



PCI Express 1.1 x1, x4 Core

User's Guide

Introduction

PCI Express is a high performance, scalable, well defined standard for a wide variety of computing and communications platforms. It has been defined to provide software compatibility with existing PCI drivers and operating systems.

Lattice's PCI Express core provides a x1 or x4 endpoint solution from the electrical SERDES interface to the transaction layer. This solution supports both the 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 LatticeECP2M family, 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 both the LatticeECP2M and LatticeSCM families of devices.
- The **x4 Downgraded x1 Core** also targets both the 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 LatticeECP2M family. This is a reduced LUT count x1 core with a 16-bit datapath.

Please also refer to the Lattice website for the following documents, solutions, and IP related to the PCI Express 1.1 x1, x4 core:

- LatticeECP2M PCI Express Development Kit
- PCI Express Endpoint IP Core Demo for LatticeECP2M and LatticeSCM
- LatticeSC PCI Express x4 and x8 Evaluation Boards
- LatticeECP2M PCI Express x4 Evaluation Board
- Scatter-Gather Direct Memory Access (DMA) Controller

PCI Express ispLeverCORE Quick Facts

			PCI Express IP Core			
			x1 Endpoint	x4 Endpoint ⁴	x4 Endpoint, Soft LTSSM ^{1, 3, 4}	x4 Endpoint, Hard LTSSM ^{2, 3, 4}
Core Requirements	FPGAs Family Supported		LatticeECP2M and LatticeSCM		LatticeSCM	
	Minimal Device Needed		LFE2M-20E-6F484C	LFE2M-50E-6F672C	LFSC3GA15E-6F900C	LFSC3GA15E-6F900C
Data Path Width	LatticeECP2M		16	64	–	–
	LatticeSCM				64	64
Performance/ Resource Utilization by Device	LFE2M-20E-6F484C	LUTs	5964	–	–	–
		sysMEM EBRs	4	–	–	–
		Registers	4043	–	–	–
	LFE2M-50E-6F672C	LUTs	–	12194	–	–
		sysMEM EBRs	–	11	–	–
		Registers	–	9568	–	–
	LFSC3GA25E-6FC1020C	LUTs	–	–	8122	–
		sysMEM EBRs	–	–	13	–
		Registers	–	–	6064	–
	LFSC3GA80E-6FC1704C	LUTs	–	–	–	4701
		sysMEM EBRs	–	–	–	13
		Registers	–	–	–	4696
Design Tool Support	Lattice Implementation		ispLEVER 7.1			
	Synthesis		Synplicity Synplify Pro 9.4L			
	Simulation		Aldec Active HDL 7.3 SP1 Lattice Edition			
			Mentor ModelSim 6.3 SE			

1. LTSSM MACO not available in LFSC3GA25; soft LTSSM is required.

2. LTSSM MACO is available in LFSC3GA15/40/80/115 devices.

3. Resource utilization and performance results for x1 and x4 mode are identical in LatticeSCM devices.

4. When the x4 cores downgrades to x1 mode, utilization and performance results for x1 are identical to x4 mode for both LatticeECP2M and LatticeSCM families.

General Features

The Lattice PCI Express IP core supports the following features.

PHY Layer Features

- 2.5Gbps CML electrical interface
- PCI Express 1.1 electrical compliance
- Many options for signal integrity including differential output voltage, transmit pre-emphasis and receiver equalization
- Serialization and de-serialization
- 8b10b symbol encoding/decoding
- Link state machine for symbol alignment
- Clock tolerance compensation supports +/- 300ppm
- 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 retry 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
- 512, 1k, 2k, or 4k byte 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
- Supported in -5, -6, and -7 speed grades
 - LatticeSCM supports both 1.2V and 1.0V core voltage (up to a 50% power reduction)

IP Core Features

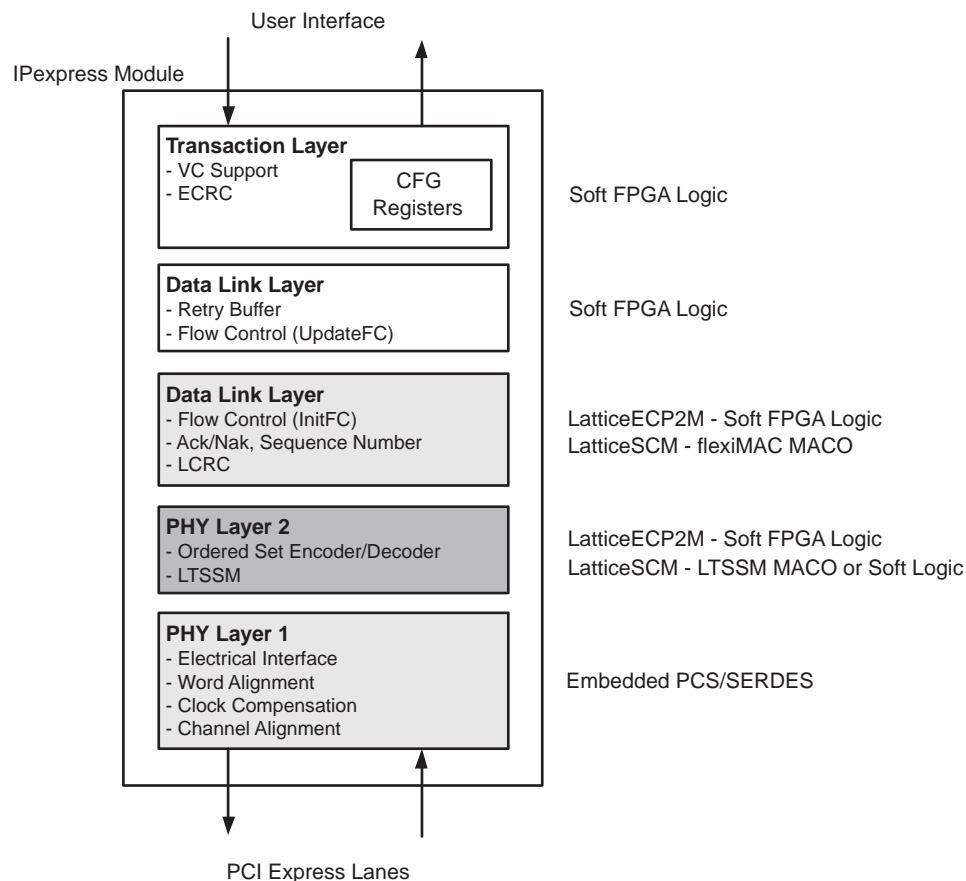
Description

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

The ispLEVER design tool IPexpress is used to customize and create a complete IP module for the user to instantiate in a design. Inside the module created by IPexpress 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.

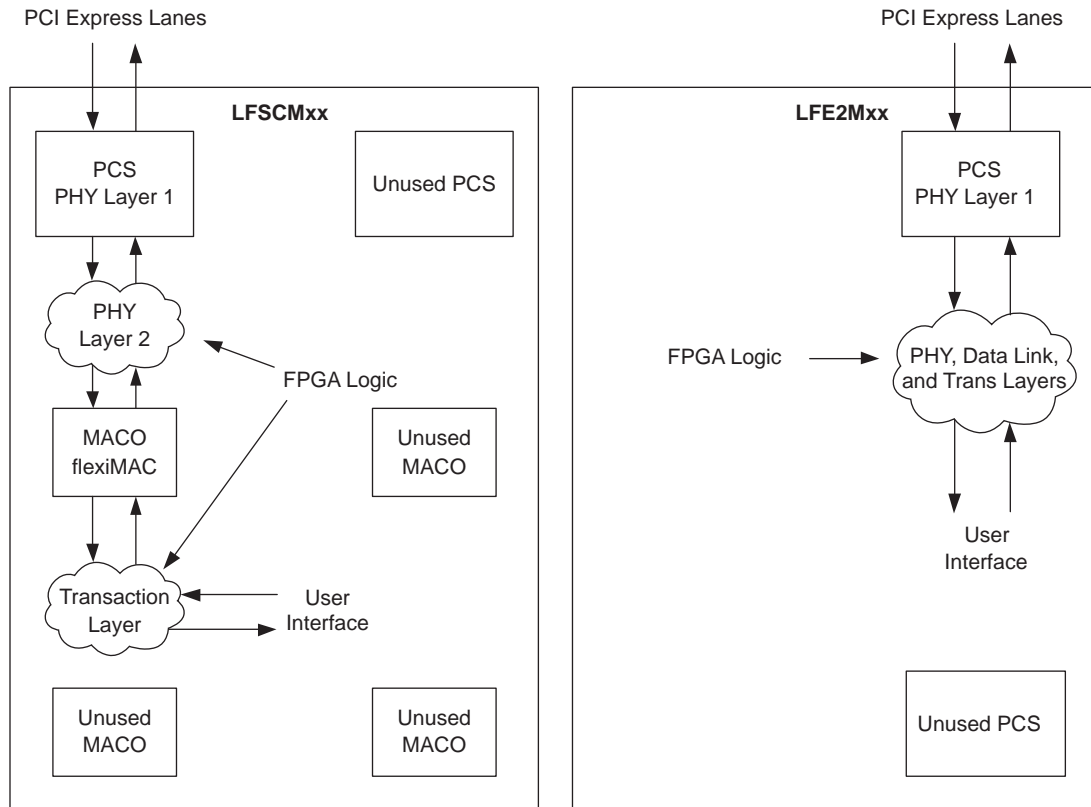
Figure 1 provides a high-level block diagram to illustrate the main functional blocks and the technology used to implement PCI Express functions.

Figure 1. PCI Express IP Core Technology and Functions



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

Figure 2. PCI Express Core Implementation in the LatticeSCM 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 ispLEVER design tools to select regions nearby the hard blocks to achieve the timing goals.

Block Diagram

Figure 3 provides a high-level interface representation.

Figure 3. PCI Express Interfaces

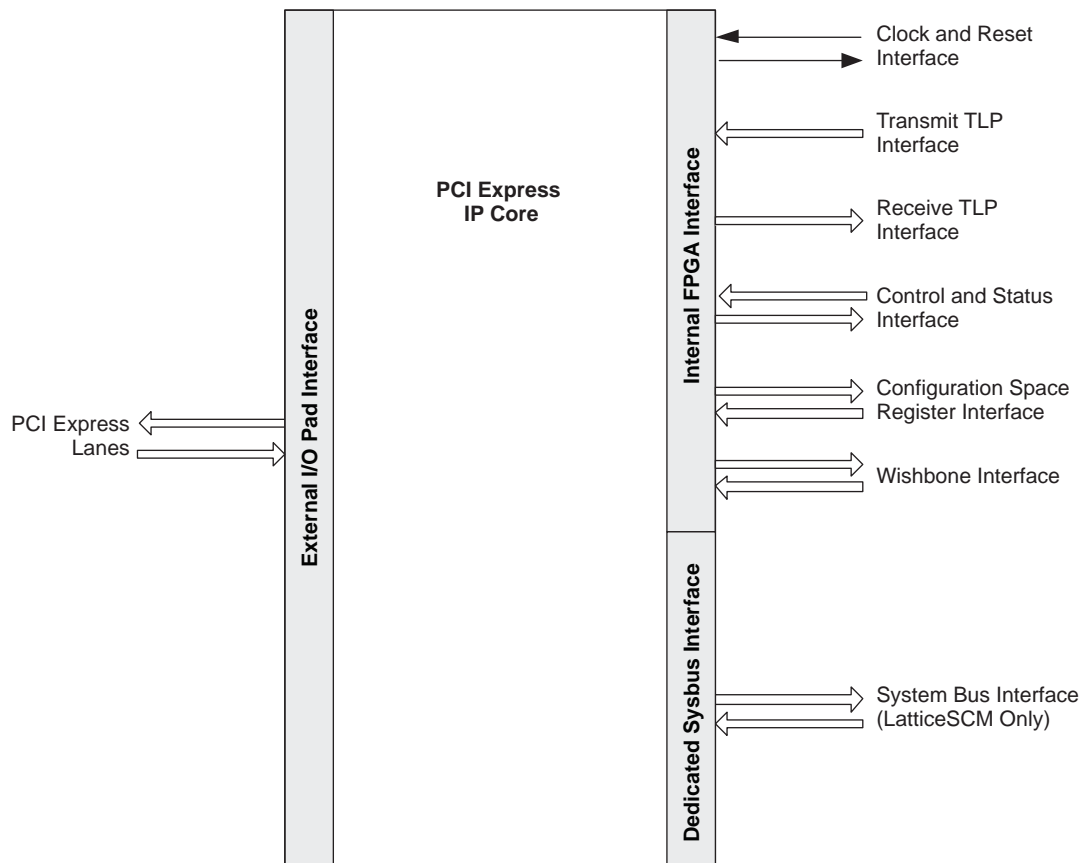


Table 1 provides the list of ports and descriptions for the PCI Express IP core.

Table 1. PCI Express Port List

Port Name	Direction	Clock	Description
Clock and Reset Interface			
refclk[p,n] (LatticeECP2M only)	Input		100MHz PCI Express differential reference clock used to generate the 2.5Gbps data.
refclk_250 (LatticeSCM only)	Input		250MHz 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 250MHz required for this port. This topic is covered in the Clocking Scheme section of this document.
sys_clk_125	Output		125MHz clock derived from refclk to be used in the user application.
rst_n	Input		Active-low asynchronous datapath and state machine reset. This port will be connected to the GSR for the entire device. This reset is pulsed after bitstream download and will not need to be asserted by the user.
serdes_rst	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.

Table 1. PCI Express Port List (Continued)

Port Name	Direction	Clock	Description
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 Appendix A 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		<p>PCI Express 2.5Gbps 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.</p> <p>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</p> <p>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</p>
hdout[p,n]_[0,1,2,3]	Output		<p>PCI Express 2.5Gbps 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.</p> <p>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</p> <p>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</p>
Transmit TLP Interface			
tx_data_vc0[n:0]	Input	sys_clk_125	<p>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 [31:25] Byte N+4 [24:16] Byte N+5 [15: 8] Byte N+6 [7: 0] Byte N+7</p> <p>Native x1 Transmit data bus [15:8] Byte N [7:0] Byte N+1</p>
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.

Table 1. PCI Express Port List (Continued)

Port Name	Direction	Clock	Description
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 x2 or 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 may 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 may 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 1. PCI Express 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 [31:25] Byte N+4 [24:16] 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 IPexpress.
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 - Locked Completions - Configuration Read/Write Type 1 - Vendor Defined Message - 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.
[ph,pd, npn,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 1. PCI Express 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 1. PCI Express Port List (Continued)

Port Name	Direction	Clock	Description
phy_ltssm_substate[2:0]	Output	sys_clk_125	<p>PHY Layer LTSSM current substate. Each major LTSSM state has a series of substates.</p> <p>When phy_ltssm_state=DETECT</p> <p>000 - DET_WAIT 001 - DET_QUIET 010 - DET_GODET1 011 - DET_ACTIVE1 100 - DET_WAIT12MS 101 - DET_GODET2 110 - DET_ACTIVE2 111 - DET_EXIT</p> <p>When phy_ltssm_state=POLLING</p> <p>000 - POL_WAIT 001 - POL_ACTIVE 010 - POL_COMPLIANCE 011 - POL_CONFIG 100 - POL_EXIT</p> <p>When phy_ltssm_state=CONFIG</p> <p>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</p> <p>When phy_ltssm_state=L0</p> <p>000 - L0_WAIT 001 - L0_L0 010 - L0_L0RX 011 - L0_L0TX 100 - L0_IDLE_0 101 - L0_IDLE_1 110 - L0_EXIT</p> <p>When phy_ltssm_state=L0s</p> <p>000 - L0s_RX_WAIT 001 - L0s_RX_ENTRY 010 - L0s_RX_IDLE 011 - L0s_RX_FTS 100 - L0s_RX_EXIT</p> <p>When phy_ltssm_state=L1</p> <p>000 - L1_WAIT 001 - L1_ENTRY 010 - L1_IDLE 011 - L1_EXIT</p> <p>When phy_ltssm_state=L2</p> <p>000 - L2_WAIT 001 - L2_IDLE</p>

Table 1. PCI Express 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 = 1 010 - Link width = 2 100 - Link width = 4
phy_pol_compliance	Output	sys_clk_125	Active high indicator that the LTSSM is in the Polling.Compliance state.
phy_mloopback	Output	sys_clk_125	PHY Layer LTSSM is in the Master Loopback mode.
phy_sloopback	Output	sys_clk_125	PHY Layer LTSSM is in the Slave Loopback mode.
phy_l0s_tx_state[2:0]	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]	Output	sys_clk_125	PHY Layer LTSSM L1 state 00 - Wait 01 - Entry 10 - Idle 11 - Exit
phy_l2_state[1:0]	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	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	Output		This output clock is used to transmit and receive data using the Master Loopback data path. This port is only available if the Master Loopback feature is enabled in IPexpress.
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 IPexpress.

Table 1. PCI Express 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 IPexpress. [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 IPexpress. [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 IPexpress. [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 IPexpress. [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 prog_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_pmtyp DLLP 10 - Send DLLP using tx_vsd_data Vendor Defined DLLP 11 - Not used
tx_pmtyp[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 1. PCI Express 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
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 is 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 is an input and is used to turn on ECRC checking.
cmpln_tout	Input	sys_clk_125	Completion Timeout Indicator. Used to force non-fatal error message generation and also set appropriate bit in AER.
cmpltr_abort	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. The header data should be provided on the same clock cycle as the unexp_cmpln or cmpltr_abort.
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.
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].

Table 1. PCI Express Port List (Continued)

Port Name	Direction	Clock	Description
inta_n	Input	sys_clk_125	<p>Legacy INTx interrupt request. Falling edge will produce a ASSERT_INTx message and set the Interrupt Status bit to a 1. Rising edge will produce a DEASSERT_INTx 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.</p> <p>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.</p> <p>If the inta_n port is low indicating an ASSERT_INTx 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_INTx message. This can be automatically performed by using a logic OR between the inta_n and cmd_reg_out[5] port.</p>
msi[7:0]	Input	sys_clk_125	<p>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.</p> <p>[7] - MSI 8 [6] - MSI 7 [5] - MSI 6 [4] - MSI 5 [3] - MSI 4 [2] - MSI 3 [1] - MSI 2 [0] - MSI 1</p>
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.
pm_power_state[1:0]	Output	Async	<p>Power State in the Power Management Capability Structure. Software sets this state to place the endpoint in a particular state.</p> <p>00 - D0 01 - D1 10 - D2 11 - D3</p>
load_id	Input	Async	This port is only present when the "Load IDs from Ports" checkbox is enabled in IPexpress. 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_vendor_id[15:0], and subsys_id[15:0] will be loaded into the proper configuration registers.

Table 1. PCI Express Port List (Continued)

Port Name	Direction	Clock	Description
vendor_id[15:0]	Input	Async	This port is only present when the "Load IDs from Ports" checkbox is enabled in IPexpress. 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 IPexpress. 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 IPexpress. 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 IPexpress. 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 IPexpress. 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 IPexpress. This port will load the subsystem device ID for the core on the rising edge of load_id.
Wishbone Interface¹			
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
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
Interrupt output 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 data path 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

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

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

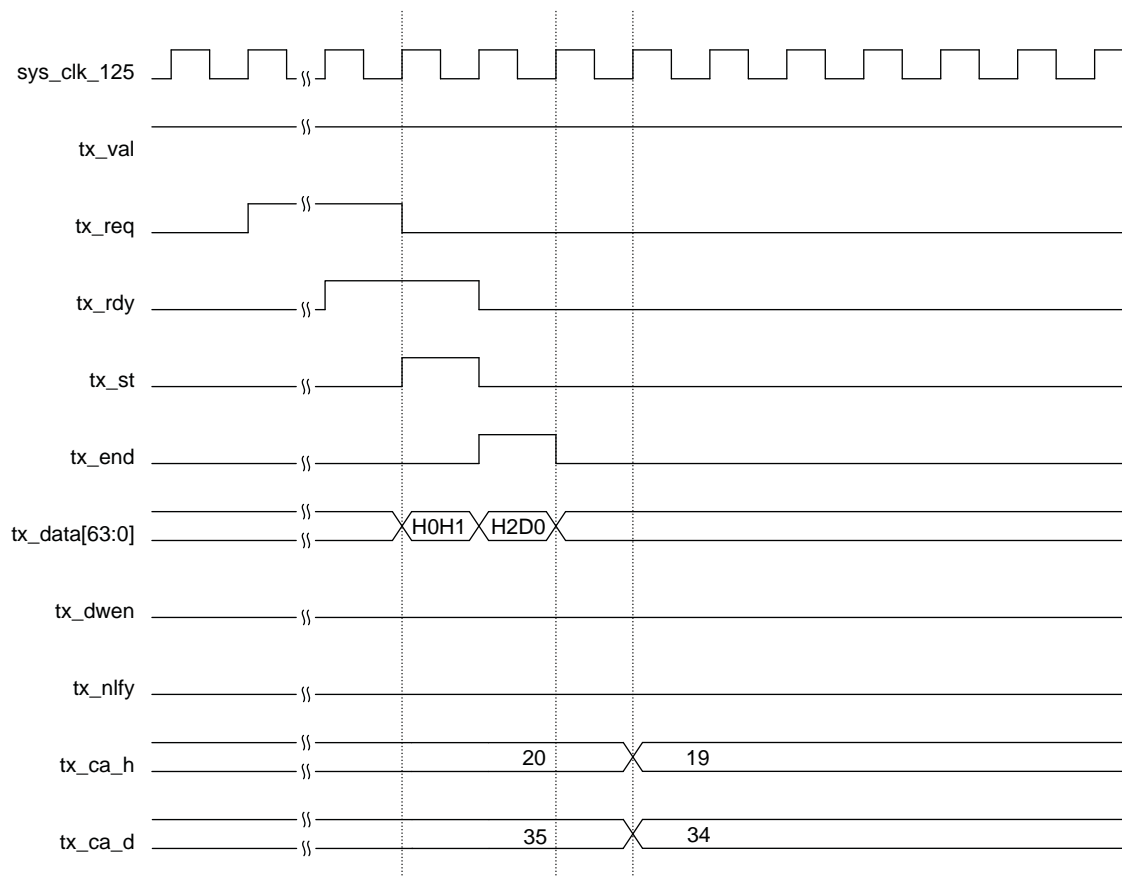


Figure 5. Transmit Interface x4, 3DW Header, 2 DW Data

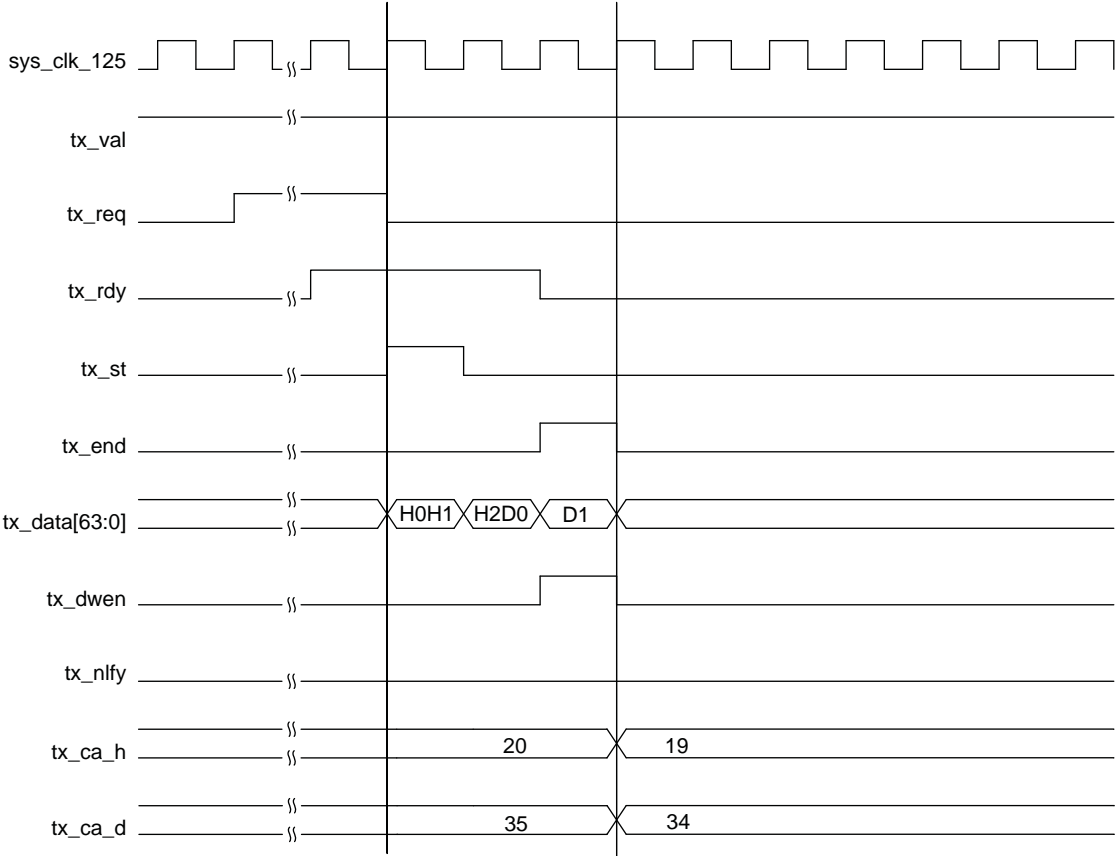


Figure 6. Transmit Interface x4, 4DW Header, 0 DW

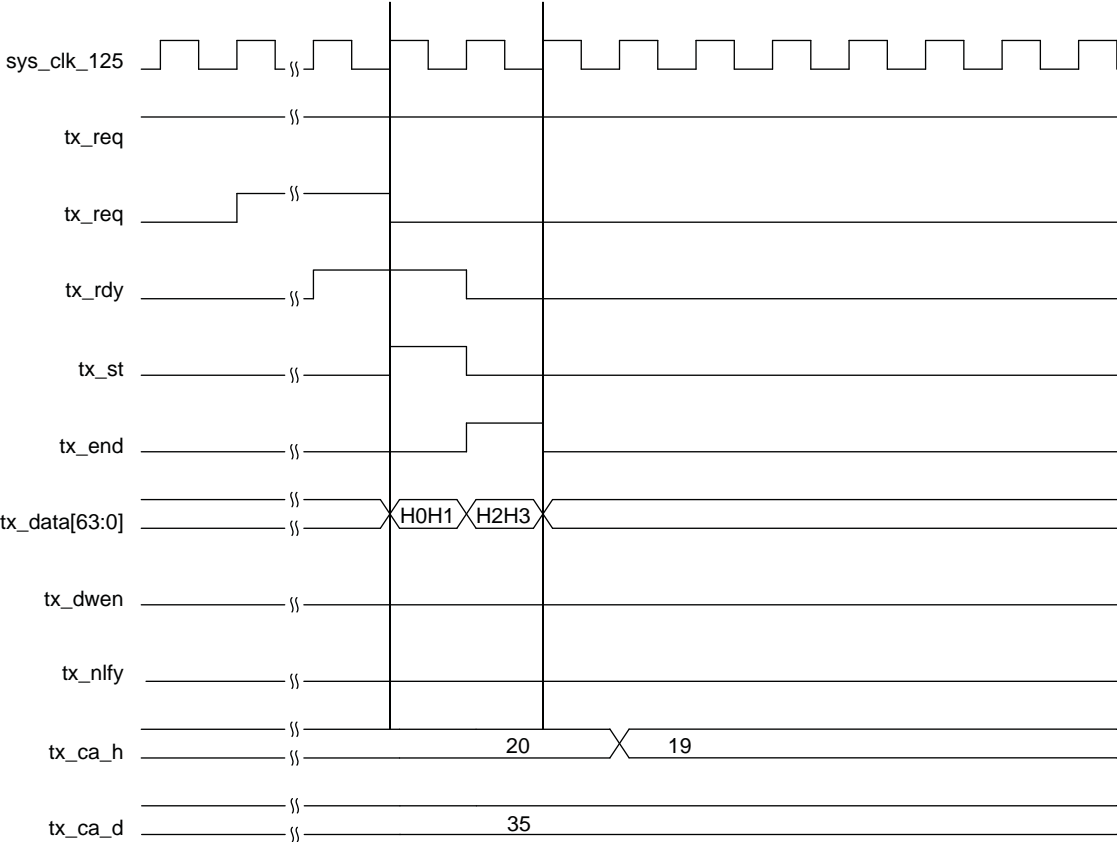


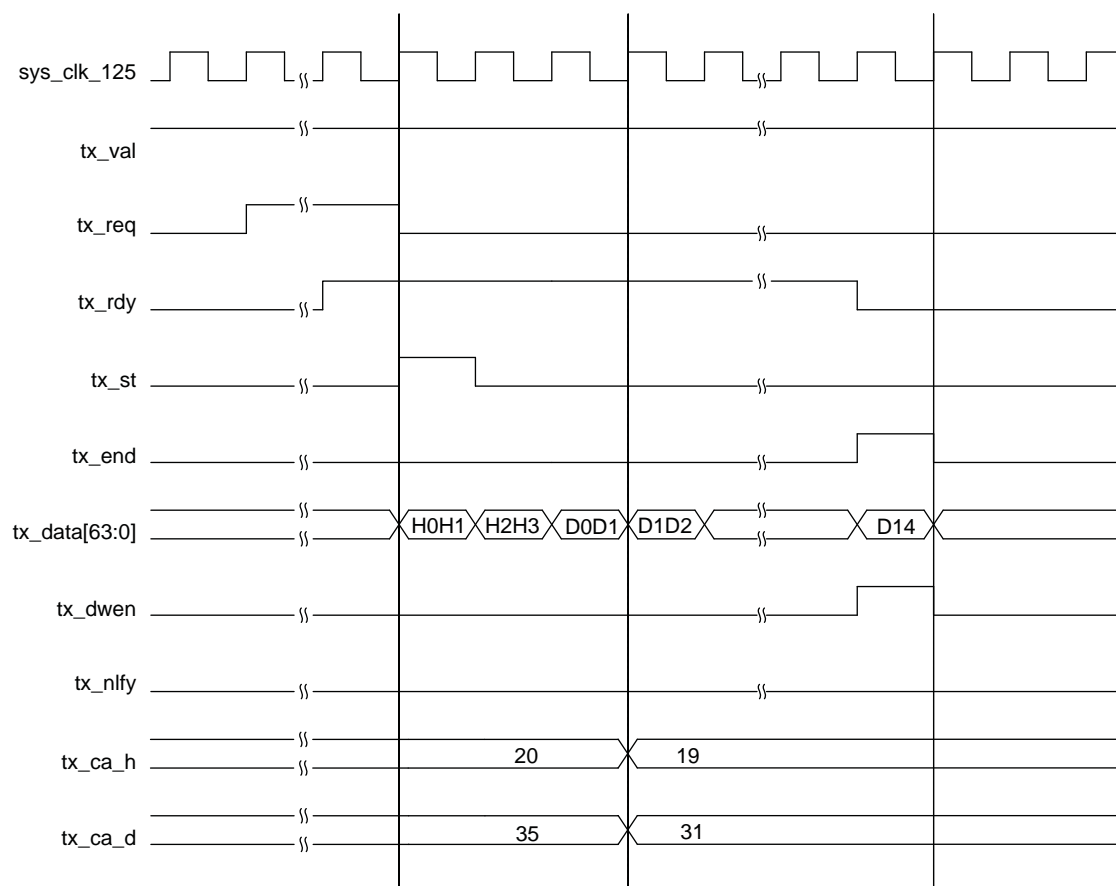
Figure 7. Transmit Interface x4, 4DW Header, Odd Number of DWs

Figure 8. Transmit Interface x4, Burst of Two TLPs

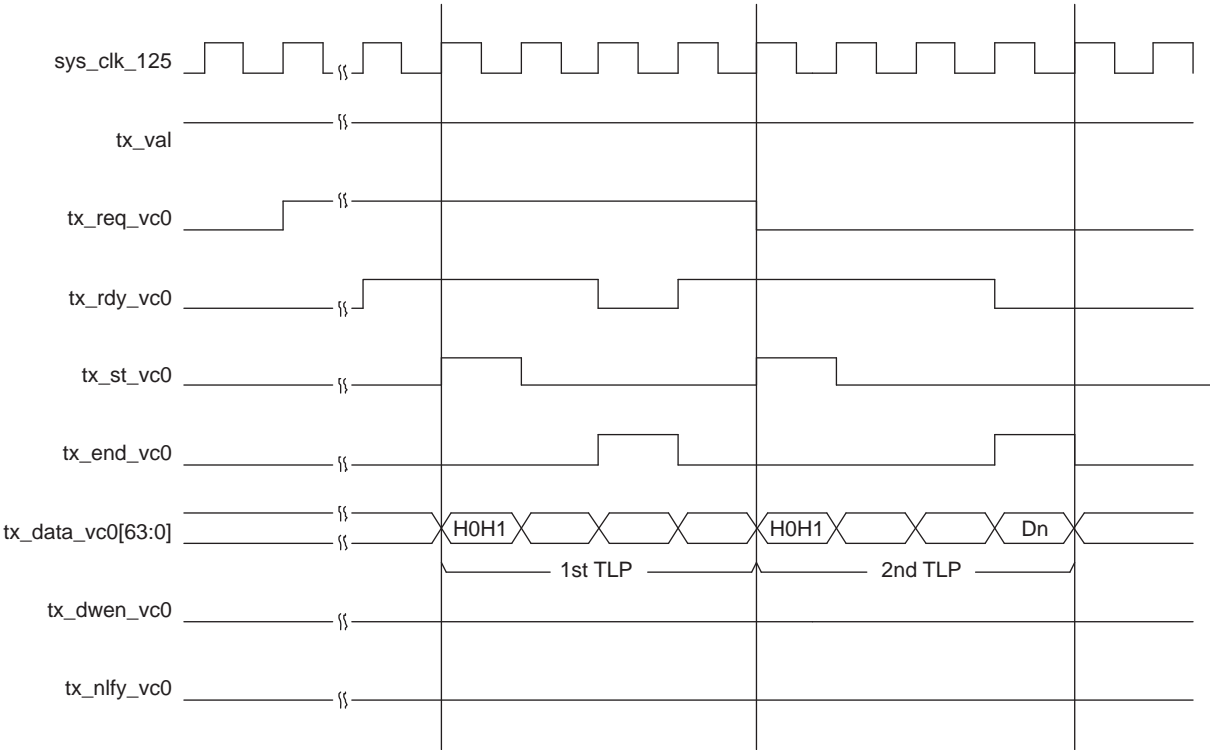


Figure 9. Transmit Interface x4, Nullified TLP

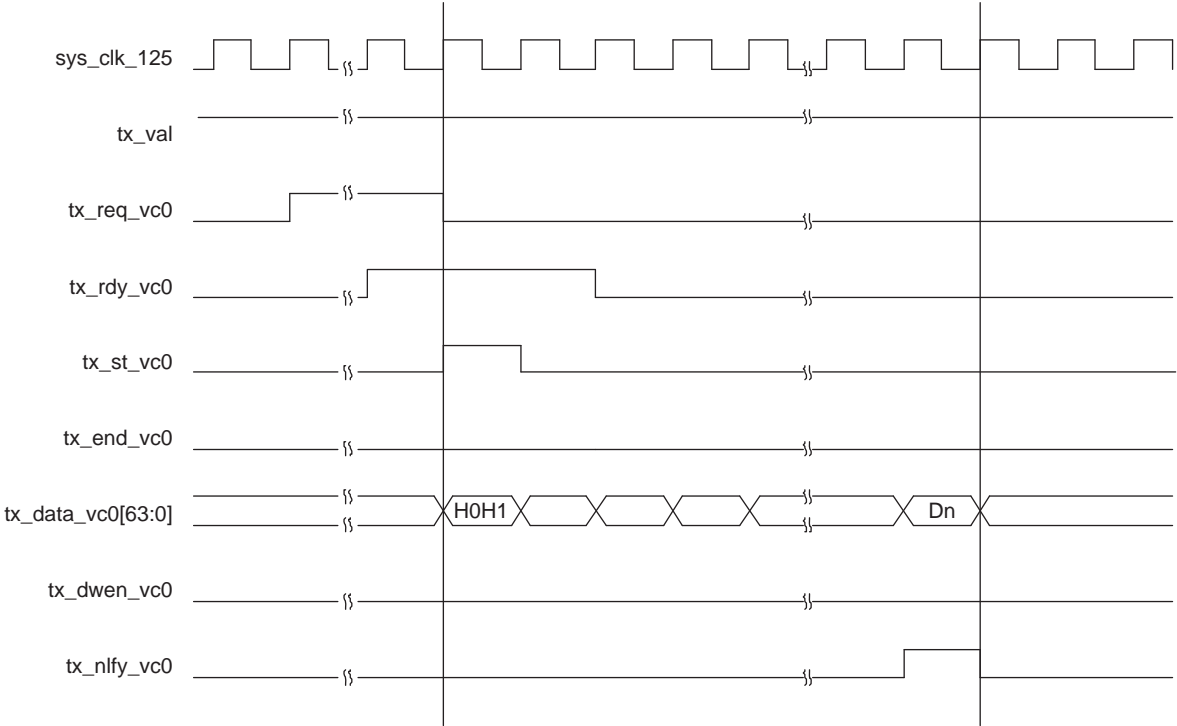


Figure 10. Transmit Interface x1 Downgrade Using tx_val

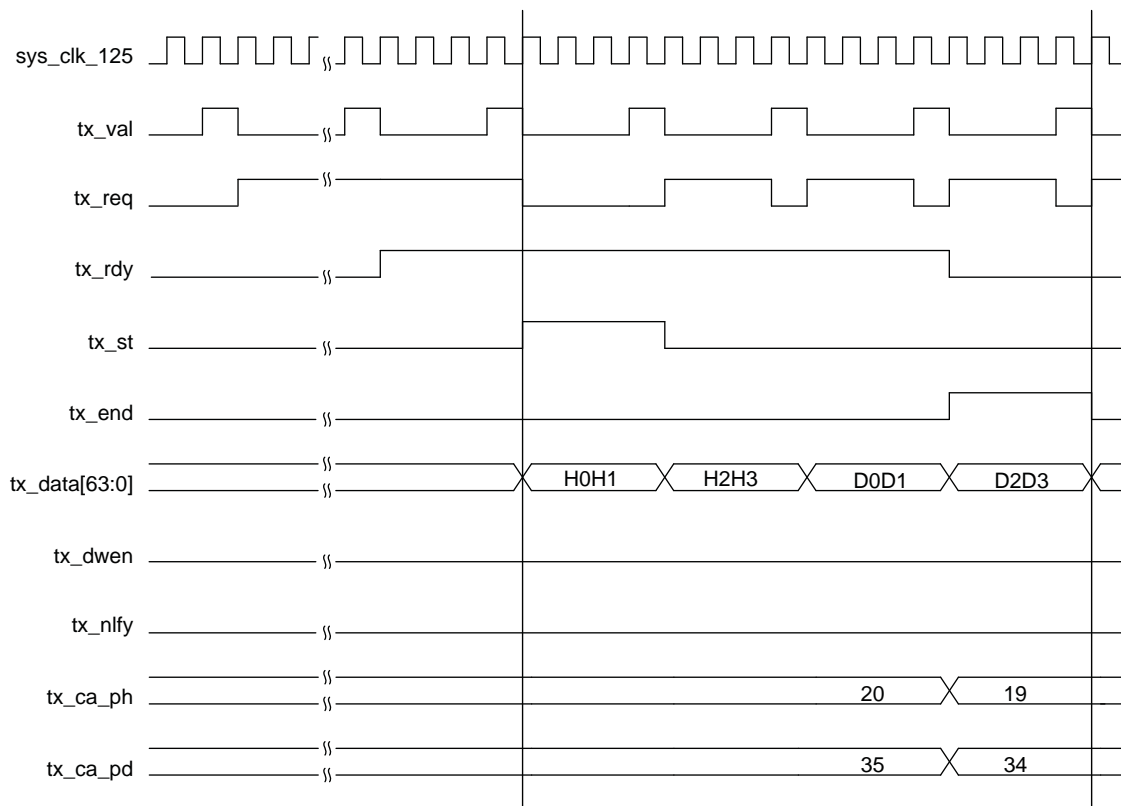
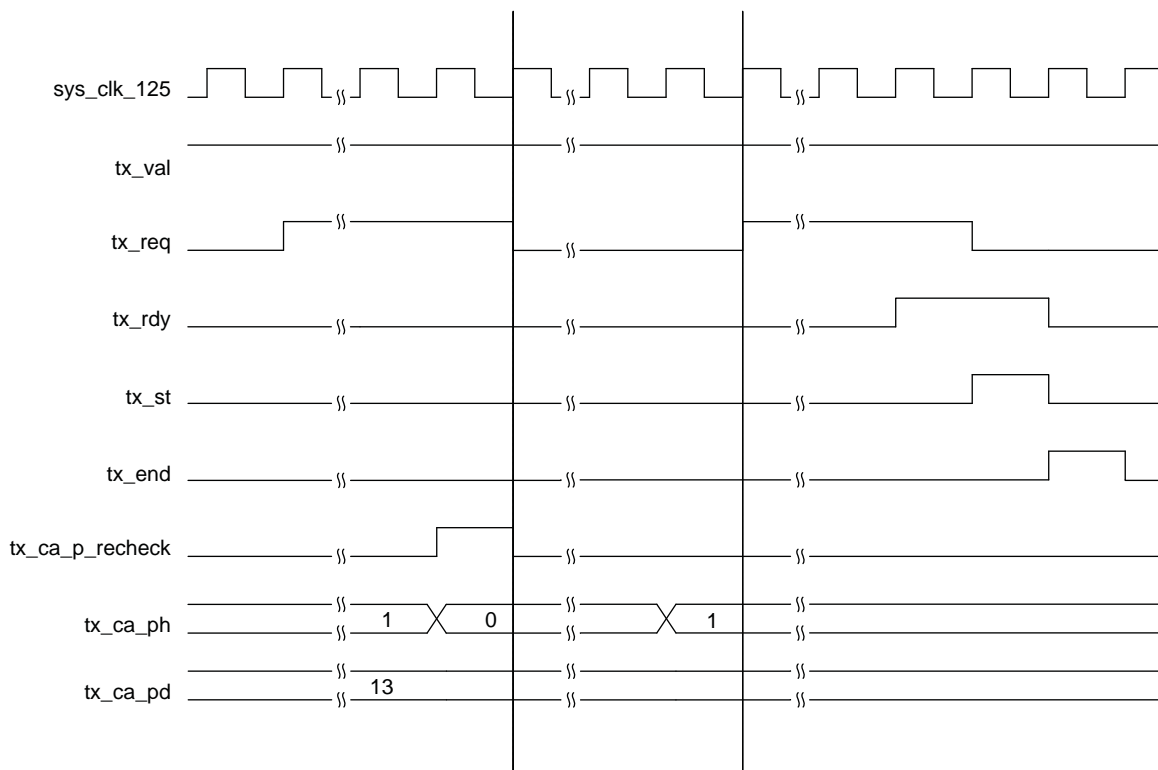


Figure 11. Transmit Interface x4 Posted Request with tx_ca_p-recheck Assertion



Transmit TLP Interface Waveforms for Native x1

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

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

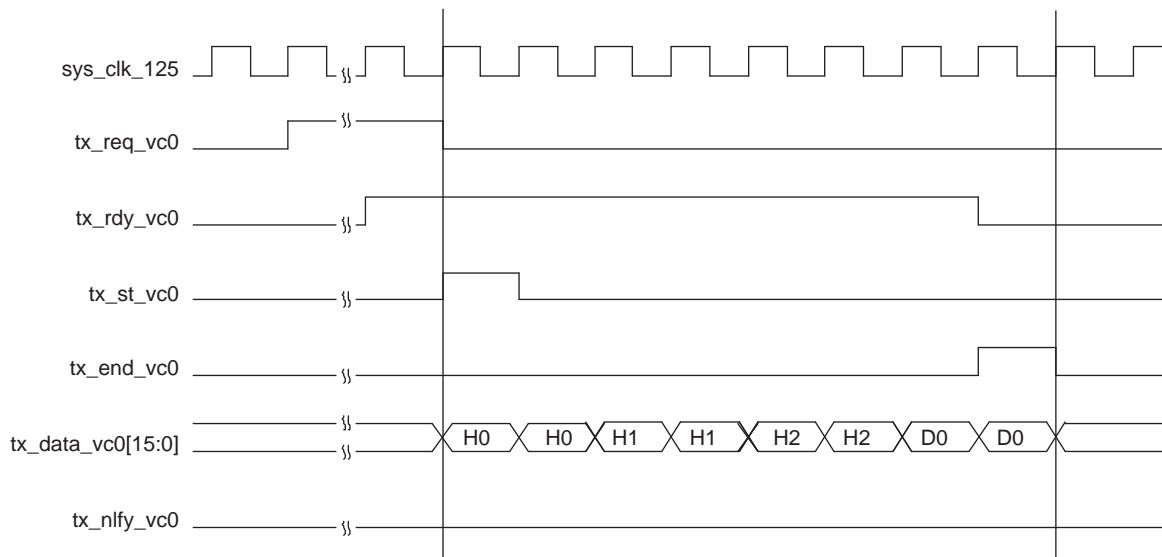


Figure 13. Transmit Interface Native x1, Burst of Two TLPs

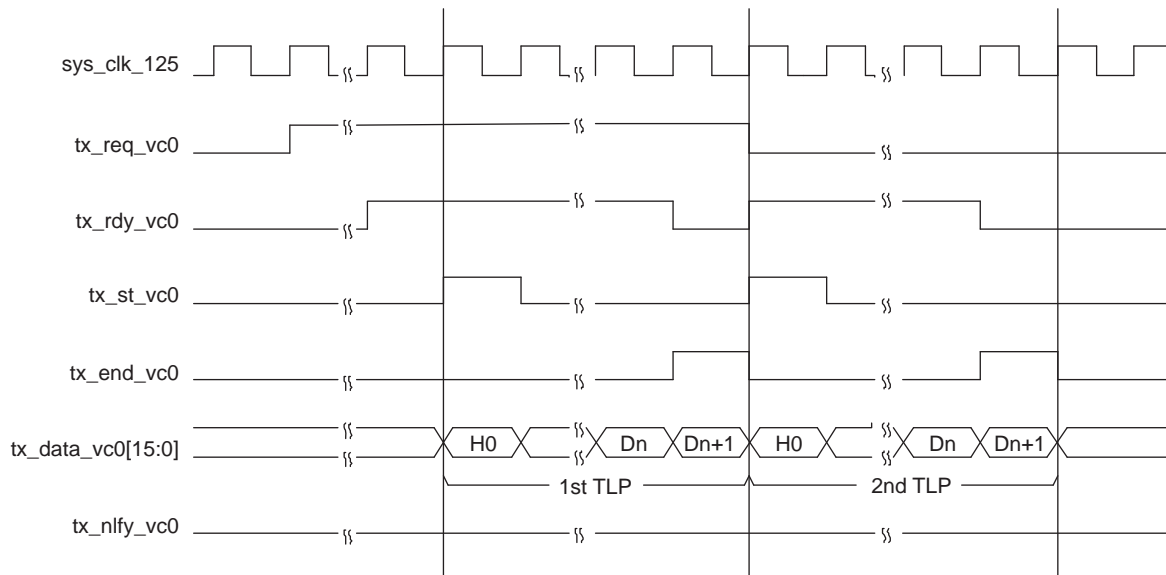
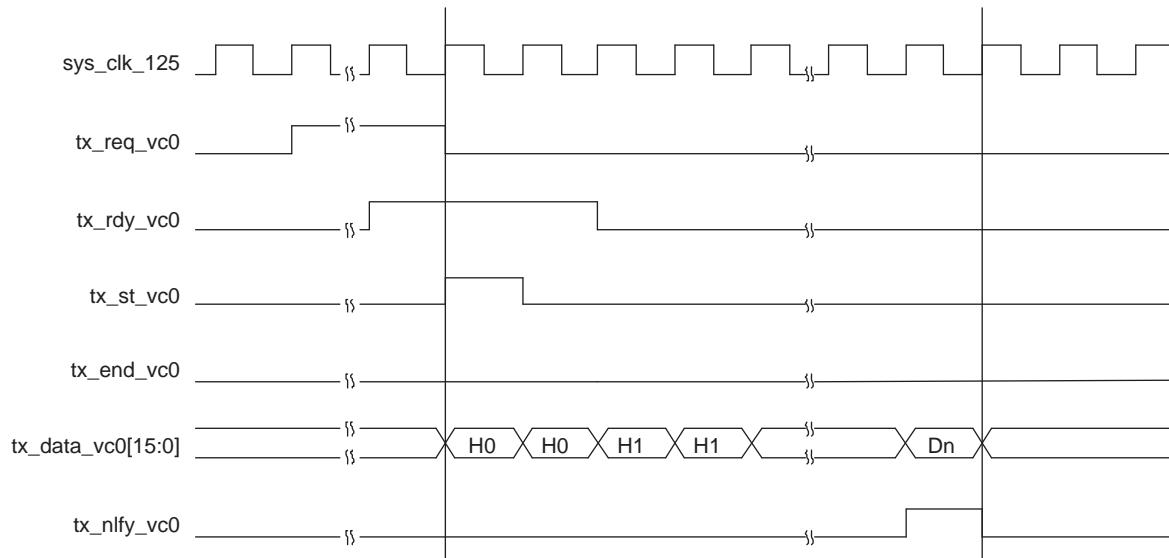
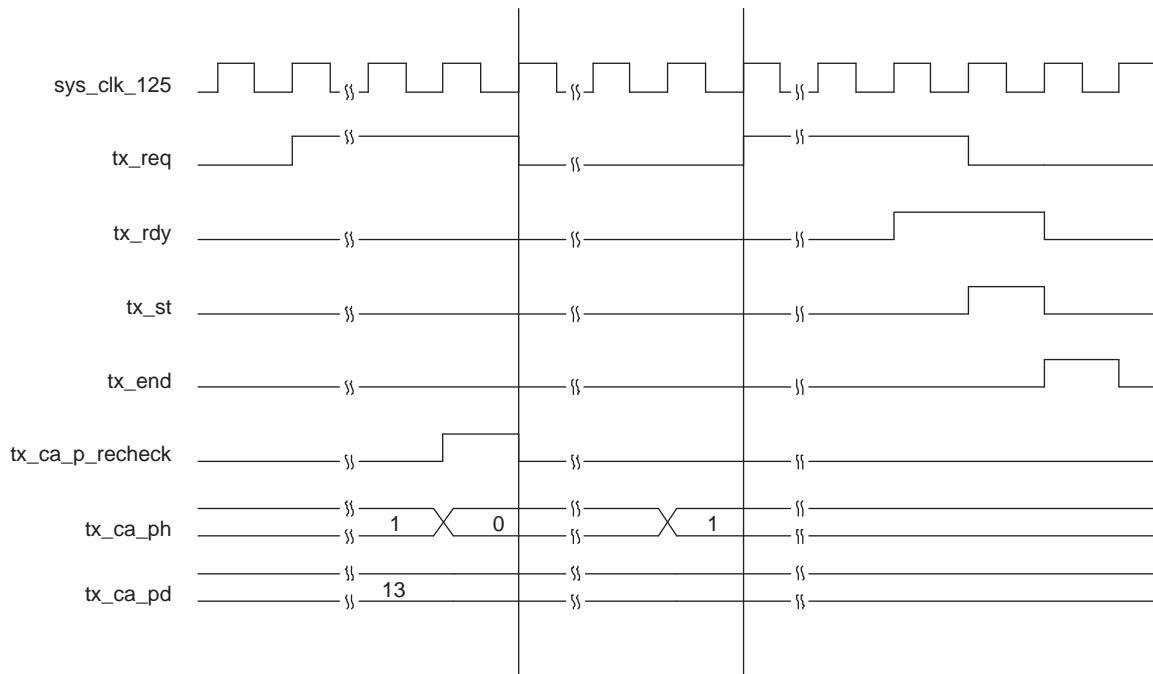


Figure 14. Transmit Interface Native x1, Nullified TLP**Figure 15. Transmit Interface Native x1 Posted Request with tx_ca_p-recheck Assertion**

Receive TLP Interface

In the receive direction, TLPs will come from the core as they are received on the PCI Express lanes. Config reads 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 16 through Figure 19 provides 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 16. Receive Interface, Clean TLP

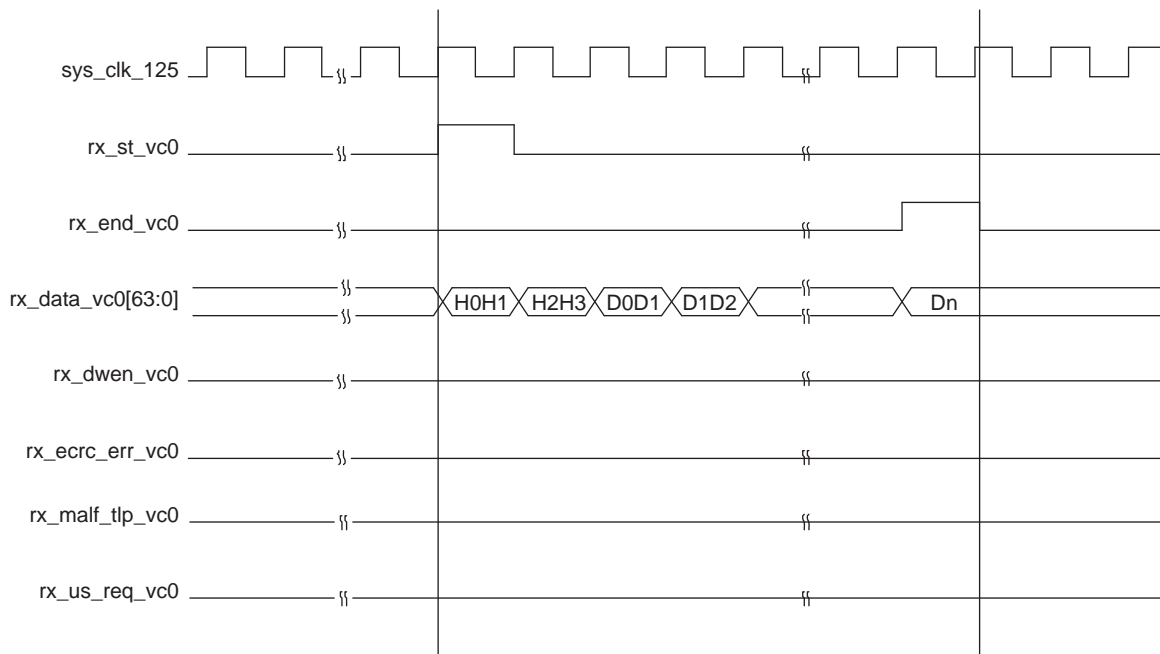


Figure 17. Receive Interface, ECRC Errored TLP

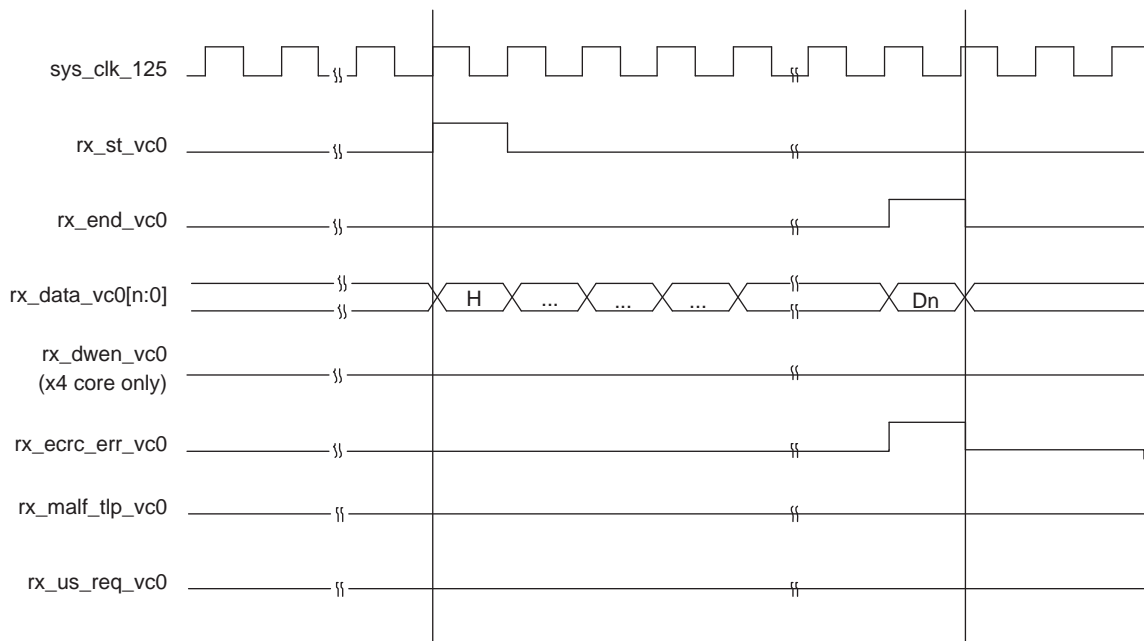
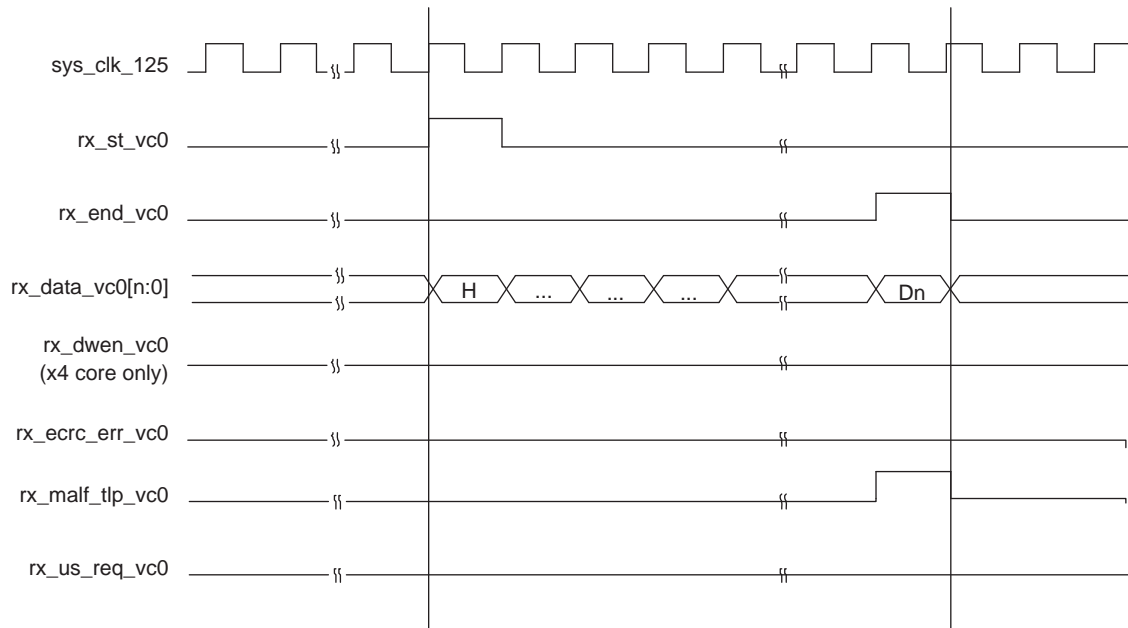
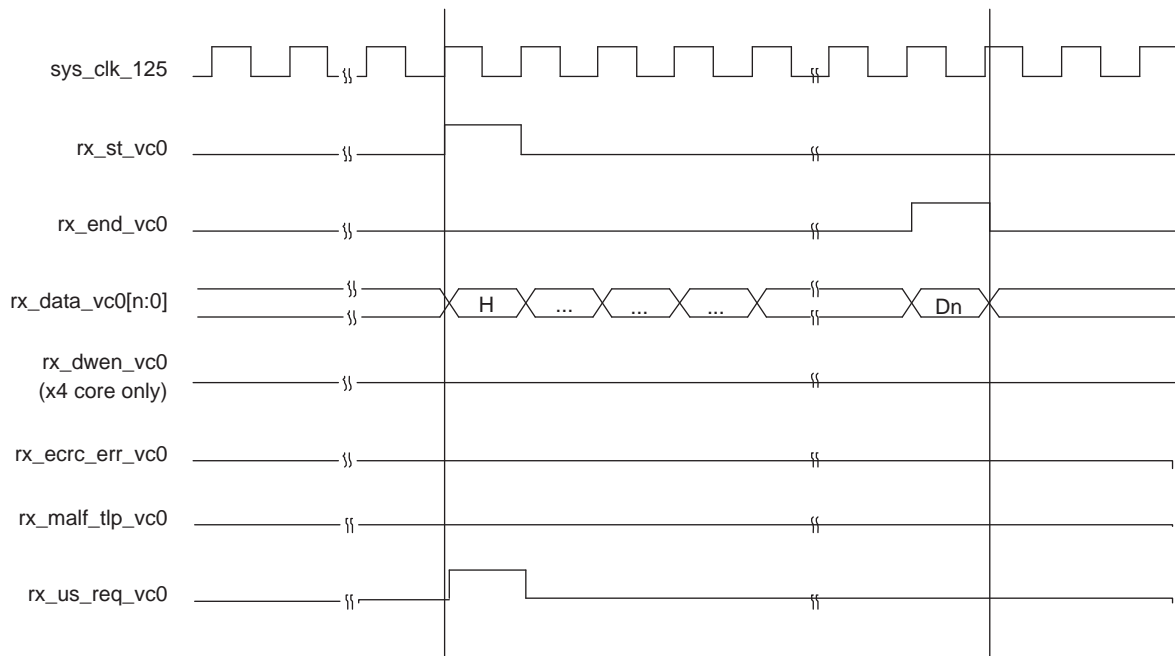


Figure 18. Receive Interface, Malformed TLP**Figure 19. Receive Interface, Unsupported Request TLP**

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.

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. 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 provide a NAK for this TLP and the far end will retransmit.

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 needs 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 this number is less than 4 then the number of data credits is 1. If this number is greater than or equal to 4 then simply shift the length field right by 2 to divide by 4. 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 provide a NAK for this TLP and the far end will retransmit.

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. Once the TLP's data is stored the Completion credits for the CplD must be released to the far end. This is done using the cplh_processed_vc0, cpld_processed_vc0, and cpld_num_vc0 ports. Each CplD TLP takes 1 header credit. There is 1 data credit used per 4 DWs of data. The length field (rx_data_vc0[41:32]) provides the number of DWs used in the CplD TLP. If this number is less than 4 then the number of data credits is 1. If this number is greater than or equal to 4 then simply shift the length field right by 2 to divide by 4. The number of credits used should then be placed on cpld_num_vc0[7:0]. Assert cplh_processed_vc0 and cpld_processed_vc0 for 1 clock cycle to latch in the cpld_num_vc0 port and release credits.

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 Configuration Type1 request, 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)

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.

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, unexp_cmpln, and err_tlp_header ports to provide error conditions to the AER.

Handling of Configuration Requests

Table 2 provides the Configuration Space memory map.

Table 2. 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 - 0xA3	PCI Express CS	24 - 28
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 may 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. The memory map for the Wishbone interface is provided in Table .

Table 3. 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
		26:3			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
Status	1008-100B	31:24	R	0	Reserved
		23:20	R		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	COW		PHY Receive/Rx Electrical Idle. For x1 Core [15:13] are Reserved.
		11:7	R		LTSSM State *
		6:3	R		DLL/Link Control SM Status [6] - DL Inactive State [5] - DL Init State [4] - DL Active State [3] - DL Up State
		2:0	R		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 3. Wishbone Interface Memory Map (Continued)

Type	Address (hex)	Bits	Default	Description
		12		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	Reserved
		29:16		ACK/NAK Latency Timer
		15		Reserved
		14:10		Number of FTS
		9:0		SKP Insert Counter
	1014-1017	31:29	R/W	Reserved
		28:24		Update Frequency for Posted Header
		23:16		Update Frequency for Posted Data
		15:13		Reserved
		12:8		Update Frequency for Non-posted Header
		7:0		Update Frequency for Non-posted Data
	1018-101B	31:29	R/W	Reserved
		28:24		Update Frequency for Completion Headers
		23:16		Update Frequency for Completion Data
		15:12		Reserved
		11:0		Update Timer
	101C-101F	31:16	R/W	Reserved
		15:0		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 Write

IP Core Implementation

Getting Started with the Design Flow

This document provides information on how to use the Lattice PCI Express IP core but does not include all of the information found in the PCI-SIG PCI Express 1.1 specification. It is strongly encouraged for users to consult the PCI-SIG PCI Express 1.1 specification, located at www.pci-sig.com, to understand the framework in which this core is utilized.

The Lattice PCI Express core is supported in ispLEVER® as part of IPexpress™. For more information on the IPexpress design flow please see the [Lattice IPexpress Quick Start Guide](#). This guide provides a high-level overview of the IPexpress application, installation of new cores, and core customization.

A tutorial for using IPexpress is provided in the [ispLeverCore™ IP Module Evaluation Tutorial](#). This document guides the user through the complete design flow from module generation to place and route in ispLEVER.

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

Using the PCI Express IP Core

The PCI Express IP core is an ispLeverCORE which uses the standard IPexpress design flow. It is configured and created using the IPexpress GUI provided with the ispLEVER design tools. IPexpress creates all of the files necessary for the user to instantiate the IP core, simulate the core, synthesize the user's design using the core as a black box, and place and route the design in ispLEVER including the core.

You can use the IPexpress software tool to help generate new configurations of this IP core. Details regarding the usage of IPexpress can be found in the IPexpress and ispLEVER help system. For more information on the ispLEVER design tools, visit the Lattice website at www.latticesemi.com/software.

All LatticeSCM MACO IP utilized by the PCIexpress IP core 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. Larger FPGA devices will have more instances of MACO IP. Please refer to the Lattice web pages on LatticeSCM and MACO IP for more information.

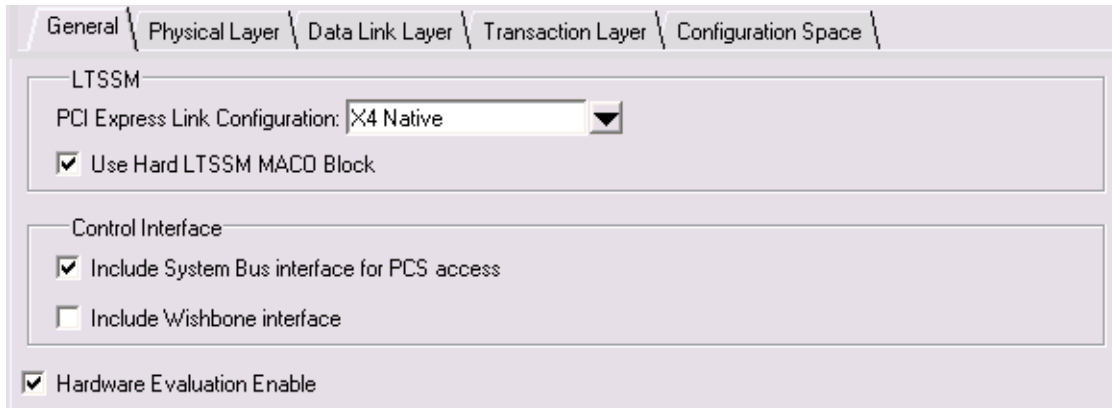
LatticeSCM MACO IP cores are available free of charge through ispLEVER. A license key is required to enable timing simulation and bitstream generation. Please contact your local Lattice sales office to obtain your MACO IP license key. A license is also required for the LatticeECP2M version of the PCI Express IP core. Please contact Lattice sales to obtain a license.

For more information, please refer to the [Lattice IPexpress Quick Start Guide](#) and the [Lattice ispLeverCore IP Tutorial](#) for IPexpress.

Creating the IP

IPexpress is used to create all IP and architectural modules in ispLEVER. For the PCI Express core the user is provided with several options separated into multiple tabs.

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

Figure 20. PCI Express IP Core General Options in IPexpress**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, [Lattice 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 the [LatticeSC/M flexiPCS Data Sheet](#).

Appendix A of this document 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 1).

Figure 21. PCI Express IP Core Physical Layer Options in IPexpress

General | Physical Layer | Data Link Layer | Transaction Layer | Configuration Space

LTSSM

☐ Include Master Loop back data path

☒ Include Polling Compliance State

Include Master Loopback Data Path

This feature is currently not supported in the PCI Express core.

Include Polling Compliance State

This option includes the Polling Compliance state to the LTSSM. The Polling Compliance state is used to generate the compliance data pattern used for PCI-SIG electrical testing. This state and data pattern generation is not required for normal operation. By removing the Polling Compliance state a minimal set of FPGA resources can be saved. The Polling Compliance state is part of the MACO LTSSM and therefore is always selected if the MACO LTSSM is used.

Figure 22. PCI Express IP Core Data Link Layer Options in IPexpress

General | Physical Layer | Data Link Layer | Transaction Layer | Configuration Space

Retry Buffer Sizing

Maximum Payload Size (Bytes) ☒ 512 ☐ 1k ☐ 2k ☐ 4k

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:(3750-4095)	4095

Maximum TLP Size

This option is used to size the Retry Buffer contained in the Data Link Layer. The user should select the largest size TLP that will be used in the application. The retry buffer uses Embedded Block RAM (EBR) and will be sized accordingly. The following table provides a total EBR count for the core based on Max TLP Size.

Table 4. Total EBR Count Based on Max TLP Size

Max TLP Size	LatticeECP2M Native x1 EBR Usage	LatticeECP2M Native x4 and x1 Downgraded EBR Usage	LatticeSCM EBR Usage ^{1, 2}
512B	4	11	13
1KB	5	11	13
2KB	9	13	17
4KB	15	18	27

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 LTSSM MACO block used.

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.

Figure 23. PCI Express IP Core Transaction Layer Options in IPexpress

The screenshot shows the IPexpress configuration tool with the 'Transaction Layer' tab selected. The 'Include ECRC Support' checkbox is unchecked. Under the 'Initial Receive Credits' section, the following options are visible:

- ☐ Infinite PH credits: Initial PH credits available:(1-127) 8
- ☐ Infinite PD credits: Initial PD credits available:(8-2047) 32
- ☐ Infinite NPH credits: Initial NPH credits available:(1-127) 8
- ☐ Infinite NPD credits: Initial NPD credits available:(1-2047) 32

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.

Infinite NPH Credits

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

Initial NPH Credits Available

If NPH 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 NPD Credits

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

Initial NPD Credits Available

If NPD 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.

Figure 24. PCI Express IP Core Configuration Space Options in IPexpress

General \ Physical Layer \ Data Link Layer \ Transaction Layer \ Configuration Space

Type0 Config Space

Device ID 0000

Vendor ID 1204

Class Code 000000

Rev ID 00

BIST 00

Header Type 00

BAR0 fffc0000

☒ BAR0 Enable

BAR1 fffc0000

☒ BAR1 Enable

BAR2 00000000

☐ BAR2 Enable

BAR3 00000000

☐ BAR3 Enable

BAR4 00000000

☐ BAR4 Enable

BAR5 00000000

☐ BAR5 Enable

CardBus CIS Pointer 00000000

Subsystem ID 0000

Subsystem Vendor ID 0000

ExpROM Base Addr 00000000

☐ Expansion ROM Enable

☐ Load IDs from ports

Power Management Capability Structure

Power Management Cap Reg [31:16] 0000

Data Scale Multiplier 0

Power Consumed in D0 (Watts) 00

Power Consumed in D1 (Watts) 00

Power Consumed in D2 (Watts) 00

Power Consumed in D3 (Watts) 00

Power Dissipated in D0 (Watts) 00

Power Dissipated in D1 (Watts) 00

Power Dissipated in D2 (Watts) 00

Power Dissipated in D3 (Watts) 00

MSI Capability Structure

☒ Use Message Signaled Interrupts

Number of Messages Requested 1

PCIe Capability Structure

Max Payload Size (Bytes) 128

Device Capabilities Register [27:3] 00000000

☒ Enable Relaxed Ordering

Maximum Link Width 4

Link Capabilities Register [17:10] 00

Device Serial Number

Device Serial Number 0000000000000000

Advanced Error Reporting

☐ Use Advanced Error Reporting

Terminate All Config TLPs

☒ Terminate All Config TLPs

Configuration Space Options

This page of IPexpress 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.

CardBus CIS Pointer

This register is used to point to the location of the Card Information Structure is present in the solution.

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.

Max Payload Size

This option allows the endpoint to advertise the max payload size supported by the endpoint.

Note: In the v3.3 release settings of the Max Payload Size, less than or equal to 512 will incorrectly advertise as 512 in the resulting netlist. This issue will be corrected in the next release.

Device Capabilities Register (27:3)

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

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.

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.

Link Capabilities Register (17:10)

This 8-bit field sets the Link Capabilities Register bits 17:10.

Device Serial Number

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

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.

Terminate All Config 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 inside the core or the user wants to implement certain PCI-SIG ECNs which add registers to capability structures then allowing configuration requests to come to the receive interface will be necessary.

IPexpress-Created Files and Top Level Directory Structure

The design flow for IP created with IPexpress uses a post-synthesized module (NGO) for synthesis and a protected model for simulation. The post-synthesized module is customized and created during IPexpress generation. The

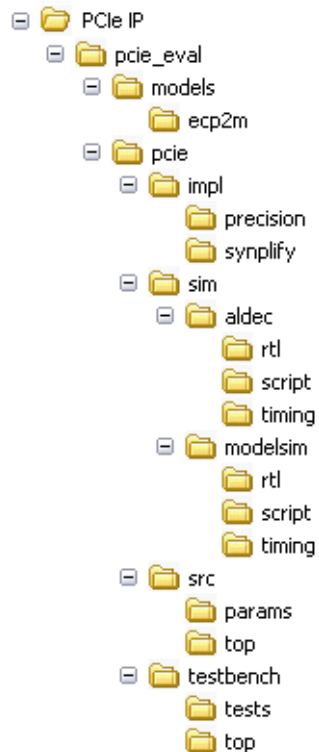
protected simulation model is not customized during IPexpress and relies on parameters provided to customize behavior during simulation.

Table 5 provides a list of files created by IPexpress and how they are used.

Table 5. File List

File	Sim	Synthesis/ ispLEVER	Description
<user name>_inst.v			This file provides an instance template for the IP.
<user name>.v	Yes		This file provides the PCI Express core for simulation.
<user name>_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.
<user name>_bb.v		Yes	This file provides the synthesis black box for the user's synthesis.
<user name>.ngo		Yes	This file provides the synthesized IP core used by ispLEVER. 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 ispLEVER project directory. For the LatticeSC this file is named pcie_pcs.txt. For the LatticeECP2M x4 Native and x1 Downgraded this file is named pcs_pipe_8b_x4.txt. For the LatticeECP2M x1 Native this file is named pcs_pipe_8b_x1.txt.
<user name>.lpc			This file contains the IPexpress options used to recreate or modify the core in IPexpress.
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.
<user name>_eval			This directory contains a sample design. This design is capable of performing a simulation and running through ispLEVER.
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 <user name>_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 <user name>_eval/models/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 <user name>_eval/models/ecp2m directory.
pcs_pipe_top.ngo	Yes		This file provides the synthesized PIPE interface used by ispLEVER. This file needs to be pointed to by the Build step using the search path property.
LatticeSCM-Specific Files			
<user name>_ba_beh.v	Yes		This file provides the back-end simulation library for the PCI Express core.

IPexpress creates several files that are used throughout the design cycle. Most of the files created are customized to the user's module name specified in IPexpress. Figure 25 provides the directory structure created by IPexpress for the PCI Express core, using the evaluation configuration as the example.

Figure 25. PCI Express Core Directory Structure

Most of the files required to be used by the PCI Express core reside in the root directory created by IPexpress. This includes the synthesis black box, simulation model, and example preference file.

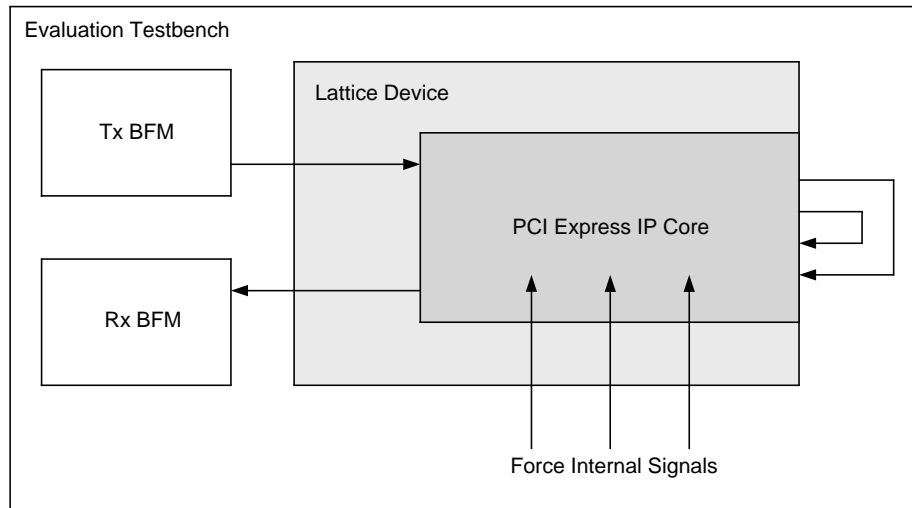
The <username>_eval directory provides an evaluation design which can be used to determine the size of the IP core and a design which can be pushed through ispLEVER including front-end and timing simulations. The models directory provides the library element for the PCS (and PIPE interface for the LatticeECP2M). The <username> directory contains the sample design which pulls the user ports out to external pins. This design can be used to determine the size of the core and to push through the mechanics of the ispLEVER design flow. The impl directory provides either a Precision® RTL synthesis flow or a Synplify® synthesis flow. The sim directory provides rtl and timing simulation for both the Active-HDL® and ModelSim® simulators. The src directory provides the top-level source code for the eval design. The testbench provides a top-level testbench and test case files.

Simulation Strategies

Included with the core from IPexpress is the eval testbench located in the <user name> directory. The intent of the eval testbench is to show the core performing in functional simulation as well as 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 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 26 illustrates this.

Figure 26. PCI Express x4 Core Evaluation Testbench Block Diagram

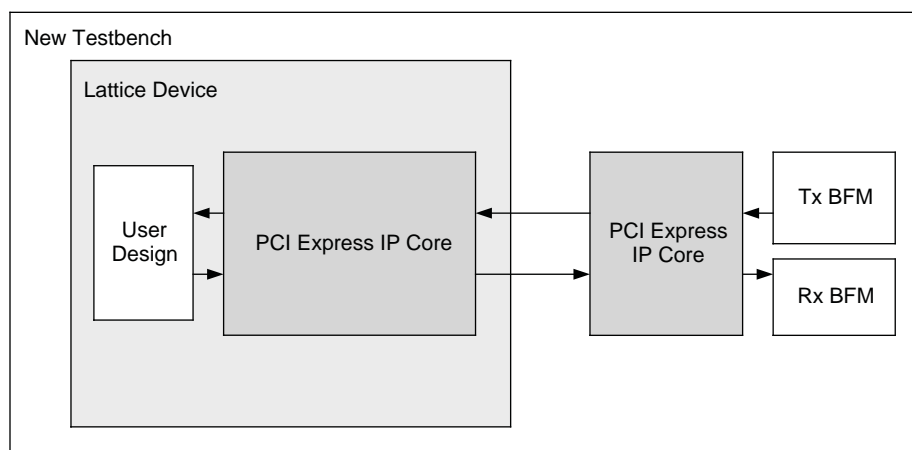
This testbench scheme works for its intent, but it is not easily extendible for the purposes of a Bus Functional Model (BFM) to emulate a real user system. Users can take the testbench provided and modify it to build in their own specific tests.

Sometimes the testbench is oriented differently than users anticipate. Users may 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.

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 27 a block diagram that illustrates a new testbench orientation which can be created by the user.

Figure 27. 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. 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 BFM's 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.

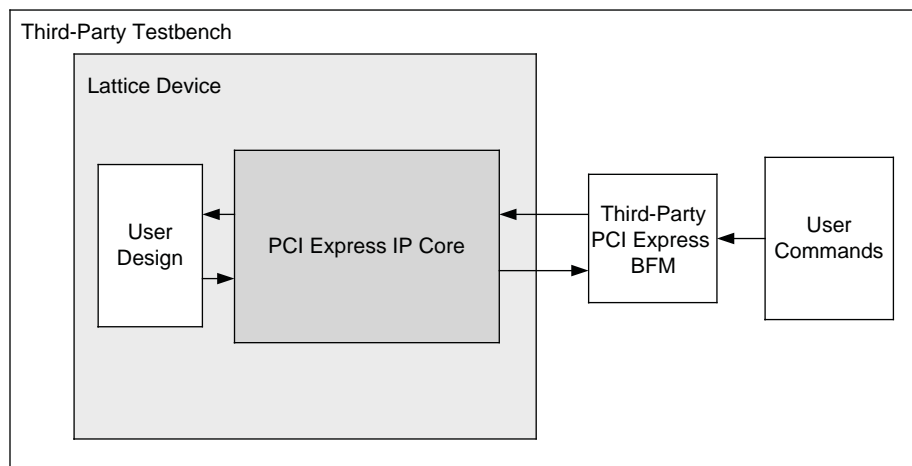
It should also be noted that this approach does not allow for PCI Express layer error insertion.

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 BFM's 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. There are other third party vendors for PCI Express including Denali® and Cadence®.

Figure 28. 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.

Simulation Details

Simulation support in the IP core is provided for Aldec® and ModelSim® simulators. The PCI Express core simulation model is generated from IPexpress with the name <user name>.v. This file calls <user name>_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.

LatticeECP2M PIPE Simulation

The LatticeECP2M PCI Express simulation also requires the PIPE module. This simulation model is found in the <user name>_eval/models/pcs_pipe_top.v file. In the same directory are a few other files that the pcs_pipe_top module requires. These include the following files.

```
pci_exp_params.v
pipe_top.v
pcs_top.v
ctc.v
sync1s.v
PCSC.v
```

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
```

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/ecp2m/pipe_top.v \
    pcie_eval/models/ecp2m/pcs_top.v \
    pcie_eval/models/ecp2m/ctc.v \
    pcie_eval/models/ecp2m/sync1s.v \
    pcie_eval/models/ecp2m/PCSC.v \
    pcie_eval/models/ecp2m/pcs_pipe_top.v

# Compile the user design
vlog top.v

# Compile the testbench
vlog tb.v

# Load the design and libraries
vsim -L ecp2m_vlg -L pcsc_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 6 provides the new values for each of the LTSSM counters when the SIMULATE variable is defined.

Table 6. LTSSM Counters

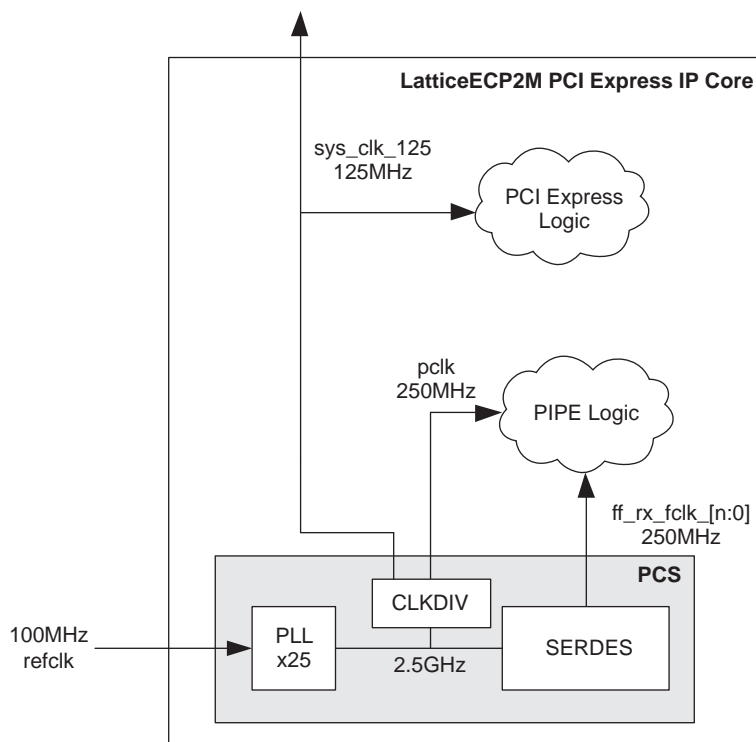
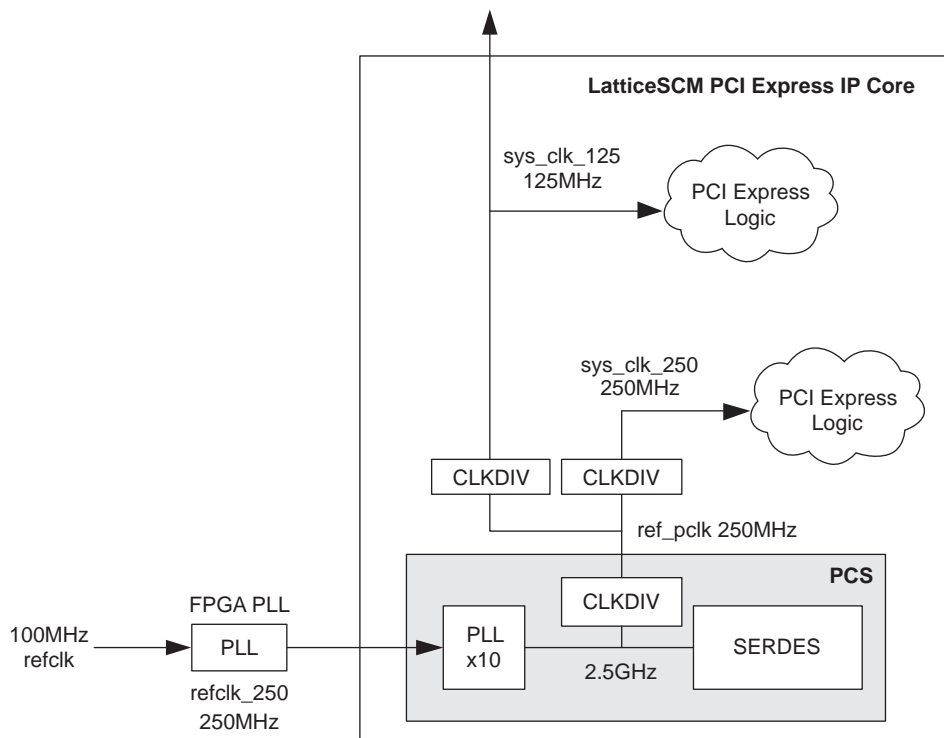
Counter	Normal Value	SIMULATE Value	Description
CNT_1MS	1 ms	800 ns	Electrical Order set received to Electrical Idle condition detected by Loop-back Slave
CNT_1024T1	1024 TS1	48 TS1	Number of TS1s transmitted in Polling.Active
CNT_2MS	2 ms	1200 ns	Configuration.Idle (CFG_IDLE)
CNT_12MS	12 ms	800 ns	Detect.Quiet (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

Implementation Details

The following section discusses several implementation details such as locating the IP and SERDES lanes, setting up the IP core for various modes and applying design constraints in ispLEVER.

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). The following figures provide the internal clocking structures of the IP core in both the LatticeSC and LatticeECP2M families.

Figure 29. LatticeECP2M PCI Express Clocking Scheme**Figure 30. LatticeSCM PCI Express Clocking Scheme**

LatticeECP2M Clocking

The LatticeECP2M clocking solution uses the 100MHz differential refclk provided from the PCI Express link connected directly to the REFCLKP/N of the SERDES. The 100Ohm differential termination is included inside the SERDES so external resistors are not required on the board.

Inside the SERDES, a PLL creates the 2.5Gbps rate from which a transmit 250MHz 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 125MHz clock for the user interface.

LatticeSCM Clocking

The LatticeSCM clocking solution uses the 100MHz differential refclk provided from the PCI Express link. This clock must be connected to an LVDS preferred pin for an FPGA PLL. The 100Ohm 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 250MHz from the 100MHz refclk. A high bandwidth FPGA PLL can be created using IPexpress. Connect the 250MHz 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 2.5Gbps rate will be created. From the SERDES a 250MHz 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 125MHz clock for the user interface.

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 7 lists the site names for the hard blocks on the different device arrays.

Table 7. PCS, flexiMAC and LTSSM Location Site Names for Lattice Devices

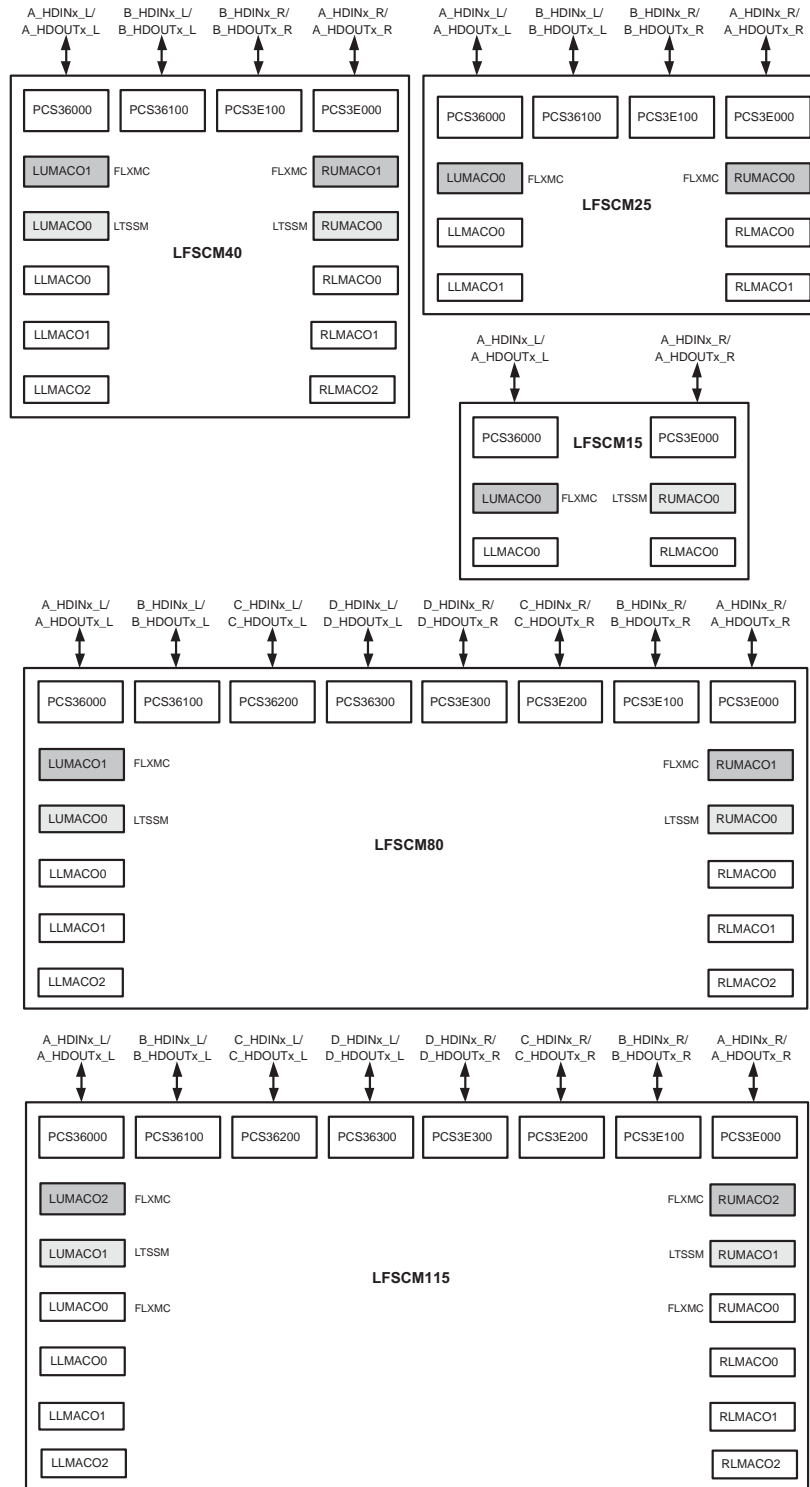
Device	PCS	flexiMAC	LTSSM
LFE2M20	URPCS		
LFE2M35	URPCS		
LFE2M50	URPCS LRPCS		
LFE2M70	URPCS ULPCS LRPCS		
LFE2M100	URPCS ULPCS LRPCS LLPCS		
LFSCM15	PCS36000 PCS3E000	LUMACO0	RUMACO0
LFSCM25	PCS36000 PCS36100 PCS3E000 PCS3E100	RUMACO0 LUMACO0	

Table 7. PCS, flexiMAC and LTSSM Location Site Names for Lattice Devices (Continued)

Device	PCS	flexiMAC	LTSSM
LFSCM40	PCS36000 PCS36100 PCS3E000 PCS3E100	RUMACO1 LUMACO1	RUMACO0 LUMACO0
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

Locating the LatticeSCM Hard Elements

Figure 31 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 31. 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 ispLEVER. 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";
```

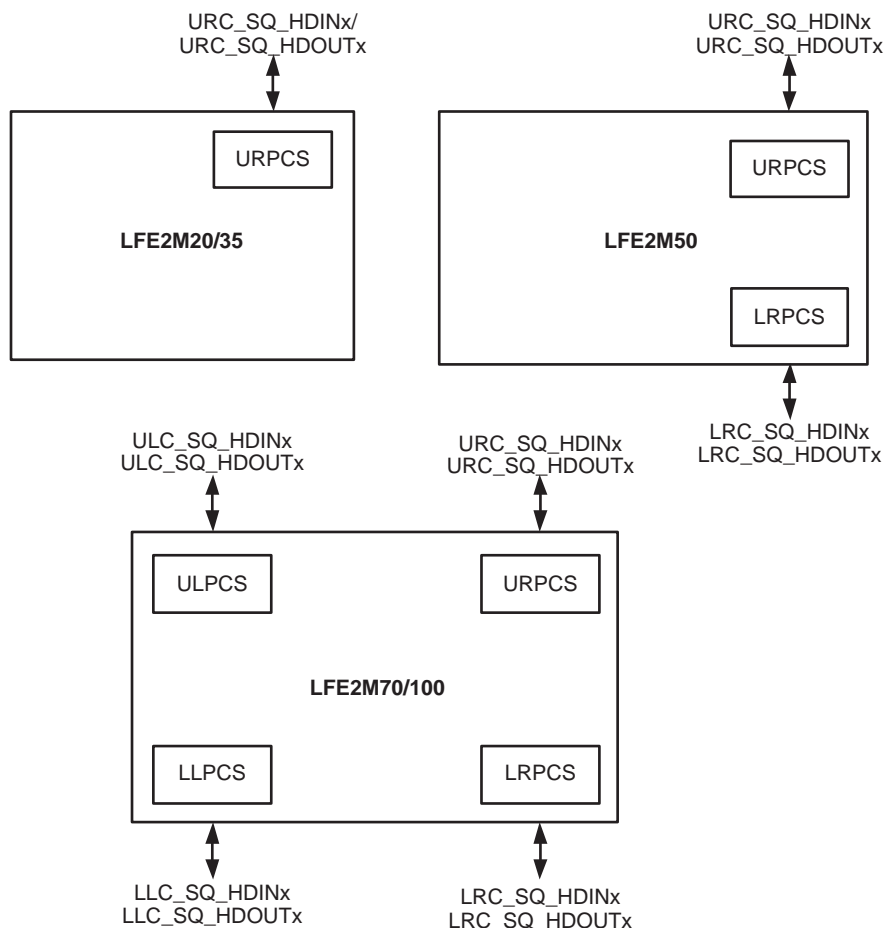
If the MACO LTSSM is not used the hierarchy of the design changes slightly. Here are the locate preferences when using the LUT based LTSSM.

```
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 LatticeECP2M Hard Elements

Figure 32 provides a block diagram with placement positions of the PCS/SERDES quads in the LatticeECP2M devices.

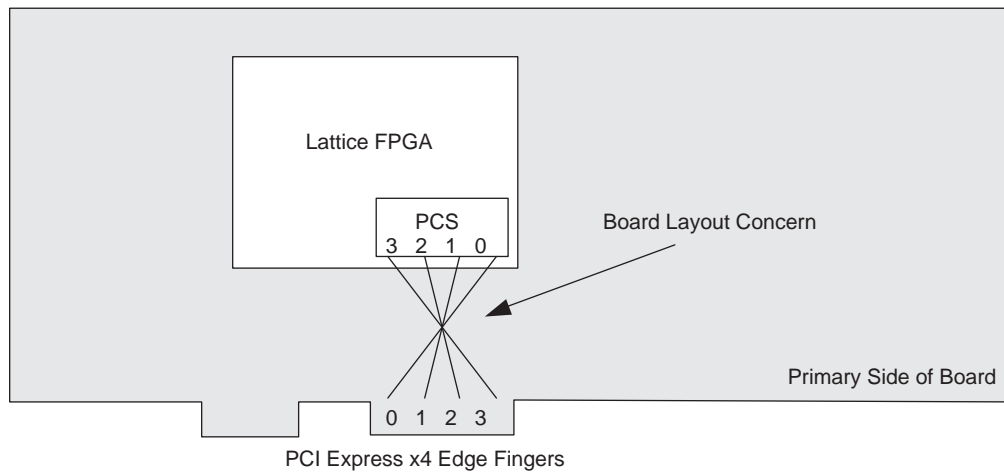
Figure 32. LatticeECP2M Device Arrays with PCS/SERDES

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.

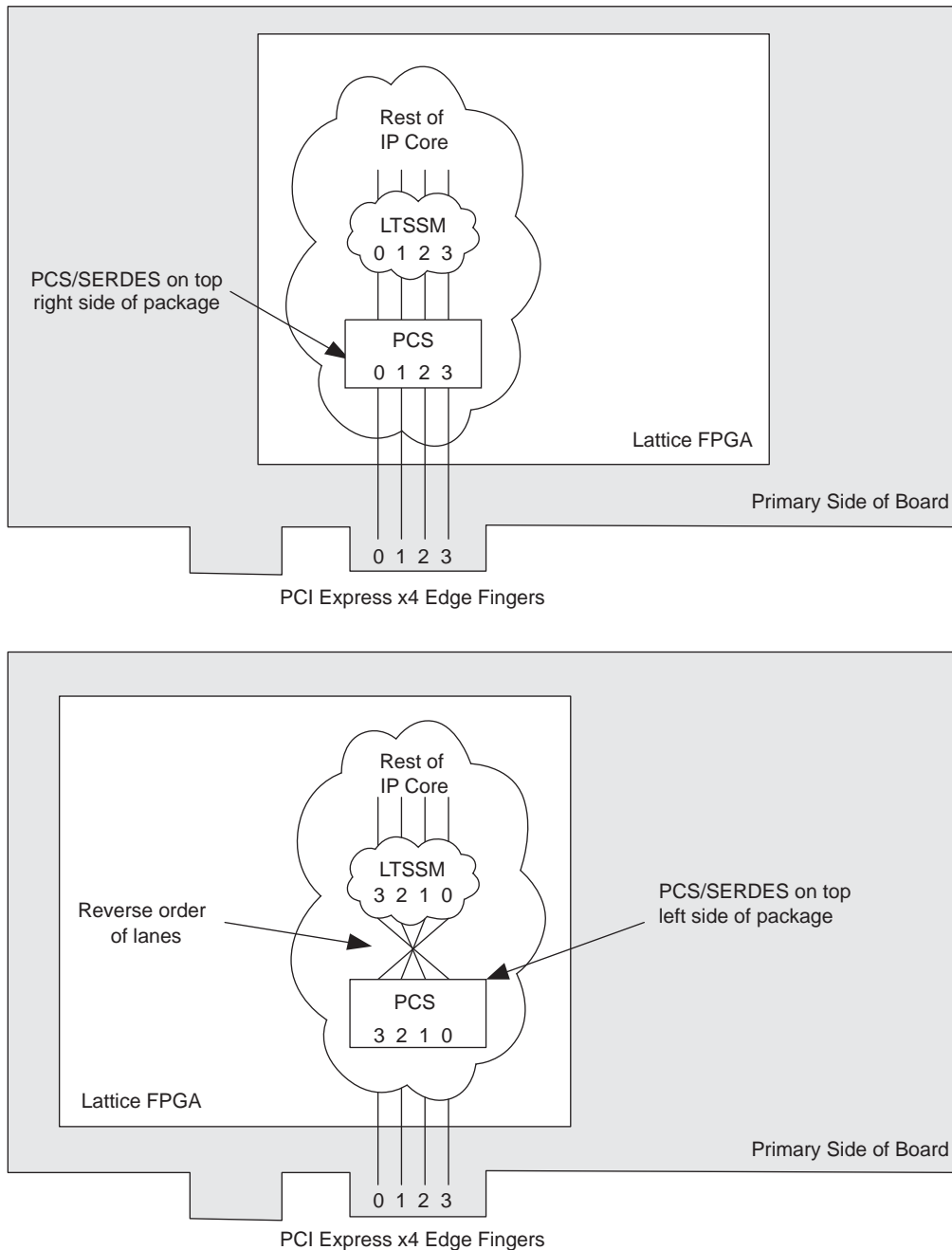
```
LOCATE COMP "<instance name>/ul_pcs_pipe/pcs_top_0/pcs_inst_0" SITE "URPCS" ;
```

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 may 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 33 provides an example of the board layout concern.

Figure 33. Example of Board Layout Concern with x4 Link

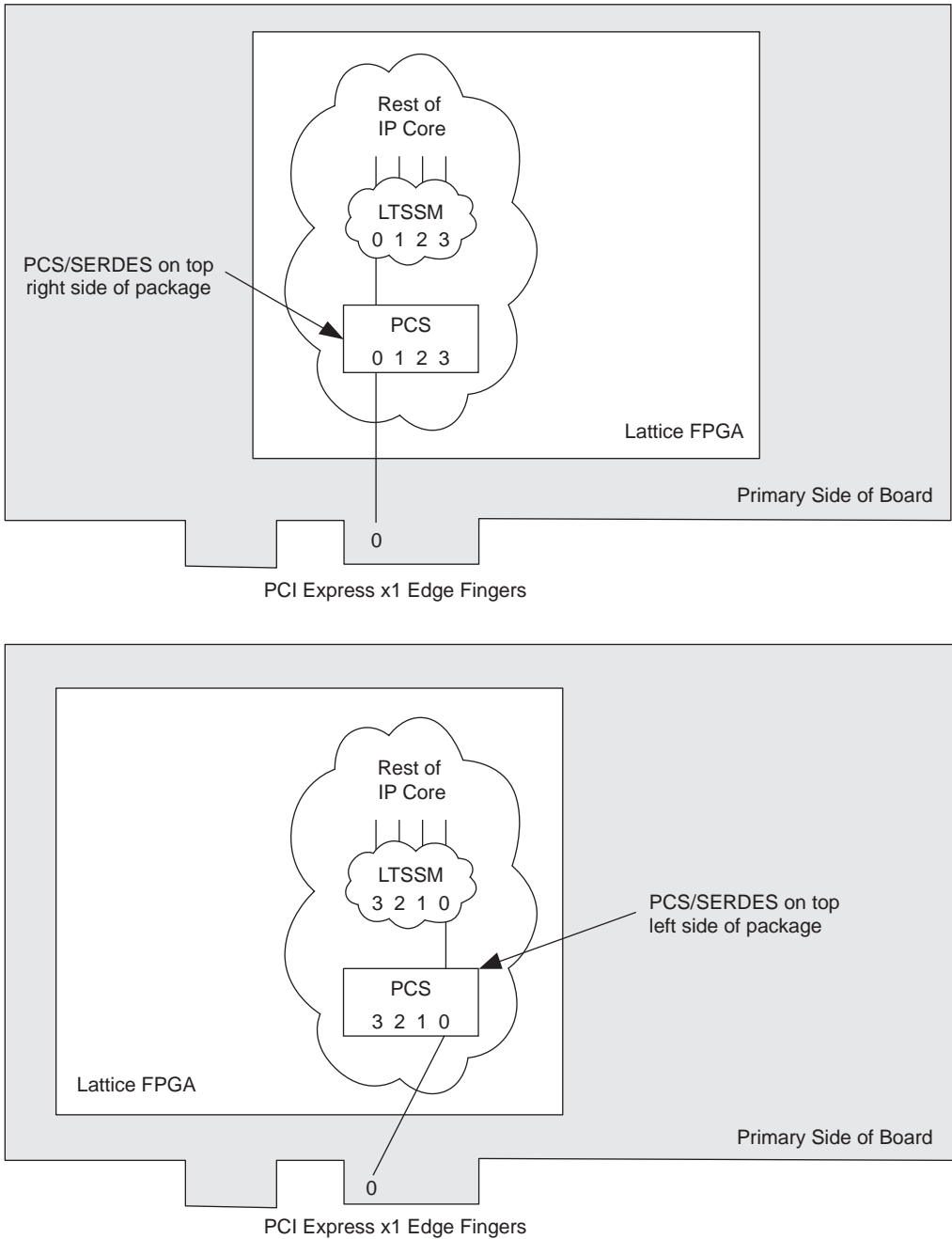
To accommodate this layout dilemma, the Lattice PCI Express solution provides an option to reverse the order of the SERDES lanes to the LTSSM block of the PCI Express core. This allows the board layout to connect edge finger lane 0 to SERDES lane 3, edge finger lane 1 to SERDES lane 2, edge finger lane 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 34 provides a diagram of a normal and a reversed IP core implementation.

Figure 34. Implementation of x4 IP Core to Edge Fingers

As shown in Figure 34, 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 35 provides a diagram of a x1 IP core to illustrate the recommended solution in the board layout.

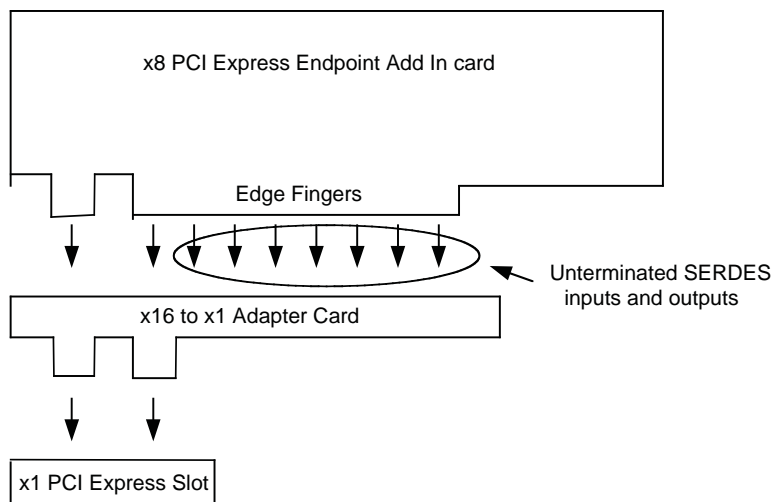
Figure 35. Implementation of x1 IP Core to Edge Fingers



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 36 provides the stack up of an endpoint add in card with an adapter card.

Figure 36. 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.

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 IPexpress.

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

How to Set Up for x4 (No Flip)

This is the default condition that is created from IPexpress. Simply use the autoconfig file to setup the channels. The flip_lanes port should be tied low.

How to Set Up for x4 (Flipped)

LatticeECP2M

No changes required. Simply use the pcs_pcie_8b_x4.txt file generated from IPexpress.

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 pcs_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.

How to Set Up for x1 DowngradedLatticeECP2M

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 IPexpress.

How to Set Up for x1 (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.

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 either the ispLEVER Design Planner 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.

LatticeECP2M

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

LatticeSCM

```
FREQUENCY NET "<instance name>/ul_flxmc_sys_pcie/sys_clk_250_inferred_clock"
250 MHz ;
FREQUENCY NET "<instance name>/ul_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.

LatticeECP2M

```
BLOCK PATH FROM CELL "*ctc_reset_chx*";
BLOCK NET "<instance name>/ul_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_skip_cnt*" TO CELL "*cnt_done_nfts_rx*" 2 X;
MULTICYCLE FROM CELL "*nfts_rx_skip_cnt*" TO CELL "*ltssm_nfts_rx_skip*" 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 125MHz. Based on the connectivity of the design the name of this clock net may 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>/ul_flxmc_sys_pcie/sys_clk_250_inferred_clock" TO CLKNET "sys_clk_125" ;

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

```

Errors and Warnings

During the process of running ispLEVER 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.

LatticeECP2M Errors and WarningsPlace & Route Design

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

```

WARNING - par: The driver of primary clock net
pcie/ul_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/ul_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/ul_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/ul_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/ul_flxmc_sys_pcie/ul_flxmc_pcie_core/ul_phy_dll/ul_flxmc_top_ebr/flxmc_top_mib will be checked during bitgen

WARNING - map: Security for MACO block
pcie/ul_flxmc_sys_pcie/ul_flxmc_pcie_core/ul_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/ul_flxmc_sys_pcie/ul_flxmc_pcie_core/ul_phy_dll/maco_pmi_distributed_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/ul_flxmc_sys_pcie/ul_flxmc_pcie_core/ul_phy_dll/maco_pmi_distributed_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/ul_flxmc_sys_pcie/ul_flxmc_pcie_core/ul_phy_dll/maco_pmi_distributed_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/ul_flxmc_sys_pcie/ul_flxmc_pcie_core/ul_phy_dll/maco_pmi_distributed_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.

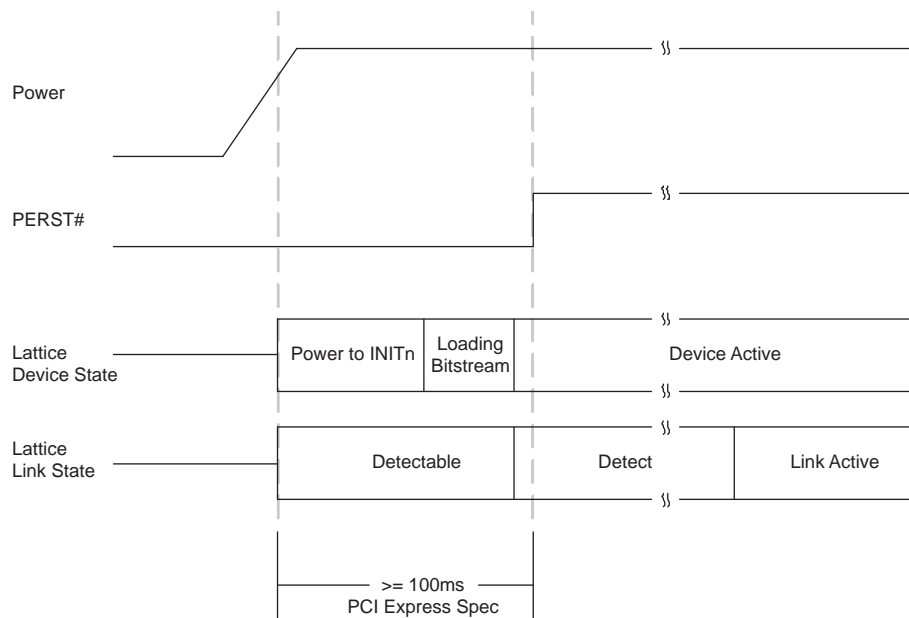
These warnings inform the user that a SLICE is programmed in DPRAM mode which allows a constant write to the RAM. This is an expected implementation of the RAM which is used in the PCI Express design.

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.

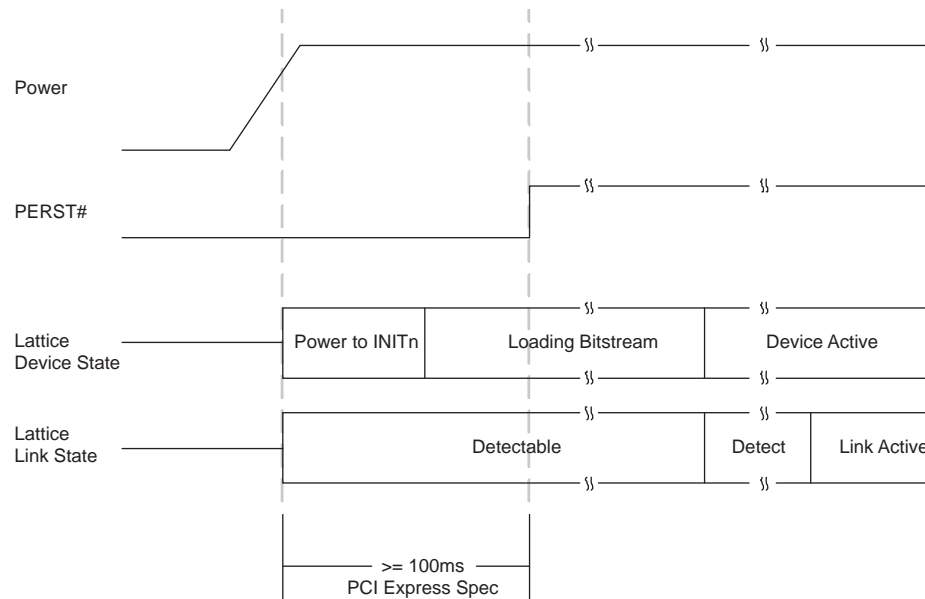
From the point of power stable to at least 100ms the PERST# must remain asserted. Different PCI Express systems will hold PERST# longer than 100ms, but the minimum time is 100ms. Shown below in Figure 37 is a best case timing diagram of the Lattice device with respect to PERST#.

Figure 37. 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 38 shows a worst case timing diagram of the Lattice device with respect to PERST#.

Figure 38. 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-ohm differential resistor on the receiver data lines when power is applied. This 100-ohm differential resistance will allow the device to be detected by the link partner. This state is shown 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, Tables 8 and 9 contain the relative numbers for the LatticeECP2M and LatticeSCM families.

Table 8. 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	5	8	15	17	22	ms

1. 8-bit wide Flash and external CPLD interfacing to LatticeECP2M at 150MHz SLAVE_PARALLEL mode.

Table 9. 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.

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 application note 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.

Troubleshooting

Table 10 provides some troubleshooting tips for the user when the core does not work as expected.

Table 10. 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 Appendix A 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 may 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 may 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.

References

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](#)

LatticeECP2M

- DS1006, [LatticeECP2M Family Data Sheet](#)
- TN1124, [Lattice SERDES/PCS Usage Guide](#)
- TN1108, [LatticeECP2M sysCONFIG Usage Guide](#)
- TN1114, [Electrical Recommendations for Lattice SERDES](#)

- TN1166, [PCI Express SIG Compliance Overview for Lattice Semiconductor FPGAs](#)
- AN8077, [Parallel Flash Programming and FPGA Configuration](#)

IPexpress/ispLeverCORE IP

- [IPexpress Quick Start Guide](#)
- [Lattice ispLeverCORE IP Tutorial](#)

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

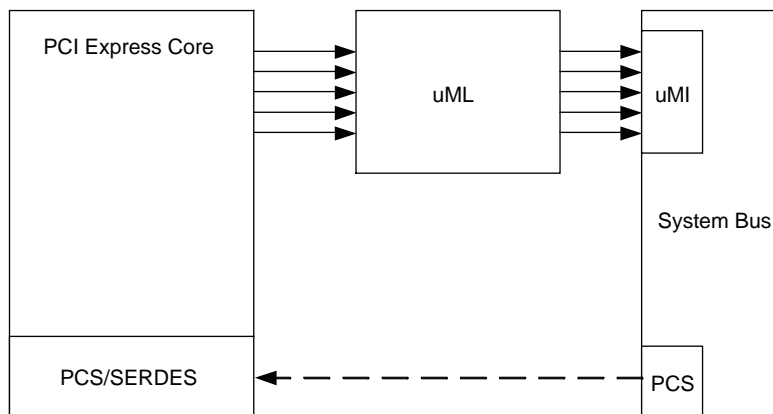
Date	Version	Change Summary
August 2008	01.0	Initial release.

Appendix A. LatticeSCM Reference Design for Multi-Lane Link

This appendix discusses how to use the LatticeSCM PCI Express IP core as a multi-lane link. A multi-lane link requires the LatticeSCM PCS MultiChannel Aligner (MCA) to align the lanes. The MCA requires that the user set the width of channels that need to be aligned and then issue a datapath reset. The uML module described in this appendix discusses how this is performed and provide sample code.

Figure 39 provides a block diagram of how the PCI Express IP core is enabled for multi-lane links using the embedded system bus and uML blocks.

Figure 39. Block Diagram of PCI Express IP core and uML



The theory behind the alignment solution is that during the DETECT and POLLING states of the LTSSM alignment is not required. Alignment is only required during the CONFIG state at which point the number of lanes to be configured has been identified. This information is passed to the uML block which via the system bus changes the MCA control register inside the PCS to the width requested and then issues a reset to the PCS receive path to perform the alignment.

This system also works in reverse. When the LTSSM falls back to the DETECT state the MCA control register will be reset back to a no-alignment state so that POLLING can reacquire the number of lanes in the link.

uML Module

Included in Table 11 is the port list of the uML module.

Table 11. uML Port List

Port Name	Direction	Clock	Description
rst_n	Input	Async	Async reset to reset the uML state machine
User Master Interface (UMI) to the System Bus			
umi_clk	Input		Main clock of the uML. This clock is the same clock used for the umi_clk of the system bus.
umi_ack	Input	umi_clk	System bus interface signals for the UMI
umi_retry	Input		
umi_err	Input		
umi_addr	Output		
umi_wdata[35:0]	Output		
umi_wr_n	Output		
umi_rdy	Output		
umi_lock	Output		
IP Core Status			
lsm_status[3:0]	Input	Async	Link State Machine result from the PCS.
ltssm_state[3:0]	Input	umi_clk	LTSSM state from the PCI Express core.
phy_realign_req	Input	umi_clk	Indicator from the PCI Express core that the LTSSM is in the CONFIG.Complete state. This kicks off the state machine to write the new registers based on the link width. Note: phy_realign_req is the output port name of the core.
phy_cfgln[3:0]	Input	umi_clk	Indicates which lanes of the PCI Express link are configured
IP Core Control			
rx_rst	Output	umi_clk	Active high reset sent to PCI Express core to reset rx datapath.
prog_done	Output	umi_clk	Active high signal sent to the PCI Express core to allow the LTSSM to proceed to the L0 state.

uML Description

The uML consists of two state machines. The first state machine controls when and what is written to the UMI interface. The second state machine actually performs the UMI writes.

There are two main event points of the main State Machine (SM). The first is when the LTSSM enters DETECT. At this point the module needs to reset all of the control registers back to a x1 setting. The second event point is when the LTSSM enters the Config.Complete state. At this event the module needs to determine the link width and set the control registers appropriately.

The UMI state machine (umi_sm) has one event point which is the assertion of umi_start. Once umi_start is asserted the umi_sm deasserts umi_done and looks for umi commands in the umi_cmds array. The umi_sm will look at the umi_cmds array until all commands have been written into the UMI.

The user can easily extend the abilities of the UMI block by increasing the number of states of the main state machine. Also the number of commands issued by the UMI for each state can be extended by increasing the size of the umi_cmds array.

Appendix B. LatticeSCM-15/40/80/115 Devices

Table 12. Performance and Resource Utilization¹

IPexpress User-Configurable Mode ²	Slices	LUTs	Registers	sysMEM™ EBRs	MACO Blocks	f _{MAX} (MHz)
Config 1 - Soft LTSSM	5678	8122	6064	13	1	250
Config 2 - Hard LTSSM	3811	4701	4696	13	2	250

1. Performance and utilization characteristics are generated using LFSC3GA80E-6FC1704C, with Lattice's ispLEVER 7.1 software. When using this IP core in a different density, speed, or grade within the LatticeSCM family, performance and utilization may vary. 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.

Configuration and Licensing

You can use the IPexpress software tool to help generate new configurations of this IP core. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and ispLEVER help system. For more information on the ispLEVER design tools, visit the Lattice website at www.latticesemi.com/software.

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. Larger FPGA devices will have more instances of MACO IP. Please refer to the Lattice web pages on LatticeSCM and MACO IP for more information.

Lattice SCM MACO IP cores are available free of charge through ispLEVER. However a license key is required to enable timing simulation and bitstream generation. Please contact your local Lattice sales office to obtain your MACO IP license key.

Appendix C. LatticeSCM-25 FPGAs

Table 13. Performance and Resource Utilization¹

IPexpress User-Configurable Mode ²	Slices	LUTs	Registers	sysMEM™ EBRs	MACO Blocks	f _{MAX} (MHz)
Config 1 - Soft LTSSM	5678	8122	6064	13	1	250

1. Performance and utilization characteristics are generated using LFSC3GA25E-6FF1020C, with Lattice's ispLEVER 7.1 software. When using this IP core in a different density, speed, or grade within the LatticeSCM family, performance and utilization may vary. 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.

Configuration and Licensing

You can use the IPexpress software tool to help generate new configurations of this IP core. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and ispLEVER help system. For more information on the ispLEVER design tools, visit the Lattice website at www.latticesemi.com/software.

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. Larger FPGA devices will have more instances of MACO IP. Please refer to the Lattice web pages on LatticeSCM and MACO IP for more information.

Lattice SCM MACO IP cores are available free of charge through ispLEVER. However a license key is required to enable timing simulation and bitstream generation. Please contact your local Lattice sales office to obtain your MACO IP license key.

Appendix D. Performance and Resource Utilization LatticeECP2M-50 FPGAs - PCI Express x1 Endpoint

Table 14. Performance and Resource Utilization¹

IPexpress User-Configurable Mode	Slices	LUTs	Registers	sysMEM EBRs	f _{MAX} (MHz)
Config 1	4139	6137	3898	4	125

1. Performance and utilization characteristics are generated using LFE2M-50E-6F900C, with Lattice's ispLEVER 7.1 software. When using this IP core in a different density, speed, or grade within the LatticeECP2M family, performance and utilization may vary. 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 x1 Endpoint IP core targeting LatticeECP2M devices is PCI-EXP1-PM-U3. Table 15 lists the standard configurations that are available for the core, however any configuration can be created.

You can use the IPexpress software tool to help generate new configurations of this IP core. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and ispLEVER help system. For more information on the ispLEVER design tools, visit the Lattice website at www.latticesemi.com/software.

Table 15. Parameter Settings for Config Demos

	Config Demo 1	Config Demo 4
Maximum TLP Size	512 Bytes	512 Bytes
Number of Virtual Channels	1	1
Low Priority Extended Virtual Channels	0	0
Enable CRC	No	Yes
Enable AER	No	Yes
Enable MSI	Yes	Yes
Wishbone Interface	No	No
Enable BAR 0	Yes	Yes
Enable BAR 1	Yes	Yes
Enable BAR 2	No	Yes
Enable BAR 3	No	Yes
Enable BAR 4	No	Yes
Enable BAR5	No	Yes
Enable Expansion ROM BAR	No	No

Appendix E. Performance and Resource Utilization LatticeECP2M-50 FPGAs - PCI Express x4 Endpoint

Table 16. Performance and Resource Utilization¹

IPexpress User-Configurable Mode	Slices	LUTs	Registers	sysMEM EBRs	f _{MAX} (MHz)
Config 1	9105	12100	9688	11	125

1. Performance and utilization characteristics are generated using LFE2M-50E-6F900C, with Lattice's ispLEVER 7.1 software. When using this IP core in a different density, speed, or grade within the LatticeECP2M family, performance and utilization may vary. 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 17 lists the standard configurations that are available for the core, however any configuration can be created.

You can use the IPexpress software tool to help generate new configurations of this IP core. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and ispLEVER help system. For more information on the ispLEVER design tools, visit the Lattice website at www.latticesemi.com/software.

Table 17. Parameter Settings for Config Demos

	Config Demo 1	Config Demo 2
Maximum TLP Size	512 Bytes	512 Bytes
Number of Virtual Channels	1	1
Low Priority Extended Virtual Channels	0	0
Enable CRC	No	No
Enable AER	No	Yes
Enable MSI	Yes	Yes
Wishbone Interface	No	No
Enable BAR 0	Yes	Yes
Enable BAR 1	Yes	Yes
Enable BAR 2	No	No
Enable BAR 3	No	No
Enable BAR 4	No	No
Enable BAR5	No	No
Enable Expansion ROM BAR	No	No

Appendix F. LatticeECP2M-35 FPGAs

Table 18. Performance and Resource Utilization¹

IPexpress User-Configurable Mode ²	Slices	LUT4s	Registers	EBRs	f _{MAX} (MHz)
Config Demo - no AER, no ECRC, 512 byte max TLP, Polling Compliance	4071	5865	3968	4	250

1. Performance and utilization characteristics are generated using LFE2M-35E672C with Lattice's ispLEVER 7.1 software. When using this IP core in a different density, speed, or grade within the LatticeECP2M family, performance and utilization may vary.

Table 19. Config Demo IP Core Features

Config Demo IP Core Feature	Value
Maximum TLP Size	512 Bytes
Number of Virtual Channel 1	1
Low Priority Extended Virtual Channels	0
Enable CRC	No
Enable AER	No
Wishbone Interface	No
Polling Compliance	Yes
Enable BAR 0	Yes (8K)
Enable BAR 1	Yes (32K)
Enable BAR 2	No
Enable BAR 3	No
Enable BAR 4	No
Enable BAR5	No
Enable Expansion ROM BAR	No

Configuration and Licensing

You can use the IPexpress software tool to help generate new configurations of this IP core. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and ispLEVER help system. For more information on the ispLEVER design tools, visit the Lattice website at www.latticesemi.com/software.