



## LatticeECP2M 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 IP core provides x4 and x1 endpoint solutions from the electrical SERDES interface to the transaction layer. The LatticeECP2M™ PCI Express IP core utilizes the PCS/SERDES built into the LatticeECP2M devices to support the physical layer.

## Features

The Lattice PCI Express IP core supports the following features.

### PHY Layer Features

- 2.5Gbps CML Electrical Interface
- PCI Express 1.1 Electrical Compliant
- Many Options for Signal Integrity Including Differential Output Voltage, Transmit Pre-emphasis and Receiver Equalization
- Serialization and De-serialization
- 8b/10b 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

### 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 Management 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 Core (0x0-0x3C)
- PCI Express Capability Structure Registers Contained Inside 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
- Extended Capability register for virtual channel support contained inside core
- Remaining Configuration Requests can optionally be terminated by the core or passed to the user to support other Capability Structures or support new ECN registers

**Top Level IP Support**

- 100 MHz reference clock input
- 125 MHz 64-Bit Data Path User Interface for both x1 and x4
- 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 Single Function Endpoint Topology
- Higher Layer Control of LTSSM via Ports
- Access to Select Configuration Space Information via Ports
- Supported in -6 LatticeECP2M Speed Grades

**Getting Started**

This document provides details on how to use the Lattice PCI Express IP core. It does not include all of the information found in the PCI-SIG PCI Express 1.1 specification. It is recommended that the reader is familiar with the PCI-SIG PCI Express 1.1 specification to understand the framework in which this core is to be 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 customizing cores.

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 on this specific core see the LatticeECP2M PCI Express ReadMe document. This file is available once the core is installed in ispLEVER. This document provides information on creating an evaluation version of the core for use in ispLEVER and simulation.

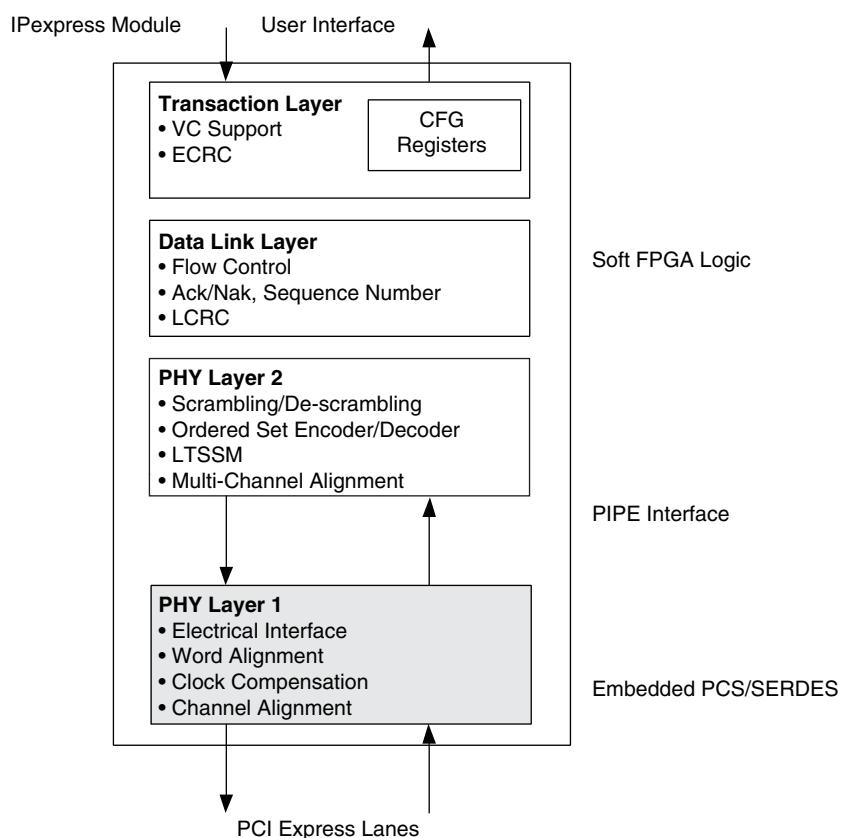
## Description

The LatticeECP2M PCI Express IP core is implemented in different FPGA technologies. These technologies include soft FPGA fabric elements such as LUTs, registers, and embedded block RAMs (EBRs). Embedded hard elements within the LatticeECP2M PCS/SERDES are also utilized. The IP interfaces to the Embedded PCS/SERDES using the PIPE interface.

The ispLEVER design tool IPexpress is used to customize and create the complete IP module for the user to instantiate in their 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 the 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 an implementation representation of a LFE2Mxx device with the PCI Express core.

## Block Diagram

Figure 2 provides a high level interface representation.

**Figure 2. PCI Express Interfaces**

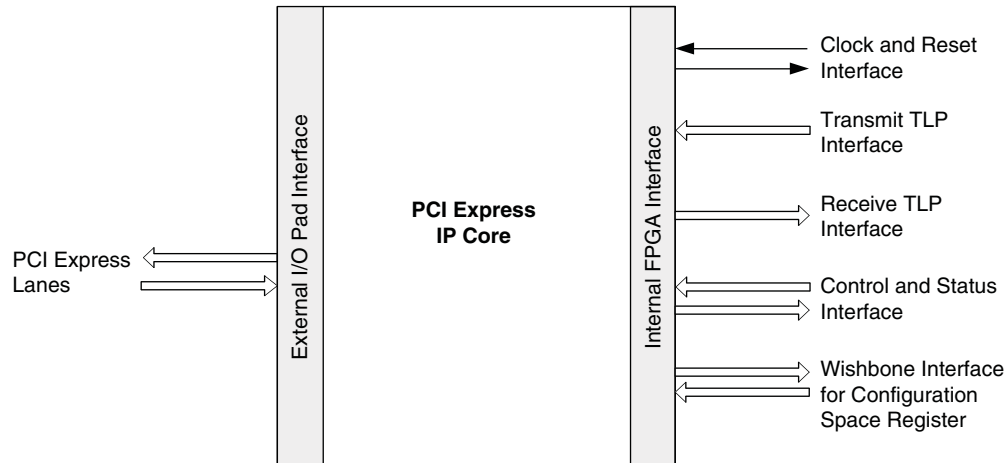


Table 1 provides the list of ports and descriptions for the PCI Express IP core.

**Table 1. PCI Express Port List**

Port Name	Direction	Clock	Description
<b>Clock and Reset Interface</b>			
refclk[n,p]	Input		100MHz differential reference clock to the SERDES PLL for generation of the sys_clk_250 clock.
sys_clk_125	Output		125MHz clock derived from sys_clk_250 to be used in the user application.
rst_n	Input		Active-low asynchronous datapath and state machine reset.
sys_clk_250	Input		250 MHz clock used by the PIPE interface logic and portions of the Physical layer.
<b>PCI Express Lanes</b>			
hdin[p,n][0,1,2,3]	Input		PCI Express 2.5Gbps CML inputs for lanes 0,1,2, and 3.
hdout[p,n][0,1,2,3]	Output		PCI Express 2.5Gbps CML outputs for lanes 0,1,2, and 3.
<b>Transmit TLP Interface</b>			
tx_data_vc0[63:0]	Input	sys_clk_125	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
tx_req_vc0	Input	sys_clk_125	Active high transmit request. This port should be asserted when the user wants to send a TLP. If several TLPs will be provided in a burst this port can remain high until all TLPs have been sent.
tx_rdy_vc0	Output	sys_clk_125	Active high transmit ready indicator. Tx_st should be provided next clock cycle after tx_rdy is high. This port will go low between TLPs.
tx_st_vc0	Input	sys_clk_125	Active high transmit start of TLP indicator

**Table 1. PCI Express Port List (Continued)**

Port Name	Direction	Clock	Description
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.
tx_dwen_vc0	Input	sys_clk_125	Active high transmit 32-bit word indicator. Used if only bits [63:32] provide valid data.
tx_val	Output	sys_clk_125	Active high transmit clock enable. When a x4 is down graded to a x2 or x1 this signal is used as the clock enable to downshift the transmit bandwidth.
tx_ca_[ph,nph,cplh]_vc0[8:0]	Output	sys_clk_125	[7:0] - Transmit Interface credit available bus. This port will decrement as TLPs are sent and may 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 far end receiver has infinite credits. If this bit is high then bits [7:0] should be ignored.
tx_ca_[pd,npd,cpld]_vc0[12:0]	Output	sys_clk_125	[7:0] - Transmit Interface credit available bus. This port will decrement as TLPs are sent and may increment as UpdateFCs are received. DLLP is received from the PCI Express line. pd - posted data npd - non-posted data cpld - completion data [12] - This bit indicates the far end receiver has infinite credits. If this bit is high, then bits [11:0] should be ignored.
tx_ca_p_recheck_vc0	Output	sys_clk_125	Active high signal that indicates the core sent an internally generated Posted TLP which changed the tx_ca_p[h,d]_vc0 port. This may require a recheck of the credits available if the user was about to send a Posted TLP.
tx_ca_cpl_recheck_vc0	Output	sys_clk_125	Active high signal that indicates the core sent an internally generated Completion TLP which changed the tx_ca_cpl[h,d]_vc0 port. This may require a recheck of the credits available if the user was about to send a Completion TLP.
<b>Receive TLP Interface</b>			
rx_data_vc0[63:0]	Output	sys_clk_125	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
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.
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.

**Table 1. PCI Express Port List (Continued)**

Port Name	Direction	Clock	Description
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, nph,npd,cplh, cpld] _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 UPDATE_FREQ.
[ph,nph,cplh]_processed_vc0	Input	sys_clk_125	Active high indicator to inform the IP core of how many header credits (Posted/Non-Posted/Completion) have been processed. Each clock cycle high counts as one credit processed. The core will generate the required UpdateFC DLLP using this information.
[pd,npd,cpld]_processed_vc0	Input	sys_clk_125	Active high indicator to inform the IP core of how many data credits (Posted/Non-Posted/Completion) have been processed. Each clock cycle high counts as one credit processed. The core will generate the required UpdateFC DLLP using this information.
<b>Wishbone Interface Enabled</b>			
CLK_I	Input		100MHz clk from wishbone
RST_I	Input	Async	Reset signal from wishbone
SEL_I [3:0]	Input	CLKI	Bank selection to wishbone
WE_I	Input	CLKI	Write(1)/read(0) to wishbone
STB_I	Input	CLKI	Strobe signal to wishbone
CYC_I	Input	CLKI	Cycle signal to wishbone
DAT_I[31:0]	Input	CLKI	Write data to wishbone
ADR_I[12:0]	Input	CLKI	Address to wishbone
CHAIN_RDAT_in[31:0]	Input	CLKI	Daisy chain data_in from previous slave
CHAIN_ACK_in	Input	CLKI	Daisy chain ack_in from previous slave
ACK_O	Output	CLKI	Ack signal from wishbone
IRQ_O	Output	CLKI	Interrupt ack signal from wishbone
DAT_O[31:0]	Output	CLKI	Read data from wishbone

**Table 1. PCI Express Port List (Continued)**

Port Name	Direction	Clock	Description
<b>Wishbone Interface Disabled</b>			
no_pcie_train	Input	Static	Active high signal disables LTSSM training and forces the LTSSM to L0. 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 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.
force_disable_scr	Input	Async	Disables the scrambler
hl_snd_beacon	Input	sys_clk_125	Active high request to send a Beacon when the LTSSM is in L2 state.
hl_disable_scr	Input	Static	Active high to set the disable scrambling bit in the TS1/TS2 sequence. This signal is typically used for debug purposes.
hl_gto_dis	Input	Static	Must be tied to '0'.
hl_gto_det	Input	sys_clk_125	Active high request to go to Detect state when LTSSM is in Disable.
hl_gto_hrst	Input	Static	Must be tied to '0'.
hl_gto_l0stx	Input	sys_clk_125	Active high request to go to Tx L0s when LTSSM is in L0.
hl_gto_l0stxfts	Input	sys_clk_125	Active high request to transmit FTS when LTSSM is in Tx 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 or L1.
hl_gto_cfg	Input	sys_clk_125	Active high request to go to Config when LTSSM is in Recovery. This is for the purpose of changing the link width and is typically handled by the Root Complex.
tx_dllp_val	Input	sys_clk_125	Transmit user DLLP command. Asserted for 1 clock cycle based on the encoding below. 00 - Nothing to send 01 - Send power management message DLLP defined by tx_pmttype 10 - Send vendor-specified DLLP defined by tx_vsd_data 11 - Not used
tx_pmttype[2:0]	Input	sys_clk_125	Power Management DLLP Message. Assert with the same timing as tx_dllp_val. 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 specified data to send in DLLP.
tx_dllp_sent	Output	sys_clk_125	Requested DLLP was sent.
rxdp_pmd_type[2:0]	Output	sys_clk_125	Receive power management DLLP 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-specified DLLP data. Assert with the same timing as tx_dllp_val for sending Vendor Specified DLLP.
rxdp_dllp_val	Output	sys_clk_125	Active high power management DLLP message received



**Table 1. PCI Express Port List (Continued)**

Port Name	Direction	Clock	Description
<b>PHY Layer Status</b>			
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
phy_ltssm_substate[2:0]	Output	sys_clk_125	Phy Layer LTSSM current substate. Each major LTSSM state has a series of substates.
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.
<b>Master Loopback Interface</b>			
tx_lbk_rdy	Output	sys_clk_125	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.
tx_lbk_kcntl[3:0]	Input	sys_clk_125	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_125	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_125	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]

**Table 1. PCI Express Port List (Continued)**

Port Name	Direction	Clock	Description
rx_lbk_data[31:0]	Output	sys_clk_125	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
<b>DATA LINK LAYER</b>			
dl_inactive	Output	sys_clk_125	Data Link Layer is the DL_Inactive state.
dl_init	Output	sys_clk_125	Data Link Layer is in the DL_Init state.
dl_active	Output	sys_clk_125	Data Link Layer is in the DL_Active state.
dl_up	Output	sys_clk_125	Data Link Layer is in the DL_Active state and is now providing TLPs to the Transaction Layer.
<b>TRANSACTION LAYER</b>			
ecrc_gen_enb	Input	Static	Active high ECRC generation enable. Only available if ECRC is enabled in IPexpress.
ecrc_chk_enb	Input	Static	Active high ECRC checking enable. Only available if ECRC is enabled in IPexpress.
cmpln_tout	Input	sys_clk_125	Completion Timeout Indicator. Can be pulsed high for 1 clock cycle to force the core to generate and send a non-fatal error message and also set appropriate bit in AER.
cmpltr_abort	Input	sys_clk_125	Complete or Abort Indicator. Can be pulsed high for 1 clock cycle to force the core to generate and send a non-fatal error message and also set appropriate bit in AER.
unexp_cmpln	Input	sys_clk_125	Unexpected Completion Indicator. Can be pulsed high for 1 clock cycle to force the core to generate and send a non-fatal error message and also set appropriate bit in AER.
np_req_pend	Input	sys_clk_125	Sets device Status[5] to indicate pending Non-Posted requests by the device. Should be clear when all outstanding requests are completed either by timeout or completions from the Root complex.
err_tlp_header[127:0]	Input	Static	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.
<b>CONFIGURATION REGISTERS</b>			
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	Legacy INTx interrupt request. Falling edge will produce a INTx_ASSERT message and set the Interrupt Status bit to a 1. Rising edge will produce a INTx_DEASSERT 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.
msi[7:0]	Input	sys_clk_125	MSI interrupt request. Rising edge on a bit will produce a MemWr TLP for a MSI interrupt for the provided address and data by the root complex. [7] - MSI 8 [6] - MSI 7 [5] - MSI 6 [4] - MSI 5 [3] - MSI 4 [2] - MSI 3 [1] - MSI 2 [0] - MSI 1
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	PowerState in the Power Management Capability Structure. Software sets this state to place the endpoint in a particular state. 00 - D0 01 - D1 10 - D2 11 - D3

## 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 of the far end before sending the TLP. This information is found on the tx\_ca\_[ph,pd,nph,npd,cplh,cpld]\_vc0 bus. There must be enough credits available for the entire TLP to be sent. It takes the IP core 3 clocks from tx\_st assertion to update the tx\_ca\_[ph,pd,nph,npd,cplh,cpld] signals.

The user then needs to 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 user must check available credit again. If there is enough credit, user can proceed with the sending data based on tx\_rdy\_vc0. If the credit becomes not sufficient, 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 should 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.

Figure 3. through Figure 8 provides timing diagrams for the tx interface signals. These timing diagrams are representative for a core configured as a x4. Figure 9 provides a timing diagram of a downgraded x1 link. Figure 10 provides a timing diagram of a Posted request sequence with tx\_ca\_p\_recheck asserted due to the IP core internally generating error message.

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

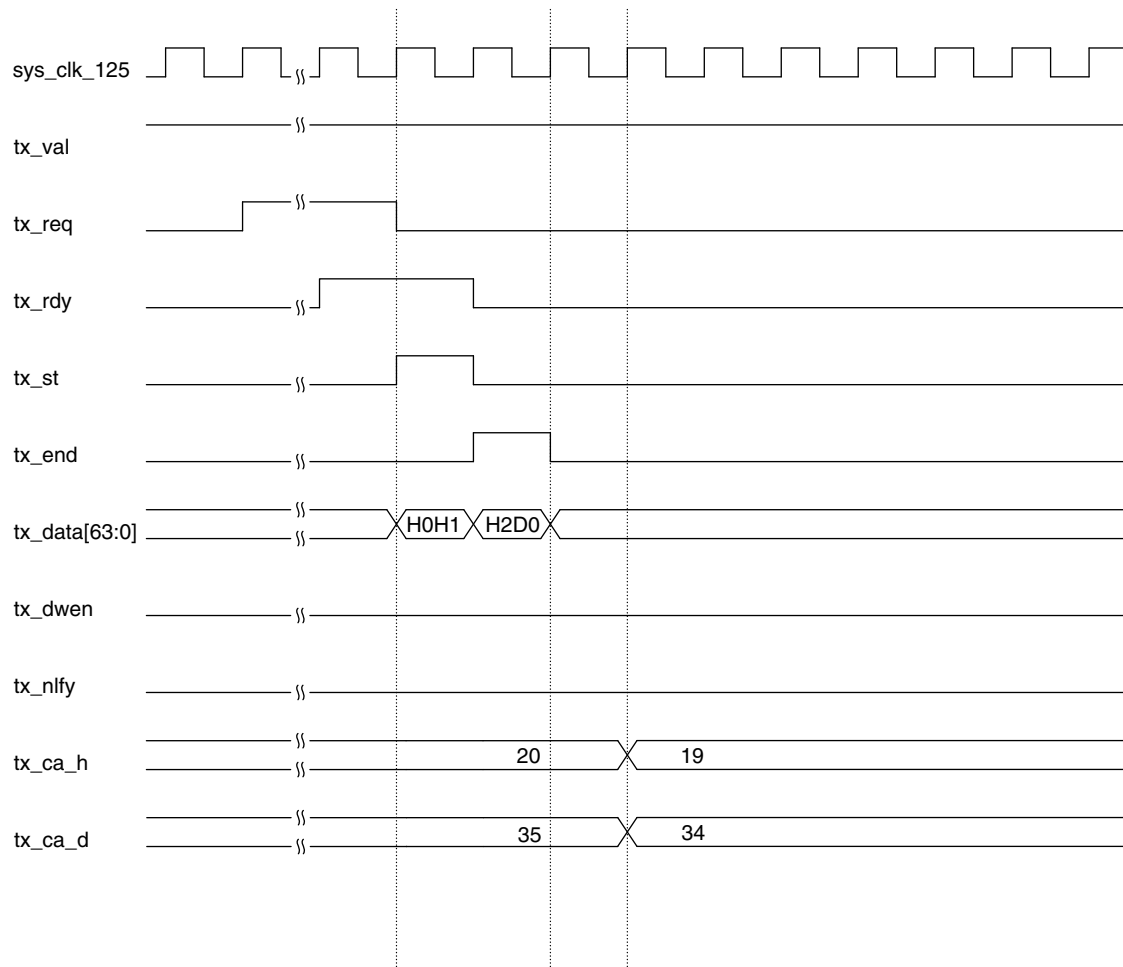


Figure 4. Transmit Interface x4, 3DW Header, 2 DW Data

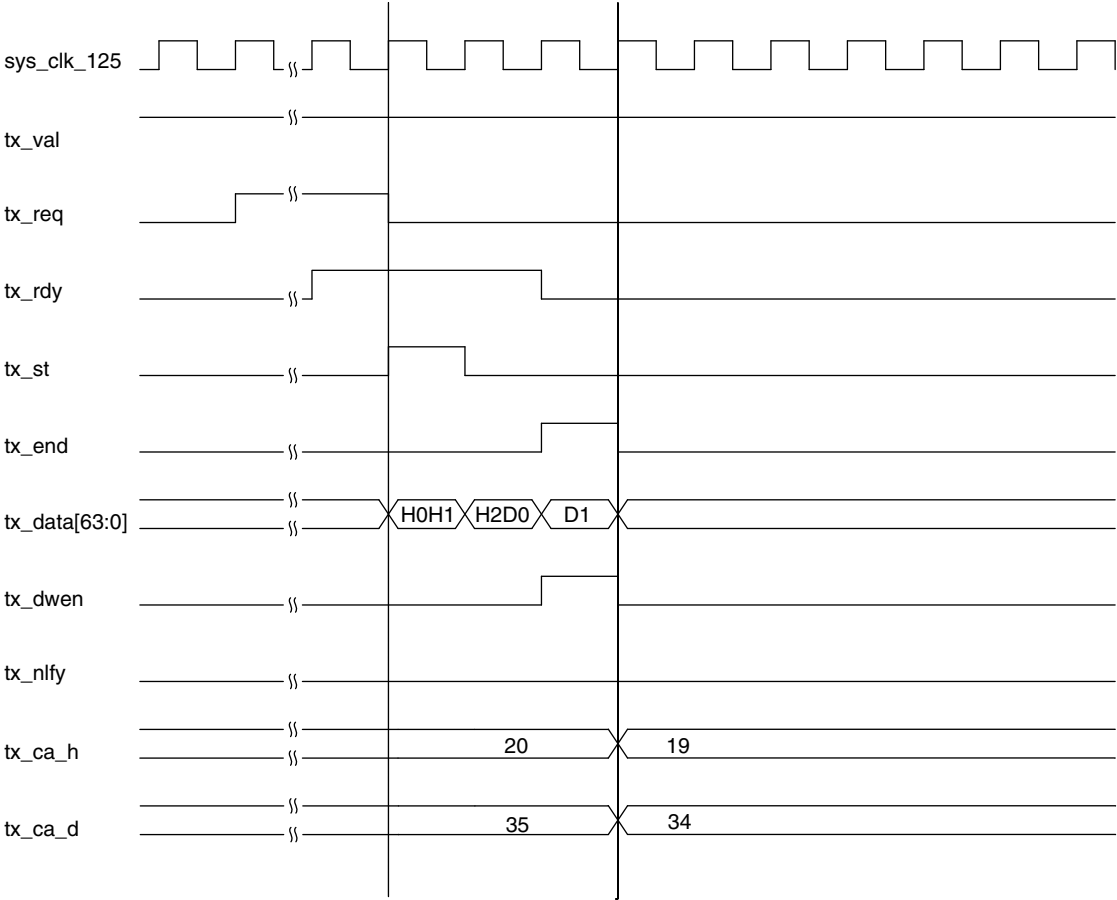


Figure 5. Transmit Interface x4, 4DW Header, 0 DW

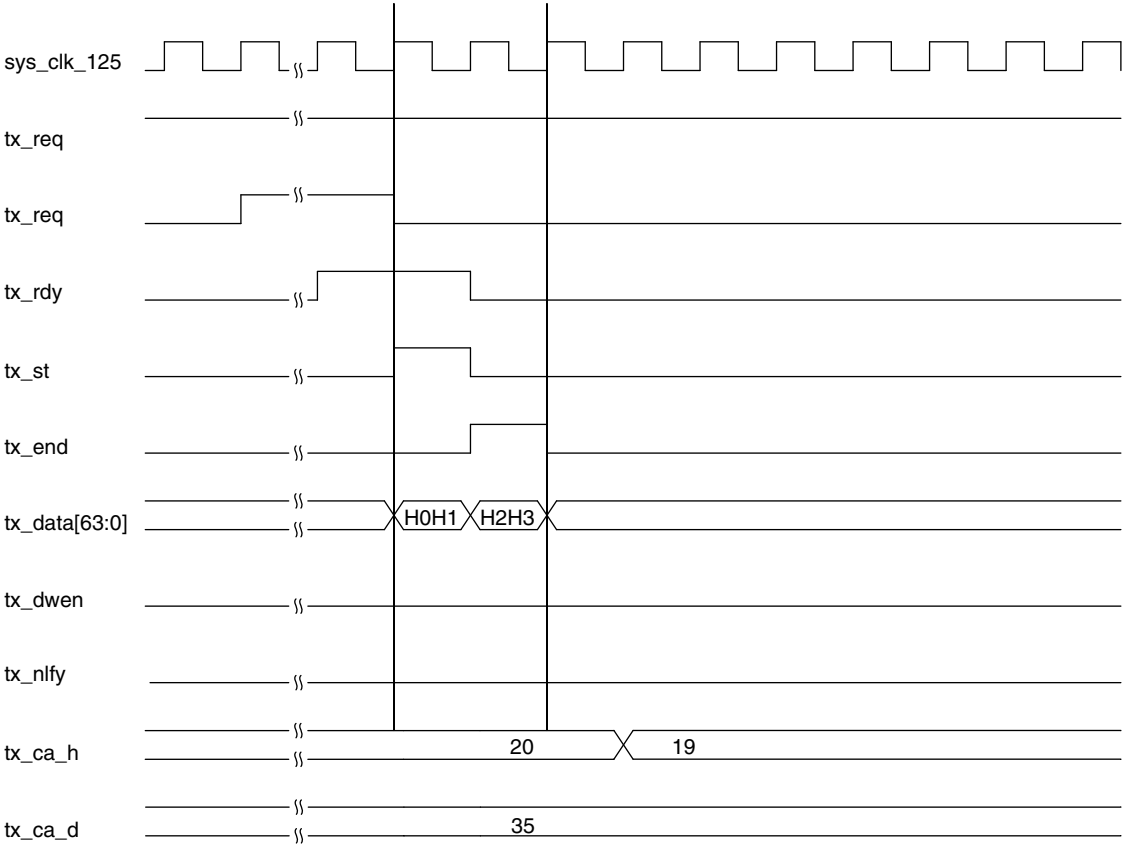
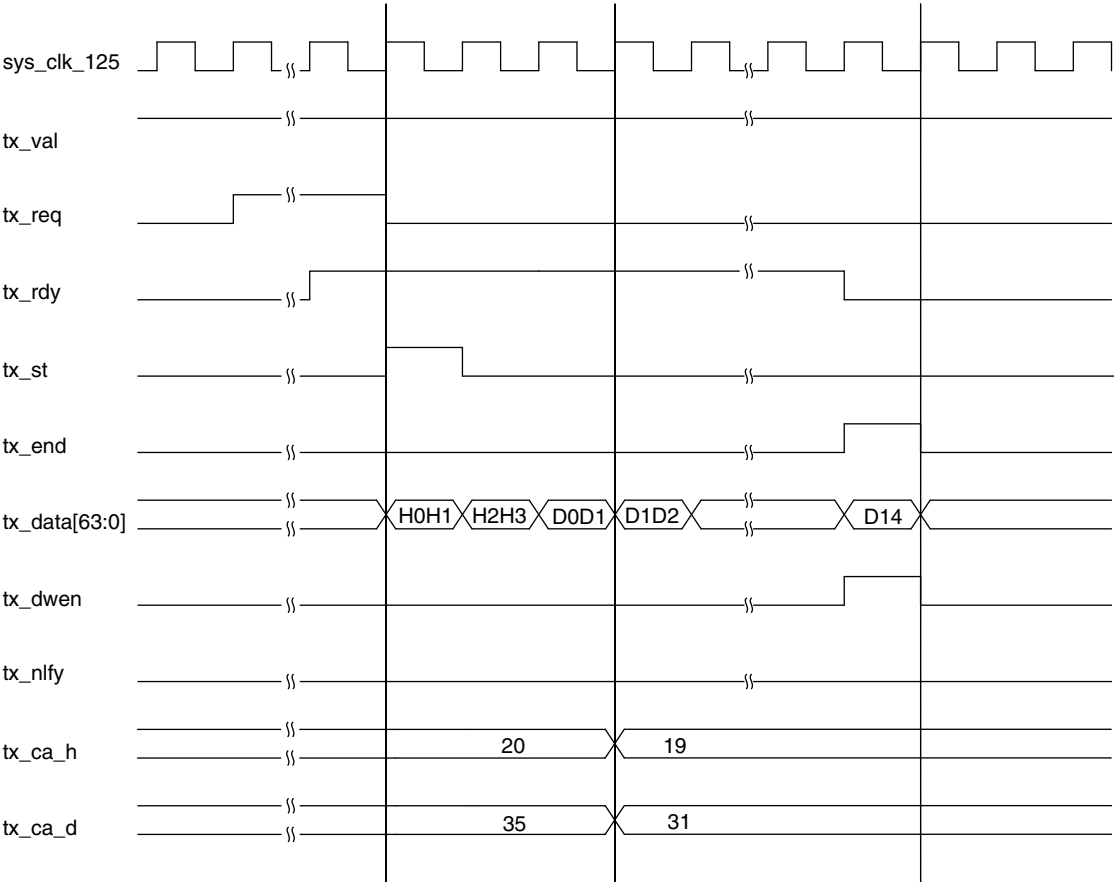


Figure 6. Transmit Interface x4, 4DW Header, Odd Number of DWs



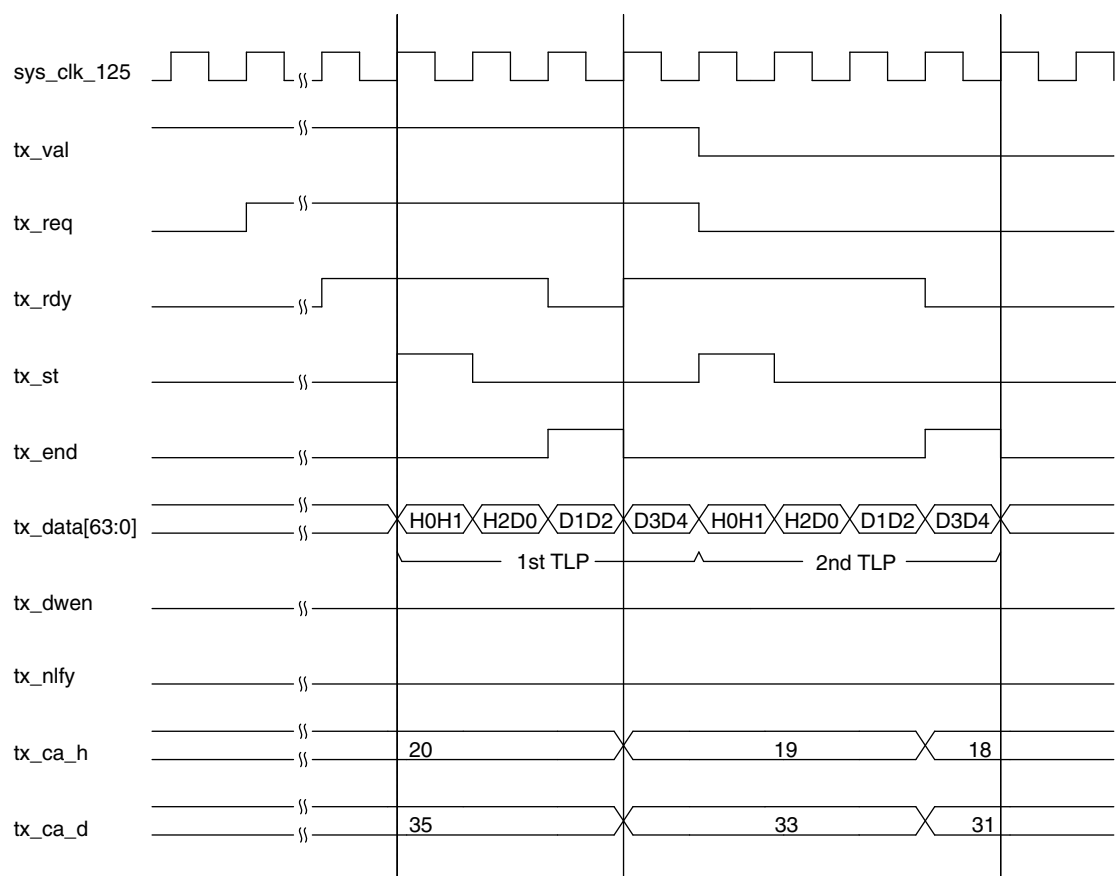
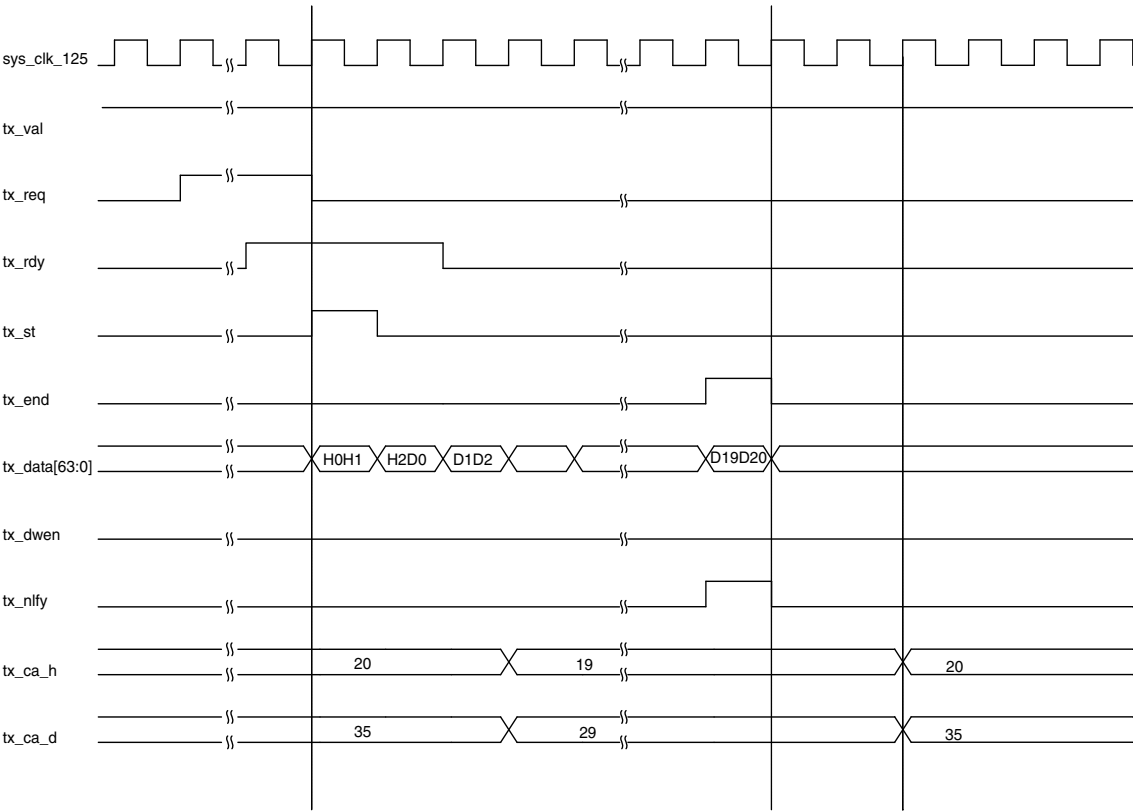
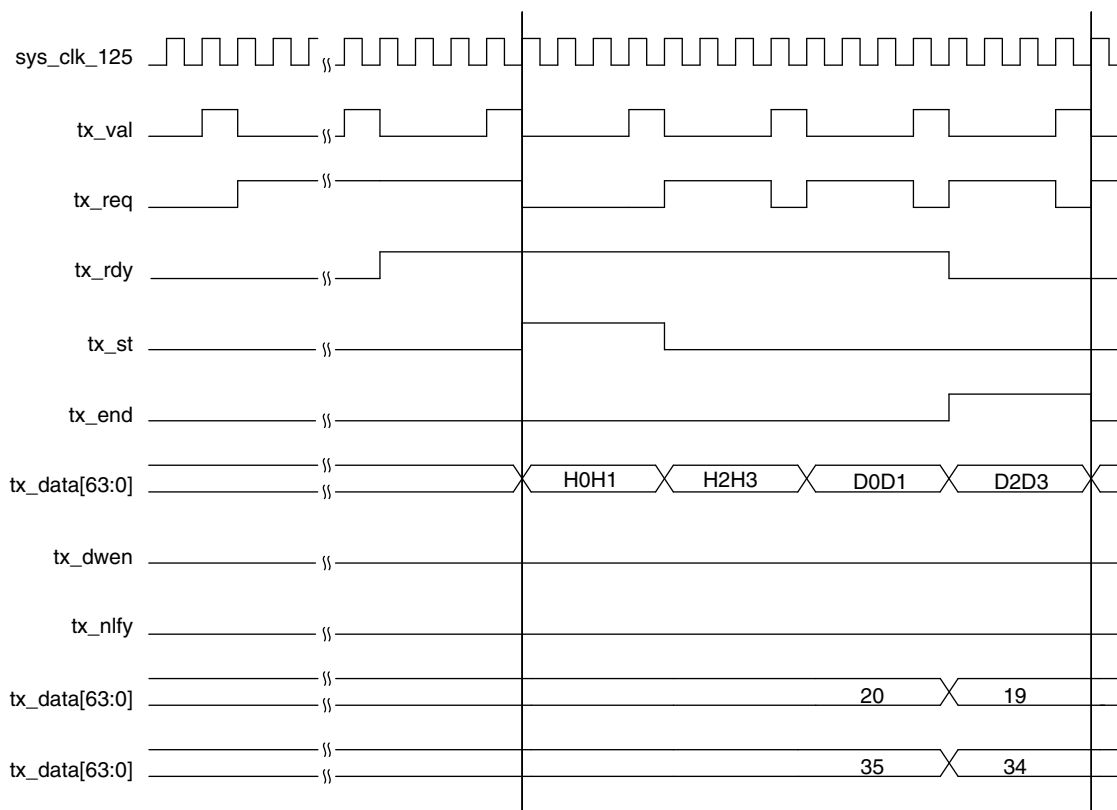
**Figure 7. Transmit Interface x4, Burst of 2 TLPs**



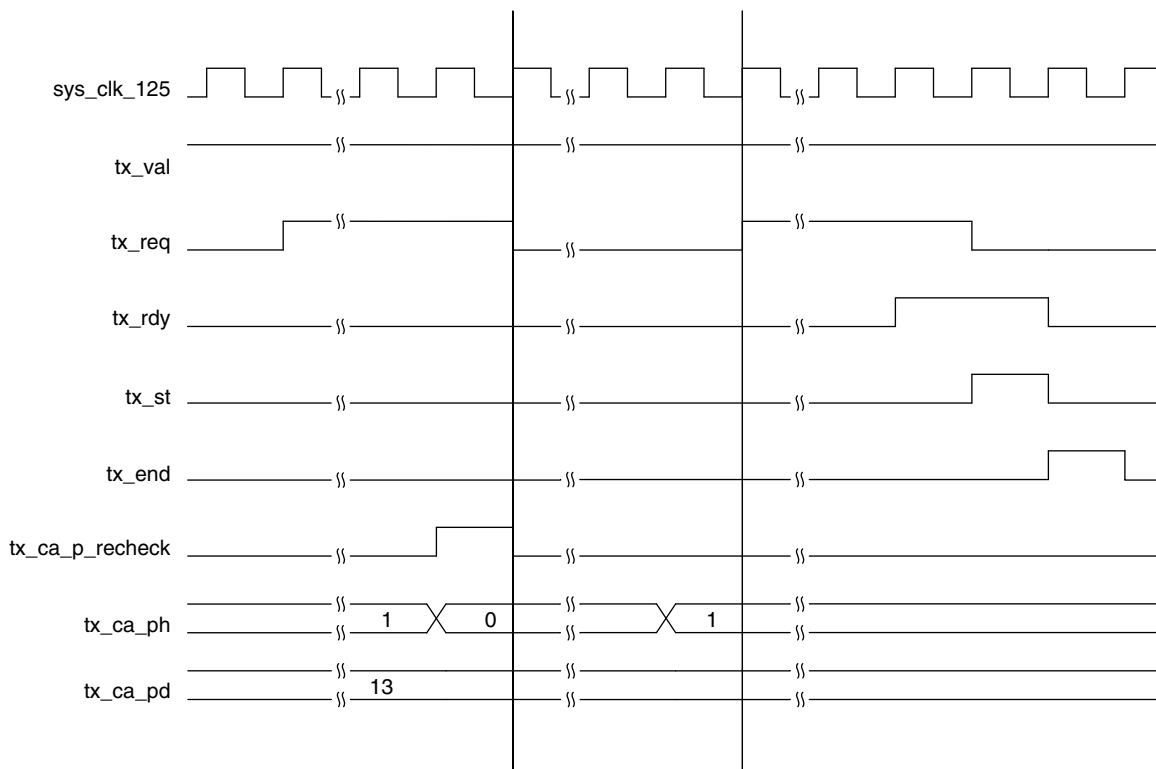
Figure 8. Transmit Interface x4, Nullified TLP



**Figure 9. Transmit Interface x1 Downgrade using tx\_val**



**Figure 10. Transmit Interface x4 Posted Request with tx\_ca\_p-recheck Assertion**



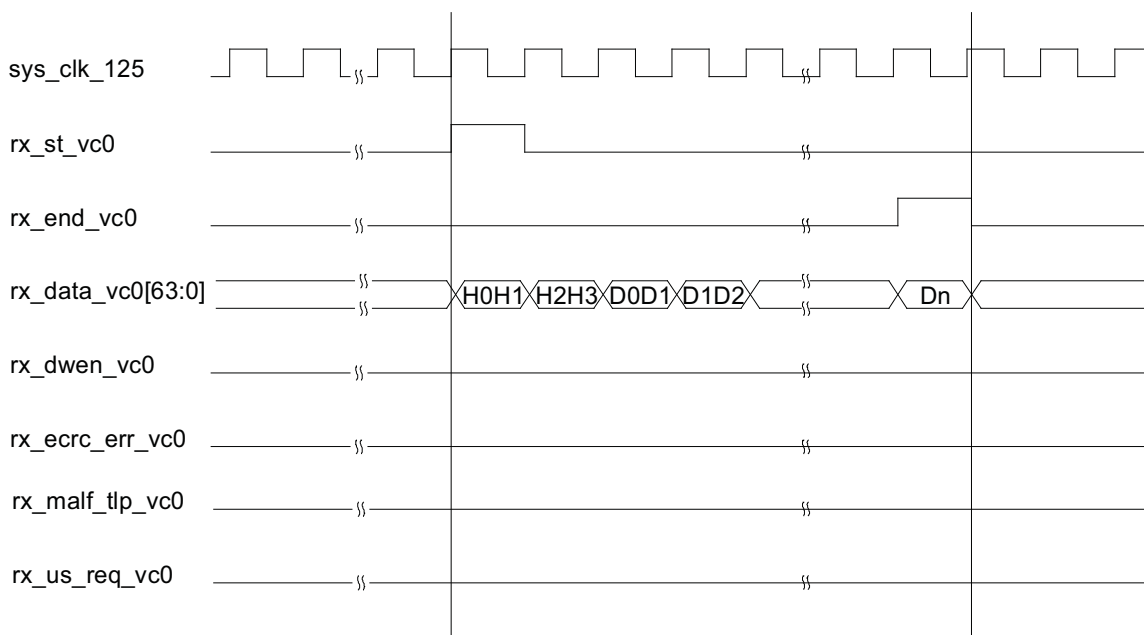
## Receive TLP Interface

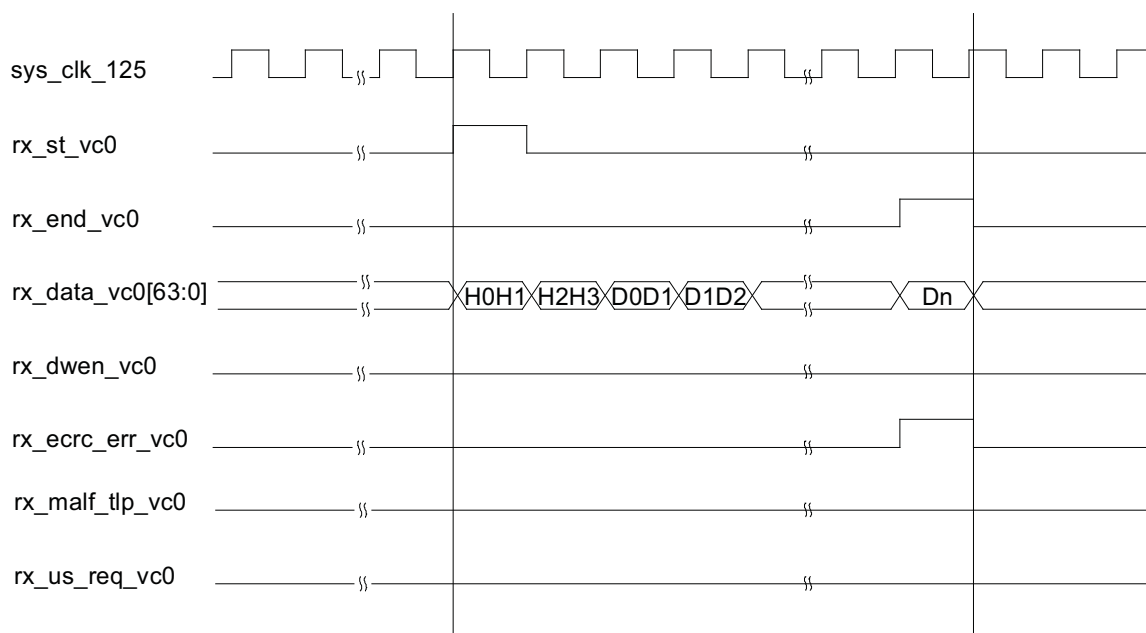
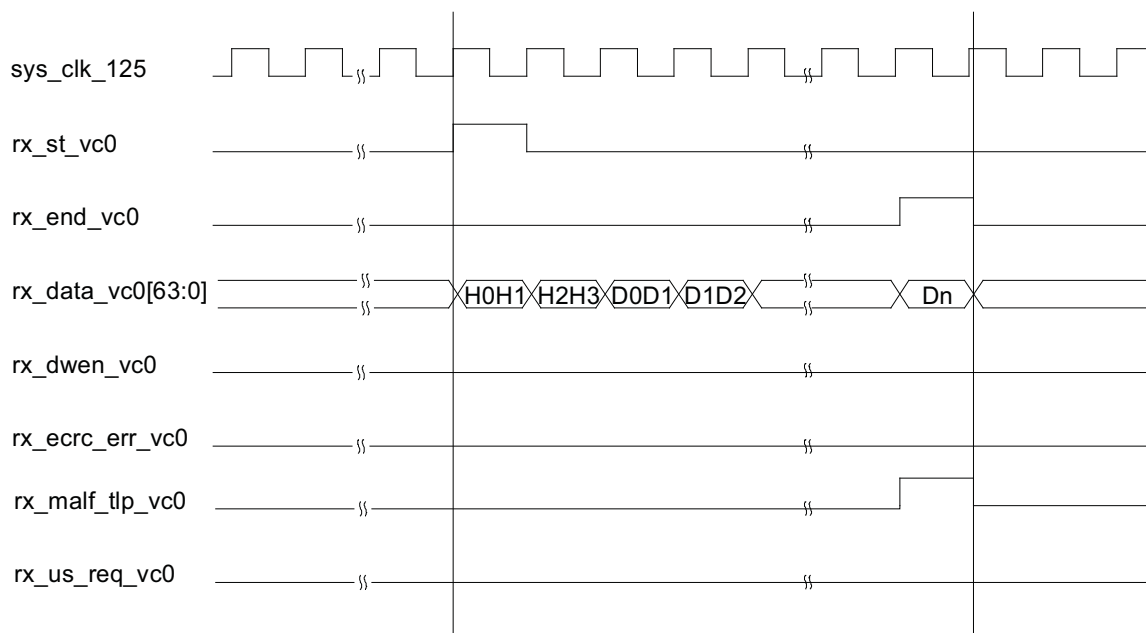
In the receive direction TLPs will come from the core as they are received on the PCI Express lanes. Config reads and config write TLPs to registers inside the core will be terminated inside the core. All other TLPs will be provided to the user. Also, if the core enables any of the BARs the TLP will 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.

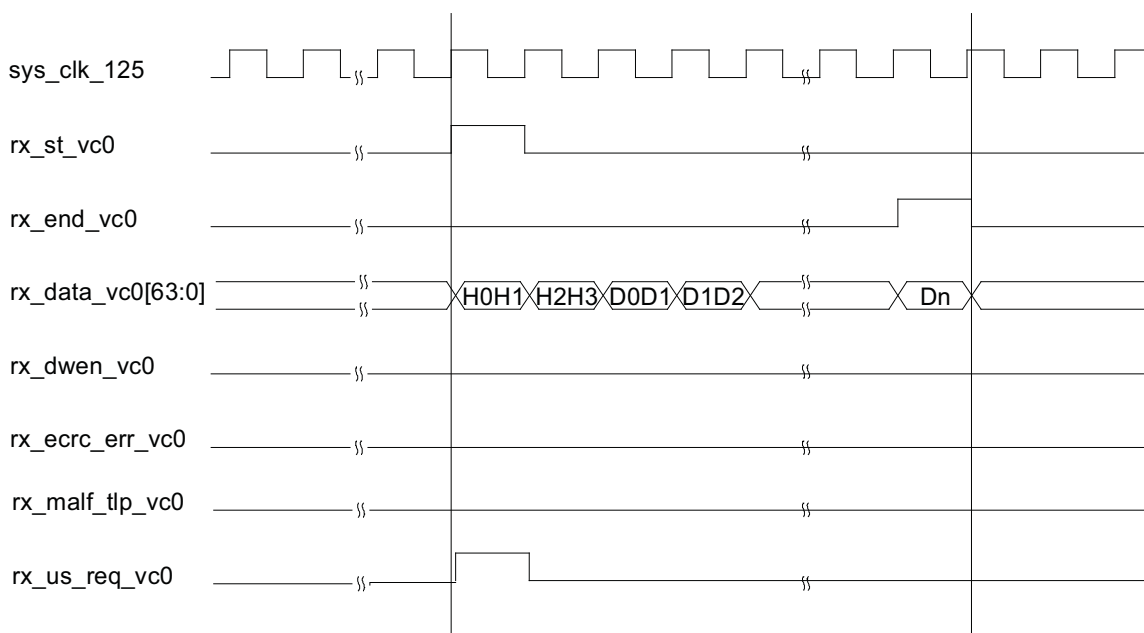
When a TLP is sent to the user the rx\_st\_vc0 signal will be asserted with the first 64-bit word of the TLP. The remaining TLP data will be provided on consecutive clock cycles until the last word with rx\_end\_vc0 asserted. If the TLP contains a ECRC error the rx\_ecrc\_err\_vc0 signal will be asserted at the end of the TLP. If the TLP has a length problem the rx\_malf\_tlp\_vc0 will be asserted at any time during the TLP. Figure 11 through Figure 14 provides timing diagrams of the receive interface.

The receive path does not utilize the same clock enable scheme as the transmit path. 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 11. Receive Interface, Clean TLP**



**Figure 12. Receive Interface, ECRC Errored TLP****Figure 13. Receive Interface, Malformed TLP**

**Figure 14. Receive Interface, Unsupported Request TLP**

## Using the Transmit and Receive Interface

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 8 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, and pd\_processed\_vc0 ports. Each MWr 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 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. Each cycle that p[d/h]\_processed\_vc0 is held high releases 1 posted data/header credit to the far end.

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 CplID 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. This read completion boundary forces the size of a total read data to be broken up into 64 or 128 byte CplD TLPs. 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. Since endpoint must advertise infinite completion credits for both header and data, the user must also make sure there are enough receive completion header/data buffer space for the completion packet return by the far end. 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 and stored until all CplDs associated with the current Tag are received. In typical applications, endpoints are required to advertise infinite completion data/header credits. There is no need to release credits to the far end for this type of application. For user designs that advertise non-infinite completion credits, credits must be released to the far end once the TLP's data is stored. This is done using the `cplh_processed_vc0` and `cpld_processed_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. Each cycle that `cpl[d/h]_processed_vc0` is held high releases 1 Completion data/header credit to the far end.

## 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.

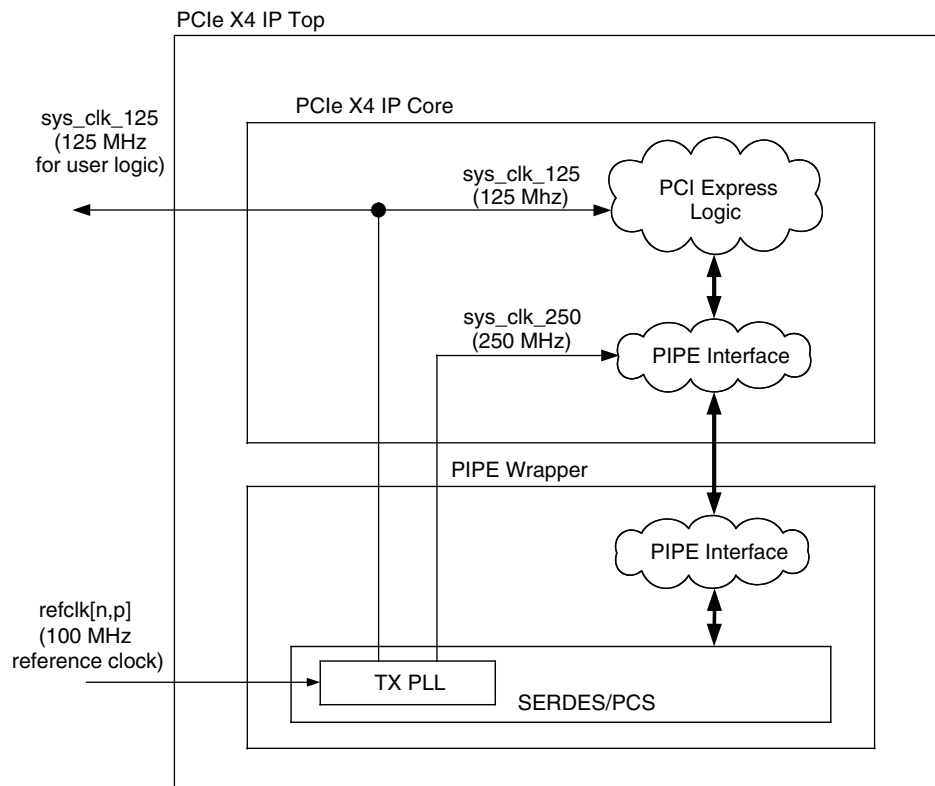
**Table 2. PCI Express core configuration space memory map**

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

## Clocking Scheme

**Figure 15. PCI Express IP Core Clocking Scheme**



Sys\_clk\_250 is used to clock portions of the designs that user a 32-bit data path. sys\_clk\_125 is used for the remainder of the logic that use a 64 bit data path and the user interface.

24



## Using the PCI Express IP Core

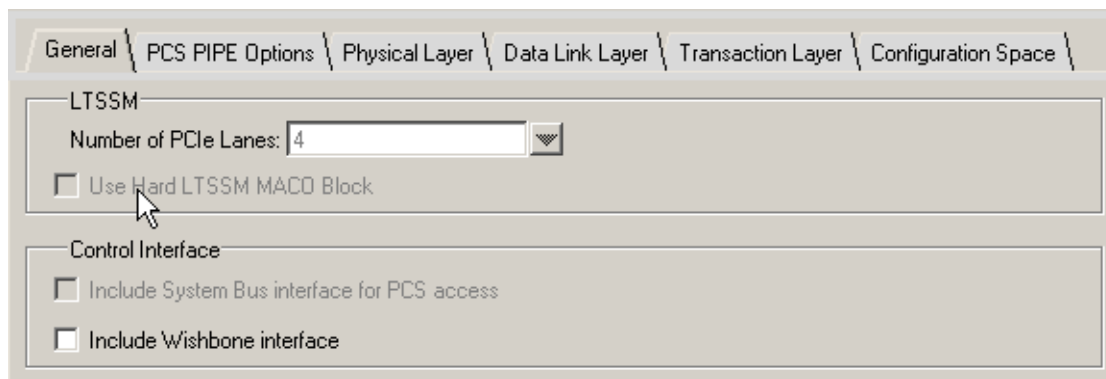
The PCI Express IP core is configured and created using the IPexpress GUI provided with the ispLEVER design tools. IPexpress creates all of the files necessary for the user to instantiate the IP core, simulate the core, synthesize the user's design using the core as a black box, and place and route the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

### 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 are those used for the PCI Express reference design.

**Figure 16. PCI Express IP Core General Options in IPexpress**



#### Number of PCI Express Lanes

Specifies the number of PCI Express 2.5Gbps lanes. The LatticeECP2M PCI Express IP core supports the generation of a x4 core only. The user can create a x1 link by powering down the unused channels in the pcie\_pcs.txt file. This will be discussed later in this user guide.

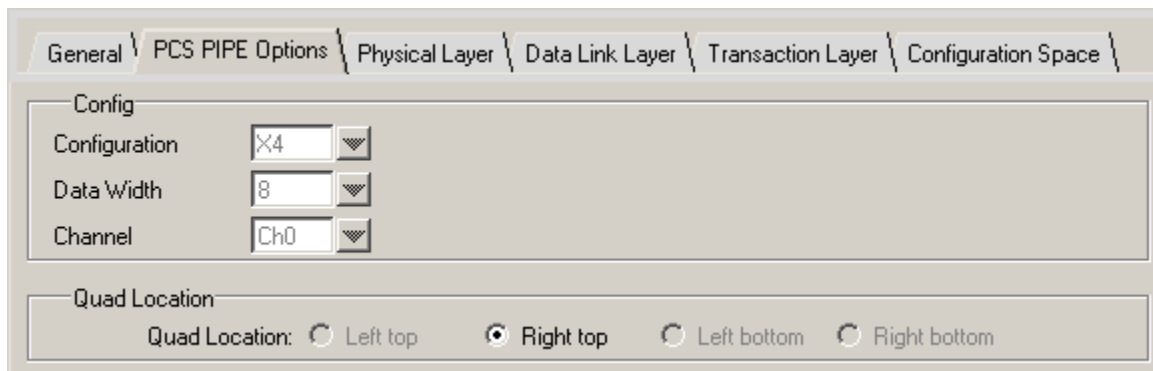
#### Include System Bus Interface for PCS Access

This option is not available for LatticeECP2M,

#### Include Wishbone Interface

This option includes a Wishbone interface to access certain features of the PCI Express core.

**Figure 17. PCI Express IP Core PCS PIPE Options in IP Express**



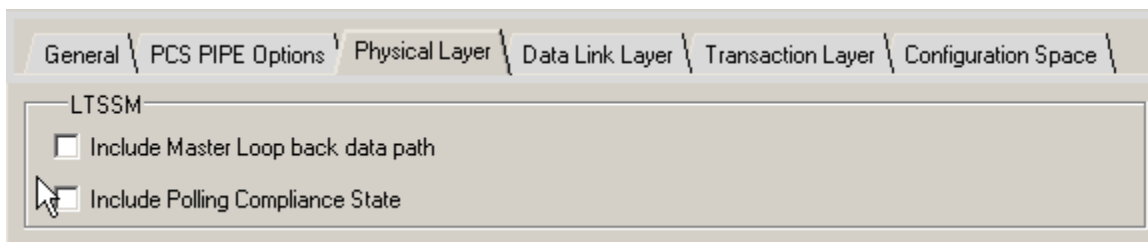
#### Config

The options in this area are hard coded.

**Quad Location**

This option specifies the location for the PCS SERDES in multi quad device.

**Figure 18. PCI Express IP Core Physical Layer Options in IPexpress**

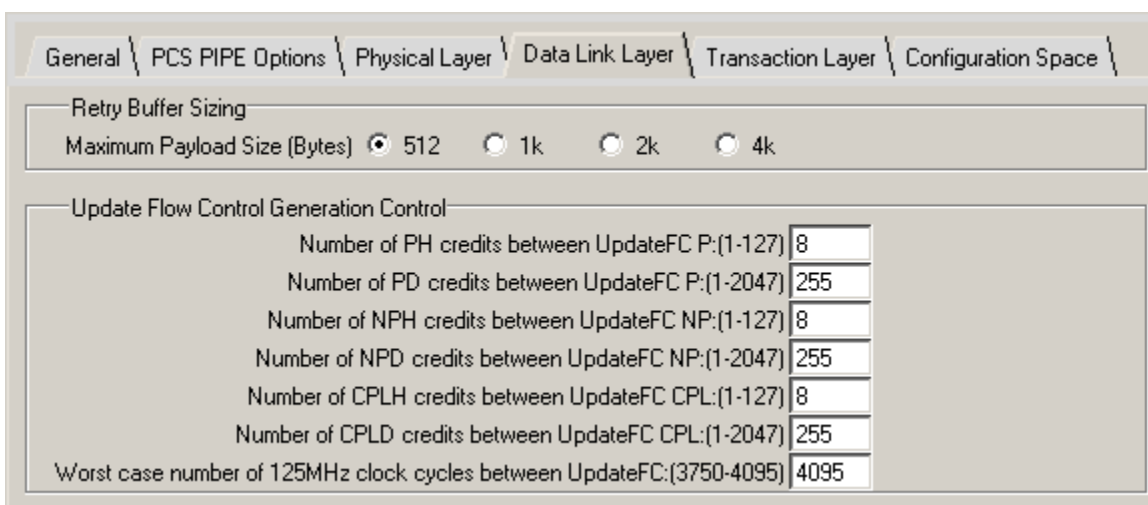
**Include Master Loop back data path**

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.

**Figure 19. PCI Express IP Core Data Link Layer Options in IPexpress**

**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 3. Total EBR Count Based on Max TLP Size**

Max TLP Size	Total EBRs in Core
512B	11
1KB	11
2KB	3
4KB	18

**Update Flow Control Generation Control**

There are two factors controlling 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.

**Number of CPL TLPs between UpdateFC**

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

**Number of CPLD TLPs between UpdateFC**

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

**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 20. PCI Express IP Core Transaction Layer Options in IPexpress**

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

- ☐ Infinite PH credits: Initial PH credits available:(1-127) 127
- ☐ Infinite PD credits: Initial PD credits available:(8-2047) 2047
- ☐ Infinite NPH credits: Initial NPH credits available:(1-127) 127
- ☐ Infinite NPD credits: Initial NPD credits available:(1-2047) 2047
- ☒ Infinite CPLH credits: Initial CPLH credits available:(1-127) 0
- ☒ Infinite CPLD credits: Initial CPLD credits available:(8-2047) 0

**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.

**Infinite CPLH credits**

This option should always be selected since the PCI Express specification requires an endpoint to advertise infinite credits for the Completion Header.

**Initial CPLH credits available**

If CPLH 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 CPLD credits**

This option should always be selected since the PCI Express specification requires an endpoint to advertise infinite credits for the Completion Data.

**Initial CPLD credits available**

If CPLD 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 21. PCI Express IP Core Configuration Space Options in IPexpress**

The screenshot shows the 'Configuration Space' tab in the IPexpress tool. It contains several configuration sections:

- Type0 Config Space:** Fields for Device ID (e235), Vendor ID (1204), Class Code (ff0000), Rev ID (01), BIST (00), Header Type (00), and BAR0 (ffffe000). Checkboxes for BAR0 Enable, BAR1 Enable, BAR2 Enable, BAR3 Enable, BAR4 Enable, and BAR5 Enable are present. Fields for BAR1 (ffff8000), BAR2 (00000000), BAR3 (00000000), BAR4 (00000000), and BAR5 (00000000) are also shown. Other fields include CardBus CIS Pointer (00000000), Subsystem ID (e235), Subsystem Vendor ID (1204), ExpROM Base Addr (00000000), and Expansion ROM Enable.
- Power Management Capability Structure:** Power Management Cap Reg [31:16] (0000), Data Scale Multiplier (0), and power consumption/dissipation fields for D0, D1, D2, and D3 (all 00).
- MSI Capability Structure:** Use Message Signaled Interrupts (unchecked), Number of Messages Requested (1).
- PCIe Capability Structure:** Max Payload Size (Bytes) (128), Device Capabilities Register [27:3] (00000000), Maximum Link Width (4), and Link Capabilities Register [17:10] (00).
- Device Serial Number:** Device Serial Number (0000000000000000).
- Advanced Error Reporting:** Use Advanced Error Reporting (unchecked).
- Terminate All Config TLPs:** Terminate All Config TLPs (checked).

**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 32 bit register contains information for the enabled BAR such as amount of address space, decoder type for memory bar (32/64), prefetchable attribute, and type (memory/IO). For 64 bit BAR the BARs will be paired.

**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.

**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.

**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.

**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.

**Created Files**

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. For more information on the directory structure created see the Lattice ispLeverCORE IP Tutorial.

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 4 provides a list of files created by IPexpress and describes how they are used.

**Table 4. File List**

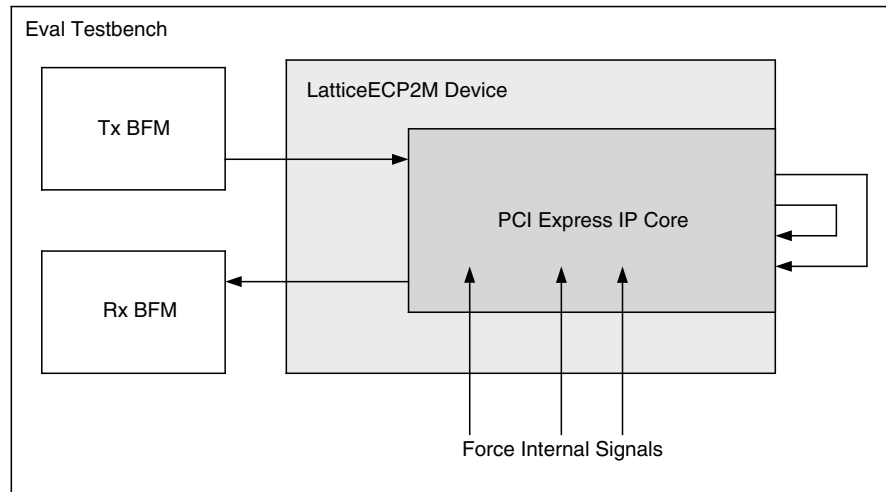
File	Sim	Synthesis/ ispLEVER	Description
<user name>_inst.v			This file provides an instance template for the IP. This file is located at <ipExpress_dir_name>.
<user name>_top.v	Yes	Yes	This file provides a sample top level design instantiating the IP core and the pcs_pipe_top wrapper for simulation and synthesis. This file is located at <ipExpress_dir_name>/pcie_x4_eval/<user name>/src/top/<user name>_top.v.
<user name>_beh.v	Yes		This file provides the front-end IP simulation model. This file is located at <ipExpress_dir_name>.
pci_exp_params.v	Yes		This file provides the user options of the IP for the simulation model. This file is located at <ipExpress_dir_name>.
pci_exp_ddefines.v	Yes		This file provides parameters necessary for the simulation. This file is located at <ipExpress_dir_name>.
<user name>_bb.v		Yes	This file provides the synthesis black box for the IP for instantiation in the user top level design for 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. This file is located at <ipExpress_dir_name>.
pcs_pipe_top.v	Yes	Yes	This file contains the top level PCS pipe wrapper for interfacing the PCS to the IP core. This file is located at <ipExpress_dir_name>/pcie_x4_eval/models/ecp2m.
pcs_top.v	Yes	Yes	This file contains the PCS wrapper fused by pcs_pipe_top. This file is located at <ipExpress_dir_name>/pcie_x4_eval/models/ecp2m.
pipe_top.v	Yes	Yes	This file contains the pipe functionality used by pcs_pipe_top. This file is located at <ipExpress_dir_name>/pcie_x4_eval/models/ecp2m.
pcs_pipe_bb.v		Yes	This file provides the synthesis black box for pcs_pipe_top.v. It can be used if the user chooses to use the synthesized pcs_pipe_top generated by ipExpress. This file is located at <ipExpress_dir_name>/pcie_x4_eval/models/ecp2m.
pcs_pipe_top.ngo		Yes	This file provides the synthesized pcs_pipe_top generated by ipExpress. It is needed by ispLEVER if the user chooses to instantiate the synthesis black box for pcs_pipe_top.v. This file is located at <ipExpress_dir_name>.
pcs_pipe_8b_X4.txt	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. This file is located at <ipExpress_dir_name>.
<user name>_eval.lpf		Yes	This file contains sample ispLEVER preferences to control place and route. These preferences should be included in the user's preference file. This file is located at <ipExpress_dir_name>/pcie_x4_eval/<user name>/impl/synplify.
<user name>.lpc			This file contains the IPexpress options used to recreate or modify the core in IPexpress. This file is located at <ipExpress_dir_name>.
pmi_*.ngo		Yes	These files contains the memories used by the IP core. These files need to be pointed to by the Build step by using the search path property. This file is located at <ipExpress_dir_name>.
<user name>_eval	Yes		This directory contains a sample design. This design is capable of performing a simulation and running through ispLEVER. This file is located at <ipExpress_dir_name>.

## 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.

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 22 illustrates this.

**Figure 22. LatticeECP2M PCI Express x4 Core Evaluation Testbench Block Diagram**



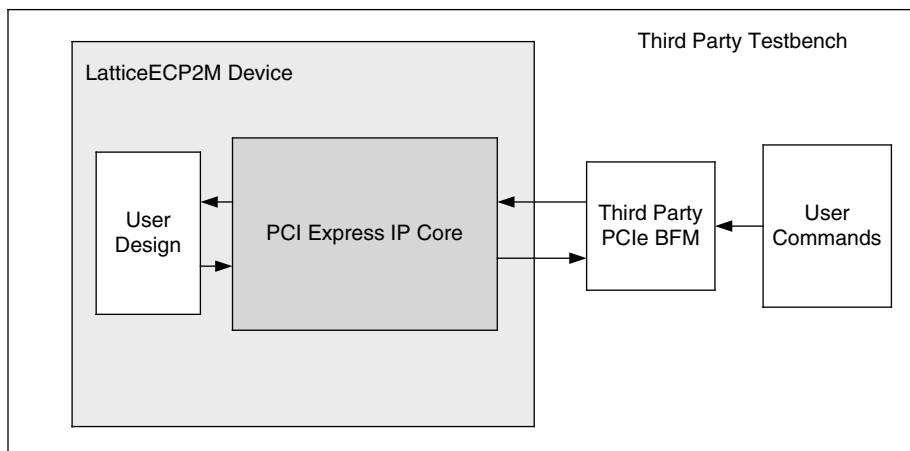
This testbench scheme works for its intent, but it is not easily extendible for the purposes of adding a Bus Functional Model (BFM) to emulate a real user system. In such a system, the endpoint is connected to the root complex device via the serial lanes. One possible approach is a verification environment where the endpoint (DUT) is connected to a Bus Functional Model capable of emulating a root complex device.

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.

### Third Party VIP

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

Lattice has chosen the Synopsys® PCI Express VIP for development of the PCI Express core. There are other third party vendors for PCI Express including Denali® and Cadence®.

**Figure 23. LatticeECP2M 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 ModelSim® simulators. The simulation model for the IP core is generated with the core in IPexpress with the name <user name>.v. This file calls two external libraries. One for the PCS/SERDES quad and the other for the PMI memories. The PCS/SERDES quad library is located in the isp-TOOLS directory structure in the modelsim/lattice/verilog/ecp2/pcsa\_mti\_work directory. The PMI library is located in the modelsim/lattice/verilog/pmi/pmi\_work directory.

It is necessary to load the pcie\_exp\_params.v and pcie\_exp\_ddefines.v files during simulation. These files provide "define constants" that are necessary for the simulation model.

Below is a sample ModelSim .do file to compile and simulate the evaluation design.

```
vlog \
+define+DEBUG=0 \
+define+SIMULATE \
+incdir+../../../../pcie_x4_core/testbench/top \
+incdir+../../../../pcie_x4_core/testbench/tests \
+incdir+../../../../src/params \
+incdir+../../../../pcie_x4_core/src/params \
../../../../pcie_x4_core/src/params/pci_exp_params.v \
../../../../pcie_x4_core/testbench/top/eval_pcie_x4.v \
../../../../pcie_x4_core/testbench/top/eval_tbtv.v \
../../../../pcie_x4_core/testbench/top/eval_tbrx.v \
../../../../pcie_x4_core_beh.v \
../../../../models/ecp2m/sync1s.v \
../../../../models/ecp2m/ctc.v \
../../../../models/ecp2m/pipe_top.v \
../../../../models/ecp2m/PCSC.v \
../../../../models/ecp2m/pcs_top.v \
../../../../models/ecp2m/pcs_pipe_top.v \
../../../../pcie_x4_core.v \
../../../../pcie_x4_core/src/top/pcie_x4_core_top.v -work work

vsim -t lps -c work.tb_top -L work -L ecp2m_vlg -L pmi_work -L pcsc_mti_work
```

Assuming the `pcie_x4_core_top.v` file is used as the top level, the above scripts can be modified to compile modules at the `pcie_x4_core_top` level as below:

```
vlog \
+define+DEBUG=0 \
+define+SIMULATE \
+incdir+../../../../src/params \
+incdir+../../../../pcie_x4_core/src/params \
../../../../pcie_x4_core/src/params/pci_exp_params.v \
../../../../pcie_x4_core_beh.v \
../../../../models/ecp2m/sync1s.v \
../../../../models/ecp2m/ctc.v \
../../../../models/ecp2m/pipe_top.v \
../../../../models/ecp2m/PCSC.v \
../../../../models/ecp2m/pcs_top.v \
../../../../models/ecp2m/pcs_pipe_top.v \
../../../../pcie_x4_core.v \
../../../../pcie_x4_core/src/top/pcie_x4_core_top.v -work work
```

## 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.

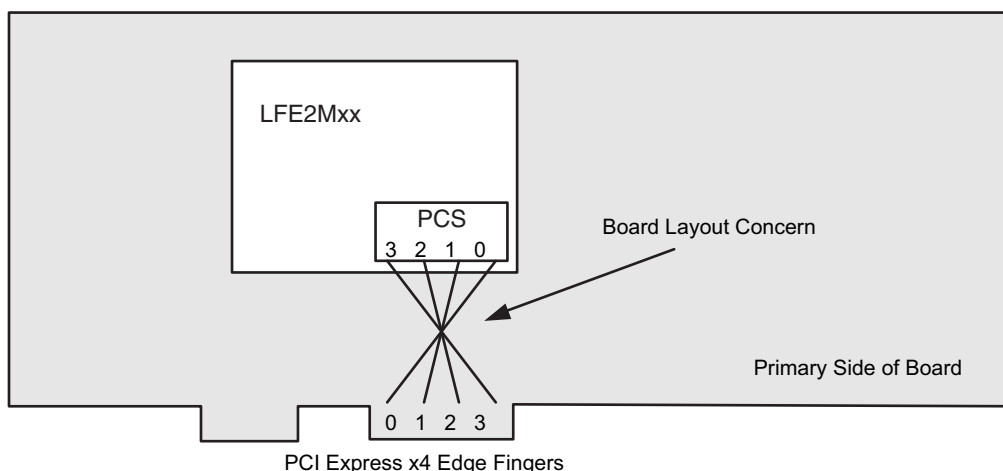
### Locating the IP

The LatticeECP2M PCI Express IP 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 IP core in a defined location on the device array. The hard blocks of the PCS/SERDES will drive the location of the IP.

### Board Layout Concerns for Add-in Cards

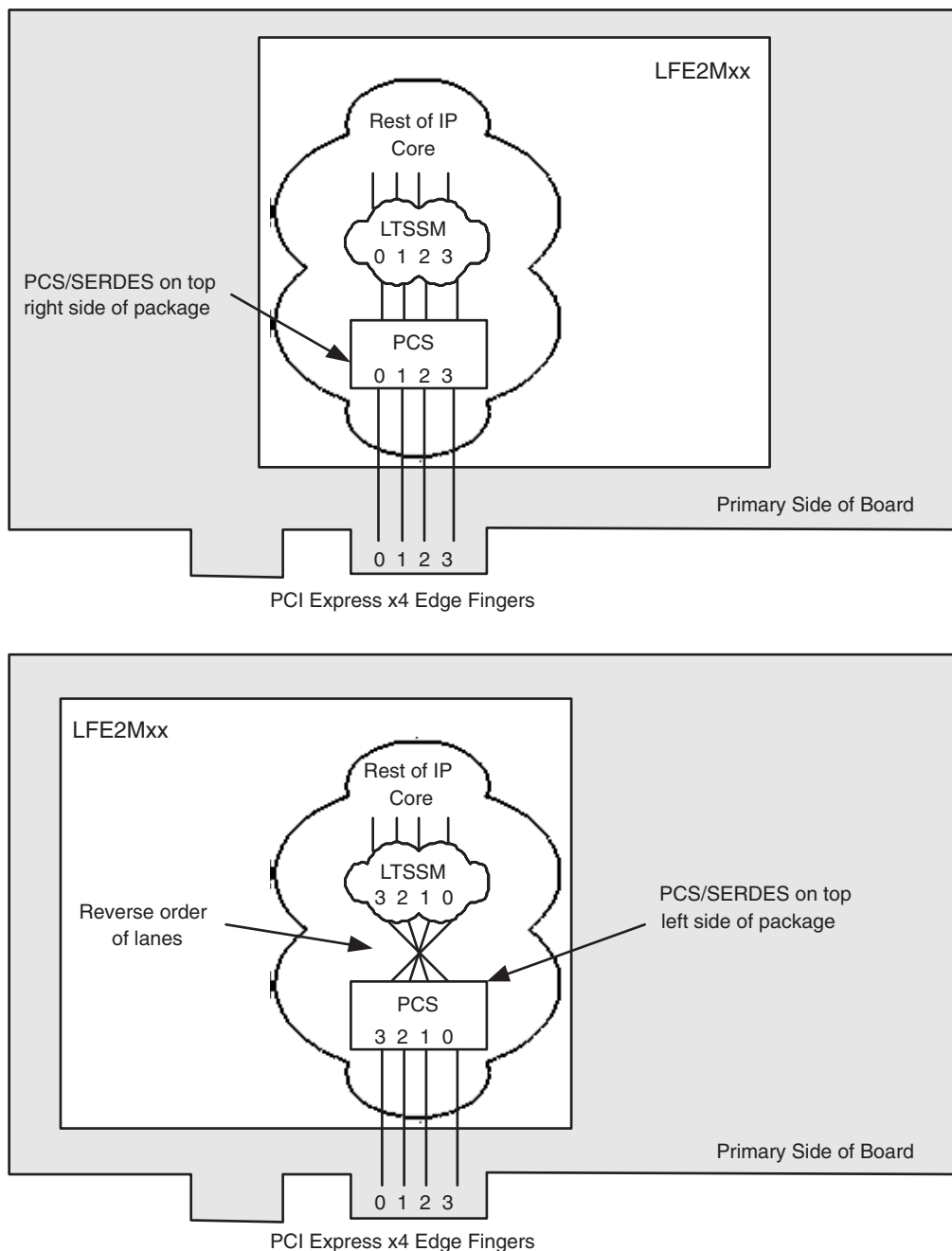
The PCI Express Add-in card connector edge-finger is physically designed for a particular orientation of lanes. The LatticeECP2M device package pinout also has a defined orientation of pins for the SERDES channels. The board layout will connect the PCI Express edge-fingers to the LatticeECP2M SERDES 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 24 provides an example of the board layout concern.

**Figure 24. Example of Board Layout Concern with x4 Link**



To allow 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, users can modify the top level design to reverse the connection order of the SERDES lanes to the LTSSM block of the IP core. This is required so that the PCI Express edge-finger lane 0 always connects to the logical LTSSM lane 0. Figure 25 provides a diagram of a normal and a reversed IP core implementation.

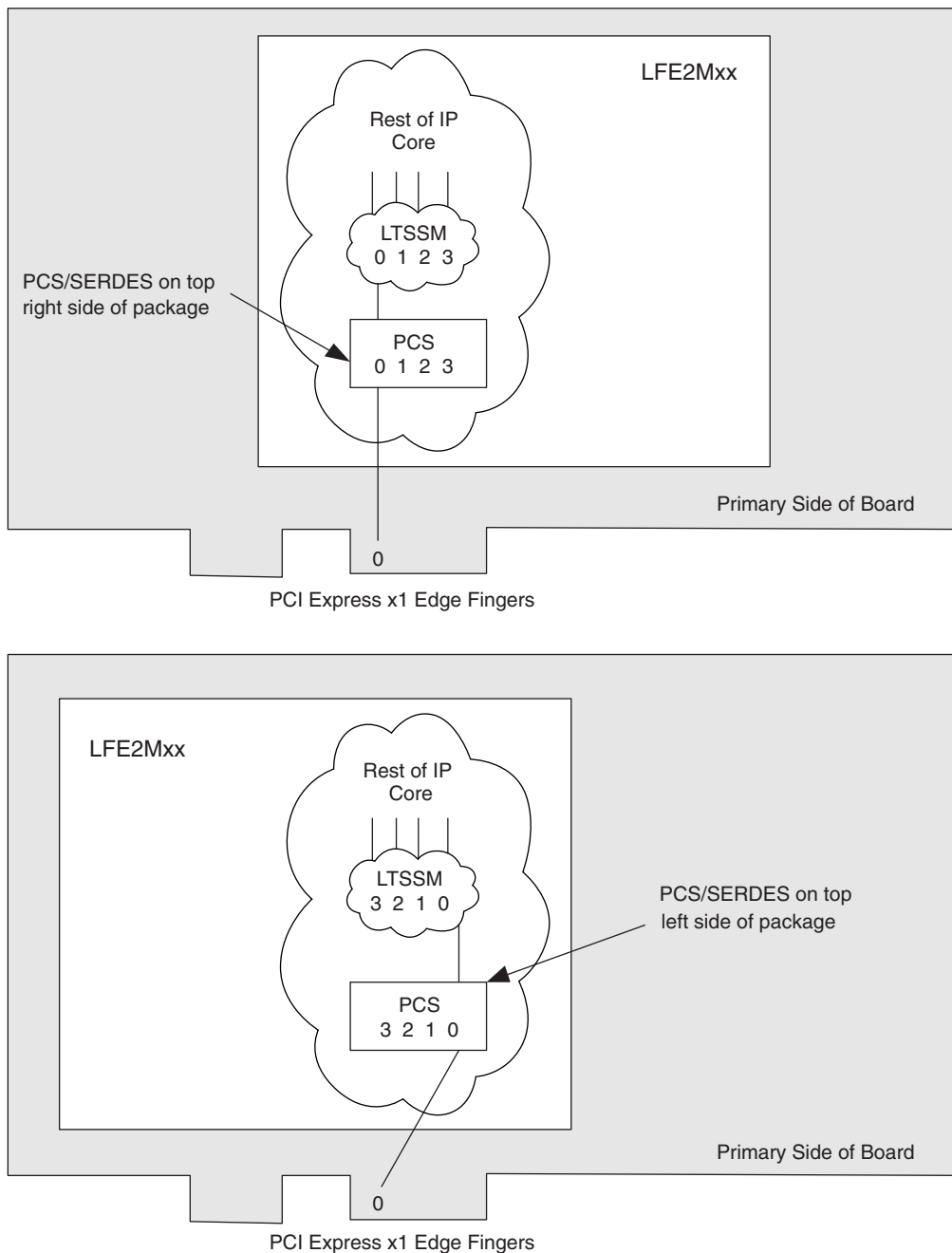
**Figure 25. Implementation of x4 IP Core to Edge-Fingers**



As shown in Figure 25, 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.

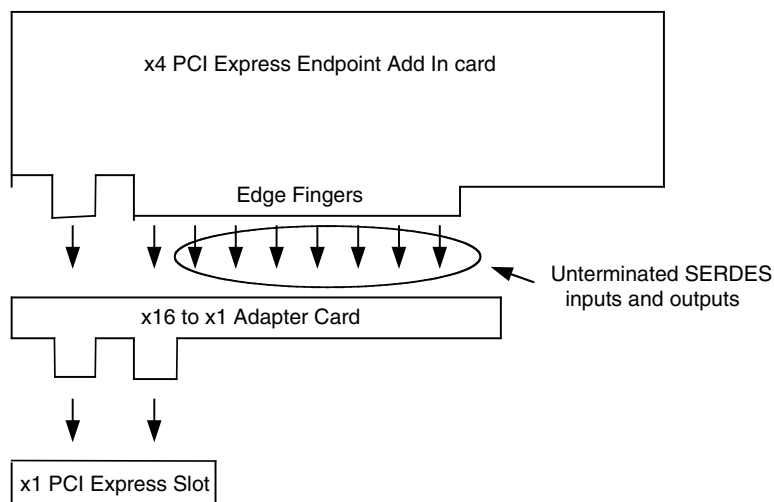
Figure 26 provides a diagram of a x1 IP core to illustrate the recommended solution in the board layout.

**Figure 26. 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 x4 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 27 provides the stack up of an endpoint add in card with an adapter card.

**Figure 27. 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 the LatticeECP2M 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 SERDES

### Set Up for x4

The PCS/SERDES memory map is configured during bitstream loading using the autoconfig file `pcs_pipe_8b_X4.txt` generated by IPExpress. This file contains parameters for setting up various aspects of the SERDES operation like protocol, channel operating mode, channel data width. The `pcs_pipe_8b_X4.txt` file enables all 4 channels of the selected SERDES quad for x4 operation.

```
# Select pipe protocol
PROTOCOL      "PIPE"

# Enable all 4 channels in the quad
CH0_MODE      "GROUP1"
CH1_MODE      "GROUP1"
CH2_MODE      "GROUP1"
CH3_MODE      "GROUP1"
```

The IP core automatically handles downgrade to x1 as required by the PCI Express specification.

### Set Up for x1

The user can modify the `pcs_pipe_X4.txt` file to force the IP core into x1 only mode. This is done by enable only one channel and turn off the other three channels.

```
# Selects pipe protocol
PROTOCOL      "PIPE"

# Enables channel 0
CH0_MODE      "SINGLE"

# Disables channel 1, 2, 3
CH1_MODE      "DISABLE"
```

---

```
CH2_MODE      "DISABLE"
CH3_MODE      "DISABLE"
```

## 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. LatticeECP2M50 or larger devices contains multiple SERDES quad so a location constraint is required to select the among the quads. This constraint is not needed for LatticeECP2M35 or LatticeECP2M20 since there is only one quad per device.

```
LOCATE COMP "u1_pcs_pipe_pcsc_inst" SITE "URPCS" ;
```

Refer to Appendix A for the SERDES/PCS Quad locations for LatticeECP2M devices.

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

```
FREQUENCY NET "pclk" 250.000000 MHz ;
FREQUENCY NET "u1_pcs_pipe/ff_rx_fclk_0" 250.000000 MHz ;
FREQUENCY NET "u1_pcs_pipe/ff_rx_fclk_1" 250.000000 MHz ;
FREQUENCY NET "u1_pcs_pipe/ff_rx_fclk_2" 250.000000 MHz ;
FREQUENCY NET "u1_pcs_pipe/ff_rx_fclk_3" 250.000000 MHz ;
```

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_inferred_clock" 125 MHz;
```

These clocks need to be routed on a primary clock route to provide the best signal integrity.

```
USE PRIMARY NET "pclk" ;
USE PRIMARY NET "u1_pcs_pipe/ff_rx_fclk_0" ;
USE PRIMARY NET "u1_pcs_pipe/ff_rx_fclk_1" ;
USE PRIMARY NET "u1_pcs_pipe/ff_rx_fclk_2" ;
USE PRIMARY NET "u1_pcs_pipe/ff_rx_fclk_3" ;
USE PRIMARY NET "sys_clk_125_inferred_clock" ;
```

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.

```
MULTICYCLE "M1" START CLKNET "pclk"
END CLKNET "sys_clk_125_inferred_clock" 8.000000 ns ;

MULTICYCLE "M2" START CLKNET "sys_clk_125_inferred_clock" END CLKNET "pclk"
8.000000 ns ;
```

Note that the synthesis tool may change the net names and instance names so the constraints may need to be modified accordingly.

## 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)

## PCI Express Power Up

There are two reset related requirements that must be considered for a PCI Express device:

1. The root complex can attempt to access a device as early as 100 ms following reset of that device.
2. The root complex must allow at least 1 second following reset of a device to determine that the device is broken due to failure to return Successful Completion status for valid Configuration Requests.

The Lattice ECP2M device input CML buffers internal 50-Ohm termination is turned on from power-up. This allows the LatticeECP2M device to be detected before the bitstream is fully loaded. Due to the FPGA programming time, the following strategy should be considered when powering up the device:

1. Using parallel flash if need to for faster programming time.
2. Allowing bitstream programming of the LatticeECP2M as soon as possible following power up by not connecting PROGRAM and GSR to PERST#.

The relative program times for LatticeECP2M devices are provided in Table 5.

**Table 5. LatticeECP2M Power Up Timing Specifications**

Specification	ECP2M20	ECP2M35	ECP2M50	ECP2M70	ECP2M100	Units
Power to INITn	50	50	50	50	50	ms
Worst-case Programming Time (SPI at 41 MHz)	146	244	390	488	634	ms
Worst-case Programming Time (Parallel Flash with CPLD) <sup>1</sup>	5	8	15	17	22	ms

1. 8-bit wide flash and external CPLD interfacing to LatticeECP2M at 150MHz SLAVE\_PARALLEL mode.

## References

- DS1007, *LatticeECP2M Family Data Sheet*
- Lattice Technical Note TN1124, *Lattice SERDES/PCS Usage Guide*
- Lattice Technical Note TN1108, *LatticeECP2M sysCONFIG Usage Guide*
- Lattice Technical Note TN1114, *Electrical Recommendations for Lattice SERDES*
- Lattice Technical Note TN1166, *PCI Express SIG Compliance Overview for Lattice Semiconductor FPGAs*

## Technical Support Assistance

Hotline: 1-800-LATTICE (North America)  
 +1-503-268-8001 (Outside North America)  
 e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)  
 Internet: [www.latticesemi.com](http://www.latticesemi.com)

---

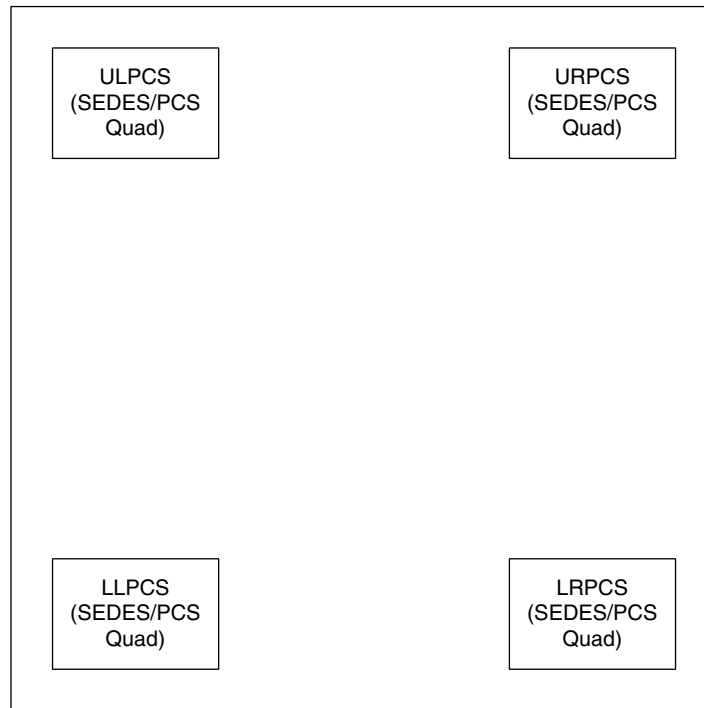
## Revision History

Date	Version	Change Summary
December 2007	01.0	Initial release.
January 2008	01.1	Added Appendix B. Performance and Resource Utilization LatticeECP2M-50 FPGAs - PCI Express x1 Endpoint.
		Updated Appendix C. Performance and Resource Utilization LatticeECP2M-50 FPGAs - PCI Express x4 Endpoint.

## Appendix A. SERDES/PCS Quads in LatticeECP2M Devices

The number of SERDES/PCS Quads and their locations vary depend on the LatticeECP2M device. The LatticeECP2M70 and LatticeECP2M100 contain four quads as shown in Figure 28. Table 6 shows the quads and their locations in all LatticeECP2M devices.

**Figure 28. SERDES/PCS Quads in the LatticeECP2M70/100 Device**



**Table 6. SERDES/PCS Quads in LatticeECP2M Devices**

Device	ECP2M20	ECP2M35	ECP2M50	ECP2M70	ECP2M100
Quad URC	Yes	Yes	Yes	Yes	Yes
Quad LRC	—	—	Yes	Yes	Yes
Quad ULC	—	—	—	Yes	Yes
Quad LLC	—	—	—	Yes	Yes

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

**Table 7. Performance and Resource Utilization**

IPexpress User-Configurable Mode	Max TLP	# BARs	AER	ECRC	LUTs	EBRs
Config Demo 1	512	2	No	No	5964	4
Config Demo 4	512	6	Yes	Yes	6578	4

Table 8 lists the IP core parameter settings for the Config Demos.

### Ordering Part Number

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

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](http://www.latticesemi.com/software).

**Table 8. 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 C. Performance and Resource Utilization LatticeECP2M-50 FPGAs - PCI Express x4 Endpoint

**Table 9. Performance and Resource Utilization**

IPexpress User-Configurable Mode	Max TLP	# BARs	AER	ECRC	LUTs	EBRs
Config Demo 1	512	2	No	No	12194	11
Config Demo 2	512	2	Yes	No	12660	11

Table 8 lists the IP core parameter settings for the Config Demos.

### Ordering Part Number

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

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](http://www.latticesemi.com/software).

**Table 10. 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 D. LTSSM Substate Encoding

phy\_ltssm\_substate encoding when LTSSM is in DETECT

3'b000 - DET\_WAIT  
3'b001 - DET\_QUIET  
3'b010 - DET\_GODET1  
3'b011 - DET\_ACTIVE1  
3'b100 - DET\_WAIT12MS  
3'b101 - DET\_GODET2  
3'b110 - DET\_ACTIVE2  
3'b111 - DET\_EXIT

phy\_ltssm\_substate encoding when LTSSM is in POLLING

3'b000 - POL\_WAIT  
3'b001 - POL\_ACTIVE  
3'b010 - POL\_COMPLIANCE  
3'b011 - POL\_CONFIG  
3'b100 - POL\_EXIT

phy\_ltssm\_substate encoding when LTSSM is in CONFIG

3'b000 - CFG\_WAIT  
3'b001 - CFG\_LINK\_WIDTH\_ST  
3'b010 - CFG\_LINK\_WIDTH\_ACC  
3'b011 - CFG\_LANE\_NUM\_WAIT  
3'b100 - CFG\_LANE\_NUM\_ACC  
3'b101 - CFG\_COMPLETE  
3'b110 - CFG\_IDLE  
3'b111 - CFG\_EXIT

phy\_ltssm\_substate encoding when LTSSM is in L0

3'b000 - L0\_WAIT  
3'b001 - L0\_L0  
3'b010 - L0\_L0RX  
3'b011 - L0\_L0TX  
3'b100 - L0\_EIDLE\_0  
3'b101 - L0\_EIDLE\_1  
3'b110 - L0\_EXIT

phy\_ltssm\_substate encoding when LTSSM is in L0s RX

3'b000 - L0s\_RX\_WAIT  
3'b001 - L0s\_RX\_ENTRY  
3'b010 - L0s\_RX\_IDLE  
3'b011 - L0s\_RX\_FTS  
3'b100 - L0s\_RX\_EXIT

phy\_ltssm\_substate encoding when LTSSM is in L1

3'b000 - L1\_WAIT  
3'b001 - L1\_ENTRY  
3'b010 - L1\_IDLE  
3'b011 - L1\_EXIT

phy\_ltssm\_substate encoding when LTSSM is in L2

3'b000 - L2\_WAIT  
3'b001 - L2\_IDLE

3'b010 - L2\_TRNSWAKE

3'b011 - L2\_EXIT

phy\_ltssm\_substate encoding when LTSSM is in RECOVERY

3'b000 - RCVRY\_WAIT

3'b001 - RCVRY\_RCVRLK

3'b010 - RCVRY\_RCVRCFG

3'b011 - RCVRY\_IDLE

3'b100 - RCVRY\_EXIT

phy\_ltssm\_substate encoding when LTSSM is in LOOPBACK

3'b000 - LBK\_WAIT

3'b001 - LBK\_ENTRY

3'b010 - LBK\_ACTIVE

3'b100 - LBK\_EIDLE

3'b101 - LBK\_LEXIT

3'b110 - LBK\_EXIT

phy\_ltssm\_substate encoding when LTSSM is in RECOVERY

3'b000 - HRST\_WAIT

3'b001 - HRST\_MRST

3'b010 - HRST\_SRST

3'b011 - HRST\_EXIT

phy\_ltssm\_substate encoding when LTSSM is in DISABLE

3'b000 - DIS\_WAIT

3'b001 - DIS\_TS1

3'b010 - DIS\_EIDLE\_0

3'b011 - DIS\_EIDLE\_1

3'b100 - DIS\_EXIT