LatticeECP2M™ PCS PIPE IP Core

User's Guide

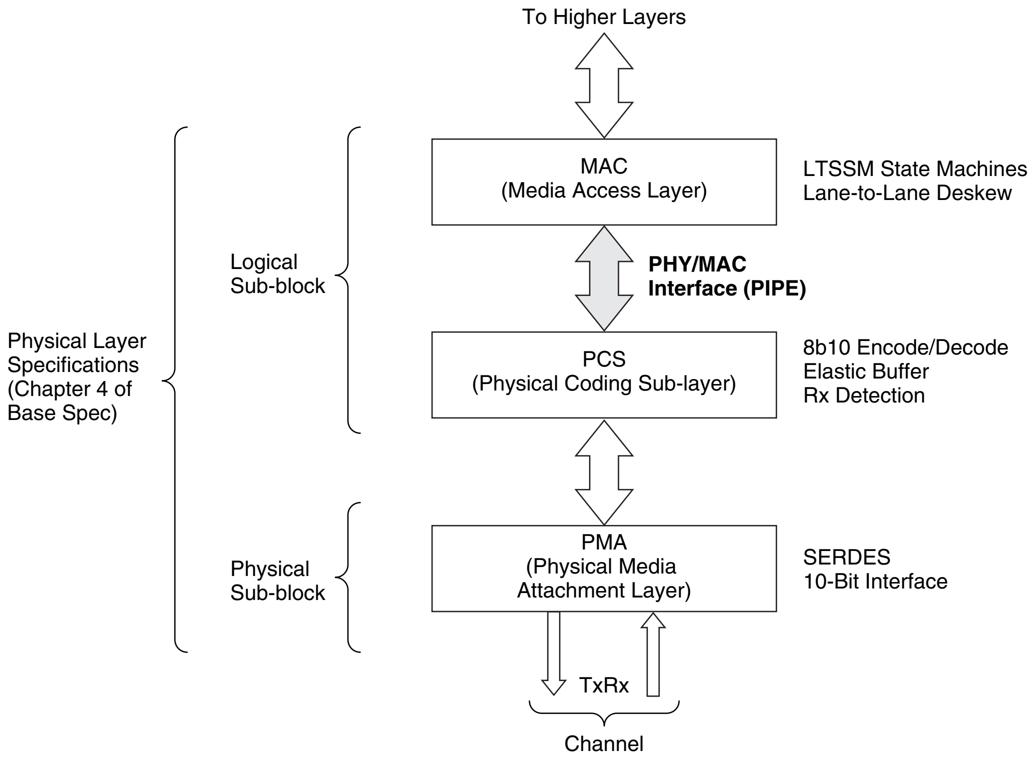
March 2009
ipug77_01.0

Introduction

Intel defined the PHY Interface for PCI Express (PIPE) as a standard interface between a PHY device and the Media Access (MAC) layer for PCI Express (PCIe) applications. The PIPE interface allows the PCI Express PHY device and the MAC layer to be implemented in discrete form (using an off-the-shelf PHY device) or in integrated form. The partitioning of the PCI Express Physical Layer shown Figure 1 illustrates this flexibility.

The Lattice PCS PIPE IP core offers PCI Express PHY device functionality, compliant to the Intel PIPE Architecture Draft Version 1.00 (PIPE Ver_1.00), to any endpoint solutions. The PCS PIPE IP core utilizes the SERDES/PCS integrated in LatticeECP2M™ FPGAs. The Lattice PCS PIPE IP core can be configured to support a link with one or four lanes.

Figure 1. PHY Layer Partitioning



General Features

The LatticeECP2M PCS PIPE IP core supports the following features.

PIPE Section

- Fully compliant to PIPE Ver_1.00
- Standard PCI Express PHY interface allows for multiple IP sources
- Selectable 8-bit or 16-bit interface to transmit and receive PCI Express data
- Holding registers/FIFOs for staging transmit and receive data

SERDES/PCS Section

- Selectable SERDES Quad location for LatticeECP2M50 and larger devices
- Selectable x1or x4 PCI Express implementations
- Selectable SERDES Channel for PCI Express x1 mode

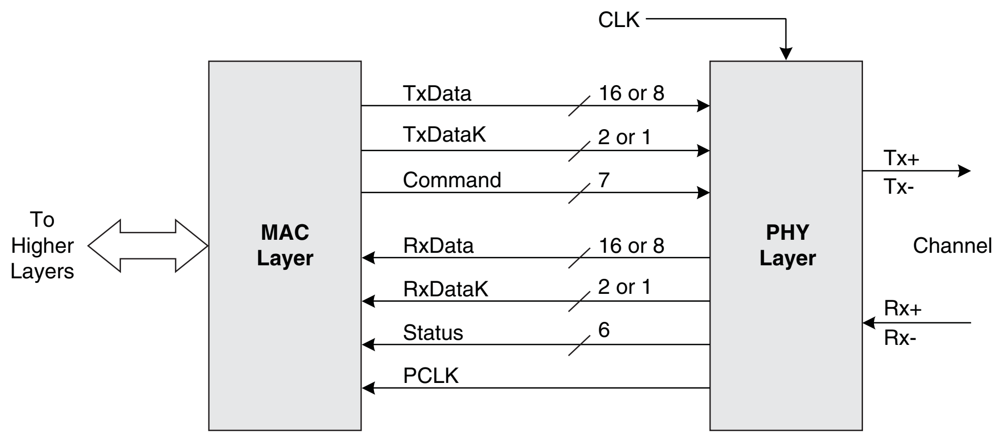
- Clock/data recovery from the serial stream
- Direct disparity control for use in transmitting compliance pattern
- 8b10b encoder/decoder and error indication
- Receiver detection
- 2.5GT/s full-duplex rate per channel

IP Core Features

Description

Figure 2 shows the signals between the PHY and the MAC layers as defined by the PIPE Ver.100 document. Table 1 provides a listing and description of the signals.

Figure 2. PHY/ MAC PIPE Interface

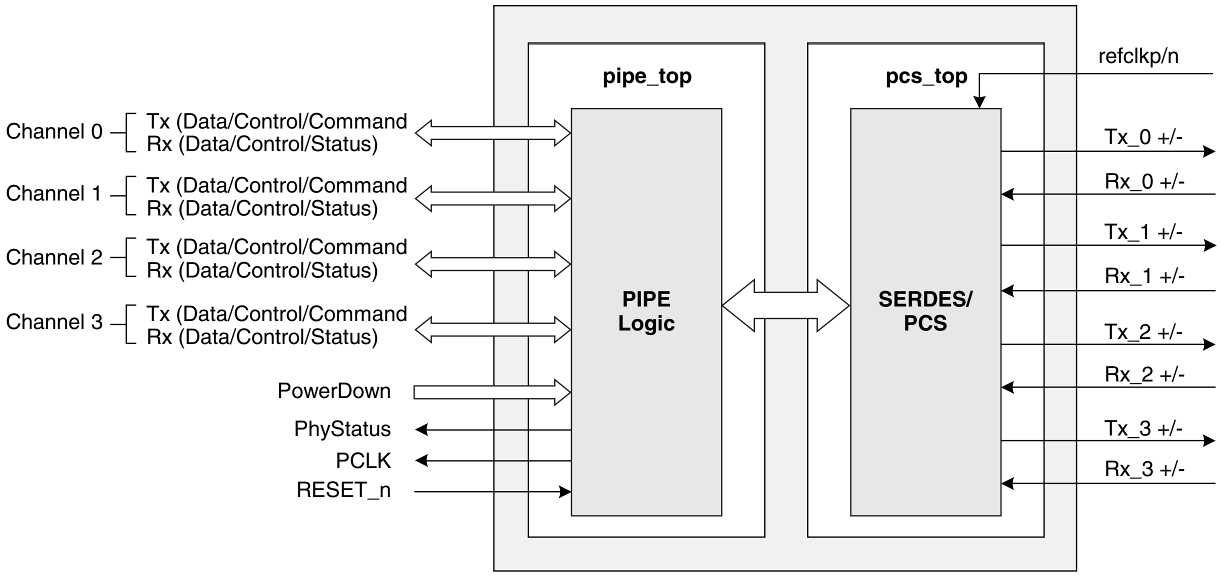


Block Diagram

The PCS PIPE IP core, as an implementation of a PIPE-compliant PHY device, consists of two main functional blocks: the pcs_top module, containing the SERDES/PCS quad, and the pipe_top module. The pipe_top module contains the logic required to transmit parallel data from the MAC, transfer received data to the MAC layer and other functions conforming to the PIPE specifications.

Figure 3 shows a high-level block diagram that illustrates the two main functional blocks.

Figure 3. PCS PIPE IP Block Diagram - 4 Channels



Signal Descriptions

Table 1 lists the ports and their descriptions for the PCS PIPE IP core. Note that the signal direction “In/Out” is referenced from the PHY device. Thus, an “Out” signal is driven by the PHY and an “In” signal is input to the PHY. For more information on the operational timing involving these signals, please refer to the PIPE Ver_1.00 specification.

Table 1. PCS PIPE IP Port List

Signal	Direction	Description
PIPE Interface		
Clock		
PCLK	Out	For an 8- bit data width, the PCLK frequency is 250 MHz. For a 16-bit data width, the PCLK frequency is 125 MHz.
Command Interface Signals		
RESET_n	In	Active low reset input. This input should be driven directly form the PCI Express fundamental reset signal PERST#.
TxDetectRx/ Loopback	In	Used to tell the PHY to begin a receiver detection operation or to begin loopback. 1 = Request transmitter to do far-end receiver detection 0 = Normal data operation
TxElecIdle	In	Forces Tx Output to electrical idle when asserted in all power states. 1 = Force SERDES transmitter to output Electrical Idle 0 = Normal operation When de-asserted while in P0 (operational) (as indicated by power down signals), valid data is present on the TxData[..] and TxDataK [..] ports and that the data should be transmitted.
TxCompliance	In	Sets the running disparity to negative. Used when transmitting the compliance pattern.
RxPolarity	In	Tells the PHY to invert the polarity of the received data. 1 = PHY inverts the polarity 0 = PHY does not invert the polarity

Table 1. PCS PIPE IP Port List (Continued)

Signal	Direction	Description
PowerDown[1:0]	In	Power Management States: [1] [0] Description 0 0 P0, Normal operation 0 1 P0s, Power Saving, low recovery time latency. Currently unsupported. 1 0 P1, Lower power, longer recovery time latency (64 us max) 1 1 P2, Lowest power. Currently unsupported. The PCS PIPE IP supports P1 and P0 states. *Note: The LatticeECP2M SERDES/PCS block is never powered down since the relative power saving in relation to the rest of the FPGA is minimal. The MAC layer must not generate the unsupported P0s and P2 states.*
Status Interface		
RxValid[3:0]	Out	Indicates symbol lock and valid data on RxData and RxDataK. 1 = Lane is synchronized to COM symbol (hex BC) 0 = Lane has either lost symbol lock or still in the process locking to COM symbol
RxElecIdle[3:0]	Out	Each bit, when logic 1, indicates the corresponding lane's receiver is electrically idle. RxElecIdle0 refers to lane 0, etc.
PhyStatus	Out	PhyStatus is asserted to indicate successful receiver detection during the P1 state and also after transitions on supported power management states.
RxStatus[2:0]	Out	Indicates receiver status during detection and error codes associated with the received data stream. Receiver Detection Phase: [2] [1] [0] Description 0 0 0 No Receiver detected 0 1 1 Receiver detected, Gen 1 rate Receiving Data Phase: [2] [1] [0] Description 0 0 0 Received data OK 0 0 1 1 SKP added 0 1 0 1 SKP removed 0 1 1 Reserved 1 0 0 Currently unsupported 1 0 1 Currently unsupported 1 1 0 Currently unsupported 1 1 1 Currently unsupported
Transmit Data Interface		
TxData[15:0]/ TxData[7:0]	In	Transmit Data. For the 16-bit interface,. bits [7:0] corresponds to the first symbol to be transmitted, and bits [15:8] the second symbol. For 8-bit data width only TxData[7:0] bits are applicable.
TxDataK[1:0]/ TxDataK	In	Data/Control indicator for corresponding TxData bytes. For 16-bit interfaces, bit 0 corresponds to the lower-byte of TxData, bit 1 to the upper-byte. A value of zero for either bit 0 or bit 1 location indicates TxData is a data byte; a value of 1 indicates TxData is a control byte.
Receive Data Interface		
RxData[15:0]/ RxData[7:0]	Out	Received Data. For the 16-bit interface, bits [7:0] corresponds to the first symbol received, and bits [15:8] the second symbol. For 8-bit data width only RxData[7:0] bits are applicable.
RxDataK[1:0]/ RxDataK	Out	Data/Control indicator associated with RxData. For 16-bit interfaces, bit 0 corresponds to the lower byte of RxData, bit 1 to the upper byte. A value of zero for either the bit 0 or bit 1 location indicates RxData is a data byte; a value of 1 indicates RxData is a control byte.
SERDES/PCS		
Clock/Data/Status		
refclkp, refclkn	In	Differential, 100 MHz reference clock. These inputs must be driven directly from the PCI Express REFCLK +/- source.
hdout[p/n][0,1,2,3]	Out	SERDES serial differential outputs.

Table 1. PCS PIPE IP Port List (Continued)

Signal	Direction	Description
hdin[p/n],[0,1,2,2]	In	SERDES serial differential inputs.
ffs_rlol_ch[0,1,2,3]	Out	SERDES outputs indicating the status of the Clock/Data Recovery (CDR) process for each channel. A value of 1 indicates the CDR has either lost lock or is still in the process of locking to serial input data.
SERDES Client Interface (SCI)		
sciwritedata[7:0]	In	Write data for the SERDES configuration registers.
sciaddress[5:0]	In	Address of SERDES configuration register to be written to or read from.
sciselaux	In	Select SERDES Auxiliary Channel for write/read operations
scienaux	In	Enable SERDES Auxiliary Channel for write/read operations.
scird	In	Set to logic 1 for read operation.
sciwstn	In	Set to Logic 0 for write operation.
scireaddata[7:0]	Out	Read data output.
Support Signals		
ffc_quad_rst	In	This input is directly connected to the SERDES/PCS quad reset signal. The entire quad is reset when this input is driven to logic 1. PCI Express signal PERST# is inverted and used to drive this input.
ffs_plol	Out	This status output, when logic 1, indicates the SERDES transmit PLL has either lost lock or has not acquired lock to the differential 100MHz clock applied at the refclkp/refclkn inputs.
PCLK_by_2	Out	125 MHz output clock in 8-bit mode. This clock is phase-aligned with PCLK and is intended for MACs with 16-bit internal data paths.
phy_l0	In	This active-high signal is asserted for one PCLK cycle by the MAC to indicate that LTSSM state has successfully reached the L0 state. This input signal is active for one PCLK cycle. This input signal and phy_cfgln[3:0] can be used to keep in reset unconfigured lanes. Please see Figure 4.
phy_cfgln[3:0]	In	Each bit is driven to logic 1 by the MAC to indicate the corresponding lane is successfully configured. These inputs must be tied to logic 0 when not in use.
ctc_disable	In	This input is used for debugging only and must be set to logic 0 for normal operation.
flip_lanes	In	Used for mapping SERDES channels to PIPE lanes. When connected to logic 1, PCS PIPE channels 0, 1, 2 and 3 corresponds to SERDES channels 0, 1, 2 and 3. When connected to logic 0, PCS PIPE channels 3, 2, 1 and 0 are mapped to SERDES channels 0, 1, 2 and 3, respectively.

Interface Description

PIPE Interface

The operational timing characteristics of the PIPE interface signals comply with the PIPE Ver_1.00 specification. Please refer to this specification for complete information.

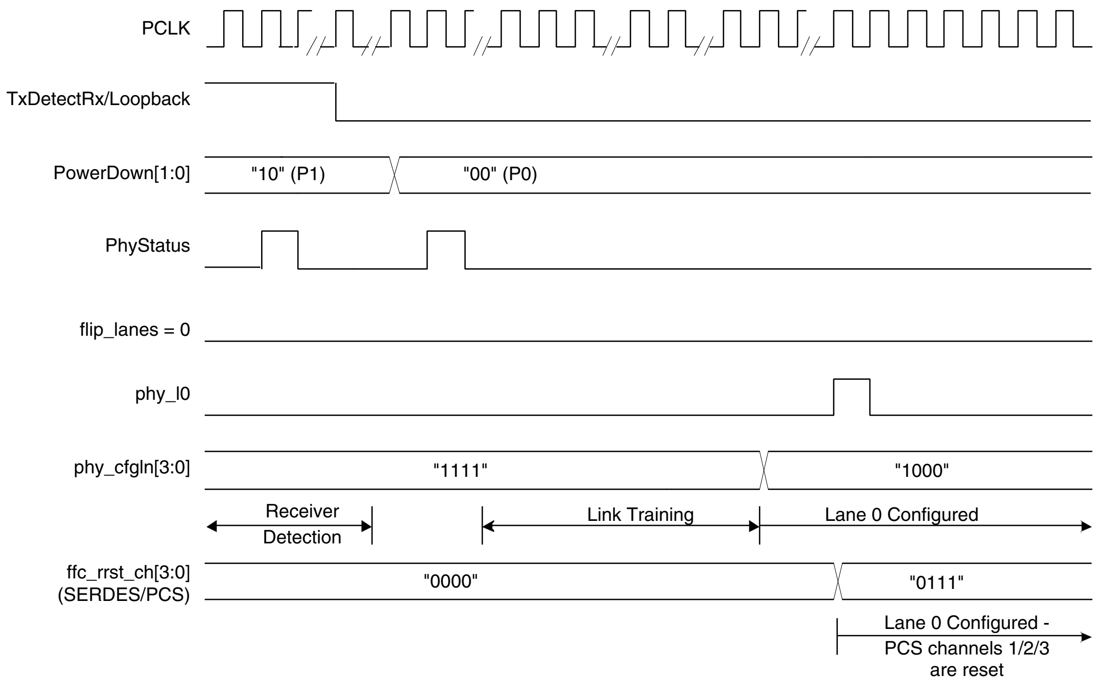
Using the phy_l0 and phy_cfgln[3:0] Inputs

During link training, it is possible that links that are supposed to operate with four lanes will only have one or two lanes configured. In this case, the link operates in downgrade mode. The PCS PIPE IP core includes input signals to support MACs operating in downgrade mode from either x4 to x2 or x4 to x1. The objective is to hold the PCS channels corresponding to the unconfigured lanes in reset state. With the PCS channel in reset state, the output signals will be inactive. However, the corresponding SERDES section will still be powered up. The SERDES/PCS channel of the unconfigured lanes will be in electrical idle state.

Figure 4 shows the operational timing diagram to accomplish this objective.

TxDetectRx/Loopback is shown asserted and PHY is in P1 state, indicating receiver detection is in progress. PhyStatus is asserted by the PCS PIPE to indicate successful receiver detection. The MAC de-asserts TxDetectRx/Loopback and changes to P0 state. Link training can only occur during P0 state, as shown. When LTSSM state reaches L0, the MAC can assert phy_l0 and provide, on phy_cfgln[3:0], the status indicating the lanes that have been successfully configured. The values on phy_cfgln[3:0] (representing the un-configured lanes) are used to set corresponding ffc_rrst_ch[3:0] input to logic 1 and hold in reset the affected PCS channels.

Figure 4. Downgrade to x1 - Holding Unused PCS Channels in Reset



SERDES/PCS-IP Interface – Channel Usage and Byte Order

Table 2 shows the mapping of the SERDES Quad/Channels to the corresponding byte-order at the PCS/IP Interface for three different IP core configurations. The normal byte order for the Lattice Native x4 IP core corresponds to the setting flip_lanes=0. For other IP vendors, the normal byte order may correspond to the setting flip_lanes=1.

In order to support both Native x4 and x4-Downgrade-to-x1, the design must use the correct normal byte ordering as processed by the IP core. If this is not done correctly, the device may function correctly in Native x4 mode but not in Downgrade-to-x1 mode.

If PCI Express lanes are required to be reversed due to board layout restrictions, the connection of flip_lanes can be set accordingly to take into account the byte-ordering requirements of the IP core.

Table 2. SERDES/PCS-IP Interface – Channel Usage and Byte Order

PCI Express Lane/ SERDES Channel	Native x4		x4-Downgrade-to-x1 (phy_cfgln[3:0]=1000b)		Native x1 (Suggested)	
	flip_lanes= 0	flip_lanes= 1	flip_lanes= 0	flip_lanes= 1	flip_lanes= 0	flip_lanes= 1
0	3 (MSB)	0	Active	Active	Active	Active
1	2	1	Held in reset	Held in reset	Disabled	Disabled
2	1	2	Held in reset	Held in reset	Disabled	Disabled
3	0	3	Held in reset	Held in reset	Disabled	Disabled

IP Core Implementation

LatticeECP2M devices feature up to 16 channels of embedded SERDES with associated Physical Coding Sublayer (PCS) logic. Each channel supports full-duplex serial data transfers at rates up to 3.125 Gbps. The PCS logic is configurable and supports numerous industry-standard high-speed data transfer protocols, including the PCI Express protocol.

See the LatticeECP2/M Family Data Sheet (Architecture section) for information on the available SERDES/PCS quads in each device.

The PCS PIPE IP core is intended to be integrated with a PCI Express MAC IP core, that has a PIPE Ver_1.00 compliant interface, targeting the LatticeECP2M device family.

Additional information on this core is also available in the LatticeECP2M PCS PIPE Readme document. The Readme file is available once the IP core is installed in ispLEVER® and the core configuration is generated via IPexpress™.

Getting Started with the Design Flow

This document provides information on how to use the Lattice PCS PIPE IP core but does not include all of the information found in the PIPE Ver_1.00 specification. It is strongly encouraged for users to consult the PIPE Ver_1.00 specification to understand the framework in which this core is utilized.

The Lattice PCS PIPE IP core is supported in ispLEVER as part of IPexpress. For more information on the IPexpress design flow please see the IPexpress Quick Start Guide. This guide provides a high-level overview of the IPexpress application, installation of new cores, and core customization.

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

Using the PCS PIPE IP Core

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

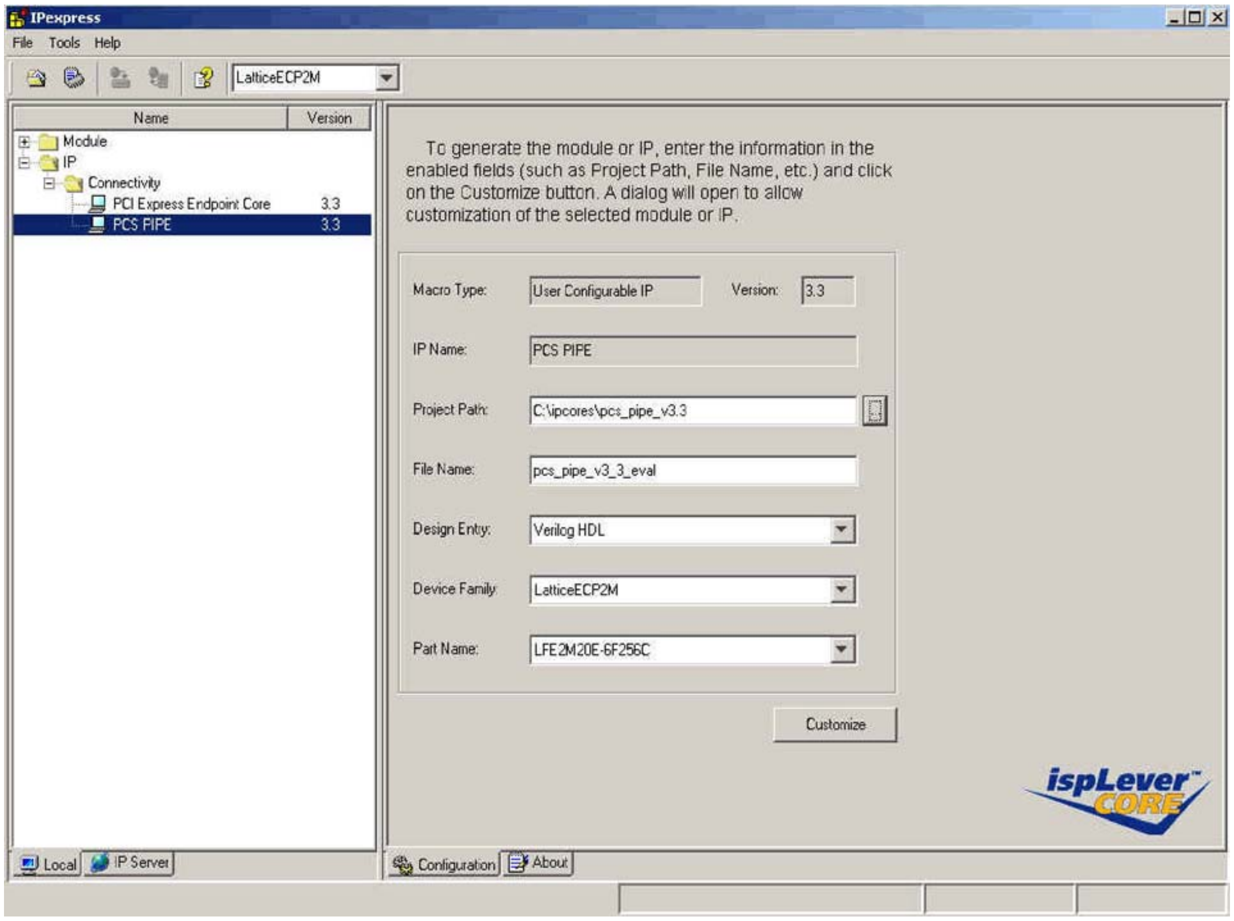
You can use the IPexpress software tool to help generate new configurations of this IP core. Details regarding the usage of IPexpress can be found in the IPexpress and ispLEVER help systems. For more information on the ispLEVER design tools, visit the Lattice website.

Generating the IP Core

The IPexpress tool is used to generate all Lattice IP cores. This tool generates all the files necessary to instantiate the IP core and synthesize a design with the core as a black box. The generated files also allow the user to functionally simulate and implement the generated IP core into the target device. Synthesis and PAR constraints for the IP core are also provided as starting points to help in the implementation and timing closure of the design.

Figure 5 shows the IPexpress tool GUI with the PCS PIPE IP core installed and an example for each required entry. Information entered in the File Name box will be the module name, also <username>. To specify a custom core, click the **Customize** button.

Figure 5. IPexpress GUI with Installed PCS PIPE IP Core



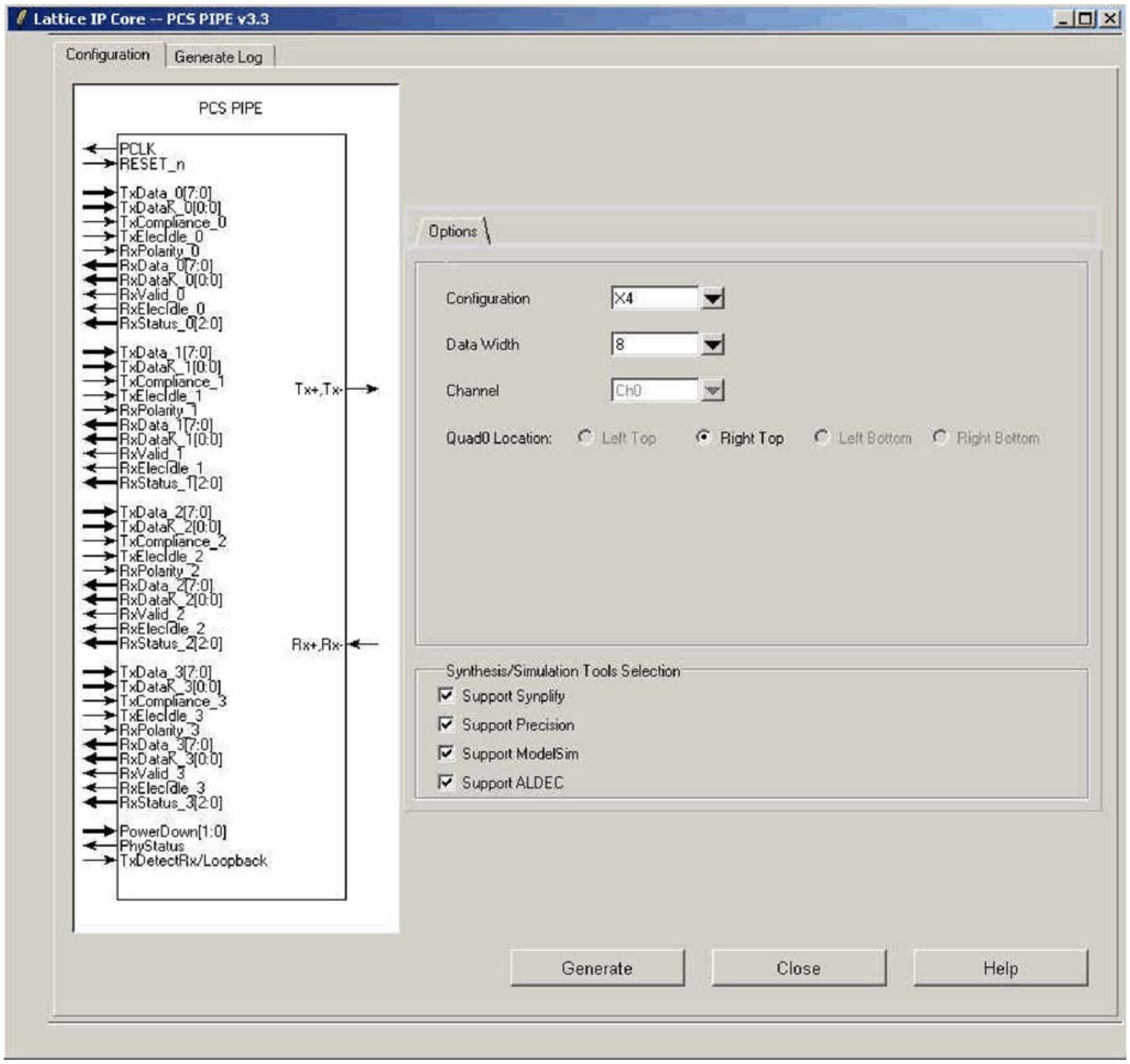
To generate the module or IP, enter the information in the enabled fields (such as Project Path, File Name, etc.) and click on the Customize button. A dialog will open to allow customization of the selected module or IP.

Macro Type: User Configurable IP Version: 3.3
IP Name: PCS PIPE
Project Path: C:\ipcores\pcs_pipe_v3.3
File Name: pcs_pipe_v3_3_eval
Design Entry: Verilog HDL
Device Family: LatticeECP2M
Part Name: LFE2M20E-6F256C

Figure 6 shows the GUI for customizing the IP core. Since the target device entered in Figure 5 is the LatticeECP2M20, the Quad0 location is automatically selected as “Right Top”. TN1124, LatticeECP2M SERDES/PCS Usage Guide, provides detailed information.

After selecting the parameters, click the **Generate** button to generate the customized IP core and associated files. All the generated files will be deposited under the Project Path directory entered in the IPexpress GUI shown in Figure 5.

Figure 6. PCS PIPE x4/8-Bit Configuration - Single SERDES Quad



Configuration: X4
Data Width: 8
Channel: Ch0
Quad0 Location: Left Top [✓ Right Top] Left Bottom Right Bottom

Synthesis/Simulation Tools Selection
- ✓ Support Synplify
- ✓ Support Precision
- ✓ Support ModelSim
- ✓ Support ALDEC

Configuration
Specifies the number of PCI Express lanes the PCS PIPE IP core is required to support. Configuration is for either x1 or x4 mode.

Datawidth
Specifies the datawidth of each lane at the PIPE interface. Supports either 8-bit or 16-bit widths, as defined in the PIPE specification.

Channel
In x1 mode, this option allows the user to select any one of the four SERDES channels in the chosen quad. All four SERDES channels in the chosen quad are automatically selected in x4 mode.

Quad0 Location

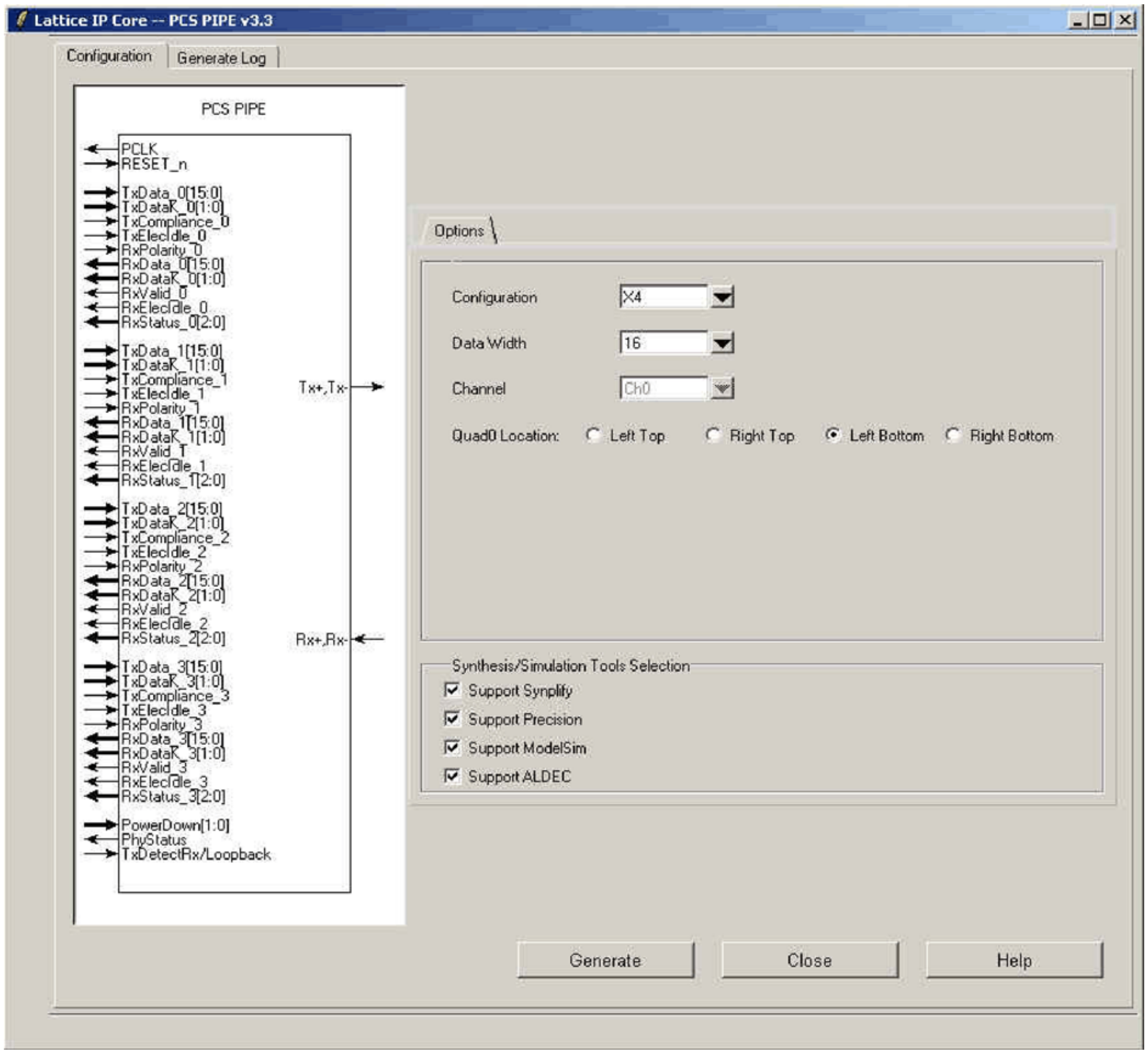
Specifies the physical location of the SERDES/PCS block for devices with more than one SERDES/PCS quad. In general, depending on the selected device and the number of SERDES/PCS quads and location, one or more constraints with the following format will be included in the .lpf file generated by IPexpress.

```
LOCATE COMP "pcs_instance_urpcs" SITE "URPCS" ; # Right Top
LOCATE COMP "pcs_instance_ulpcs" SITE "ULPCS" ; # Left Top
LOCATE COMP "pcs_instance_lrpcs" SITE "LRPCS" ; # Right Bottom
LOCATE COMP "pcs_instance_llpcs" SITE "LLPCS" ; # Left Bottom
```

LatticeECP2M70E and Larger Devices

Figure 7 shows the IPexpress GUI when targeting the LatticeECP2M70 and larger devices that have four SERDES/PCS quads available.

Figure 7. Multiple SERDES/PCS Quads for Larger Devices



Configuration: X4

Data Width: 16

Channel: Ch0

Quad0 Location: Left Top Right Top [✓ Left Bottom] Right Bottom

Synthesis/Simulation Tools Selection

- [✓] Support Synplify
- [✓] Support Precision
- [✓] Support ModelSim
- [✓] Support ALDEC

Generated Files

IPexpress generates several files to help implement a design. Most of the filenames generated are derived from the module name specified in IPexpress.

Table 3 provides a list of files created by IPexpress and describes their use.

Table 3. File List

File/Directory	Simulation	Synthesis /ispLEVER	Description
<user name>_inst.v			This file is for use as the instance template of the IP core.
<user name>.v	Yes		This file is the "wrapper" of the IP core and used for simulation.
pcs_pipe_params.v	Yes		This file provides the configuration options of the IP for the simulation model.
<user name>_bb.v		Yes	This file is the "black-box" model of the core for user's synthesis.
<user name>.ngo		Yes	This file is the synthesized IP core converted to .ngo format for used by ispLEVER back-end tools
pcs_pipe_<config[1]>.txt	Yes	Yes	This file contains settings to configure the SERDES/PCS for PCI Express operating environment.SERDES. This file is copied in to the simulation directory as well as the ispLEVER project directory.
<user name>.lpf		Yes	This file contains preferences for the PAR tool. These preferences should be included in the user's preference file.
<user name>.lpc			This file contains the IPexpress options used to re-generate or modify the core by importing in IPexpress.
pcs_pipe_eval			This directory contains a sample design. The design can be simulated and implemented through ispLEVER.

1. config = 8b_X1 for the 8-bit, 1-channel configuration
 = 8b_X4 for the 8-bit, 4-channel configuration.

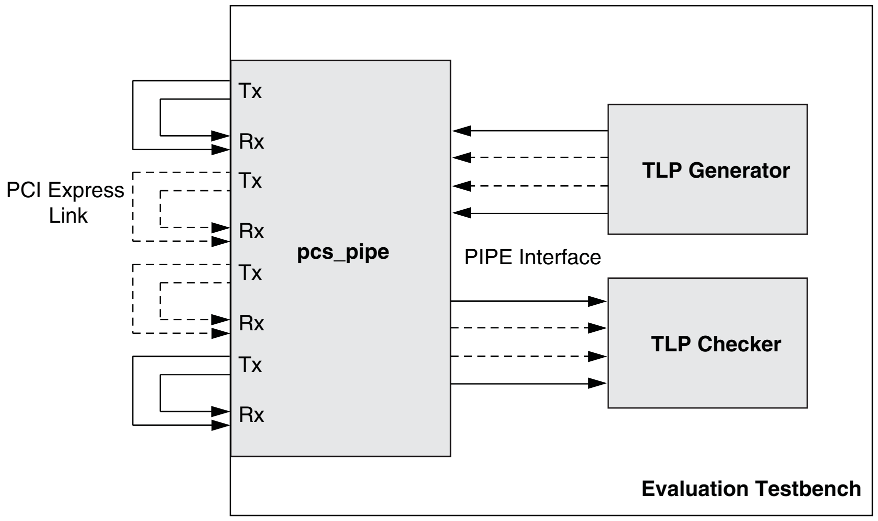
 = 16b_X1 for the 16-bit, 1-channel configuration.
 = 16b_X4 for the 16-bit, 4-channel configuration.

Simulation

Included with the IPexpress generated core is an evaluation testbench located under the <user name>\testbench directory. The testbench shows the behavior of the IP core in functional simulation as well as post-PAR timing simulation. Most of the IP cores for communications applications can be used in a loop-back fashion where serial transmit data is looped-back as the received data. This mechanism is used during IP core functional verification as well as in actual hardware validation.The same widely-adapted loop-back method is also used in the testbench.

In a real application, however, the PCS PIPE IP core will be an integrated part of a PCI Express endpoint (upstream port) as the PHY device and is connected to the PHY device of a downstream port. In the evaluation testbench, a few commands are used to force the PCS PIPE to advance the LTSSM to L0 state without the benefit of connecting to a downstream port. Once a link is established, Transaction Layer Packets (TLPs) are sent through the link to show the transmit and receive interface. The received TLPs are compared with the transmitted TLPs at the PIPE interface. Figure 8 illustrates the loop-back method used in the evaluation testbench.

Figure 8. PCS PIPE Evaluation Testbench - Loop-back Method



The Readme file generated with the IP core provides information on running the simulation of the evaluation package. Simulation support for the evaluation package is provided for Aldec® Active-HDL® and Mentor Graphics® ModelSim® simulators. A sample ModelSim simulation script file, eval_beh_rtl.do is available under the directory <Project_Path>\pcs_pipe_eval\<user name>\sim\modelsim\script. ModelSim is strictly for Verilog-only simulation with the evaluation testbench.

Active-HDL is also available for mixed-Verilog/VHDL simulation of the evaluation package. In this case, the IP core top-level is in VHDL while the modules instantiated have corresponding Verilog representations.

Implementation Details

Clocking Strategy

A system board with PCI Express slots provides differential 100 MHz reference clock for each slot. The SERDES/PCS block outputs both 125 MHz (ff_tx_h_clk) and 250 MHz (ff_tx_f_clk) clocks using the PCI Express differential 100 MHz as the reference clock. These two clocks are used in the PCS PIPE IP and also made available the MAC layer device and user-application modules. Within the PCS PIPE IP, received data at the SERDES output is written into a FIFO, on a per-channel basis, using the channel's "recovered clock" (ff_rx_fclk) running at 250 MHz as the "write clock". Data is read out of the channel's FIFO with ff_tx_f_clk as the read clock.

Figure 9 shows how the clocks are used in the PCS PIPE with an 8-bit data width. PCLK_by_2 runs at 125 MHz and is driven directly from the SERDES ff_tx_h_clk output. PCLK runs at 250 MHz and is driven directly by the SERDES ff_tx_f_clk output. MAC devices with a 16-bit internal data path and 8-bit PIPE interface can use both PCLK_by_2 and PCLK for byte-unpacking required in the transmit path and byte-packing with the received data.

Figure 9. PCS PIPE IP Core Clocking for 8-bit Data Width

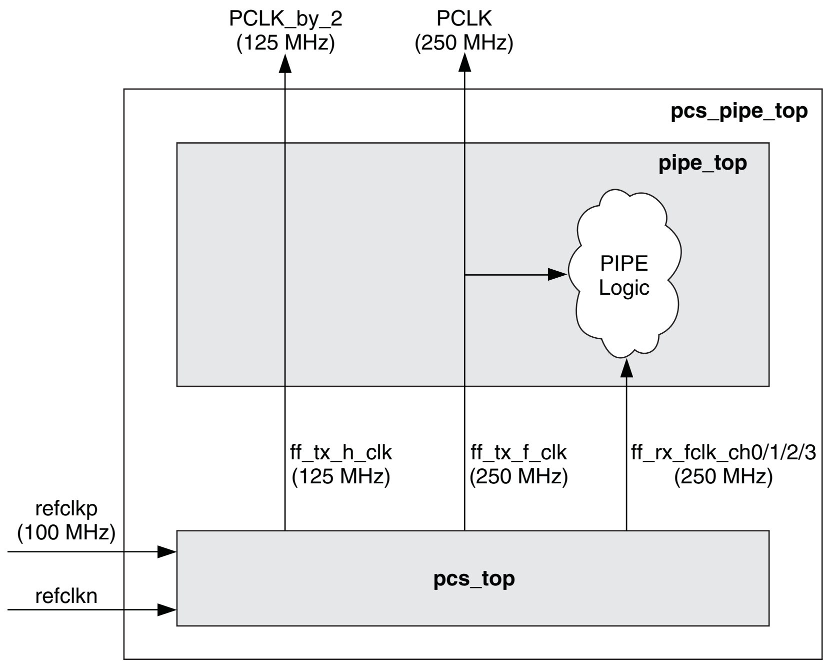
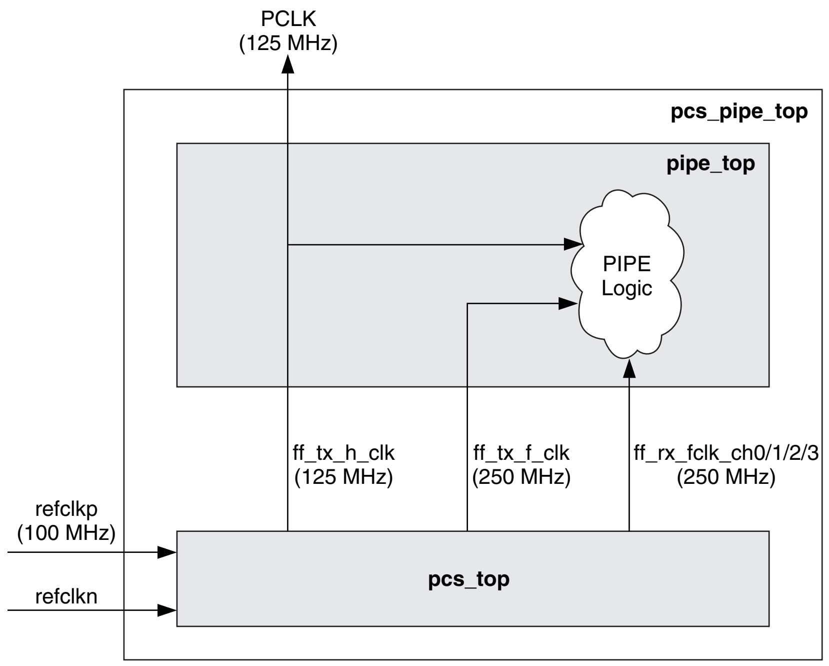


Figure 10 shows how the clocks are used in the PCS PIPE with a 16-bit data width. In this case, PCLK runs at 125 MHz and is driven directly from the SERDES ff_tx_h_clk output. Byte-packing/unpacking is done within the PIPE logic.

Figure 10. PCS PIPE IP Core Clocking for 16-bit Data Width



Setting Design Timing Constraints

Timing constraints required for the PCS PIPE IP core are included in the <username>.lpf, one of the files generated by IPexpress. The timing constraints for the different IP core configurations are as follows.

x1, 8-bit data width - Channel 0

```
FREQUENCY NET “PCLK_c” 250 MHz HOLD_MARGIN 0.2 nS ;
```

x1, 16-bit data width - Channel 0

```
FREQUENCY NET “PCLK_c” 125 MHz HOLD_MARGIN 0.2 nS ;
```

x4, 16-bit data width

```
FREQUENCY NET “PCLK_c” 250 MHz HOLD_MARGIN 0.2 nS ;
FREQUENCY NET “PCLK_by_2_c” 125 MHz HOLD_MARGIN 0.2 nS ;
FREQUENCY NET “u1_dut/u1_core/ff_rx_fclk_0” 250 MHz HOLD_MARGIN 0.2 nS ;
FREQUENCY NET “u1_dut/u1_core/ff_rx_fclk_1” 250 MHz HOLD_MARGIN 0.2 nS ;
FREQUENCY NET “u1_dut/u1_core/ff_rx_fclk_2” 250 MHz HOLD_MARGIN 0.2 nS ;
FREQUENCY NET “u1_dut/u1_core/ff_rx_fclk_3” 250 MHz HOLD_MARGIN 0.2 nS ;
```

The 250 MHz and 125 MHz clock nets must be routed using the target device’s primary clock routing resources to meet frequency and timing requirements. The constraints are also included in the .lpf file, as shown in the example below.

```
USE PRIMARY NET “PCLK_c” ;
```

When integrating the PCS PIPE IP core with a MAC layer device and user-application modules, the constraints for the applicable IP configuration can be used and added to the constraints for the overall design. Minor editing might be required to account for the difference in the levels of hierarchy when integrating into the overall design.

Errors and Warnings

Warning messages will be generated by ispLEVER when implementing a design with the PCS PIPE IP core in the x4 configuration. This is because the per-channel recovered clocks drive external PIPE wrapper logic. For the x1, 8-bit or 16-bit configuration, no additional external logic is driven by the recovered clock.

Place & Route Design - x4 Configuration Only

The following warnings will be present in the automake.log file after completion of PAR.

```
WARNING - par: The driver of primary clock net u1_dut/u1_core/ff_rx_fclk_0
          is not placed on one of the PIO sites which are dedicated for
          primary clocks.  This primary clock will be routed to a H-spine
          through general routing resource or be routed as secondary clock
          and may suffer from excessive delay or skew.

WARNING - par: The driver of primary clock net u1_dut/u1_core/ff_rx_fclk_1
          is not placed on one of the PIO sites which are dedicated for
          primary clocks.  This primary clock will be routed to a H-spine
          through general routing resource or be routed as secondary clock
          and may suffer from excessive delay or skew.

WARNING - par: The driver of primary clock net u1_dut/u1_core/ff_rx_fclk_2
          is not placed on one of the PIO sites which are dedicated for
          primary clocks.  This primary clock will be routed to a H-spine
          through general routing resource or be routed as secondary clock
          and may suffer from excessive delay or skew.
```

```
WARNING - par: The driver of primary clock net u1_dut/u1_core/ff_rx_fclk_3
          is not placed on one of the PIO sites which are dedicated for
          primary clocks.  This primary clock will be routed to a H-spine
          through general routing resource or be routed as secondary clock
          and may suffer from excessive delay or skew.
```

These warnings indicate that the “recovered clocks” from the SERDES/PCS module are using general routing to connect to a primary clock entry point. This is expected for the LatticeECP2M architecture.

Integrating the PCS PIPE

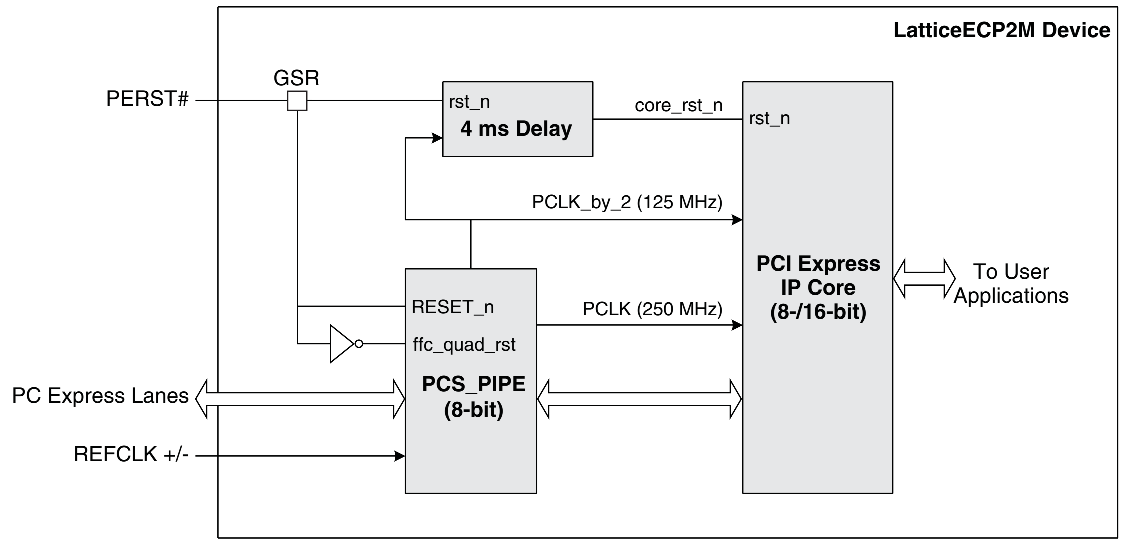
8-bit PIPE Interface

Integrating the PCS PIPE with a similar PIPE-compliant MAC layer device is fairly straight-forward. Figure 11 shows the block diagram of the PCS PIPE integrated with the Lattice PCI Express IP core targeting a LatticeECP2M device. PERST#, REFCLK+/- and PCI Express lane signals represent connections to a system board’s PCI Express slot.

The PCI Express fundamental reset signal, PERST#, resets the SERDES/PCS via the ffc_quad_rst input and the PIPE logic section via the RESET_n input. The Lattice PCI Express IP core requires an 8-bit PIPE interface while the internal data path is 16 bits. PERST# is delayed in the 4 msec delay block and the output, core_rst_n, is applied to the reset input of the PCI Express IP core. The 4 msec delay effectively extends the LTSSM Detect.Active state to 16 msec from the nominal 12 msec. This additional 4 msec delay is well within the PCI Express Base Specification which states that specified LTSSM “timeout values are minus 0 seconds and plus 50% unless explicitly stated otherwise”.

The block diagram shown in Figure 11 is used to implement the Lattice PCI Express x4 Endpoint Reference Design. The addition of the 4 msec delay greatly enhanced the reliability and robustness of the endpoint’s ability to establish a link and interoperability with several different platforms.

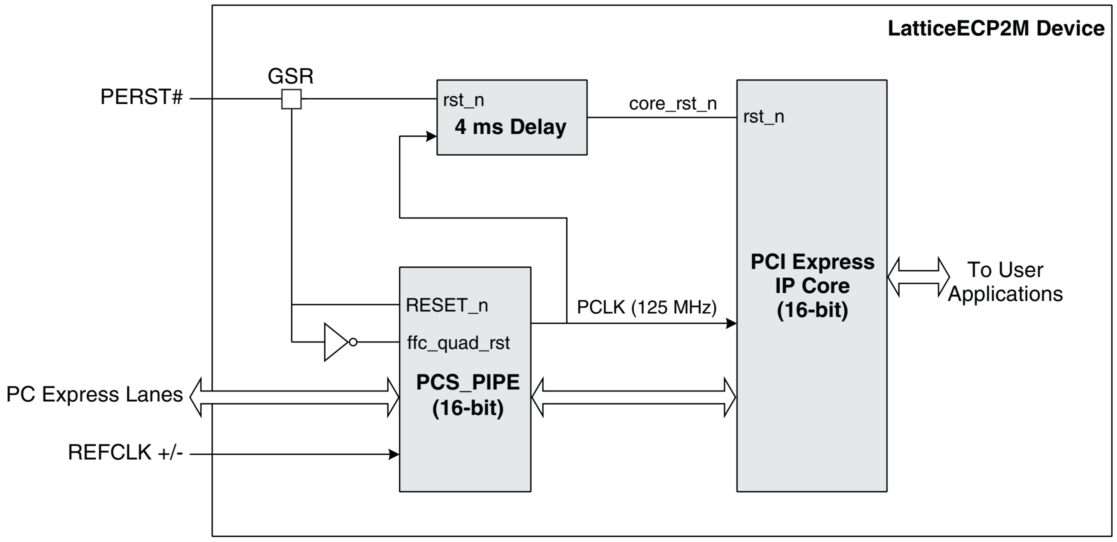
Figure 11. PCS PIPE Integrated with Lattice PCI Express IP Core



16-bit PIPE Interface

Figure 12 shows the block diagram of the PCS PIPE integrated with the PLDA® PCI Express XpressLite® IP core targeting a LatticeECP2M device. PERST#, REFCLK+/- and PCI Express lane signals represent connections to a system board’s PCI Express slot. The implementation is very similar to the 8-bit case.

Figure 12. 16-Bit PCS PIPE Integrated with the PLDA PCI Express XpressLite IP Core



References

- PHY Interface for the PCI Express Architecture Draft Version 1.86, Intel Corporation
- PCI Express Base Specification, Rev 1.1, PCI-SIG
- LatticeECP2/M Family Data Sheet
- TN1114 - Electrical Recommendations for Lattice SERDES
- TN1124 - LatticeECP2M SERDES/PCS Usage Guide

Technical Support Assistance

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

Revision History

Date	Version	Change Summary
March 2009	01.0	Initial release.