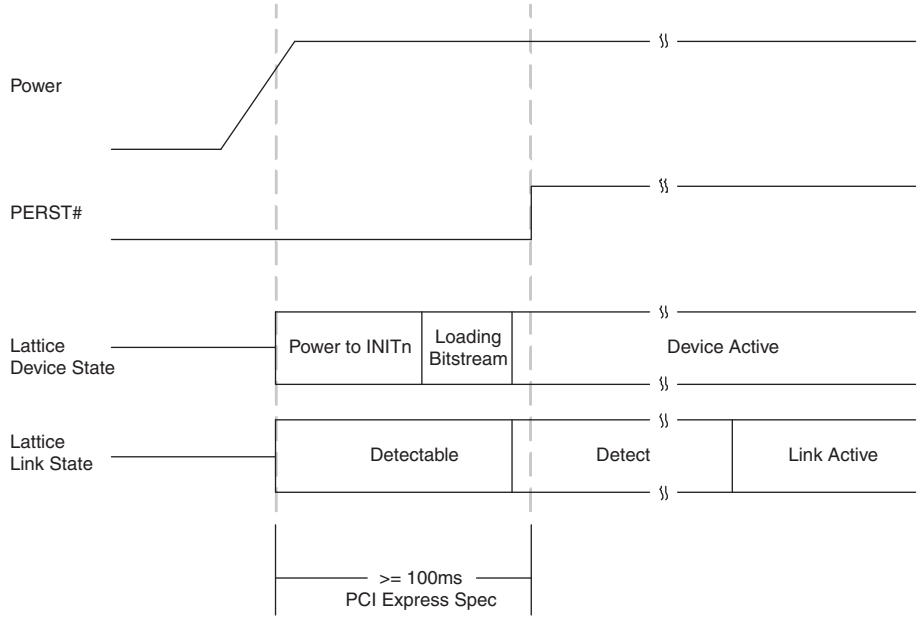
This section provides circuit board-level requirements and constraints associated with using the PCI Express IP core.

PCI Express Power-Up

The PCI Express specification provides aggressive requirements for Power Up. As with all FPGA devices Power Up is a concern when working with tight specifications. The PCI Express specification provides the specification for the release of the fundamental reset (PERST#) in the connector specification. The PERST# release time (TPVPERL) of 100ms is used for the PCI Express Card Electromechanical Specification for Add-in Cards.

From the point of power stable to at least 100 ms the PERST# must remain asserted. Different PCI Express systems will hold PERST# longer than 100 ms, but the minimum time is 100 ms. Shown below in Figure 5-9 is a best case timing diagram of the Lattice device with respect to PERST#.

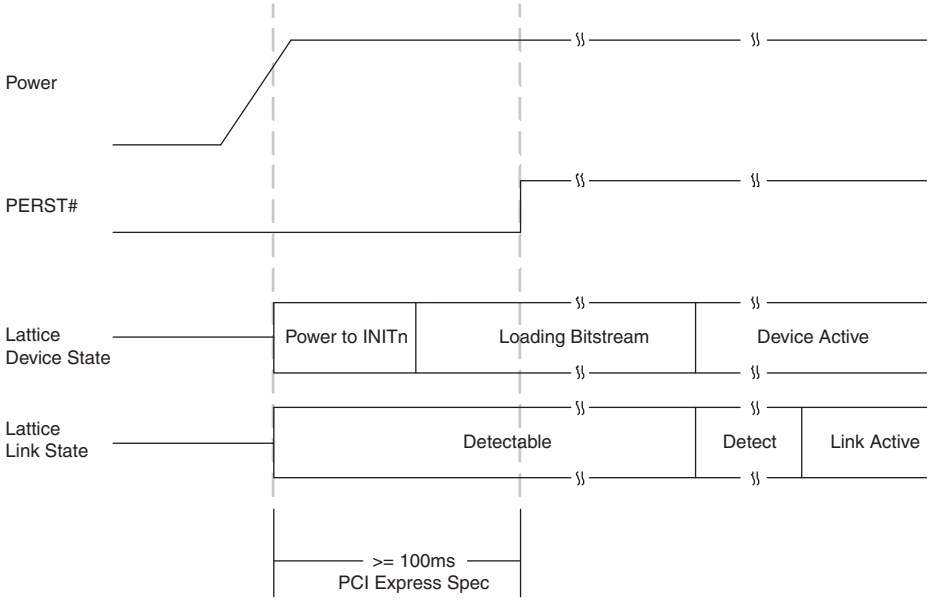
Figure 5-9. Best Case Timing Diagram, Lattice Device with Respect to PERST#



If the Lattice device has finished loading the bitstream prior to the PERST# release, then the PCI Express link will proceed through the remainder of the LTSSM as normal.

In some Lattice devices the device will not finish loading the bitstream until after the PERST# has been released. Figure 5-10 shows a worst case timing diagram of the Lattice device with respect to PERST#.

Figure 5-10. Worst Case Timing Diagram, Lattice Device with Respect to PERST#



If the Lattice device does not finish loading the bitstream until after the release of PERST#, then the link will still be established. The Lattice device turns on the 100 Ω differential resistor on the receiver data lines when power is applied. This 100 Ω differential resistance will allow the device to be detected by the link partner. This state is show

above as “Detectable”. If the device is detected the link partner will proceed to the Polling state of the LTSSM. When the Lattice device goes through Detect and then enters the Polling state the link partner and Lattice device will now cycle through the remainder of the LTSSM.

In order to implement a power-up strategy using Lattice devices, Table 5-4, Table 5-5, and Table 5-6 contain the relative numbers for the LatticeECP2M, LatticeECP3, and LatticeSCM families.

Table 5-4. LatticeECP2M Power Up Timing Specifications

Specification	ECP2M20	ECP2M35	ECP2M50	ECP2M70	ECP2M100	Units
Power to INITn	28	28	28	28	28	ms
Worst-case Programming Time (SPI at 41 MHz)	146	244	390	488	634	ms
Worst-case Programming Time (Parallel Flash with CPLD) [1]	18	31	49	61	79	ms

1. 8-bit wide Flash and external CPLD interfacing to LatticeECP2M at 41 MHz SLAVE_PARALLEL mode.

Table 5-5. LatticeECP3 Power Up Timing Specifications

Specification	ECP3-17	ECP3-35	ECP3-70	ECP3-95	ECP3-150	Units
Power to INITn	23	23	23	23	23	ms
Worst-case Programming Time (SPI at 33 MHz)	136	249	682	682	1082	ms
Worst-case Programming Time (Parallel Flash with CPLD) [1]	17	31	85	85	135	ms

1. 8-bit wide Flash and external CPLD interfacing to LatticeECP3 at 33 MHz SLAVE_PARALLEL mode.

Table 5-6. LatticeSCM Power Up Timing Specifications

Specification	LFSC15	LFSC25	LFSC40	LFSC80	LFSC115	Units
Power to INITn	125	125	125	125	125	ms
Worst-case Programming Time (SPI at 50 MHz)	90.6	156.7	236.4	450.5	714.2	ms
Worst-case Programming Time (Parallel Flash with CPLD)[1]	3.7	6.5	9.8	18.6	29.5	ms

1. 8-bit wide Flash and external CPLD interfacing to LatticeSCM at 150MHz SLAVE_PARALLEL mode.

These warnings inform the user that a SLICE is programmed in DPRAM mode which allows a constant write to the RAM. This is an expected implementation of the RAM which is used in the PCI Express design.

To reduce the bitstream loading time of the Lattice device a parallel Flash device and CPLD device can be used. The use of parallel Flash devices and Lattice devices is documented in AN8077, Parallel Flash Programming and FPGA Configuration.

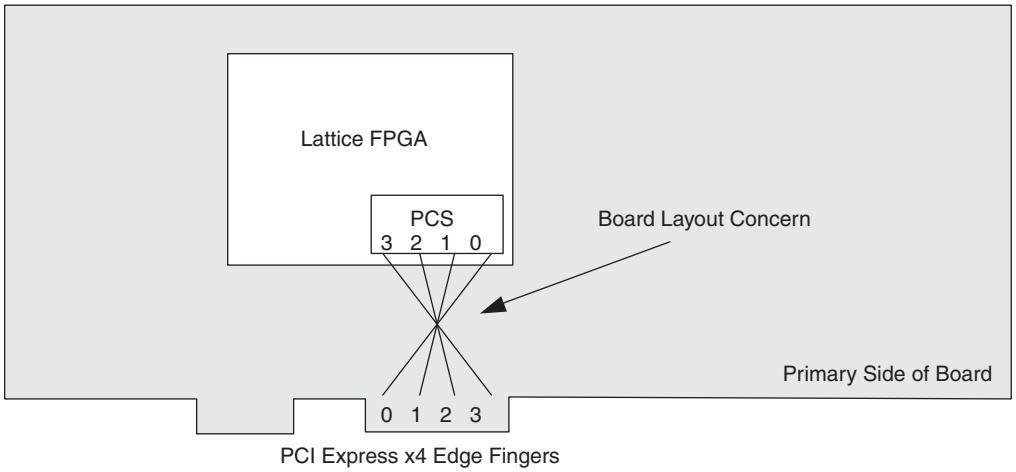
During initialization the PROGRAM and GSR inputs to the FPGA can be used to hold off bitstream programming. These should not be connected to PERST# as this will delay the bitstream programming of the Lattice device.

Board Layout Concerns for Add-in Cards

The PCI Express Add-in card connector edge finger is physically designed for a particular orientation of lanes. The LatticeSCM device package pinout also has a defined orientation of pins for the SERDES channels. The board layout will connect the PCI Express edge fingers to the LatticeSCM SERDES channels. For multi-lane implementations there might be a layout concern in making this connection. On some packages lane 0 of the edge fingers will align with lane 0 of the SERDES and likewise for channels 1, 2 and 3. However, in other packages lane 0 of the edge fingers will need to cross lanes 1, 2 and 3 to connect to lane 0 of the SERDES. It will not be possible to follow

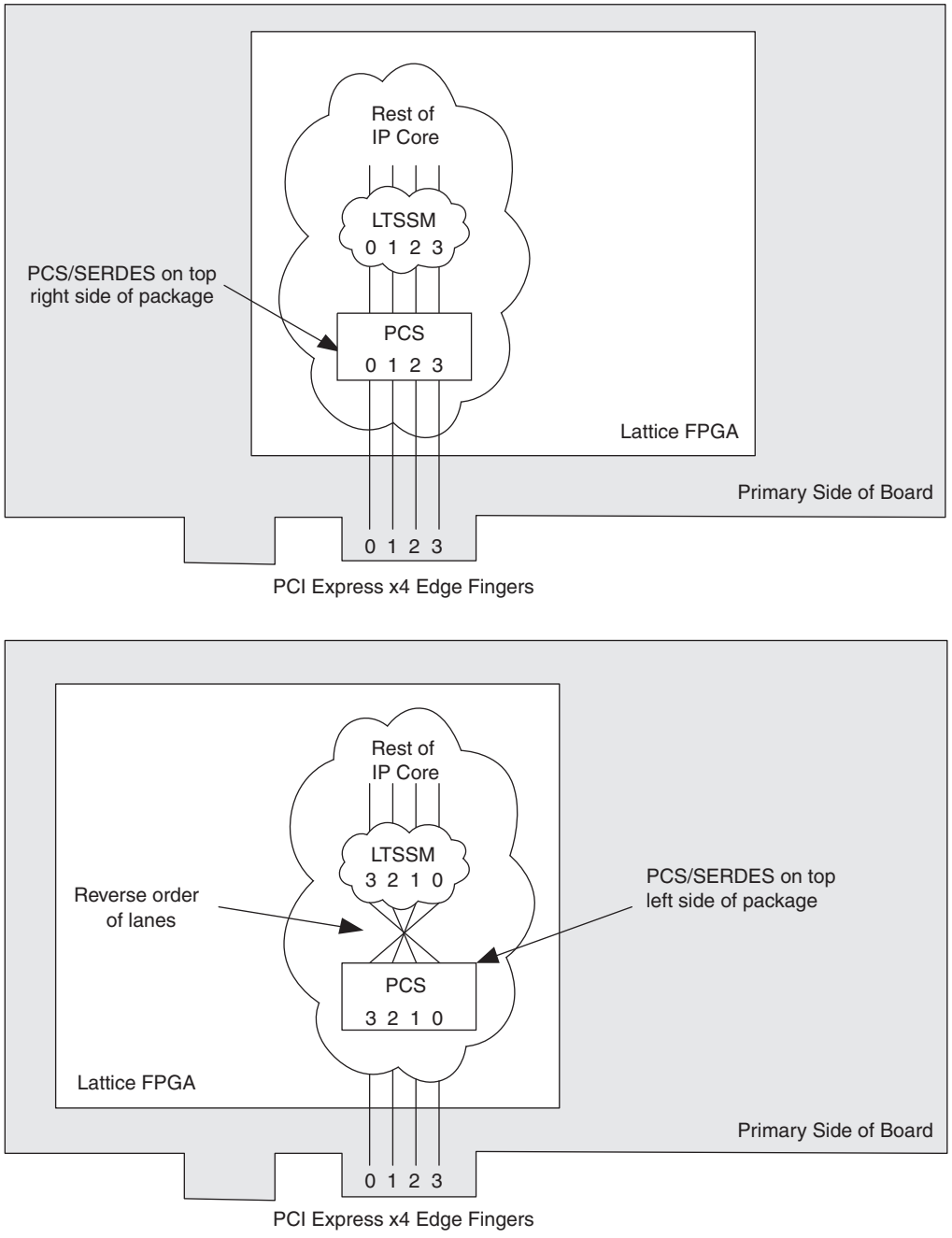
best practice layout rules and cross SERDES lanes in the physical board design. Figure 5-11 provides an example of the board layout concern.

Figure 5-11. Example of Board Layout Concern with x4 Link



To accommodate this layout dilemma, the Lattice PCI Express solution provides an option to reverse the order of the SERDES lanes to the LTSSM block of the PCI Express core. This allows the board layout to connect edge finger lane 0 to SERDES lane 3, edge finger lane 1 to SERDES lane 2, edge finger lane 2 to SERDES lane 1, and edge finger lane 3 to SERDES lane 0. The PCI Express core will then perform a reverse order connection so the PCI Express edge finger lane 0 always connects to the logical LTSSM lane 0. This lane connection feature is controlled using the flip_lanes port. When high, this port will connect the SERDES channels to the PCI Express core in the reversed orientation. The user must be aware when routing the high speed serial lines that this change has taken place. PCI Express lane 0 will need to connect to SERDES channel 3, etc. Figure 5-12 provides a diagram of a normal and a reversed IP core implementation.

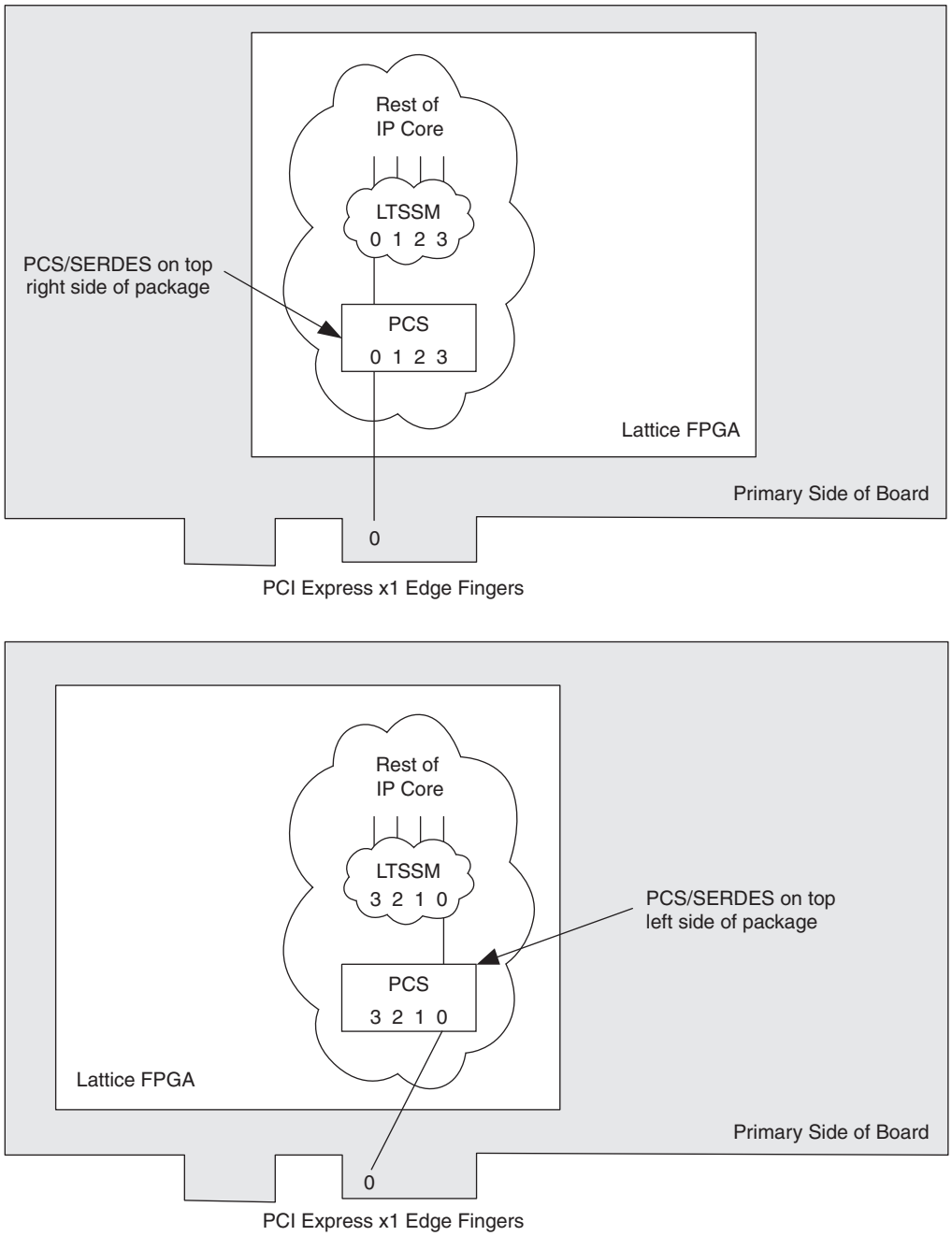
Figure 5-12. Implementation of x4 IP Core to Edge Fingers



As shown in Figure 5-12, this board layout condition will exist on SERDES that are located on the top left side of the package. When using a SERDES quad located on the top left side of the package the user should reverse the order of the lanes inside the IP core.

Figure 5-13 provides a diagram of a x1 IP core to illustrate the recommended solution in the board layout.

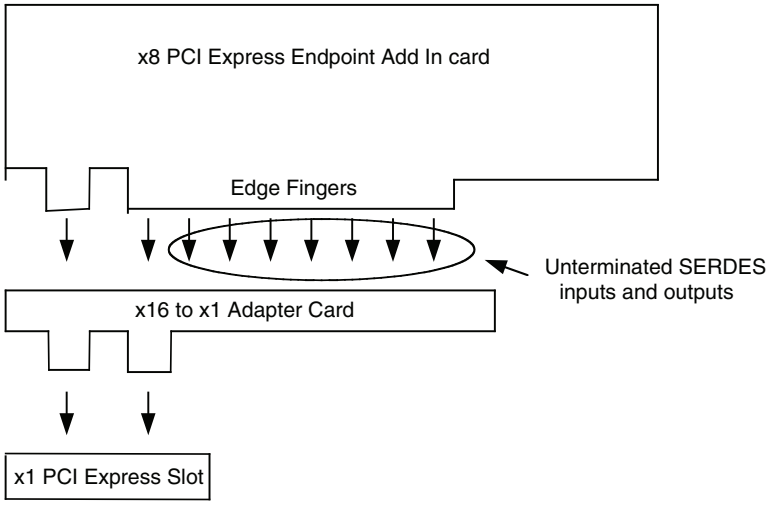
Figure 5-13. Implementation of x1 IP Core to Edge Fingers



Adapter Card Concerns

A PCI Express adapter card allows a multi-lane PCI Express endpoint to be plugged into a PCI Express slot that supports less lanes. For example, a x16 endpoint add in card could use an adapter card to plug into a x1 slot. Adapter cards simply plug onto the edge fingers and only supply connections to those on the edge fingers of the adapter card. Figure 5-14 provides the stack up of an endpoint add in card with an adapter card.

Figure 5-14. PCI Express Endpoint Add In Card



An adapter card simply connects edge fingers to edge fingers. Any of the lanes that are not used by the adapter card are sitting in the adapter card slot. They are unterminated. In Lattice devices, all SERDES channels that are powered up need to be terminated. When using an adapter card the unused channels must be powered down. This can be accomplished by simply editing the autoconfig file for the PCS and not powering up the unused channels. This will provide a bitstream that is suitable for adapter cards.

LatticeECP3 and LatticeECP2M PIPE Simulation

The LatticeECP3 and LatticeECP2M PCI Express simulation also requires the PIPE module. This simulation model is found in the **<*username*>_eval/models/pcs_pipe_top.v**. The same directory contains few other files required for pcs_pipe_top module. The pipe module implements wrapper logic around lattice SERDE/PCS to make it a PIPE compliant interface and will be connected to the PIPE compliant interface of PCI express IP. It also implements a reset sequencing logic for SERDES/PCS usage as suggested by tech notes TN1176, LatticeECP3 SERDES/PCS Usage Guide and TN1124, LatticeECP2/M SERDES/PCS Usage Guide. It also generates reset signal(pcie_ip_rstn) to connect to PCI express IP. Refer to file <username>_top.v" for detailed connections of PIPE interface between PCI express IP and PCS PIPE module.

Below is a sample Aldec.do file (which can also be used with ModelSim) to compile and simulate the IP core.

LatticeSCM

```
# Compile the PCIe IP core
vlog +define+SIMULATE=1 pci_exp_params.v pci_exp_ddefines.v pcie_beh.v pcie.v

# Compile the user design
vlog top.v

# Compile the testbench
vlog tb.v

# Load the design and libraries
vsim -L sc_vlg -L pcsa_mti_work -L pmi_work work.tb
```

LatticeECP3 and LatticeECP2M

```
# Compile the PCIe IP core
vlog +define+SIMULATE=1 pci_exp_params.v pci_exp_ddefines.v pcie_beh.v pcie.v

# Compile the PIPE
vlog +define+SIMULATE=1 pci_exp_params.v \
                        pcie_eval/models/[ecp3/ecp2m]/pipe_top.v \
                        pcie_eval/models/[ecp3/ecp2m]/pcs_top.v \
                        pcie_eval/models/[ecp3/ecp2m]/ctc.v \
                        pcie_eval/models/[ecp3/ecp2m]/sync1s.v \
                        pcie_eval/models/[ecp3/ecp2m]/tx_reset_sm.v \
                        pcie_eval/models/[ecp3/ecp2m]/rx_reset_sm.v \
                        pcie_eval/models/[ecp3/ecp2m]/PCSC.v or PCSD.v \
                        pcie_eval/models/[ecp3/ecp2m]/pcs_pipe_top.v
# Compile the user design
vlog top.v


# Compile the testbench
vlog tb.v

# Load the design and libraries for ECP2M
vsim -L ecp2m_vlg -L pcsc_mti_work -L pmi_work work.tb


# Load the design and libraries for ECP3
vsim -L ecp3m_vlg -L pcsd_mti_work -L pmi_work work.tb
```

Simulation Behavior

When setting the SIMULATE variable for the simulation model of the PCI Express core several of the LTSSM counters are reduced. Table 5-7 provides the new values for each of the LTSSM counters when the SIMULATE variable is defined.

Table 5-7. LTSSM Counters

Counter	Normal Value	SIMULATE Value	Description
CNT_1MS	1 ms	800 ns	Electrical Order set received to Electrical Idle condition detected by Loopback Slave
CNT_1024T1	1024 TS1	48 TS1	Number of TS1s transmitted in Polling.Active
CNT_2MS	2 ms	1200 ns	Configuration.Idle (CFG_IDLE)
CNT_12MS	12 ms	800 ns	Detect.Quiet (DET_QUIET)
CNT_24MS	24 ms	1600 ns	Polling.Active (POL_ACTIVE), Configuration.Linkwidth.Start (CFG_LINK_WIDTH_ST), Recovery.RcvrLock (RCVRY_RCVRLK)
CNT_48MS	48 ms	3200 ns	Polling.Configuration (POL_CONFIG), Recovery.RcvrCfg (RCVRY_RCVRCFG)
CNT_50MS	50 ms	20 us	Completion time out
CNT_100MS	100 ms	4000 ns	LoopBack Master time out

Troubleshooting

Table 5-8 provides some troubleshooting tips for the user when the core does not work as expected.

Table 5-8. Troubleshooting

Symptom	Possible Reason	Troubleshooting
LTSSM does not transition to L0 state	LatticeSCM: The uML module was not used to control the PCS programming.	See the Resource Utilization section and the uML design which controls the PCS programming for a multi lane link in the LatticeSCM.
	The PCI Express slot does not support the advertised link width.	Some PC systems do not support all possible link width configurations in their x16 slots. Try using the slot as a x1 and working up to the x4 link width.
Board is not recognized by the PC	Driver not installed or did not bind to the board.	Check to make sure the driver is installed and the DeviceID and VendorID match the drivers IDs defined in the .inf file.
Software application locks up	Endpoint is stalled and cannot send TLPs.	Check to make sure the amount of credits requested is correct. If the endpoint cannot complete a transaction started by the application software, the software will “hang” waiting on the endpoint.
PC crashes	Endpoint might have violated the available credits of the root complex.	A system crash usually implies a hardware failure. If the endpoint violates the number of credits available, the root complex can throw an exception which can crash the machine.
	Endpoint might have created a NAK or forced a retrain.	Certain motherboards are forgiving of a NAK or LTSSM retrain while others are not. A retrain can be identified by monitoring the phy_ltssm_state vector from the PCI Express core to see if the link falls from the L0 state during operation.
Endpoint stops working after hours of operation	The hardware timer is included in the device.	The hardware timer is utilized when an IP core does not have a license. The hardware timer holds the device in reset after a few hours of operation. Powering the endpoint off and then on will restore functionality for the period of time dictated by the hardware timer.

Core Verification

The functionality of the Lattice PCI Express Endpoint IP core has been verified via simulation and hardware testing in a variety of environments, including:

- Simulation environment verifying proper PCI Express endpoint functionality when testing with a Synopsys DesignWare behavioral model in root complex mode.
- PCI-SIG certification via hardware validation of the IP implemented on Lattice FPGA evaluation boards. Specific testing has included:
  - Verifying proper protocol functionality (transaction layer, data link layer and DUT response to certain error conditions) when testing with the Agilent E2969A Protocol Test Card (PTC) and PTC test suite.
  - Note: PTC was used for both in-house testing and testing at PCI-SIG workshops.
  - Verifying proper protocol functionality with the PCI-SIG configuration test suite.
  - Verifying electrical compliance.
    Note: Electrical compliance testing has been verified at PCI-SIG and also in-house by the Lattice PDE group.
- Interop testing with multiple machines at PCI-SIG workshops and in-house.
- Using the Agilent E2960A PCI Express Protocol Tester and Analyzer for analyzing and debugging PCI Express bus protocol. The Tester is used for sending and responding to PCI Express traffic from the DUT.

Core Compliance

A high-level description of the PCI-SIG Compliance Workshop Program and summary of the compliance test results for our PCI Express Endpoint IP core is provided in TN1166, PCI Express SIG Compliance Overview for Lattice Semiconductor FPGAs (August 2007). As described in TN1166, the Lattice PCI Express IP core successfully passed PCI-SIG electrical, Configuration Verifier (CV) and link and transaction layer protocol testing. The PCI Express IP core also passed the 80% interoperability testing program specified by PCI-SIG. In accordance with successfully completing PCI-SIG compliance and interoperability testing, the Lattice PCI Express Endpoint Controller IP cores are currently included on the PCI-SIG Integrators List.

Support Resources

This chapter contains information about Lattice Technical Support, additional references, and document revision history.

Lattice Technical Support

There are a number of ways to receive technical support.

E-mail Support

techsupport@latticesemi.com

Local Support

Contact your nearest Lattice sales office.

Internet

www.latticesemi.com

PCIe Solutions Web Site

Lattice provides customers with low-cost and low-power programmable PCIe solutions that are ready to use right out of the box. A full suite of tested and interoperable PCIe solutions is available that includes development kits with evaluation boards and hardware and software reference designs. These solutions are valuable resources to jump start PCIe applications from a board design and FPGA design perspective. For more information on Lattice's PCIe solutions, visit:

http://www.latticesemi.com/solutions/technologysolutions/pciexpresssolutions.cfm?source=topnav

PCI-SIG Website

The Peripheral Component Interconnect Special Interest Group (PCI-SIG) website contains specifications and documents referred to in this user's guide. The PCi-SIG URL is:

http://www.pcisig.com.

References

LatticeECP3

- DS1021, LatticeECP3 Data Sheet
- TN1176, LatticeECP3 SERDES/PCS Usage Guide

LatticeECP2M

- DS1006, LatticeECP2M Family Data Sheet
- TN1114, Electrical Recommendations for Lattice SERDES
- TN1165, PCI Express and SGMII/GbE Applications in the Same LatticeECP2M SERDES Quad
- TN1166, PCI Express SIG Compliance Overview for Lattice Semiconductor FPGAs
- TN1124, LatticeECP2/M SERDES/PCS Usage Guide
- AN8077, Parallel Flash Programming and FPGA Configuration

LatticeSCM

- DS1004, LatticeSC/M Family Data Sheet
- DS1005, LatticeSC/M Family flexiPCS Data Sheet
- TN1085, LatticeSC MPI/System Bus
- TN1098, LatticeSC sysCLOCK PLL/DLL User's Guide
- TN1114, Electrical Recommendations for LatticeSC SERDES
- TN1166, PCI Express SIG Compliance Overview for Lattice Semiconductor FPGAs
- AN8077, Parallel Flash Programming and FPGA Configuration

Revision History

Date	Document Version	IP Core Version	Change Summary
October 2014	2.2	5.3	Added missing LTSSM sub states and info about CLKDIV component.
			Updated inta port description. description for Link Cap reg field from GUI.
December 2013	02.1	5.3	Supports reset sequence logic for ECP3 devices in pcs_pipe_top module.
			Added BAR setting info and example.
			Added reference to tech notes for SERDES/PCS usage.
			Corrected MSI and Wishbone descriptions.
			Updated Technical Support Assistance information.
August 2012	02.0	5.2	Added Wishbone Byte/Bit Ordering information to Wishbone Interface text section.
February 2012	01.9	5.2	Updated document with new corporate logo.
			PCI Express IP Core Port List – Updated description for tx_val.
			PCI Express IP Core Port List – Updated description for phy_cfgln_sum[2:0].
December 2010	01.8	5.0	Updated for PCI Express 2.0.
			Updated for Lattice Diamond 1.1 and ispLEVER 8.1 SP1 design software.
			Added support for dev_cntl_2_out port and Device Capabilities 2 Register [4:0] throughout document.
			PCI Express IP Core Port List – Added flr_rdy_in, sys_clk_125 and dev_cntl_2_out ports.
			Updated screen shots: — PCI Express IP Core General Options in the IPexpress Tool — PCI Express IP Core Configuration Space Options in the IPexpress Tool — IPexpress Configuration GUI (Diamond Version)
			Updated PCI Express Capability Version text section.
September 2010	01.7	4.3	Updated Table 2-4. Wishbone Interface Memory Map
			Updated the following table and figures in Chapter 3: — Table 3-1, IP Core Parameters — Figure 3-1, PCI Express IP Core General Options in the IPexpress Tool — Figure 3-4, PCI Express IP Core Physical Layer Options in the IPexpress Tool — Figure 3-5, PCI Express IP Core Data Link Layer Options in the IPexpress Tool — Figure 3-6, PCI Express IP Core Transaction Layer Options in the IPexpress Tool — Figure 3-7, PCI Express IP Core Configuration Space Options in the IPexpress Tool Added new parameter descriptions.

Date	Document Version	IP Core Version	Change Summary
August 2010	01.6	4.3	Changed document title to “PCI Express 1.1 x1, x4 Endpoint IP Core User’s Guide.”
			Updated the following figures: — Figure 2-1, PCI Express IP Core Technology and Functions — Figure 2-4, Transmit Interface x4, 3DW Header, 1 DW Data — Figure 2-5, Transmit Interface x4, 3DW Header, 2 DW Data — Figure 2-6, Transmit Interface x4, 4DW Header, 0 DW — Figure 2-7, Transmit Interface x4, 4DW Header, Odd Number of DWs — Figure 2-8, Transmit Interface x4, Burst of Two TLPs — Figure 2-10, Transmit Interface x1 Downgrade Using tx_val — Figure 2-13, Transmit Interface Native x1, Burst of Two TLPs — Figure 2-14, Transmit Interface Native x1, Nullified TLP
			Updated the following tables: Table 2-2, Unsupported TLPs Which Can be Received by the IP Table 2-7, Transaction Layer Error List Table 5-1, LatticeECP3 and LatticeECP2M Clocking Resources Used
			Updated Appendix A, Resource Utilization.
July 2010	01.5	4.2	Updated Table 2-4, Wishbone Interface Memory Map.
July 2010	01.4	4.2	Added support for Diamond software throughout.
February 2010	01.3	4.1	Divided the document into chapters and added a table of contents.
			Updated Table 1-1, PCI Express IP Core Quick Facts in Chapter 1.
			Updated Chapter 2, as follows: — Updated Figure 2-13, Transmit Interface Native x1, Burst of Two TLPs. — Updated Figure 2-19, Receive Interface, Unsupported Request TLP — Added Table 2-5, Physical Layer Error List — Table 2-6, Data Link Layer Error List — Table 2-7, Transaction Layer Error List
			Updated Chapter 3 as follows: — Added Table 3-1, IP Core Parameters — Added new screen captures and new parameter descriptions.
			Updated Chapter 4, as follows: — Added Licensing the IP Core section. — Added Getting Started. — Updated IPexpress-Created Files and Top Level Directory Structure section.
			Added Chapter 6, Core Verification.
			Combined several appendices into one appendix, deleted LatticeECP2M-25 Utilization.
October 2009	01.2	4.1	Added footnote for “minimal devices” in Table 1-1, PCI Express IP Core Quick Facts.
			Updated rst_n pin description.
			Updated tx_rdy_vc0 pin description.
			Clarified tx_rdy_vc0/tx_st_vc0 timing relationship in Transmit TLP Interface description.
			Clarified credit releasing requirements.
July 2009	01.1	4.0	Added support for LatticeECP3 FPGA family.
August 2008	01.0	3.3	Initial release.

Resource Utilization

PCI Express 2.0 x1, x4 IP CoreThis appendix gives resource utilization information for Lattice FPGAs using the PCI Express IP core.

The IPexpress tool is the Lattice IP configuration utility, and is included as a standard feature of the Diamond and ispLEVER design tools. Details regarding the usage of the IPexpress tool can be found in the IPexpress tool and Diamond or ispLEVER help system. For more information on the Diamond or ispLEVER design tools, visit the Lattice web site at: www.latticesemi.com/software.

LatticeSCM-15/40/80/115 Utilization

Table A-1 shows the resource utilization for the PCI Express x1 Endpoint core implemented in a LatticeSCM 15/40/80/115 FPGA. Table A-2 lists the parameter settings for the IP core configuration shown in Table A-1.

Table A-1. Resource Utilization[1]

IPexpress Configuration[1]	Slices	LUTs	Registers	sysMEM EBRs	MACO Blocks
Config 1 - Soft LTSSM	5761	8174	6134	11	1
Config 1 - Hard LTSSM	3892	4922	4719	11	2

1.Performance and utilization data are generated targeting an LFSC3GA80E-6FC1704C using Lattice Diamond 1.0 and Synplify Pro D-2009.12L-1 software. Performance might vary when using a different software version or targeting a different device density or speed grade within the LatticeSCM family. Hard LTSSM MACO is available for LFSC3GA15/40/80/115 devices, and soft LTSSM is not required. Utilization and performance results for PCI Express x1 and x4 mode are identical in LatticeSCM devices. When the x4 core downgrades to x1 mode, utilization and performance results for x1 are identical to x4 mode.

Ordering Part Number

LatticeSCM MACO IP cores, including the LatticeSCM PCI Express IP core, are directly available through the IPexpress capability of the Diamond or ispLEVER software.

Table A-2. Parameter Settings for Config Demo

	Config 1
Retry Buffer Size	512 Bytes
Enable ECRC	No
Enable AER	No
Wishbone Interface	No
Enable BAR 0	Yes
Enable BAR 1	Yes
Enable BAR 2	No
Enable BAR 3	No
Enable BAR 4	No
Enable BAR5	No
Enable Expansion ROM BAR	No

Configuration

All MACO IP is pre-engineered and hard-wired into the MACO structured ASIC blocks of the LatticeSCM family. Each LatticeSCM device contains a different collection of MACO IP. Refer to the Lattice web pages on LatticeSCM and MACO IP for more information.

LatticeSCM-25 Utilization

Table A-3 shows the resource utilization for the PCI Express x1 Endpoint core implemented in a LatticeSCM-25 FPGA. Table A-4 lists the parameter settings for the IP core configuration shown in Table A-3.

Table A-3. Resource Utilization[1]

IPexpress Configuration[1]	Slices	LUTs	Registers	sysMEM EBRs	MACO Blocks
Config 1 - Soft LTSSM	5761	8174	6134	11	1

1. Performance and utilization data are generated targeting an LFSC3GA25E-6FF1020C using Lattice Diamond 1.0 and Synplify Pro D-2009.12L-1 software. Performance might vary when using a different software version or targeting a different device density or speed grade within the LatticeSCM family. Soft LTSSM is required as hard LTSSM MACO is not available for LFSC3GA25 devices Utilization and performance results for PCI Express x1 and x4 mode are identical in LatticeSCM devices. When the x4 core downgrades to x1 mode, utilization and performance results for x1 are identical to x4 mode.

Ordering Part Number

LatticeSCM MACO IP cores, including the LatticeSCM PCI Express IP core, are directly available through the IPexpress capability of the ispLEVER software.

Table A-4. Parameter Settings for Config Demo

	Config 1
Retry Buffer Size	512 Bytes
Enable ECRC	No
Enable AER	No
Wishbone Interface	No
Enable BAR 0	Yes
Enable BAR 1	Yes
Enable BAR 2	No
Enable BAR 3	No
Enable BAR 4	No
Enable BAR5	No
Enable Expansion ROM BAR	No

Configuration

All MACO IP is pre-engineered and hard-wired into the MACO structured ASIC blocks of the LatticeSCM family. Each LatticeSCM device contains a different collection of MACO IP. Refer to the Lattice web pages on LatticeSCM and MACO IP for more information.

LatticeECP2M Utilization (x1 Endpoint)

Table A-5 shows the resource utilization for the PCI Express x1 Endpoint core implemented in a LatticeECP2M FPGA. Table A-6 lists the parameter settings for the IP core configuration shown in Table A-5.

Table A-5. Resource Utilization[1]

IPexpress Configuration	Slices	LUTs	Registers	sysMEM EBRs
Config 1	4334	6370	4031	4

1. Performance and utilization data are generated targeting an LFE2M-50E-6F900C using Lattice Diamond 1.0 and Synplify Pro D-2009.12L-1 software. Performance might vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2M family.

Ordering Part Number

The Ordering Part Number (OPN) for the PCI Express x1 Endpoint IP core targeting LatticeECP2M devices is PCI-EXP1-PM-U3.

Table A-6. Parameter Settings for Config Demo

	Config 1
Retry Buffer Size	512 Bytes
Enable ECRC	No
Enable AER	No
Wishbone Interface	No
Enable BAR 0	Yes
Enable BAR 1	Yes
Enable BAR 2	No
Enable BAR 3	No
Enable BAR 4	No
Enable BAR5	No
Enable Expansion ROM BAR	No

LatticeECP2M Utilization (x4 Endpoint)

Table A-7 shows the resource utilization for the PCI Express x4 Endpoint core implemented in a LatticeECP2M FPGA. Table A-8 lists the parameter settings for the IP core configuration shown in Table A-7.

Table A-7. Resource Utilization[1]

IPexpress Configuration	Slices	LUTs	Registers	sysMEM EBRs
Config 1	9456	12558	9824	11

1. Performance and utilization data are generated targeting an LFE2M-50E-6F900C using Lattice Diamond 1.0 and Synplify Pro D-2009.12L-1 software. Performance might vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2M family. When the x4 core downgrades to x1 mode, utilization and performance results for x1 are identical to x4 mode.

Ordering Part Number

The Ordering Part Number (OPN) for the PCI Express x4 Endpoint IP core targeting LatticeECP2M devices is PCI-EXP4-PM-U3.

Table A-8. Parameter Settings for Config Demo

	Config 1
Retry Buffer Size	512 Bytes
Enable ECRC	No
Enable AER	No
Wishbone Interface	No
Enable BAR 0	Yes
Enable BAR 1	Yes
Enable BAR 2	No
Enable BAR 3	No
Enable BAR 4	No
Enable BAR5	No
Enable Expansion ROM BAR	No

LatticeECP3 Utilization (x1 Endpoint)

Table A-9 shows the resource utilization for the PCI Express x1 Endpoint core implemented in a LatticeECP3 FPGA. Table A-10 lists the parameter settings for the IP core configuration shown in Table A-9.

Table A-9. Resource Utilization[1]

IPexpress Configuration[1]	Slices	LUTs	Registers	sysMEM EBRs
Config 1	4076	6229	4033	4

1. Performance and utilization data are generated targeting an LFE3-95E-7FN1156CES using Lattice Diamond 1.0 and Synplify Pro D-2009.12L-1 software. Performance might vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family.

Ordering Part Number

The Ordering Part Number (OPN) for the PCI Express x1 Endpoint IP core targeting LatticeECP3 devices is PCI-EXP1-E3-U3.

Table A-10. Parameter Settings for Config Demo

	Config 1
Retry Buffer Size	512 Bytes
Enable ECRC	No
Enable AER	No
Wishbone Interface	No
Enable BAR 0	Yes
Enable BAR 1	Yes
Enable BAR 2	No
Enable BAR 3	No
Enable BAR 4	No
Enable BAR 5	No
Enable Expansion ROM BAR	No

LatticeECP3 Utilization (x4 Endpoint)

Table A-11 shows the resource utilization for the PCI Express x4 Endpoint core implemented in a LatticeECP3 FPGA. Table A-12 lists the parameter settings for the IP core configuration shown in Table A-11.

Table A-11. Resource Utilization[1]

IPexpress Configuration[1]	Slices	LUTs	Registers	sysMEM EBRs
Config 1	8937	12355	9758	11

1. Performance and utilization data are generated targeting an LFE3-95E-7FN1156CES using Lattice Diamond 1.0 and Synplify Pro D-2009.12L-1 software. Performance might vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family. When the x4 core downgrades to x1 mode, utilization and performance results for x1 are identical to x4 mode.

Ordering Part Number

The Ordering Part Number (OPN) for the PCI Express x4 Endpoint IP core targeting LatticeECP3 devices is PCI-EXP4-E3-U3.

Table A-12. Parameter Settings for Config Demo

	Config 1
Retry Buffer Size	512 Bytes
Enable ECRC	No
Enable AER	No
Wishbone Interface	No
Enable BAR 0	Yes
Enable BAR 1	Yes
Enable BAR 2	No
Enable BAR 3	No
Enable BAR 4	No
Enable BAR 5	No
Enable Expansion ROM BAR	No

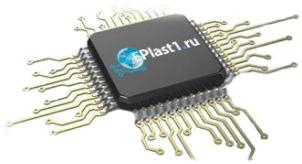
Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits,General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.



Как с нами связаться

Телефон: 8 (812) 309 58 32 (многоканальный)
Факс: 8 (812) 320-02-42
Электронная почта: org@eplast1.ru
Адрес: 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.