LatticeCORE

# PCI Express 1.1 Root Complex Lite x1, x4 IP Core User's Guide

February 2012
IPUG85_01.1

# Table of Contents


**Chapter 1. Introduction ..... 4**
- Quick Facts ..... 4
- Features ..... 5
  - PHY Layer ..... 5
  - Data Link Layer ..... 5
  - Transaction Layer ..... 5
  - Top Level IP Support ..... 6

**Chapter 2. Functional Description ..... 7**
- Overview ..... 7
- Interface Description ..... 18
  - Transmit TLP Interface ..... 18
  - Transmit TLP Interface Waveforms for x1 ..... 23
  - Receive TLP Interface ..... 24
- Message Decode Interface ..... 26
  - Interrupt Signaling Messages ..... 26
  - Error Signaling Messages ..... 27
- Using the Transmit and Receive Interfaces ..... 28
  - As a Receiver ..... 28
  - As a Transmitter ..... 29

**Chapter 3. Parameter Settings ..... 30**
- General Tab ..... 31
  - PCI Express Link Configuration ..... 31
  - Include Master Loopback Data Path ..... 31
  - Maximum Payload Size ..... 31
  - Include ECRC Support ..... 32
- Flow Control Tab ..... 32
  - Initial Receive Credits ..... 32
  - Infinite PH Credits ..... 33
  - Initial PH Credits Available ..... 33
  - Infinite PD Credits ..... 33
  - Initial PD Credits Available ..... 33
  - Infinite NPH Credits ..... 33
  - Initial NPH Credits Available ..... 33
  - Infinite NPD Credits ..... 33
  - Initial NPD Credits Available ..... 33
  - Infinite CPLH Credits ..... 33
  - Initial CPLH Credits Available ..... 33
  - Infinite CPLD Credits ..... 33
  - Initial CPLD Credits Available ..... 33
  - Update Flow Control Generation Control ..... 33
  - Number of P TLPs Between UpdateFC ..... 34
  - Number of PD TLPs Between UpdateFC ..... 34
  - Number of NP TLPs Between UpdateFC ..... 34
  - Number of NPD TLPs Between UpdateFC ..... 34
  - Number of CPL TLPs Between UpdateFC ..... 34
  - Number of CPLD TLPs Between UpdateFC ..... 34
  - Worst Case Number of 125MHz Clock Cycles Between UpdateFC ..... 34

**Chapter 4. IP Core Generation and Evaluation ..... 35**
- Licensing the IP Core ..... 35


© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Licensing Requirements for LatticeECP2M/LatticeECP3 ..... 35
Getting Started ..... 35
IPexpress-Created Files and Top Level Directory Structure ..... 37
Running Functional Simulation ..... 40
Synthesizing and Implementing the Core in a Top-Level Design ..... 40
Hardware Evaluation ..... 41
Enabling Hardware Evaluation in Diamond ..... 41
Enabling Hardware Evaluation in ispLEVER ..... 41
Updating/Regenerating the IP Core ..... 41
Regenerating an IP Core in Diamond ..... 41
Regenerating an IP Core in ispLEVER ..... 42
**Chapter 5. Using the IP Core ..... 43**
Simulation and Verification ..... 43
Simulation Strategies ..... 43
Third Party Verification IP ..... 44
FPGA Design Implementation ..... 44
Setting Up the Core ..... 44
Setting Up for x4 (No Flip) ..... 44
Setting Up for x4 (Flipped) ..... 45
Setting Design Constraints ..... 45
Errors and Warnings ..... 45
Clocking Scheme ..... 46
Locating the IP ..... 47
Board-Level Implementation Information ..... 50
PCI Express Power-Up ..... 50
Board Layout Concerns ..... 52
Troubleshooting ..... 55
**Chapter 6. Core Verification ..... 56**
**Chapter 7. Support Resources ..... 57**
Lattice Technical Support ..... 57
Online Forums ..... 57
Telephone Support Hotline ..... 57
E-mail Support ..... 57
Local Support ..... 57
Internet ..... 57
PCIe Solutions Web Site ..... 57
PCI-SIG Website ..... 57
References ..... 58
LatticeECP3 ..... 58
LatticeECP2M ..... 58
Revision History ..... 58
**Appendix A. Resource Utilization ..... 59**
Configuration ..... 59
LatticeECP2M Utilization (x4 RC-Lite) ..... 59
Ordering Part Number ..... 59
LatticeECP3 Utilization (x4 RC-Lite) ..... 59
Ordering Part Number ..... 59
LatticeECP2M Utilization (x1 RC-Lite) ..... 60
Ordering Part Number ..... 60
LatticeECP3 Utilization (x1 RC-Lite) ..... 60
Ordering Part Number ..... 61

# Introduction

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

Lattice's PCI Express Root Complex (RC) Lite core provides an x1 or x4 root complex solution from the electrical SERDES interface, physical layer, data link layer and a minimum transaction layer in PCI express protocol stack. This IP is a lighter version of the root complex intended to be used in simple local bus bridging applications. This solution supports the LatticeECP3™ and LatticeECP2M™ FPGA device families and is an extremely economical, high value FPGA platform.

This user's guide covers two versions of the Lattice PCI Express RC-Lite core:

- The **PCI Express x4 RC-Lite Core** targets the LatticeECP3 and LatticeECP2M families of devices.
- The **PCI Express x1 RC-Lite Core** targets the LatticeECP3 and LatticeECP2M families. This is a reduced LUT count x1 core with a 16-bit datapath.

Refer to Lattice's PCIe Solutions web site at:

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

## Quick Facts

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

***Table 1-1. PCI Express RC-Lite IP Core Quick Facts***

| | | PCI Express RC-Lite IP Configuration | | | |
|---|---|---|---|---|---|
| | | x4 RC | | Native x1 RC | |
| **Core Requirements** | FPGA Families Supported | LatticeECP3 and LatticeECP2M | | | |
| | Minimal Device Needed [1] | LFE3-17E-7FN484C | LFE2M-20E-6F484C | LFE3-17E-7FN484C | LFE2M-20E-6F484C |
| **Typical Resource Utilization** | Targeted Device | LFE3-70E-7FN672C | LFE2M-50E-6F672C | LFE3-70E-7FN672C | LFE2M-50E-6F672C |
| | Data Path Width | 64 | 64 | 16 | 16 |
| | LUTs | 10650 | 10900 | 4700 | 4800 |
| | sysMEM EBRs | 9 | 9 | 5 | 5 |
| | Registers | 8500 | 8500 | 3100 | 3150 |
| **Design Tool Support** | Lattice Implementation | Diamond® 1.1 or ispLEVER® 8.1 | | | |
| | Synthesis | Synopsys® Synplify® Pro for Lattice D-2009.12L-1 | | | |
| | | Mentor Graphics® Precision® RTL | | | |
| | Simulation | Aldec Active-HDL® 8.2 (Windows only, Verilog and VHDL) | | | |
| | | Mentor Graphics ModelSim® SE 6.5F (Verilog Only) | | | |
| | | Cadence® NC-Verilog® (Linux only) | | | |

1. The packages specified in the Minimal Device Needed row relate to the many user interface signals implemented as I/Os in the evaluation design. Depending on the application, it might be possible to implement a design in a package with fewer I/O pins since the majority of the user interface signals are terminated inside the FPGA.

## Features

The Lattice PCI Express RC-Lite IP core supports the following features.

### PHY Layer

- 2.5 Gbps 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 +/- 300 ppm
- Framing and application of symbols to lanes
- Data scrambling and de-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
- LCRC generation/checking
- Sequence number appending/checking/removing
- Retry buffer and management
- Receiver buffer

### Transaction Layer

- Transmit and Receive Flow control
- Malformed and poisoned TLP detection
- Optional ECRC generation/checking
- INTx message TLP decoding and interrupt signaling to user
- Error message TLP decoding and signaling to user.
- 128, 256, 512, 1k, 2k or 4k bytes maximum payload size

## Top Level IP Support

- 125 MHz user interface
  - x4 supports a 64-bit data path
  - x1 supports a 16-bit data path
- 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
- Higher layer control of LTSSM via ports

# Functional Description

This chapter provides a functional description of the Lattice PCI Express RC-Lite IP core.

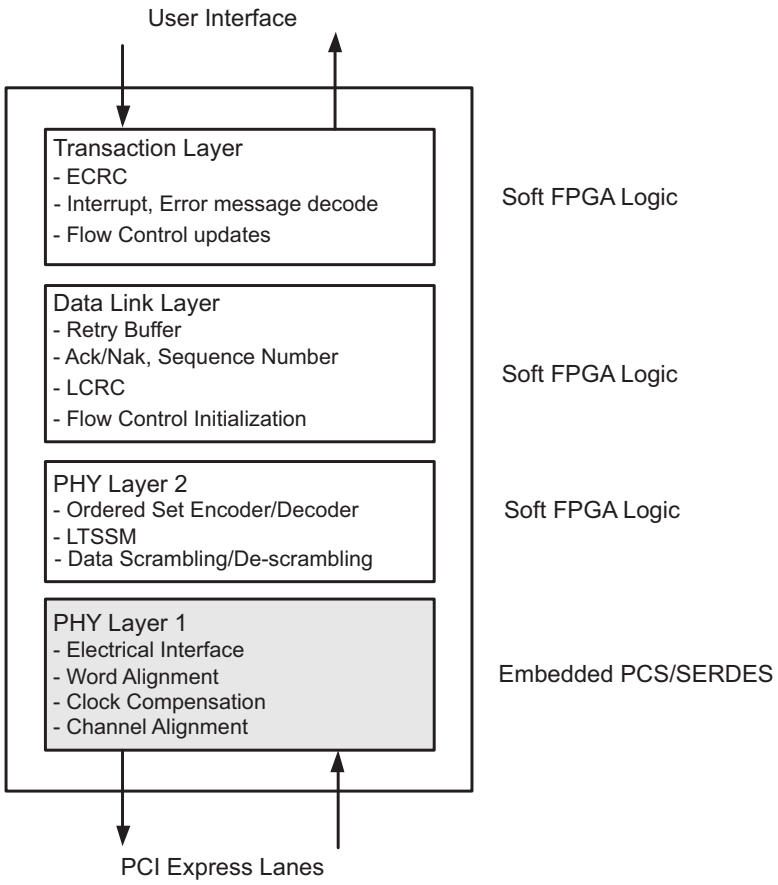
## Overview

The PCI Express RC-Lite IP core is implemented in several different FPGA technologies. These technologies include soft FPGA fabric elements such as LUTs, registers, embedded block RAMs (EBRs) and embedded hard elements with the PCS/SERDES.

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

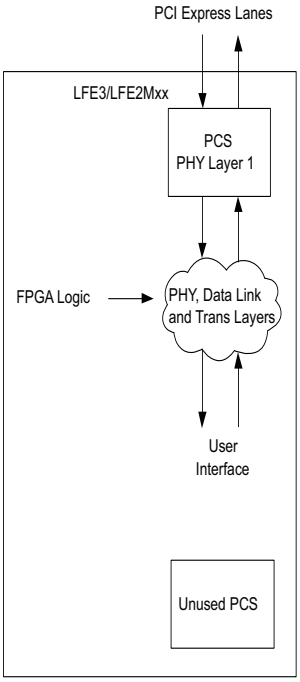
Figure 2-1 provides a high-level block diagram to illustrate the main functional blocks and the technology used to implement PCI Express RC-Lite IP core functions.

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




As the PCI Express RC-Lite IP core proceeds through the Diamond or ispLEVER software design flow specific technologies are targeted to their specific locations on the device. Figure 2-2 provides implementation representations of the LFE3/LFE2M devices with a PCI Express RC-Lite IP core.

*Figure 2-2. PCI Express RC-Lite IP Core Implementation in LatticeECP3 and LatticeECP2M Devices*




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 SERDES/PCS blocks as described in "Overview" on page 7. The FPGA logic placement and routing is the job of the Diamond or ispLEVER design tools to select regions nearby the hard SERDES/PCS blocks to achieve the timing goals.

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

***Figure 2-3. PCI Express RC-Lite IP Core Interfaces***

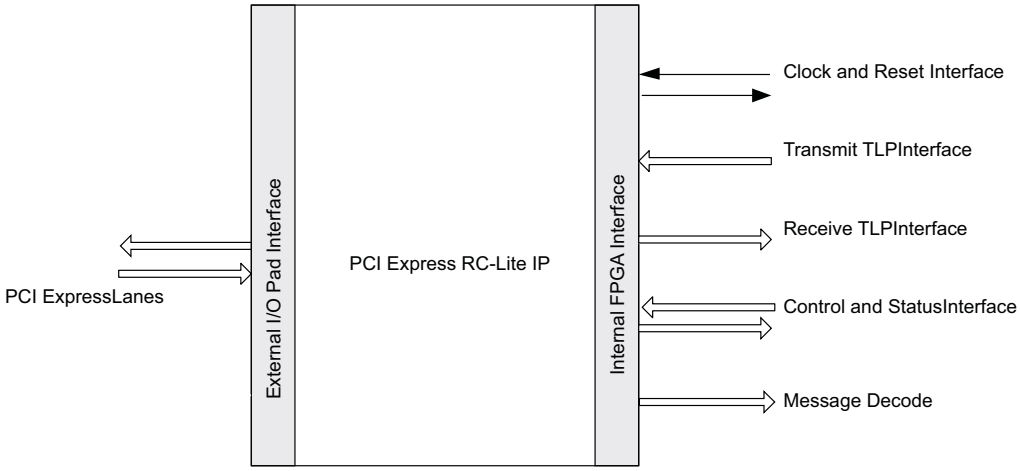



Table 2-1 provides the list of ports and descriptions for the PCI Express RC-Lite IP core.

***Table 2-1. PCI Express RC-Lite IP Core Port List***

| Port Name | Direction | Clock | Description |
|---|---|---|---|
| **Clock and Reset Interface** | | | |
| refclk[p,n] | Input | | 100 MHz PCI Express differential reference clock used to generate the 2.5 Gbps data. |
| sys_clk_125 | Output | | 125 MHz clock derived from refclk to be used in the user application. |
| rst_n | Input | | Active-low asynchronous data path and state machine reset. This port will be connected to the GSR for the entire device. This reset is pulsed after bit stream download and will not need to be asserted by the user. |
| hdin[p,n]_[0,1,2,3] | Input | | PCI Express 2.5 Gbps CML inputs for lanes 0,1,2, and 3. The port “flip_lanes” is used to define the connection of PCS/SERDES channel to PCI Express lane.<br>flip_lanes=0<br>hdin[p,n]_0 - PCI Express Lane 0<br>hdin[p,n]_1 - PCI Express Lane 1<br>hdin[p,n]_2 - PCI Express Lane 2<br>hdin[p,n]_3 - PCI Express Lane 3<br><br>flip_lanes=1<br>hdin[p,n]_0 - PCI Express Lane 3<br>hdin[p,n]_1 - PCI Express Lane 2<br>hdin[p,n]_2 - PCI Express Lane 1<br>hdin[p,n]_3 - PCI Express Lane 0 |

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

| Port Name | Direction | Clock | Description |
|---|---|---|---|
| hdout[p,n]_[0,1,2,3] | Output | | PCI Express 2.5 Gbps CML outputs for lanes 0,1,2, and 3. The port “flip_lanes” is used to define the connection of PCS/SERDES channel to PCI Express lane.<br><br>flip_lanes=0<br>hdout[p,n]_0 - PCI Express Lane 0<br>hdout[p,n]_1 - PCI Express Lane 1<br>hdout[p,n]_2 - PCI Express Lane 2<br>hdout[p,n]_3 - PCI Express Lane 3<br><br>flip_lanes=1<br>hdout[p,n]_0 - PCI Express Lane 3<br>hdout[p,n]_1 - PCI Express Lane 2<br>hdout[p,n]_2 - PCI Express Lane 1<br>hdout[p,n]_3 - PCI Express Lane 0 |
| **Transmit TLP Interface** | | | |
| tx_data_vc0[n:0] | Input | sys_clk_125 | x4 Transmit data bus<br>[63:56] Byte N<br>[55:48] Byte N+1<br>[47:40] Byte N+2<br>[39:32] Byte N+3<br>[31:24] Byte N+4<br>[23:16] Byte N+5<br>[15: 8] Byte N+6<br>[7: 0] Byte N+7<br><br>x1 Transmit data bus<br>[15:8] Byte N<br>[7:0] Byte N+1 |
| tx_req_vc0 | Input | sys_clk_125 | Active High transmit request. This port is asserted when the user wants to send a TLP. If several TLPs will be provided in a burst, this port can remain High until all TLPs have been sent. |
| tx_rdy_vc0 | Output | sys_clk_125 | Active High transmit ready indicator. Tx_st should be provided next clock cycle after tx_rdy is High.This port will go Low between TLPs. |
| tx_st_vc0 | Input | sys_clk_125 | Active High transmit start of TLP indicator. |
| tx_end_vc0 | Input | sys_clk_125 | Active High transmit end of TLP indicator. This signal must go Low at the end of the TLP. |
| tx_nlfy_vc0 | Input | sys_clk_125 | Active High transmit nullify TLP. Can occur anywhere during the TLP. If tx_nlfy_vc0 is asserted to nullify a TLP the tx_end_vc0 port should not be asserted. The tx_nlfy_vc0 terminates the TLP. |
| tx_dwen_vc0 | Input | sys_clk_125 | Active High transmit 32-bit word indicator. Used if only bits [63:32] provide valid data. This port is available only on the x4 core. |
| tx_val | Output | sys_clk_125 | Active High transmit clock enable. When a x4 is downgraded to a x1, this signal is used as the clock enable to downshift the transmit bandwidth. This port is available only on the x4 core. |

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

| Port Name | Direction | Clock | Description |
|---|---|---|---|
| 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.<br>Ph - Posted header<br>Nph - Non-posted header<br>Cplh - Completion header<br>This credit interface is only updated when an UpdateFC DLLP is received from the PCI Express line.<br>[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.<br>This port will decrement as TLPs are sent and increment as UpdateFCs are received.<br>pd - posted data<br>npd - non-posted data<br>cpld - completion data<br>[12] - This bit indicates the receiver has infinite credits. If this bit is High, then bits [11:0] should be ignored.<br>[11:0] - The amount of credits available at the receiver. |

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

| Port Name | Direction | Clock | Description |
|---|---|---|---|
| **Receive TLP Interface** | | | |
| rx_data_vc0[n:0] | Output | sys_clk_125 | x4 Receive data bus<br>[63:56] Byte N<br>[55:48] Byte N+1<br>[47:40] Byte N+2<br>[39:32] Byte N+3<br>[31:24] Byte N+4<br>[23:16] Byte N+5<br>[15: 8] Byte N+6<br>[7: 0] Byte N+7<br><br>x1 Receive data bus<br>[15:8] Byte N<br>[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 x4 core. |
| rx_ecrc_err_vc0 | Output | sys_clk_125 | Active High ECRC error indicator. Indicates a ECRC error in the current TLP.<br>This port is available only if ECRC feature is selected while generating the IP core. |
| rx_pois_tlp_vc0 | Output | sys_clk_125 | Active High poisoned TLP indicator. Asserted if “poisoned (EP) “ bits is set in any TLP with data. |
| rx_malf_tlp_vc0 | Output | sys_clk_125 | Active High malformed TLP indicator. Indicates a problem with the current TLPs length or format. |
| [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 UpdateFC timer to expire. |
| [ph,nph,cplh]_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. |
| [pd,npd,cpld]_processed_vc0 | Input | sys_clk_125 | Active High enable for [pd, npd,cpld]_num_vc0 port. The user should place the number of data credits processed on the [pd, npd,cpld]_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 or CPLD credits processed. It is enabled by the [pd, npd,cpld]_processed_vc0 port. |
| **Control and Status** | | | |
| **PHYSICAL LAYER** | | | |
| flip_lanes | Input | Async | Reverses the lane connections to the 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.<br>0-Lane 0 connects to SERDES Channel 0, etc.<br>1-Lane 0 connects to SERDES Channel 3, etc. |

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

| Port Name | Direction | Clock | Description |
|---|---|---|---|
| phy_ltssm_state[3:0] | Output | sys_clk_125 | PHY Layer LTSSM current state<br>0000 - Detect<br>0001 - Polling<br>0010 - Config<br>0011 - L0<br>0100 - L0s<br>0101 - L1<br>0110 - L2<br>0111 - Recovery<br>1000 - Loopback<br>1001 - Hot Reset<br>1010 - Disabled |

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

| Port Name | Direction | Clock | Description |
|---|---|---|---|
| phy_ltssm_substate[2:0] | Output | sys_clk_125 | PHY Layer LTSSM current sub state. Each major LTSSM state has a series of sub states.<br><br>When phy_ltssm_state=DETECT<br>000 - DET_WAIT<br>001 - DET_QUIET<br>010 - DET_GODET1<br>011 - DET_ACTIVE1<br>100 - DET_WAIT12MS<br>101 - DET_GODET2<br>110 - DET_ACTIVE2<br>111 - DET_EXIT<br><br>When phy_ltssm_state=POLLING<br>000 - POL_WAIT<br>001 - POL_ACTIVE<br>010 - POL_COMPLIANCE<br>011 - POL_CONFIG<br>100 - POL_EXIT<br><br>When phy_ltssm_state=CONFIG<br>000 - CFG_WAIT<br>001 - CFG_LINK_WIDTH_ST<br>010 - CFG_LINK_WIDTH_ACC<br>011 - CFG_LANE_NUM_WAIT<br>100 - CFG_LANE_NUM_ACC<br>101 - CFG_COMPLETE<br>110 - CFG_IDLE<br>111 - CFG_EXIT<br><br>When phy_ltssm_state=L0<br>000 - L0_WAIT<br>001 - L0_L0<br>010 - L0_L0RX<br>011 - L0_L0TX<br>100 - L0_EIDLE_0<br>101 - L0_EIDLE_1<br>110 - L0_EXIT<br><br>When phy_ltssm_state=L0s<br>000 - L0s_RX_WAIT<br>001 - L0s_RX_ENTRY<br>010 - L0s_RX_IDLE<br>011 - L0s_RX_FTS<br>100 - L0s_RX_EXIT<br><br>When phy_ltssm_state=L1<br>000 - L1_WAIT<br>001 - L1_ENTRY<br>010 - L1_IDLE<br>011 - L1_EXIT<br><br>When phy_ltssm_state=L2<br>000 - L2_WAIT<br>001 - L2_IDLE |
| phy_cfgln_sum[2:0] | Output | sys_clk_125 | Link Width<br>000 - No link defined<br>001 - Link width = 1<br>100 - Link width = 4 |

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

| Port Name | Direction | Clock | Description |
|---|---|---|---|
| phy_pol_compliance | Output | sys_clk_125 | Active High indicator that the LTSSM is in the Polling.Compliance state. |
| phy_force_cntl[3:0] | Input | sys_clk_125 | These signals to be used only for debug purpose.<br>[0] - force_lsm_active - Forces the Link State Machine for all of the channels to the linked state.<br>[1] - force_rec_ei - Forces the detection of a received electrical idle.<br>[2] - force_phy_status - Forces the detection of a receiver during the LTSSM Detect state on all of the channels.<br>[3] - force_disable_scr - Disables scrambler and de-scrambler. |
| phy_ltssm_cntl[15:0] | Input | sys_clk_125 | [0] – no_pcie_train - 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.<br>[1] - retrain - Active High request to re-train the link when LTSSM is in L0.<br>[2] – hl_disable_scr - Active High to set the disable scrambling bit in the TS1/TS2 sequence<br>[3] – hl_gto_dis – Active High request to go to Disable state when LTSSM is in Config or Recovery.<br>[4] – hl_gto_det – Active High request to go to Detect state when LTSSM is in L2 or Disable.<br>[5] – hl_gto_hrst - Active High request to go to Hot Reset when LTSSM is in Recovery.<br>[6] – hl_gto_l0stx - Active High request to go to L0s when LTSSM is in L0.<br>[7] – hl_gto_l0stxfts - Active High request to go to L0s and transmit FTS when LTSSM is in L0s.<br>[8] – hl_gto_l1 - Active High request to go to L1 when LTSSM is in L0.<br>[9] – hl_gto_l2 - Active High request to go to L2 when LTSSM is in L0.<br>[13:10] – hl_gto_lbk - Active High request to go to Loopback when LTSSM is in Config or Recovery<br>[14] – hl_gto_rcvry - Active High request to go to Recovery when LTSSM is in L0, L0s or L1.<br>[15] – hl_gto_cfg - Active High request to go to Config when LTSSM is in Recovery. |
| 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 the IPexpress tool. |
| 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 the IPexpress tool.<br><br>**x4 port**<br>[7] - K control on tx_lbk_data[63:56]<br>[6] - K control on tx_lbk_data[55:48]<br>[5] - K control on tx_lbk_data[47:40]<br>[4] - K control on tx_lbk_data[39:32]<br>[3] - K control on tx_lbk_data[31:24]<br>[2] - K control on tx_lbk_data[23:16]<br>[1] - K control on tx_lbk_data[15:8]<br>[0] - K control on tx_lbk_data[7:0]<br><br>**x1 port**<br>[1] - K control on tx_lbk_data[15:8]<br>[0] - K control on tx_lbk_data[7:0] |

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

| Port Name | Direction | Clock | Description |
|---|---|---|---|
| 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 the IPexpress tool.<br><br>**x4 port**<br>[63:56] - Lane 7 data<br>[55:48] - Lane 6 data<br>[47:40] - Lane 5 data<br>[39:32] - Lane 4 data<br>[31:24] - Lane 3 data<br>[23:16] - Lane 2 data<br>[15:8] - Lane 1 data<br>[7:0] - Lane 0 data<br><br>**x1 port**<br>[15:8] - Lane 1 data<br>[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 the IPexpress tool.<br><br>**x4 port**<br>[7] - K control on rx_lbk_data[63:56]<br>[6] - K control on rx_lbk_data[55:48]<br>[5] - K control on rx_lbk_data[47:40]<br>[4] - K control on rx_lbk_data[39:32]<br>[3] - K control on rx_lbk_data[31:24]<br>[2] - K control on rx_lbk_data[23:16]<br>[1] - K control on rx_lbk_data[15:8]<br>[0] - K control on rx_lbk_data[7:0]<br><br>**x1 port**<br>[1] - K control on rx_lbk_data[15:8]<br>[0] - K control on rx_lbk_data[7:0] |
| 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 the IPexpress tool.<br><br>**x4 port**<br>[63:56] - Lane 7 data<br>[55:48] - Lane 6 data<br>[47:40] - Lane 5 data<br>[39:32] - Lane 4 data<br>[31:24] - Lane 3 data<br>[23:16] - Lane 2 data<br>[15:8] - Lane 1 data<br>[7:0] - Lane 0 data<br><br>**x1 port**<br>[15:8] - Lane 1 data<br>[7:0] - Lane 0 data |
| **DATA LINK LAYER** | | | |

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

| Port Name | Direction | Clock | Description |
|---|---|---|---|
| dll_status[7:0] | Output | sys_clk_125 | [0] – dl_up - Data Link Layer is in the up and ready to send/receive TLPs from/to Transaction Layer.<br>[1] - dl_init - Data Link Layer is in the DL_Init state.<br>[2] – dl_inactive - Data Link Layer is the DL_Inactive state.<br>[3] – bad_dllp – Data link Layer received a bad DLLP.<br>[4] – dlerr – Data Link Layer Protocol error.<br>[5] – bad_tlp - Data link Layer received a bad TLP.<br>[6] – rply_tout – Indicates a replay timeout<br>[7] – rnum_rlor – Indicates replay number roll over which initiates Link re-training. |
| tx_dllp_val | Input | sys_clk_125 | Active High power message send command.<br>00 - Nothing to send<br>01 - Send DLLP using tx_pmtype DLLP<br>10 - Send DLLP using tx_vsd_data Vendor Defined DLLP<br>11 - Not used |
| tx_pmtype[2:0] | Input | sys_clk_125 | Transmit power message type<br>000 - PM Enter L1<br>001 - PM Enter L2<br>011 - PM Active State Request L1<br>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. |
| rxdp_pmd_type[2:0] | Output | sys_clk_125 | Receive power message type<br>000 - PM Enter L1<br>001 - PM Enter L2<br>011 - PM Active State Request L1<br>100 - PM Request Ack |
| rxdp_vsd_data[23:0] | Output | sys_clk_125 | Received vendor-defined DLLP data. |
| rxdp_dllp_val | Output | sys_clk_125 | Active High DLLP received |
| **TRANSACTION LAYER** | | | |
| ecrc_gen_enb | Input | Async | Active High enable for ECRC generation. When asserted, IP generates and inserts ECRC to transmitting TLPs. When asserted, the TD bit in the transmit TLP header must be set.<br>This port is available only if ECRC feature is selected while generating the IP core. |
| ecrc_chk_enb | Input | Async | Active High enable for ECRC checking. When asserted, IP checks ECRC in received TLPs.<br>This port is available only if ECRC feature is selected while generating the IP core. |
| **MESSAGE DECODES** | | | |
| int[a,b,c,d]_n | Output | sys_clk_125 | Active Low interrupt wires.<br>0 – Received Assert INTx message TLP.<br>1 – Received De-Assert INTx message TLP. |
| nftl_err_msg | Output | sys_clk_125 | Active High signal indicates receiving a NON-FATAL ERROR message TLP. |
| ftl_err_msg | Output | sys_clk_125 | Active High signal indicates receiving a FATAL ERROR message TLP. |
| corr_err_msg | Output | sys_clk_125 | Active High signal indicates receiving a CORRECTABLE ERROR message TLP. |

# Interface Description

This section describes the datapath user interfaces of the IP core. Both the transmit and receive interfaces use the TLP as the data structure.

## 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,cplh,cpld]_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. If there is enough credit, the user can proceed with sending the data based on tx_rdy_vc0. If the credit becomes insufficient, tx_req_vc0 must be de-asserted 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 will assert tx_st_vc0.

The tx_rdy_vc0 signal 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 x4 Core

Figure 2-4 through Figure 2-10 provide timing diagrams for the tx interface signals with a 64-bit datapath.

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

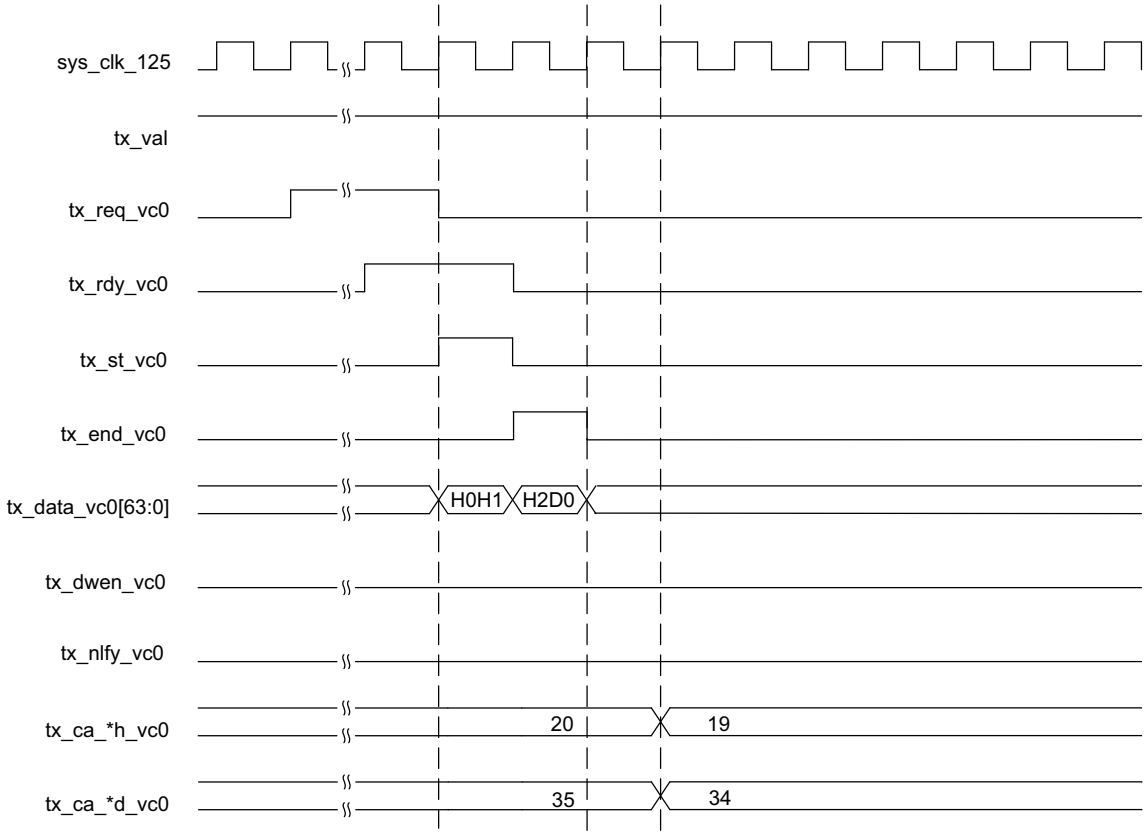


*Figure 2-5. Transmit Interface x4, 3DW Header, 2 DW Data*

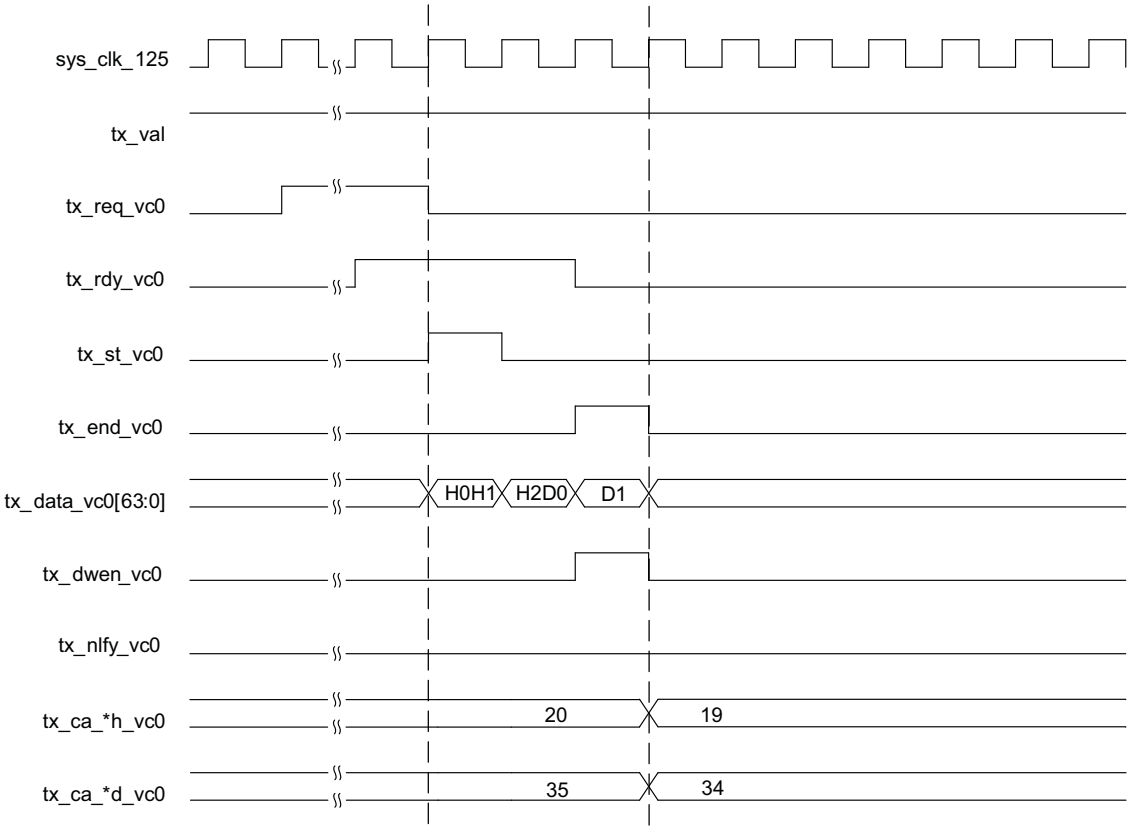


*Figure 2-6. Transmit Interface x4, 4DW Header, 0 DW*

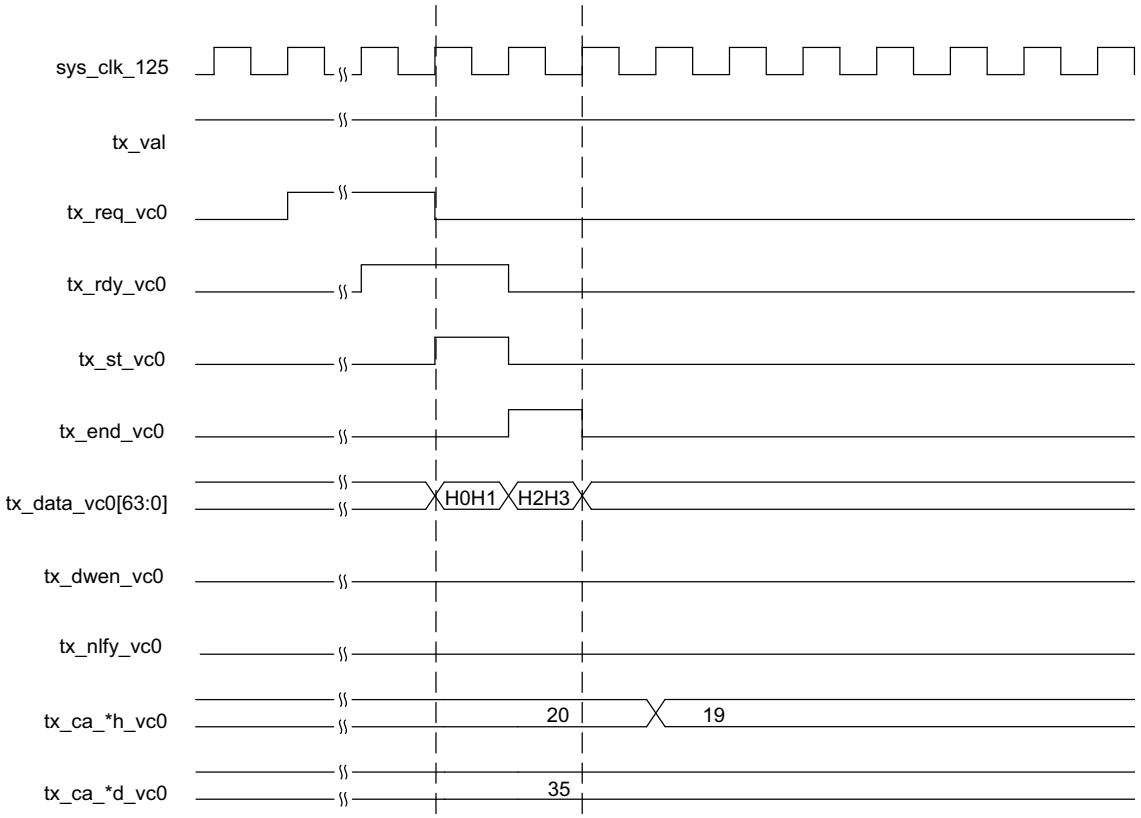


*Figure 2-7. Transmit Interface x4, 4DW Header, Odd Number of DWs*

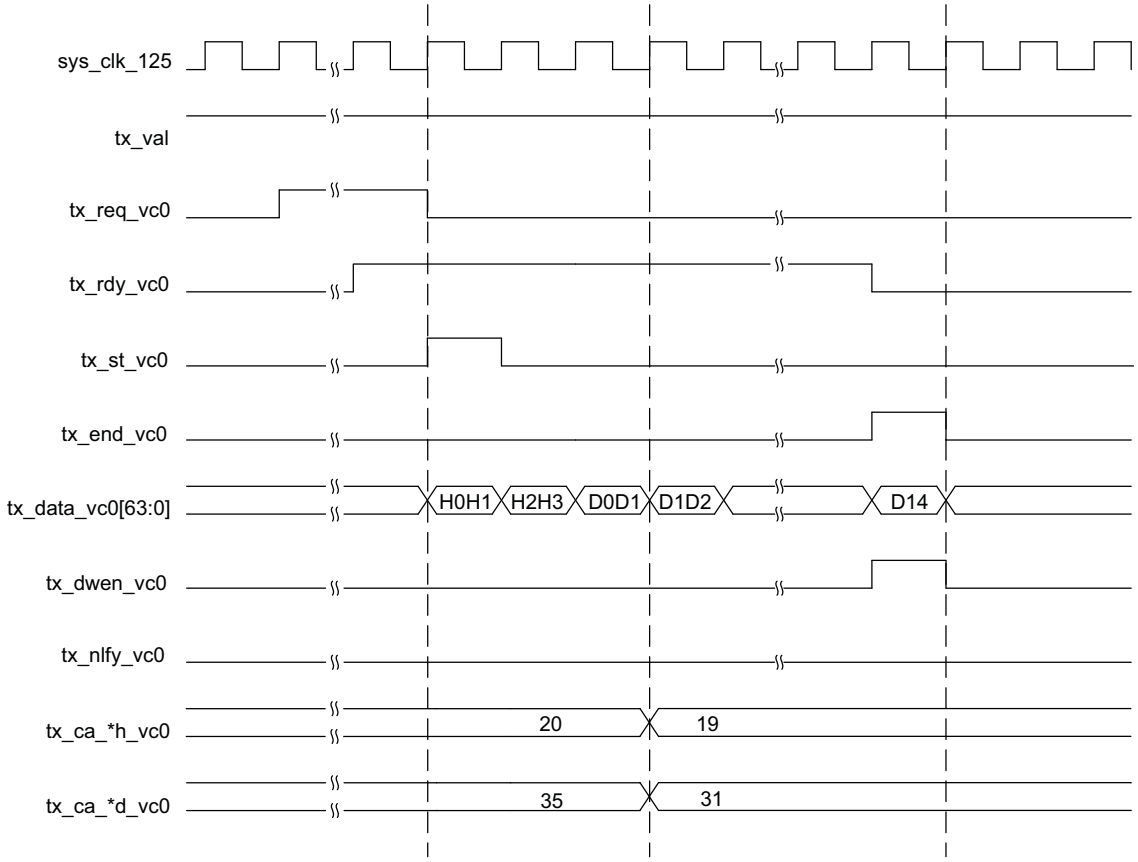


*Figure 2-8. Transmit Interface x4, Burst of Two TLPs*

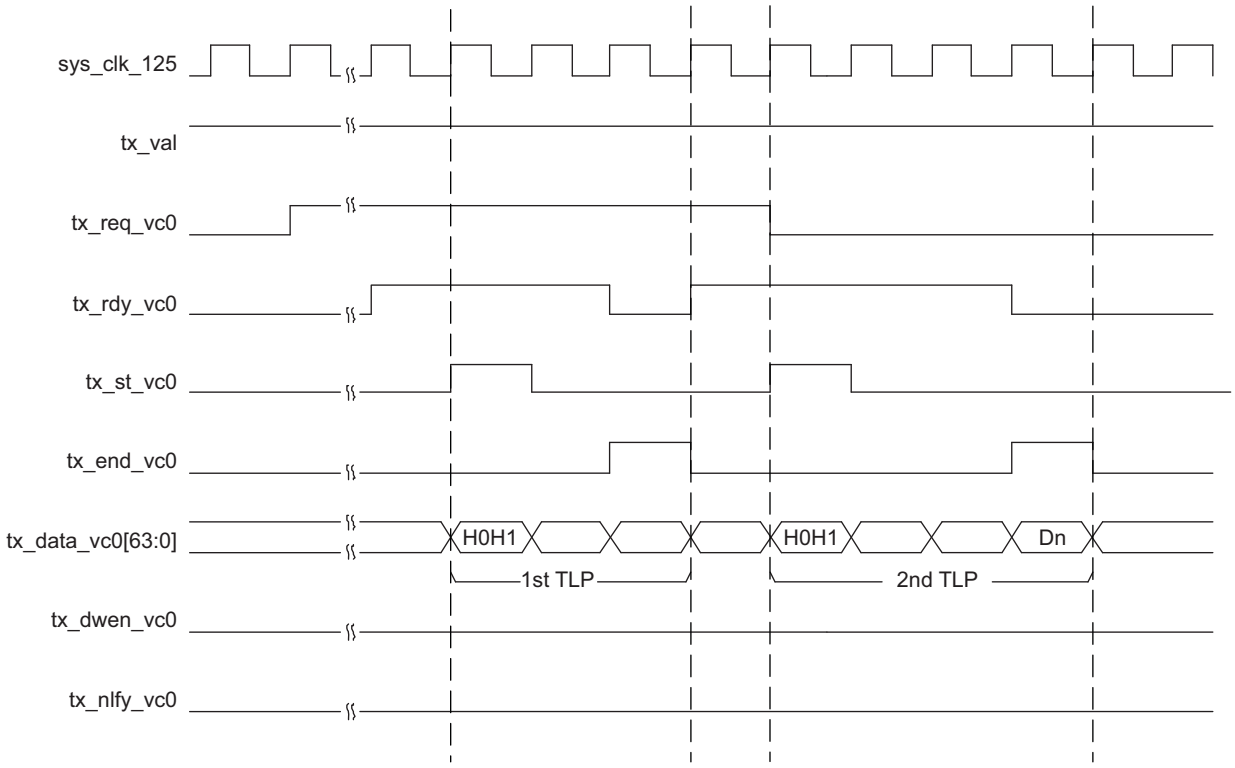



*Figure 2-9. Transmit Interface x4, Nullified TLP*

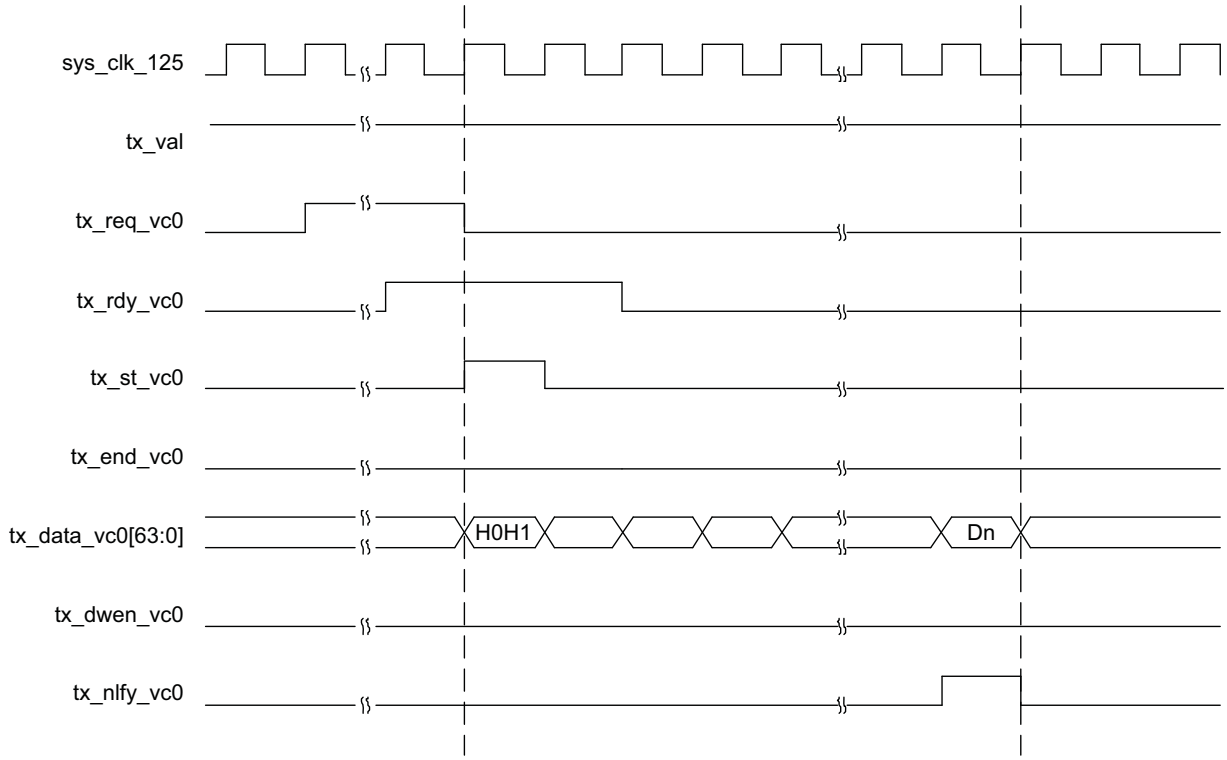
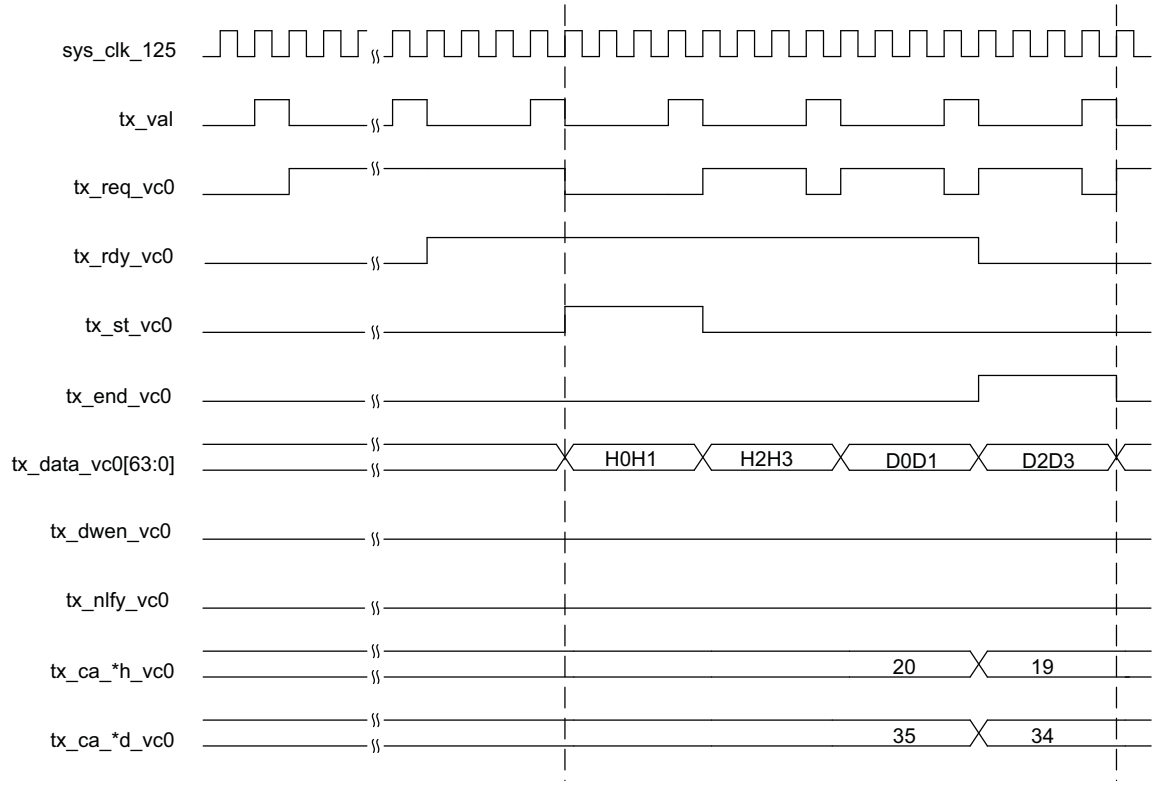


*Figure 2-10. Transmit Interface x4 Downgraded to x1 Using tx_val*




## Transmit TLP Interface Waveforms for x1

Figure 2-11 through Figure 2-13 provide timing diagrams for the transmit interface signals with a 16-bit datapath.

*Figure 2-11. Transmit Interface x1, 3DW Header, 1 DW Data*

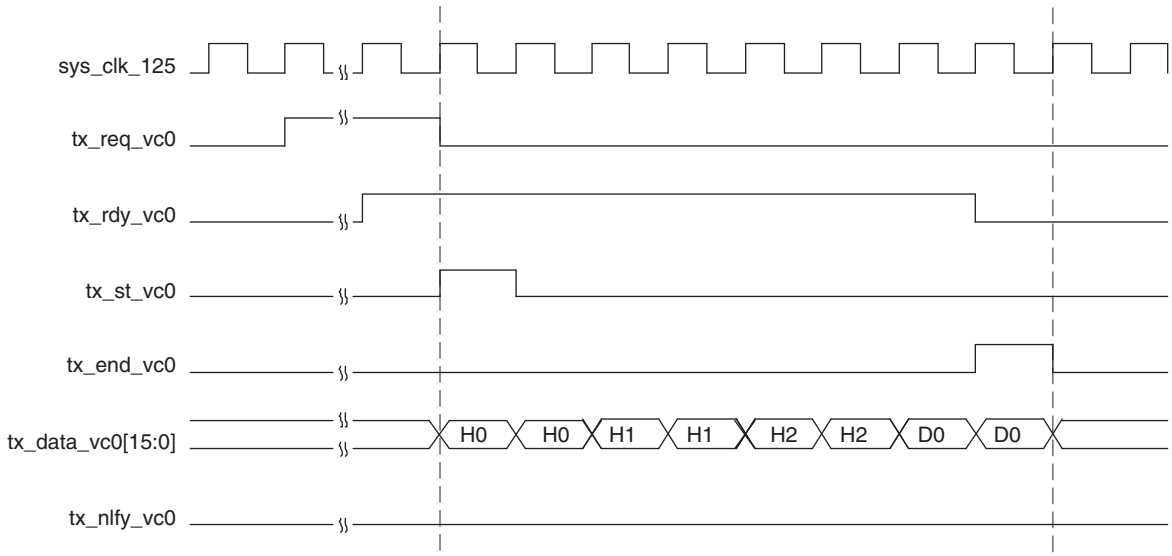


*Figure 2-12. Transmit Interface x1, Burst of Two TLPs*

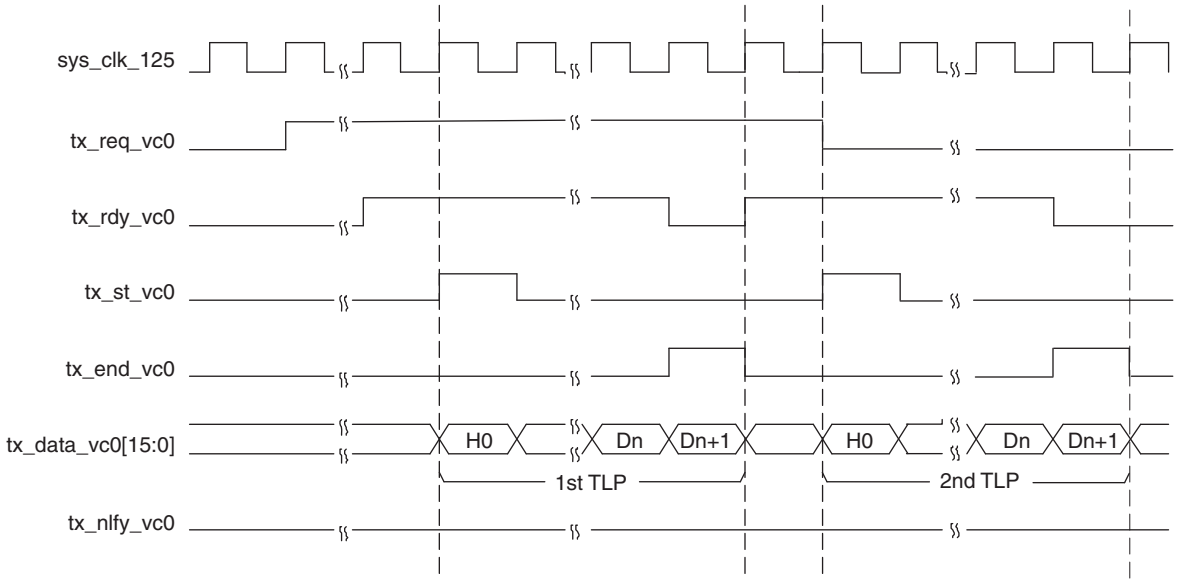
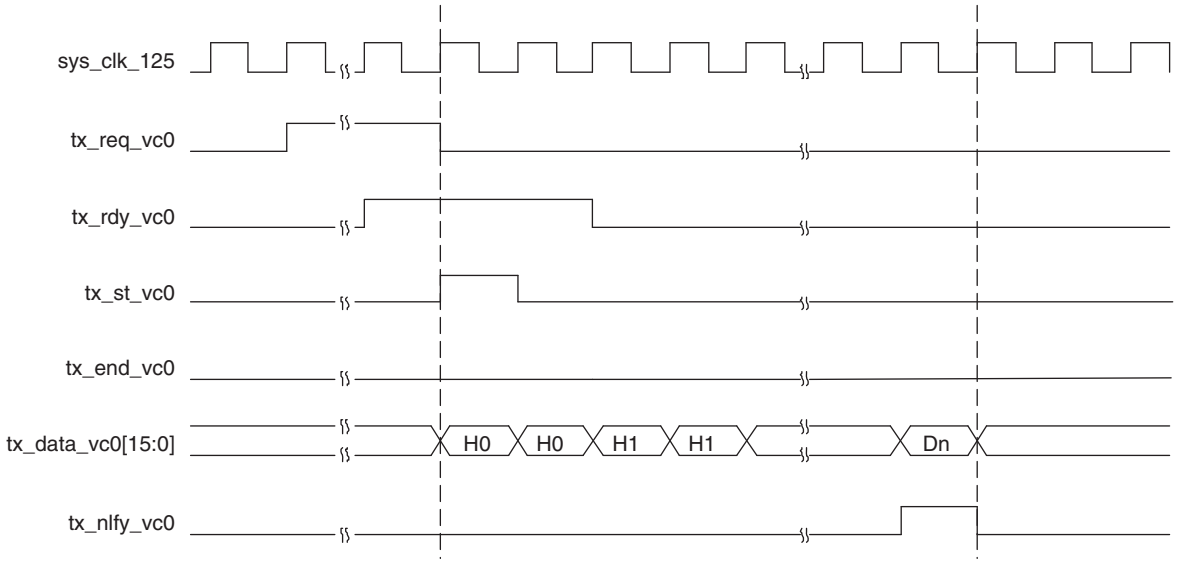



*Figure 2-13. Transmit Interface x1, Nullified TLP*




## Receive TLP Interface

In the receive direction, TLPs will come from the core as they are received on the PCI Express lanes. Interrupt messages and Error Message TLPs are processed by the IP core and corresponding signals are provided through ports. Interrupt message TLPs are decoded to ports inta_n, intb_n, intc_n and Intd_n. Error message TLPs are decoded to ports ftl_err_msg, nftl_err_msg and cor_err_msg. Figure 2-14 through Figure 2-16 provide timing diagrams of the these ports.

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 problem the rx_malf_tlp_vc0 will be asserted at any time during the TLP. Figure 2-14 through Figure 2-16 provide timing diagrams of the receive interface.

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

***Figure 2-14. Receive Interface, Clean TLP***

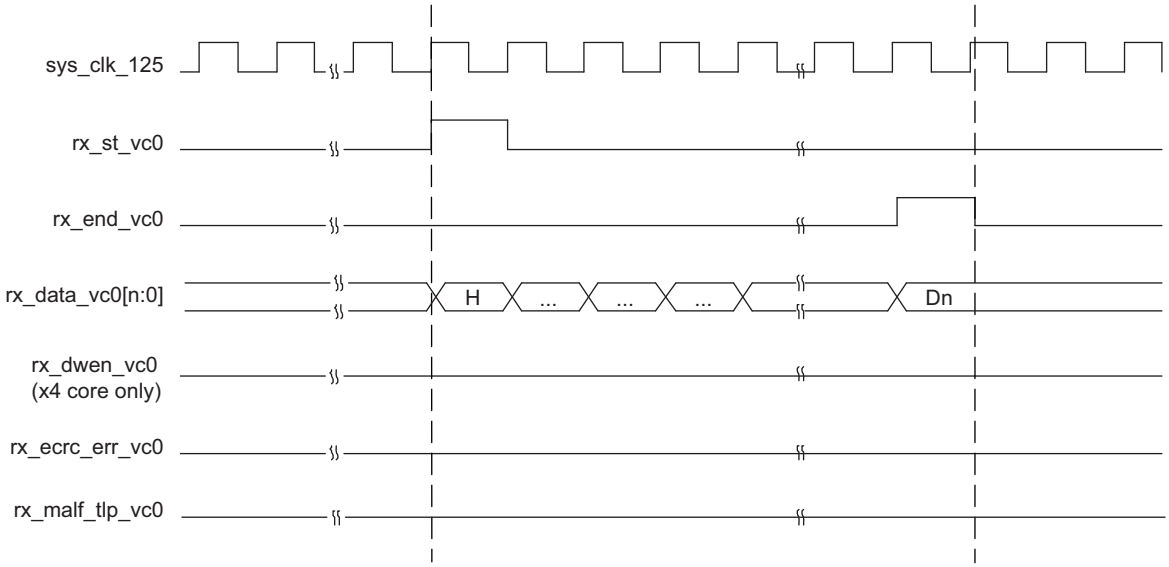



***Figure 2-15. Receive Interface, ECRC Errored TLP***

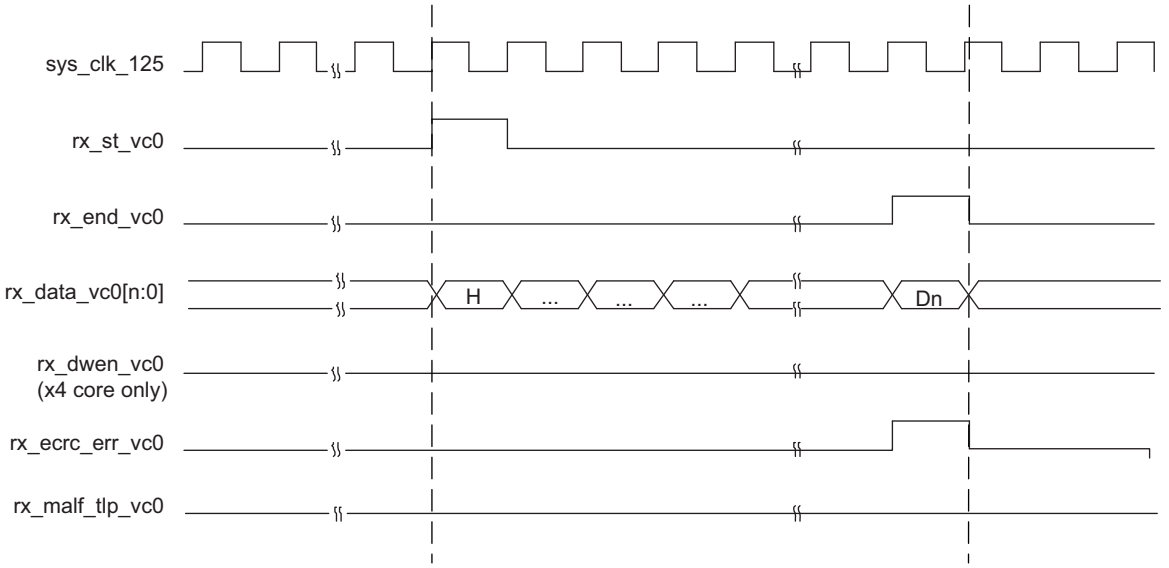
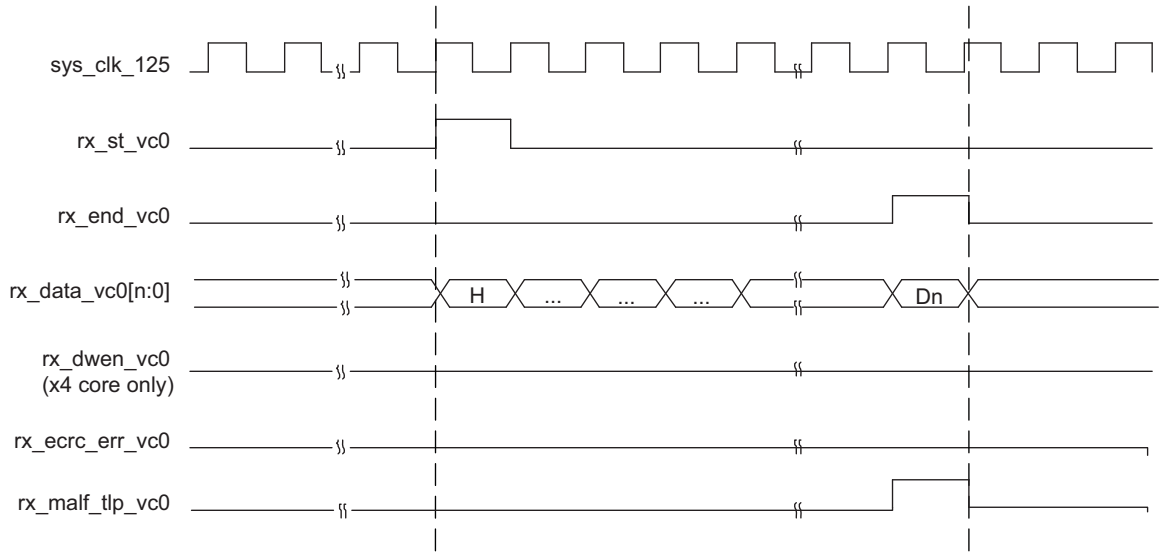


***Figure 2-16. Receive Interface, Malformed TLP***




# Message Decode Interface

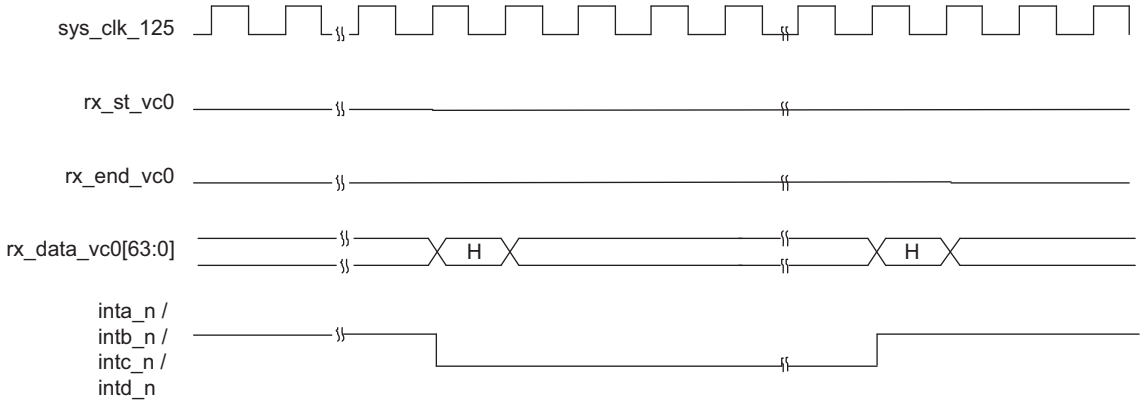
## Interrupt Signaling Messages

As a receiver, RC-Lite will decode all INTx Interrupt signaling messages and signal corresponding interrupts on ports to the user. Refer to signal descriptions of ports “inta_n, intb_n, intc_n and intd_n” in Table 2-1 on page 9.

The signals “inta_n, intb_n, intc_n and intd_n” are active Low and are set to 1’b1 after the reset. When RC-Lite receives a valid “Assert_INTA, Assert_INTB, Assert_INTC or Assert_INTD” messages, corresponding port “inta_n, intb_n, intc_n or intd_n” is reset to 1’b0 and held until a corresponding “Deassert_INTA, Deassert_INTB, Deassert_INTC or Deassert_INTD” message is received.

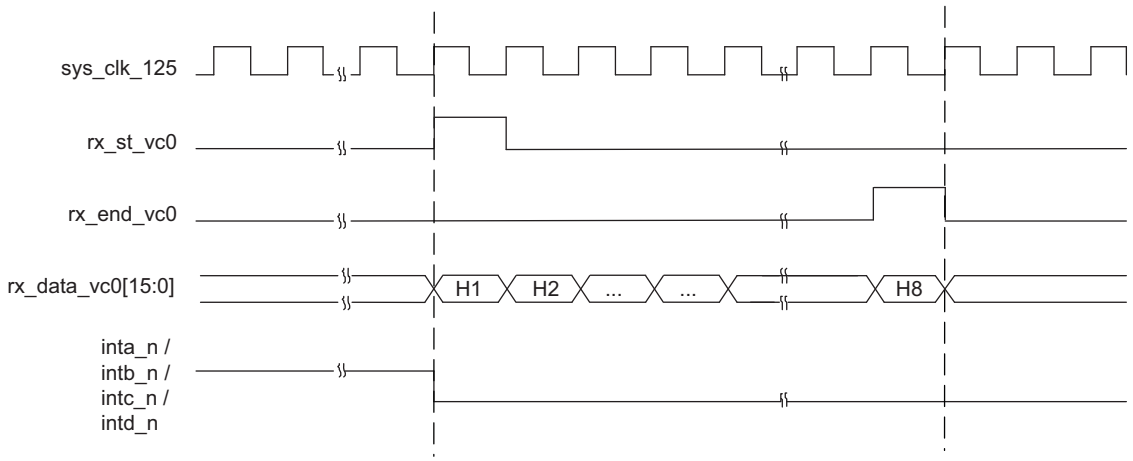
For designs using x4, whenever these ports change from 1’b1 to 1’b0 or 1’b0 to 1’b1, the data bus rx_data_vc0[63:0] contain the byte0 through byte7 of the message TLP header in the following clock cycle. Refer to Figure 2-17 for an interface diagram.

***Figure 2-17. Message Decode Interface x4, INTx signaling***




For designs using x1, the rx_st_vc0[15:0] signal will be asserted with the first word of the TLP. The remaining seven words of the interrupt message will be provided on consecutive clock cycles until the last word with rx_end_vc0 is asserted. Refer to Figure 2-18 for an interface diagram.

*Figure 2-18. Message Decode Interface x1, INTx signaling*

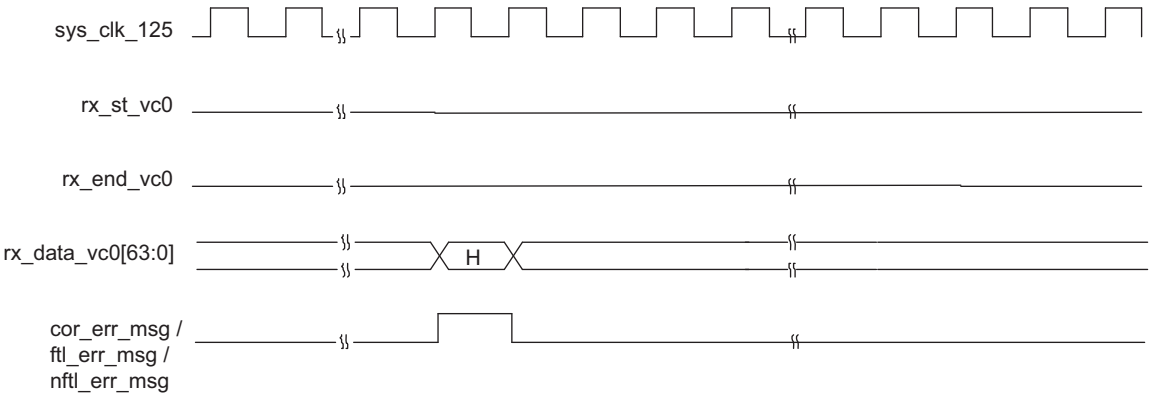



## Error Signaling Messages

As a receiver, RC-Lite will decode all Error signaling messages and signal corresponding errors on ports to the user. Refer to signal descriptions of ports “ftl_err_msg, nftl_err_msg and cor_err_msg” in Table 2-1 on page 9.

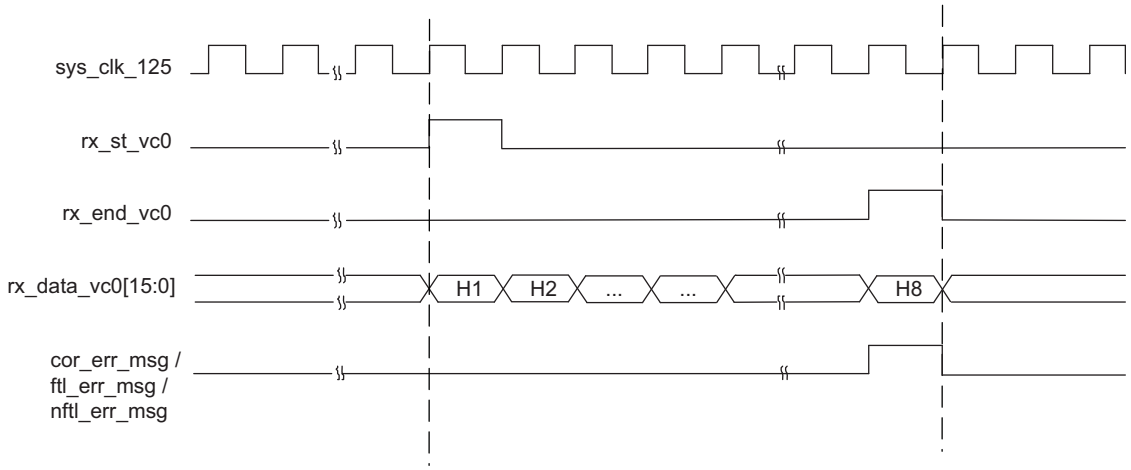
The signals “cor_err_msg, nftl_err_msg and ftl_err_msg” are active High and are set to 1’b0 after the reset. For designs using x4, when RC-Lite receives a valid “ERR_COR or ERR_NONFATAL or ERR_FATAL” message, port “cor_err_msg or nftl_err_msg or ftl_err_msg” is set to 1’b1 for one clock cycle. Whenever these ports are pulsed, the data bus rx_data_vc0[63:0] contains the byte0 through byte7 of the message TLP header. Refer to Figure 2-19 for an interface diagram.

*Figure 2-19. Message Decode Interface x4, Error signaling*




For designs using x1, rx_st_vc0[15:0] signal will be asserted with the first word of the TLP. The remaining seven words of the error message will be provided on consecutive clock cycles until the last word with both rx_end_vc0 and err_msg is asserted. Refer to Figure 2-20 for an interface diagram.

*Figure 2-20. Message Decode Interface x1, Error signaling*




## Using the Transmit and Receive Interfaces

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

### As a Receiver

As a receiver, RC-Lite can receive any type of TLP from an endpoint. When a TLP other than Interrupt or error message TLP is received, the PCI Express RC-Lite core will forward the TLP to the user interface. The rx_st_vc0 will be asserted with rx_data_vc0 providing the first eight bytes of the TLP. The TLP will terminate with the rx_end_vc0 port asserting. The user must now terminate the received TLP by release credit returns and completions for a non-posted request. The user logic must decode release credits for all received TLPs except for errored TLPs. If the core finds any errors in the current TLP, the error will be indicated on the rx_ecrc_err_vc0 or rx_malf_tlp_vc0 port.

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

If the TLP is a 32-bit MRd TLP or a 64-bit MRd TLP, the address needs to be read creating a completion TLP with the data. A Completion with Data (CplD) TLP 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 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 that the lower address of the current CplD allows the far end to piece the entire read data back together.

If the TLP is a CPL TLP, once the TLP is processed the completion credits for the CPL TLP must be released to the far end. This is done using the cplh_processed_vc0, cpld_processed_vc0, and cpld_num_vc0 ports. Each CPL TLP takes one header credit. There is one data credit used per four DWs of data. The length field provides the

number of DWs used in the TLP. If the TLP length is on the 4DW boundary, the number of credits is the TLP length divided by four. If the TLP length is not on the 4DW boundary, the number of credits is the length field + 1 (round up by 1). The number of credits used should then be placed on cpld_num_vc0[7:0]. Assert cplh_processed_vc0 and cpld_processed_vc0 for one clock cycle to latch in the cpld_num_vc0 port and release credits.

### As a Transmitter

As a transmitter, the RC-Lite can send any type of TLP to the far end endpoint. 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 four. 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 one 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.

To send Cpl TLP, the user must assemble the Cpl TLP and then check to see if the credits are available to send the TLP. The Cpl TLP without data consumes only header credit and value should be compared against the tx_ca_cplh port value. If tx_ca_cplh[8] is High, this indicates the far end has infinite credits available. The Cpl with data TLP also consumes data credits and is the length field divided by four. This value should be compared against the tx_ca_cpld port value. If tx_ca_cpld[12] is High, this indicates the far end has infinite credits available.

The IPexpress tool is used to create IP and architectural modules in the Diamond and ispLEVER software. Refer to "IP Core Generation and Evaluation" on page 35 for a description on how to generate the IP.

Table 3-1 provides the list of user configurable parameters for the PCI Express RC-Lite IP core. The parameter settings are specified using the PCI Express RC-Lite IP core Configuration GUI in IPexpress.

***Table 3-1. IP Core Parameters***

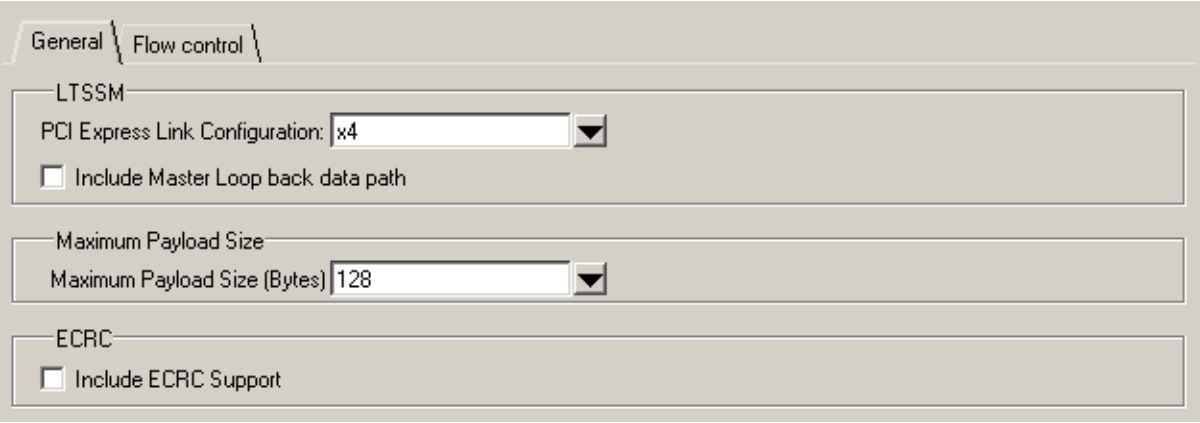
| Parameters | Range/Options | Default |
|---|---|---|
| **General** | | |
| PCI Express Link Configuration | x4 , x1 | x4 |
| Include Master Loop back data path | Yes / No | No |
| Include ECRC support | Yes / No | No |
| Maximum Payload Size (Bytes) | 128, 256, 512, 1k, 2k, 4k | 128 |
| **Flow Control** | | |
| 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 |
| Number of CPLH credits between UpdateFC CPL | 1 - 127 | 8 |
| Number of CPLD credits between UpdateFC CPL | 1 - 2047 | 255 |
| Worst case number of 125 MHz clock cycles between UpdateFC | 3650 - 4095 | 4095 |
| Infinite PH Credits | Yes / No | Yes |
| Initial PH credits available | 1 - 127 | 0 |
| Infinite PD Credits | Yes / No | Yes |
| Initial PD credits available | 8 - 255 | 0 |
| Infinite NPH Credits | Yes / No | Yes |
| Initial NPH credits available | 1 - 127 | 0 |
| Infinite NPD Credits | Yes / No | Yes |
| Initial NPD credits available | 8 - 255 | 0 |
| Infinite CPLH Credits | Yes / No | Yes |
| Initial CPLH credits available | 1 - 127 | 0 |
| Infinite CPLD Credits | Yes / No | Yes |
| Initial CPLD credits available | 8 - 255 | 0 |

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

## General Tab

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

***Figure 3-1. PCI Express RC-Lite General Tab***




The General tab consists of the following parameters:

### PCI Express Link Configuration

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

- x4 – This is a x4 link width using a 64-bit datapath. This configuration can dynamically downgrade to a x1 link width.
- x1 – This is a x1 link width using a 16-bit datapath.

### Include Master Loopback Data Path

This option includes additional transmit and receive data path ports to the IP, if the device needs to be used as a loopback master in Loopback state of the LTSSM. In Table 2-1 on page 9, refer to following I/O ports:

tx_lbk_rdy

tx_lbk_kcntl

tx_lbk_data

rx_lbk_kcntl

rx_lbk_data

### Maximum Payload Size

This option selects the maximum pay load size to be supported in the IP core and will be used to check the length of the received packets and also to size the Retry Buffer contained in the Data Link Layer. The retry buffer uses Embedded Block RAM (EBR) and will be sized accordingly. Table 3-2 provides a total EBR count for the core based on Max Payload Size.

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

| Maximum payload size | LatticeECP3/ECP2m X4 EBR usage | LatticeECP3/ECP2m X1 EBR usage |
|---|---|---|
| 128 B | 9 | 3 |
| 256 B | 9 | 3 |
| 512 B | 9 | 3 |

*Table 3-2. Total EBR Count Based on Maximum Payload Size (Continued)*

| Maximum payload size | LatticeECP3/ECP2m X4 EBR usage | LatticeECP3/ECP2m X1 EBR usage |
|---|---|---|
| 1 kB | 9 | 4 |
| 2 kB | 11 | 8 |
| 4 kB | 16 | 14 |

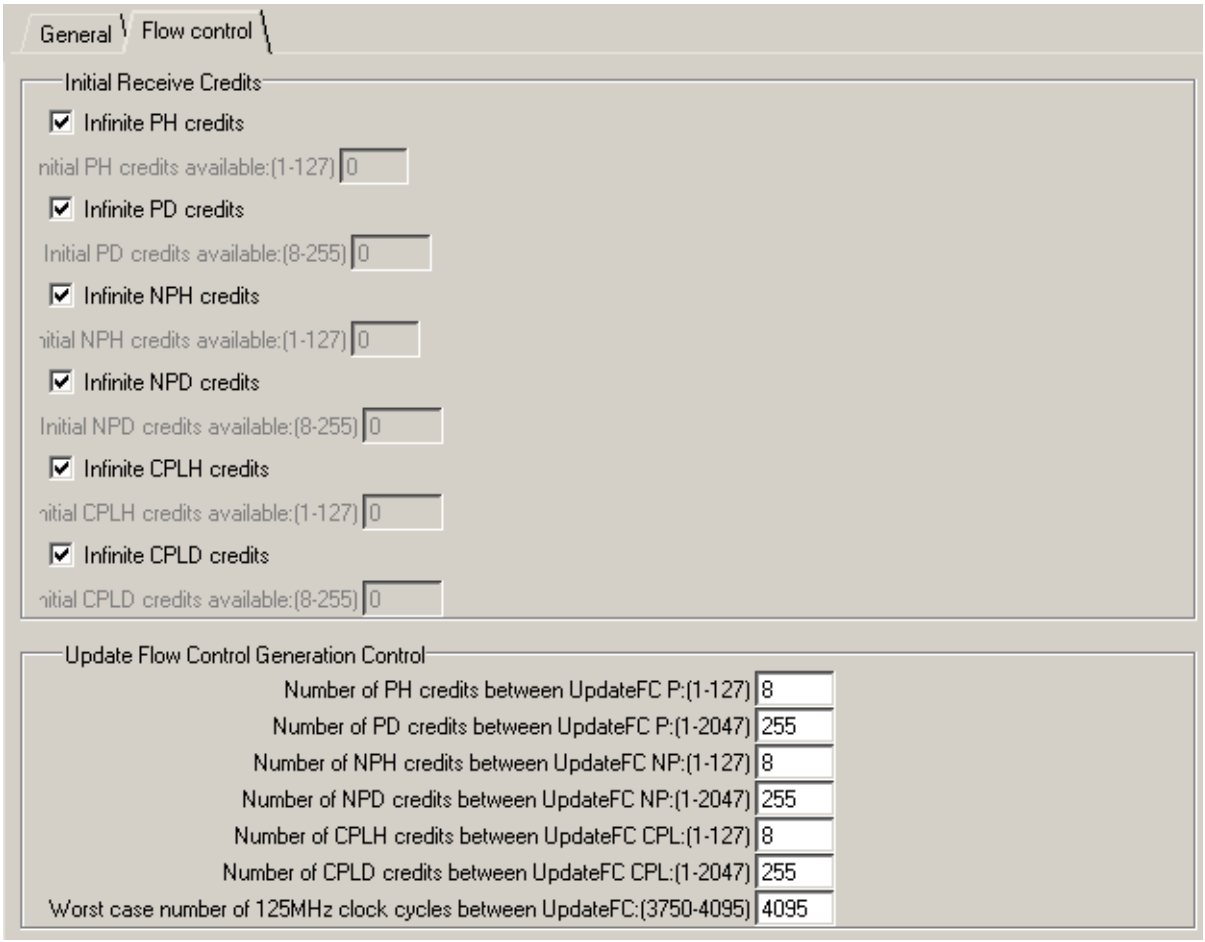
### Include ECRC Support

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.

## Flow Control Tab

Figure 3-2 shows the contents of the Flow Control tab.

*Figure 3-2. PCI Express RC-Lite Flow Control Tab*




### 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 device will have an infinite buffer for PH credits. This is typically used if the device 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 device will have an infinite buffer for PD credits. This is typically used if the device 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 device will have an infinite buffer for NPH credits. This is typically used if the device 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 device will have an infinite buffer for NPD credits. This is typically used if the device 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 is used if the device will have an infinite buffer for CPLH credits. This is typically used if the device will terminate any CPLH TLP immediately.

### 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 is used if the device will have an infinite buffer for CPLD credits. This is typically used if the device will terminate any CPLD TLP immediately.

### 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.

### 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 device. 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-NP.

### Number of CPLD TLPs Between UpdateFC

This control sets the number of completion with 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.

# IP Core Generation and Evaluation

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

The Lattice PCI Express RC-Lite IP core can be used in LatticeECP2M and LatticeECP3 device families.

## Licensing the IP Core

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

## Licensing Requirements for LatticeECP2M/LatticeECP3

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

http://www.latticesemi.com/products/intellectualproperty/aboutip/isplevercoreonlinepurchas.cfm

Users may download and generate the PCI Express RC-Lite IP core for Lattice ECP2M and LatticeECP3 and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The PCI Express RC-Lite IP core for LatticeECP2M and Lattice ECP3 also supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited time (approximately four hours) without requiring an IP license (see "Hardware Evaluation" on page 41 for further details). However, a license is required to enable timing simulation, to open the design in the Diamond or ispLEVER EPIC tool, and to generate bitstreams that do not include the hardware evaluation timeout limitation.

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

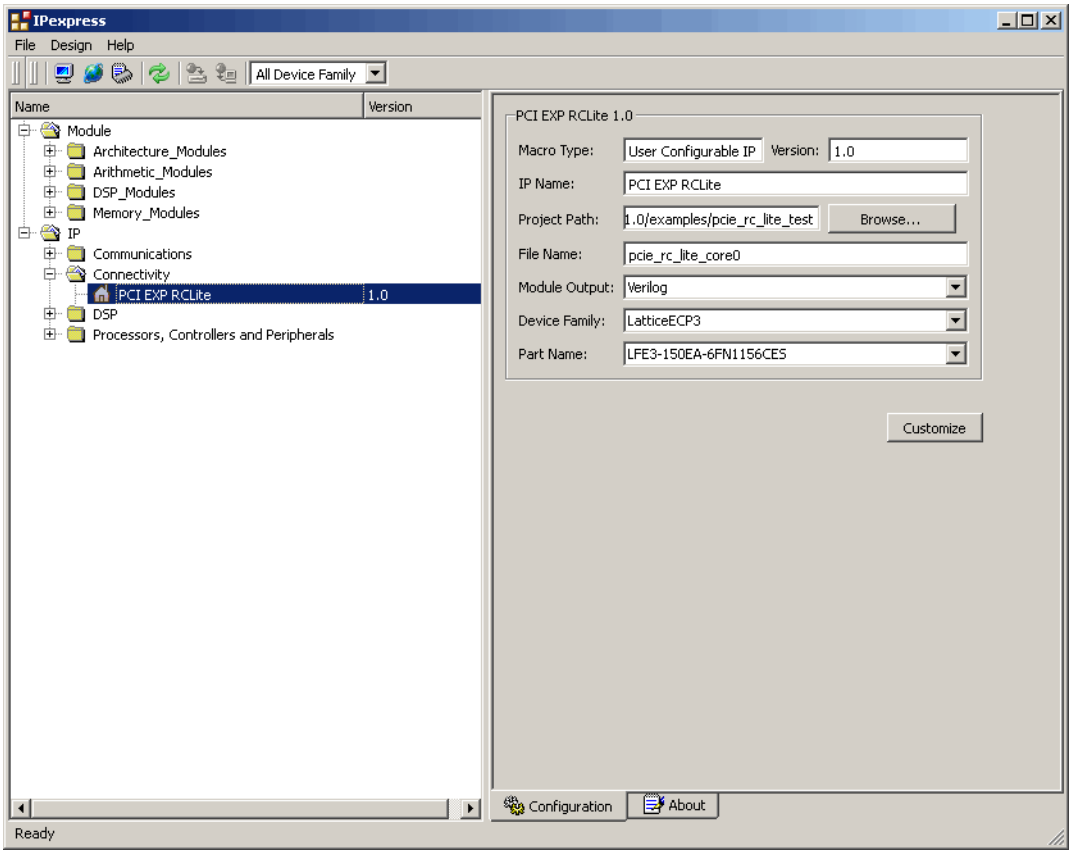
# Getting Started

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

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

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

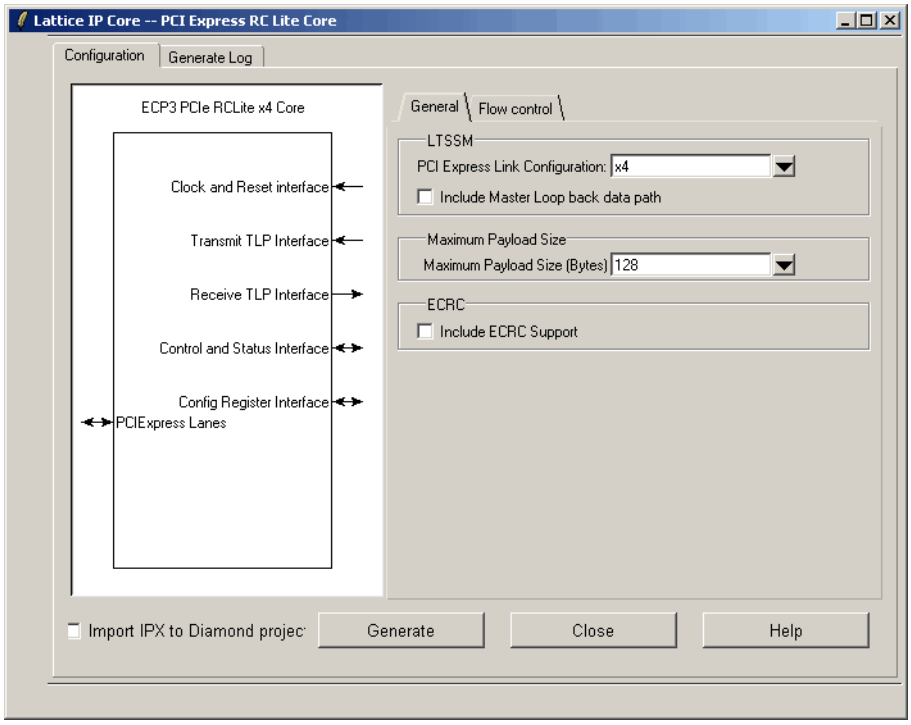
*Figure 4-1. IPexpress Dialog Box (Diamond Version)*




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

To create a custom configuration, the user clicks the **Customize** button in the IPexpress tool dialog box to display the PCI Express RC-Lite IP core Configuration GUI, as shown in Figure 4-2. From this dialog box, the user can select the IP parameter options specific to their application. Refer to "Parameter Settings" on page 30 for more information on the PCI Express RC-Lite IP core parameter settings. Additional information and known issues about the PCI Express RC-Lite IP core are provided in a ReadMe document that may be opened by clicking on the Help button in the Configuration GUI.

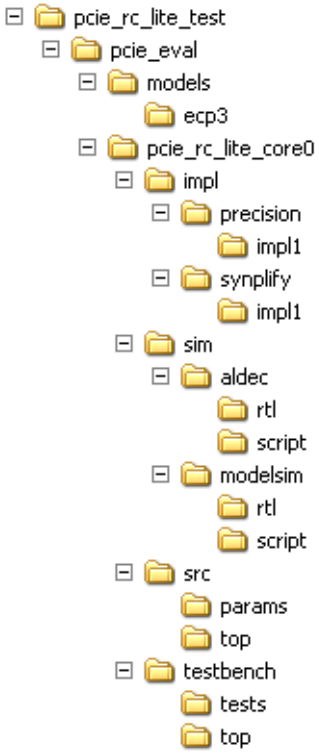
*Figure 4-2. Configuration GUI (Diamond Version)*




## IPexpress-Created Files and Top Level Directory Structure

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

*Figure 4-3. LatticeECP3 PCI Express RC-Lite IP Core Directory Structure*




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

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

*Table 4-1. File List*

| File | Sim | Synthesis/ | Description |
|---|---|---|---|
| <*username*>_inst.v | | | This file provides an instance template for the IP. |
| <*username*>.v | Yes | | This file provides the PCI Express RC-Lite IP core for simulation. |
| <*username*>_beh.v | Yes | | This file provides the front-end simulation library for the PCI Express RC-Lite IP core. |
| pci_exp_params.v | Yes | | This file provides the user options of the IP for the simulation model. |
| pci_exp_ddefines.v | Yes | | This file provides parameters necessary for the simulation. |
| <*username*>_bb.v | | Yes | This file provides the synthesis black box for the user's synthesis. |
| <*username*>.ngo | | Yes | This file provides the synthesized IP core. |
| PCS autoconfig file | Yes | Yes | This file contains the PCS/SERDES memory map initialization. This file must be copied in to the simulation directory as well as the Diamond or ispLEVER software project directory.<br>For the LatticeECP3/LatticeECP2M x4 Native and x1 Downgraded this file is named pcs_pipe_8b_x4.txt.<br>For the LatticeECP3/LatticeECP2M x1 Native this file is named pcs_pipe_8b_x1.txt. |

*Table 4-1. File List (Continued)*

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

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

The `\pcie_eval` and subtending directories provide files supporting PCI Express RC-Lite IP core evaluation. The `\pcie_eval` directory contains files/folders with content that is constant for all configurations of the PCI Express RC-Lite IP core. The `\<username>` subfolder (`\pcie_rc_lite_core0` in this example) contains files/folders with content specific to the <*username*> configuration.

The PCI Express RC-Lite IP core ReadMe document is also provided in the `\pcie_eval` directory.

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

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

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

The `\<username>` directory contains the sample design for the configuration specified by the customer. The `\<username>\impl` directory provides project files supporting Precision RTL and Synplify synthesis flows. The sample design pulls the user ports out to external pins. This design and associated project files can be used to

determine the size of the core and to push it through the mechanics of the Diamond or ispLEVER software design flow.

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

## Running Functional Simulation

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

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

- pci_exp_params.v
- pci_exp_ddefines.v

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

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

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

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

1. Open Active-HDL.
2. Under the Tools tab, select **Execute Macro**.
3. Browse to the directory \<*project_dir*>\pcie_eval\<*username*>\sim\aldec and execute the Active-HDL "do" script shown.

## Synthesizing and Implementing the Core in a Top-Level Design

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

The top-level file pcie_core0_eval_top.v provided in
\<*project_dir*>\pcie_eval\<*username*>\src\top
supports the ability to implement the PCI Express RC-Lite IP core in isolation. Push-button implementation of this top-level design with either Synplify or Precision RTL Synthesis is supported via the Diamond or ispLEVER software project files <*username*>_eval.syn located in the
\<*project_dir*>\pcie_eval\<*username*>\impl\synplify and the
\<*project_dir*>\pcie_eval\<*username*>\impl\precision directories, respectively.

*To use this project file in Diamond:*

Choose **File > Open > Project**.

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

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

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

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

*To use this project file in ispLEVER:*

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

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

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

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

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

## Hardware Evaluation

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

### Enabling Hardware Evaluation in Diamond

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

### Enabling Hardware Evaluation in ispLEVER

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

## Updating/Regenerating the IP Core

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

### Regenerating an IP Core in Diamond

*To regenerate an IP core in Diamond:*

1. In IPexpress, click the **Regenerate** button.

2. In the Regenerate view of IPexpress, choose the IPX source file of the module or IP you wish to regenerate.

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

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

## Regenerating an IP Core in ispLEVER

*To regenerate an IP core in ispLEVER:*

1. In the IPexpress tool, choose **Tools > Regenerate IP/Module**.
2. In the Select a Parameter File dialog box, choose the Lattice Parameter Configuration (.lpc) file of the IP core you wish to regenerate, and click **Open**.
3. The Select Target Core Version, Design Entry, and Device dialog box shows the current settings for the IP core in the Source Value box. Make your new settings in the Target Value box.
4. If you want to generate a new set of files in a new location, set the location in the LPC Target File box. The base of the .lpc file name will be the base of all the new file names. The LPC Target File must end with an .lpc extension.
5. Click **Next**. The IP core's dialog box opens showing the current option settings.
6. In the dialog box, choose desired options. To get information about the options, click **Help**. Also, check the About tab in the IPexpress tool for links to technical notes and user guides. The IP core might come with additional information. As the options change, the schematic diagram of the IP core changes to show the I/O and the device resources the IP core will need.
7. Click **Generate**.
8. Click the **Generate Log** tab to check for warnings and error messages.

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

## Simulation and Verification

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

### Simulation Strategies

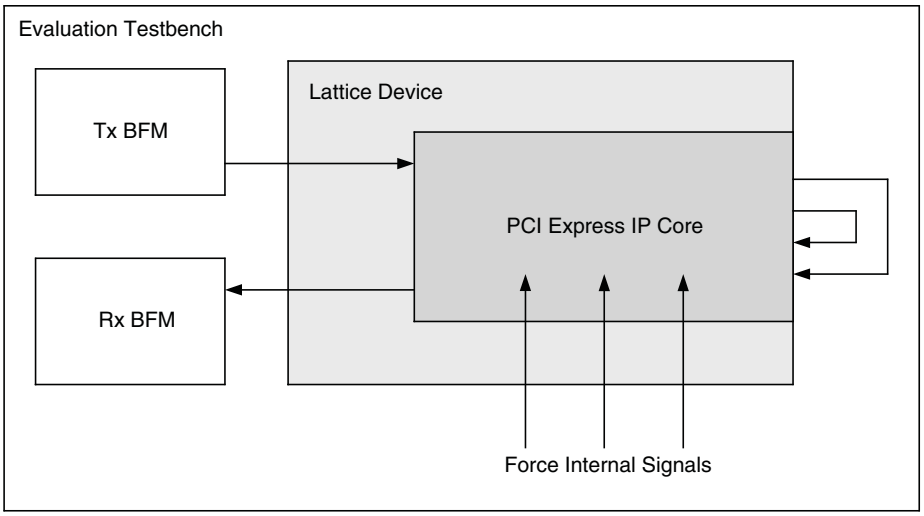
Included with the core from the IPexpress tool is the evaluation testbench located in the `<username>` directory. The intent of the evaluation testbench is to show the core performing in simulation, as well as to provide timing simulations post place and route. Many communication cores work in a loopback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

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

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

Figure 5-1 illustrates the evaluation testbench process.

***Figure 5-1. PCI Express RC-Lite IP Core Evaluation Testbench Block Diagram***




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

Sometimes the testbench is oriented differently than users anticipate. Users might wish to interface to the PCI Express RC-Lite IP 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.

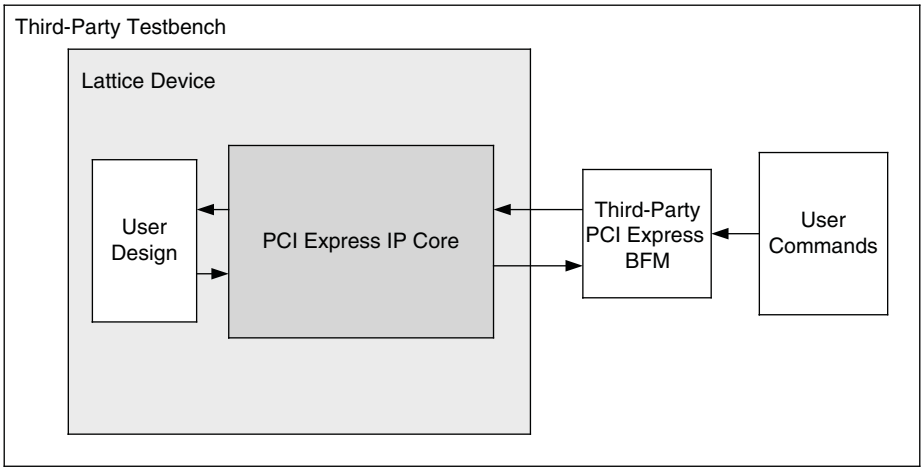
Users simulating a multi-lane core at the serial level should give consideration to lane ordering. Lane ordering is dependant on the layout of the chip on a board. Refer to "Board Layout Concerns" on page 52 for further information.

### Third Party Verification IP

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

Lattice has chosen the Synopsys PCI Express verification IP for development of the PCI Express RC-Lite IP core, as shown in Figure 5-2. There are other third party vendors for PCI Express RC-Lite IP core including Denali® and Cadence.

***Figure 5-2. PCI Express RC-Lite IP Core Testbench with Third-Party VIP***




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

## FPGA Design Implementation

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

## Setting Up the Core

This section describes how to set up the PCI Express RC-Lite IP 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 bit stream loading using the autoconfig file generated with the IPexpress tool.

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

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

### Setting Up for x4 (No Flip)

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

## Setting Up for x4 (Flipped)

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

## Setting Design Constraints

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

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

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

There are also several places inside the PCI Express RC-Lite IP 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.

```
BLOCK PATH FROM CELL "*ctc_reset_chx*";
BLOCK PATH FROM CELL "*core_rst_n*";
BLOCK NET "*rst_n_c*";
MULTICYCLE FROM CELL "*nfts_rx_skp_cnt*" TO CELL "*cnt_done_nfts_rx*" 2 X;
MULTICYCLE FROM CELL "*nfts_rx_skp_cnt*" TO CELL "*ltssm_nfts_rx_skp*" 2 X;
```

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

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

## Errors and Warnings

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

Place & Route Design

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

```
WARNING - par: (user pref. primary clock) Signal
          "u1_pcie_top/u1_pcs_pipe/ff_rx_fclk_0" is selected as a primary
          clock;  however, according to the architecture, the driver of this
          signal cannot drive the clock tree directly, therefore general
          routing has to be used and the design may experience increased
          injection delay.
WARNING - par: (user pref. primary clock) Signal
          "u1_pcie_top/u1_pcs_pipe/ff_rx_fclk_1" is selected as a primary
          clock;  however, according to the architecture, the driver of this
```

```
          signal cannot drive the clock tree directly, therefore general
          routing has to be used and the design may experience increased
          injection delay.
WARNING - par: (user pref. primary clock) Signal
          "u1_pcie_top/u1_pcs_pipe/ff_rx_fclk_2" is selected as a primary
          clock;  however, according to the architecture, the driver of this
          signal cannot drive the clock tree directly, therefore general
          routing has to be used and the design may experience increased
          injection delay.
WARNING - par: (user pref. primary clock) Signal
          "u1_pcie_top/u1_pcs_pipe/ff_rx_fclk_3" is selected as a primary
          clock;  however, according to the architecture, the driver of this
          signal cannot drive the clock tree directly, therefore general
          routing has to be used and the design may experience increased
          injection delay.
```

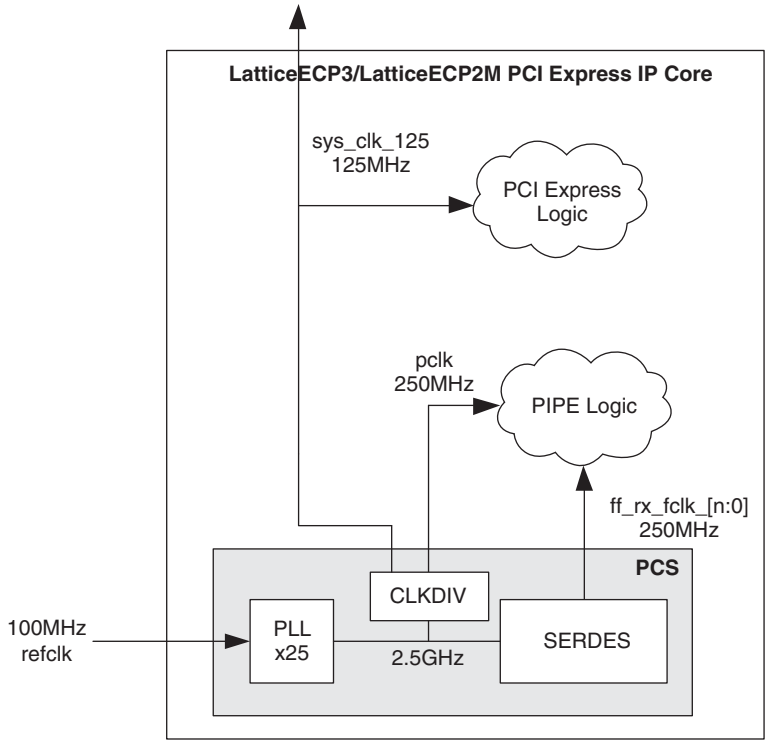
These warnings inform the user that the receive clocks from the PCS module are using general routing to connect to a primary clock connection point. This is expected for this architecture and clock connection.

## 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 RC-Lite IP core is clocked using a 125MHz clock (sys_clk_125).

Figure 5-3 provides the internal clocking structures of the IP core in the LatticeECP3 and LatticeECP2M families.

*Figure 5-3. LatticeECP3 and LatticeECP2M PCI Express Clocking Scheme*




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

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

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

*Table 5-1. LatticeECP3 and LatticeECP2M Clocking Resources Used*

| Type | Number | Notes |
|---|---|---|
| Primary Clocks | 3/6 | 3 for x1 interface.<br>6 for x4 interface. |

## Locating the IP

The PCI Express RC-Lite 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 core in a defined location on the device array. The hard blocks' fixed locations will drive the location of the IP. Table 5-2 lists the site names for the hard blocks on the different device arrays.

*Table 5-2. PCS Location Site Names for Lattice Devices*

| Device | PCS |
|---|---|
| LFE3-17 | PCSA |
| LFE3-35 | PCSA |

*Table 5-2. PCS Location Site Names for Lattice Devices*

| Device | PCS |
|---|---|
| LFE3-70[1] | PCSA<br>PCSB<br>PCSC |
| LFE3-95[1] | PCSA<br>PCSB<br>PCSC |
| LFE3-150[1] | PCSA<br>PCSB<br>PCSC<br>PCSD |
| LFE2M20 | URPCS |
| LFE2M35 | URPCS |
| LFE2M50 | URPCS<br>LRPCS |
| LFE2M70 | URPCS<br>ULPCS<br>LRPCS |
| LFE2M100 | URPCS<br>ULPCS<br>LRPCS<br>LLPCS |

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

### Locating the Hard Elements

Figure 5-4 provides a block diagram with placement positions of the PCS/SERDES quads in the LatticeECP3 devices.

*Figure 5-4. LatticeECP3 Device Arrays with PCS/SERDES*

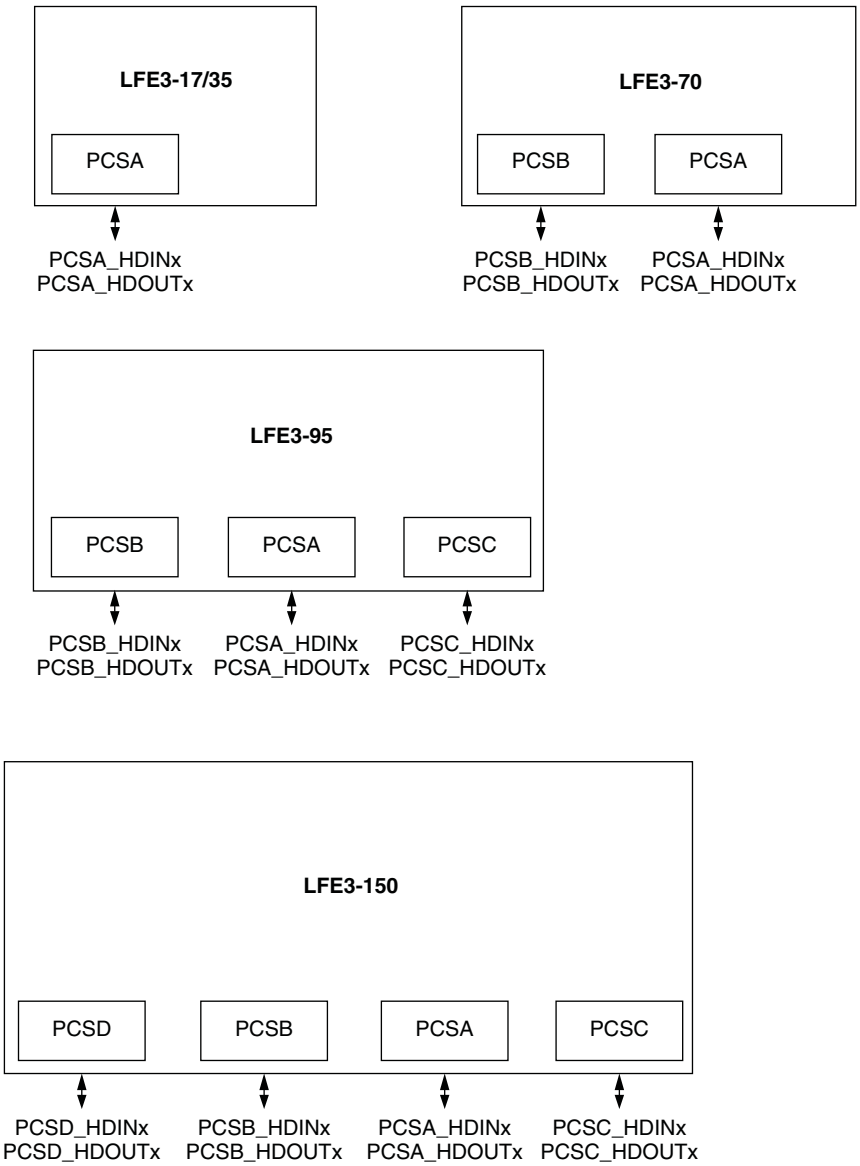
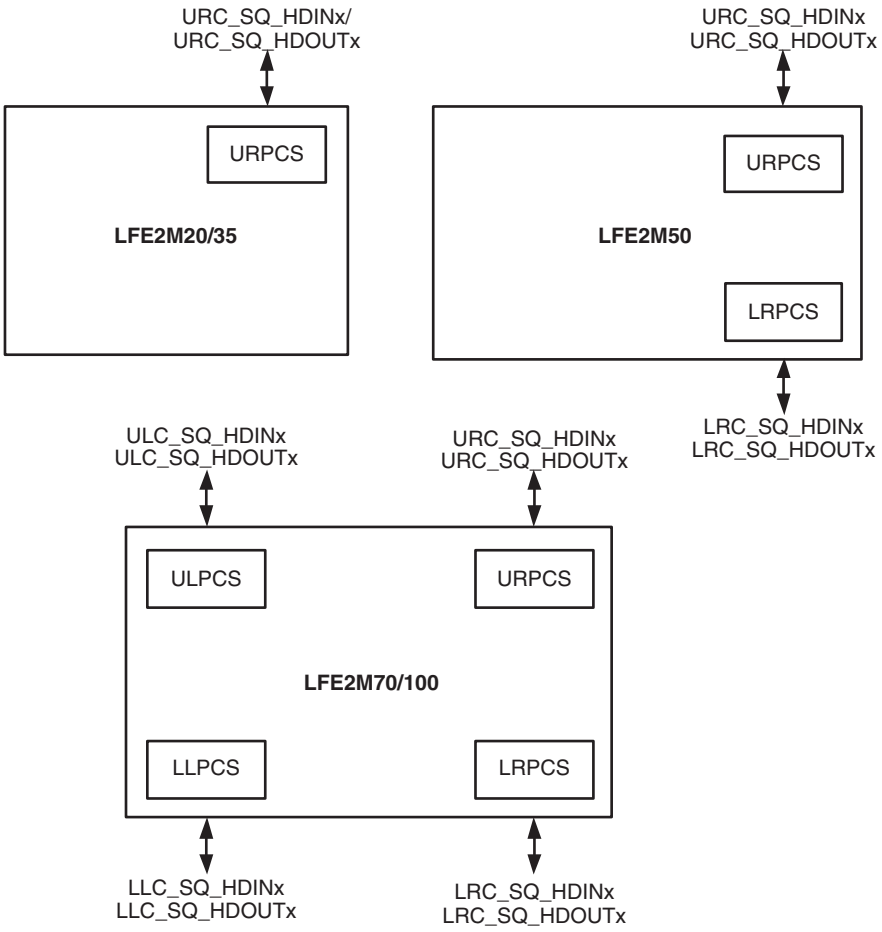



Figure 5-5 provides a block diagram with placement positions of the PCS/SERDES quads in the LatticeECP2M devices.

*Figure 5-5. 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 RC-Lite IP core interface on the board layout. Below is an example of locating the PCS in a LatticeECP2M device.

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

## Board-Level Implementation Information

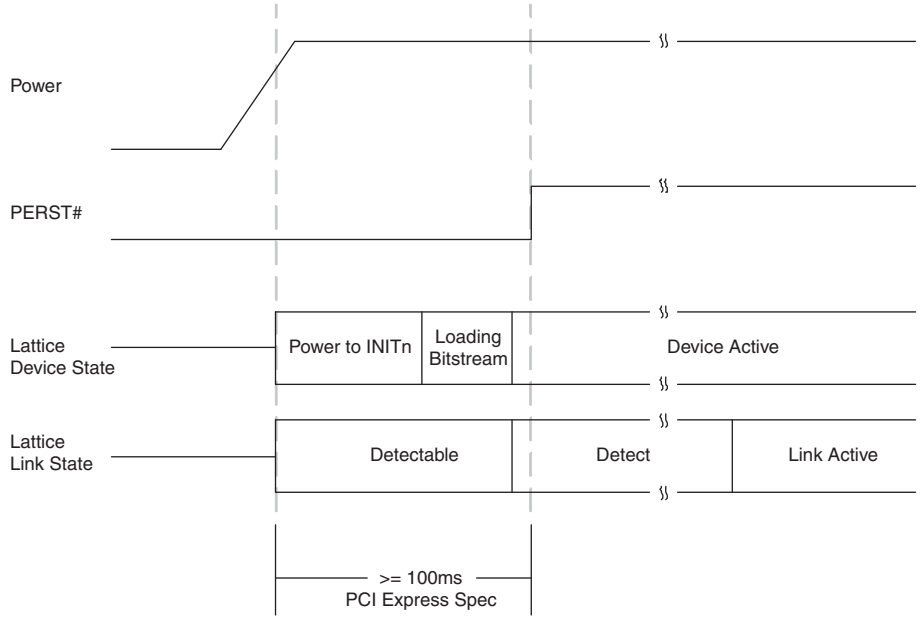
This section provides circuit board-level requirements and constraints associated with using the PCI Express RC-Lite IP core.

### PCI Express Power-Up

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

From the point of power stable to at least 100 ms the PERST# must remain asserted. Different PCI Express systems will hold PERST# longer than 100 ms, but the minimum time is 100 ms. Shown below in Figure 5-6 is a best case timing diagram of the Lattice device with respect to PERST#.

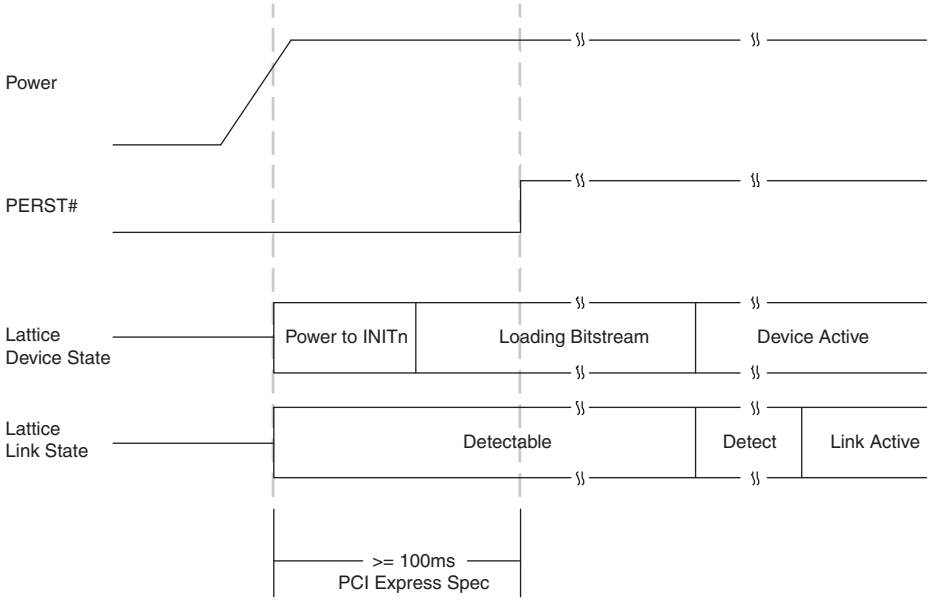
*Figure 5-6. Best Case Timing Diagram, Lattice Device with Respect to PERST#*




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

In some Lattice devices the device will not finish loading the bitstream until after the PERST# has been released. Figure 5-7 shows a worst case timing diagram of the Lattice device with respect to PERST#.

*Figure 5-7. Worst Case Timing Diagram, Lattice Device with Respect to PERST#*




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

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

In order to implement a power-up strategy using Lattice devices, Table 5-3 contains the relative numbers for the LatticeECP2M families.

***Table 5-3. LatticeECP2M Power Up Timing Specifications***

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

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

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

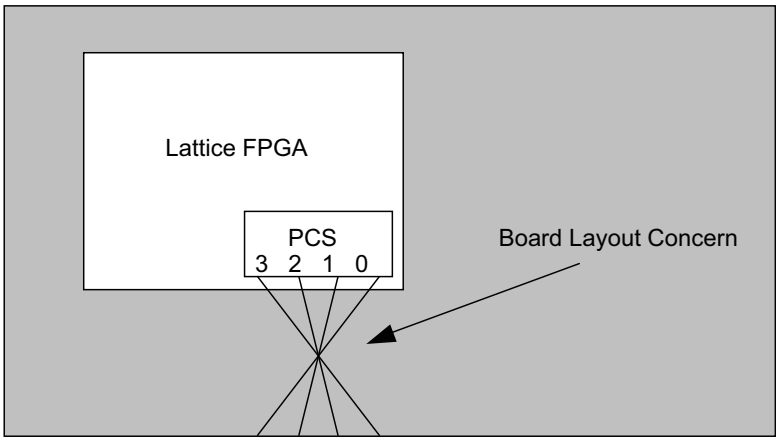
To reduce the bitstream loading time of the Lattice device a parallel Flash device and CPLD device can be used. The use of parallel Flash devices and Lattice devices is documented in AN8077, *Parallel Flash Programming and FPGA Configuration.*

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

## Board Layout Concerns

For multi-lane implementations there might be a layout concern in making the connection on board for a particular orientation of lanes. On some packages lane 0 of board will align with lane 0 of the SERDES and likewise for channels 1, 2 and 3. However, in other packages lane 0 of board will need to cross lanes 1, 2 and 3 to connect to lane 0 of the SERDES. It will not be possible to follow best practice layout rules and cross SERDES lanes in the physical board design. Figure 5-8 provides an example of the board layout concern.

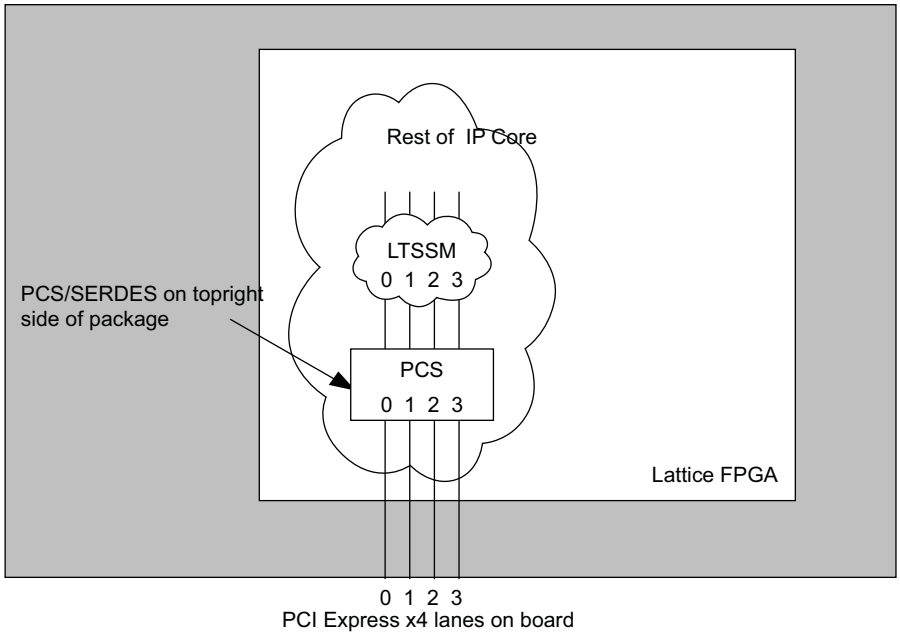
***Figure 5-8. Example of Board Layout Concern with x4 Link***

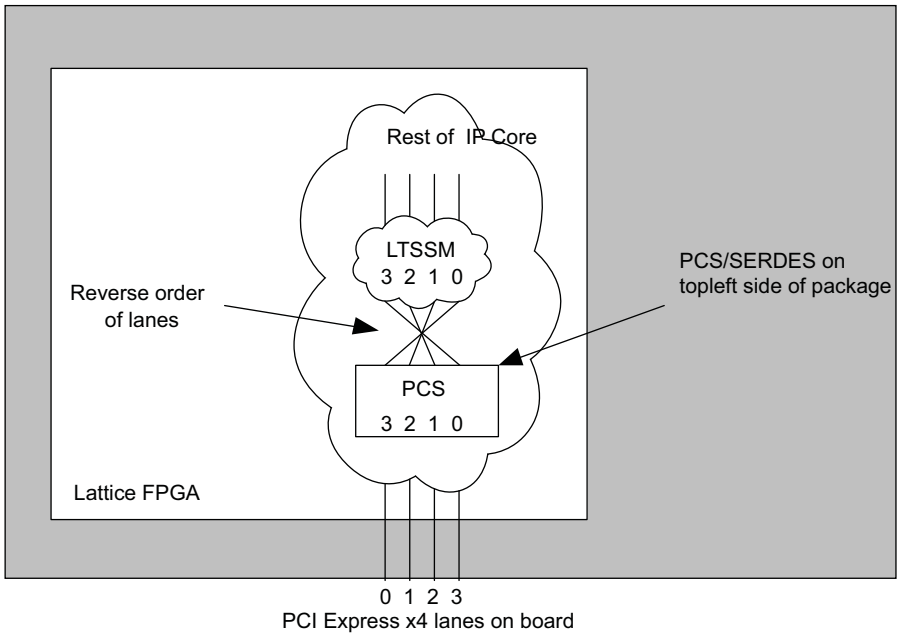



To accommodate this layout dilemma, the Lattice PCI Express solution provides an option to reverse the order of the SERDES lanes to the LTSSM block of the PCI Express RC-Lite IP core. This allows the board layout to connect board lane 0 to SERDES lane 3, board lane 1 to SERDES lane 2, board lane 2 to SERDES lane 1, and board lane

3 to SERDES lane 0. The PCI Express RC-Lite IP core will then perform a reverse order connection so the PCI Express board 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 RC-Lite IP core in the reversed orientation. The user must be aware when routing the high speed serial lines that this change has taken place. PCI Express lane 0 will need to connect to SERDES channel 3, etc. Figure 5-9 provides a diagram of a normal and a reversed IP core implementation.

***Figure 5-9. Implementation of x4 IP Core to Edge Fingers***




As shown in Figure 5-9, 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.

### PIPE Simulation

The LatticeECP3 and LatticeECP2M PCI Express simulation also requires the PIPE module. This simulation model is found in the `<username>_eval/models/pcs_pipe_top.v` file. In the same directory are a few other files that are 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/PCSD.v

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

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

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

# Compile the user design
vlog top.v


# Compile the testbench
vlog tb.v

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


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

### Simulation Behavior

When setting the SIMULATE variable for the simulation model of the PCI Express RC-Lite IP core several of the LTSSM counters are reduced. Table 5-4 provides the new values for each of the LTSSM counters when the SIMULATE variable is defined.

***Table 5-4. LTSSM Counters***

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

## Troubleshooting

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

***Table 5-5. Troubleshooting***

| Symptom | Possible Reason | Troubleshooting |
|---|---|---|
| LTSSM does not transition to L0 state. | 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 device cannot complete a transaction started by the application software, the software will “hang” waiting on the device. |
| PC crashes. | Endpoint might have violated the available credits of the root complex. | A system crash usually implies a hardware failure. If the device violates the number of credits available, the root complex can throw an exception which can crash the machine. |
| | Endpoint might have created a NAK or forced a retrain. | Certain motherboards are forgiving of a NAK or LTSSM retrain while others are not. A retrain can be identified by monitoring the phy_ltssm_state vector from the PCI Express core to see if the link falls from the L0 state during operation. |
| Device stops working after hours of operation. | The hardware timer is included in the device. | The hardware timer is utilized when an IP core does not have a license. The hardware timer holds the device in reset after a few hours of operation. Powering the device off and then on will restore functionality for the period of time dictated by the hardware timer. |

# Core Verification

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

Simulation environment verifying proper PCI Express Root complex functionality when testing with a Synopsys DesignWare behavioral model in Endpoint mode.

Using the Agilent E2960A PCI Express Protocol Tester and Analyzer for analyzing and debugging PCI Express bus protocol. The Tester is used for sending and responding to PCI Express traffic from the DUT.

# Support Resources

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

## Lattice Technical Support

There are a number of ways to receive technical support.

### Online Forums

The first place to look is Lattice Forums (http://www.latticesemi.com/support/forums.cfm). Lattice Forums contain a wealth of knowledge and are actively monitored by Lattice Applications Engineers.

### Telephone Support Hotline

Receive direct technical support for all Lattice products by calling Lattice Applications from 5:30 a.m. to 6 p.m. Pacific Time.

- For USA & Canada: 1-800-LATTICE (528-8423)
- For other locations: +1 503 268 8001

In Asia, call Lattice Applications from 8:30 a.m. to 5:30 p.m. Beijing Time (CST), +0800 UTC. Chinese and English language only.

- For Asia: +86 21 52989090

### E-mail Support

- techsupport@latticesemi.com
- techsupport-asia@latticesemi.com

### Local Support

Contact your nearest Lattice Sales Office.

### Internet

www.latticesemi.com

### PCIe Solutions Web Site

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

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

### PCI-SIG Website

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

http://www.pcisig.com.

# References

## LatticeECP3

- HB1009, *LatticeECP3 Family Handbook*

## LatticeECP2M

- HB1003, *LatticeECP2M Family Handbook*
- TN1114, *Electrical Recommendations for Lattice SERDES*
- TN1165, *PCI Express and SGMII/GbE Applications in the Same LatticeECP2M SERDES Quad*
- TN1166, *PCI Express SIG Compliance Overview for Lattice Semiconductor FPGAs*
- AN8077, *Parallel Flash Programming and FPGA Configuration*

# Revision History

| Date | Document Version | IP Core Version | Change Summary |
|---|---|---|---|
| November 2010 | 01.0 | 1.0 | Initial release. |
| February 2012 | 01.1 | 1.0 | Updated document with new corporate logo. |
| | | | PCI Express RC-Lite IP Core Quick Facts table – Updated information for the minimal device needed. |
| | | | PCI Express RC-Lite IP Core Port List – Updated description for tx_val. |
| | | | PCI Express RC-Lite IP Core Port List – Updated description for phy_cfgln_sum[2:0]. |

# Resource Utilization

This appendix gives resource utilization information for Lattice FPGAs using the PCI Express RC-Lite IP core.

## Configuration

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

## LatticeECP2M Utilization (x4 RC-Lite)

Table A-1 shows the resource utilization for the PCI Express x4 RC-Lite IP core implemented in a LatticeECP2M FPGA. Table A-2 lists the parameter settings for the IP core configuration shown in Table A-1.

***Table A-1. Resource Utilization[1]***

| IPexpress Configuration[1] | Slices | LUTs | Registers | sysMEM EBRs |
|---|---|---|---|---|
| Config 1 | 8185 | 10889 | 8469 | 9 |

1. Performance and utilization data are generated targeting an LFE2M-50E-6F900C using Lattice Diamond 1.1 and Synplify Pro D- 2009.12L-1 software. Performance might vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2M family.

## Ordering Part Number

The Ordering Part Number (OPN) for the PCI Express x4 RC-Lite IP core targeting LatticeECP2M devices is PCI-ERC4-PM-U1.

***Table A-2. Parameter Settings for Config Demo***

| | Config 1 |
|---|---|
| PCI express Link | x4 |
| Maximum payload size supported | 128 Bytes |
| Include ECRC support | No |
| Other parameters | Default |

## LatticeECP3 Utilization (x4 RC-Lite)

Table A-3 shows the resource utilization for the PCI Express x4 RC-Lite IP core implemented in a LatticeECP3 FPGA. Table A-4 lists the parameter settings for the IP core configuration shown in Table A-3.

***Table A-3. Resource Utilization[1]***

| IPexpress Configuration[1] | Slices | LUTs | Registers | sysMEM EBRs |
|---|---|---|---|---|
| Config 1 | 7703 | 10608 | 8460 | 9 |

1. Performance and utilization data are generated targeting an LFE3-95E-7FN1156CES using Lattice Diamond 1.1 and Synplify Pro D- 2009.12L-1 software. Performance might vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family.

## Ordering Part Number

The Ordering Part Number (OPN) for the PCI Express x4 RC-Lite IP core targeting LatticeECP3 devices is PCI-ERC4-E3-U1.

*Table A-4. Parameter Settings for Config Demo*

| | Config 1 |
|---|---|
| PCI express Link | x4 |
| Maximum payload size supported | 128 Bytes |
| Include ECRC support | No |
| Other parameters | Default |

## LatticeECP2M Utilization (x1 RC-Lite)

Table A-1 shows the resource utilization for the PCI Express x1 RC-Lite IP core implemented in a LatticeECP2M FPGA. Table A-2 lists the parameter settings for the IP core configuration shown in Table A-1.

*Table A-5. Resource Utilization[1]*

| IPexpress Configuration[1] | Slices | LUTs | Registers | sysMEM EBRs |
|---|---|---|---|---|
| Config 1 | 3260 | 4770 | 3096 | 3 |

1. Performance and utilization data are generated targeting an LFE2M-50E-6F900C using Lattice Diamond 1.1 and Synplify Pro D- 2009.12L-1 software. Performance might vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2M family.

### Ordering Part Number

The Ordering Part Number (OPN) for the PCI Express x4 RC-Lite IP core targeting LatticeECP2M devices is PCI-ERC1-PM-U1.

*Table A-6. Parameter Settings for Config Demo*

| | Config 1 |
|---|---|
| PCI express Link | x1 |
| Maximum payload size supported | 128 Bytes |
| Include ECRC support | No |
| Other parameters | Default |

## LatticeECP3 Utilization (x1 RC-Lite)

Table A-3 shows the resource utilization for the PCI Express x1 RC-Lite IP core implemented in a LatticeECP3 FPGA. Table A-4 lists the parameter settings for the IP core configuration shown in Table A-3.

*Table A-7. Resource Utilization[1]*

| IPexpress Configuration[1] | Slices | LUTs | Registers | sysMEM EBRs |
|---|---|---|---|---|
| Config 1 | 3059 | 4560 | 3048 | 3 |

1. Performance and utilization data are generated targeting an LFE3-95E-7FN1156CES using Lattice Diamond 1.1 and Synplify Pro D- 2009.12L-1 software. Performance might vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family.

## Ordering Part Number

The Ordering Part Number (OPN) for the PCI Express x1 RC-Lite IP core targeting LatticeECP3 devices is PCI-ERC1-E3-U1.

***Table A-8. Parameter Settings for Config Demo***

| | Config 1 |
|---|---|
| PCI express Link | x1 |
| Maximum payload size supported | 128 Bytes |
| Include ECRC support | No |
| Other parameters | Default |

Общество с ограниченной ответственностью «МосЧип» ИНН 7719860671 / КПП 771901001
Адрес: 105318, г.Москва, ул.Щербаковская д.3, офис 1107

**Данный компонент на территории Российской Федерации**

**Вы можете приобрести в компании MosChip.**

Для оперативного оформления запроса Вам необходимо перейти по данной ссылке:

http://moschip.ru/get-element

Вы можете разместить у нас заказ для любого Вашего проекта, будь то серийное производство или разработка единичного прибора.

В нашем ассортименте представлены ведущие мировые производители активных и пассивных электронных компонентов.

Нашей специализацией является поставка электронной компонентной базы двойного назначения, продукции таких производителей как XILINX, Intel (ex.ALTERA), Vicor, Microchip, Texas Instruments, Analog Devices, Mini-Circuits, Amphenol, Glenair.

Сотрудничество с глобальными дистрибьюторами электронных компонентов, предоставляет возможность заказывать и получать с международных складов практически любой перечень компонентов в оптимальные для Вас сроки.

На всех этапах разработки и производства наши партнеры могут получить квалифицированную поддержку опытных инженеров.

Система менеджмента качества компании отвечает требованиям в соответствии с ГОСТ Р ИСО 9001, ГОСТ РВ 0015-002 и ЭС РД 009

**Офис по работе с юридическими лицами:**

105318, г.Москва, ул.Щербаковская д.3, офис 1107, 1118, ДЦ «Щербаковский»

Телефон: +7 495 668-12-70 (многоканальный)

Факс: +7 495 668-12-70 (доб.304)

E-mail: info@moschip.ru

Skype отдела продаж:
moschip.ru
moschip.ru_4
moschip.ru_6
moschip.ru_9