LatticeECP2/M SERDES/PCS Usage Guide

July 2021 Technical Note FPGA-TN-02254

Introduction to PCS

The LatticeECP2M™ FPGA family combines a high-performance FPGA fabric, high-performance I/Os and large embedded RAM in a single industry-leading architecture. All LatticeECP2M devices also feature up to 16 channels of embedded SERDES with associated Physical Coding Sublayer (PCS) logic. The PCS logic can be configured to support numerous industry-standard, high-speed data transfer protocols.

Each channel of PCS logic contains dedicated transmit and receive SERDES for high-speed full-duplex serial data transfers at data rates up to 3.125 Gbps. The PCS logic in each channel can be configured to support an array of popular data protocols including Ethernet (1GbE and SGMII), PCI Express, CPRI, and OBSAI. In addition, the protocol-based logic can be fully or partially bypassed in a number of configurations to allow users flexibility in designing their own high-speed data interface.

The PCS also provides bypass modes that allow for a direct 8-bit or 10-bit interface from the SERDES to the FPGA logic. Each SERDES pin can be independently DC-coupled and can allow for both high-speed and low-speed operation on the same SERDES pin for such applications as Serial Digital Video.

Features

- Up to 16 Channels of High-Speed SERDES
  - 250 Mbps to 3.125 Gbps per channel
  - 3.125Gbps operation with low 100mW power per channel
  - Receive equalization and transmit pre-emphasis for small form factor backplane operation
  - Supports PCI Express, Ethernet (1GbE and SGMII) plus multiple other standards
  - Supports user specified generic 8b10b mode
  - Beacon support for PCI Express
  - Out-of-band signal interface for low speed inputs (video application)
- Multiple Clock Rate Support
  - Separate reference clocks for each PCS quad allow easy handling of multiple protocol rates on a single device
- Full Function Embedded Physical Coding Sub-layer (PCS) Logic Supporting Industry Standard Protocols
  - Up to 16 channels of full-duplex data supported per device
  - Multiple protocol support on one chip
  - Supports popular 8b10b based packet protocols
  - SERDES Only mode allows direct 8- or 10-bit interface to FPGA logic
- Gigabit Ethernet Support
  - IEEE 1000BASE-X compliant
  - 8b10b encoding/decoding
  - Insertion of /I2/ symbols into the receive data stream for auto-negotiation support
  - Comma character word alignment
  - Clock Tolerance Compensation circuit
- PCI Express Support
  - x1 to x4 support in one PCS quad
  - Integrated Word aligner
  - 8b10b encoding/decoding
  - Clock Tolerance Compensation circuit
  - Electrical Idle and Receiver Detection support
  - Support for Beacon Transmission and Beacon Detection

- Multiple Protocol Compliant Clock Tolerance Compensation (CTC) Logic
  - Compensates for frequency differential between reference clock and received data rate
  - Allows user defined Skip pattern of 1, 2, or 4 bytes in length
- Integrated Loopback Modes for System Debugging
  - Three loopback modes are provided for system debugging

Supported Standards

The supported standards are listed in Table 18-1.

Table 18-1. Supported SERDES Standards (Fully Supported)

Standard	Rates (Mbps)	REFCLK (MHz)	FPGA CLK (MHz)	Encoding	Signal Type
PCI Express	2500	100	250	8b10b	CML
GbE/SGMII	1250	125	125	8b10b	CML
Generic 8b10b	250 to 3125	25 to 312.5	25 to 312.5	8b10b/None	CML
10-bit SERDES Only[1]	250 to 3125	25 to 312.5	25 to 312.5	None	CML
8-bit SERDES Only[1]	250 to 3125	25 to 312.5	25 to 312.5	None	CML
SD-SDI[2]	143, 177, 270, 360	14.3, 17.7, 27, 36	143, 177, 135, 180	SMPTE scrambled	CML
HD-SDI	1483.5, 1485	148.35, 148.5	148.35, 148.5	SMPTE scrambled	CML
CPRI	614.4 1228.8	61.44 122.88	61.44 122.88	8b10b	CML

1. 8-bit SERDES Only Mode and 10-bit SERDES Only Mode bypass the Link Align/Comma Align, 8b10b encoder/decoder and the CTC. It does not bypass the CDR.
2. Serial Digital Interface (SDI) for Standard Definition (SD): 143Mbps, 177Mbps bypasses the SERDES/PCS Block. The clock and data come in to the FPGA through the RX pins but get into the FPGA core via the BSCAN path. CDR is done in the FPGA core. In the transmit direction, decimation is used for these “low” bit rates. To do the CDR in the FPGA, reference clock of 14.3MHz and 17.7MHz are required respectively. 270Mbps is the most common frequency. This will go through the 10-bit data path.

It is possible to support XAUI, SRIO, OBSAI, CPRI, 1XFC, 2XFC, PICMG 3.1, PICMG 3.4, PICMG 3.5 and 3G-SDI standards with the SERDES modes specified above. Contact Lattice Semiconductor Technical Support Group for additional information.

Architecture Overview

The PCS logic is arranged in quads containing logic for four independent full-duplex data channels. Table 18-2 shows the availability of SERDES/PCS quads for each device in the LatticeECP2M family.

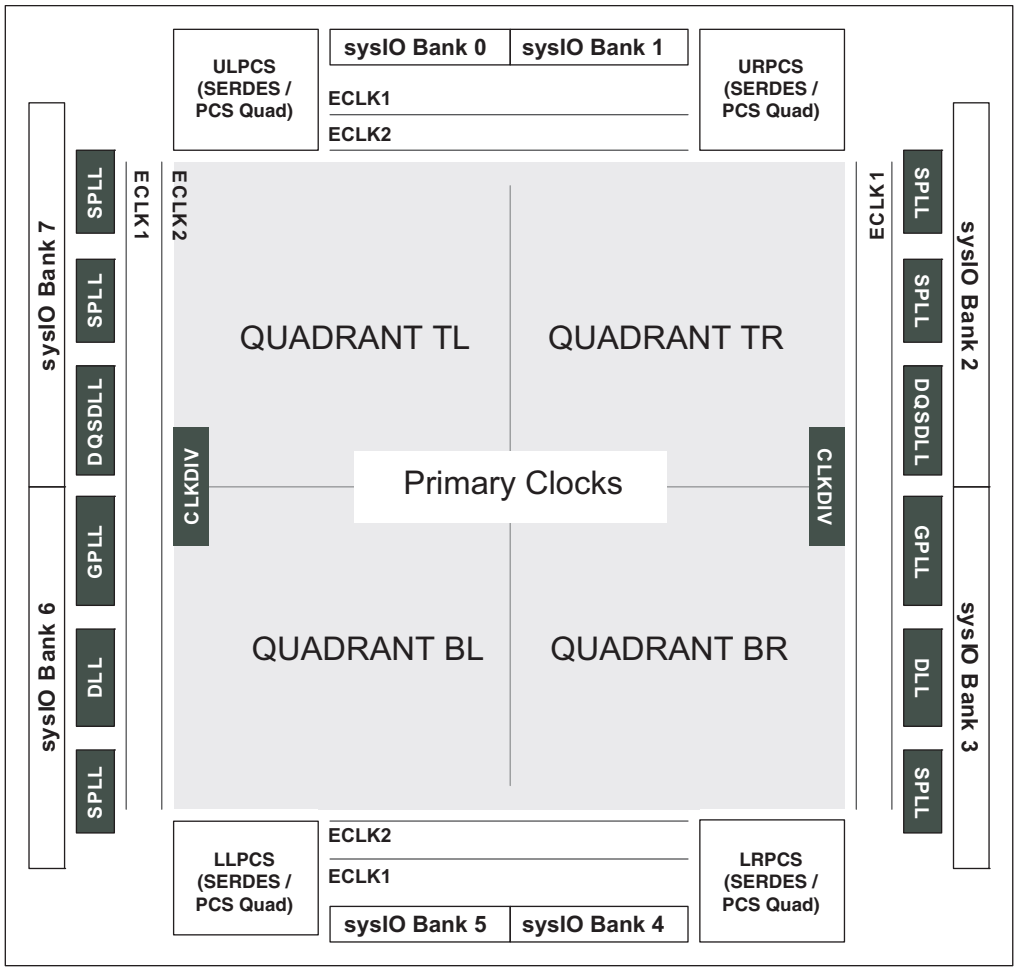
Table 18-2. SERDES/PCS Quads per LatticeECP2M Device

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

PCS Quad

Figure 18-1 is a layout of a LatticeECP2M device showing the arrangement of PCS quads on the device (the largest array containing 4 quads is shown. Other devices have fewer quads).

Figure 18-1. LatticeECP2M70/100 Block Diagram



Every quad can be programmed into one of several protocol based modes. Each quad requires its own reference clock which can be sourced externally from package pins or internally from the FPGA logic.

Since each quad has its own reference clock, different quads can support different standards on the same chip. This feature makes the LatticeECP2M family of devices ideal for bridging between different standards.

PCS quads are not dedicated solely to industry standard protocols. Each quad (and each channel within a quad) can be programmed for many user defined data manipulation modes. For example, modes governing user-defined word alignment, and clock tolerance compensation can be programmed for non-protocol operation.

PCS Quad and Channels

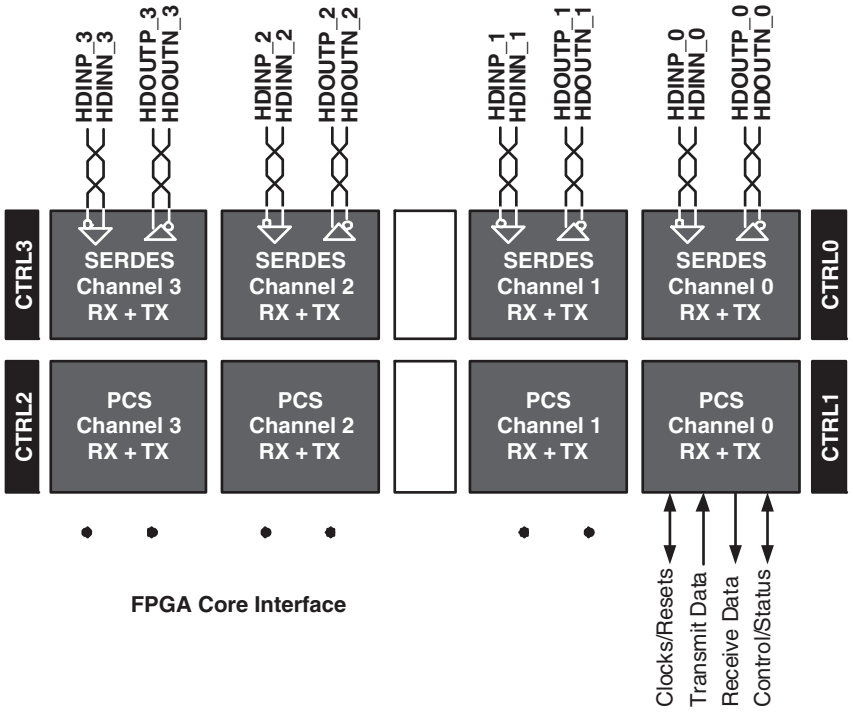
Each quad on a device supports up to four channels of full-duplex data. The user can utilize anywhere from one to four channels in a quad depending on the application. Many options can be set by the user for each channel independently within a given quad.

Figure 18-1 also shows an example of a device with four PCS quads which contain a total of 16 PCS channels. Quads are named according to the location of the respective quad on the LatticeECP2M array: URPCS (Upper Right PCS), ULPCS (Upper Left PCS), LRPCS (Lower Right PCS), LLPCS (Lower Left PCS).

Per Channel PCS/FPGA Interface Ports

All PCS quads regardless of chosen mode have the same external high speed serial interface at the package pins. However, every PCS mode has its own unique list of input/output ports from/to the FPGA logic appropriate to the protocol chosen for the quad. A detailed description of the quad input/output signals for each mode is provided in this document. A simplified diagram showing the channels within a single quad is shown in Figure 18-2.

Figure 18-2. PCS Quad Block Diagram



Locating a PCS Quad

LatticeECP2M-50 and larger devices include two to four PCS Quads.

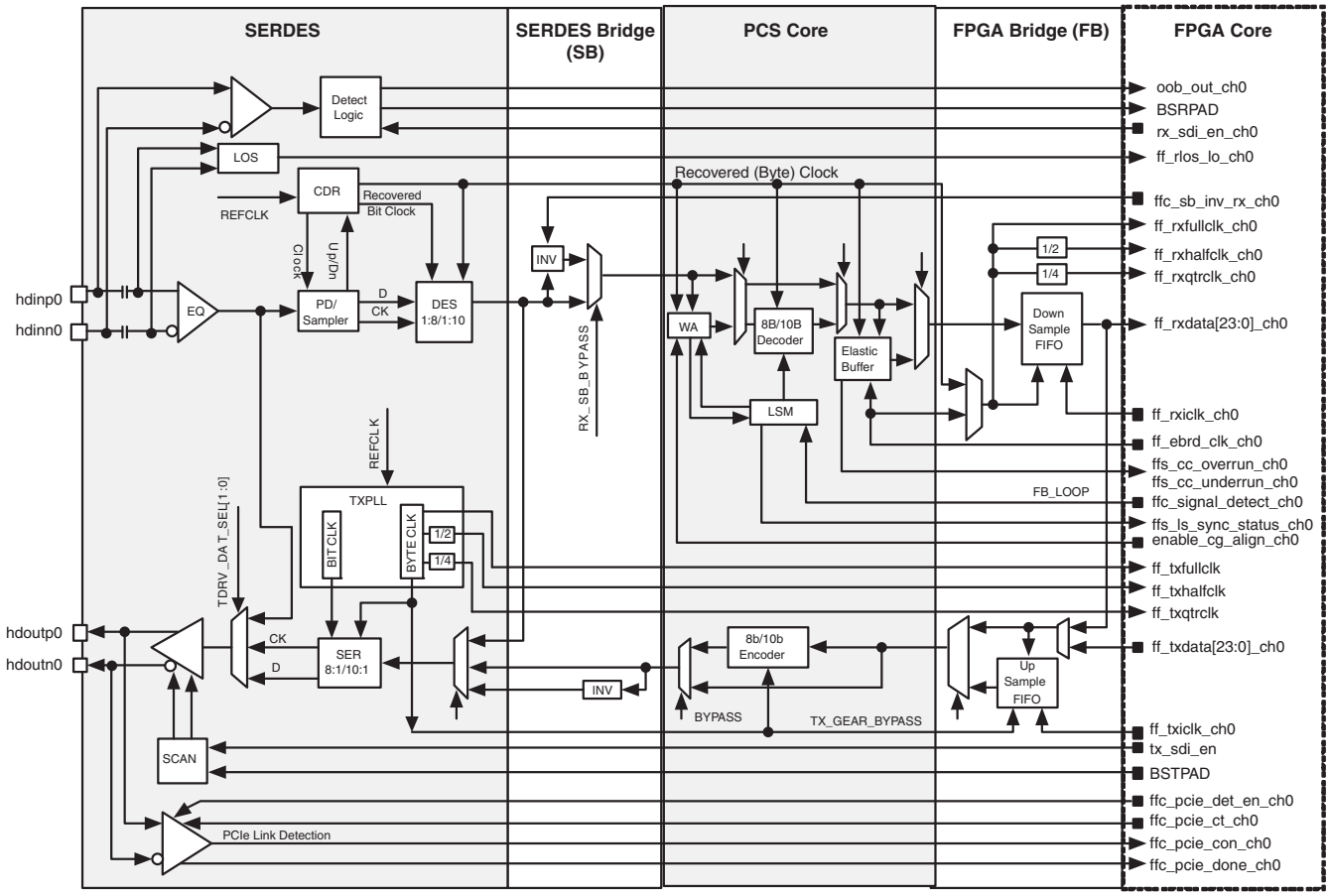
Users can locate each PCS Quad in a desired location using the LOCATE preference in the preference file. Below is an example of the preference, LOCATE.

```
LOCATE COMP "PCS_instantiation_1" SITE URPCS ;
```

Detailed Channel Block Diagram

Figure 18-3 is a detailed block diagram representation of the major functionality in a single channel of the LatticeECP2M SERDES/PCS. This diagram shows all the major blocks and the majority of the control and status signals that are visible to the user logic in the FPGA. This diagram also shows the major sub-blocks in the channel-SERDES, SERDES Bridge, PCS Core and the FPGA Bridge.

Figure 18-3. LatticeECP2M SERDES/PCS Detailed Channel Block Diagram



A general description of the FPGA interface signals follows.

Clocks and Resets

A PCS quad supplies per channel locked reference clocks and per channel recovered receive clocks to the FPGA logic interface. Each PCS quad provides these clocks on both primary and secondary FPGA clock routing. The PCS/FPGA interface also has ports for the transmit and receive clocks supplied from the FPGA fabric for all four channels in each quad.

Each quad has reset inputs to force reset of both the SERDES and PCS logic in a quad or just the SERDES. In addition, separate resets dedicated for the PCS logic are provided for each channel for both the transmit and receive directions.

Transmit Data Bus

The signals for the transmit data path are from the FPGA to the FPGA Bridge in the PCS Block. For high-speed standards, the datapath can be geared 2:1 to the internal PCS data path, which is 8 bits wide (+ control/status signals). The highest speed of the interface for PCI Express x1 is 250MHz in non-geared mode. With gearing (i.e. a 16-bit wide data path), the maximum speed is 156.25MHz (for XAUI 4x channel mode). The SERDES and PCS will support data rates up to 3.125Gbps data that correspond to an interface speed of 156.25MHz (with 2:1 gearing).

Receive Data Bus

The signals for the receive path are from the FPGA Bridge in the PCS Block to the FPGA. The data path may be geared 2:1 to the internal PCS data path which is 8 bits wide. It is possible to disable the gearing via a software register bit, in which case, the bus widths are halved. When the data is geared, the lower bits (ff_rx_d[9:0]) are the octet that has been received first and the higher bits (ff_rx_d[19:10]) are the octet that has been received second. If the data is not geared, the lower bits ((ff_rx_d[9:0]) are the active bits and the higher bits should not be used.

Table 18-3. Data Bus Usage by Mode

Data Bus PCS Cell Name[5]	G8B10B	CPRI	PCI Express	GIGE	8BSER	10BSER	SDI
FF_TX_D_0_0	ff_txdata_ch0[0]						
FF_TX_D_0_1	ff_txdata_ch0[1]						
FF_TX_D_0_2	ff_txdata_ch0[2]						
FF_TX_D_0_3	ff_txdata_ch0[3]						
FF_TX_D_0_4	ff_txdata_ch0[4]						
FF_TX_D_0_5	ff_txdata_ch0[5]						
FF_TX_D_0_6	ff_txdata_ch0[6]						
FF_TX_D_0_7	ff_txdata_ch0[7]						
FF_TX_D_0_8	ff_tx_k_cntrl_ch0[0]				GND	ff_txdata_ch0[8]	
FF_TX_D_0_9	ff_force_disp_ch0[0]][1]			GND		ff_txdata_ch0[9]	
FF_TX_D_0_10	ff_disp_sel_ch0[0]][1]			ff_xmit_ch0[0]][2]	GND		
FF_TX_D_0_11	GND		ff_pci_ei_en_ch0[0]	ff_correct_disp_ch0[0]	GND		
FF_TX_D_0_12	ff_txdata_ch0[8]					ff_txdata_ch0[10]	
FF_TX_D_0_13	ff_txdata_ch0[9]					ff_txdata_ch0[11]	
FF_TX_D_0_14	ff_txdata_ch0[10]					ff_txdata_ch0[12]	
FF_TX_D_0_15	ff_txdata_ch0[11]					ff_txdata_ch0[13]	
FF_TX_D_0_16	ff_txdata_ch0[12]					ff_txdata_ch0[14]	
FF_TX_D_0_17	ff_txdata_ch0[13]					ff_txdata_ch0[15]	
FF_TX_D_0_18	ff_txdata_ch0[14]					ff_txdata_ch0[16]	
FF_TX_D_0_19	ff_txdata_ch0[15]					ff_txdata_ch0[17]	
FF_TX_D_0_20	ff_tx_k_cntrl_ch0[1]				GND	ff_txdata_ch0[18]	
FF_TX_D_0_21	ff_force_disp_ch0[1]][1]			GND		ff_txdata_ch0[19]	
FF_TX_D_0_22	ff_disp_sel_ch0[1]][1]			ff_xmit_ch0[1]][2]	GND		
FF_TX_D_0_23	GND		ff_pci_ei_en_ch0[1]	ff_correct_disp_ch0[1]	GND		
FF_RX_D_0_0	ff_rxdata_ch0[0]						
FF_RX_D_0_1	ff_rxdata_ch0[1]						
FF_RX_D_0_2	ff_rxdata_ch0[2]						
FF_RX_D_0_3	ff_rxdata_ch0[3]						
FF_RX_D_0_4	ff_rxdata_ch0[4]						
FF_RX_D_0_5	ff_rxdata_ch0[5]						
FF_RX_D_0_6	ff_rxdata_ch0[6]						
FF_RX_D_0_7	ff_rxdata_ch0[7]						
FF_RX_D_0_8	ff_rx_k_cntrl_ch0[0]				NC	ff_rxdata_ch0[8]	
FF_RX_D_0_9	ff_disp_err_ch0[0]		ff_rxstatus0_ch0[0]	ff_disp_err_ch0[0]	NC	ff_rxdata_ch0[9]	
FF_RX_D_0_10	ff_cv_ch0[0]][3]		ff_rxstatus1_ch0[0]	ff_cv_ch0[0]][3]	NC		
FF_RX_D_0_11	NC		ff_rxstatus2_ch0[0]	ff_rx_even_ch0[0]][4]	NC		
FF_RX_D_0_12	ff_rxdata_ch0[8]					ff_rxdata_ch0[10]	
FF_RX_D_0_13	ff_rxdata_ch0[9]					ff_rxdata_ch0[11]	
FF_RX_D_0_14	ff_rxdata_ch0[10]					ff_rxdata_ch0[12]	
FF_RX_D_0_15	ff_rxdata_ch0[11]					ff_rxdata_ch0[13]	
FF_RX_D_0_16	ff_rxdata_ch0[12]					ff_rxdata_ch0[14]	
FF_RX_D_0_17	ff_rxdata_ch0[13]					ff_rxdata_ch0[15]	

Table 18-3. Data Bus Usage by Mode (Continued)

Data Bus PCS Cell Name[5]	G8B10B	CPRI	PCI Express	GIGE	8BSER	10BSER	SDI
FF_RX_D_0_18	ff_rxdata_ch0[14]					ff_rxdata_ch0[16]	
FF_RX_D_0_19	ff_rxdata_ch0[15]					ff_rxdata_ch0[17]	
FF_RX_D_0_20	ff_rx_k_cntrl_ch0[1]				NC	ff_rxdata_ch0[18]	
FF_RX_D_0_21	ff_disp_err_ch0[1]		ff_rxstatus0_ch0[1]	ff_disp_err_ch0[1]	NC	ff_rxdata_ch0[19]	
FF_RX_D_0_22	ff_cv_ch0[1]][3]		ff_rxstatus1_ch0[1]	ff_cv_ch0[1]][3]	NC		
FF_RX_D_0_23	NC		ff_rxstatus2_ch0[1]	ff_rx_even_ch0[1]][4]	NC		

1. The force_disp signal will force the disparity for the associated data word on bits [7:0] to the column selected by the tx_disp_sel signal. If disp_sel is a one, the 10-bit code is taken from the 'current RD+' column (positive disparity). If the tx_disp_sel is a zero, the 10-bit code is taken from the 'current RD-' (negative disparity) column.
2. The auto-negotiation state machine generates the signal xmit. It is used to interact with the GIGE Idle State Machine in the hard logic.
3. When there is a code violation, the packet PCS 8b10b decoder will replace the output from the decoder with hex EE and K asserted (K=1 and d=EE is not part of the 8b10b coding space).
4. rx_even is a signal generated by the GIGE Link State Machine for the use of the GIGE Auto-negotiation and Receive State Machines (which is part of the IP core).
5. FF_TX_D_0_0: FPGA Fabric Transmit Data Bus Channel 0 Bit 0.

Control

Each mode has its own set of control signals which allows direct control of various PCS features from the FPGA logic. In general, each of these control inputs duplicate the effect of writing to a corresponding control register bit or bits. The ispLEVER® design tools give the user the option to bring these ports out to the FPGA interface.

Status

Each mode has its own set of status or alarm signals that can be monitored by the FPGA logic. In general, each of these status outputs correspond to a specific status register bit or bits. The ispLEVER design tools give the user the option to bring these port out to the PCS FPGA interface. Refer to the Mode Specific Control/Status Signals section for detailed information about control and status signals.

SCI (SERDES Client Interface) Bus

The SERDES Client Interface (SCI) is a soft IP that allow the SERDES/PCS Quad block to be controlled by registers as oppose to the configuration memory cells. It is a simple register configuration interface.

Using This Technical Note

The ispLEVER design tools from Lattice support all modes of the PCS. Most modes are dedicated to applications for a specific industry standard data protocol. Other modes are more general purpose modes which allow a user to define their own custom application settings. ispLEVER design tools allow the user to define the mode for each quad in their design. This document describes operation of the SERDES and PCS for all modes supported by ispLEVER. If you are using Lattice Diamond™ design software, see Appendix E.

This document provides a thorough description of the complete functionality of the embedded SERDES and associated PCS logic. Electrical and Timing Characteristics of the embedded SERDES are provided in the LatticeECP2/M Family Data Sheet. Operation of the PCS logic is provided in the PCS section. A table of all status and control registers associated with the SERDES and PCS logic which can be accessed via the SCI Bus is provided in the Memory Map section. Package pinout information is provided in the Architecture section of the LatticeECP2/M Family Data Sheet.

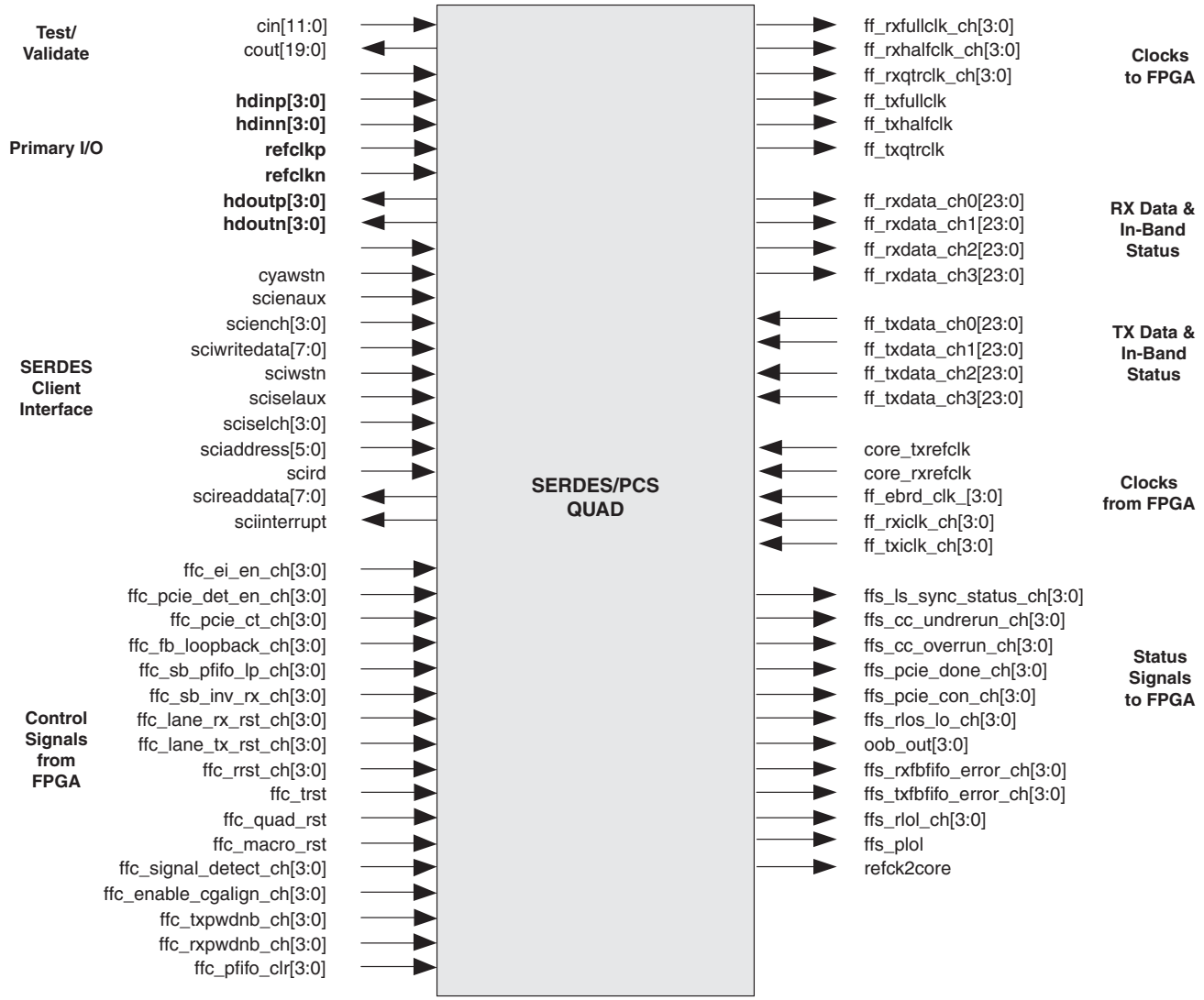
SERDES/PCS

The quad contains four channels with both Rx and Tx circuits, and an auxiliary channel that contains the Tx PLL. The reference clock to the Tx PLL can be provided either by the primary differential reference clock pins or by the FPGA core. The quad SERDES/PCS macro performs the serialization and de-serialization function for four lanes

of data. In addition, the PLL within the SERDES/PCS block provides the system clock for the FPGA logic. The quad also supports both full-data-rate and half-data-rate modes of operation on each Tx and Rx circuit independently.

The block level diagram is shown in Figure 18-4.

Figure 18-4. SERDES_PCS Block Signal Interface



I/O Definitions

Table 18-4 lists all default and optional input and outputs to/from a PCS quad. Users can choose optional ports for a PCS quad using the IPexpress™ GUI.

Table 18-4. SERDES_PCS I/O Descriptions

Signal Name	I/O	Type	Description	Default/ Optional
Primary I/O, SERDES Quad				
hdinp0	I	Channel	High-speed CML input, positive, channel 0	D
hdinn0	I	Channel	High-speed CML input, negative, channel 0	D
hdinp1	I	Channel	High-speed CML input, positive, channel 1	D
hdinn1	I	Channel	High-speed CML input, negative, channel 1	D
hdinp2	I	Channel	High-speed CML input, positive, channel 2	D
hdinn2	I	Channel	High-speed CML input, negative, channel 2	D
hdinp3	I	Channel	High-speed CML input, positive, channel 3	D
hdinn3	I	Channel	High-speed CML input, negative, channel 3	D
hdoutp0	O	Channel	High-speed CML output, positive, channel 0	D
hdoutn0	O	Channel	High-speed CML output, negative, channel 0	D
hdoutp1	O	Channel	High-speed CML output, positive, channel 1	D
hdoutn1	O	Channel	High-speed CML output, negative, channel 1	D
hdoutp2	O	Channel	High-speed CML output, positive, channel 2	D
hdoutn2	O	Channel	High-speed CML output, negative, channel 2	D
hdoutp3	O	Channel	High-speed CML output, positive, channel 3	D
hdoutn3	O	Channel	High-speed CML output, negative, channel 3	D
refclkp	I	Quad	Reference Clock input, positive, Dedicated CML input	D
refclkn	I	Quad	Reference Clock input, negative, Dedicated CML input	D
Receive / Transmit Data Bus (See Table for Detailed Data Bus Usage)				
ff_rxdata_ch0[23:0]	O	Channel	Data Signals for the channel 0 receive path	D
ff_rxdata_ch1[23:0]	O	Channel	Data Signals for the channel 1 receive path	D
ff_rxdata_ch2[23:0]	O	Channel	Data Signals for the channel 2 receive path	D
ff_rxdata_ch3[23:0]	O	Channel	Data Signals for the channel 3 receive path	D
ff_txdata_ch0[23:0]	I	Channel	Data Signals for the channel 0 transmit path	D
ff_txdata_ch1[23:0]	I	Channel	Data Signals for the channel 1 transmit path	D
ff_txdata_ch2[23:0]	I	Channel	Data Signals for the channel 2 transmit path	D
ff_txdata_ch3[23:0]	I	Channel	Data Signals for the channel 3 transmit path	D
Control Signals				
ffc_sb_inv_rx_ch[3:0]	I	Channel	Control the inversion of received data. 1 = Invert the data 0 = Do not invert the data	O
ffc_enable_cgalign_ch[3:0][4]	I	Channel	Control comma aligner. 1 = Enable comma aligner 0 = Lock comma aligner at current position.	O
ffc_signal_detect_ch[3:0][4]	I	Channel	Control Link State Machine 1 = Enable Link State Machine 0 = Disable Link State Machine	O
ffc_fb_loopback_ch[3:0]	I	Channel	FPGA Bridge Loopback. 1 = Enable loopback from Rx to Tx 0 = Normal data operation	O

Table 18-4. SERDES_PCS I/O Descriptions (Continued)

Signal Name	I/O	Type	Description	Default/ Optional
ffc_sb_pfifo_lp_ch[3:0]	I	Channel	SERDES Bridge Parallel Loopback 1 = Enable loopback from Rx to Tx, 0 = Normal data operation	O
ffc_pfifo_clr_ch[3:0]	I	Channel	SERDES Bridge Parallel Loopback FIFO Clear 1 = Reset Loopback FIFO 0 = Normal Loopback operation	D
rx_sdi_en	I	Channel	These signals are used in BSCAN mode only.	O
tx_sdi_en	I	Quad		
Reset Signals				
ffc_lane_rx_rst_ch[3:0]	I	Channel	Active high, asynchronous input. Resets individual Rx channel logic only in PCS.	D
ffc_lane_rx_tst_ch[3:0]	I	Channel	Active high, asynchronous input. Resets individual Tx channel logic only in PCS.	D
ffc_rrst_ch[3:0]	I	Channel	Active high. Resets selected digital logic in the SERDES Receive channel	D
ffc_trst	I	Quad	Active high, resets selected digital logic in all SERDES Transmit channels.	D
ffc_quad_rst	I	Quad	Active high, asynchronous input. Resets all SERDES channels including the auxiliary channel and PCS.	D
ffc_macro_rst	I	Quad	Active high, asynchronous input to SERDES quad. Resets all SERDES channels including the AUX channel but not PCS logic.	D
ffc_txpwdnb_ch[3:0]	I	Channel	Active low transmit channel power down. 0 = Transmit Channel Power Down.	D
ffc_rxpwdnb_ch[3:0]	I	Channel	Active low receive channel power down. 0 = Receive Channel Power Down.	D
Status Signals				
ffs_rlos_lo_ch[3:0]	O	Channel	Loss of signal detection for each channel. Register bits rlos_hset[2:0] are used to set the threshold. Low threshold is not user accessible. 1 = Loss of signal 0 = Signal detected	D
ffs_ls_sync_status_ch[3:0]	O	Channel	1 = Lane is synchronous to commas. 0 = Lane has not found comma.	D
ffs_cc_underrun_ch[3:0][6]	O	Channel	1 = Receive clock compensator FIFO underrun error, 0 = No FFIFO errors.	O
ffs_cc_overrun_ch[3:0][6]	O	Channel	1 = Receive clock compensator FIFO overrun error 0 = No FIFO errors.	O
ffs_rxfbfifo_error_ch[3:0]	O	Channel	1 = Receive FPGA bridge FIFO error 0 = No FIFO errors	D
ffs_txfbfifo_error_ch[3:0]	O	Channel	1 = Transmit FPGA bridge FIFO error 0 = No FIFO errors.	D
ffs_rlol_ch[3:0]	O	Channel	1 = Receive CDR loss of lock 0 = Lock maintained	D
ffs_plol	O	Quad	1 = Transmit PLL loss of lock 0 = Lock maintained	D
oob_out_ch[3:0][3]	O	Channel	Single ended outputs to video SERDES (in FPGA).	D
refck2core	O	Quad	Reference clock to FPGA core.	O

Table 18-4. SERDES_PCS I/O Descriptions (Continued)

Signal Name	I/O	Type	Description	Default/ Optional
Clock Signals to FPGA				
ff_rxfullclk_ch[3:0]	O	Channel	Receive channel recovered clock. In user mode, the source is always the channel's recovered clock. For standards such as GbE, 10 GbE that support clock compensation, the source is the respective transmit channel's system clock. For PCS bypass modes, it is also the Tx system clock, thus requiring raw mode to actually be done using either 8b10b mode with the 8b10b decoder disabled (10-bit or 20-bit data path).	D
ff_rxhalfclk_ch[3:0]	O	Channel	Receive channel recovered half clock. In 2:1 gearing mode, it is a divide-by-2 output.	D
ff_rxqtrclk_ch[3:0]	O	Channel	Receive channel recovered quarter clock. Available for further 2:1 gearing.	O
ff_txfullclk	O	Quad	Tx PLL full rate clock.	D
ff_txhalfclk	O	Quad	Tx PLL half clock.	D
ff_txqtrclk	O	Quad	Tx PLL quarter clock	O
Clock Signals from FPGA				
core_rxrefclk	I	Quad	Rx Reference clock from FPGA logic, for CDR PLL	D
core_txrefclk	I	Quad	Tx Reference clock from FPGA logic, for Tx SERDES PLL	D
ff_ebrd_clk_[3:0]	I	Channel	Receive channel clock input from FPGA for CTC FIFO (Elastic Buffer) read	D
ff_rxiclk_ch[3:0]	I	Channel	Receive channel clock input from FPGA. Used to clock the Rx FPGA Interface FIFO with a clock synchronous to the reference and/or receive reference clock.	D
ff_txiclk_ch[3:0]	I	Channel	Transmit channel clock input from FPGA.Per channel transmit clock inputs from FPGA. Used to clock the Tx FPGA. Interface FIFO with clock synchronous to the reference clock. Also used to clock the Rx FPGA Interface FIFO with a clock synchronous to the reference clock when CTC is used.	D
SERDES Client Interface (SCI)				
scienaux	I	R	1: sciwdata is written to the quad control registers 0: memory data is written to the quad control registers	O
scien_ch[3:0]	I	R	1: sciwdata is written to the channel control registers 0: memory data is written to the channel control registers	O
sciselaux	I	R	1: select quad registers	O
scisel_ch[3:0]	I	R	1: select channel registers	O
sciaddress[5:0]	I	R	Address bus input	O
scireaddata[7:0]	O	R	Read data output	O
sciwritedata[7:0]	I	R	Write data input	O
scird	I	R	1: Read data select 0: Read data not selected	O
sciwstn	I	R	Write strobe	O
sciinterrupt	O	R	Interrupt output	O
cyawstn[5]	I	R	1: Copy all memory cells to registers if sciwstn = 0 0: Default	O
SERDES Characterization / Test Bus				
cin[11:0]	I	R	Characterization test bus logic data input	D

Table 18-4. SERDES_PCS I/O Descriptions (Continued)

Signal Name	I/O	Type	Description	Default/ Optional
cout[19:0]]	O	R	Characterization test bus logic data output	D

1. During configuration, both HDOUTP and HDOUTN are pulled high to VCCOB.
2. The Generic 8b10b PCS module includes four PCI control and status signals as options. If not used, the control signals may be tied to GND and the status signals may be left float.
3. The only way to get a signal out without using the CDR is to use the OOB_OUT signal. This signal is available in SDI mode only.
4. These signals appear in the PCS port list when the external link state machine is selected. Refer to Figure 18-28.
5. For factory use only.
6. These signals are pulses. In order to properly monitor these status signals, they must be latched.

SERDES/PCS Functional Description

Devices in the LatticeECP2M family have from one to four quads of embedded SERDES/PCS logic. Each quad, in turn, supports four independent full-duplex data channels. A single channel can support a data link and each quad can support up to four such channels. Note that mode selection is done on a per quad basis. For example, the selection of Gigabit Ethernet mode for a quad dedicates all four channels in that quad to Gigabit Ethernet mode.

The embedded SERDES CDR PLLs and Tx PLLs support data rates which cover a wide range of industry standard protocols.

Figure describes the major blocks and sub-blocks in a SERDES/PCS channel.

- SERDES
  - Equalizer
  - CDR (Clock and Data Recovery)
  - Deserializer
  - PreEmphasis
  - Serializer
  - Serial Loopback
- SERDES Bridge (SB)
  - Inverter: inverts receive data. Required by PCI Express
  - SERDES Bridge Parallel Loopback
- PCS Core
  - Word Alignment
  - 8b10b Decoder
  - 8b10b Encoder
  - Link State Machine
  - Elastic Buffer (CTC)
- FPGA Bridge (FB)
  - Down-sample FIFO
  - Up-sample FIFO
  - PCS Parallel Loopback

SERDES

Equalizer

As the data rate of digital transmission advances over Gbps, frequency-dependent attenuation results in severe intersymbol interference in the received signal and makes it mandatory to use equalizer in the data transceiver to recover data correctly. Three pole positions are provided: low, medium and high frequency range.

Pre-Emphasis

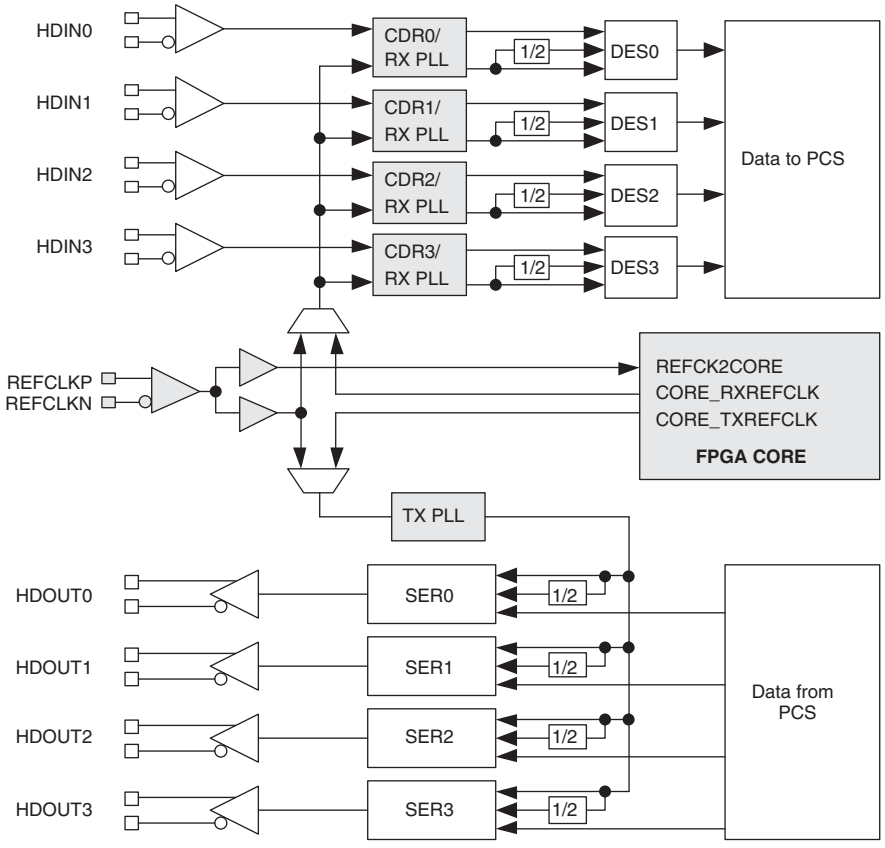
Pre-emphasis refers to a system process designed to increase the magnitude of some frequencies with respect to the magnitude of other frequencies. The goal is to improve the overall signal-to-noise ratio by minimizing the adverse effects of such phenomena as attenuation differences or saturation of recording media in subsequent parts of the system. User can select up to 80% of pre-emphasis.

Reference Clock Usage

One reference clock (REFCLK) is supported in the LatticeECP2M family. The Tx PLL and the four Rx PLLs all run at the same frequency, which is a multiple of the reference clock frequency. The Tx serializer in each channel can be independently programmed to run at this rate (full-data-rate mode) or half of this rate (half-data-rate mode). Similarly, the Rx deserializer in each channel can be independently programmed to run at this rate (full-data-rate mode) or half of this rate (half-data-rate mode). If all Tx and Rx are programmed in the same mode (normally this will be full-rate) then all four channels in the quad will run at the same frequency, both Tx and Rx.

The transmit PLL in the SERDES is able to lock to either an external reference clock from the pins, or a reference clock provided from the FPGA core (core_txrefclk). The receive CDRs in the SERDES is able to lock to either an external reference clock from the pins, or a reference clock provided by the FPGA core (core_rxrefclk).

Figure 18-5. Block Diagram, Reference Clock Usage



Reference Clock Sources

refclkp, refclkn

Dedicated CML input. This is the first choice unless different clock sources for rx and tx are used. The clock signal may be CML, LVDS or LVPECL. Refer to FPGA-TN-02077, Electrical Recommendations for Lattice SERDES, for example interface circuits.

core_rxrefclk, core_txrefclk

Reference clock from FPGA logic. The Primary Clock pad (PCLK) should be used as the clock input pin to the FPGA. The clock signal may be CML, LVDS, LVPECL or single-ended.

FPGA PLL

When an FPGA PLL is used as the reference clock, the reference clock to PLL should be assigned to a dedicated PLL input pad. The FPGA PLL output jitter may not meet system specifications at higher data rates. Use of an FPGA PLL is not recommended in jitter-sensitive applications.

Full-Data-Rate and Half-Data-Rate

Each Tx Serializer and Rx Deserializer can be split into full-data-rate and half-data-rate, allowing two different data rates in each direction and in each channel.

The Channel Based Protocol Mode must be selected to use this dual rate feature.

Example:

1. In the **Quad Tab window** of IPexpress (Figure 18-22), **G8B10B Mode, Channel Based Protocol Mode, Channel 0 (Full Rate) and Channel 1 (Half Rate)** are selected.
2. In the **Reference Clock (CM) window** (Figure 18-24), under the **Full Rate Channel** column, enter the following:
   - Serial Bit Clock Rate: 2.5 GHz
   - Reference Clock Multiplier: 10X
   - Leave the other three entries
3. The **Half Rate Channel** column will display calculated values for Half Rate:

	Full Rate Channel	Half Rate Channel
Serial Bit Clock Rate	2.5 GHz	1.25 GHz
Reference Clock Multiplier	10X	5X
Calculated Reference Clock Rate	250 MHz	250 MHz
FPGA Interface Data Bus Width	8	8
Calculated FPGA Interface Clock Rate	250 MHz	125 MHz

Full Clock, Half Clock and Quarter Clock Usage

In most cases, txfullclk is used for ff_rxiclk_chx, ff_txiclk_chx, ff_ebrd_clk_x as illustrated in Figure 18-32.

The IPexpress GUI automatically calculates the FPGA Interface Clock Frequency when Reference Clock Multiplier and FPGA Interface Data Bus Width is selected.

Table 18-5 illustrates clock usage examples in all possible combinations of the refclk_multiplier modes and 8-bit or 16-bit interface Data Bus widths.

Table 18-5. Clock Usage Example – G8B10B Mode, REFCLK = 120MHz

Reference Clock Multiplier	10xH	10x	20xH	20x
Bit Rate	**600 Mbps**	**1.2 Gbps**	**1.2 Gbps**	**2.4 Gbps**
8-Bit Interface Example				
rxfullclk[1]	60	120	120	240
rxhalfclk	30	60	60	120
txfullclk	120	120	240	240
txhalfclk	60	60	120	120
txqtrclk	30	30	60	60
16-Bit Interface Example				
rxfullclk	60	120	120	240
rxhalfclk[1]	30	60	60	120
txfullclk	120	120	240	240
txhalfclk	60	60	120	120
txqtrclk	30[2, 3]	30	60	60

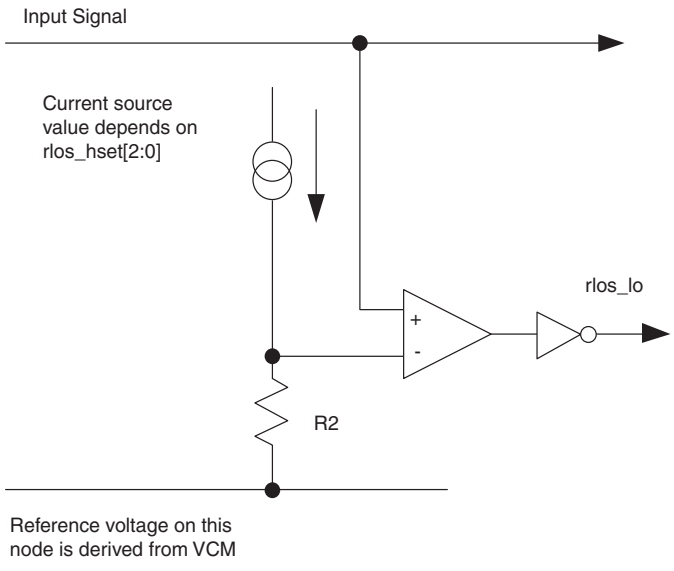
1. These recovered clocks are used as the source of rxiclk in CTC BYPASS mode. See Figures 18-34 and 18-36.
2. The clocks in the shaded cells are used as the FPGA interface clocks in each mode.
3. When this mode is selected, the 'PLL Quarter Clock' must be checked in the Optional Port tab window of the configuration GUI (see Figure 18-28).

The VCO in the Full Data Rate Channel is the same as the bit clock. In Half Data Rate Channel, the bit clock is half of the VCO.

Loss Of Signal (LOS)

Each channel contains a programmable loss-of-signal detector as shown in Figure 18-6. The loss-of-signal threshold depends on the value of the programmable current source. The current source value is chosen using the rlos_hset[2:0] control bits.

Figure 18-6. Loss-of Signal Detector



Note: rlos_lset[2:0] control bits and associated status bits are internal use only.

Loss Of Lock

Both the transmit PLL and the individual channel CDRs have digital counter-based, loss-of-lock detectors. If the transmit PLL loses lock, the loss-of-lock for the PLL is asserted and remains asserted until the PLL reacquires lock.

If a CDR loses lock, the loss-of-lock for that channel is asserted and locking to the reference clock retrains the VCO in the CDR. When this is achieved, loss-of-lock for that channel is de-asserted and the CDR is switched back over to lock to the incoming data. The CDR will either remain locked to the data, or will go back out of lock again in which case the re-training cycle will repeat.

Tx Lane-to-Lane Skew
A control bit, sync_toggle, has been added to reset all the active Tx channels to start serialization with bit0. Most multi-channel protocol standards have requirements to ensure that the Tx lane-to-lane skew is within a certain specification. This is to ensure that most of the Rx De-skew (Multi-channel alignment, which is not supported in hard PCS by Lattice ECP2M) is for channel (trace) de-skewing.

The reset to the Tx serializers is generated either by toggling the sync_toggle control bit or by a transition in PLL Loss of Lock. The reset is applied to all active Tx serializers. If both these source signals are level, then the Tx serializers are operating normally.

PCS Functional Setup
The LatticeECP2M PCS can be configured for use in various applications. Setup is chosen with the ispLEVER® IPexpress module generation tool which allows the user to select the mode and feature options for the PCS. Option selections are saved in an auto-configuration file which is subsequently used by the ispLEVER bitstream generator to write the user selections into the bitstream. To change PCS option selections it is recommended that the user re-run IPexpress to regenerate a PCS module and create a new auto-configuration file. Some options can be changed by manually editing the auto-configuration file before running the bitstream generator.

After configuration, PCS options can be changed dynamically by writing to PCS registers via the optional SERDES Client Interface (SCI) bus. The SERDES Client Interface is soft IP that allows the SERDES/PCS quad to be controlled by registers as opposed to configuration memory cells. A table of control and status registers accessible through the SCI is provided in the Memory Map section of this document.

Auto-Configuration File
Initial register setup for each PCS mode can be performed by using the autoconfiguration feature in ispLEVER. The module generator provides an auto-configuration file which contains the quad and channel register settings for the chosen mode. This file can be referred to for front-end simulation and also can be integrated into the bitstream. When an auto-configuration file is integrated into the bitstream all the quad and channel registers will be set to values defined in the auto-configuration file during configuration. The SCI (SERDES Client Interface) is therefore not needed if all quads are to be set via auto-configuration files. However, the SCI must be included in a design if the user needs to change control registers or monitor status registers during operation.

Transmit Data

The PCS quad transmit data path consists of 8b10b Encoder and Serializer per channel.

8b10b Encoder
This module implements an 8b10b encoder as described within the IEEE 802.3ae-2002 1000BASE-X specification. The encoder performs the 8-bit to 10-bit code conversion as described in the specification, along with maintaining the running disparity rules as specified. The 8b10b encoder can be bypassed on a per channel basis by setting the attribute CHx_8B10B to “BYPASS” where x is the channel number.

Serializer
The 8b10b encoded data undergoes parallel to serial conversion and is transmitted off chip via the embedded SERDES.

Receive Data

The PCS quad receive data path consists of the following sub-blocks per channel: Deserializer, Word Aligner, 8b10b Decoder, Optional Link State Machine, and Optional Receive Clock Tolerance Compensation (CTC) FIFO.

Deserializer
Data is brought on-chip to the embedded SERDES where it goes from serial to parallel.

Word Alignment (Byte Boundary Detect)
This module performs the comma codeword detection and alignment operation. The comma character is used by the receive logic to perform 10-bit word alignment upon the incoming data stream. The word aligner can be bypassed on a per channel basis by setting attribute CHx_COMMA_ALIGN to “BYPASS” where x is the channel number. The comma description can be found in section 36.2.4.9 of the 802.3.2002 1000BASE-X specification as well as section 48.2.6.3, Figure 48-7 of the 10GBASE-X specification.

A number of programmable options are supported within the word alignment module:

- Software enable control (in User Configured - UC mode).

  Note: UC_Mode refers to 8-bit SERDES Only, 10-bit SERDES Only, SD-SDI, HD-SDI.

- Ability to set two programmable word alignment characters (typically one for positive and one for negative disparity) and a programmable per bit mask register for alignment compare. Alignment characters and the mask register is set on a per quad basis. For many protocols, the word alignment characters can be set to “XXX0000011” (jhgfiedcba bits for positive running disparity comma character matching code groups K28.1, K28.5, and K28.7) and “XXX1111100” (jhgfiedcba bits for negative running disparity comma character matching code groups K28.1, K28.5, and K28.7). However the user can define any bit pattern up to 10 bits long.

- The first alignment character is defined by the 10-bit value assigned to attribute COMMA_A. This value applies to all channels in a PCS quad.

- The second alignment character is defined by the 10-bit value assigned to attribute COMMA_B. This value applies to all channels in a PCS quad.

- The mask register defines which word alignment bits to compare (a ‘1’ in a bit of the mask register means check the corresponding bit in the word alignment character register). The mask registers defined by the 10-bit value assigned to attribute COMMA_M. This value applies to all channels in a PCS quad.

When attribute CHx_COMMA_ALIGN is set to ‘AUTO’, one of the protocol based Link State machines will control word alignment. For more information on the operation of the protocol based Link State Machines, see the Protocol Specific Link State Machine description below.

8b10b Decoder
The 8b10b decoder implements an 8b10b decoder operation as described with the IEEE 802.3-2002 specification. The decoder performs the 10-bit to 8-bit code conversion along with verifying the running disparity. The 8b10b decoder can be bypassed on a per channel basis by setting attribute CHx_8B10B to “BYPASS” where x is the channel number.

When a code violation is detected, the ff_rxdata receive data is set to 0xEE with ff_rx_k_cntrl_ch set to ‘1’.

Protocol Specific Link State Machine
The PCS implements link state machines for various protocols that are used in various quad modes.

When a protocol specific Link State Machine is selected, that channel’s Link State Machine must be enabled by setting the protocol CH(0-3)_COMMA_ALIGN to “AUTO”. Selection of the specific Link State Machine that is enabled in each mode is described below and summarized in Figure 18-7.

The Link State Machine for Gigabit Ethernet is selected when attribute PROTOCOL is “GIGE”. Link synchronization is achieved after the successful detection and alignment of the required number of consecutive aligned code words. The Gigabit Ethernet link synchronization state machine implements the Synchronization State Diagram shown in Figure 36-9 of the 802.3- 2002 1000BASE-X specification.

In 'G8B10B' and '10-bit SERDES Only' protocols, the Gigabit Ethernet Link State Machine is used when COMMA_ALIGN is set to 'AUTO'.

External Link State Machine Option

When attribute CHx_COMMA_ALIGN is set to “DYNAMIC”, the protocol specific Link State Machines are bypassed. When ffc_enable_cgalign_ch(0-3) is high, the word aligner will lock alignment and stay locked. It will stop comparing incoming data to the user-defined word alignment characters and will maintain current alignment on the first successful compare to either the COMMA_A or COMMA_B. When ffc_enable_cgalign_ch(0-3) is pulsed low, the word aligner will re-lock on the next match to one of the user-defined word alignment characters. If desired, ffc_enable_cgalign_ch(0-3) can be controlled by a Link State Machine implemented externally to the PCS quad to allow a change in word alignment only under specific conditions.

Figure 18-7 illustrates the link state machine options.

Figure 18-7. PCS Word Aligner and Link State Machine Options

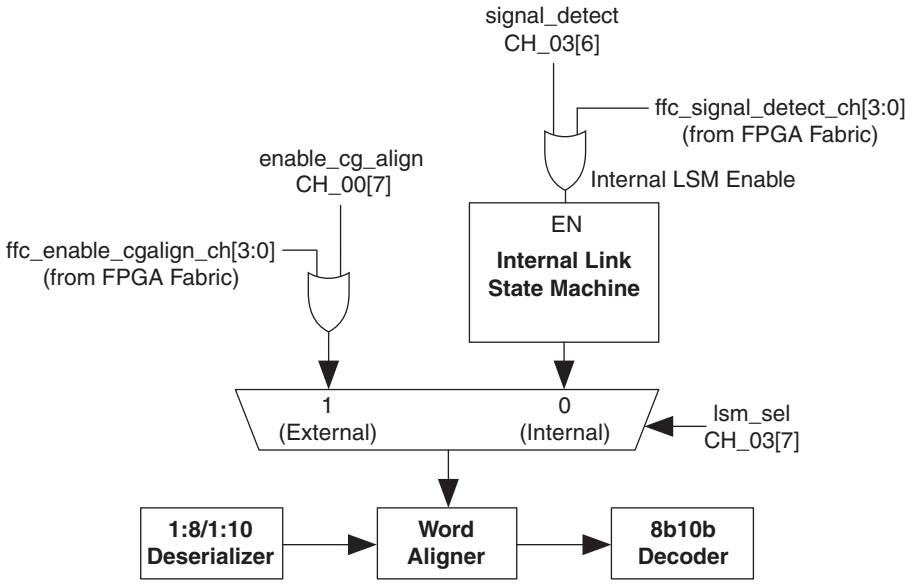


Table 18-6. Link State Machine and Word Aligner Selection

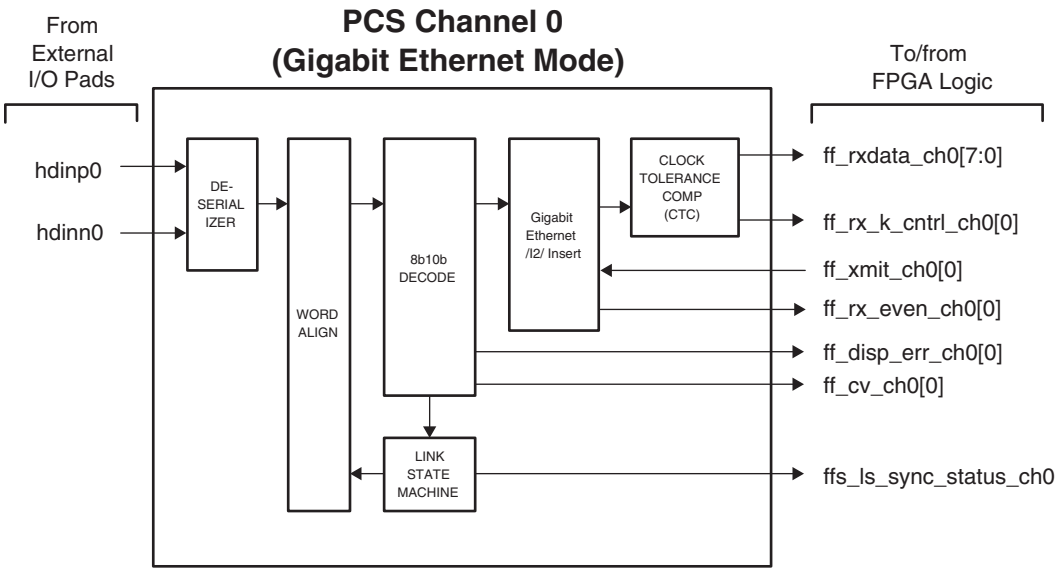
COMMA ALIGN Mode	Description
AUTO	WA enabled, LSM enabled (GbE LSM: default).
DYNAMIC	WA enabled, LSM disabled. The cg_align and sig_detect signals are set to 0 so that the potential external LSM to control both signals. The External Link State Machine option in the Optional Port tab of the GUI must be also selected.
BYPASS	WA bypassed, LSM disabled. Users may develop word aligner in the FPGA core and provide their own cg_align and sig_detect signals to the logic.

When a Link State Machine is selected and enabled, for a particular channel, that channel’s ffs_ls_sync_status_ch(0-3) status signal will go high upon successful link synchronization.

Idle Insert for Gigabit Ethernet Mode

Generic 8b10b mode also has the option to select the Link State Machine for word alignment. The PCS set to Gigabit Ethernet Mode provides for insertion of /I2/ symbols into the receive data stream for auto-negotiation. Gigabit Ethernet auto-negotiation is performed in soft logic. This function inserts a sequence of 8 /I2/ ordered sets every 2048 clock cycles. /I2/ insertion is controlled by the ff_xmit_ch(0-3) input to the PCS which is driven from the auto-negotiation soft logic. The signal ff_rx_even_ch(0-3)[0] from the PCS to the auto-negotiation soft logic is also provided. Figure 18-8 shows one channel (channel 0 in this example) of receive logic when the PCS is set to Gigabit Ethernet Mode showing these control/status signals.

Figure 18-8. PCS Receive path for Gigabit Ethernet Mode (Channel 0 Example)



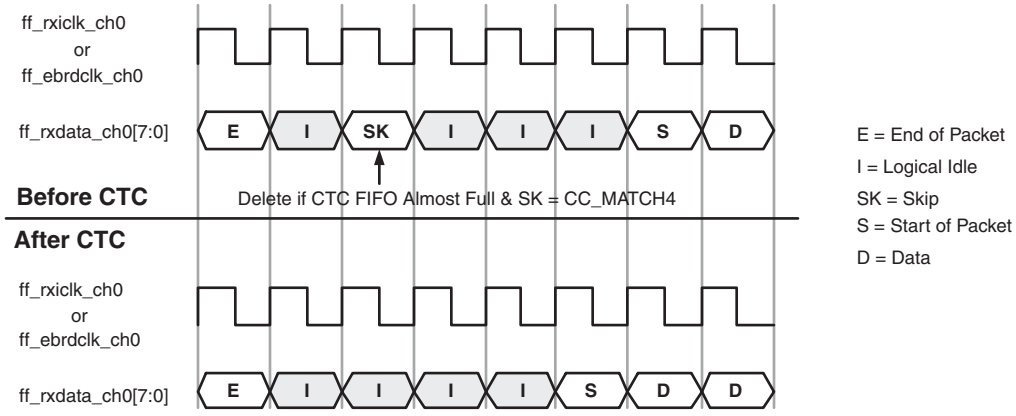
Clock Tolerance Compensation

The Clock Tolerance Compensation module performs clock rate adjustment between the recovered receive clocks and the locked reference clock. Clock compensation is performed by inserting or deleting bytes at pre-defined positions, without causing loss of packet data. A 16-Byte elasticity FIFO is used to transfer data between the two clock domains and will accommodate clock differences of up to the specified ppm tolerance for the LatticeECP2M SERDES (See DC and Switching Characteristics section of the LatticeECP2/M Family Data Sheet).

A channel has the Clock Tolerance Compensation block enable when that channel's attribute CHx_CTC_BYP is set to "NORMAL". The CTC is bypassed when that channel's attribute CHx_CTC_BYP is set to "BYPASS".

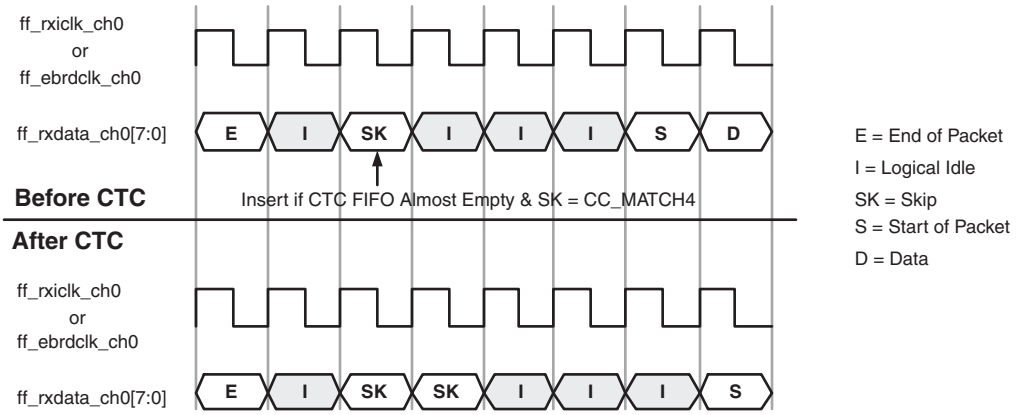
A diagram illustrating 1 byte deletion is shown in Figure 18-9:

Figure 18-9. Clock Tolerance Compensation 1 Byte Deletion Example



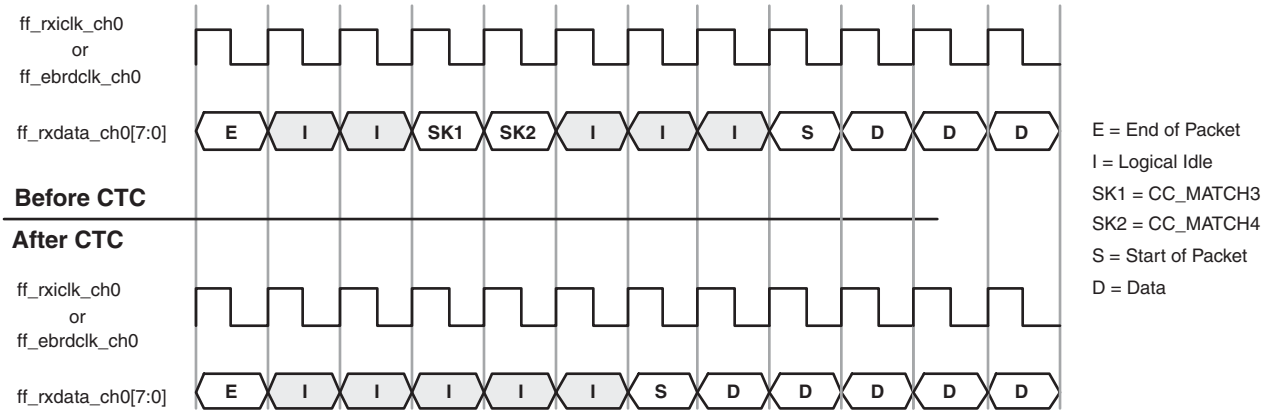
A diagram illustrating 1 byte insertion is shown in Figure 18-10:

Figure 18-10. Clock Tolerance Compensation 1 Byte Insertion Example



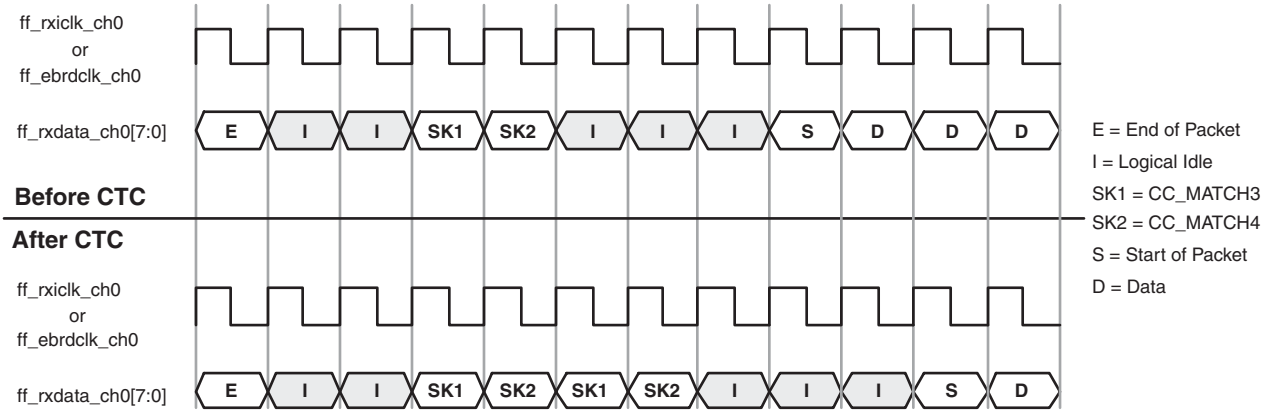
A diagram illustrating 2 byte deletion is shown in Figure 18-11:

Figure 18-11. Clock Tolerance Compensation 2 Byte Deletion Example



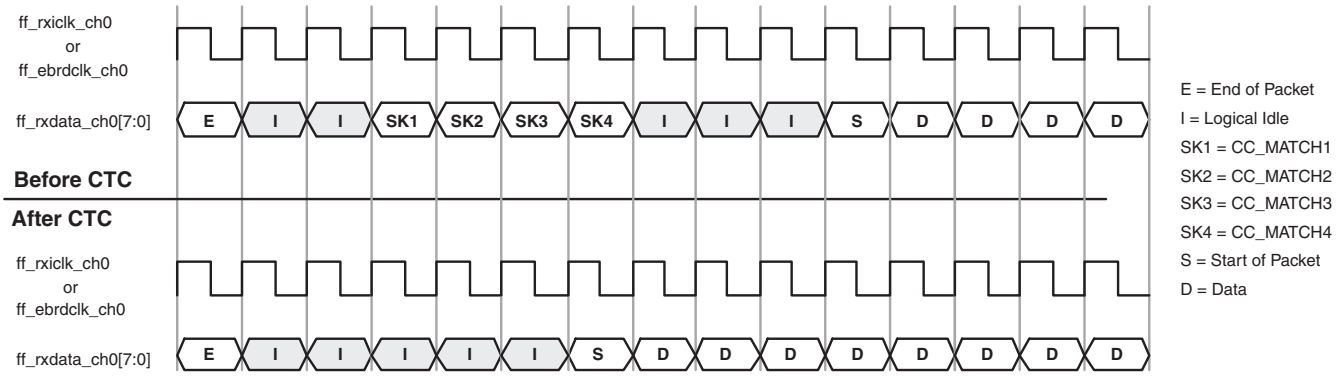
A diagram illustrating 2 byte insertion is shown in Figure 18-12:

Figure 18-12. Clock Tolerance Compensation 2 Byte Insertion Example



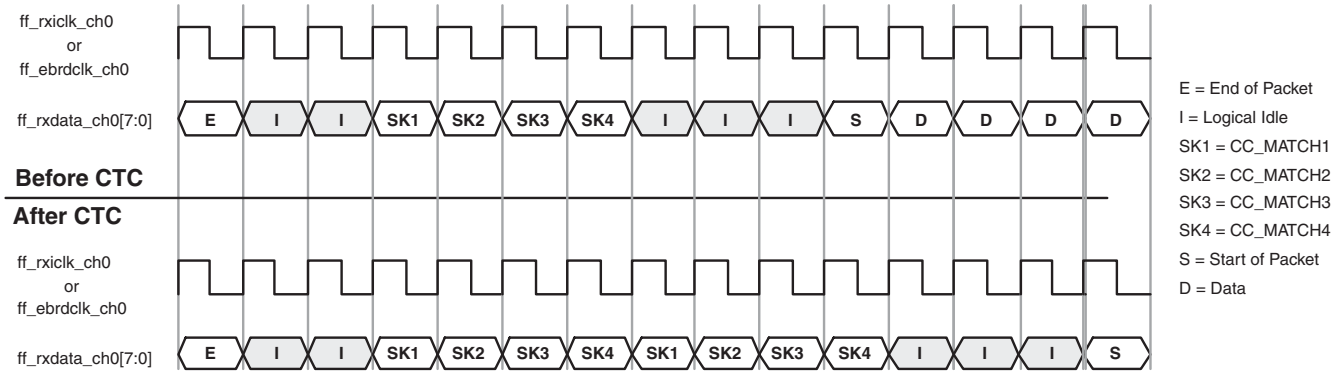
A diagram illustrating 4 byte deletion is shown in Figure 18-13:

Figure 18-13. Clock Tolerance Compensation 4 Byte Deletion Example



A diagram illustrating 4 byte insertion is shown in Figure 18-14:

Figure 18-14. Clock Tolerance Compensation 4 Byte Insertion Example



Clock compensation values are set on a quad basis. The CTC can be bypassed on a per channel basis by setting attribute CHx_CTC_BYP to “BYPASS” where x is the channel number. Setting CHx_CTC_BYP to “NORMAL” means the CTC is active. In the ispLEVER module generator, defining a channel as “Single” creates an auto-configuration file which enables CTC. Defining a channel as “MCA Group1” or “MCA Group 2” creates an auto-configuration file which bypasses CTC. When the CTC is used, the following settings for clock compensation must be set as appropriate for the intended application:

- Set the insertion/deletion pattern length using the CC_MATCHMODE attribute. This sets the number of skip bytes the CTC compares to before performing an insertion or deletion. Values for CC_MATCHMODE are “MATCH_4” (1 byte insertion/deletion), “MATCH_3_4” (2 bytes insertion/deletion), and “MATCH_1_2_3_4” (4 bytes insertion/deletion) to 1. The minimum inter-packet gap must also be set as appropriate for the targeted application. The inter-packet gap is set by assigning values to attribute CC_MIN_IPG. Allowed values for CC_MIN_IPG are “0”, “1”, “2”, and “3”. The minimum allowed inter-packet gap after skip character deletion is performed based on these attribute settings is described in Table 18-7 below.
- The Skip byte or ordered set must be set corresponding to the CC_MATCHMODE chosen. For 4 byte insertion/deletion (CC_MATCHMODE = “MATCH_1_2_3_4”), the first byte must be assigned to attribute MATCH_1, the second byte must be assigned to attribute MATCH_2, the third byte must be assigned to attribute MATCH_3, and the fourth byte must be assigned to attribute MATCH_4. Values assigned are 10 bit binary values. For example, if a 4 byte skip ordered set is /K28.5/D21.4/D21.5/D21.5, then “MATCH_1” should be “0110111100”, “MATCH_2”, = “0010010101”, and “MATCH_3” = “MATCH_4” = “0010110101”. For 2 byte insertion/deletion (CC_MATCHMODE = “MATCH_3_4”), the first byte must be assigned to attribute MATCH_3, and the second byte must be assigned to attribute MATCH_4. For 1 byte insertion/deletion (CC_MATCHMODE = “MATCH_4”), the skip byte must be assigned to attribute MATCH_4.

- The clock compensation FIFO high water and low water marks must be set to appropriate values for the targeted protocol. Values can range from 0 to 15 although the high water mark must be set to a value higher than or equal to the low water mark. The high water mark is set by assigning a value to attribute CCHMARK. Allowed values for CCHMARK are hex values ranging from “0” to “F”. The low water mark is set by assigning a value to attribute CCLMARK. Allowed values for CCLMARK are hex values ranging from “0” to “F”.
- Clock compensation FIFO overrun can be monitored on a per channel basis on the PCS/FPGA interface port labeled ffs_cc_overrun_ch(0-3) if “Error Status Ports” is selected when generating the PCS block with the isp-LEVER module generator.
- Clock compensation FIFO underrun can be monitored on a per channel basis on the PCS/FPGA interface port labeled ffs_cc_underrun_ch(0-3) if “Error Status Ports” is selected when generating the PCS block with the isp-LEVER module generator.

Calculating Minimum Inter-packet Gap

Table 18-7 shows the relationship between the user-defined values for inter-packet gap (defined by the CC_MIN_IPG attribute), and the guaranteed minimum number of bytes between packets after a skip character deletion from the PCS. The table shows the inter-packet gap as a multiplier number. The minimum number of bytes between packets is equal to the number of bytes per insertion/deletion times the multiplier number shown in the table. For example, if the number of bytes per insertion/deletion is 4 (CC_MATCHMODE is set to “MATCH_1_2_3_4”), and the minimum inter-packet gap attribute C_MIN_IPG is set to “2”, then the minimum inter-packet gap is equal to 4 (CC_MATCHMODE = “MATCH_1_2_3_4”) times 3 (Table 18-7 with CC_MIN_IPG = “2”) or 12 bytes. The PCS will not perform a skip character deletion until the minimum number of inter-packet bytes have passed through the CTC.

Table 18-7. Minimum Inter-packet Gap Multiplier

CC_MIN_IPG	Insertion/Deletion Multiplier Factor
“0”	1 X
“1”	2 X
“2”	3 X
“3”	4 X

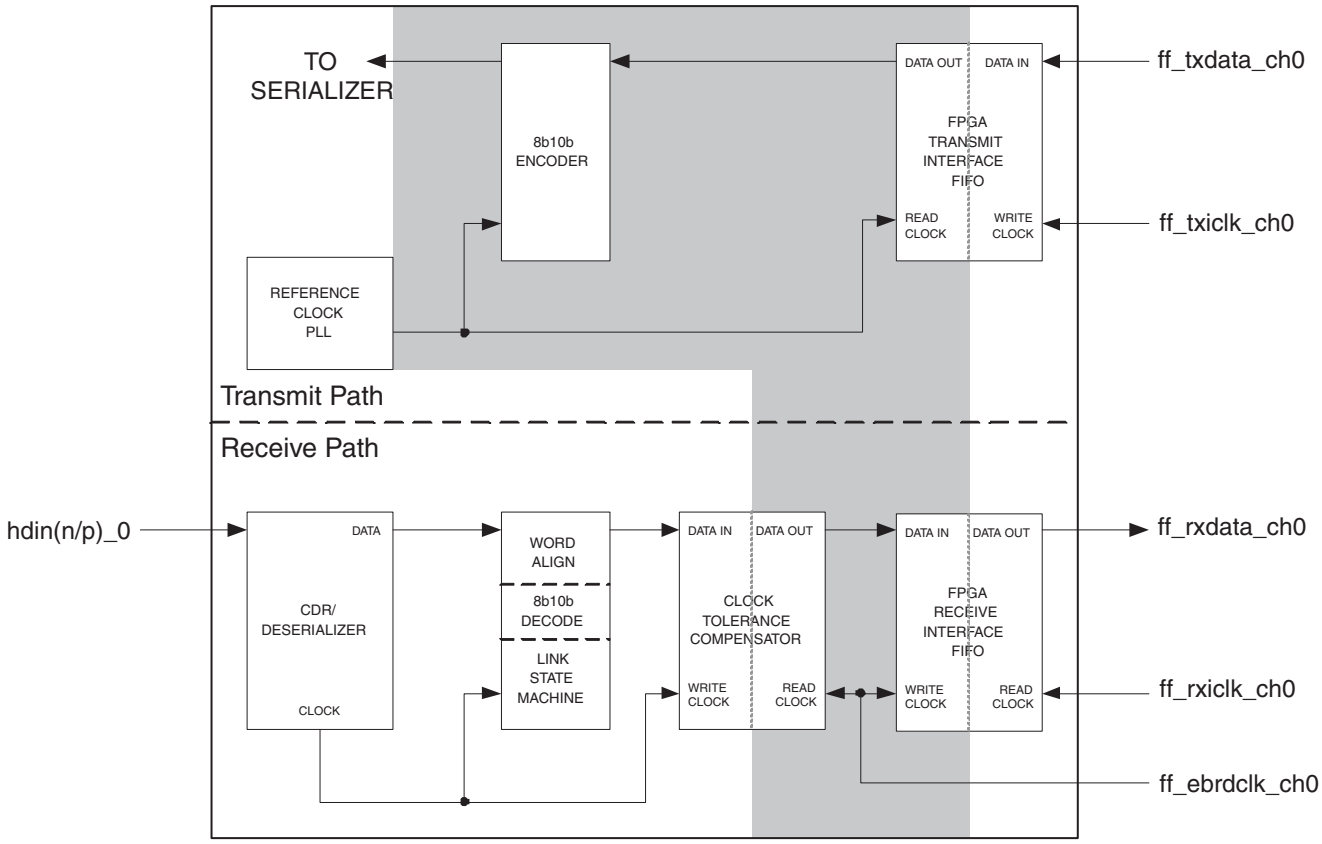
Clock Domains

Figure 18-15 shows the clock domains for both transmit and receive directions for a single channel inside the PCS for modes which utilize the Clock Tolerance Compensation (CTC) block.

On the transmit side, a clock domain transfer from the ff_txiclk_ch input at the FPGA interface to the locked reference clock occurs in the FPGA Transmit Interface FIFO. The FPGA Transmit Interface FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. These FIFOs (one per channel) cannot compensate for frequency variations.

On the receive side, a clock domain transfer occurs from the channel recovered clocks to the locked reference clock at the Clock Tolerance Compensator (CTC) block. The CTC can adjust for frequency differences between the recovered receive clock and the reference clock up to the maximum phase difference specification for the LatticeECP2M. Downstream from the CTC, another clock interface occurs between the locked reference clock and the ff_rxiclk_ch at the FPGA Receive Interface FIFO. The FPGA Receive Interface FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. The FPGA Receive Interface FIFOs (one per channel) cannot compensate for frequency variations.

Figure 18-15. PCS Clock Domain Transfers for CTC Modes



To guarantee a synchronous interface, both the input transmit clocks and input receive clock should be driven from one of the output reference clocks for a PCS quad set to Generic 8b10b mode. Figure 18-16 illustrates the possible connections that would result in a synchronous interface.

Figure 18-16. Synchronous Input Clocks to PCS Quad with CTC Used

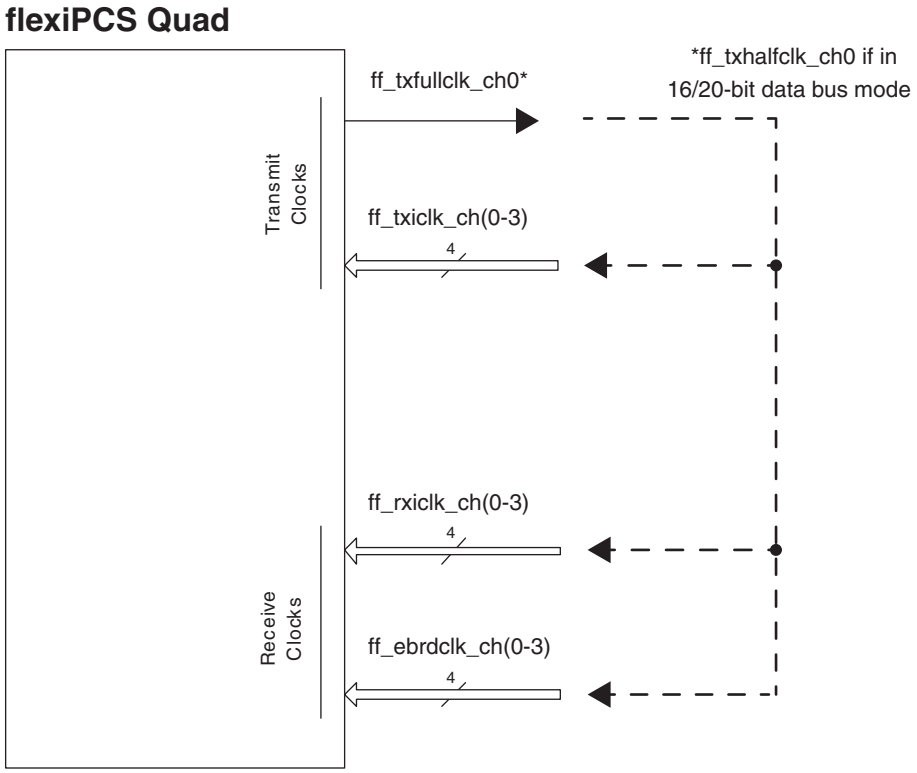
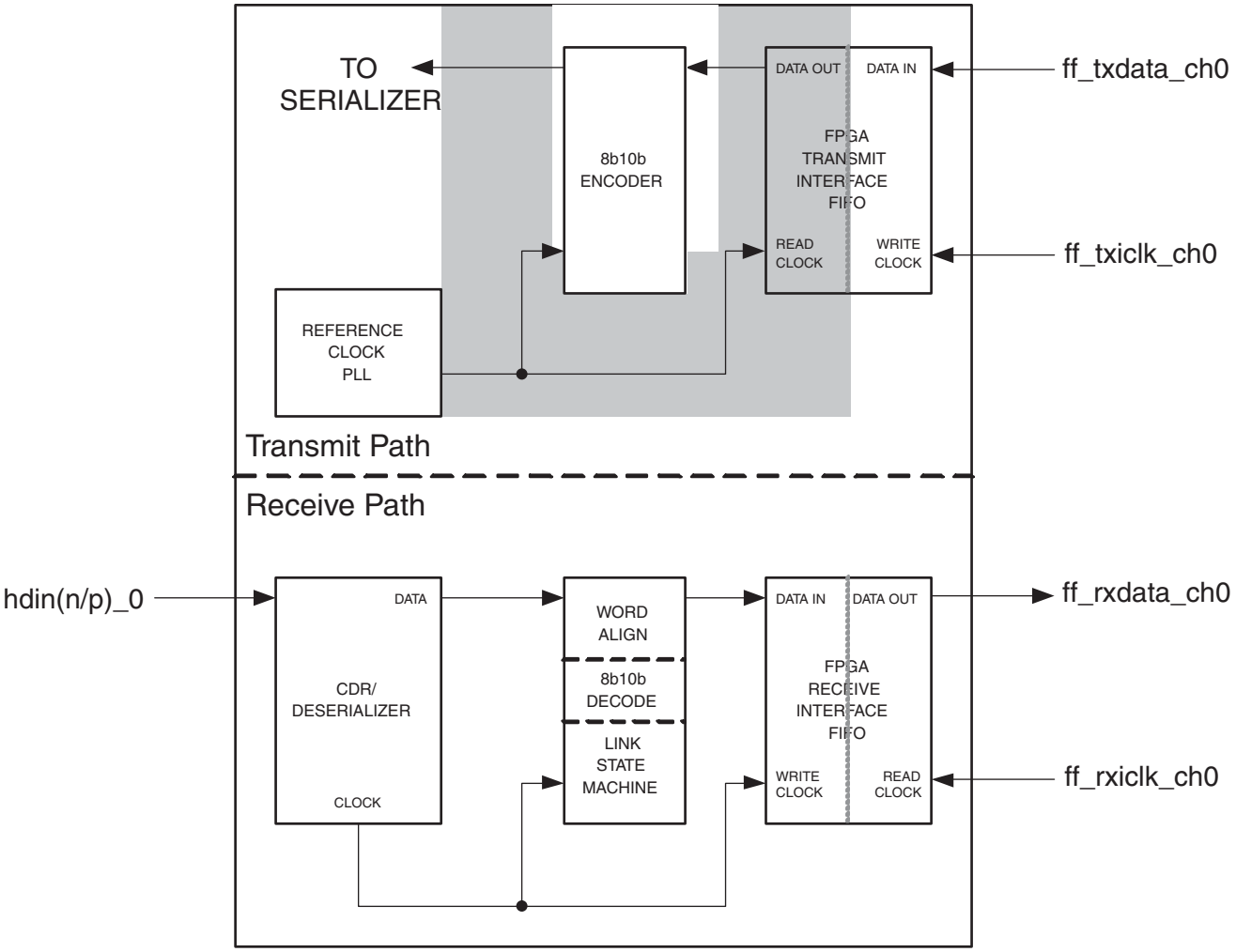


Figure 18-17 shows the clock domains for both transmit and receive directions for a single channel inside the PCS which do not utilize the Clock Tolerance Compensation (CTC) block.

On the transmit side, a clock domain transfer from the ff_txiclk_ch input at the FPGA interface to the locked reference clock occurs in the FPGA Transmit Interface FIFO. The FPGA Transmit Interface FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. These FIFOs (one per channel) cannot compensate for frequency variations.

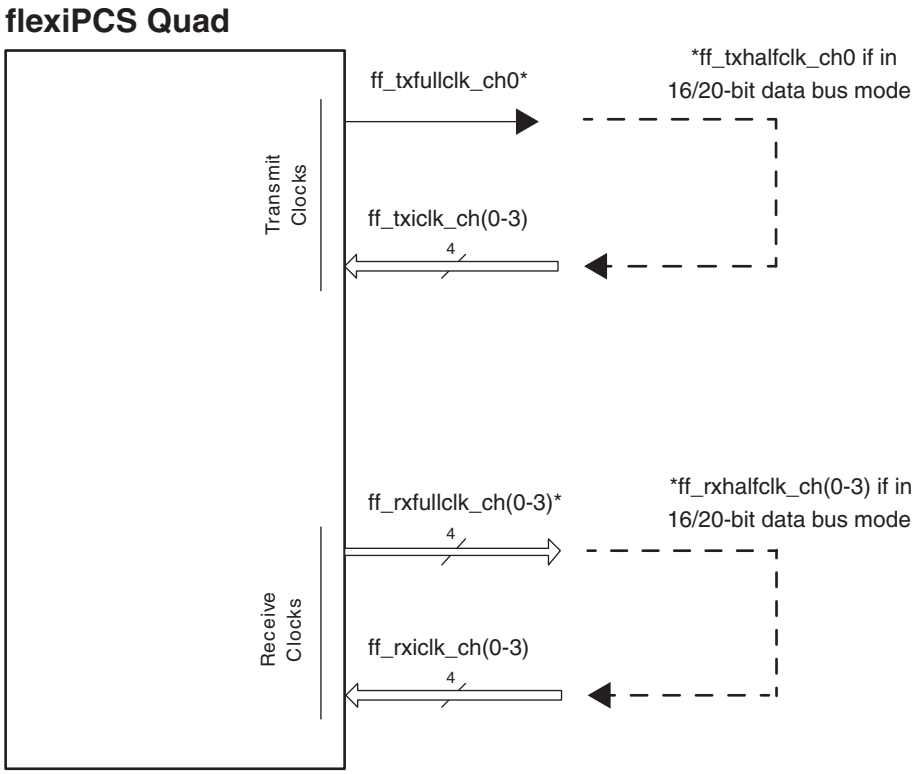
On the receive side, a clock domain transfer occurs between the recovered receive clock and the ff_rxiclk_ch at the FPGA Receive Interface FIFO. The FPGA Receive Interface FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. The FPGA Receive Interface FIFOs (one per channel) cannot compensate for frequency variations.

Figure 18-17. PCS Clock Domain Transfers for Non-CTC Modes



To guarantee a synchronous interface, both the input transmit clocks and input receive clock should be driven from one of the output reference clocks for a PCS quad set to Generic 8b10b mode. Figure 18-18 illustrates the possible connections that would result in a synchronous interface.

Figure 18-18. Synchronous Input Clocks to PCS Quad without CTC



Spread Spectrum Clocking (SSC) Support

LatticeECP2M SERDES/PCS does not have a Spread Spectrum generator but it can receive Spread Spectrum data.

SSC support in LatticeECP2M applies to all protocols that require similar support. The ports on the two ends of Link must transmit data at a rate that is within 600pm of each other at all times. This is specified to allow bit-rate clock source with a +/- 300ppm tolerance. The data rate can be modulated from +0% to -0.5% of the nominal data rate, at a modulation rate in the range of 30KHz to 33KHz. Along with the +/- 300ppm tolerance limit, both ports require the same bit rate clock when the data is modulated with an SSC.

The root complex is responsible for spreading the reference clock. The endpoint then uses that same clock to pass back the spectrum through the TX. Thus, there is no need for a separate RXREFCLK.

A predominant application of this is the add-in card. Add-in cards are not required to use the REFCLK from the connector but must receive and transmit with the same SSC as the PCI Express connector REFCLK.

Serial Digital Video and Out-Of-Band Low Speed SERDES Operation

The SERDES receiver buffers can be used to input low speed (<250Mbps: Out-Of-Band signal, OOB) by bypassing the receiver CDR and associated SERDES/PCS logic. This feature is useful for applications where the same pin is needed for both high speed and low speed data transfers down to the DC rate, as required by Serial Digital Video applications.

LatticeECP2M SERDES/PCS supports the standard definition serial digital interface, SD-SDI(143Mbps, 177Mbps, 270Mbps, 360Mbps) and the high definition serial digital interface, HD-SDI (1.485Gbps and 1.483.5Gbps).

It is required that both of these share the same receive and transmit pins. One possible implementation is shown in Figure 18-19.

The Out-Of-Band (OOB) signal port at the PCS/FPGA interface is used to input a signal slower than 250 Mbps.

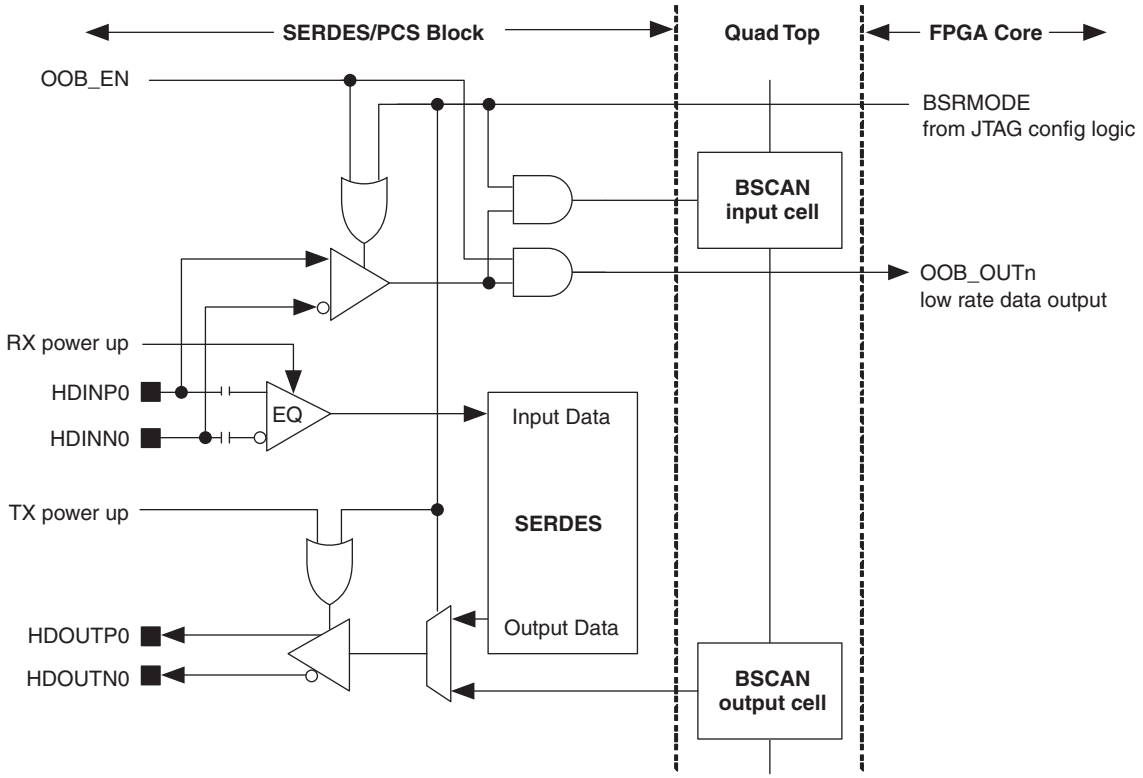
This port is available when the SD-SDI mode is selected. The OOB_OUT signal is passed directly from the SERDES input buffers to the FPGA interface and is not locked to the reference clock (see Figure 18-3).

When driving a SERDES input buffer with a low speed signal, the SERDES input buffer should be set to DC mode, which is done on a per-channel basis by selecting 'DC' in the RX I/O Coupling drop-down box of the IPexpress PCS configuration GUI (see Figure 18-25).

Though the discussion above talks about Serial Digital Video, it should be noted that similar usage is intended for any low bit rate applications that will do RX Clock Data Recovery in the FPGA logic and require decimation in the TX direction.

The input BSCAN circuit appears in parallel with the high-speed SERDES and can be used to route input data to a lower-speed Deserializer located elsewhere in the device (not in the quad). An enable signal (one per channel) is required to turn on the input BSCAN circuit independent of the BSCAN state machine.

Figure 18-19. One Possible Implementation of Serial Digital Video Support



Configuration GUIs

IPexpress is used to create and configure SERDES and PCS blocks. Designers use the graphical user interface (GUI) to select the SERDES Protocol Standard for a particular quad or channel. IPexpress takes the input from this GUI and generates a configuration file (.txt file) and HDL netlist. The HDL model is used in the simulation and synthesis flow. The configuration file contains attribute level map information. This file is input for simulation and the ispLEVER bitgen program. It is strongly recommended that designers make changes and updates in IPexpress and then regenerate the configuration file. In some exceptional occasions, users can modify the configuration file. Figure 18-20 shows the tools flow when using IPexpress to generate the SERDES/PCS block for the SERDES Protocol Standard.

When a project is saved in a different directory, this configuration file (.txt) should be manually moved to the same directory as the project file.

Figure 18-20. SERDES_PCS ispLEVER User Flow

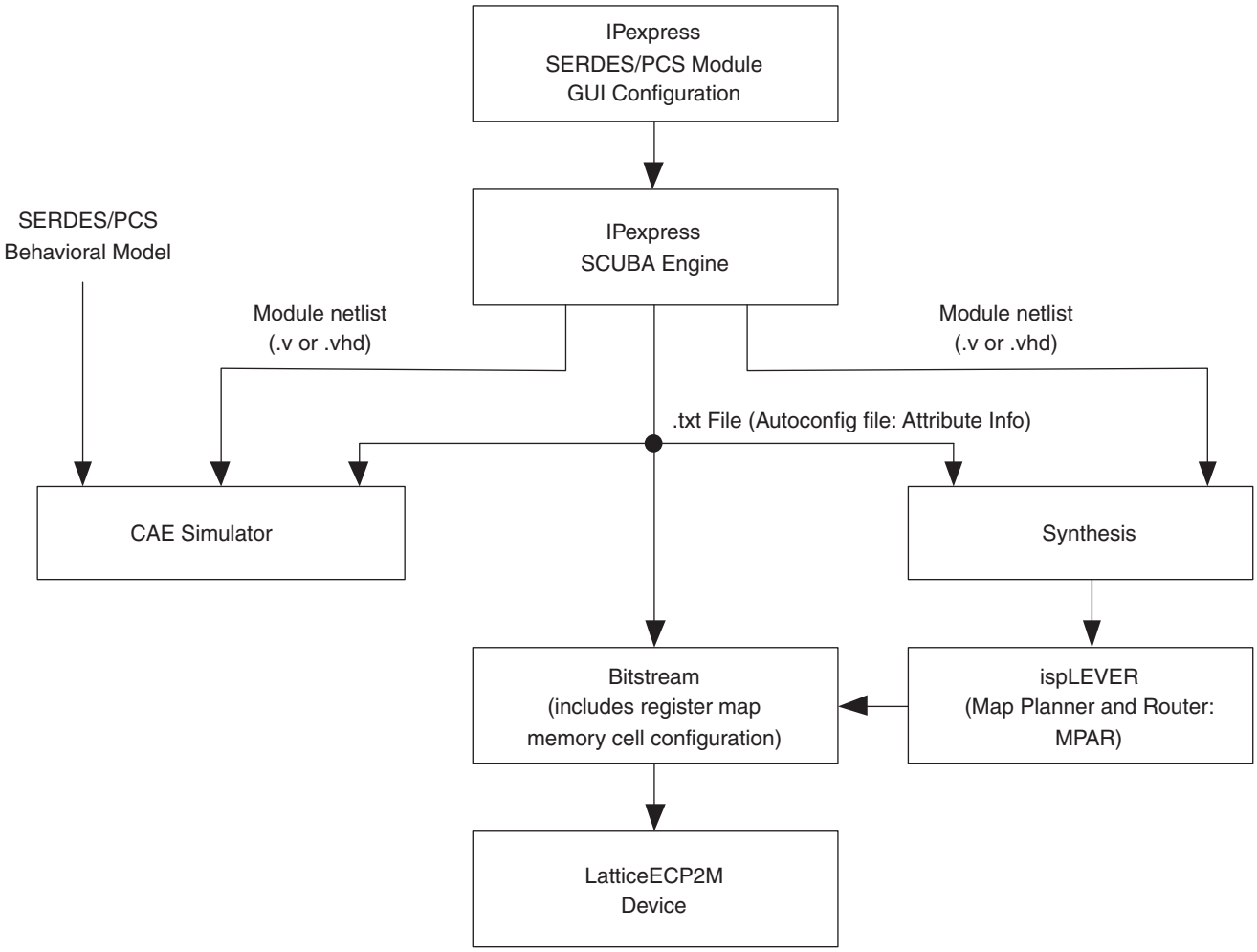
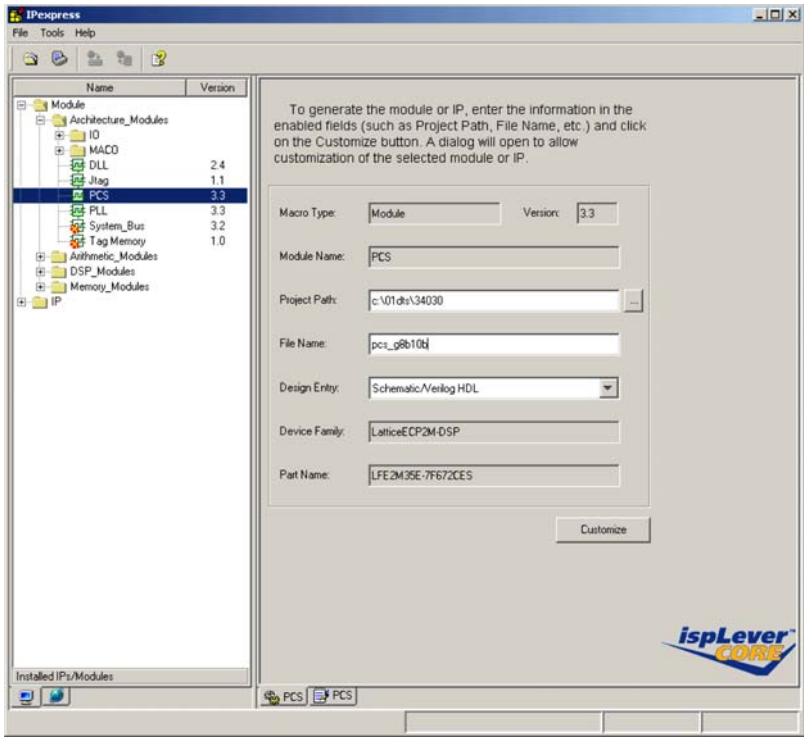


Figure 18-21 shows the main window when PCS is selected in the IPexpress GUI.

Figure 18-21. IPexpress PCS Main Window



Quad Setup Tab

Figure 18-22 shows the Quad Setup Tab window when the file name is entered and the **Customize** button is checked in the main window. The first entry required in this window is to select Protocol Mode from Quad Based Mode or Channel Based Mode. Other entries on this screen are channel selection and group selections. There are five additional tabs shown in Figures 18-23 through 18-28, listing all the user-accessible attributes with default values settings:

Figure 18-22. Configuration GUI - Quad Setup Tab

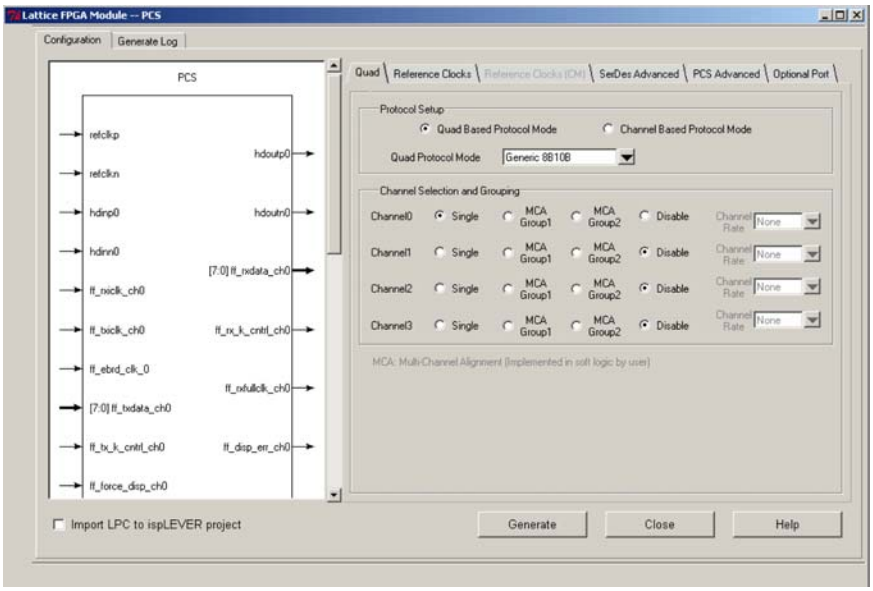


Table 18-8. SERDES_PCS GUI Attributes - Quad Tab Setup

GUI Text	Attribute Names	Range	Default Value
Protocol Setup		Quad Based Protocol Mode, Channel Based Protocol Mode	Quad Based Protocol Mode
Quad Protocol Mode	PROTOCOL	PCI Express, Gigabit Ethernet, Generic 8b10b, 10-bit SERDES Only, 8-bit SERDES Only, SD-SDI, HD-SDI[2]	Generic 8b10b
Single	CH_MODE	Enable the channel	Disable
MCA Group1[3]	CH_MODE	Multi-Channel Alignment Group 1	Disable
MCA Group2[3]	CH_MODE	Multi-Channel Alignment Group 2	Disable
Disable	CH_MODE	Disable the channel	Disable
Channel Rate[1]	CH_MODE	Full Rate, Half Rate	Full Rate

1. Channel Rate selection is applicable only in the Channel Based Protocol Mode.
2. Protocol Attribute Names: PCI Express = PCIE, Gigabit Ethernet = GIGE, Generic 8b10b = G8B10B, 8-bit SERDES Only = 8BSER, 10-bit SERDES Only = 10BSER, SD-SDI = SDSDI, HD-SDI = HDSDI.
3. Multi-Channel-Alignment is for transmitter lane-to-lane skew alignment. Receiver Multi-Channel-Alignment is not provided in hard PCS. MCA Group1 and MCA Group2 set the channels to CTC Bypass mode so that users can build the Multi-Channel Alignment in the FPGA core. The two groups are provided for identification of channels in Multi-Protocol applications.

Reference Clock Setup Tab

In this tab, the attributes of the Tx and Rx reference clock sources are selected. Users can select either a REFCLK or CORE_RXREFCLK as a Rx reference clock source and CORE_TXREFCLK as a Tx reference clock source for the quad. Similarly, in a quad all the channels use the same common Tx and Rx reference sources. Further, there is a tool to provide the required clock rate and multiplier settings for a particular data rate. In addition, for a given data bus width the tool provides the required clock rate to interface the quad to the core.

Figure 18-23. Configuration GUI - Reference Clocks Setup Tab

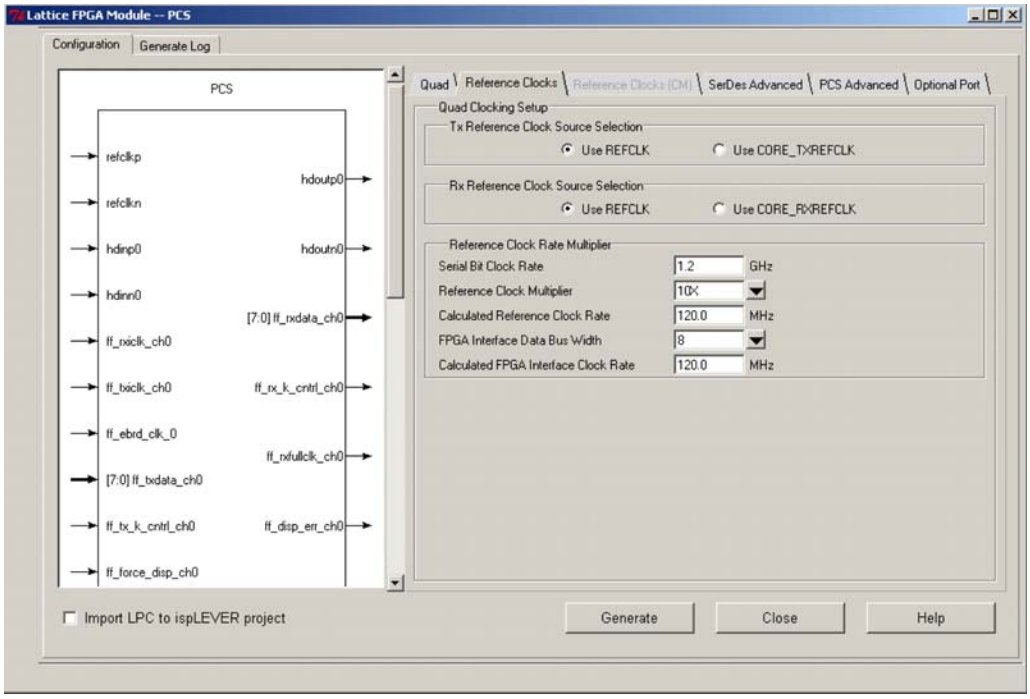


Table 18-9. SERDES_PCS GUI Attributes (LatticeECP2M) - Reference Clocks Setup Tab

GUI Text	Attribute Names	Button/ Box Type	Range	Default Value	Comment
Tx Reference Clock Source Selection	PLL_SRC	Radio	REFCLK, CORE_TXREFCLK	REFCLK	
Rx Reference Clock Source Selection	CH0_CDR_SRC CH1_CDR_SRC CH2_CDR_SRC CH3_CDR_SRC	Radio	REFCLK, CORE_RXREFCLK	REFCLK	
Serial Bit Clock Rate (GHz)	DATARANGE[1]	Check Box	0.27 to 3.125	2.5	LOW: MEDLOW: MED: MEDHIGH: HIGH:
Reference Clock Multiplier	CH0_REFCK_MULT CH1_REFCK_MULT CH2_REFCK_MULT CH3_REFCK_MULT	Drop Down	See Table 18-10	25X	
Calculated Reference Clock Rate (MHz)[2]	CP: REFCLK_RATE	Text Box	—		Not an editable field
FPGA Interface Data Bus Width	CH0_DATA_WIDTH CH1_DATA_WIDTH CH2_DATA_WIDTH CH3_DATA_WIDTH	Drop Down	See Table 18-10	8	
Calculated FPGA Interface Clock Rate (MHz)[2]	CP: FPGAINTCLK_RATE	Text Box	—		Not an editable field

1. DATARANGE:
   For Production Devices: Low ≤ 500 Mbps, 500 Mbps < Medlow ≤ 1.0 Gbps, 1.0 Gbps < Med < 2.0 Gbps, 2.0 Gbps ≤ Medhigh < 2.5 Gbps, 2.5 Gbps ≤ High ≤ 3.2 Gbps
   For Engineering Samples: Low ≤ 540 Mbps, 540 Mbps < Medlow ≤ 1.0 Gbps, 1.0 Gbps < Med < 2.0 Gbps, 2.0 Gbps ≤ Medhigh < 2.5 Gbps, 2.5 Gbps ≤ High ≤ 3.2 Gbps
   Refer to Table 18-99 for details on the control bits setting.
2. 8-bit SERDES Only Mode and 10-bit SERDES Only Mode bypass the Link Align/Comma Align, 8b10b encoder/decoder and the CTC. It does not bypass the CDR.

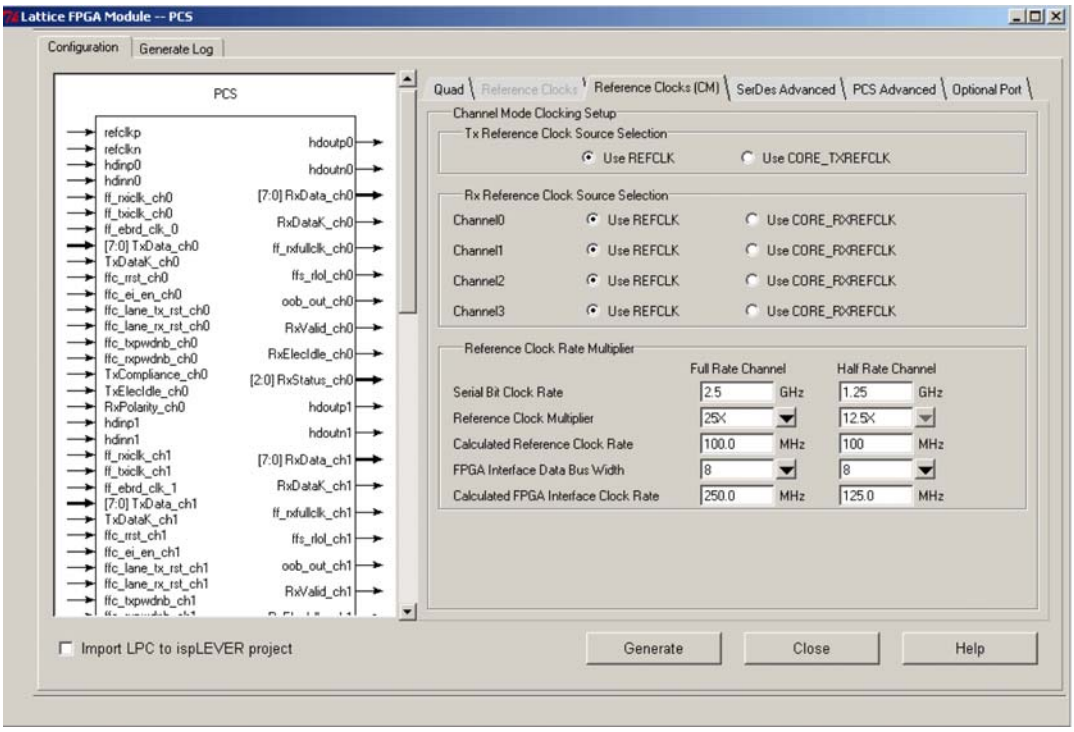
Table 18-10. Reference Clock Multiplier and FPGA Interface Data Bus Width by Protocol

Protocol	Reference Clock Multiplier	FPGA Interface Data Bus Width
PCI Express	20X, 25X	8, 16
GbE	10XH, 10X, 20X	8, 16
G8B10B	10XH, 10X, 20X	8, 16
10-bit SERDES Only	10XH, 10X, 20X	10, 20
8-bit SERDES Only	8HX, 8X, 16X	8, 16

Reference Clock Setup Tab (Channel Mode)

In this tab, the Tx reference clock selected is common to all channels but Rx reference clocks can be either REFCLK or CORE_RXREFCLK by channel.

Figure 18-24. Configuration GUI - Reference Clocks Setup Tab (Channel Mode)



When a user selects **Channel Based Protocol Mode** in the **Quad** tab and sets a channel or channels as half rate mode, this tab displays the half rate mode clock data.

SERDES Advance Setup

This tab is used to access the advanced attributes of the transmit and receive SERDES for all four channels. Transmit attributes such as PreEmphasis, termination, differential output voltage selection are selected. Receive attributes such as equalization, termination, I/O coupling are selected. Attributes for Transmit SERDES clock and PLL are also selected.

Figure 18-25. Configuration GUI - SERDES Advanced Setup Tab

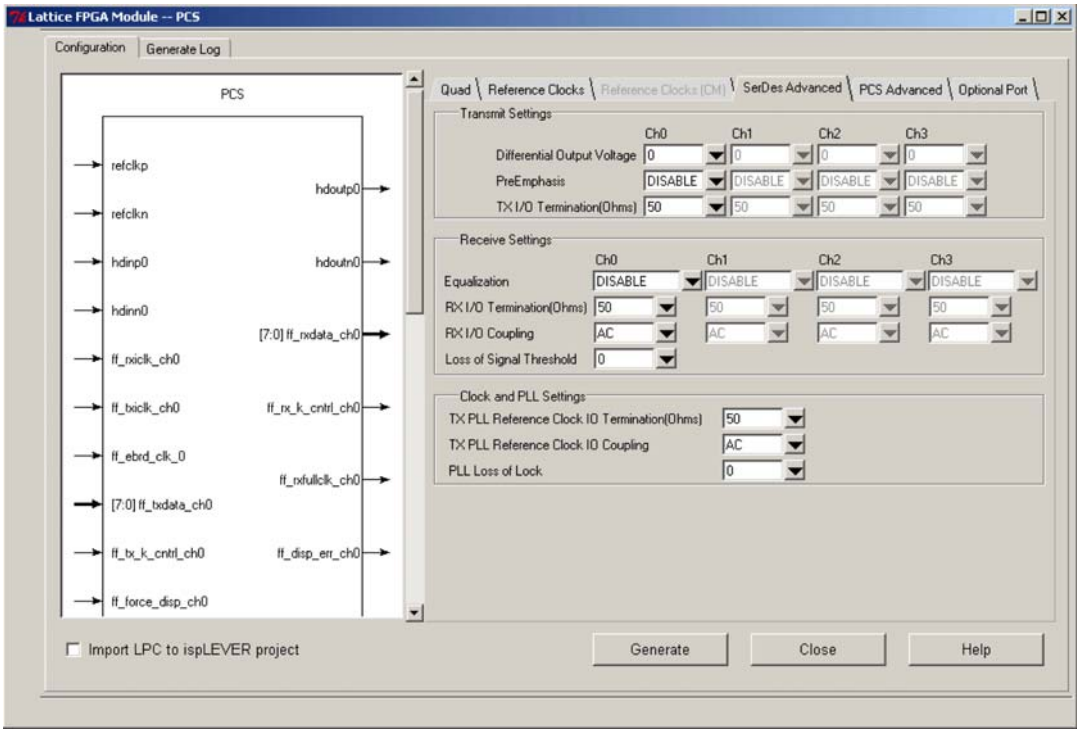


Table 18-11. SERDES_PCS GUI Attributes (LatticeECP2M) - SERDES Advanced Setup Tab

GUI Text		Attribute Names	Button/ Box Type	Range		Default Value
				PCI Express, GIGE	G8B10B, 8bSER 10bSER	
Differential Output Voltage	LatticeECP2M-35	CH0_TDRV_AMP CH1_TDRV_AMP CH2_TDRV_AMP CH3_TDRV_AMP	Drop Down	0(1040mV: default), 1(1280mV), 2 (1320mV), 3 (1360mV), 4 (640mV), 5 (760mV), 6 (870mV), 7 (990mV)		0
	All other devices			0(990mV: default), 1(1250mV), 2 (1300mV), 3 (1350mV), 4 (610mV), 5 (730mV), 6 (820mV), 7 (940mV)		
PreEmphasis	LatticeECP2M-35	CH0_TX_PRE CH1_TX_PRE CH2_TX_PRE CH3_TX_PRE	Drop Down	Disable, 0 (0%), 1 (16%), 2 (36%), 3 (40%), 4 (44%), 5 (56%), 6 (80%)		DISABLE
	All other devices			Disable, 0 (0%), 1 (12%), 2 (26%), 3 (30%), 4 (33%), 5 (40%), 6 (53%)		
Tx I/O Termination (Ohms)[3]		CH0_RTERM_TX CH1_RTERM_TX CH2_RTERM_TX CH3_RTERM_TX	Drop Down	50, 75, 5K		50
Equalization[1]		CH0_RX_EQ CH1_RX_EQ CH2_RX_EQ CH3_RX_EQ	Drop Down	Mid_High, Long_High Disable	Mid_Low, Mid_Med Mid_High, Long_Low Long_Med, Long_High Disable	DISABLE
Rx I/O Termination (Ohms)[3]		CH0_RTERM_RX CH1_RTERM_RX CH2_RTERM_RX CH3_RTERM_RX	Drop Down	50, 60, 75, High		50
Rx I/O Coupling		CH0_RX_DCC CH1_RX_DCC CH2_RX_DCC CH3_RX_DCC	Drop Down	AC, DC		AC[2]
Loss of Signal Threshold		LOS_THRESHOLD	Drop Down	0 (default), 1 (+10%), 2 (+15%),3 (+25%) 4 (-10%), 5 (-15%), 6 (-25%), 7 (-30%)		0
Tx PLL Reference Clock I/O Termination (Ohms)[3]		PLL_TERM	Drop Down	50, 2K		50
Tx PLL Reference Clock I/O Coupling		PLL_DCC	Drop Down	AC, DC		AC
PLL Loss of Lock		PLL_LOL_SET	Drop Down	Lock	Unlock	0
				0 (+/-600ppmx) 1 (+/-300ppm) 2 (+/-1500ppm) 3 (+/-4000ppm)	0 (+/-1200ppm) 1 (+/-2000ppm) 2 (+/-2200ppm) 3 (+/-6000ppm)	

1. Refer to Appendix D for details.
2. The typical capacitor value of the internal on-chip AC coupling is 5 pF.
3. Termination resistors and their usage

RX I/O Termination:

- 50: So far all of the protocols except SMTPE use 50 Ohms termination resistor.
- 60: Provided for flexibility purpose only.
- 75: SMPTE uses 75 Ohm termination resistor.
- HIGH: Such as PCI Express Rx detection.

TX I/O Termination:

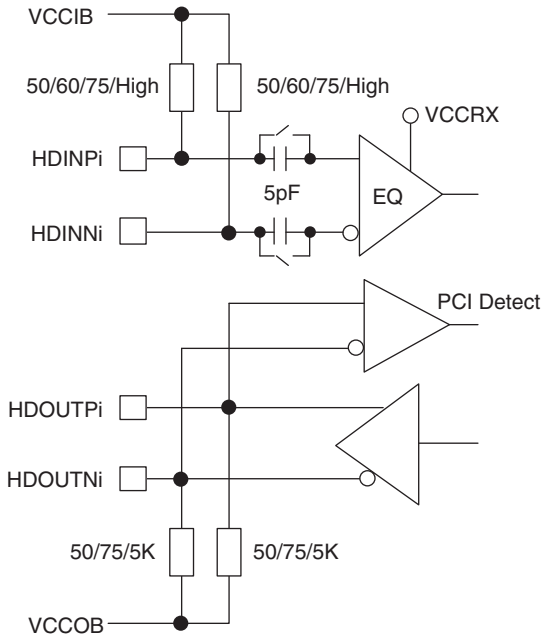
- 50: So far all of the protocols except SMTPE use 50 Ohms termination resistor.
- 75: SMPTE uses 75 Ohm termination resistor.
- 5K: Such as PCI Express electric idle and PCI Express Rx detection.

TX PLL Termination:

- 50: If there is no 50 Ohm termination resistor on PCB board
- 2K: If there is 50 Ohm termination resistor on PCB board

High speed I/O termination topology is shown in Figure 18-26.

Figure 18-26. High-Speed I/O Terminations



PCS Advanced Setup

This tab is used to access the advanced attributes of the transmit and receive PCS for all four channels. Polarity of each individual Tx and Rx channel can be individually selected. The operating mode (e.g. 8b10b) of the individual channels can be selected. In addition, word alignment values such as comma values, comma mask and comma align can be selected. This tab is also used for setting values for the Clock Tolerance Compensation block.

Figure 18-27. Configuration GUI - PCS Advanced Setup Tab

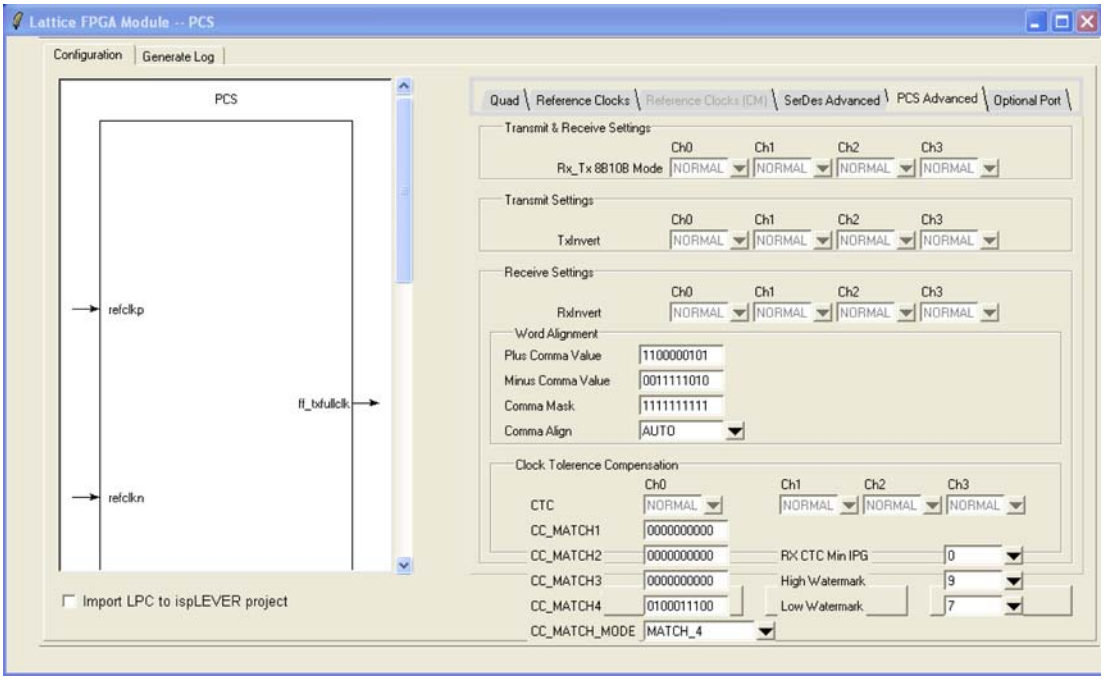


Table 18-12. SERDES/PCS GUI - PCS Advanced Setup Tab

GUI Text	Attribute Name	8-Bit SERDES Only	10-Bit SERDES Only	G8B10B	PCI Express	GIGE	SD-SDI	HD-HDI	Default
TX Invert	CHx_TX_SB	Normal, Invert							Normal
RX Invert	CHx_RX_SB								Normal
RX_TX 8b10b Mode	CHx_8B10B	Bypass		Normal			Bypass		Normal
Plus Comma Value	COMMA_A[4]	N/A	Note 1				N/A		1100000101
Minus Comma Value	COMMA_B[4]								00111111010
Comma Mask	COMMA_M								1111111111
Comma Align[2]	CHx_COMMA_ALIGN	Bypass	Auto, Dynamic, Bypass	Auto, Dynamic	Auto		Bypass		Auto
CTC[3]	CHx_CTC_BYP	Bypass[5]			Normal		Bypass		
CC_MATCH1	CC_MATCH1	N/A	Note 3				N/A		0000000000
CC_MATCH2	CC_MATCH2								0000000000
CC_MATCH3	CC_MATCH3								0100011100
CC_MATCH4	CC_MATCH4								0100011100
CC_MATCH_MODE	CC_MATCH_MODE		MATCH_3_4		MATCH_4				MATCH_4
RX CTC Min IFG	RX CTC Min IFG		0, 1, 2, 3						0
High Watermark	CCHMARK		0, 1, 2, ..., 14, 15						9
Low Watermark	CCLMARK								7

1. Refer to the Word Alignment section of this document for detailed information.
2. Refer to Table 18-6.
3. Refer to the Clock Tolerance Compensation section of this document for detailed information.
4. By definition, COMMA_A and COMM_B are one set of 8b10b encoded control character with positive and negative running disparities. For example, BC(K28.5) and FD(K29.7) cannot be used as the COMMA_A and COMM_B in a single design. The usage of COMMA_A and COMM_B in 8bit mode and 16bit mode is exactly the same in both modes. Users need to follow the rule in order to get the ls_sync. For example, 1 GbE needs K28.5+D5.6 or D16.2 as IDLE (word alignment and sync state machine).
5. GIGE or PCI Express modes may be used if CTC Normal mode is preferred. Note that the GIGE and PCI Express modes are tuned for their specific data rates.

Optional Setup

This tab allows is used to select the dynamic logic inversion and dynamic external link state machine capability per channel. In addition, users can enable the SCI, error reporting, PLL quarter clock, and loop-back capability.

Figure 18-28. Configuration GUI - Optional Setup Tab

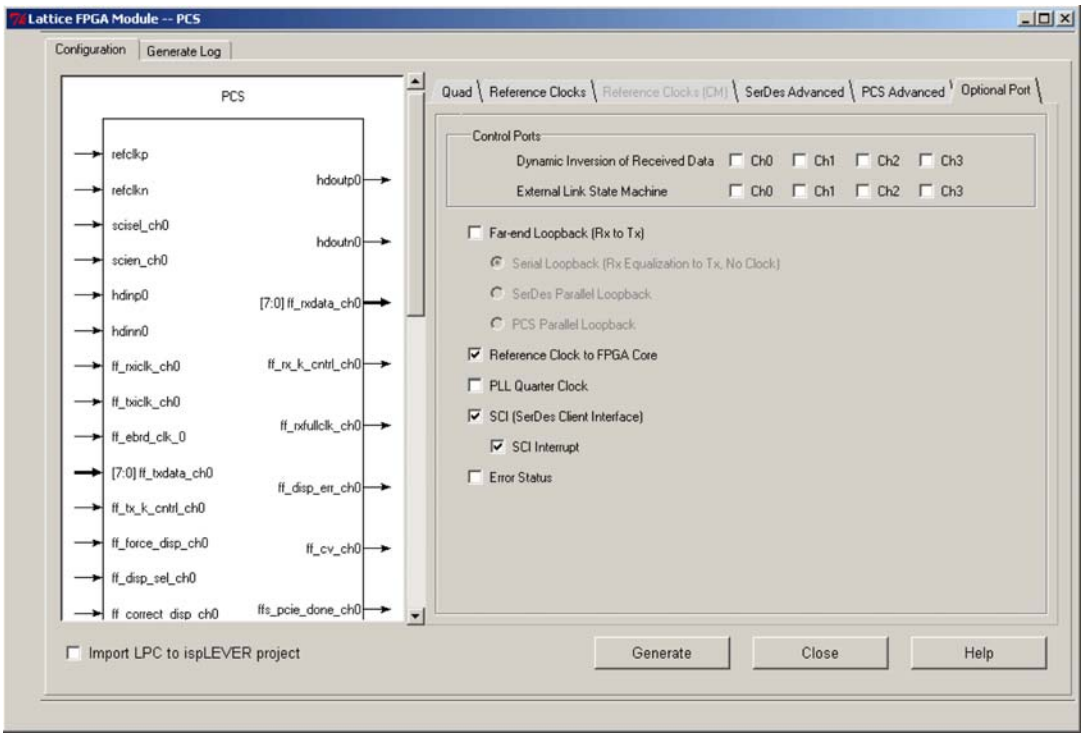


Table 18-13. Tab5. SERDES_PCS GUI Attributes (LatticeECP2M) - Optional Setup Tab

GUI Text	Attribute Names	Button/ Box Type	Range	Default Value
Dynamic Inversion of Receive Data		Check Box	TRUE, FALSE	FALSE
External Link State Machine[1]		Check Box	TRUE, FALSE	FALSE
Loopback (Rx to Tx)		Check Box	TRUE, FALSE	FALSE
Loopback Type	OS_SSLB OS_SPLBPORTS OS_PCSLBPORTS	Radio Box	Serial Loopback, SERDES Parallel Loopback, PCS Parallel Loopback	Serial Loopback
Reference Clock to FPGA Core	OS_REFCK2CORE	Check Box	TRUE, FALSE	FALSE
PLL Quarter Clock	OS_PLLQCLKPORTS	Check Box	TRUE, FALSE	FALSE
SCI		Check Box	TRUE, FALSE	FALSE
SCI Interrupt	OS_INT_ALL	Check Box	TRUE, FALSE	FALSE
Error Status		Check Box	TRUE, FALSE	FALSE

1. When Dynamic mode is selected in the Comma Align option, the External State Machine must be also selected.

Configuration File Description

IPexpress generates this file which contains attribute level map information. This file is input for simulation, synthesis and the ispLEVER bitgen program. It is strongly recommended that designers make changes in the IPexpress and then regenerate the configuration file. In some exceptional occasions, users can modify the Configuration File. The configuration file uses “txt” as the file type extension.

Below is an example of the configuration file.

```
# This file is used by the simulation model as well as the ispLEVER bitstream
# generation process to automatically initialize the PCSC quad to the mode
# selected in the IPexpress. This file is expected to be modified by the
# end user to adjust the PCSC quad to the final design requirements.

DEVICE_NAME "LFE2M35E"
PROTOCOL    "G8B10B"
CH0_MODE    "SINGLE"
CH1_MODE    "DISABLE"
CH2_MODE    "DISABLE"
CH3_MODE    "DISABLE"
PLL_SRC     "REFCLK"
DATARANGE     "HIGH"
CH0_CDR_SRC     "REFCLK"
CH0_DATA_WIDTH     "8"
CH0_REFCK_MULT     "10X"
#REFCLK_RATE     250.0
#FPGAINTCLK_RATE     250.0
CH0_TDRV_AMP     "0"
CH0_TX_PRE     "DISABLE"
CH0_RTERM_TX     "50"
CH0_RX_EQ     "DISABLE"
CH0_RTERM_RX     "50"
CH0_RX_DCC     "AC"
LOS_THRESHOLD     "0"
PLL_TERM     "50"
PLL_DCC     "AC"
PLL_LOL_SET     "0"
CH0_TX_SB     "NORMAL"
CH0_RX_SB     "NORMAL"
CH0_8B10B     "NORMAL"
COMMA_A     "1100000101"
COMMA_B     "0011111010"
COMMA_M     "1111111111"
CH0_COMMA_ALIGN     "AUTO"
CH0_CTC_BYP     "BYPASS"
CC_MATCH1     "0000000000"
CC_MATCH2     "0000000000"
CC_MATCH3     "0100011100"
CC_MATCH4     "0100011100"
CC_MATCH_MODE     "MATCH_4"
CC_MIN_IPG     "0"
CCHMARK     "4"
CCLMARK     "4"
OS_REFCK2CORE     "0"
OS_PLLQCLKPORTS     "0"
```

LatticeECP2M PCS in Gigabit Ethernet Mode

Gigabit Ethernet (1000BASE-X) Idle Insert

Idle pattern insertion is required for Clock Compensation and Auto Negotiation. Auto Negotiation is done in FPGA logic. This module automatically inserts /I2/ symbols into the receive data stream during auto-negotiation. While auto-negotiating, the link partner will continuously transmit /C1/ and /C2/ ordered sets. The clock-compensator will not delete these ordered sets as it is configured to only insert/delete /I2/ ordered sets. In order to prevent overruns and underruns in the clock-compensator, /I2/ ordered sets must be periodically inserted to provide insertion/deletion opportunities for the clock compensator.

While performing auto-negotiation, this module will insert a sequence of 8 /I2/ ordered sets (2 bytes each) every 2048 clock cycles. As this module is after the 8b10b decoder, this operation will not introduce any running disparity errors. These /I2/ ordered sets will not be passed on to the FPGA receive interface as the GMII interface is driven to IDLE by the Rx state machine during auto-negotiation. Once auto-negotiation is complete, /I2/ insertion is disabled to prevent any corruption of the received data.

Note that this state machine is active only during auto-negotiation. The auto-negotiation state machine and the GbE receive state machines are implemented in the soft logic. This state machine depends on the signal ff_xmit_ch[3:0] from the auto-negotiation state machine. This signal is provided on the TX data bus. Even though this signal is relatively static (especially after auto-negotiation) it is included in the TX data bus. It provides the signal rx_even_ch[3:0]. This is sent out on the receive data bus to the FPGA logic.

Gigabit Ethernet Idle Insert and ff_correct_disp_ch[3:0] Signal Usage

The ff_correct_disp_ch[3:0] signal is used on the transmit side of the QuadPCS to ensure that an inter-packet gap begins in the negative disparity state. Note that at the end of an Ethernet frame, the current disparity state of the transmitter can be either positive or negative, depending on the size and data content of the Ethernet frame. However, from the FPGA soft-logic side of the QuadPCS, the current disparity state of the QuadPCS transmitter is unknown. This is where the ff_correct_disp_ch[3:0] signal comes into play. If the ff_correct_disp_ch[3:0] signal is asserted for one clock cycle upon entering an interpacket gap, it will force the QuadPCS transmitter to insert an IDLE1 ordered-set into the transmit data stream if the current disparity is positive. However, if the current disparity is negative, then no change is made to the transmit data stream.

From the FPGA soft-logic side of the QuadPCS, the interpacket gap is typically characterized by the continuous transmission of the IDLE2 ordered set which is as follows:

ff_tx_k_cntrl_ch[3:0]=1, ff_txdata= 0xBC ff_tx_k_cntrl_ch[3:0]=0, ff_txdata=0x50.

Note that in the PCS channel, IDLE2s mean that current disparity is to be preserved. IDLE1s mean that the current disparity state should be flipped. Therefore, it is possible to ensure that the interpacket gap begins in a negative disparity state. If the disparity state before the interpacket gap is negative, then a continuous stream of IDLE2s are transmitted during the interpacket gap. If the disparity state before the interpacket gap is positive, then a single IDLE1 is transmitted followed by a continuous stream of IDLE2s.

In the FPGA soft-logic side of the QuadPCS, the interpacket gap is always driven with IDLE2s into the QuadPCS. The ff_correct_disp_ch[3:0] signal is asserted for one clock cycle, k_cntrl=0, data=0x50 when the interpacket gap first begins. If necessary, the QuadPCS will convert this IDLE2 into an IDLE1. For the remainder of the interpacket gap, IDLE2s should be driven into the QuadPCS and the ff_correct_disparity_chx signal should remain deasserted.

LatticeECP2M PCS in PCI Express Mode

An LatticeECP2M quad set to PCI Express Mode in IPexpress has additional ports at the FPGA interface to enable electrical functions required by the PCI Express specification such as receiver detection. Table 18-14 describes the PCI Express Mode specific ports.

Table 18-14. PCI Express Mode Specific Ports

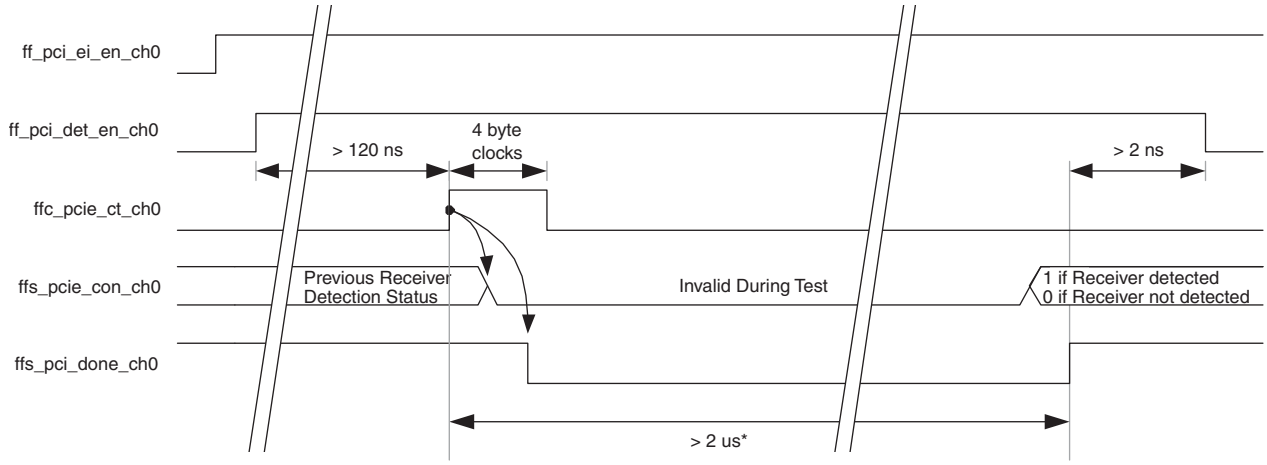
Signal	Direction	Class	Description
ffs_pcie_done_ch[3:0]	O	Channel	1 = Far-end receiver detection complete 0 = Far-end receiver detection incomplete
ffs_pcie_con_ch[3:0]	O	Channel	Result of far-end receiver detection. 1 = Far end receiver detected 0 = Far end receiver not detected.
ffc_pcie_det_en_ch[3:0]	I	Channel	FPGA logic (user logic) informs the SERDES block that it will be requesting for a PCI Express Receiver Detection operation. 1=Enable PCI Express Receiver Detect 0 = Normal operation
ffc_pcie_ct_ch[3:0]	I	Channel	1 = Request transmitter to do far-end receiver detection 0 = Normal data operation
ffc_ei_en_ch[3:0]	I	Channel	Control transmission of electrical idle by SERDES transmitter 1 = Force SERDES transmitter to output electrical idle 0 = Normal operation

Receiver Detection

Figure 18-29 shows a Receiver Detection sequence. A Receiver Detection test can be performed on each channel of a quad independently. Before starting a Receiver Detection test, the transmitter must be put into electrical idle by setting the **ff_pci_ei_en_ch** input high. The Receiver Detection test can begin 120 ns after tx_elec_idle is set high by driving the appropriate **ffc_pci_det_en_ch** high. This puts the corresponding SERDES Transmit buffer into receiver detect mode by setting the driver termination to high impedance and pulling both differential outputs to VCCOB through the high impedance driver termination.

Setting the SERDES Transmit buffer into receiver detect state takes up to 120 ns. After 120ns, the receiver detect test can be initiated by driving the channel's **ffc_pcie_ct_ch** input high for four byte (word) clock cycles. The corresponding channel's **ffc_pcie_done_ch** is then cleared asynchronously. After enough time for the receiver detect test to finish has elapsed (determined by the time constant on the Transmit side), the **ffs_pcie_done_ch** receiver detect status port will go high and the Receiver Detect status can be monitored at the **ffs_pcie_con_ch** port. If at that time the **ffs_pcie_con_ch** port is high, then a receiver has been detected on that channel. If, however, the **ffs_pcie_con_ch** port is low, then no receiver has been detected for that channel. Once the Receiver Detect test is complete, **ff_pci_ei_en_ch** can be deasserted.

Figure 18-29. PCI Express Mode Receiver Detection Sequence (Example for Channel 0)



PCI Express Beacon Support

This section highlights how the LatticeECP2M PCS can support Beacon Detection and Transmission. The PCI Express requirements for Beacon Detection are presented with the PCS support for Beacon Transmission and Beacon Detection.

- **Beacon Detection Requirements** (PCI Express Base Specification, Rev 1.0a, Chapter 4, Page 209-210)
  - Beacon is required for exit from L2 (P2) state.
  - Beacon is a DC balanced signal of periodic arbitrary data, which is required to contain some pulse widths >= 2ns (500Mhz) and < 16us (30Khz).
  - Maximum time between pulses should be < 16us.
  - DC balance must be restored within < 32us.
  - For pulse widths > 500ns, output beacon voltage level must be 6db down from VTX-DIFFp-p (800mV to 1200mV).
  - For pulse widths < 500ns, output beacon voltage level must be <= VTX-DIFFp-p and >= 3.5db down from VTX-DIFFp-p.
- **PCS Beacon Detection Support**
  - The signal loss threshold detection circuit senses if the specified voltage level exists at the receiver buffer. This is indicated by ffs_rlos_lo_ch(0-3) signal.
  - This setting can be used both for PCI Express Electrical Idle Detection and PCI Express Beacon Detection (when in power state P2).
  - The remote transmitting device can have a Beacon Output voltage of 6db down from VTX-DIFFpp (i.e 201mV). If this signal can be detected, it can be stated that Beacon is detected.
- **PCS Beacon Transmission Support**
  - Sending the K28.5 character (IDLE) (5 1's followed by 5 0's) provides a periodic pulse with of 2ns occurring every 2ns (1.0UI = 400ps, multiplied by 5 = 2ns). This meets the lower requirement. The output beacon voltage level can then be $V_{TX\text{-}DIFFp\text{-}p}$. This would be valid Beacon Transmission.

PCS Loopback Modes

The LatticeECP2M family of devices provides three loopback modes controlled by control signals at the PCS/FPGA interface for convenient testing of the external SERDES/board interface and the internal PCS/FPGA logic interface. Three loopback modes are provided to loop received data back onto the transmit data path. The loopback modes are useful for checking the high speed serial SERDES package pin connections as well as the embedded SERDES and/or PCS logic.

Serial Loopback Mode

Loops serial receive clock/data back onto the transmit buffer without passing through the CDR or de-serializer. This feature is intended for internal testing purpose but users can also use it.

Selecting the Serial Loopback option in the IPexpress GUI will set only the LB_CTL[3:0] to '0010' (refer to Table 18-77).

The TDRV_DAT_SEL[1:0] register bits should be also set to '11' via SCI to enable Serial Loopback mode. The LB_CTL[3:0] register bits are also accessible via SCI.

SERDES Parallel Loopback Mode

Loops parallel receive data back onto the transmit data path without passing through the PCS logic. SERDES parallel loopback can be selected for each channel individually by setting the appropriate ffc_sb_pfifo_lp_ch(0-3) to a "1".

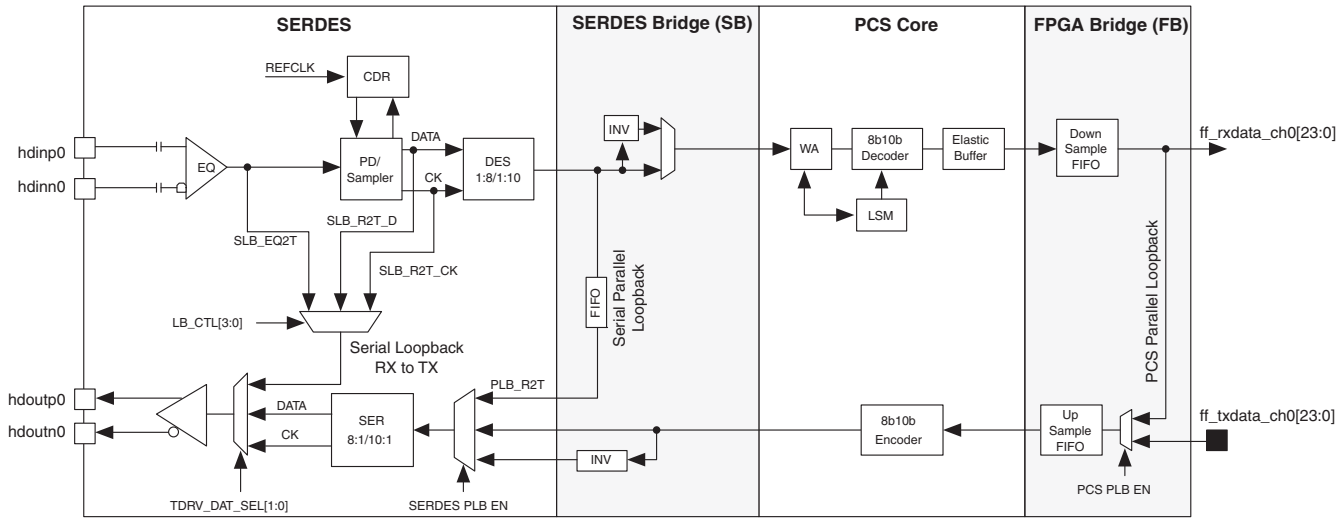
PCS Parallel Loopback Mode

Loops parallel receive data back onto the transmit data path without passing across the PCS/FPGA interface. Input serial data at the SERDES hdin package pins passes through the SERDES receive logic where it is converted to

parallel data, passes through the entire PCS receive logic path, is looped back through the entire PCS transmit logic path, is reconverted back to serial data by the SERDES transmitter and sent out onto the hdout SERDES package pins. PCS Parallel loopback can be selected for each channel individually by setting the appropriate ffc_fb_loopback_ch(0-3) port to a “1”.

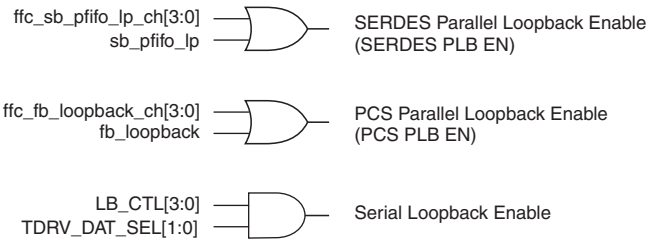
Figure 18-30 illustrates the three loopback modes for a single channel.

Figure 18-30. Three Loopback Modes



The two parallel loopback modes described above provide not only control signals from the FPGA core but also Control Register bits. When the control register bits are not set for the loopback mode, the data path can switch between Loopback mode and Normal Data Flow by control signals from the FPGA core. Figure 18-31 describes this logic. Refer to Tables 18-77 and 18-78 for detailed register settings.

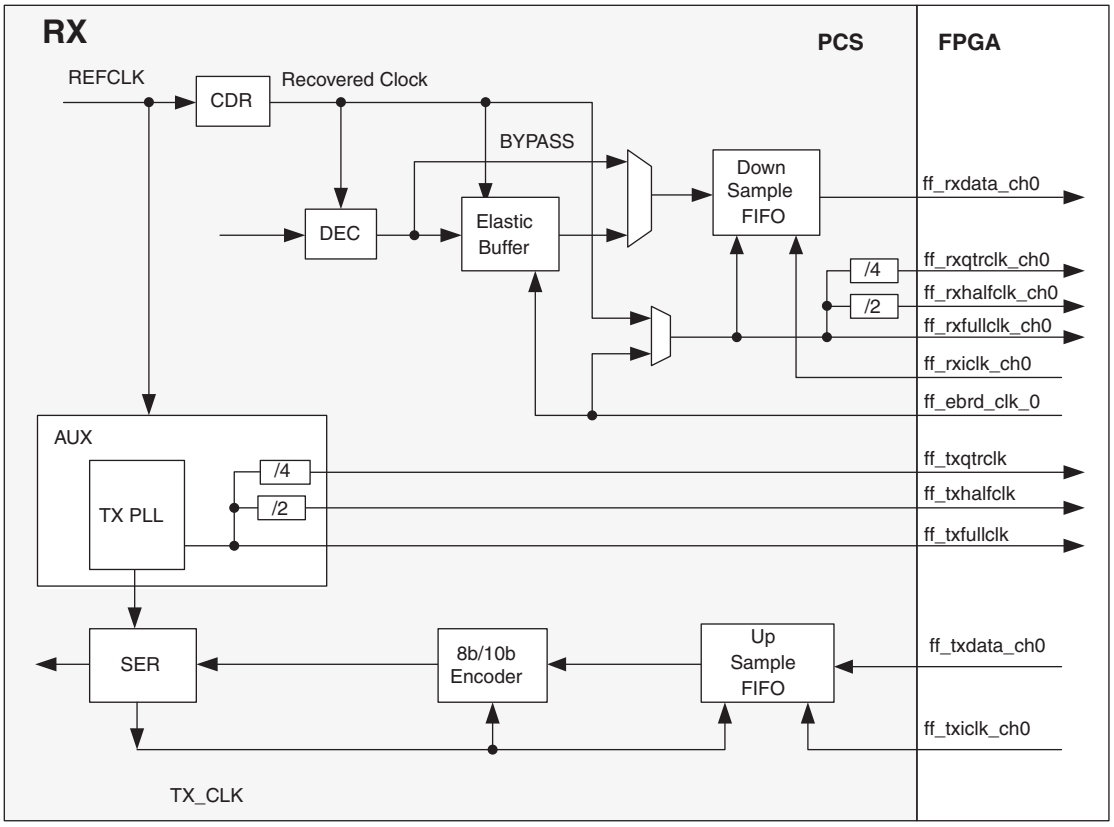
Figure 18-31. Loopback Enable Signals



FPGA Interface Clocks Usage

Figure 18-32 shows a conceptual diagram of the later stage of the PCS Core and the FPGA Bridge and the major clocks that cross the boundary between PCS and the FPGA.

Figure 18-32. Conceptual PCS/FPGA Clock Interface Diagram



In the above diagram and in the subsequent clock diagrams in this section, note that suffix “i” indicates the index [0:3] i.e., one for each channel.

None of the clock muxes have logic to guard against clock slicing or glitching. It is a requirement that if any of the selectors to the clock muxes are changed by writes to register bits, the software agent will reset the logic clocked by that muxed clock.

The PCS outputs 15 clocks. There are three transmit clocks (per quad) and 12 receive clocks (per channel). The three transmit clocks provide full rate, half rate and quarter rate clocks and are all generated from the Tx PLL. The full rate and half rate transmit clocks need to be send directly via dedicated route to the Center Clock Mux in the FPGA. There are also three clocks (full, half and quarter rates) per receive channel. All 15 clocks can be used as local (secondary) or global (primary) clocks for the FPGA logic as required. Divided-by-two clocks are used when the gearing is in 2:1 mode (the gearing is selectable only on a quad basis).

The transmit clock is used on the write port of the Up Sample FIFO (or Phase Shift FIFO, depending on the case). One of the two receive clocks is connected to the read clock of the Down Sample FIFO. The other clocks the read port of the Elastic Buffer FIFO and potentially (depending on the case) the write port of the Down Sample FIFO.

Based on the whether the Elastic Buffer and the Up Sample FIFO are bypassed or not and whether we are in 8b10b mode or 16b20b mode, four use cases are possible. The active paths are highlighted with weighted lines. It is also indicated how many and what kind of clock trees are required. There are some modes that would more commonly be preferred by the user.

This section describes the operation for the six different cases that are supported. The six cases are outlined in Table 18-15.

Table 18-15. Six Interface Cases Between SERDES/PCS Quad and FPGA Core

Interface	Data Width	Rx CTC FIFO	Rx Phase-Shift/Down-Sample FIFO	Tx Phase-Shift/ Up-Sample FIFO
Case I-a[1]	8b10b	Yes	Yes	Yes
Case I-b[1]	8b10b	Bypass	Yes	Yes
Case II-a[1]	16b20b	Yes	Yes	Yes
Case II-b[1]	16b20b	Bypass	Yes	Yes

1. In all cases in which the Tx phase-shift (up-sample) FIFO is used, it is never bypassed. Deep inside the SERDES/PCS block, the datapath width is 8/10 bits wide and the byte clock runs at 1/10 of the SERDES line rate. For example, if the SERDES line rate is 3.125Gbps, then the byte clock is 312.5MHz.

2-to-1 Gearing

For guaranteed performance of the FPGA global clock tree, it is recommended to use a 16/20 bit wide interface for SERDES line rates greater than 2.5Gbps. In this interface, the FPGA interface clocks are running at half the byte clock frequency.

Even though the 16/20 bit wide interface running at half the byte clock frequency can be used with all SERDES line rates, the 8/10 bit wide interface is preferred when the SREDES line rate is low enough to allow it (2.5Gbps and below) because this results in the most efficient implementation of IP in the FPGA core.

The decision matrix for the six interface cases is explained in Table 18-16.

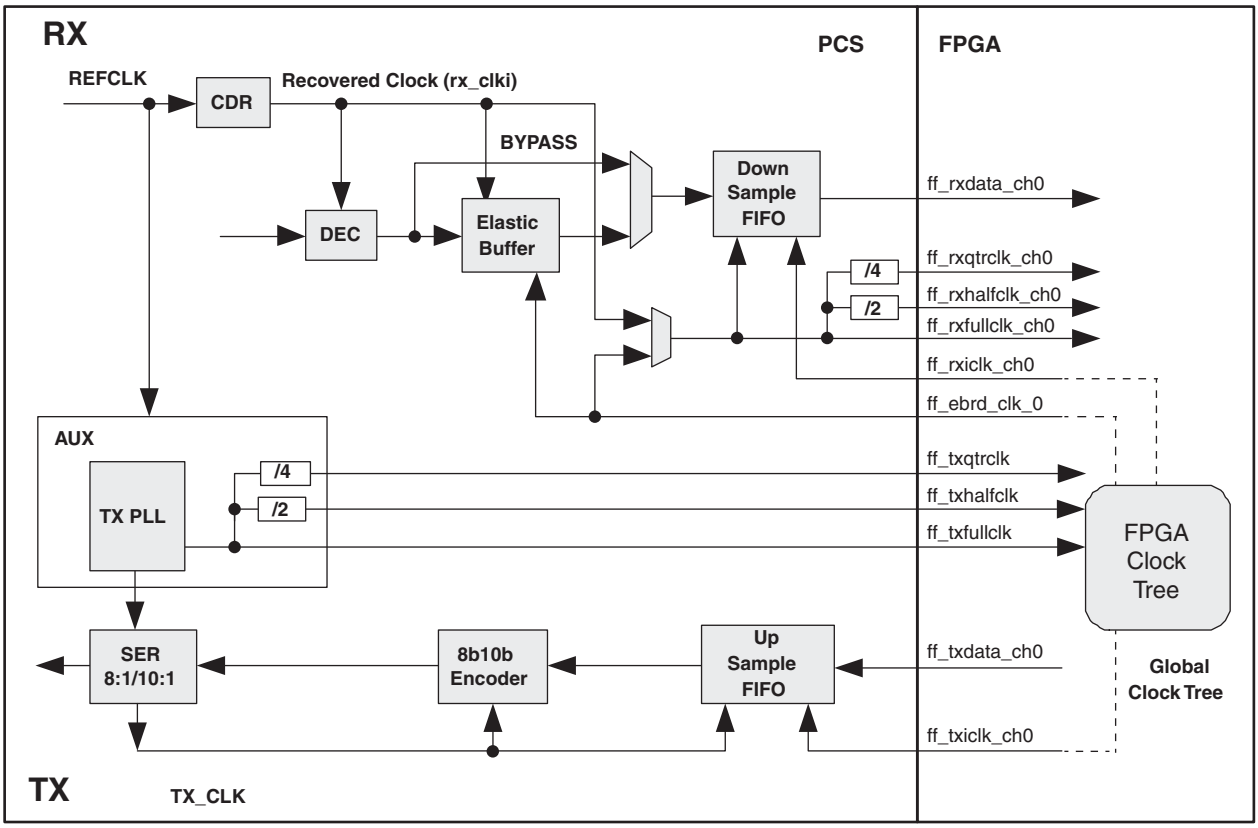
Table 18-16. Decision Matrix for Six Interface Cases

SERDES Line Rate	Datapath Width	MCA Required?	CTC Required?	Interface Case
2.5 Gbps and below	8/10 bit (1:1 gearing)	No, single-channel link	Yes	Case I_a[1]
			No	Case I_b[2]
		Yes, multi-channel link	Must bypass, not available	Case I_b[3]
3.2 Gbps and below	16/20 bit (2:1 gearing)	No, single-channel link	Yes	Case II_a[4]
			No	Case II_b[5]
		Yes, multi-channel link	Must bypass, not available	Case II_b[6]

1. This case is intended for SINGLE-channel links at line rates of 2.5Gbps and lower (8/10 bit wide interface) that require clock tolerance compensation in the quad. CTC is required when both ends of the link have separate reference clock sources that are within +/- 300ppm of each other. Case I_a is used if the IP in the core requires the Rx phase-shift FIFO. Case I_b is used if the IP does not require this FIFO.
2. This case is intended for SINGLE-channel links at line rates of 2.5Gbps and lower (8/10 bit wide interface) that do NOT require clock tolerance compensation in the quad. CTC is not required when both ends of the link are connected to the same reference clock source. This is often the case for chip-to-chip links on the same circuit board. There is exactly 0ppm difference between the reference clocks and so CTC is not required and can be bypassed. CTC is also not required in the quad when this function is performed by the IP in the core.
3. This case is intended for MULTI-channel links at line rates of 2.5Gbps and lower (8/10 bit wide interface). Multi-channel alignment MUST be done by the IP in the core, there is no provision to do MCA in the quad. Since MCA must be done prior to CTC, the CTC FIFO in the quad MUST be bypassed when MCA is required and so both MCA and CTC (if required) are done by the IP in the core.
4. This case is intended for SINGLE-channel links at line rates of 3.2Gbps and lower that require a 2:1 gearbox between the quad and the FPGA core (16/20 bit wide interface). Clock tolerance compensation is included in the quad. CTC is required when both ends of the link have separate reference clock sources that are within +/- 300ppm of each other.
5. This case is intended for SINGLE-channel links at line rates of 3.2Gbps and lower that require a 2:1 gearbox between the quad and the FPGA core (16/20 bit wide interface). Clock tolerance compensation is NOT included in the quad. CTC is not required when both ends of the link are connected to the same reference clock source. This is often the case for chip-to-chip links on the same circuit board. There is exactly 0ppm difference between the reference clocks and so CTC is not required and can be bypassed. CTC is also not required in the quad when this function is performed by the IP in the core.
6. This case is intended for MULTI-channel links at line rates of 3.2Gbps and lower that require a 2:1 gearbox between the quad and the FPGA core (16/20 bit wide interface). Multi-channel alignment MUST be done by the IP in the core, there is no provision to do MCA in the quad. Since MCA must be done prior to CTC, the CTC FIFO in the quad MUST be bypassed when MCA is required and so both MCA and CTC (if required) are done by the IP in the core.

Case I_a: 8/10bit, EB and DS FIFOs NOT Bypassed

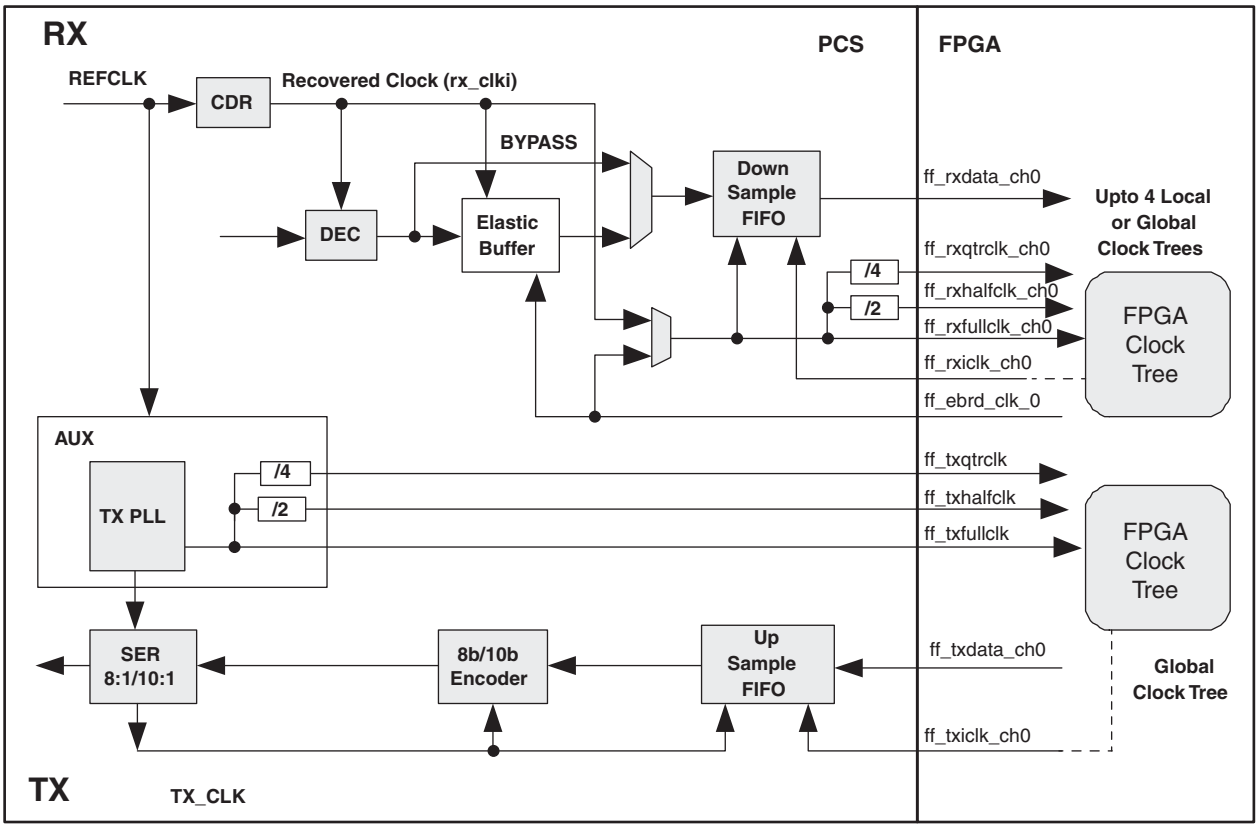
Figure 18-33. 8b10b, EB and DS FIFOs NOT Bypassed



1. The Up Sample FIFO is acting as a Phase Shift FIFO only in this case.
2. The Down Sample FIFO is acting as a Phase Shift FIFO only in this case.
3. The quad level full rate clock from the Tx PLL (ff_tx_f_clk) has direct access to the FPGA Center Clock Mux. This is a relatively higher performance path. A Global Clock Tree out of the Center Clock Mux is used to clock the user's interface logic in the FPGA. Some leaf nodes of the clock tree are connected to the FPGA Transmit Input clock (ff_txi_clki), the Elastic Buffer FIFO Read Clock per Channel (ff_ebrd_clki) via CIB Clk input and the FPGA Receive Input clock (ff_rxi_clki). This case is probably the most common single channel use case.

Case I_b: 8/10bit, EB FIFO Bypassed

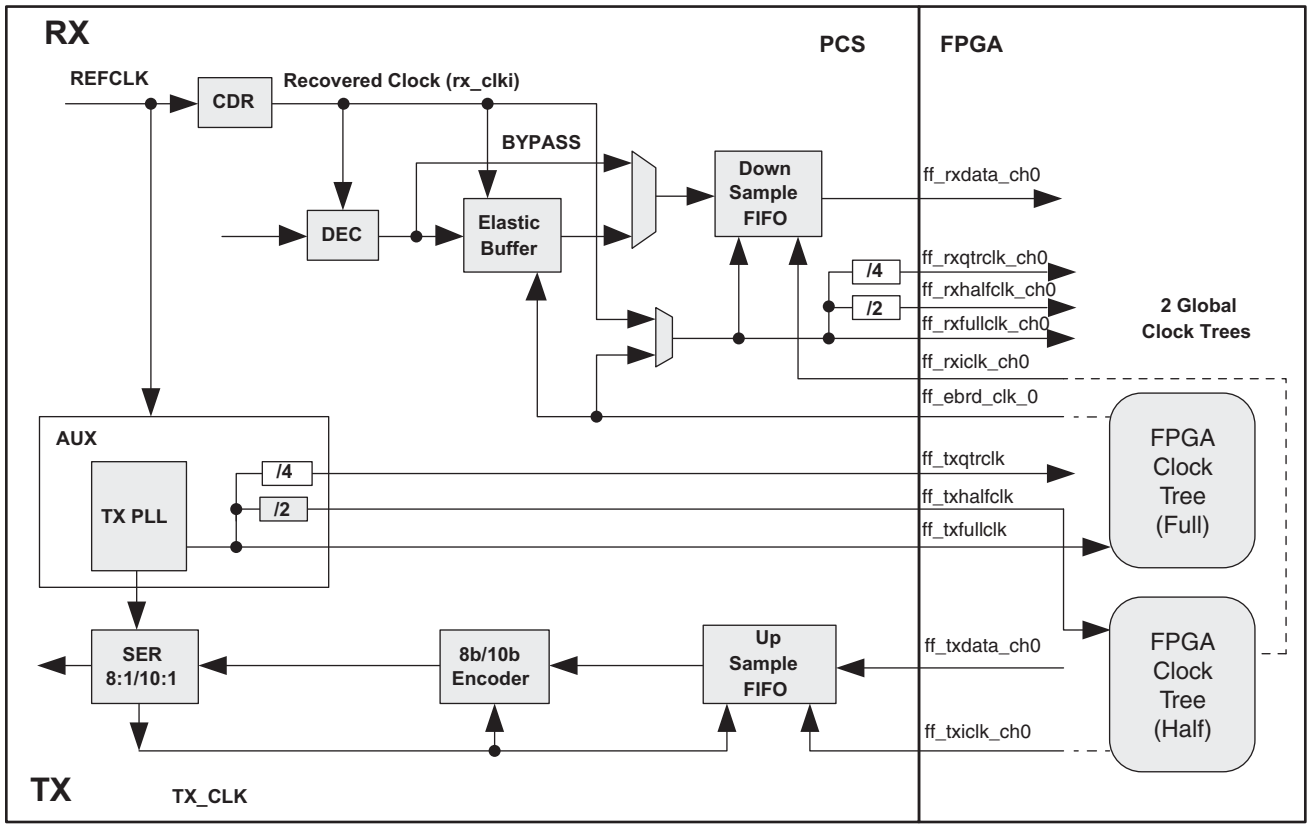
Figure 18-34. 8b10b, EB FIFO Bypassed



1. The Up Sample FIFO is acting as a Phase Shift FIFO only in this case.
2. The Down Sample FIFO is acting as a Phase Shift FIFO only in this case.
3. The Tx FPGA Channel input clock is clocked similarly as in the previous case using a clock tree driven by a direct connection of the full rate transmit FPGA output clock to the FPGA center clock mux Once the Elastic Buffer is bypassed, the recovered clock needs to control the write port of the Down Sample FIFO. The recovered clock of each channel may need to drive a separate local or global clock tree (i.e., up to 4 local or global clock trees per quad). The clock tree will then drive the FPGA receive clock input to control the read port of the Down Sample FIFO. The reason for bypassing the Elastic Buffer FIFO in this case is most likely for doing multi-channel alignment (MCA) in the FPGA core. It implies that CTC using an elastic buffer will be done in the FPGA core. The CTC FIFOs can be written by either the recovered clocks or by a master recovered clock. The read of the CTC FIFO will be done using the Tx clock via the Tx clock tree.

Case II_a: 16/20bit, EB and DS FIFOs NOT Bypassed

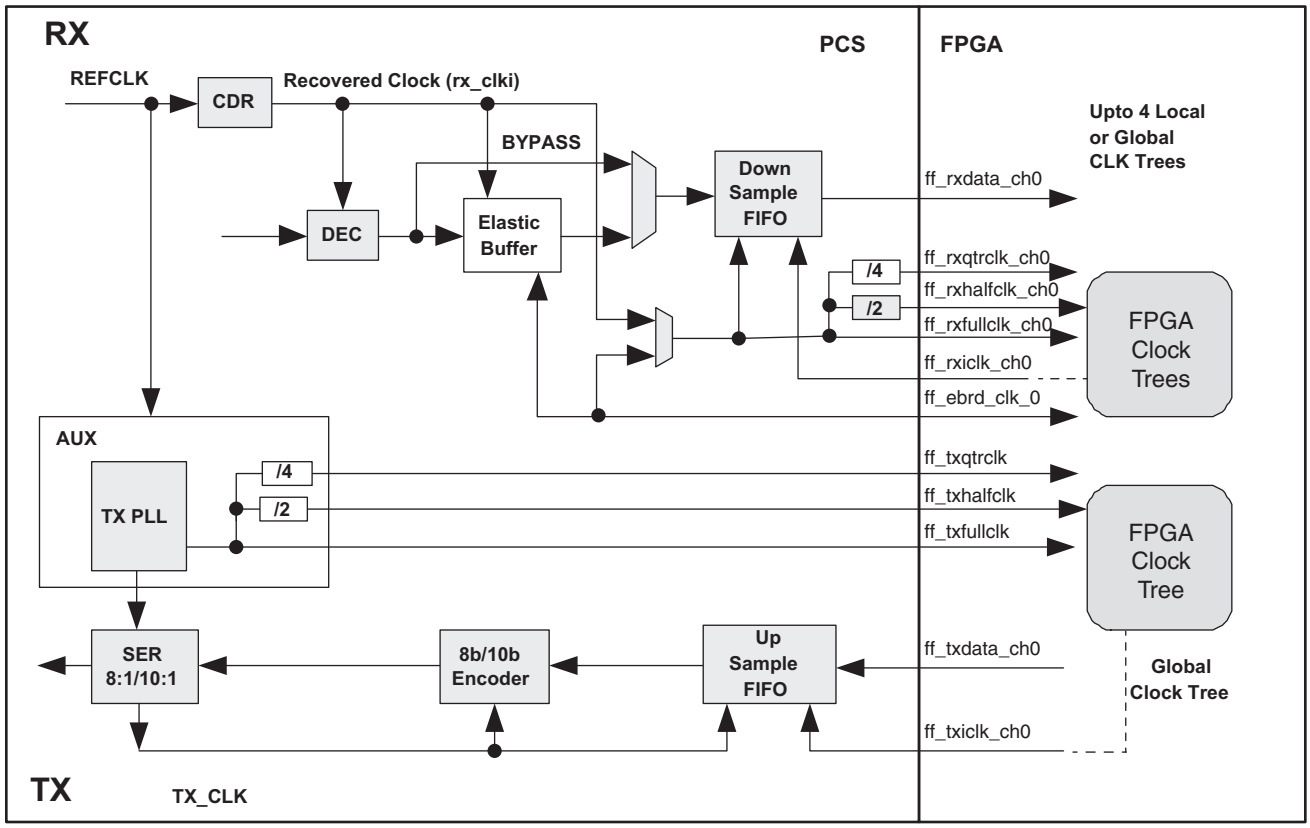
Figure 18-35. 16/20bit, EB and DS FIFOs NOT Bypassed



1. The Up Sample FIFO is acting both as a Phase Shift FIFO and Up Sample FIFO in this case.
2. The Down Sample FIFO is acting both as a Phase Shift FIFO and Down Sample FIFO in this case.
3. This is a very common single channel use case when the FPGA is unable to keep up with full byte frequency. Two clock trees are required. These clock trees are driven by direct access of transmit full rate clock and transmit half rate clock to the FPGA clock center mux. The full rate clock tree drives the Elastic Buffer read port and the Down Sample FIFO write port. The half rate clock tree drives the Down Sample FIFO and the FPGA logic.

Case II_b: 16/20bit, DS NOT Bypassed

Figure 18-36. 16/20 Bit, DS FIFO NOT Bypassed



1. The Up Sample FIFO is acting both as a Phase Shift FIFO and Up Sample FIFO in this case.
2. The Down Sample FIFO is acting both as a Phase Shift FIFO and Down Sample FIFO in this case.
3. This is a very common multi-channel alignment (MCA) use case when the FPGA is unable to keep up with full byte frequency. The receive clock trees (up to 4) can be local or global. They are running half rate clock. The transmit clock tree is driven by direct access of transmit half rate clock to the FPGA clock center mux. In this case, in FPGA logic after the MCA is done, a CTC will be required. The Tx clock tree will clock the read port of the CTC and the master channel receive clock the write port of the CTC. It is important to note that both the MCA and CTC need to be done across 16/20bits in this use case.

SERDES/PCS Block Latency

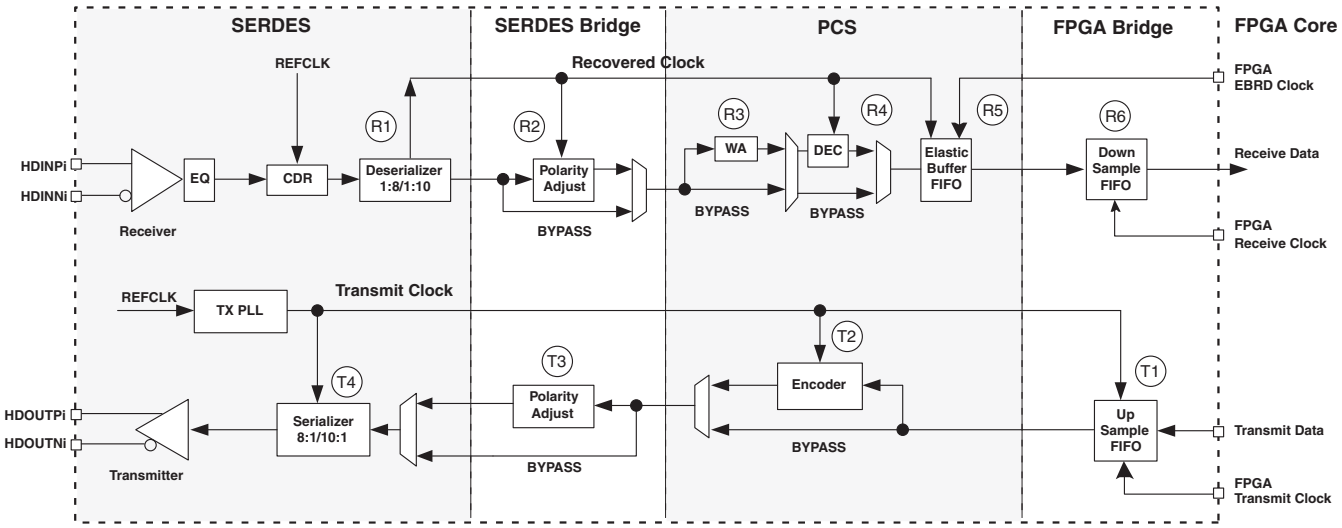
Table 18-17 describes the latency of each functional block in the transmitter and receiver. Latency is given in parallel clock cycles. Figure 18-37 shows the location of each block.

Table 18-17. SERDES/PCS Latency Breakdown (Parallel Clock Cycle)[1]

Item	Description	Min.	Average	Max.	Fixed	Bypass	Units
Transmit Data Latency							
T1	FPGA Bridge Transmit[2]	1	3	5		1	word clk
T2	8b10b Encoder	—	—	—	2	1	word clk
T3	SERDES Bridge Transmit	—	—	—	2	1	word clk
T4[3]	Serializer: 8-bit mode	—	—	—	15 + Δ1	—	UI + ps
	Serializer: 10-bit mode	—	—	—	18 + Δ1	—	UI + ps
Receive Data Latency							
R1[3]	Deserializer: 8-bit mode	—	—	—	10 + Δ2	—	UI + ps
	Deserializer: 10-bit mode	—	—	—	12 + Δ2	—	UI + ps
R2	SERDES Bridge Receive	—	—	—	2	1	word clk
R3	Word Alignment	3.1	—	4	—	0	word clk
R4	8b10b Decoder	—	—	—	1	1	word clk
R5	Clock Tolerance Compensation	7	15	23		1	word clk
R6	FPGA Bridge Receive[2]	1	3	5		1	word clk

1. PCS internal Parallel Clock. This clock rate is same as the rxfullclk in Table 18-5.
2. FPGA Bridge latency varies by UP/DOWN Sample FIFO read/write. These numbers were presented for 8bit/10bit interface. The depth of Down Sample/Up Sample FIFO is 4. The earliest read can be done after write clock cycle (1 clock) in Down Sample FIFO. The latest read will be done after the FIFO is full (4 + 1 = 5). For 16b/20b interface, the numbers become doubled. Min = 2, Max = 10. This latency depends on the internal FIFO flag operation.
3. Δ1 = -245ps, Δ2 = 700ps

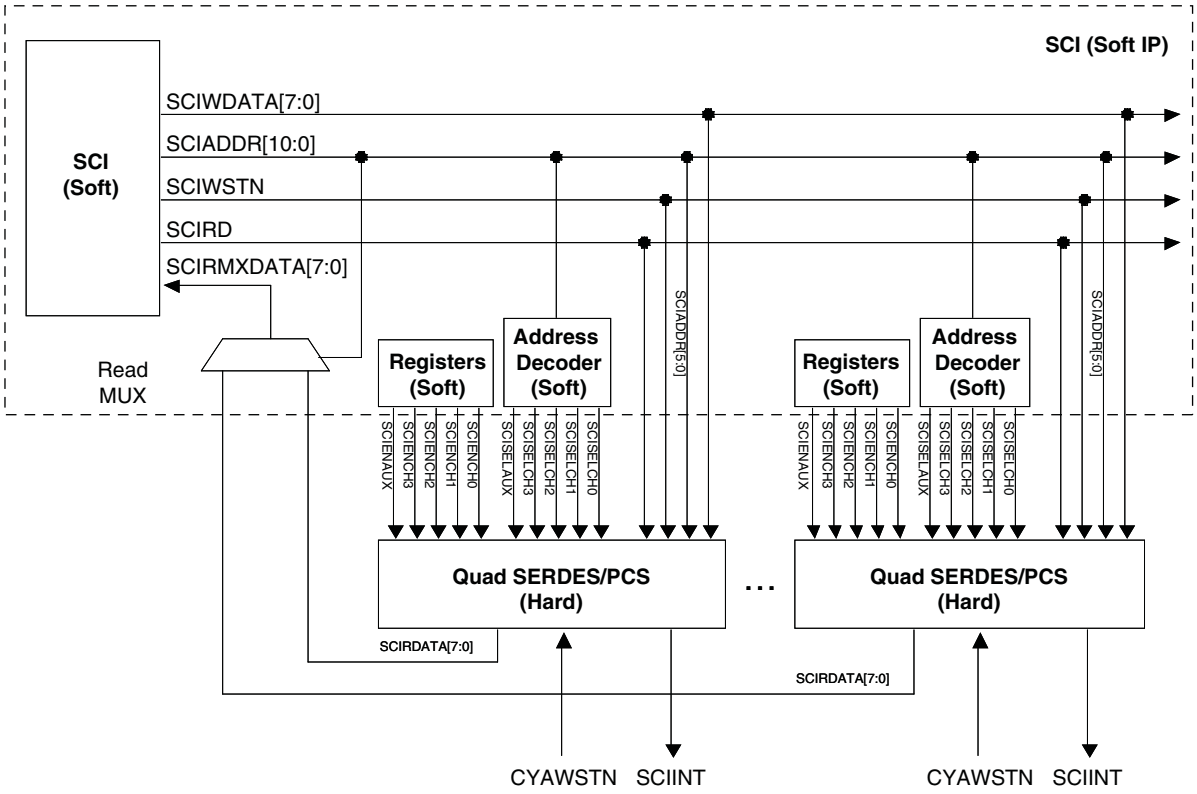
Figure 18-37. Transmitter and Receiver Block Diagram



SERDES Client Interface (SCI)

The SERDES Client Interface (SCI) consists of both soft IP that resides in the FPGA core and permanent logic that is included in the SERDES/PCS quad. The SCI allows the SERDES/PCS quad to be controlled by registers as opposed to the configuration memory cells. It is a simple register configuration interface. It shows all the major signals required. The block diagram of the soft IP part of the SCI that resides in the FPGA core is shown in Figure 18-38.

Figure 18-38. SCI Interface Block Diagram



The soft IP that resides in the FPGA core should be developed by users per their interface scheme. Contact the Lattice Technical Support Group for example code.

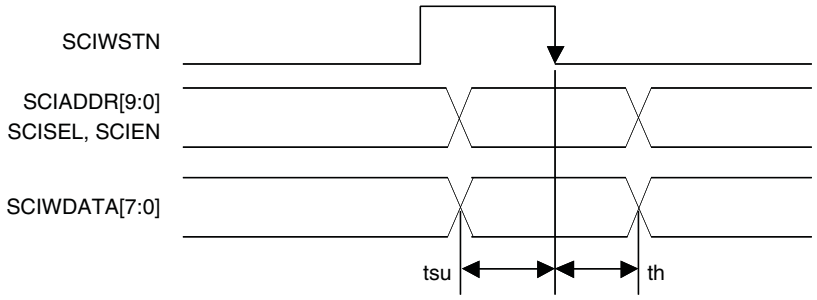
The SCIADDR bus is six bits wide within the block. The bus width at the block boundary is 11 bits. The upper five bits are used for quad block selection and channel selection. Table 18-18 shows the SCI address map for the SERDES quad.

Table 18-18. SCI Address Map for Up to Four SERDES/PCS Quads

Address Bits	Description
SCIADDR[5:0]	Register address bits 000000 = select register 0 000001 = select register 1 … 111110 = select register 62 111111 = select register 63
SCIADDR[8:6]	Channel address bits 000 = select channel 0 001 = select channel 1 010 = select channel 2 011 = select channel 3 100 = select aux channel 101 = unused 110 = unused 111 = unused
SCIADDR[10:9]	Quad address bits 00 = select quad 0 01 = select quad 1 10 = select quad 2 11 = select quad 3

Read and write operations through this interface are asynchronous. In the WRITE cycle the Write data and Write address need to be setup and held in relation to the falling edge of the SCIWSTN. In the READ cycle the timing has to be in relation with the SCIRD pulse. Figures 18-39 and 18-40 shows the WRITE and READ cycles respectively.

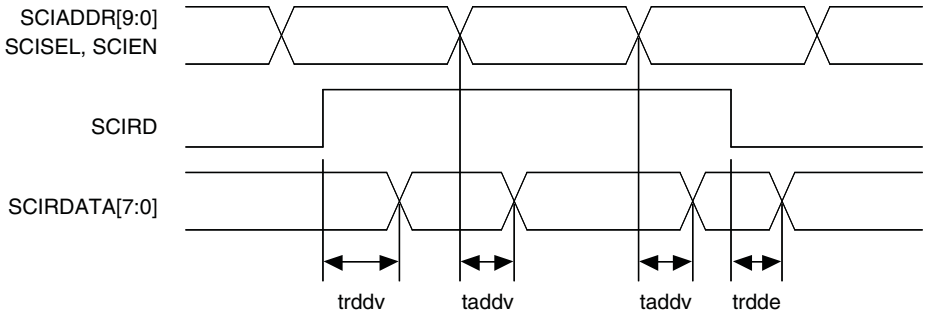
Figure 18-39. SCI WRITE Cycle, Critical Timing



Notes :
1) tsu is the setup time for address and write data prior to the falling edge of the write strobe.
2) th is the hold time for address and write data after the falling edge of the write strobe.

Note: To avoid accidental writing to control registers, registers should be used at the SCI input ports to drive them low at power-up reset.

Figure 18-40. SCI READ Cycle, Critical Timing



Notes:
1) trddv is the time from assertion of the read pulse until read data is valid.
2) taddv is the time from address change until data is valid while read pulse is asserted.
3) trdde is the hold time of the read data after the deassertion of the read pulse.

Table 18-19. Timing Parameters

Parameter	Typical Value	Units
tsu, trddv, taddv	1.127	ns
th, trdde	0.805	ns

The SCI interface is as simple as memory read/write. Here is an example of the pseudo code:

Write

```
Cycle 1 :  Set sciaddr[5:0], sciwdata[7:0], scien* = 1'b1 scisel* = 1'b1
Cycle 2 :  Set sciwstn from 0 => 1
Cycle 3 :  Set sciwstn from 1 => 0, scien* = 1'b0, scisel* = 1'b0
```

Read

```
Cycle 1 :  Set sciaddr[5:0], scisel* = 1'b1
Cycle 2 :  Set scird from 0 => 1
Cycle 3 :  Obtain reading data from scirdata[7:0]
Cycle 4 :  Set scird from 1 => 0
```

Interrupts and Status

The status bit may be read via the SCI, which is a byte wide and thus reads the status of eight interrupt status signals at a time. The SCIINT signal goes high to indicate that an interrupt event has occurred. The user is then required to read the QIF status register that indicates whether the interrupt came from the quad or one of the channels. This register is NOT cleared-on-read. It is cleared when all interrupt sources from the quad or channel is cleared.

Once the aggregated source of interrupt is determined, the user can read the registers in the associated quad or channel to determine the actual source of the interrupt. Tables 18-20 and 18-21 list all the sources of interrupt.

Table 18-20. Quad Interrupt Sources

Quad SCI_INT Source	Description	Register Name
int_quad_out	Quad Interrupt. If there is an interrupt event anywhere in the quad this register bit will be active. This register bit gets cleared when all interrupt events have been cleared.	PCS Quad Status Register 1
int_cha_out[0:3]	Channel Interrupt. If there is an interrupt event anywhere in the respective channel this register bit will be active. These register bits are cleared when all interrupt sources in the respective channel have been cleared.	PCS Quad Status Register 1
ls_sync_statusn_[0:3]_int ls_sync_status_[0:3]_int	Link Status Low (out of sync) channel interrupt Link Status High (in sync) channel interrupt	PCS Quad Status Register 3
~PLOL, PLOL	Interrupt generated on ~PLOL and PLOL - PLL Loss of Lock	SERDES Quad Status Register 2

Table 18-21. Channel Interrupt Sources

Quad SCI_INT Source	Description	Register Name
fb_tx_fifo_error_int fb_rx_fifo_error_int cc_overrun_int cc_underrun_int	FPGA Bridge Up Sample Tx FIFO Error Interrupt FPGA Bridge Down Sample Rx FIFO Error Interrupt CTC (Elastic Buffer) Overrun and Underrun Interrupts	PCS Channel General Interrupt Status Register 4
pci_det_done_int rlos_lo_int ~rlos_lo_int rlos_hi_int ~rlos_hi_int rlol_int ~rlol_int	Interrupt generated for pci_det_done Interrupt generated for rlos_lo Interrupt generated for ~rlos_lo Interrupt generated for rlos_hi Interrupt generated for ~rlos_hi Interrupt generated for rlol Interrupt generated for ~rlol	SERDES Channel Interrupt Status Register 5

SERDES Client Interface Application Example

Lattice Semiconductor's ORCAstra FPGA configuration software is a PC-based Graphical User Interface (GUI) that allows users to configure the operational mode of a Lattice FPGA by programming control bits in the FPGA registers. SERDES/PCS status information is displayed on-screen in real time, and any configuration can be saved to control registers for additional testing. Use of the GUI does not interfere with the programming of the FPGA core portion. More information and downloadable files for ORCAstra can be found on the Lattice Semiconductor website at www.latticesemi.com/products/designsoftware/orcastra.cfm.

For example:

Users can switch the reference clock multiplier mode between 10x mode and 20x mode by changing refck_mode[1](D7) and refck_mode[0](D6) bits in the Quad PCS Control Register, ser_ctl_3_qd_13. The SCI address of the register is 13(H) and ORCAstra Address = 113 (H). The Quad Reset (QD_18[5]) must be toggled to achieve this transition.

The read/write operation can be achieved in two different ways.

1. In the main window, users can read and write the content of control register read status registers.
2. In the sub-windows, users can select pull-down menu options, or check/uncheck ON/OFF options.

Table 18-22. SCI Address Map

Address Bit	Description
SCIADDR[5:0]	Register address bits 000000 = register 0 000001 = register 1 ..... 111110 = register 62 111111 = register 63
SCIADDR[8:6]	Channel address bits 000 = channel 0 001 = channel 1 010 = channel 2 011 = channel 3 100 = Aux channel
SCIADDR[10:9]	Quad address bits 00 = quad 0 01 = quad 1 10 = quad 2 11 = quad 3

Dynamic Configuration of SERDES/PCS Quad

The SERDES/PCS quad can be controlled either by the configuration memory cells or by registers that are accessed through the optional “SERDES Client Interface” (SCI). The SCI consists of both a soft IP that resides in the FPGA core and permanent logic that is included in the SERDES/PCS quad. The SCI is only available if the soft IP is present in the FPGA core.

After configuration is complete, those configuration memory cells that have associated registers are automatically copied into the registers. Subsequent changes to the contents of the registers will not affect the value stored in the configuration memory cells but will of course change the operation of the SERDES/PCS quad.

When controlled by the configuration memory cells, it is a requirement that the SERDES/PCS quads must reach a functional state after configuration is complete, without further intervention from the user. This means that any special reset sequences that are required to initialize the SERDES/PCS quad must be handled automatically by the hardware. In other words, use of the SCI is optional, the SERDES/PCS quad must NOT assume that the soft IP is present in the FPGA core.

SERDES Debug Capabilities

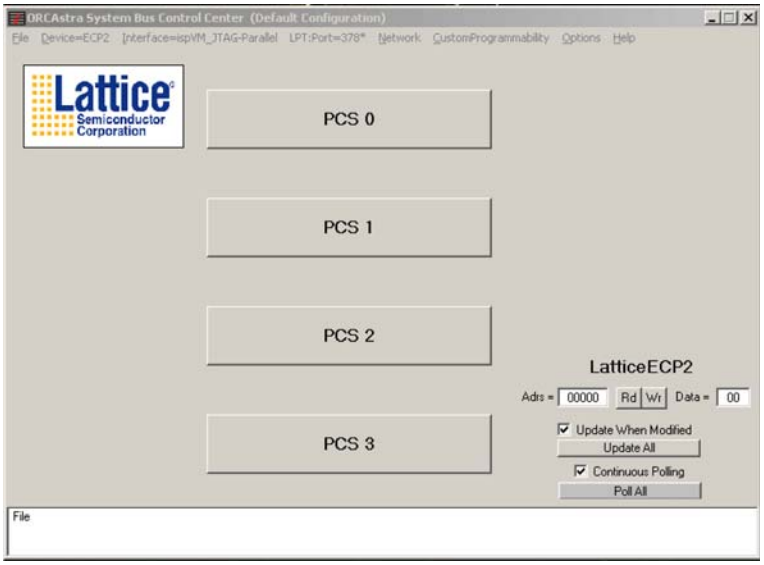
Lattice has tools to help in the debug of the SERDES/PCS operation in LatticeECP2M devices.

Lattice ORCAstra software is a PC-based graphical user interface for configuring the operational mode of a LatticeECP2M device by programming control bits in the on-chip registers. This helps you quickly explore configuration options without going through a lengthy re-compile process or making changes to your board. Configurations created in the GUI can be saved to memory and re-loaded for later use. To use ORCAstra, SCI IP must be instantiated in the user logic.

A macro capability is also available to support script-based configuration and testing. The GUI can also be used to display system status information in real time. Use of the ORCAstra software does not interfere with the programming of the FPGA.

Figure 18-41 shows the ORCAstra GUI top-level window. Users can read and write in this window without going through the subwindows for each PCS channel by read and write data at Adrs cell. When invoked, ORCAstra will automatically recognize the device type. Or, device types can be selected under the device pull-down menu.

Figure 18-41. ORCAstra Top-Level Screen Shot

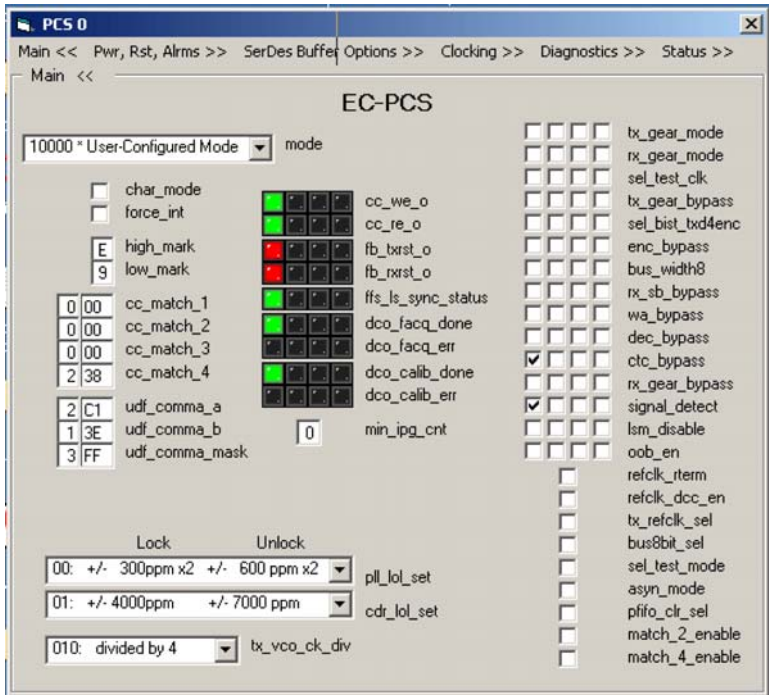


By default, the Data Box shown in Figure 18-41 follows Little Endian byte order (i.e., the most significant bit is placed on the right). Users can change to Big Endian order by selecting **Display Data reversed in Data Box** under the **Options** tab.

Double clicking on the **PCS0 (Quad 0)** button will open the main window as shown in Figure 18-42.

These standard Windows menus control the selection of the device and interface. They also support various configuration options, including setting up and saving configurations with stored files.

Figure 18-42. ORCAstra Main Window

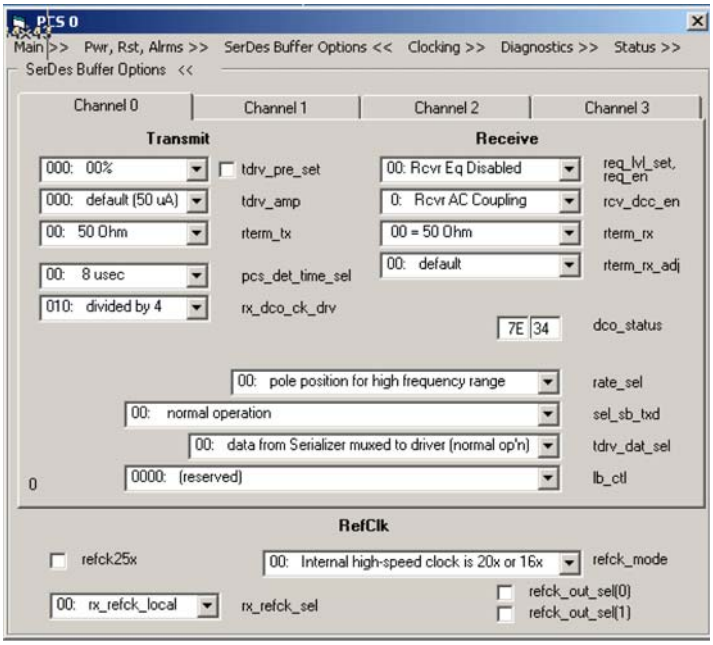


Control Boxes and Buttons, Status Boxes and the Text Window

Moving the cursor over a control box and clicking the left mouse button sets the control bits. Both the bit location and function for the selected box are displayed in the text window and will match those of the register map tables in the device's data sheet. Only the function is displayed when the cursor is over the bit name. Status boxes are similar to control boxes but have an LED appearance and a colored background.

Figure 18-43 shows SERDES Buffer Options window. Configuration options can be selected from the pull-down menu.

Figure 18-43. SERDES Buffer Options Window



More information and downloadable files for ORCAstra can be found on the Lattice Semiconductor website at the following address: www.latticesemi.com/products/designsoftware/orcastra.cfm.

Other Design Considerations

LatticeECP2M-35 vs. All Other LatticeECP2M Devices

Important Note: The SERDES settings (in the .txt file) for the LatticeECP2M-35 are different from all other LatticeECP2M devices. The difference includes default settings that are hidden in the .txt file. When there is device migration from LatticeECP2M-35 to all other LatticeECP2M devices, users must re-generate the PCS module from the IPexpress GUI.

Engineering Samples vs. Production Devices

Engineering samples and production devices have their own simulation models and libraries. The library name for the engineering sample is pcsc_ mti_work_revA.

Simulation of the SERDES/PCS

SERDES/PCS simulation support is provided by a pre-compiled model for ModelSim® and NC-Verilog®. The simulation model that is pre-compiled is a behavioral model for the SERDES and the RTL of the PCS. Table 18-23 provides the location for each of the simulation models.

Table 18-23. Simulation Model Locations

Simulator	Model Location
Cadence NC-Verilog/VHDL, NC-Sim, Synposys VCS, Mentor Graphics ModelSim, Aldec Riviera Pro	ispTools\cae_library\simulation\blackbox

Note: Model file names with “revA” are for LatticeECP2M-35 engineering samples only.

Models that are distributed in .zip files need to be decompressed before the model can be used.

Reset Usage in Simulation

If any of the PCS reset signals is not tied to GND, the designer must be careful if the same reset signal is used in the FPGA core. For example, if one of the reset signals is used to reset both the PCS and a counter in the FPGA core, and the counter uses txfullclk, the counter will not work. The counter cannot reset itself because txfullck is not running when the reset is active.

Depending on the reference clock sources, the reset assertion time can vary.

For simulation only, it is recommended to use a minimum active duration of 100ns for PCS reset signals.

16/20-bit Word Alignment

The PCS receiver can not recognize the 16-bit word boundary. With COMMA_ALIGN in 'AUTO' mode, the PCS can only do BYTE alignment. The 16-bit word alignment should be done in the FPGA fabric and is fairly straight forward. The simulation model works in the same way. It can be enhanced if users implement an alignment scheme as described below.

For example, if transmit data before at the FPGA interface are:

`YZABCDEFGHIJKLM`... (each letter is a byte, 8-bit or 10-bit)

Then the incoming data in PCS after 8b10b decoder and before rx_gearbox are:

`YZABCDEFGHIJKLM`....

After rx_gearbox, they can become:

1. `{ZY}{BA}{DC}{FE}{HG}{JI}{LK}`....

or

2. `{AZ}{CB}{ED}{GF}{IH}{KJ}{ML}`...

Clearly sequence 2 is not aligned. It has one byte offset, but 16/20-bit alignment is needed. Let’s say the special character 'A' should be always placed in the lower byte.

Flopping one 20-bit data combines with the current 16/20-bit data to form 32/40-bit data as shown below:

1. `{DCBA}{HGFE}{LKJI}`....

```
     ^
     | **Found the A in lower 10-bit, set the offset to '0', send out aligned
     data 'BA'
```

Next clock cycle:

```
{FEDC}{JIHG}{NMLK}....
      ^
      | **send out aligned data 'DC'
```

etc.

After the 16/20-bit alignment, the output data are:

```
{ZY}{BA}{DC}{FE}{HG}{JI}{LK}....
```

2.
```
{CBAZ}{GFED}{KJIH}....
    ^
    | **Found the A in upper 10-bit, set the offset to '10', send out aligned
    data 'BA'
```

Next clock cycle:

```
{EDCB}{IHGF}{MLKJ}....
  ^
  | **send out aligned data 'DC'
```

etc.

After the 20-bit alignment, the output data are:

```
{ZY}{BA}{DC}{FE}{HG}{JI}{LK}....
```

Note: The LSB of a 8/10-bit byte or a 16/20-bit word is always transmitted first and received first.

Switching Between 10XH, 10X and 20X Reference Clock Multiplier Modes Using SCI

Designers can change the bit rates between 10XH, 10X and 20X reference clock multiplier modes using the SERDES Client Interface. This is a useful feature in a system where different rates are received at times and the receiver is able to sweep between different rates and lock to one of them. For example, a system can sweep three different rates of 622Mbps, 1.244Gbps and 2.488Gbps and lock to one of them whichever is received.

There are a few control register bits associated with different rates.

```
CH_0A[D1]: rate_mode_tx
CH_0B[D1]: rate_mode_rx
QD_13[D6]: refclk_mode[0]
QD_18[D5]: quad_rst
```

Switching Between 20X to 20XH or 10X to 10XH Mode in 16-Bit Interface

In addition to the control register settings described above, the transmit interface clock(ff_txiclk_chn) input must be switched from txhalfclk to txqtrclk as described in Table 18-5. ff_rxiclk_chn is not affected in this case. The CDR PLL will automatically tune to the incoming data rate and provide the correct rxhalfclk as described in Table 18-5.

Contact the Lattice Semiconductor Technical Support Group for detailed information.

Off-Chip AC Coupling

When off-chip AC coupling is required, the recommended capacitor values are shown in Table 18-24.

Table 18-24. Off-chip AC Coupling Capacitor Values

Protocol	Min.	Typ.	Max.	Remark	Units
8b10b	4.7	10		@ 3.125Gbps	nF
PCI Express rev1.1	75		200		nF

When a DC balanced pattern is used, such as 8b10b encoding, a capacitor of minimum 4.7nF, is required to reduce the edge degradation.

Data patterns with longer run lengths require larger capacitance values to reduce pattern dependent jitter.

Refer to Lattice technical note FPGA-TN-02077, Electrical Recommendations for Lattice SERDES, for more information.

Unused Quad/Channel and Power Supply

On unused channels, VCCTX, VCCRX, VCCP and VCCAUX33 should be powered up. VCCIB, VCCOB, HDINP/N, HDOUTP/N and REFCLKP/N should be left floating. Unused channel outputs are tristated, with approximately 10 KOhm internal resistor connecting between the differential output pair.

VCCAUX33 supplies power to termination resistors. It is recommended to have the PI filter like the other power supplies. VCCAUX33 noise will directly be coupled to high-speed I/O, HDIN/HDOUT. If VCCAUX(FPGA core power supply) is very clean, it can be connected to VCCAUX33.

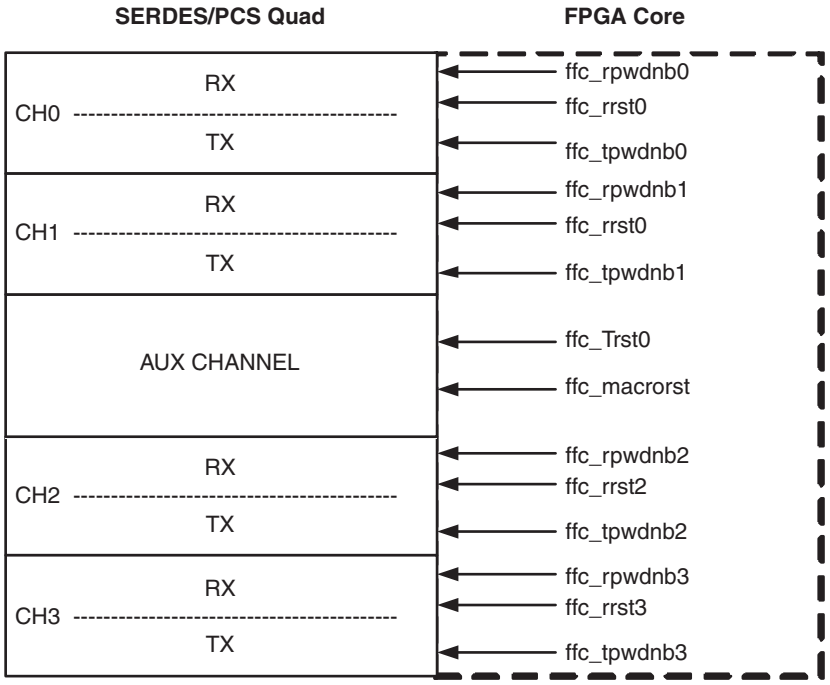
Also, unused SERDES is configured in power down mode by default.

Reset and Power-Down Control

The SERDES quad has reset and power-down controls for the whole macro as well as individual reset and power-down controls for each transmitter and receiver as shown in Figure 18-44. The reset signals are active high and the power-down signals are active low. The operation of the various reset and power-down controls are described in the following sections.

Note: When the device is powering up and the chip level power-on-reset is active, the SERDES control bits (in the PCS) will be cleared (or will take on their default value). This will put the SERDES quad into the power-down state.

Figure 18-44. SERDES/PCS Quad Reset and Power-Down Controls



Reset generation and distribution is handled by the clocks and resets block. Typically, all resets are via hard reset and various FPGA fabric resets. Reset pulse widths are a function of the source. However, all should be at least a clock wide. The reset logic is shown in Figure 18-45 and the corresponding table is in Table 18-25.

Table 18-25. SERDES/PCS Reset Table

Reset Signals		PCS[1] TX	PCS[1] RX	SERDES TX	SERDES RX	PCS CTRL Registers	TX PLL	CDR PLL
FPGA	Control Register							
ffc_lane_tx_rst_ch[3:0]	lane_tx_rst[3:0]	X						
ffc_lane_rx_rst_ch[3:0]	lane_rx_rst[3:0]		X					
ffc_quad_rst	quad_rst	X	X	X	X		X	X
ffc_macro_rst	macro_rst			X	X		X	X
ffc_rrst_ch[3:0][2]	rrst[3:0][2]				X			
ffc_trst_ch[3:0][3]	trst[3:0][3]						X	
TRI_ION (configuration)		X	X	X	X	X		

1. Includes SB (SERDES Bridge), PCS core and FB (FPGA Bridge) sub-blocks.
2. For internal use only. This reset should always be tied to '0'.
3. Only resets TX PLL loss of lock (ffx_plol).

Figure 18-45. SERDES/PCS Reset Diagram

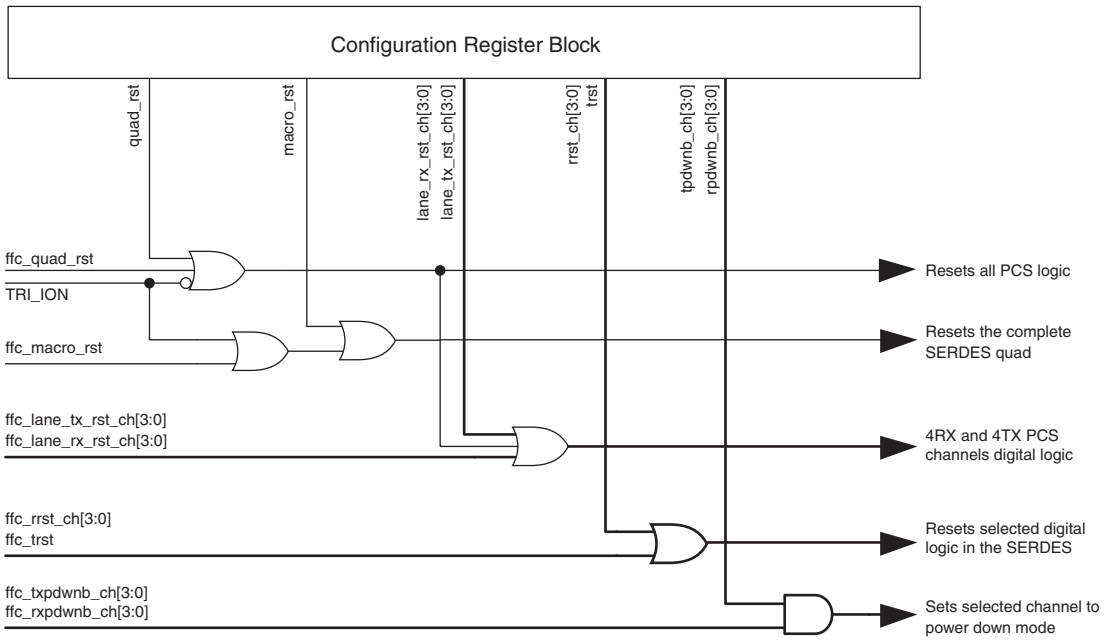


Table 18-26. Reset Controls Description[1, 2, 3]

Rest Signal		Description
FPGA	Control Register	
ffc_quad_rst	quad_rst	Active high, asynchronous input. Resets all SERDES channels including the auxiliary channel and PCS. This reset includes macro_rst, txpll, cdr, lane_tx_rst, and lane_rx_rst.
ffc_macro_rst	macro_rst	Active high, asynchronous input to the SERDES quad. Gated with software register bit. This reset is for the SERDES block only and TXPLL and CDRPLL are included.
ffc_lane_tx/rx_rst_ch[0:3]	lane_tx/rx_rst[0:3]	Active high, asynchronous input. Resets individual TX/RX channel in SB, PCS core and FB blocks.
ffc_rrst_ch[0:3]	rrst[0:3]	Resets loss-of-lock (rlol) and loss-of-signal circuits. Does not reset CDR PLL.
ffc_trst	trst	Resets the loss-of-lock of AUX PLL (plol).

1. For all channels in the quad running in full-data-rate mode, parallel side clocks are guaranteed to be in-phase.
2. For all channels in the quad running in half-data-rate mode, each channel has a separate divide-by-two circuit. Since there is no mechanism in the quad to guarantee that these divide by two circuits are in phase after de-assertion of “macrorst”, the PCS design should assume that the dividers (and therefore the parallel side clocks) are NOT in phase.
3. In half-data-rate mode, since there is no guarantee that the parallel side clocks are in phase, this may add channel-to-channel skew to both transmit and receive sides of a multi-channel link.

Power-Down Controls Description

Each Rx and Tx channel can be individually powered-down by a software register bit or a control signal from the FPGA. The individual channel power-down control bits will only power down selected blocks within the SERDES macro and the high-speed I/O buffers.

Table 18-27. Power-Down Controls Descriptions

Signal		Description
FPGA	Register	
macropdb		Active low asynchronous input to the SERDES quad, acts on all channels including the auxiliary channel. When driven low, it powers down the whole macro including the transmit PLL. All clocks are stopped and the macro power dissipation is minimized.
ffc_txpwdnb_ch[0:3]	tpwdnb[0:3]	Active Low Transmit Channel Power Down – Powers down the serilizer and output driver.
ffc_rxpwdnb_ch[0:3]	rpwdnb[0:3]	Active Low Receive Channel Power Down – Powers down CDR, input buffer (equalizer and amplifier) and loss-of-signal detector.

Table 18-28. Reset Pulse Specification

Parameter	Description	Min.	Typ.	Max.	Units
$t_{MACRORST}$	Macro reset high time	1			µs
t_{RRST}	Channel RX reset high time	3			ns
t_{TRST}	Quad TX reset high time	3			ns

Table 18-29. Power-Down/Power-Up Timing Specification

Parameter	Description	Min.	Typ.	Max.	Unit
t_{PWRDN}	Power-down time after macropdb			10	µs
t_{PWRUP}	Power-up tim after macropdb			100	µs

SERDES/PCS Reset

Reset Sequence and Reset State Diagram

After power-up and configuration, all SERDES resets and FPGA resets are applied.

Lock Status Signals Definitions

ffs_plol : 1 = TX PLL loss of lock.
: 0 = TX PLL lock.
It takes 1,400,000 UI to declare the lock of the TX PLL.

ffs_rlol_ch[3:0] : 1 = CDR loss of lock.
: 0 = Lock maintained.
It takes 400,000 reference clock cycles (worst case) to declare the lock of the CDR PLL.

ffs_rlos_lo_ch[3:0] : 1 = Loss of signal detection for each channel.
: 0 = Signal detected.

The ffs_rlol_ch[3:0] status signal is an indicator of the CDR lock status as defined above. However, during the CDR locking process, the CDR PLL will lock to the reference clock when there is no input data present. This purpose of this feature is to avoid ignoring the input data when it is restored.

In order to ensure the presence of input data during CDR lock status checking, it is recommended to use the ffs_rlos_lo_ch[3:0] signal in conjunction with the ffs_rlol_ch[3:0] signal.

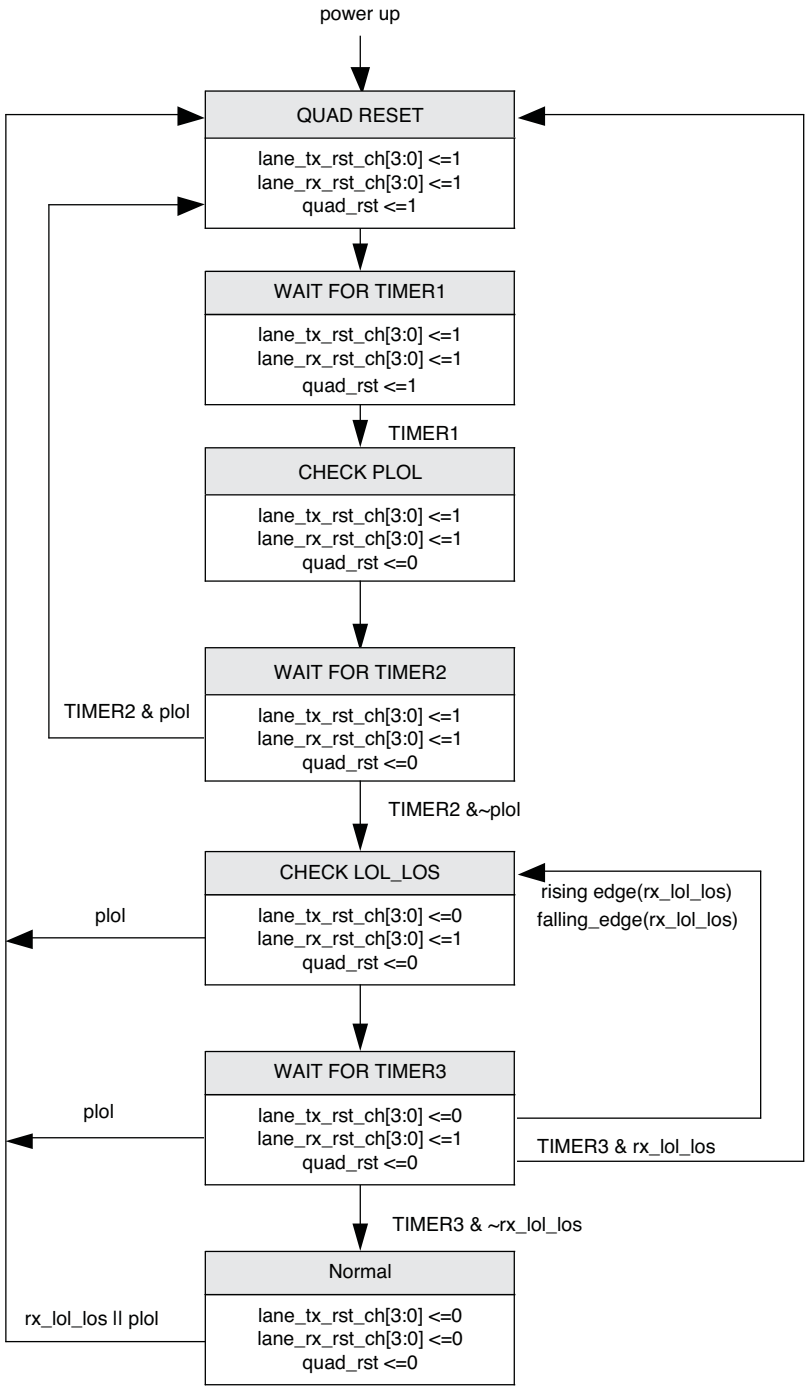
In most applications, combining the two status signals, ffs_rlol_ch[3:0]_s and ffs_rlos_lo_ch[3:0], will work as the indicator of loss of CDR lock. But in applications where high-speed traffic is heavily loaded (i.e. multi-channel, multi-quad), ffs_rlos_lo_ch[3:0] may not represent the correct signal status due to the sensitive nature of the detection circuit. It is strongly recommended to use the protocol-level CDR lock status signal in the RX reset sequence instead of ffs_rlol_ch[3:0].

Reset Sequence

0. QUAD_RESET – At power up, assert ffc_quad_rst, ffc_lane_tx_rst_ch[3:0] and ffc_lane_rx_rst_ch[3:0].
1. WAIT_FOR_TIMER1 – Start TIMER1. Wait a minimum of 20 ns.
2. CHECK_PLOL – Release ffc_quad_rst.
3. WAIT_FOR_TIMER2 – Start TIMER2. If TIMER2 expires and the TX PLL is not locked, go to step 0.
4. CHECK_LOL_LOS – Wait for rx_lol_los[3:0] to go low. Reset TIMER3.
5. WAIT_FOR_TIMER3 – Wait for both ffs_rlol_ch[3:0] and ffs_rlos_lo_ch[3:0] to go low
   If there is a transition in rx_lol_los (ffs_rlol_ch[3:0] || ffs_rlos_lo_ch[3:0]), go to step 4.
   If TIMER2 expires with rx_lol_los = 1, go to step 0.
6. NORMAL – Release ffc_lane_tx_rst_ch[3:0]. If ffs_plol goes high during normal operation, go to step 0.

The reset sequence state diagram is described in Figure 18-46.

Figure 18-46. Reset State Diagram



Notes:
TIMER1: ffc_quad_rst asserted for a minimum of 20 ns.

TIMER2: Time to declare TX PLL lock : 1,400,000 UI.

TIMER 3: Time for rx_lol_los signal to stay low (400,000 reference clock cycles).
Any FPGA clock can be used to satisfy the timer requirement.

In Figure 18-46, rx_lol_los is defined as ffs_rlol_ch[3:0] || ffs_rlos_lo_ch[3:0]. If a protocol-level CDR lock status signal is provided, it can replace the ffs_rlol_ch[3:0] signal.

Power Supply Sequencing Requirements

When using the SERDES with 1.5V VCCIB or VCCOB, the SERDES should not be left in a steady state condition with the 1.5V power applied and the 1.2V power not applied. Both the 1.2V and the 1.5V power should be applied to the SERDES at nominally the same time. The normal variation in ramp_up times of power supplies and voltage regulators is not a concern.

References

- Technical notes:
  - TN1029, FPSC SERDES CML Buffer Interface
  - TN1084, LatticeSC SERDES Jitter
  - FPGA-TN-02077, Electrical Recommendations for Lattice SERDES
  - TN1159, LatticeECP2/M Pin Assignment Recommendations
- HB1003, LatticeECP2/M Family Handbook

Technical Support Assistance

Submit a technical support case through techsupport@latticesemi.com.

Appendix A. Memory Map

Configuration Register Definition

There are specific quad-level registers and channel-level registers. In each category, there are SERDES specific registers and PCS specific registers. Within these sub-categories there are:

- Control registers
- Status registers
- Status registers with clear-on-read
- Interrupt Control registers
- Interrupt Status registers
- Interrupt Source registers (clear-on-read)

All register bits shown below are with D0 as the LEAST significant bit on the right.

The following indications are the nomenclature for what types of register each is: (R/W = Read/Write, RO = Read Only, CR = Clear on a read).

Each of these register bits is “shadowed” with a memory cell. These memory cells are only programmable through bitstream control. After configuration is complete, the configuration memory cells that have associated registers are automatically copied into the registers. Subsequent changes to the contents of the registers will not affect the value stored in the configuration memory cells but will change the operation of the SERDES/PCS quad.

All “reserved” bits are written to zero.

Per Quad Register Overview

Table 18-30. Quad Interface Register Map

BA[1]	Register Name	D7	D6	D5	D4	D3	D2	D1	D0
1. PER QUAD CONTROL REGISTERS (28)									
Per Quad PCS Control Registers (17)									
00	qd_00	sync_toggle	force_int	char_mode	xge_mode	rio_mode	pcie_mode	fc_mode	uc_mode[5]
01	qd_01	bist_rpt_ch_sel[1][6]	bist_rpt_ch_sel[0]	bist_res_sel[1]	bist_res_sel[0]	bist_time_sel[1]	bist_time_sel[0]	bist_head_sel[1]	bist_head_sel[0]
02	qd_02	high_mark[3]	high_mark[2]	high_mark[1]	high_mark[0]	low_mark[3]	low_mark[2]	low_mark[1]	low_mark[0]
03	qd_03	min_ipg_cnt[1]	min_ipg_cnt[0]	match_4_enable	match_2_enable		pfifo_clr_sel	asyn_mode	sel_test_clk
04	qd_04	cc_match_1[7]	cc_match_1[6]	cc_match_1[5]	cc_match_1[4]	cc_match_1[3]	cc_match_1[2]	cc_match_1[1]	cc_match_1[0]
05	qd_05	cc_match_2[7]	cc_match_2[6]	cc_match_2[5]	cc_match_2[4]	cc_match_2[3]	cc_match_2[2]	cc_match_2[1]	cc_match_2[0]
06	qd_06	cc_match_3[7]	cc_match_3[6]	cc_match_3[5]	cc_match_3[4]	cc_match_3[3]	cc_match_3[2]	cc_match_3[1]	cc_match_3[0]
07	qd_07	cc_match_4[7]	cc_match_4[6]	cc_match_4[5]	cc_match_4[4]	cc_match_4[3]	cc_match_4[2]	cc_match_4[1]	cc_match_4[0]
08	qd_08	cc_match_4[9]	cc_match_4[8]	cc_match_3[9]	cc_match_3[8]	cc_match_2[9]	cc_match_2[8]	cc_match_1[9]	cc_match_1[8]
09	qd_09	udf_comma_mask[7]	udf_comma_mask[6]	udf_comma_mask[5]	udf_comma_mask[4]	udf_comma_mask[3]	udf_comma_mask[2]	udf_comma_mask[1]	udf_comma_mask[0]
0A	qd_0a	udf_comma_a[7]	udf_comma_a[6]	udf_comma_a[5]	udf_comma_a[4]	udf_comma_a[3]	udf_comma_a[2]	udf_comma_a[1]	udf_comma_a[0]
0B	qd_0b	udf_comma_b[7]	udf_comma_b[6]	udf_comma_b[5]	udf_comma_b[4]	udf_comma_b[3]	udf_comma_b[2]	udf_comma_b[1]	udf_comma_b[0]
0C	qd_0c	udf_comma_a[9]	udf_comma_a[8]	udf_comma_b[9]	udf_comma_b[8]	udf_comma_mask[9]	udf_comma_mask[8]	bist_mode	bist_en
0D	qd_0d	bist_udf_def_header [7]	bist_udf_def_header [6]	bist_udf_def_header [5]	bist_udf_def_header [4]	bist_udf_def_header [3]	bist_udf_def_header [2]	bist_udf_def_header [1]	bist_udf_def_header [0]
0E	qd_0e	bist_udf_def_header [15]	bist_udf_def_header [14]	bist_udf_def_header [13]	bist_udf_def_header [12]	bist_udf_def_header [11]	bist_udf_def_header [10]	bist_udf_def_header [9]	bist_udf_def_header [8]
0F	qd_0f	bist_bus8bit_sel	bist_ptn_sel[2]	bist_ptn_sel[1]	bist_ptn_sel[0]	bist_udf_def_header [19]	bist_udf_def_header [18]	bist_udf_def_header [17]	bist_udf_def_header [16]
10	qd_int_10[2]	ls_sync_status_3_int_ctl	ls_sync_status_2_int_ctl	ls_sync_status_1_int_ctl	ls_sync_status_0_int_ctl	ls_sync_statusn_3_int_ctl	ls_sync_statusn_2_int_ctl	ls_sync_statusn_1_int_ctl	ls_sync_statusn_0_int_ctl
Per Quad SERDES Control Registers (6)									
11	qd_11	reserved	reserved	tx_refck_sel	refck_dcc_en	refck_rterm	refck_out_sel[2]	refck_out_sel[1]	refck_out_sel[0]
12	qd_12	refck25x	bus8bit_sel	rlos_hset[2]	rlos_hset[1]	rlos_hset[0]	rlos_lset[2]	rlos_lset[1]	rlos_lset[0]
13	qd_13	refck_mode[1]	refck_mode[0]	reserved	reserved	reserved	reserved	cdr_lol_set[1]	cdr_lol_set[0]
14	qd_14	reserved	reserved	reserved	pll_lol_set[1]	pll_lol_set[0]	tx_vco_ck_div[2]	tx_vco_ck_div[1]	tx_vco_ck_div[0]
15	qd_15	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
16	qd_int_16[2]	plol_int_ctl	~plol_int_ctl	reserved	reserved	reserved	reserved	reserved	reserved
Per Quad Clock Reset Registers (5)									
17	qd_17	lane_rx_rst3	lane_rx_rst2	lane_rx_rst1	lane_rx_rst0	lane_tx_rst3	lane_tx_rst2	lane_tx_rst1	lane_tx_rst0
18	qd_18	macropdb	macro_rst	quad_rst	trst	rrst[3]	rrst[2]	rrst[1]	rrst[0]
19	qd_19	bist_rx_data_sel	bist_bypass_tx_gate	bist_sync_head_req [1]	bist_sync_head_req [0]	sel_sd_rx_clk3	sel_sd_rx_clk2	sel_sd_rx_clk1	sel_sd_rx_clk0
1A	qd_1a	ff_rx_clk_sel_2[3]	ff_rx_clk_sel_2[2]	ff_rx_clk_sel_2[1]	ff_rx_clk_sel_2[0]	ff_rx_clk_sel_1[3]	ff_rx_clk_sel_1[2]	ff_rx_clk_sel_1[1]	ff_rx_clk_sel_1[0]
1B	qd_1b	ff_tx_clk_sel[2]	ff_tx_clk_sel[1]	ff_tx_clk_sel[0]	reserved	ff_rx_clk_sel_0[3]	ff_rx_clk_sel_0[2]	ff_rx_clk_sel_0[1]	ff_rx_clk_sel_0[0]
2. PER QUAD STATUS REGISTERS (9)									
Per Quad PCS Status Registers (5)									
20	qd_20				int_qua_out	int_cha[3]	int_cha[2]	int_cha[1]	int_cha[0]
21	qd_21[3]	ls_sync_status_3	ls_sync_status_2	ls_sync_status_1	ls_sync_status_0	ls_sync_statusn_3	ls_sync_statusn_2	ls_sync_statusn_1	ls_sync_statusn_0
22	qd_int_22[4]	ls_sync_status_3_int	ls_sync_status_2_int	ls_sync_status_1_int	ls_sync_status_0_int	ls_sync_statusn_3_int	ls_sync_statusn_2_int	ls_sync_statusn_1_int	ls_sync_statusn_0_int
23	qd_23	bist_report[7]	bist_report[6]	bist_report[5]	bist_report[4]	bist_report[3]	bist_report[2]	bist_report[1]	bist_report[0]
24	qd_24	bist_report[15]	bist_report[14]	bist_report[13]	bist_report[12]	bist_report[11]	bist_report[10]	bist_report[9]	bist_report[8]
Per Quad SERDES Status Registers (4)									
25	qd_25[3]	plol	~plol	reserved	reserved	reserved	reserved	reserved	reserved
26	qd_int_26[4]	plol_int	~plol_int	reserved	reserved	reserved	reserved	reserved	reserved
27	qd_27	reserved	reserved	pll_calib_status[5]	pll_calib_status[4]	pll_calib_status[3]	pll_calib_status[2]	pll_calib_status[1]	pll_calib_status[0]
28	qd_28	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved

1. BA = Base Address (Hex)
2. Interrupt control register related to an interruptible status (int_sts_x) register.
3. Status register which has an associated interruptible status (int_sts_x) register.
4. Interruptible status register; clear on read (has associated control register and status register)
5. uc_mode: 8-bit SERDES Only and 10-bit SERDES Only.
6. BIST is a built-in PRBS generator and checker for internal use only.

Note: Default value is "0", unless otherwise specified.

Per Quad PCS Control Register Details

Table 18-31. PCS Control Register 1 (QD_00)

Bit	Name	Description	Type	Default
7	sync_toggle	Transition = Reset the 4 Tx Serializer to minimize Tx Lane-to-Lane Skew Level = Normal operation of Tx Serializers		
6	force_int	1 = Force to generate interrupt signal 0 = Normal operation	R/W	0
5	char_mode	1 = Enable SERDES characterization mode 0 = Disable SERDES characterization mode	R/W	0
4	xge_mode	1 = Selects 10Gb Ethernet 0 = Selects 1Gb Ethernet mode	R/W	0
3	rio_mode	1 = Selects Rapid-IO mode 0 = Selects other mode (10GbE, 1GbE)	R/W	0
2	pcie_mode	1 = PCI Express mode of operation 0 = Selects other mode (RapidIO, 10GbE, 1GbE)	R/W	0
1	fc_mode	1 = Select Fibre Channel mode 0 = Selects other mode (PCI Express, RapidIO, 10GbE, 1GbE)	R/W	0
0	uc_mode	1 = Selects User Configured mode 0 = Selects other mode (PCI Express, RapidIO, 10GbE, 1GbE)		

Note: Bits 0 to 3 are mutually exclusive. Only one of the bits can be set.

Table 18-32. PCS Control Register 2 (QD_01)

Bit	Name	Description	Type	Default
7:6	bist_rpt_ch_sel [1:0]	00 = BIST report from channel 0 01 = BIST report from channel 1 10 = BIST report from channel 2 11 = BIST report from channel 3	R/W	00
5:4	bist_res_sel [1:0]	BIST resolution selection 00 = no error 01 < 2 errors 10 < 16 errors 11 < 128 errors	R/W	00
3:2	bist_time_sel [1:0]	BIST time selection: 00 = 5e+8 cycles 01 = 5e+9 cycles 10 = 5e+6 cycles 11 = 100K cycles	R/W	00
1:0	bist_head_sel [1:0]	BIST header selection 00 = K28_5 (K28_5=10'h305 K28_5_=10'h0FA) 01 = A1A2 (MA1=10'h1F6; MA2=10'h128) 10 = 10'h1BC 11 = user defined	R/W	0

Table 18-33. PCS Control Register 3 (QD_02)

Bit	Name	Description	Type	Default
7:4	high_mark [3:0]	Clock compensation FIFO high water mark. Mean is 4'b1000	R/W	4'b0111
3:0	low_mark [3:0]	Clock compensation FIFO low water mark. Mean is 4'b1000	R/W	4'b1001

Table 18-34. PCS Control Register 4 (QD_03)

Bit	Name	Description	Type	Default
7:6	min_ipg_cnt [1:0]	Minimum IPG to enforce	R/W	2'b11
5	match_4_enable	1 = enable four character skip matching (using match 4, 3, 2, 1)	R/W	0
4	match_2_enable	1 = enable two character skip matching (using match 4,3)	R/W	1
3	Reserved			
2	pfifo_clr_sel	1 = pfifo_clr signal or channel register bit clears the FIFO 0 = pfifo_error internal signal self clears the FIFO	R/W	0
1	asyn_mode	For test only, select asynchronous reset	R/W	0
0	sel_test_clk	For test only, select test clock	R/W	0

Table 18-35. PCS Control Register 5 - CC match 1 LO (QD_04)

Bit	Name	Description	Type	Default
7:0	cc_match_1 [7:0]	Lower bits of user defined clock compensator skip pattern 1	R/W	8'h00

Table 18-36. PCS Control Register 6 - CC match 2 LO (QD_05)

Bit	Name	Description	Type	Default
7:0	cc_match_2 [7:0]	Lower bits of user defined clock compensator skip pattern 2	R/W	8'h00

Table 18-37. PCS Control Register 7 - CC match 3 LO (QD_06)

Bit	Name	Description	Type	Default
7:0	cc_match_3 [7:0]	Lower bits of user defined clock compensator skip pattern 3	R/W	8'hBC

Table 18-38. PCS Control Register 8 - CC match 4 LO (QD_07)

Bit	Name	Description	Type	Default
7:0	cc_match_4 [7:0]	Lower bits of user defined clock compensator skip pattern 4	R/W	8'h50

Table 18-39. PCS Control Register 9 - CC match HI (QD_08)

Bit	Name	Description	Type	Default
7:6	cc_match_4 [9:8]	Upper bits of user defined clock compensator skip pattern 4 [9] = Disparity error [8] = K control	R/W	2'b01
5:4	cc_match_3 [9:8]	Upper bits of user defined clock compensator skip pattern 3 [9] = Disparity error [8] = K control	R/W	2'b01
3:2	cc_match_2 [9:8]	Upper bits of user defined clock compensator skip pattern 2 [9] = Disparity error [8] = K control	R/W	2'b00
1:0	cc_match_1 [9:8]	Upper bits of user defined clock compensator skip pattern 1 [9] = Disparity error [8] = K control	R/W	2'b00

Table 18-40. PCS Control Register 10 - UDF comma mask LO (QD_09)

Bit	Name	Description	Type	Default
7:0	udf_comma_mask [7:0]	Lower bits of user defined comma mask	R/W	8'hFF

Table 18-41. PCS Control Register 11 - UDF comma a LO (QD_0A)

Bit	Name	Description	Type	Default
7:0	udf_comma_a [7:0]	Lower bits of user defined comma character ‘a’	R/W	8’h83

Table 18-42. PCS Control Register 12 - UDF comma b LO (QD_0B)

Bit	Name	Description	Type	Default
7:0	udf_comma_b [7:0]	Lower bits of user defined comma character ‘b’	R/W	8’h7C

Table 18-43. PCS Control Register 13 - UDF comma HI (QD_0C)

Bit	Name	Description	Type	Default
7:6	udf_comma_a [9:8]	Upper bits of user defined comma character ‘a’	R/W	2’b10
5:4	udf_comma_b [9:8]	Upper bits of user defined comma character ‘b’	R/W	2’b01
3:2	udf_comma_mask [9:8]	Upper bits of user defined comma mask	R/W	2’b11
1	bist_mode	1 = Continuous Bist Mode 0 = Timed BIST mode	R/W	0
0	bist_en	1 = Enable PCS BIST 0 = Normal operation.	R/W	0

Table 18-44. PCS Control Register 14 - UDF BIST header LO (QD_0D)

Bit	Name	Description	Type	Default
7:0	bist_udf_def_header [7:0]	Lower bits of user defined header for BIST	R/W	8’h00

Table 18-45. PCS Control Register 15 - UDF BIST header MD (QD_0E)

Bit	Name	Description	Type	Default
7:0	bist_udf_def_header [15:8]	Middle bits of user defined header for BIST	R/W	8’h00

Table 18-46. PCS Control Register 16 - UDF BIST header HI (QD_0F)

Bit	Name	Description	Type	Default
7	bist_bus8bit_sel	1 = 8 bit data BIST 0 = 10 bit data BIST	R/W	0
6:4	bist_ptn_sel[2:0]	BIST pattern selection: 000 = PRBS11 001 = max data rate 010 = PRBS31 011 = PRBS21100 = K28_5 101 = A1A2110 = 5150 (repeat) 111 = 2120 (repeat)	R/W	0
3:0	bist_udf_def_header [19:16]	High bits of user defined header for BIST	R/W	8’h00

Table 18-47. PCS Interrupt Control Register 17 (QD_10)

Bit	Name	Description	Type	Default
7	ls_sync_status_3_int_ctl	1 = Enable interrupt for ls_sync_status_3 (in sync) 0 = Disable interrupt for ls_sync_status_3 (in sync)	R/W	0
6	ls_sync_status_2_int_ctl	1 = Enable interrupt for ls_sync_status_2 (in sync) 0 = Disable interrupt for ls_sync_status_2 (in sync)	R/W	0
5	ls_sync_status_1_int_ctl	1 = Enable interrupt for ls_sync_status_1 (in sync) 0 = Disable interrupt for ls_sync_status_1 (in sync)	R/W	0
4	ls_sync_statusn_3_int_ctl	1 = Enable interrupt for ls_sync_status_3 when it goes low (out of sync) 0 = Disable interrupt for ls_sync_status_3 when it goes low (out of sync)	R/W	0
4	ls_sync_status_0_int_ctl	1 = Enable interrupt for ls_sync_status_0 (in sync) 0 = Disable interrupt for ls_sync_status_0 (in sync)	R/W	0
3	ls_sync_statusn_2_int_ctl	1 = Enable interrupt for ls_sync_status_2 when it goes low (out of sync) 0 = Disable interrupt for ls_sync_status_2 when it goes low (out of sync)	R/W	0
1	ls_sync_statusn_1_int_ctl	1 = Enable interrupt for ls_sync_status_1 when it goes low (out of sync) 0 = Disable interrupt for ls_sync_status_1 when it goes low (out of sync)	R/W	0
0	ls_sync_statusn_0_int_ctl	1 = Enable interrupt for ls_sync_status_0 when it goes low (out of sync) 0 = Disable interrupt for ls_sync_status_0 when it goes low (out of sync)	R/W	0

Per Quad SERDES Control Register Details

Note: Except indicated, all channels must be reset after writing any SERDES control register.

Table 18-48. SERDES Control Register 1 (QD_11)

Bit	Name	Description	Type	Default
7:6	Reserved			
5	TX_REFCK_SEL	TXPLL reference clock select 0 = refclk (differential input) 1 = core_txrefclk	R/W	0
4	REFCK_DCC_EN	1 = Reference clock DC coupling enable 0 = Reference clock DC coupling disable (default)	R/W	0
3	REFCK_RTERM	Termination at reference clock input buffer 0 = 50 ohm (default) 1 = high impedance	R/W	0
2:0	REFCL_OUT_SEL[2:0]	Refck control [0]: 0 = refck buffer enable, 1 = refck buffer disable [1]: 0 = refck2core disable, 1 = refck2core enable [2]: reserved Refer to Figure 18-47.	R/W	3'b000

Figure 18-47. Reference Clock Select Control

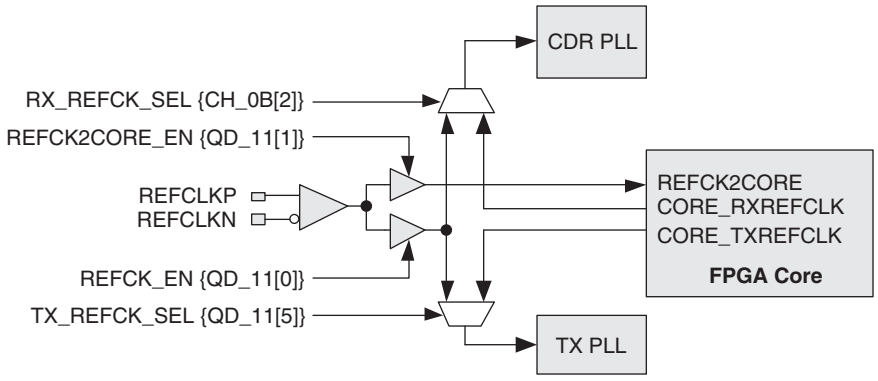


Table 18-49. SERDES Control Register 2 (QD_12)

Bit	Name	Description	Type	Default
7	REFCK25X	1 = Internal high speed bit clock is 25x (for reference clock = 100MHz only) 0 = See REFCK_MODE	R/W	0
6	BUS8BIT_SEL	1 = Select 8-bit bus width 0 = Select 10-bit bus width (default)	R/W	0
5:3	RLOS_HSET [2:0]	LOS detector reference current adjustment for larger swing 000 = default 001 = +10% 010 = +15% 011 = +25% 100 = -10% 101 = -15% 110 = -25% 111 = -30%	R/W	3'b000
2:0	RLOS_LSET [2:0]	(Internal use only)	R/W	3'b000

Table 18-50. SERDES Control Register 3 (QD_13)

Bit	Name	Description	R/W	Default
7:6	REFCK_MODE [1:0]	00 = Internal high speed bit clock is 20x or 16x 01 = Internal high speed bit clock is 10x or 8x 1X = Reserved	R/W	2'b00
5:2	Reserved			
1:0	CDR_LOL_SET [1:0]	CDR loss of lock setting: Lock / Unlock 00 = +/-1000ppm x2 / +/-1500ppm x2 01 = +/-2000ppm x2 / +/-2500ppm x2 10 = +/-4000ppm / +/-7000ppm 11 = +/-300ppm / +/-450ppm	R/W	2'b00[1]

1. This is the default value used in ispLEVER 8.1.

Table 18-51. SERDES Control Register 4 (QD_14)

Bit	Name	Description	R/W	Default
7:5	Reserved			
4:3	PLL_LOL_SET [1:0]	TxPLL loss of lock setting Lock / Unlock 00 = +/-300ppm x2 / +/-600ppm x2 01 = +/-300ppm / +/-2000ppm 10 = +/-1500ppm / +/-2200ppm 11 = +/-4000ppm / +/-6000ppm	R/W	2'b00
2:0	TX_VCO_CK_DIV[2:0]	VCO output frequency select: 000 = divided by 1 / 001 = reserved 010 = divided by 2 / 011 = reserved 100 = divided by 4 / 101 = divided by 8 110 = divided by 16 / 111 = divided by 32	R/W	3'b000

Table 18-52. SERDES Control Register 5 (QD_15)

Bit	Name	Description	R/W	Default
7:0	Reserved			

Table 18-53. SERDES Interrupt Control Register 6 (QD_16)

Bit	Name	Description	R/W	Default
7	PLOL_INT_CTL	1 = Interrupt enabled for loss of lock on PLOL. 0 = Interrupt not enabled for loss of lock PLOL.	RO CR	0
6	~PLOL_INT_CTL	1 = Interrupt enabled for obtaining lock on PLOL. 0 = Interrupt not enabled for obtaining lock PLOL.	RO CR	0
5:0	Reserved			

Per Quad Reset and Clock Control Register Details

Table 18-54. Reset and Clock Control Register 1 (QD_17)

Bit	Name	Description	R/W	Default
7	lane_rx_rst3	1 = Assert reset signal to channel 3 receive logic	R/W	0
6	lane_rx_rst2	1 = Assert reset signal to channel 2 receive logic	R/W	0
5	lane_rx_rst1	1 = Assert reset signal to channel 1 receive logic	R/W	0
4	lane_rx_rst0	1 = Assert reset signal to channel 0 receive logic	R/W	0
3	lane_tx_rst3	1 = Assert reset signal to channel 3 transmit logic	R/W	0
2	lane_tx_rst2	1 = Assert reset signal to channel 2 transmit logic	R/W	0
1	lane_tx_rst1	1 = Assert reset signal to channel 1 transmit logic	R/W	0
0	lane_tx_rst0	1 = Assert reset signal to channel 0 transmit logic	R/W	0

Table 18-55. Reset and Clock Control Register 2 (QD_18)

Bit	Name	Description	R/W	Default
7	macropdb	0 = Assert power down	R/W	1
6	macro_rst	1 = Assert macro reset	R/W	0
5	quad_rst	1 = Assert quad reset	R/W	0
4	trst	1 = Tx Reset	R/W	0
3:0	rrst [3:0]	1 = Rx channel-based reset	R/W	0

Table 18-56. Reset and Clock Control Register 3 (QD_19)

Bit	Name	Description	R/W	Default
7	bist_rx_data_sel	0 = data from SERDES 1 = data from after 8b10b decoder	R/W	0
6	bist_bypass_tx_gate	0 = start to send BIST data after finding the head 1 = force to send BIST data whether the head is found or not	R/W	0
5:4	bist_sync_head_req [1:0]	BIST sync header counter selection 00 = 5'd5 01 = 5'd8 10 = 5'd14 11 = 5'd24	R/W	00
3	sel_sd_rx_clk3	1 = Select rx_clk3[1] for DS FIFO write clock 0 = Select ff_ebrd_clk_3 for DS FIFO write clock	R/W	0
2	sel_sd_rx_clk2	1 = Select rx_clk2 for DS FIFO write clock 0 = Select ff_ebrd_clk_2 for DS FIFO write clock	R/W	0
1	sel_sd_rx_clk1	1 = Select rx_clk1 for DS FIFO write clock 0 = Select ff_ebrd_clk_1 for DS FIFO write clock	R/W	0
0	sel_sd_rx_clk0	1 = Select rx_clk0 for DS FIFO write clock 0 = Select ff_ebrd_clk_0 for DS FIFO write clock	R/W	0

1. rx_clk3 is channel3 recovered clock.

Table 18-57. Reset and Clock Control Register 4 (QD_1A)

Bit	Name	Description	R/W	Default
7	ff_rx_clk_sel3_2	1 = Disable ff_rx_f_clk for channel 3	R/W	0
6	ff_rx_clk_sel2_2	1 = Disable ff_rx_f_clk for channel 2	R/W	0
5	ff_rx_clk_sel1_2	1 = Disable ff_rx_f_clk for channel 1	R/W	0
4	ff_rx_clk_sel0_2	1 = Disable ff_rx_f_clk for channel 0	R/W	0
3	ff_rx_clk_sel3_1	1 = Enable ff_rx_h_clk for channel 3	R/W	0
2	ff_rx_clk_sel2_1	1 = Enable ff_rx_h_clk for channel 2	R/W	0
1	ff_rx_clk_sel1_1	1 = Enable ff_rx_h_clk for channel 1	R/W	0
0	ff_rx_clk_sel0_1	1 = Enable ff_rx_h_clk for channel 0	R/W	0

Table 18-58. Reset and Clock Control Register 5 (QD_1B)

Bit	Name	Description	R/W	Default
7	ff_tx_clk_sel2	1 = Disable ff_tx_f_clk for quad	R/W	0
6	ff_tx_clk_sel1	1 = Enable ff_tx_h_clk for quad	R/W	1
5	ff_tx_clk_sel0	1 = Enable ff_tx_q_clk for quad	R/W	0
4	Reserved			
3	ff_rx_clk_sel3_0	1 = Enable ff_rx_q_clk for channel 3	R/W	0
2	ff_rx_clk_sel2_0	1 = Enable ff_rx_q_clk for channel 2	R/W	0
1	ff_rx_clk_sel1_0	1 = Enable ff_rx_q_clk for channel 1	R/W	0
0	ff_rx_clk_sel0_0	1 = Enable ff_rx_q_clk for channel 0	R/W	0

Per Quad PCS Status Register Details

Table 18-59. PCS Status Register 1 (QD_20)

Bit	Name	Description	R/W	Int?
7:6	Reserved			
5	ion_delay	Delayed global resetn from tri_ion	RO	N
4	int_qua_out	Per Quad Interrupt status	RO	N
3:0	int_cha_out [3:0]	Per Channel Interrupt status	RO	N

Table 18-60. PCS Status Register 2 (QD_21)

Bit	Name	Description	R/W	Int?
7	ls_sync_status_3	1 = Alarm generated on sync_status_3. 0 = Alarm not generated on sync_status_3.	RO	Y
6	ls_sync_status_2	1 = Alarm generated on sync_status_2. 0 = Alarm not generated on sync_status_2.	RO	Y
5	ls_sync_status_1	1 = Alarm generated on sync_status_1. 0 = Alarm not generated on sync_status_1.	RO	Y
4	ls_sync_status_0	1 = Alarm generated on sync_status_0. 0 = Alarm not generated on sync_status_0.	RO	Y
3	ls_sync_statusn_3	1 = Alarm generated on sync_status_3 when it goes low (out of sync) 0 = Alarm not generated on sync_status_3 when it goes low (out of sync)	RO	Y
2	ls_sync_statusn_2	1 = Alarm generated on sync_status_2 when it goes low (out of sync) 0 = Alarm not generated on sync_status_2 when it goes low (out of sync)	RO	Y
1	ls_sync_statusn_1	1 = Alarm generated on sync_status_1 when it goes low (out of sync) 0 = Alarm not generated on sync_status_1 when it goes low (out of sync)	RO	Y
0	ls_sync_statusn_0	1 = Alarm generated on sync_status_0 when it goes low (out of sync) 0 = Alarm not generated on sync_status_0 when it goes low (out of sync)	RO	Y

Table 18-61. Packet Interrupt Status Register 3 (QD_22)

Bit	Name	Description	R/W	Int?
7	ls_sync_status_3_int	1 = Interrupt generated on sync_status_0 (in sync) 0 = Interrupt not generated on sync_status_0 (in sync)	RO CR	Y
6	ls_sync_status_2_int	1 = Interrupt generated on sync_status_1 (in sync) 0 = Interrupt not generated on sync_status_1 (in sync)	RO CR	Y
5	ls_sync_status_1_int	1 = Interrupt generated on sync_status_2 (in sync) 0 = Interrupt not generated on sync_status_2 (in sync)	RO CR	Y
4	ls_sync_status_0_int	1 = Interrupt generated on sync_status_3 (in sync) 0 = Interrupt not generated on sync_status_3 (in sync)	RO CR	Y
3	ls_sync_statusn_3_int	1 = Interrupt generated on sync_status_3 when it goes low (out of sync) 0 = Interrupt not generated on sync_status_3 when it goes low (out of sync)	RO CR	Y
2	ls_sync_statusn_2_int	1 = Interrupt generated on sync_status_2 when it goes low (out of sync) 0 = Interrupt not generated on sync_status_2 when it goes low (out of sync)	RO CR	Y
1	ls_sync_statusn_1_int	1 = Interrupt generated on sync_status_1 when it goes low (out of sync) 0 = Interrupt not generated on sync_status_1 when it goes low (out of sync)	RO CR	Y
0	ls_sync_statusn_0_int	1 = Interrupt generated on sync_status_0 when it goes low (out of sync) 0 = Interrupt not generated on sync_status_0 when it goes low (out of sync)	RO CR	Y

Table 18-62. PCS BIST Status Register 4 (QD_23)

Bit	Name	Description	R/W	Int?
7:0	bist_report [7:0]	Lower bits of BIST report	RO	N

Table 18-63. PCS BIST Status Register 5 (QD_24)

Bit	Name	Description	R/W	Int?
7:0	bist_report [15:8]	Higher bits of BIST report	RO	N

Per Quad SERDES Status Register Details

Table 18-64. SERDES Status Register 1 (QD_25)

Bit	Name	Description	R/W	Int?
7	PLOL	1 = PLL Loss of lock	RO	Y
6	~PLOL	1 = PLL lock obtained	RO	Y
5:0	Reserved			

Table 18-65. SERDES Interrupt Status Register 2 (QD_26)

Bit	Name	Description	R/W	Int?
7	PLOL_INT	1 = Interrupt generated on PLOL. 0 = Interrupt not generated PLOL.	RO CR	Y
6	~PLOL_INT	1 = Interrupt generated on ~PLOL. 0 = Interrupt not generated ~PLOL.	RO CR	Y
5:0	Reserved			

Table 18-66. SERDES Status Register 3 (QD_27)

Bit	Name	Description	R/W	Int?
7:6	Reserved			
5:0	PLL_CALIB_ STATUS[5:0]	TxPLL VCO calibration status output	RO	N

Table 18-67. SERDES Status Register 4 (QD_28)

Bit	Name	Description	R/W	Int?
7:0	Reserved			

Per Channel Register Overview

Table 18-68. Channel Interface Register Map

BA[1]	Register Name	D7	D6	D5	D4	D3	D2	D1	D0
1. CONTROL REGISTERS (13)									
Per Channel General Registers (7)									
00	ch_00	enable_cg_align	prbs_enable[6]	prbs_lock	ge_an_enable			invert_tx	invert_rx
01	ch_01	pfifo_clr	pcie_ei_en	pcs_det_time_sel[1]	pcs_det_time_sel[0]	rx_gear_mode	tx_gear_mode	rx_ch	tx_ch
02	ch_02	bus_width8	sb_bypass	sb_pfifo_lp	sb_bist_sel	enc_bypass	sel_bist_txd4enc	tx_gear_bypass	fb_loopback
03	ch_03	lsm_sel	signal_detect	rx_gear_bypass	ctc_bypass	dec_bypass	wa_bypass	rx_sb_bypass	sb_loopback
04	ch_int_04[2]					cc_underrun_int_ctl	cc_overrun_int_ctl	fb_rx_fifo_error_int_ctl	fb_tx_fifo_error_int_ctl
05	ch_05								los_hi_sel
06	ch_06								
Per Channel SERDES Registers (6)									
07	ch_07	req_en	req_lvl_set	rcv_dcc_en	rate_sel[1]	rate_sel[0]	rx_dco_ck_div[2]	rx_dco_ck_div[1]	rx_dco_ck_div[0]
08	ch_08	lb_ctl[3]	lb_ctl[2]	lb_ctl[1]	lb_ctl[0]	rterm_rxadj[1]	rterm_rxadj[0]	rterm_rx[1]	rterm_rx[0]
09	ch_09	tdrv_amp[2]	tdrv_amp[1]	tdrv_amp[0]	tdrv_pre_set[2]	tdrv_pre_set[1]	tdrv_pre_set[0]	tdrv_dat_sel[1]	tdrv_dat_sel[0]
0A	ch_0a				tdrv_pre_en	rterm_tx[1]	rterm_tx[0]	rate_mode_tx	tpwdnb
0B	ch_0b				oob_en	rx_refck_sel[1]	rx_refck_sel[0]	rate_mode_rx	rpwdnb
0C	ch_int_0c[2]		pci_det_done_int_ctl	rlos_lo_int_ctl	~rlos_lo_int_ctl	rlos_hi_int_ctl	~rlos_hi_int_ctl	rlol_int_ctl	~rlol_int_ctl
2. STATUS REGISTERS (13)									
Per Channel General Registers (6)									
20	ch_20[3]					cc_underrun	cc_overrun	fb_rx_fifo_error	fb_tx_fifo_error
21	ch_21[5]	prbs_error_cnt[7]	prbs_error_cnt[6]	prbs_error_count[5]	prbs_error_cnt[4]	prbs_error_cnt[3]	prbs_error_cnt[2]	prbs_error_cnt[1]	prbs_error_cnt[0]
22	ch_22								
23	ch_int_23	fb_tx_fifo_error_int	fb_rx_fifo_error_int			cc_underrun_int	cc_overrun_int	fb_rx_fifo_error_int	fb_tx_fifo_error_int
24	ch_24		ffs_ls_sync_status	fb_rxrst_o	fb_txrst_o			cc_re_o	cc_we_o
25	ch_25								
Per Channel SERDES Registers (7)									
26	ch_26[3]		pci_det_done	rlos_lo	~rlos_lo	rlos_hi	~rlos_hi	rlol	~rlol
27	ch_27	dco_calib_err	dco_calib_done	dco_facq_err	dco_facq_done				pci_connect
28	ch_28	dco_status[7]	dco_status[6]	dco_status[5]	dco_status[4]	dco_status[3]	dco_status[2]	dco_status[1]	dco_status[0]
29	ch_29	dco_status[15]	dco_status[14]	dco_status[13]	dco_status[12]	dco_status[11]	dco_status[10]	dco_status[9]	dco_status[8]
2A	ch_int_2a[4]		pci_det_done_int	rlos_lo_int	~rlos_lo_int	rlos_hi_int	~rlos_hi_int	rlol_int	~rlol_int
2B	ch_2b								
2C	ch_2c								

1. BA = Base Address (Hex)
2. Interrupt control register related to an interruptible status (int_sts_x) register.
3. Status register which has an associated interruptible status (int_sts_x) register.
4. Interruptible status register; clear on read (has associated control register and status register).
5. Status register with clear on read.
6. PRBS is generated by built-in logic and is for internal test only.

Note: Default value is “0”, unless otherwise specified.

Table 18-69. PCS Control Register 1 (CH_00)

Bit	Name	Description	R/W	Default
7	enable_cg_align	Only valid when operating in uc_mode 1 = Enable continuous comma alignment 0 = Disable continuous comma alignment	R/W	0
6	prbs_enable	1 = Enable PRBS generator & checker 0 = Normal operational mode.	R/W	0
5	prbs_lock	1 = Lock receive PRBS checker 0 = Unlock receive PRBS checker	R/W	0
4	ge_an_enable	1 = Enable GIGE Auto Negotiation 0 = Disable GIGE Auto Negotiation	R/W	0
3:2	Reserved			
1	invert_tx	1 = Invert transmitted data 0 = Do not invert transmitted data.	R/W	0
0	invert_rx	1 = Invert received data 0 = Do not invert received data.	R/W	0

Table 18-70. PCS Control Register 2 (CH_01)

Bit	Name	Description	R/W	Default
7	pfifo_clr	1 = Clears PFIFO if quad register bit pfifo_clr_sel is set to 1. This signal is ORed with interface signal pfifo_clr. 0 = Normal operation	R/W	0
6	pcie_ei_en	1 = PCI Express Electrical Idle 0 = Normal operation	R/W	0
5:4	pcs_det_time_sel[1:0]	PCS connection detection time 11 = 16us 10 = 4us 01 = 2us 00 = 8us	R/W	0
3	rx_gear_mode	1 = Enable 2:1 gearing for receive path on all channels 0 = Disable 2:1 gearing for receive path on all channels (no gearing)	R/W	0
2	tx_gear_mode	1 = Enable 2:1 gearing for transmit path on all channels 0 = Disable 2:1 gearing for transmit path on all channels (no gearing)	R/W	0
1	rx_ch	1 = Receive outputs can be monitored on the test characterization pins. The test characterization mode (bit 6 in pcs_ctl_4_qd_03) should be set to '1'.	R/W	0
0	tx_ch	1 = Transmit PCS inputs are sourced from test characterization ports. The test characterization mode should be enabled.	R/W	0

Table 18-71. PCS Control Register 3 (CH_02)

Bit	Name	Description	R/W	Default
7	bus_width8	1 = 8-bit bus between PCS and SER 0 = 10-bit bus between PCS and SER.	R/W	0
6	sb_bypass	1 = Bypass Tx SERDES Bridge 0 = Normal operation	R/W	0
5	sb_pfifo_lp	1 = Enable Parallel Loopback from Rx to Tx via parallel FIFO 0 = Normal data operation	R/W	0
4	sb_bist_sel	1 = Select BIST data 0 = Select Normal data	R/W	0
3	enc_bypass	1 = Bypass 8b10b encoder 0 = Normal operation	R/W	0
2	sel_bist_txd4enc	1 = Enable BIST data to Tx before 8b10b ENC 0 = Normal operation	R/W	0
1	tx_gear_bypass	1 = Bypass PCS Tx gear box 0 = Normal operation	R/W	0
0	fb_loopback	1 = Enable loopback in the PCS just before FPGA bridge from Rx to Tx. 0 = Normal data operation.	R/W	0

Table 18-72. PCS Control Register 4 (CH_03)

Bit	Name	Description	R/W	Default
7	lsm_sel	1 = selects External LSM for Word Alignment 0 = selects Internal LSM for Word Alignment	R/W	0
6	signal_detect	1 = force enabling the Rx link state machine 0 = dependent of ffc_signal_detect to enable the Rx link state machine	R/W	0
5	rx_gear_bypass	1 = Bypass PCS Rx gear box 0 = Normal operation	R/W	0
4	ctc_bypass	1 = Bypass clock toleration compensation 0 = Normal operation	R/W	0
3	dec_bypass	1 = Bypass 8b10b decoder 0 = Normal operation	R/W	0
2	wa_bypass	1 = Bypass word alignment 0 = Normal operation	R/W	0
1	rx_sb_bypass	1 = Bypass Rx SERDES Bridge 0 = Normal operation	R/W	0
0	sb_loopback	1 = Enable loopback in the PCS from Tx to Rx in SERDES bridge. 0 = Normal data operation.	R/W	0

Table 18-73. PCS Interrupt Control Register 5 (CH_04)

Bit	Name	Description	R/W	Default
7:4	Reserved			
3	cc_underrun_int_ctl	1 = Enable interrupt for cc_underrun 0 = Disable interrupt for cc_underrun.	RW	0
2	cc_overrun_int_ctl	1 = Enable interrupt for cc_overrun 0 = Disable interrupt for cc_overrun.	RW	0
1	fb_rx_fifo_error_int_ctl	1 = Enable interrupt on empty/full condition in the receive FPGA bridge FIFO.	R/W	0
0	fb_tx_fifo_error_int_ctl	1 = Enable interrupt on empty/full condition in the transmit FPGA bridge FIFO.	R/W	0

Table 18-74. PCS Control Register 6 (CH_05)

Bit	Name	Description	R/W	Default
7:1	Reserved			
0	low_hi_sel	1 = Select rlos_hi 0 = Select rlos_lo (this option is for internal use only)	R/W	0

Table 18-75. PCS Control Register 7 (CH_06)

Bit	Name	Description	R/W	Default
7:0	Reserved			

Per Channel SERDES Control Register Details

Note: Except indicated, all channels must be reset after writing any SERDES control register.

Table 18-76. SERDES Control Register 1 (CH_07)

Bit	Name	Description	R/W	Default
7	REQ_EN	1 = Receiver equalization enable 0 = Receiver equalization disable	R/W	0
6	REQ_LVL_SET	Level setting for equalization 1 = long-reach equalization 0 = mid-length route equalization	R/W	0
5	RCV_DCC_EN	1 = Receiver DC coupling enable. 0 = AC coupling (default)	R/W	0
4:3	RATE_SEL [1:0]	Equalizer pole position select: 00 = pole position for high frequency range 01 = pole position for medium frequency range 10 = pole position for low frequency range 11 = not used	R/W	2'b00
2:0	RX_DCO_CK_DIV[2:0]	VCO output frequency select: 000 = divided by 1 001 = reserved 010 = divided by 2 011 = reserved 100 = divided by 4 101 = divided by 8 110 = divided by 16 111 = divided by 32	R/W	3'b000

Table 18-77. SERDES Control Register 2 (CH_08)

Bit	Name	Description	R/W	Default
7:4	LB_CTL [3:0]	Loop back control: [3] = slb_r2t_dat_en, serial rx to tx LB enable(CDR data) [2] = slb_r2t_ck_en, serial rx to tx LB enable(CDR clock) [1] = slb_eq2t_en, serial LB from equalizer to driver enable [0] = slb_t2r_en, serial tx_to rx LB enable	R/W	4'h0
3:2	RTERM_RXADJ [1:0]	Termination resistor compensation 00 = default 01= -7% 10= -14% 11= -20%	R/W	2'b00
1:0	RTERM_RX [1:0]	00 = 50 ohm 01 = 75 ohm 10 = 2K ohm 11 = 60 ohm	R/W	2'b00

Table 18-78. SERDES Control Register 3 (CH_09)

Bit	Name	Description			R/W	Default
SERDES Control Register 3(CH_09) for LatticeECP2M-35						
7:5	TDRV_AMP[2:0]	CML driver amplitude setting (mV), VCCOB = 1.2V 000 = 1040(default) 001 = 1280 010 = 1320 011 = 1360 100 = 640 101 = 760 110 = 870 111 = 990			R/W	0
4:2	TDRV_PRE_SET[2:0]	Tx_driver pre-emphasis level setting 0 1 2 3 4 5 6	% 0 16 36 40 44 56 80		R/W	0
1:0	TDRV_DAT_SEL[1:0]	Driver output select: 00 = data from Serializer muxed to driver (normal operation) 01 = data rate clock from Serializer muxed to driver 10 = serial Rx to Tx LB (data) if slb_r2t_dat_en='1' 10 = serial Rx to Tx LB (clock) if slb_r2t_ck_en='1' 11 = serial LB from equalizer to driver if slb_eq2t_en='1'			R/W	0
SERDES Control Register 3(CH_09) for LatticeECP2M-20/50/70/100						
7:5	TDRV_AMP[2:0]	See Table 18-81			R/W	0
4.0	TDRV_PRE_SET[4:0]	Tx_driver pre-emphasis level setting			R/W	0
		3-bit setting in GUI 0 1 2 3 4 5 6	5-bit setting in register 00000 00001 00010 01010 10010 00011 00100	% 0 12 26 30 33 40 53		

Table 18-79. SERDES Control Register 4 (CH_0A)

Bit	Name	Description	R/W	Default
7	Reserved			
6:5	TDRV_DAT_SEL[1:0][1]	Driver output select: 00 = data from Serializer muxed to driver (normal operation) 01 = data rate clock from Serialzer muxed to driver 10 = serial Rx to Tx LB (data) if slb_r2t_data_en='1' 10 = serial Rx to Tx LB (clock) if slb_r2t_ck_en='1' 11 = serial LB from equalizer to driver if slb_eq2t_en='1'	R/W	0
4	TDRV_PRE_EN	1 = Tx driver pre-emphasis enable 0 = Tx driver pre-emphasis disable.	R/W	0
3:2	RTERM_TX [1:0]	Tx resistor termination select. Disabled when PCI Express feature is enabled 00 = 50 ohm (default) 01 = 75 ohm 10 = 5K ohm	R/W	2'b00
1	RATE_MODE_TX	0 = Full rate selection for transmit 1 = Half rate selection for transmit	R/W	0
0	tpwdnb	0 = Power down transmit channel 1= Power up transmit channel	R/W	0

1. For LatticeECP2M-20/50/70/100.

Table 18-80. SERDES Control Register 4 (CH_0B)

Bit	Name	Description	R/W	Default
TDRV_AMP Setting for LatticeECP2M-35				
7:6	TDRV_AMP[4:3]	See Table 18-81.	R/W	0
7:5	Reserved	For LatticeECP2M-35 only		
4	oob_en	1=Enables boundary scan input path for routing the high speed receive inputs to a lower speed SERDES in the FPGA (for out of band application).	R/W	0
3:2	rx_refck_sel [1:0]	Rx CDR Reference Clock Select 00 = refclk (differential input) 01 = core_rxrefclk 1x = reserved	R/W	2'b00
1	RATE_MODE_RX	0 = Full rate selection for receive 1 = Half rate selection for receive	R/W	0
0	rpwdnb	0 = Power down receive channel. 1= Power up receive channel	R/W	0

Table 18-81. TDRV_AMP Setting for LatticeECP2M-20/50/70/100

Bit	Name	Description			R/W	Default
CH_0B [7:6]	TDRV_AMP[4:3]	Tx_driver pre-emphasis level setting			R/W	0
CH_09 [7:5]	TDRV_AMP[2:0]	**3-bit setting in GUI**	**5-bit setting in register**	**mV**		
		0	00101	990		
		1	00001	1250		
		2	00010	1300		
		3	00011	1350		
		4	11100	610		
		5	10100	730		
		6	10110	820		
		7	01110	940		

Table 18-82. SERDES Interrupt Control Register 1 (CH_0C)

Bit	Name	Description	R/W	Default
7	Reserved			
6	pci_det_done_int_ctl	1 = Enable interrupt for detection of far-end receiver for PCI Express	R/W	0
5	rlos_lo_int_ctl	1 = Enable interrupt for Rx Loss of Signal when input levels fall below the programmed LOW threshold (using rlos_lset)	RW	0
4	~rlos_lo_int_ctl	1 = Enable interrupt for receiver Rx Loss of Signal when input level meets or is greater than programmed LOW threshold	RW	0
3	rlos_hi_int_ctl	1 = Enable interrupt for Rx Loss of Signal when input levels fall below the programmed HIGH threshold (using rlos_lset)	RW	0
2	~rlos_hi_int_ctl	1 = Enable interrupt for Rx Loss of Signal when input level meets or is greater than programmed HIGH threshold	RW	0
1	rlol_int_ctl	1= Enable interrupt for receiver loss of lock	R/W	0
0	~rlol_int_ctl	1= Enable interrupt when receiver recovers from loss of lock	R/W	0

Per Channel PCS Status Register Details

Table 18-83. PCS Status Register 1 (CH_20)

Bit	Name	Description	R/W	Int?
7:5	Reserved			
4	pfifo_error	1 = Parallel FIFO error 0 = No Parallel FIFO error	RO	Y
3	cc_underrun	1 = CC FIFO underrun 0 = CC FIFO not underrun	RO	Y
2	cc_overrun	1 = CC FIFO overrun 0 = CC FIFO not overrun	RO	Y
1	fb_rx_fifo_error	1 = FPGA bridge (FB) Rx FIFO overrun 0 = FB Rx FIFO not overrun	RO	Y
0	fb_tx_fifo_error	1 = FPGA bridge (FB) Tx FIFO overrun 0 = FB Tx FIFO not overrun	RO	Y

Table 18-84. PCS Status Register 2 (CH_21)

Bit	Name	Description	R/W	Int?
7:0	prbs_errors	Count of the number of PRBS errors. Clears to zero on read. Sticks at FF.	RO CR	N

Table 18-85. PCS Status Register 3 (CH_22)

Bit	Name	Description	R/W	Int?
7:0	Reserved			

Table 18-86. PCS General Interrupt Status Register 4 (CH_23)

Bit	Name	Description	R/W	Int?
7:4	Reserved			
3	cc_underrun_int	1 = Interrupt generated on cc_underrun 0 = Interrupt not generated on cc_underrun	RO CR	Y
2	cc_overrun_int	1 = Interrupt generated on cc_overrun 0 = Interrupt not generated on cc_overrun	RO CR	Y
1	fb_rx_fifo_error_int	1 = Interrupt generated on fb_rx_fifo_error. 0 = Interrupt not generated fb_rx_fifo_error.	RO CR	Y
0	fb_tx_fifo_error_int	1 = Interrupt generated on fb_tx_fifo_error 0 = Interrupt not generated fb_tx_fifo_error.	RO CR	Y

Table 18-87. PCS Status Register 5 (CH_24)

Bit	Name	Description	R/W	Int?
7	Reserved			
6	ffs_ls_sync_status	1 = Sync in the link state machine. 0 = Not sync in the LSM.	RO	N
5	fb_rxrst_o	1 = Normal operation 0 = FPGA bridge Rx reset	RO	N
4	fb_txrst_o	1 = Normal operation 0 = FPGA bridge Tx reset	RO	N
3	Reserved			
2	Reserved			
1	cc_re_o	1 = Elastic FIFO read enable 0 = Elastic FIFO read disable	RO	N
0	cc_we_o	1 = Elastic FIFO write enable 0 = Elastic FIFO write disable	RO	N

Table 18-88. PCS Status Register 6 (CH_25)

Bit	Name	Description	R/W	Int?
7:0	Reserved			

Per Channel SERDES Status Register Details

Table 18-89. SERDES Status Register 1 (CH_26)

Bit	Name	Description	R/W	Int?
7	Reserved			
6	pci_det_done	1 = Receiver detection process completed by SERDES transmitter. 0 = Receiver detection process not completed by SERDES transmitter.	RO CR	Y
5	rlos_lo	1= Indicates that the input signal detected by receiver is below the programmed LOW threshold	RO CR	Y
4	~rlos_lo	1= Indicates that the input signal detected by receiver is greater than or equal to the programmed LOW threshold	RO CR	Y
3	rlos_hi	1= Indicates that the input signal detected by receiver is below the programmed HIGH threshold	RO CR	Y
2	~rlos_hi	1= Indicates that the input signal detected by receiver is greater than or equal to the programmed HIGH threshold	RO CR	Y
1	rlol	1=Indicates CDR loss of lock to data. CDR is locked to reference clock.	RO	Y
0	~rlol	1 = Indicates that CDR has locked to data.	RO	Y

Table 18-90. SERDES Status Register 2 (CH_27)

Bit	Name	Description	R/W	Int?
7	DCO_CALIB_ERR	1 = indicates DCO calibration might be wrong (L/H boundary band selected)	RO	N
6	DCO_CALIB_DONE	1 = indicates DCO calibration done	RO	N
5	DCO_FACQ_ERR	1 = indicates DCO frequency acquisition error (>300ppm)	RO	N
4	DCO_FACQ_DONE	1 = indicates DCO frequency acquisition done	RO	N
3:1	Reserved			
0	pci_connect	1 = Receiver detected by SERDES transmitter (at the transmitter device). 0 = Receiver not detected by SERDES transmitter (at the transmitter device).	RO	N

Table 18-91. SERDES Status Register 3 (CH_28)

Bit	Name	Description	R/W	Int?
7:0	DCO_STATUS[7:0]	setdcoidac[7:0]	RO	N

Table 18-92. SERDES Status Register 4 (CH_29)

Bit	Name	Description	R/W	Int?
7:0	DCO_STATUS[15:8]	[1:0] = setdcoidac[9:8][7:2] = setdcoband[5:0]	RO	N

Table 18-93. SERDES Interrupt Status Register 5 (CH_2A)

Bit	Name	Description	R/W	Int?
7	Reserved			
6	pci_det_done_int	1 = Interrupt generated for pci_det_done	RO CR	Y
5	rlos_lo_int	1 = Interrupt generated for rlos_lo	RO CR	Y
4	~rlos_lo_int	1 = Interrupt generated for ~rlos_lo	RO CR	Y
3	rlos_hi_int	1 = Interrupt generated for rlos_hi	CO CR	Y
2	~rlos_hi_int	1 = Interrupt generated for ~rlos_hi	CO CR	Y
1	rlol_int	1 = Interrupt generated for rlol	CO CR	Y
0	~rlol_int	1 = Interrupt generated for ~rlol	CO CR	Y

Table 18-94. SERDES Status Register 6 (CH_2B)

Bit	Name	Description	R/W	Default
7:0	Reserved			

Table 18-95. SERDES Status Register 7 (CH_2C)

Bit	Name	Description	R/W	Default
7:0	Reserved			

Table 18-96. K Characters Recognized for Different Standards

Character	Gbe	XAUI	1xFC	PCI Express	RapidIO
K23.7 (F7)	Carrier extend			PAD	
K27.7 (FB)	SOP	ST		Start TLP	A (align)
K28.0 (1C)		SKIP R		SKIP	SC
K28.1 (3C)				FTS	
K28.2 (5C)		SoS		Start DLLP	
K28.3 (7C)		ALIGN A		IDLE	PD
K28.4 (9C)		SEQ			
K28.5 (BC)	+D5.6 or D16.2 = IDLE	SYNC K	+D21.4 +D21.5 +D21.5 = IDLE	COMMA (used for alignment)	K
K28.6 (DC)					
K28.7 (FC)					
K29.7 (FD)	EOP	T		END	R (skip)
K30.7 (FE)	ERR	ERR		END BAD	

Note: Refer to each standard specification for detailed information.

Appendix B. 8b10b Symbol Codes

Table 18-97. 8b10b Symbol Codes

Symbol Name (Mode Combination Table)	Symbol Code (8b10b Code)	10-Bit GUI Representation
K28.0	8`b000_11100	0100011100
K28.5	8`b101_11100	0110111100
K29.7	8`b111_11101	0011111101
D16.2	8`b010_10000	0001010000
D21.4	8`b100_10101	0010010101
D21.5	8`b101_10101	0010110101
K28.5+	10`b110000_0101	1100000101
K28.5-	10`b001111_1010	0011111010

Table 18-98. Lattice Mask for Symbol Code

Symbol Name (Mode Combination Table)	Symbol Code (8b10b Code)	10-Bit GUI Representation
28.5 (Mask)	10`b111111_1111	1111111111

Appendix C. Attribute Cross-Reference Table

Table 18-99. Attribute Cross-Reference Table

Independent Attribute Name	Dependent Attribute Names	Attribute Value	Register Map[3]
PROTOCOL	[QAUD_MODE, CHn_RX_DET, CHn_OOB_EN, CHn_GE_AN_EN]	GIGE : [00000,00,0,1] PCIE : [00100,00.0,0] G8B10B : [00001,00,0,0] 10BSER : [00001,00,0,0] 8BSER : [00001,00,0,0] SDSDI :[00001,00,1,0] HDSDI :[00001,00,0,0]	{QD_00[4:0], CH_01[5:4], CH_0B[4], CH_00[4]}
CH[0,1,2,3]_MODE	[CHn_TXPWDNB, CHn_RXPWDNB]	SINGLE : [11] GROUP1 : [11] GROUP2 : [11] DISABLE: [00]	{CH_0A[0], CH_0B[0]}
DATARANGE (ECP2M35 ES Device)	[PLL_DIV, CHn_CDR_DIV]	LOW : [101,101] MEDLOW : [100,100] MED : [010,010] MEDHIGH : [010,000] HIGH : [000,000]	{QD_14[2:0], CH_07[2:0]}
DATARANGE (All other Devices)		LOW : [101,101] MEDLOW : [100,100] MED : [010,010] MEDHIGH : [000,000] HIGH : [000,000]	
CH[0,1,2,3]_REFCK_MULT	[BUS8BIT_SEL, REFCK25X, REFCK_MODE, CHn_RATE_MODE_RX, CHn_RATE_MODE_TX]	8X : [1,0,1,0,0] 8XH : [1,0,1,1,1] 10X : [0,0,1,0,0] 10XH : [0,0,1,1,1] 16X : [1,0,0,0,0] 16XH : [1,0,0,1,1] 20X : [0,0,0,0,0] 20XH : [0,0,0,1,1] 25X : [0,1,0,0,0] 25XH : [0,1,0,1,1]	{QD_12[6], QD_12[7], QD_13[6], CH_0B[1], CH_0A[1]}
CH[0,1,2,3]_DATA_WIDTH	[TXCLKF, TXCLKH, CHn_RXCLKF, CHn_RXCLKH, CHn_TX_GEAR, CHn_RX_GEAR, CHn_BUS_WIDTH8]	8,BYPASS :[0,1,0000,0000,0,0,1] 8, NORMAL; 10, BYPASS; 10, NORMAL :[0,1,0000.0000,0,0,0] 16,BYPASS :[01,1111,1111,1,1,1] 16,NORMAL;20,BYPASS;20,NORMAL :[0,1,1111,1111,1,1,0]	{QD_1B[7], QD_1B[6], QD_1A[7:4], QD_1A[3:0], CH_01[2], CH_01[3], CH_02[7]}
PLL_SRC		REFCLK :[0] CORE_TXREFCLK :[1]	{QD_11[5]}
CH[0,1,2,3]_CDR_SRC		REFCLK :[00] CORE_RXREFCLK :[01]	{CH_0B[3:2]}

Table 18-99. Attribute Cross-Reference Table (Continued)

Independent Attribute Name	Dependent Attribute Names	Attribute Value	Register Map[3]
CH_[0,1,2,3]_TRDV_AMP (ECP2M35 all Device)		0:[000] 1:[001] 2:[010] 3:[011] 4:[100] 5:[101] 6:[110] 7:[111]	CH_09[7:5]
CH_[0,1,2,3]_TRDV_AMP (All other Devices)		0:[00101] 1:[00001] 2:[00010] 3:[00011] 4:[11100] 5:[10100] 6:[10110] 7:[01110]	CH_0B[7:6] CH_09[7:5]
CH_[0,1,2,3]_TX_PRE[5] (ECP2M35 ES Device)	[CHn_TRDV_PRE_EN, CHn_TRDV_PRE_SET]	DISABLE:[0,000] 0:[1,000] 1:[1,001] 2:[1,010] 3:[1,011] 4:[1,100] 5:[1,101]	{CH_0A[4], CH_09[4:2]}
CH_[0,1,2,3]_TX_PRE[6] (ECP2M35 non-ES device)		DISABLE:[0,000] 0:[1,000] 1:[1,001] 2:[1,010] 3:[1,011] 4:[1,100] 5:[1,101] 6:[1,110]	
CH_[0,1,2,3]_TX_PRE[6] (All other Devices)		DISABLE:[0,00000] 0:[1,00000] 1:[1,00001] 2:[1,00010] 3:[1,01010] 4:[1,10010] 5:[1,00011] 6:[1,00100]	{CH_0A[4], CH_09[4:0]}
CH[0,1,2,3]_RTERM_TX		50:[00] 75:[01] 5K:[10]	{CH_0A[3:2]}
CH[0,1,2,3]_RX_EQ	[CHn_REQ_EN, CHn_REQ_LVL_SET, CHn_RATE_SEL]	DISABLE :[0,0,00] MID_LOW :[1,0,10] MID_MED :[1,0,01] MID_HIGH :[1,0,00] LONG_LOW :[1,1,10] LONG_MED :[1,1,01] LONG_HIGH :[1,1,00]	{CH_07[7], CH_07[6], CH_07[4:3]}
CH[0,1,2,3]_RTERM_RX	[CHn_RX_RTERM]	50 :[00] 60 :[11] 75 :[01] HIGH[4] :[10]	{CH_08[1:0]}
CH[0,1,2,3]_RX_DCC		AC:[0] DC:[1]	{CH_07[5]}

Table 18-99. Attribute Cross-Reference Table (Continued)

Independent Attribute Name	Dependent Attribute Names	Attribute Value	Register Map[3]
LOS_THRESHOLD (ECP2M35 ES device)	[RLOS_LO, RLOS_HI]	0:[000,000] 1:[001,000] 2:[010,000] 3:[011,000] 4:[100,000] 5:[101,000] 6:[110,000] 7:[111,000]	{QD_12[2:0], QD_12[5:3]}
LOS_THRESHOLD (All other devices)	[LOS_HI_SEL, RLOS_LO, RLOS_HI]	0:[1111,000,000] 1:[1111,000,001] 2:[1111,000,010] 3:[1111,000,011] 4:[1111,000,100] 5:[1111,000,101] 6:[1111,000,110] 7:[1111,000,111]	{CH_05[0], QD_12[2:0], QD_12[5:3]}
PLL_TERM		50:[0] 2K:[1]	{QD_11[3]}
PLL_DCC		AC:[0] DC:[1]	{QD_11[4]}
PLL_LOL_SET		0:[00] 1:[01] 2:[10] 3:[11]	{QD_14[4:3]}
CH[0,1,2,3]_TX_SB	[CHn_TXPOL, CHn_TXSBBYP]	NORMAL: [0,0] INV : [1,0]	{CH_00[1], CH_02[6]}
CH[0,1,2,3]_RX_SB	[CHn_RXPOL, CHn_RXWSBBYP]	NORMAL: [0,0] INV : [1,0]	{CH_00[0], CH_03[1]}
CH[0,1,2,3]_8B10B	[CHn_TXENC, CHn_RXDEC]	NORMAL: [0,0] BYPASS: [1,1]	{CH_02[3], CH_03[3]}
COMMA_A		Note 1	{QD_0A[0:7], QD_0C[6:7]}
COMMA_B		Note 1	{QD_0B[0:7], QD_0C[4:5]}
COMMA_M		Note 1	{QD_09[0:7], QD_0C[2:3]}
CH[0,1,2,3]_COMMA_ALIGN	[CHn_RXWA, CHn_LSM_SEL, CHn_C_ALIGN, CHn_SIG_DET]	AUTO : [0,0,0,1] DYNAMIC : [0,1,0,0] BYPASS : [1,1,0,0]	{CH_03[2], CH_03[7], CH_00[7], CH_03[6]}
CH[0,1,2,3]_CTC_BYP	[CHn_RXRECCLK]	NORMAL : [0,0,0,0,0] BYPASS : [1,1,1,1,1]	{QD_19[3:0], CH_03[4]}
CC_MATCH1			{QD_04[7:0], QD_08[1:0]}
CC_MATCH2			{QD_05[7:0], QD_08[3:2]}
CC_MATCH3			{QD_06[7:0], QD_08[5:4]}
CC_MATCH4			{QD_07[7:0], QD_08[7:6]}
CC_MATCH_MODE	[MATCH_2_EN, MATCH_4_EN]	MATCH_3_4 : [1,0] MATCH_4 : [0,0] MATCH_1_2_3_4 : [0,1]	{QD_03[4], QD_03[5]}

Table 18-99. Attribute Cross-Reference Table (Continued)

Independent Attribute Name	Dependent Attribute Names	Attribute Value	Register Map[3]
CC_MIN_IPG		0: [00] 1: [01] 2: [10] 3: [11]	{QD_03[7:6]}
CCHMARK		0:[0000] 1:[0001] 2:[0010] 3:[0011] 4:[0100] 5:[0101] 6:[0110] 7:[0111] 8:[1000] 9:[1001] 10:[1010] 11:[1011] 12:[1100] 13:[1101] 14:[1110] 15:[1111]	{QD_02[7:4]}
CCLMARK		0:[0000] 1:[0001] 2:[0010] 3:[0011] 4:[0100] 5:[0101] 6:[0110] 7:[0111] 8:[1000] 9:[1001] 10:[1010] 11:[1011] 12:[1100] 13:[1101] 14:[1110] 15:[1111]	{QD_02[3:0]}
[PLL_SRC, CHn_CDR_SRC]	[REFCKLOCAL]	[CORE_TXREFCLK, CORE_RXREFCLK, CORE_RXREFCLK, CORE_RXREFCLK, CORE_RXREFCLK]:[1] [REFCLK,X,X,X,X]:[0]note2 [X,REFCLK,X,X,X]:[0] [X,X,REFCLK,X,X]:[0] [X,X,X,REFCLK,X]:[0] [X,X,X,X,REFCLK]:[0]	{QD_11[0]}
OS_SSLB	[CHn_EQ2T_EN]	0:[0] 1:[1]	{CH_08[5]}
OS_SPLBPORTS	[PFIFO_CLR_SEL, CHn_SB_PFIFO_LP]	0:[0,0] 1:[1,1]	{QD_03[2], CH_02[5]}
OS_PCSLBPORTS	[FB_LOOPBACK]	0:[0] 1:[1]	{CH_02[0]}
OS_REFCK2CORE	[REFCK2CORE]	0:[0] 1:[1]	{QD_11[1]}
OS_PLLQCLKPORTS	[TXCLKQ, CHn_RXCLKQ]	0:[0,0000] 1:[1,1111]	{QD_QB[5], QD_1B[3:0]}

Table 18-99. Attribute Cross-Reference Table (Continued)

Independent Attribute Name	Dependent Attribute Names	Attribute Value	Register Map[3]
OS_INT_ALL	[PLOL_INT, PLOLN_INT, CHn_PCIDETINT, CHn_RLOSLINT, CHn_RLOSLNINT, CHn_RLOSHINT, CHn_RLOSHNINT, CHn-RLOLINT, CHn_RLOLNINT, CHn_LSSYNCINT, CHn_LSSYNCNINT, CHn_TXFIFOINT, CHn_RXFIFOINT, CHn_CCORUNINT, CHn_CCURUNINT]	0:[0,0,0,0,0,0,0,0,0,0000, 0000,0,0,0,0] 1:[1,1,1,1,1,1,1,1,1,1111, 1111,1,1,1,1]	{QD_16[7], QD_16[6], QD_0C[1], QD_0C[2], QD_0C[3], QD_0C[4], QD_0C[5], QD_0C[6], QD_0C[7], QD_10[7:4], QD_10[3:0], CH_04[0], CH_04[1], CH_04[2], CH_04[3]}

1. 10-bit symbol code default value or specified by user in the ‘PCS Advanced setup’ configuration GUI. Since the 10-bit symbol code representation in the GUI is LSB to MSB, the bit representation is swapped appropriately in the table.
2. X = Don’t care.
3. QD_xx : Quad Register with Base Address = xx
   CH_yy : Channel Register with Base Address = yy
4. HIGH = 2K for LatticeECP2M35 ES devices and infinity for all other devices.
5. 0: 0%; 1: 16%; 2: 32%; 3:48%; 4:64%; 5:80%.
6. 0: 0%; 1: 16%; 2: 36%; 3:40%; 4:44%; 5:56%; 6:80%.

Appendix D. Protocol Specific SERDES Setup Options

Table 18-100. Protocol Specific SERDES Setup Options

Protocol	DATARATE	DATARATE Range	REFCK Multiplier	DATA WIDTH	Rx Equalization[1]
GbE	1.25	MED	20x, 10x, 5x	8, 16	DISABLE, MID_MED, LONG_MED
PCI Express	2.5	HIGH	25x, 20x		DISABLE, MID_HIGH, LONG_HIGH
XAUI	3.125	HIGH	20x	16	
Generic 8b10b	ANY_VALUE	LOW, MEDLOW, MED, MEDHIGH, HIGH	20x, 10x, 5x	8, 16	DISABLE, MID_LOW, MID_MED, MID_HIGH, LONG_LOW, LONG_MED, LONG_HIGH
8-bit SERDES_Only			16x, 8x, 4x		
10-bit SERDES_Only			20x, 10x, 5x	10, 20	

1. MID: about 20" in length
LONG: about 40" in length
LOW: less than 1.2 Gbps
MED: between 1.2 Gbps and 2 Gbps
HIGH: over 2 Gbps

Appendix E. Lattice Diamond Usage Overview

This appendix discusses the use of Lattice Diamond design software for projects that include the LatticeECP2M SERDES/PCS module .

For general information about the use of Lattice Diamond, refer to the Lattice Diamond Tutorial.

If you have been using ispLEVER software for your FPGA design projects, Lattice Diamond may look like a big change. But if you look closer, you will find many similarities because Lattice Diamond is based on the same toolset and work flow as ispLEVER. The changes are intended to provide a simpler, more integrated, and more enhanced user interface.

Converting an ispLEVER Project to Lattice Diamond

Design projects created in ispLEVER can easily be imported into Lattice Diamond. The process is automatic except for the ispLEVER process properties, which are similar to the Diamond strategy settings, and PCS modules. After importing a project, you need to set up a strategy for it and regenerate any PCS modules.

Importing an ispLEVER Design Project

Make a backup copy of the ispLEVER project or make a new copy that will become the Diamond project.

1. In Diamond, choose **File > Open > Import ispLEVER Project**.
2. In the ispLEVER Project dialog box, browse to the project's .syn file and open it.
3. If desired, change the base file name or location for the Diamond project. If you change the location, the new Diamond files will go into the new location, but the original source files will not move or be copied. The Diamond project will reference the source files in the original location.

The project files are converted to Diamond format with the default strategy settings.

Adjusting PCS Modules

PCS modules created with IPexpress have an unusual file structure and need additional adjustment when importing a project from ispLEVER. There are two ways to do this adjustment. The preferred method is to regenerate the module in Diamond. However this may upgrade the module to a more recent version. An upgrade is usually desirable but if, for some reason, you do not want to upgrade the PCS module, you can manually adjust the module by copying its .txt file into the implementation folder. If you use this method, you must remember to copy the .txt file into any future implementation folders.

Regenerate PCS Modules

1. Find the PCS module in the Input Files folder of File List view. The module may be represented by an .lpc, .v, or .vhd file.
2. If the File List view shows the Verilog or VHDL file for the module, and you want to regenerate the module, import the module's .lpc file:
   a. In the File List view, right-click the implementation folder () and choose **Add > Existing File**.
   b. Browse for the module's .lpc file, ***<module_name>*.lpc**, and select it.
   c. Click **Add**. The .lpc file is added to the File List view.
   d. Right-click the module's Verilog or VHDL file and choose **Remove**.
3. In File List, double-click the module's .lpc file. The module's IPexpress dialog box opens.
4. In the bottom of the dialog box, click **Generate**. The Generate Log tab is displayed. Check for errors and close.

In File List, the .lpc file is replaced with an .ipx file. The IPexpress manifest (.ipx) file is new with Diamond. The .ipx file keeps track of the files needed for complex modules.

Using IPexpress with Lattice Diamond

Using IPexpress with Lattice Diamond is essentially same as with ispLEVER.

The configuration GUI tabs are all the same except for the Generation Options tab. Figure 18-48 shows the Generation Options tab window.

Figure 18-48. Generation Options Tab

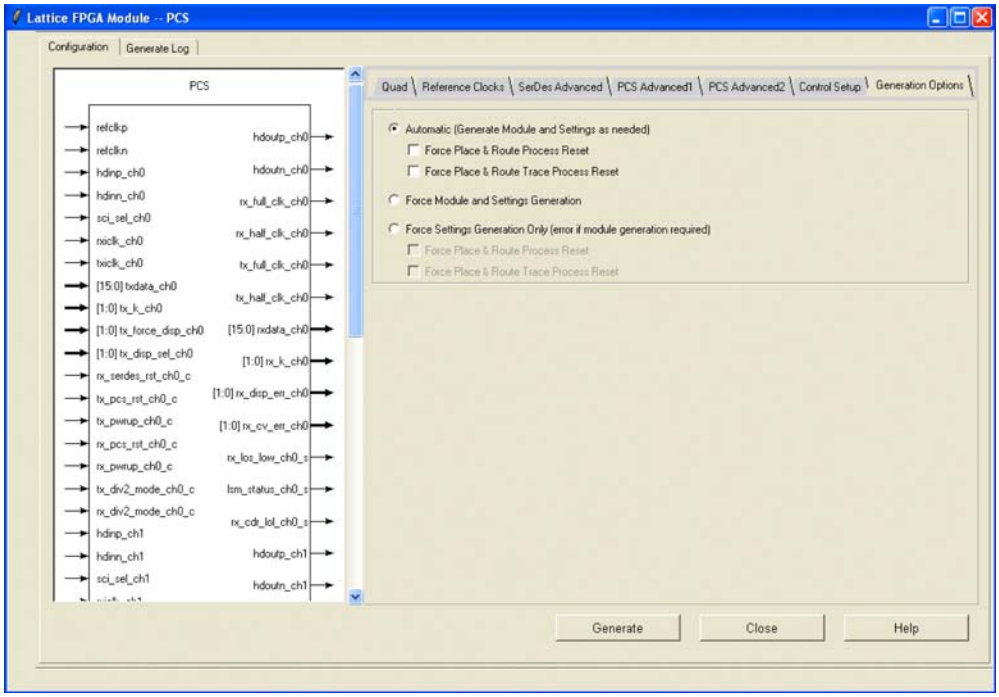


Table 18-101. SERDES_PCS GUI Attributes – Generation Options Tab

GUI Text	Description
Automatic	Automatically generates the HDL and configuration(.txt) files as needed. Some changes do not require regenerating both files.
Force Module and Settings Generation	Generates both the HDL and configuration files.
Force Settings Generation Only	Generates only the attributes file. You get an error message if the HDL file also needs to be generated.
Force Place & Route Process Reset	Resets the Place & Route Design process, forcing it to be run again with the newly generated PCS module.
Force Place & Route Trace Process Reset	Resets the Place & Route Trace process, forcing it to be run again with the newly generated PCS module.

Note:
Automatic is set as the default option. If either Automatic or Force Settings Generation Only and no sub-options (Process Reset Options) are checked and the HDL module is not generated, the reset pointer is set to Bitstream generation automatically.

After the Generation is finished, the reset marks in the process window will be reset accordingly.

Creating a New Simulation Project Using Simulation Wizard

This section describes how to use the Simulation Wizard to create a simulation project (.spf) file so you can import it into a standalone simulator.

1. In Project Navigator, click **Tools > Simulation Wizard**. The Simulation Wizard opens.
2. In the Preparing the Simulator Interface page click **Next**.
3. In the Simulator Project Name page, enter the name of your project in the Project Name text box and browse to the file path location where you want to put your simulation project using the Project Location text box and Browse button.

   When you designate a project name in this wizard page, a corresponding folder will be created in the file path you choose. Click **Yes** in the popup dialog that asks you if you wish to create a new folder.
4. Click either the Active-HDL® or ModelSim® simulator check box and click **Next**.
5. In the Process Stage page choose which type of Process Stage of simulation project you wish to create Valid types are RTL, Post-Synthesis Gate-Level, Post-Map Gate-Level, and Post-Route Gate-level+Timing. Only those process stages that are available are activated.

   Note that you can make a new selection for the current strategy if you have more than one defined in your project.

   The software supports multiple strategies per project implementation which allow you to experiment with alternative optimization options across a common set of source files. Since each strategy may have been processed to different stages, this dialog allows you to specify which stage you wish to load.
6. In the Add Source page, select from the source files listed in the Source Files list box or use the browse button on the right to choose another desired source file. Note that if you wish to keep the source files in the local simulation project directory you just created, check the **Copy Source to Simulation Directory** option.
7. Click **Next** and a Summary page appears and provides information on the project selections including the simulation libraries. By default, the Run Simulator check box is enabled and will launch the simulation tool you chose earlier in the wizard in the Simulator Project Name page.
8. Click **Finish**.

The Simulation Wizard Project (.spf) file and a simulation script DO file are generated after running the wizard. You can import the DO file into your current project if desired. If you are using Active-HDL, the wizard will generate an .ado file and if you are using ModelSim, it creates and .mdo file.

Note: PCS configuration file, (.txt) must be added in step 6.

Revision History

Date	Version	Change Summary
July 2021	3.7	Changed document number from TN1124 to FPGA-TN-02254.
		Added Disclaimers section.
		Updated document number of Electrical Recommendations for Lattice SERDES to FPGA-TN-02077.
		Updated technical support information.
		Moved Revision History to end of document and revised order of changes.
June 2013	03.6	Updated document with new corporate logo.
		Updated Technical Support Assistance information.
September 2011	03.5	SERDES_PCI I/O Descriptions table – Updated description for ffc_macro_rst signal.
		SERDES/PCS Reset diagram – Replaced lowest OR gate with an AND gate.
		Reset Pulse Specification table – Units for $t_{MACRORST}$ changed from ns to µs.
		SERDES Control Register 3 (QD_13) table – Default for CDR_LOL_SET [1:0] bit changed from 2'b10 to 2'b00.
June 2010	03.4	Added Appendix E.
February 2010	03.3	SERDES/PCS reset sequence updated.
July 2009	03.2	ffc_txpwdnb description corrected. rlol settling time changed to 4ms.
May 2009	03.1	Word Aligner Latency value updated.
March 2009	03.0	Mode-specific control and status signals are now arranged in one table (Data Bus Usage by Mode table).
		In all user modes, CTC is bypassed by hardware (SERDES/PCS GUI - PCS Advanced Setup Tab table).
January 2009	02.9	Removed references to PIPE Mode.
November 2008	02.8	Detailed definitions of reset signals added.
October 2008	02.7	Updated Serializer/Deserializer Blocks latency
		Corrected footnote 5 in the SERDES_PCS I/O Descriptions table.
		Changed title of Per Channel Register Settings for Different Standards table to K Characters Recognized for Different Standards.
August 2008	02.6	Updated High Speed I/O Terminations figure.
May 2008	02.5	SCI Address Map updated.
		Clock usage example table updated.
		Control register CH_07[2:0] corrected.
February 2008	02.4	Added detailed Reset Sequence diagram.
December 2007	02.3	TDRV_DAT_SEL tables updated for ECP2M-35 and the rest of ECP2M families.
September 2007	02.2	Updated Supported SERDES Standards table.
		Added Simulation of the SERDES/PCS section.
August 2007	02.1	SD-SDI and HD-SDI are added in full-support list. Reset sequence, Simulation Consideration, 16-bit word alignment sections added. Plus various updates.
March 2007	02.0	Re-formatted and updated majority of the document to clarify the SERDES functionality in detail.
September 2006	01.0	Initial release.