# LatticeSC/M Family flexiPCS Data Sheet

DS1005 Version 02.0, June 2011

# Table of Contents
# LatticeSC/M Family flexiPCS Data Sheet

June 2011


**Introduction to flexiPCS ..... 1-1**
flexiPCS™ Features ..... 1-1
flexiPCS Introduction ..... 1-2
Architecture Overview ..... 1-2
flexiPCS Quad ..... 1-2
flexiPCS Channel ..... 1-3
Per Channel PCS/FPGA Interface Ports ..... 1-4
Using this Data Sheet ..... 1-7
**SERDES Functionality/Electrical & Timing ..... 2-1**
Introduction ..... 2-1
LatticeSC flexiPCS Quad Module ..... 2-1
Quad and Channel Option Control ..... 2-1
Register Map Addressing ..... 2-1
Auto-Configuration ..... 2-3
SERDES Functionality ..... 2-4
SERDES Port Description ..... 2-6
SERDES Functionality Detailed Description ..... 2-8
SERDES Powerup ..... 2-8
Clocks & Resets ..... 2-9
Rate Mode Registers ..... 2-12
Locked Reference and Transmit Clocks ..... 2-17
Transmit Clocks ..... 2-18
Receive Clocks ..... 2-18
Transmit Data ..... 2-22
Transmit Data Skew ..... 2-23
Receive Data ..... 2-23
Low Speed SERDES Receiver Operation ..... 2-26
Status Interrupt Registers ..... 2-27
Receive CDR Loss of Lock Interrupt Usage ..... 2-27
flexiPCS Reset Sequences ..... 2-28
flexiPCS Power Down Control Signals ..... 2-29
SERDES Electrical & Timing Characteristics ..... 2-31
SERDES Performance ..... 2-31
SERDES High Speed Data Transmitter ..... 2-31
SERDES High Speed Data Receiver ..... 2-33
Interfacing to Reference Clock CML Buffers ..... 2-36
SERDES Power ..... 2-37
SERDES Power Supply Sequencing Requirements ..... 2-38
SERDES Power Supply Requirements ..... 2-38
SERDES Power Supply Package Requirements ..... 2-38
Power-Down/Power-Up Specification ..... 2-38
**8-bit and 10-bit SERDES Only Modes ..... 3-1**
Introduction ..... 3-1
LatticeSC flexiPCS Quad Module ..... 3-1
Quad and Channel Option Control ..... 3-1
Register Map Addressing ..... 3-2
Auto-Configuration ..... 3-3
8-bit SERDES Only Register Settings ..... 3-4



© 2011 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

8-bit SERDES Only Auto-Configuration Example ..... 3-4
10-bit SERDES Only Register Settings ..... 3-5
10-bit SERDES Only Auto-Configuration Example ..... 3-6
8-bit SERDES Only Mode ..... 3-6
8-bit SERDES Only Mode Pin Description ..... 3-8
8-bit SERDES Only Mode Detailed Description ..... 3-9
Clocks & Resets ..... 3-9
Transmit Data ..... 3-11
Receive Data ..... 3-14
Control & Status ..... 3-16
10-bit SERDES Only Mode ..... 3-16
10-bit SERDES Only Mode Pin Description ..... 3-18
10-bit SERDES Only Mode Detailed Description ..... 3-19
Clocks & Resets ..... 3-19
Transmit Data ..... 3-21
Receive Data ..... 3-24
Control & Status ..... 3-26
**Generic 8b10b Mode ..... 4-1**
Introduction ..... 4-1
LatticeSC flexiPCS Quad Module ..... 4-1
Quad and Channel Option Control ..... 4-1
Register Map Addressing ..... 4-2
Auto-Configuration ..... 4-3
Generic 8b10b Register Settings ..... 4-4
Generic 8b10b Auto-Configuration Example ..... 4-5
Generic 8b10b Mode ..... 4-6
Generic 8b10b Mode Pin Description ..... 4-8
Generic 8b10b Mode Detailed Description ..... 4-10
Clocks & Resets ..... 4-10
Transmit Data ..... 4-12
Receive Data ..... 4-16
Control & Status ..... 4-32
Status Interrupt Registers ..... 4-33
**Gigabit Ethernet Mode ..... 5-1**
Introduction ..... 5-1
LatticeSC flexiPCS Quad Module ..... 5-1
Quad and Channel Option Control ..... 5-2
Register Map Addressing ..... 5-2
Auto-Configuration ..... 5-4
Gigabit Ethernet Register Settings ..... 5-4
Gigabit Ethernet Auto-Configuration ..... 5-5
Gigabit Ethernet Mode ..... 5-6
Gigabit Ethernet Mode Pin Description ..... 5-7
Gigabit Ethernet Mode Detailed Description ..... 5-9
Clocks & Resets ..... 5-9
Transmit Data ..... 5-10
Receive Data ..... 5-13
Auto-Negotiation ..... 5-22
Control & Status ..... 5-25
Status Interrupt Registers ..... 5-26
**Fibre Channel Mode ..... 6-1**
Introduction ..... 6-1
LatticeSC flexiPCS Quad Module ..... 6-1
Quad and Channel Option Control ..... 6-2

Register Map Addressing ..... 6-2
Auto-Configuration ..... 6-4
Fibre Channel Register Settings ..... 6-4
Fibre Channel Auto-Configuration Example ..... 6-5
Fibre Channel Mode ..... 6-6
Fibre Channel Mode Pin Description ..... 6-7
Fibre Channel Mode Detailed Description ..... 6-10
Clocks & Resets ..... 6-10
Transmit Data ..... 6-12
Receive Data ..... 6-14
Control & Status ..... 6-23
Status Interrupt Registers ..... 6-23
**XAUI Mode ..... 7-1**
Introduction ..... 7-1
LatticeSC flexiPCS Quad Module ..... 7-1
Quad and Channel Option Control ..... 7-1
Register Map Addressing ..... 7-2
Auto-Configuration ..... 7-3
XAUI Register Settings ..... 7-4
XAUI Auto-Configuration Example ..... 7-5
XAUI Mode ..... 7-6
XAUI Mode Pin Descriptions ..... 7-8
XAUI Mode Detailed Description ..... 7-9
Clocks & Resets ..... 7-9
Transmit Data ..... 7-11
Receive Data ..... 7-15
Control & Status ..... 7-25
Status Interrupt Registers ..... 7-26
**PCI Express Mode ..... 8-1**
Introduction ..... 8-1
LatticeSC flexiPCS Quad Module ..... 8-1
Quad and Channel Option Control ..... 8-2
Register Map Addressing ..... 8-2
Auto-Configuration ..... 8-4
PCI Express Register Settings ..... 8-5
x4 PCI Express Auto-Configuration Example ..... 8-6
PCI Express Mode ..... 8-7
PCI Express Mode Pin Description ..... 8-9
PCI Express Mode Detailed Description ..... 8-11
Clocks & Resets ..... 8-11
Transmit Data ..... 8-12
Receive Data ..... 8-16
Control & Status ..... 8-21
x2 PCI Express Operation ..... 8-22
x4 PCI Express Operation ..... 8-26
x4 PCI Express Operation Pin Description ..... 8-28
x4 PCI Express Operation Detailed Description ..... 8-31
Clocks & Resets ..... 8-31
Transmit Data ..... 8-31
Receive Data ..... 8-31
x8 PCI Express Operation ..... 8-35
Control & Status ..... 8-37
Status Interrupt Registers ..... 8-38

**Serial RapidIO Mode ..... 9-1**
Introduction ..... 9-1
LatticeSC flexiPCS Quad Module ..... 9-1
Quad and Channel Option Control ..... 9-2
Register Map Addressing ..... 9-2
Auto-Configuration ..... 9-3
Serial RapidIO Register Settings ..... 9-4
Serial RapidIO Auto-Configuration Example ..... 9-5
Serial RapidIO Mode ..... 9-6
1x Serial RapidIO Mode Pin Descriptions ..... 9-8
Serial RapidIO Mode Detailed Description ..... 9-9
Clocks & Resets ..... 9-9
Transmit Data ..... 9-12
Receive Data ..... 9-15
Control & Status ..... 9-19
4x Serial RapidIO Operation ..... 9-20
4x Serial RapidIO Operation Pin Description ..... 9-22
4x Serial RapidIO Operation Detailed Description ..... 9-23
Clocks & Resets ..... 9-23
Transmit Data ..... 9-24
Receive Data ..... 9-27
Control & Status ..... 9-32
Status Interrupt Registers ..... 9-33
**SONET Mode ..... 10-1**
Introduction ..... 10-1
LatticeSC flexiPCS Quad Module ..... 10-1
Quad and Channel Option Control ..... 10-1
Register Map Addressing ..... 10-2
Auto-Configuration ..... 10-3
SONET Register Settings ..... 10-4
SONET Auto-Configuration ..... 10-5
SONET Mode ..... 10-6
SONET Mode Pin Description ..... 10-8
SONET Mode Detailed Description ..... 10-9
Clocks & Resets ..... 10-9
Transmit Data ..... 10-11
Receive Data ..... 10-15
Control & Status ..... 10-25
Status Interrupt Registers ..... 10-26
**Multi-Channel Alignment ..... 11-1**
Introduction ..... 11-1
Supported Protocols ..... 11-2
Alignment Within A Quad ..... 11-4
Alignment Between Quads ..... 11-13
Alignment Between Two Devices ..... 11-19
Synchronization of Aligned Channels Across Two Devices ..... 11-20
Status Interrupt Registers ..... 11-21
**flexiPCS Testing ..... 12-1**
Introduction ..... 12-1
Near-End Loopback Modes ..... 12-1
Far-End Loopback Modes ..... 12-2
PRBS Generator and Checker ..... 12-4
**Memory Map ..... 13-1**
LatticeSC flexiPCS Memory Map ..... 13-1

**Revision History ..... 14-1**

# Introduction to flexiPCS
# LatticeSC/M flexiPCS Family Data Sheet

August 2007 Data Sheet DS1005

## flexiPCS™ Features

- **Up to 32 Channels of High-Speed SERDES**
  - 600 Mbps to 3.8 Gbps per channel
  - Out-of-band signal interface allows same pin operation down to DC rates
  - Low TX jitter (0.25 UI typical at 3.125Gbps)
  - High RX jitter tolerance (0.8 UI at 3.125Gbps)
  - Low Power (105mW per channel typical)
  - SERDES Only mode allows direct 8- or 10-bit interface to FPGA logic
- **Other Key Features**
  - Four levels of pre-emphasis up to 48%
  - Two levels of equalization up to 12dB
  - Four TX programmable amplitude levels
  - Three programmable termination levels (TX)
  - Four programmable termination levels (RX)
  - Hot-plug capable
  - AC or DC coupled receiver
- **Full Function Embedded Physical Coding Sub-layer (PCS) Logic Supporting Industry Standard Protocols**
  - Up to 32 Channels of full-duplex data supported per device
  - Multiple protocol support on one chip
  - Supports popular 8b10b based packet protocols
  - Supports SONET based payloads
- **Gigabit Ethernet**
  - IEEE 1000BASE-X compliant
  - 8b10b encoding/decoding
  - Access Clause 22 PHY registers in Auto-negotiation mode
  - Comma character word alignment
  - Clock Tolerance Compensation circuit
  - CRC generation/checking
  - Provides 1G Ethernet link to GMII compliant FPGA logic interface
- **10Gb Ethernet**
  - 10GbE /A/K/R/ idle insertion and removal
  - 10GbE synchronization state machine controls comma alignment
  - 10GbE XAUI deskew state machine controls multi-channel alignment and monitors alignment status
  - Clock Tolerance Compensation logic performs idle insertion and removal
  - Provides 10G Ethernet link to XGMII compliant FPGA logic interface
- **Fibre Channel**
  - Fibre Channel link state machine to report link status
  - Fibre Channel EOF ordered set conversion to correct disparity
  - 1.02/2.04 Gbps Fibre Channel support (single channel) and 10G Fibre Channel support (4 channels)
- **PCI Express**
  - Word aligner
  - 8b10b encoding/decoding
  - Clock Tolerance Compensation circuit
  - PCI Express data scrambling and descrambling (Both D1.0 and D1.0a polynomials)
  - Multi-channel alignment for support of x1 to x32 PCI Express on one device
  - Electrical Idle and Receiver Detection support
- **Serial RapidIO**
  - Word Aligner
  - 8b10b encoding/decoding
  - Clock Tolerance Compensation circuit
  - Random /A/K/R/ insertion in transmit path and idle replacement in receive path
  - Multi-channel alignment for support of 1x to 32x RapidIO on one device
- **SONET Based Functionality**
  - STS-48 and STS-12 Framers
  - Supports any length of concatenation within an STS-12/STS-12c or STS-48/STS-48c frame
  - TOH byte insertion of A1, A2, and Section B1 bytes into transmitted data frames
  - SONET compliant scrambling and descrambling
  - B1 checking, AIS insertion/checking and RDI-L insertion and checking
  - STS-48/STS-12 pointer interpreter function with STS-1 granularity
  - TFI-5 Link Layer support
- **Multiple Protocol Compliant Multi-Channel Aligner**
  - Supports simultaneous alignment of four independent alignment groups on one device
  - Provides up to 32 channel alignment on one device and 64 channel alignment across two devices
- **Integrated Loopback Modes for System Debugging**
  - Far End Loopback (RX to TX) provided for testing line connections to and from LatticeSC device
  - Near End Loopback (TX to RX) provided to test connections across PCS/FPGA logic interface
  - Integrated 2^7 and 2^31 PRBS generator/checkers per channel.

© 2007 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

## flexiPCS Introduction

The LatticeSC family of FPGA combines a high-performance FPGA fabric, high-performance I/Os and large embedded RAM in a single industry leading architecture. All LatticeSC devices also feature up to 32 channels of embedded SERDES with associated Physical Coding Sublayer (PCS) logic. The flexiPCS logic can be configured to support numerous industry standard high-speed data transfer protocols.

Each channel of flexiPCS logic contains dedicated transmit and receive SERDES for high-speed full-duplex serial data transfers at data rates up to 3.8 Gbps. The PCS logic in each channel can be configured to support an array of popular data protocols including SONET (STS-12/STS-12c, STS-48/STS-48c, and TFI-5 support of 10 Gbps or above), Gigabit Ethernet (compliant to the IEEE 1000BASE-X specification), 1.02 or 2.04 Gbps Fibre Channel, PCI Express, and Serial RapidIO. 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.

Protocols requiring data rates above 3.8 Gbps can be accommodated by dedicating either one pair or all 4 channels in one PCS quad block to one data link. One quad can support full-duplex serial data transfers at data rates up to 15.2 Gbps. A single PCS quad can be configured to support 10Gb Ethernet (with a fully compliant XAUI interface), 10Gb Fibre Channel, and x4 PCI Express and 4x RapidIO.

The PCS also provides bypass modes that allow a direct 8-bit or 10-bit interface from the SERDES to the FPGA logic. Each SERDES pin can also 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.

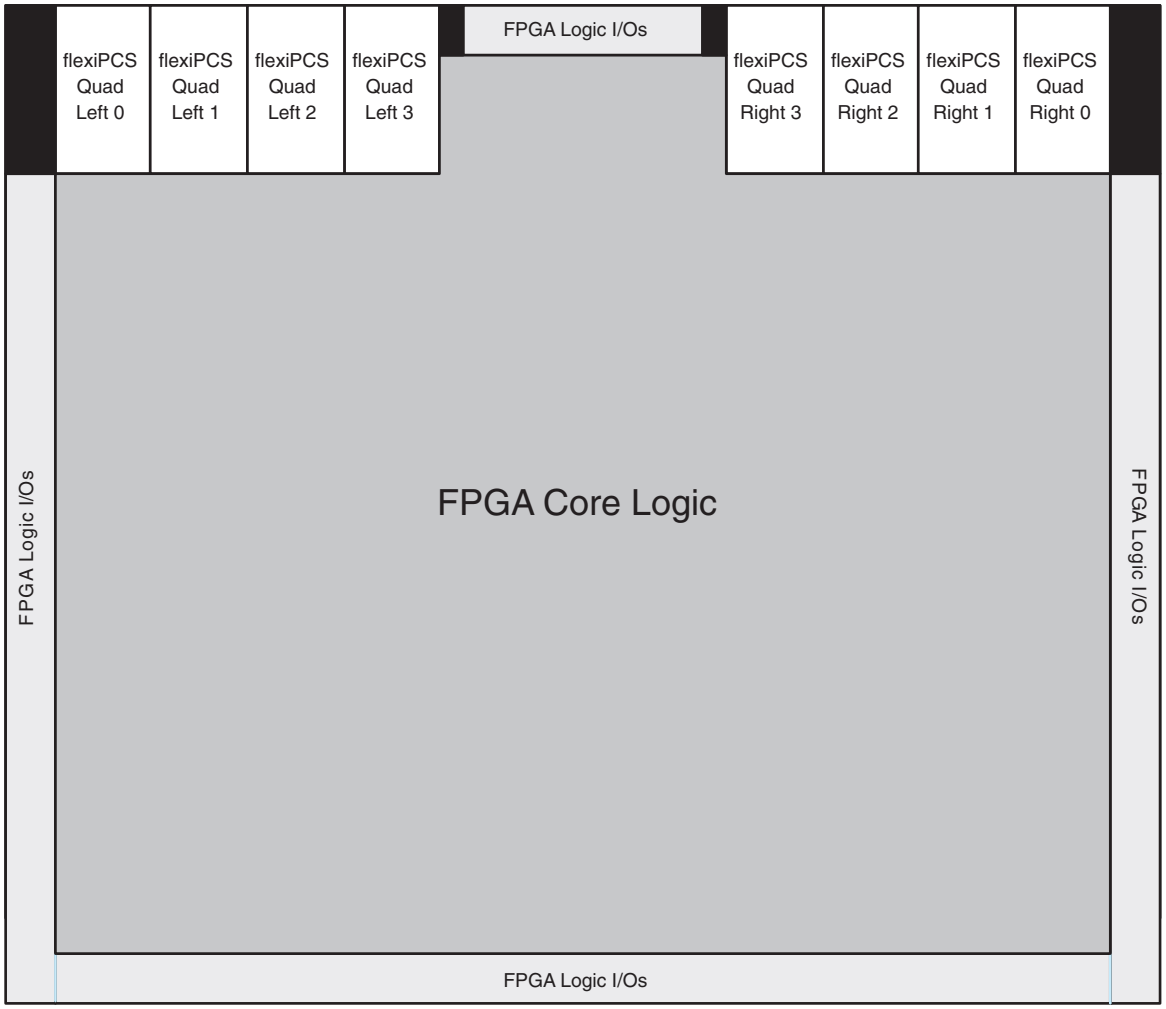
## Architecture Overview

The flexiPCS logic is arranged in quads containing logic for four independent full-duplex data channels. Each device in the LatticeSC family has up to 8 quads of flexiPCS logic. The table on the cover page of the LatticeSC/M Family Data Sheet contains the number of flexiPCS channels present on the chip. Note that in some packages (particularly lower pin count packages), not all channels from all quads on a given device may be bonded to package pins.

### flexiPCS Quad

Figure 1-1 is a layout of a LatticeSC device showing the arrangement of flexiPCS quads on the device (the largest array containing 8 quads is shown. Other devices have fewer quads).

***Figure 1-1. LatticeSC 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 LatticeSC family of devices ideal for bridging between different standards.

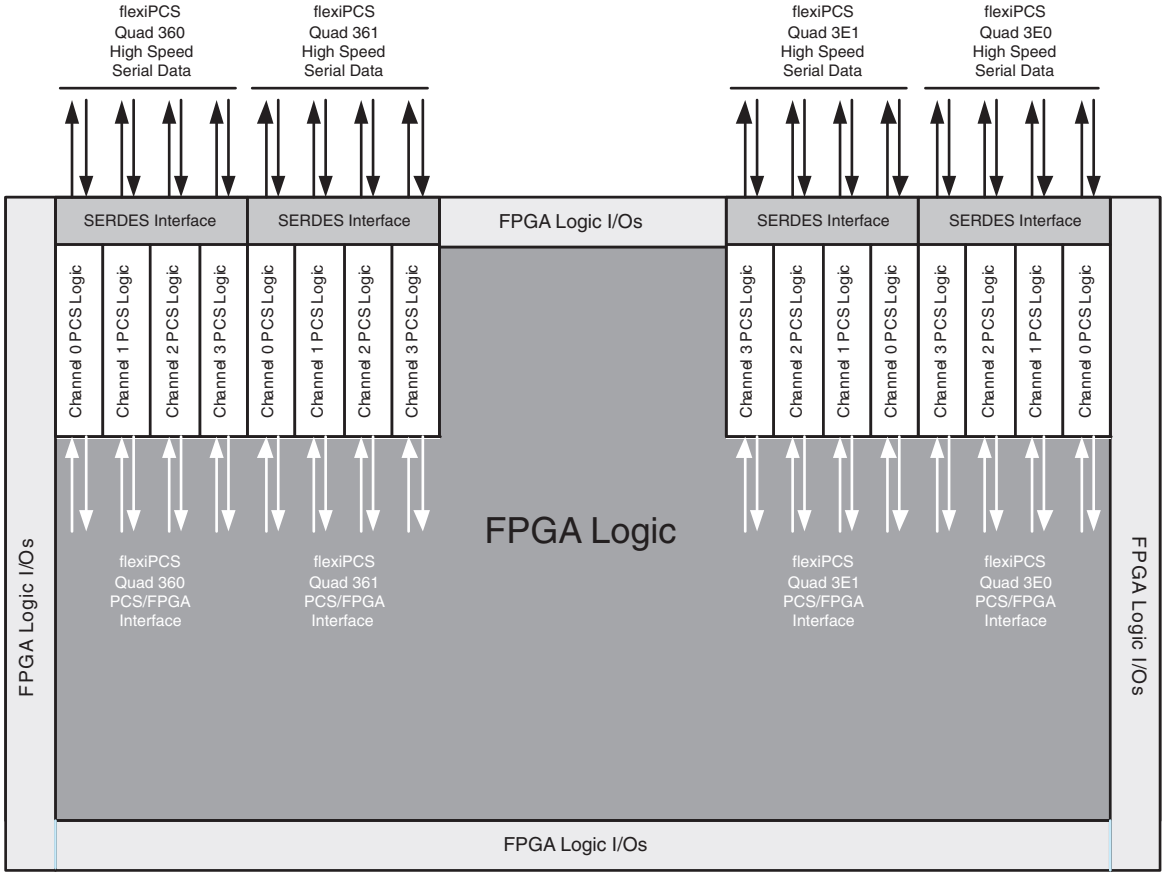
flexiPCS quads are not dedicated solely to industry standard protocols. Each quad (and each channel within a quad) can be programmed for many user defined data manipulation modes. For example, modes governing user-defined word alignment, and multi-channel alignment can be programmed for non-protocol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

## flexiPCS Channel

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 1-2 shows an example of a device with four flexiPCS quads which contain a total of 16 flexiPCS channels. Quad are named according to the base address of their control and status registers: Quad 360, Quad 361, Quad 362, Quad 363, Quad 3E0, Quad 3E1, Quad 3E2, and Quad 3E3. LatticeSC devices with less than 8 quads have quads with the lowest trailing numbers. For example, a device with two quads has quads 360 and 3E0. A device with four quads has quads 360, 3E0, 361, and 3E1.

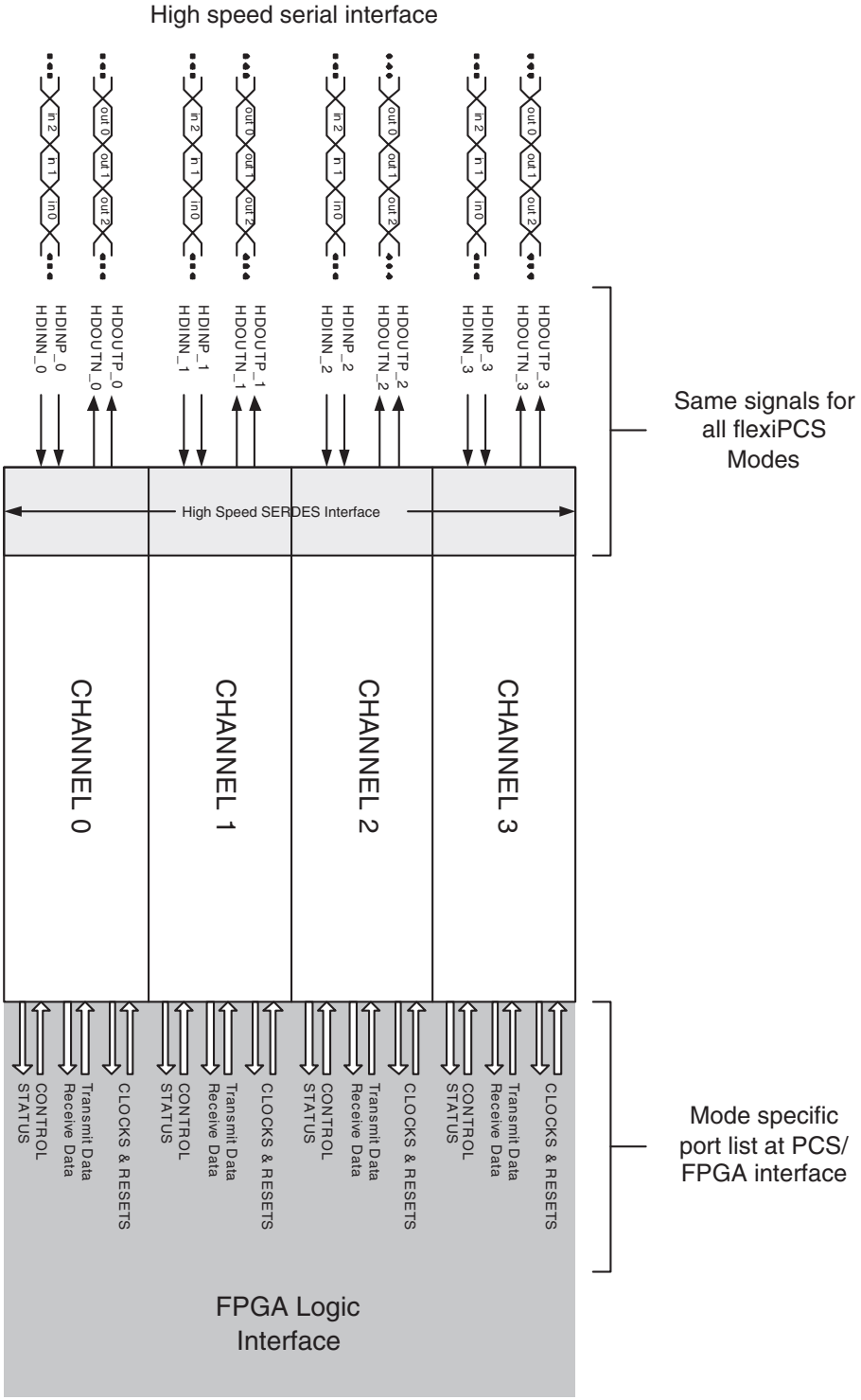
***Figure 1-2. Example of LatticeSC Device with four flexiPCS Quads***




## Per Channel PCS/FPGA Interface Ports

All flexiPCS quads regardless of chosen mode have the same external high speed serial interface at the package pins. However, every flexiPCS 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 the corresponding mode description section of the flexiPCS Data Sheet. A simplified diagram showing the channels within a single quad is shown in Figure 1-3.

***Figure 1-3. flexiPCS Quad External and Internal FPGA Interfaces***



A general description of the FPGA interface signals follows:

**Clocks & Resets**
A flexiPCS quad supplies per channel locked reference clocks and per channel recovered receive clocks to the FPGA logic interface. Each flexiPCS quad provides these clocks on both primary and general 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 the 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**
For each channel in the quad, 8-bit or 10-bit (depending on mode) transmit data from the FPGA is supplied to the PCS, where it is serialized, and sent off chip by the SERDES. A gearing option per channel is provided for a 16 or 20-bit FPGA transmit data interface which runs at one half the nominal rate.

Data is synchronized to the quad reference clock supplied either from external pins or the FPGA logic.

**Receive Data**
For each channel in the quad, serial data and clock is supplied by an external source to the SERDES pins. The data is deserialized and manipulated by the PCS logic and passed as 8-bit or 10-bit (depending on mode) data to the FPGA logic. A gearing option per channel is provided for a 16 or 20-bit FPGA receive data interface which would run at half the nominal rate. Clocks are recovered for each channel and made available at the internal FPGA logic interface.

**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.

**System Bus Control and Monitoring**
Dedicated registers accessible through the System Bus interface are provided to control a wide range of options in the SERDES and PCS. Similarly, System Bus registers are provided for monitoring status and alarm conditions. Register maps are provided at the end of the flexiPCS Data Sheet for all the control and status registers common to all the PCS quads on a device. Registers are provided which cover multi-quad and channel control and status. The System Bus addressing scheme for each of these register types is provided at the beginning of each mode section. For more information on the System Bus, refer to the TN1085, LatticeSC MPI/System Bus.

## Using this Data Sheet

The ispLEVER design tools from Lattice support all modes of the flexiPCS. 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. Nine quad modes are currently supported by the ispLEVER design flow:

Quad flexiPCS Modes:

- 8-bit SERDES Only
- 10-bit SERDES Only
- SONET (STS-12/STS-48)
- Gigabit Ethernet
- Fibre Channel (Single SERDES)
- XAUI
- Serial RapidIO
- PCI Express
- Generic 8b10b

The LatticeSC/M Family flexiPCS Data Sheet has been written in a modular fashion and is organized along the lines of the modes selectable in the ispLEVER design flow. Nine sub-sections are each dedicated to one of the above modes. In addition, several sections covering information common to all modes, including SERDES Functionality/Electrical & Timing Characteristics, Multi-channel Alignment, flexiPCS Testing and the flexiPCS Memory Map, may be of interest to users.

The ispLEVER design flow for the LatticeSC family allows the user to define the operation of the PCS on a per quad basis. Therefore, the flexiPCS Data Sheet describes the operation of each mode on a per quad basis as well. Each of the nine subsections describes the operation of a single quad. In general, the operation of a quad does not affect the choice of operation of any of the other quads on a device. The major exception is when several quads are configured to the same protocol and multi-channel alignment is required between channels on separate quads. The Multi-Channel Alignment section of this document describes how a LatticeSC device can be configured for alignment of a pair of channels, all channels within a quad, channels across multiple quads on the same device, and channels across multiple quads on two devices.

A user who is interested in the application of the flexiPCS for a specific protocol need only refer to the sub-section describing the mode dedicated to that protocol. A recommended list of sections might include:

- Introduction to flexiPCS
- SERDES Functionality/Electrical & Timing Characteristics (Contains extra detail on the SERDES functionality which is applicable to all modes)
- Specific Mode of interest (Gigabit Ethernet Mode, SONET Mode, XAUI Mode, etc.)
- Multi-Channel Alignment (If needed)
- flexiPCS Testing (If needed)
- flexiPCS Memory Map

The flexiPCS Data Sheet 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 SERDES Functionality/Electrical & Timing Characteristics section of this document. Package pinout information is provided in the Architecture section of the LatticeSC/M Family Data Sheet.

# SERDES Functionality/Electrical & Timing LatticeSC/M flexiPCS Family Data Sheet

June 2011 Data Sheet DS1005

## Introduction

e a high speed SERDES serial data interface. This section describes the operation of the LatticeSC SERDES for all modes of the SEREDES/PCS block.

The embedded SERDES PLLs support serial data rates from 600 Mbps to 3.8 Gbps which covers a wide range of industry standard protocol data transmission rates. For applications which require a lower input data rate to the SERDES buffers, low speed connections from the SERDES input buffers to the FPGA logic are also available (see the **8-bit and 10-bit SERDES Only Modes** section of the **LatticeSC/M Family flexiPCS Data Sheet**).

### LatticeSC flexiPCS Quad Module

Devices in the LatticeSC family may have up to 8 quads of embedded flexiPCS logic. Each quad in turn supports 4 independent full duplex data channels. Therefore, each quad in a LatticeSC device is capable of supporting 4 transmit and 4 receive SERDES serial data links. This section describes the operation of a single quad of flexiPCS logic.

Operation of the SERDES requires the user to provide a reference clock or clocks to each active quad to provide a synchronization reference for the SERDES PLLs. Reference clocks are shared among all four channels of a quad. If the transmit and receive frequencies are within the specified ppm tolerance for the LatticeSC SERDES (See the **SERDES Electrical &Timing Characteristics** section), only one reference clock is needed for both transmit and receive directions. If the receive frequency is not within the specified ppm tolerance, then separate reference clocks for the transmit and receive directions must be provided.

Simultaneous support of non-synchronous high-speed data rates on one device is possible with the use of multiple quads.

## Quad and Channel Option Control

Although the mode selection and reference frequency covers an entire quad, many options covering clocking and data formatting are available on a per channel basis. Options are available only through dedicated registers that can be written and read via the embedded System Bus Interface (see the System Bus section for more details on the operation of the System Bus), or during device configuration. Depending on the specific option, a register may affect operation of multiple quads, a single quad or an individual channel in a quad. Registers dedicated to multiple quads are listed in the Inter-quad Interface Register Map in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet. Registers dedicated to one quad are listed in the Quad Interface Register Map. Registers dedicated to a single channel are listed in the Channel Interface Register Map.

### Register Map Addressing

Figure 2-1 provides a map of base register addresses for any of the flexiPCS quads on a LatticeSC device. Note that different devices may have different numbers of available quads up to a total of 8 quads (or 32 channels) maximum.

Inter-quad Interface, Quad Interface, and Channel Interface Registers are listed by Offset Address. Each Inter-quad Interface Register, Quad Interface Register, and Channel Interface Register has a unique 18-bit address determined by the specific quad or channel targeted. The addressing of Inter-quad Interface Registers, Quad Interface Registers, and Channel Interface Registers is shown Figure 2-1.

Base addresses are dependant on the physical location of a given quad or channel on a LatticeSC device. Each quad and channel has an associated set of control and status registers. These registers are referred throughout this data sheet by their offset addresses. The unique 18-bit address of a control or status register for a specific

© 2011 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

quad or channel is simply the offset address of that register added to the base address for that specific quad or channel as shown in Figure 2-1.

Channel Interface Registers can be written in one of three methods. One method is to write to one specific register corresponding to one specific channel. The other two methods involve writing to multiple channel registers with just one write operation. This is referred to as a Broadcast write. A Broadcast write is useful when multiple channel registers are to be written with the same value. The first method of Broadcast writing involves writing to all four channel registers in a quad at one time. A second method of Broadcast writing involves writing to all channel registers for all quads on the left or right side of the device at one time. Therefore, a specific Channel Interface Register may be accessed through any one of three channel addresses, depending on the number of channel registers being written to at any one time.

A broadcast write to multiple quads cannot be used to power on SERDES in any device-package combination that does not have all SERDES quads bonded because of excess power on the board and jitter is also affected.

Throughout this document, the byte-wide data at each offset address is listed with a hexadecimal value with bit 0 as the MSB and bit 7 as the LSB as per the PowerPC convention. For example if the value for Quad Interface Register Offset Address 0x02 is stated to be 0x15, the bit pattern defined is as shown in Table 2-1:

***Table 2-1. Control/Status Register Bit Order Example***

| **Quad Interface Register Offset Address 0x02 = 0x15** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Data Bit 0 | Data Bit1 | Data Bit 2 | Data Bit 3 | Data Bit 4 | Data Bit 5 | Data Bit 6 | Data Bit 7 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| **1** | | | | **5** | | | |

A full list of register Offset Addresses is provided in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet.

***Figure 2-1. flexiPCS Inter-quad, Quad, and Channel Interface Register Map Base Addressing***

## Auto-Configuration

Initial register setup for each flexiPCS mode can be performed without accessing the system bus by using the auto-configuration tool, IPexpress. IPexpress generates 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

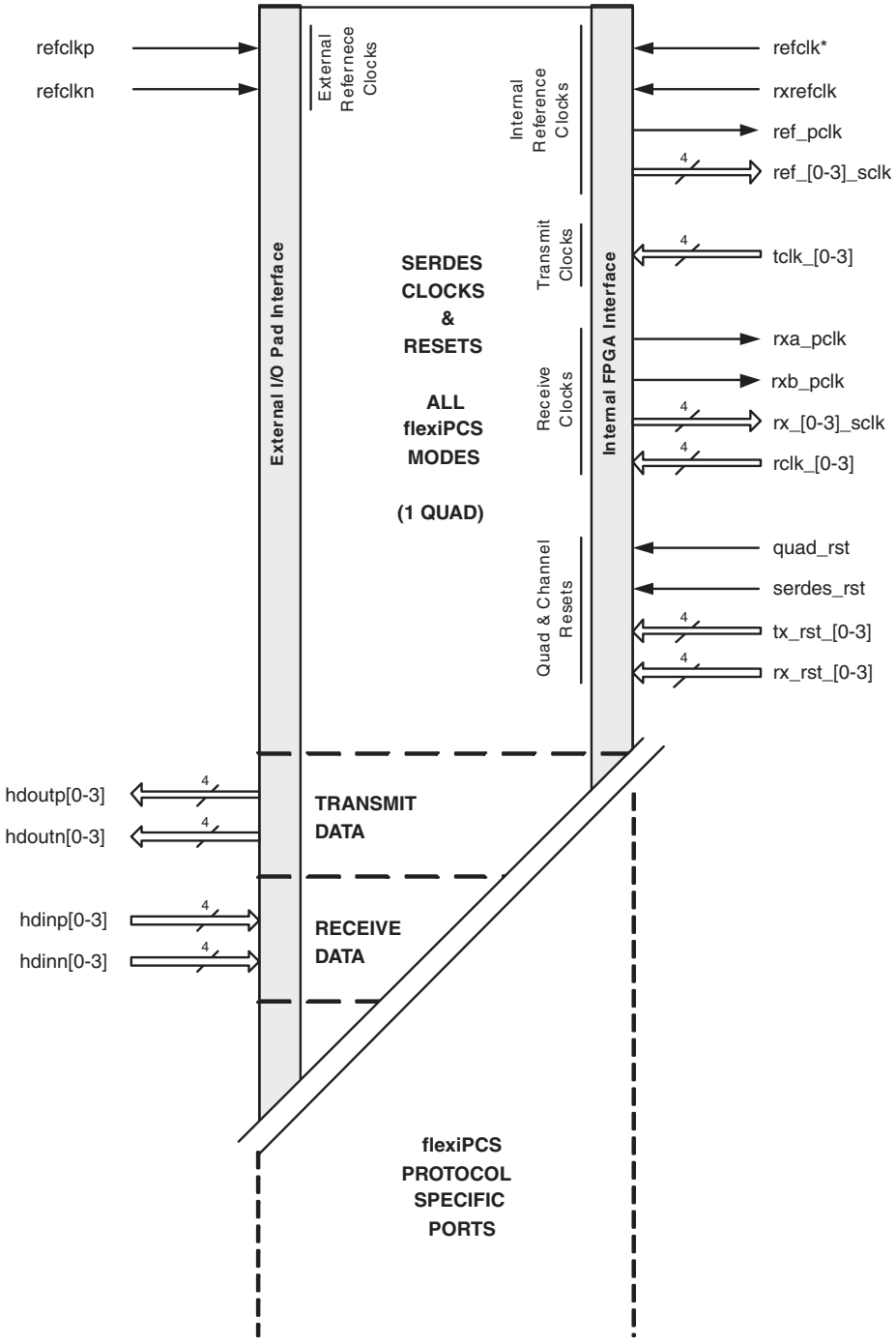
registers will be set upon reset to values defined in the auto-configuration file. The system bus is therefore not needed if all quads are to be set via auto-configuration files. However, the system bus will still need to be included in a design if the user wants to change control registers or monitor status registers during operation.

## SERDES Functionality

IPexpress allows the designer to choose the protocol for each flexiPCS quad used in a given design. All PCS modes share a common list of SERDES pins. Control of the SERDES is the same regardless of PCS protocol chosen. The specific choice of SERDES options may be constrained by the particular application corresponding to the PCS protocol chosen. This section details the control of a single quad of SERDES common to any PCS mode. For more specific information about the setup of the SERDES for a particular protocol mode, refer to the appropriate section of the **LatticeSC/M Family flexiPCS Data Sheet**.

Figure 2-2 displays the common list of SERDES clock and reset pins for one PCS quad. In addition to these pins, each PCS mode will also have data, control, and status ports specific to the protocol mode chosen. The functionality of these additional ports is discussed in the appropriate **LatticeSC/M Family flexiPCS Data Sheet** section.

***Figure 2-2. Common SERDES Port Pin Diagram***




* The refclk inputs for all active quads on a device must be connected to the same clock. The rxrefclk inputs for all active quads on a device do not have to be connected to the same clock.

## SERDES Port Description

Table 2-2 lists all the common ports to/from a PCS quad for the SERDES logic. A brief description for each port is given in the table. A detailed description of the function of each port is given in the **SERDES Functionality Detailed Description** section.

***Table 2-2. SERDES Common Ports***

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Reference Clocks** | | | |
| refclkp | In from I/O pad | N/A | Reference clock input, positive. Dedicated CML input. |
| refclkn | In from I/O pad | N/A | Reference clock input, negative. Dedicated CML input |
| refclk | In from FPGA | N/A | Optional transmit reference clock input from FPGA logic. Can be used instead of I/O pin reference clock.<br><br>The Tx refclk inputs for all active quads on a device must be connected to the same clock. |
| rxrefclk | In from FPGA | N/A | Optional receive reference clock input from FPGA logic. Can be used instead of I/O pin reference clock. The rxrefclk inputs for all active quads do not have to be connected to the same clock. |
| ref_pclk | Out to FPGA | N/A | Locked reference clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| ref_[0-3]_sclk | Out to FPGA | N/A | Per channel locked reference clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| **Transmit Clocks** | | | |
| tclk_[0-3] | In from FPGA | N/A | Per channel transmit clock inputs from FPGA. Used to clock the TX data phase compensation FIFO with clock synchronous to the reference clock. May also be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Receive Clocks** | | | |
| rxa_pclk | Out to FPGA | N/A | Recovered receive clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| rxb_pclk | Out to FPGA | N/A | Recovered receive clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| rx_[0-3]_sclk | Out to FPGA | N/A | Per channel recovered receive clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| rclk_[0-3] | In from FPGA | N/A | Per channel receive clock inputs from FPGA. May be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference and/or receive reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Resets** | | | |
| quad_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of SERDES and PCS logic. |
| serdes_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of the SERDES. |
| tx_rst_[0-3] | In from FPGA | ASYNC | Per channel active high, asynchronous reset of individual transmit channel of PCS logic. |
| rx_rst_[0-3] | In from FPGA | ASYNC | Per channel active high, asynchronous reset of individual receive channel of PCS logic. |
| **Transmit Data** | | | |
| hdoutp[0-3] | Out to I/O pad | N/A | High-speed CML serial output, positive. |
| hdoutn[0-3] | Out to I/O pad | N/A | High-speed CML serial output, negative. |

*Table 2-2. SERDES Common Ports*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Receive Data** | | | |
| hdinp[0-3] | In from I/O pad | N/A | High-speed CML serial input, positive. |
| hdinn[0-3] | In from I/O pad | N/A | High-speed CML serial input, negative. |

# SERDES Functionality Detailed Description

The following section provides a description of the operation of the SERDES for all PCS modes. The functional description is organized according to the port listing shown in Figure 2-2. The System Bus Offset Addresses for any control and status registers relevant to a given function are provided with the description of the operation of that function.

## SERDES Powerup

By default, the SERDES logic in all quads is powered down. The SERDES logic must be powered up in order to access any channel in a specific SERDES/PCS quad, regardless of the quad mode chosen. The SERDES logic in a SERDES/PCS quad is powered up by writing the appropriate Quad Interface Register Offset Address 0x28, bit 1 to a '1'. For more information regarding hot socketing requirements, see the Architecture section and DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet.

The bias levels for all SERDES channels on one side of a LatticeSC device (either left side or right side) are set by tying that side's external bias resistor to ground. Figure 2-3 shows the bias resistor connections for the SERDES/PCS quads on each side of the device. The bias resistor value must be 4.02K ohm +/- 1%. Note the bias levels for all left side SERDES channels are set in the leftmost quad (Quad Base Address 36000). This means that the leftmost quad SERDES must be powered up for proper operation of any SERDES channel on the left side (meaning Quad Interface Register 36028, bit 1 must be set to a '1'). Likewise, the bias levels for all right side SERDES channels are set in the rightmost quad (Quad Base Address 3E000). This means that the rightmost quad SERDES must be powered up for proper operation of any SERDES channel on the right side (meaning Quad Interface Register 3E028, bit 1 must be set to a '1').

*Figure 2-3. SERDES Internal Bias Generators*

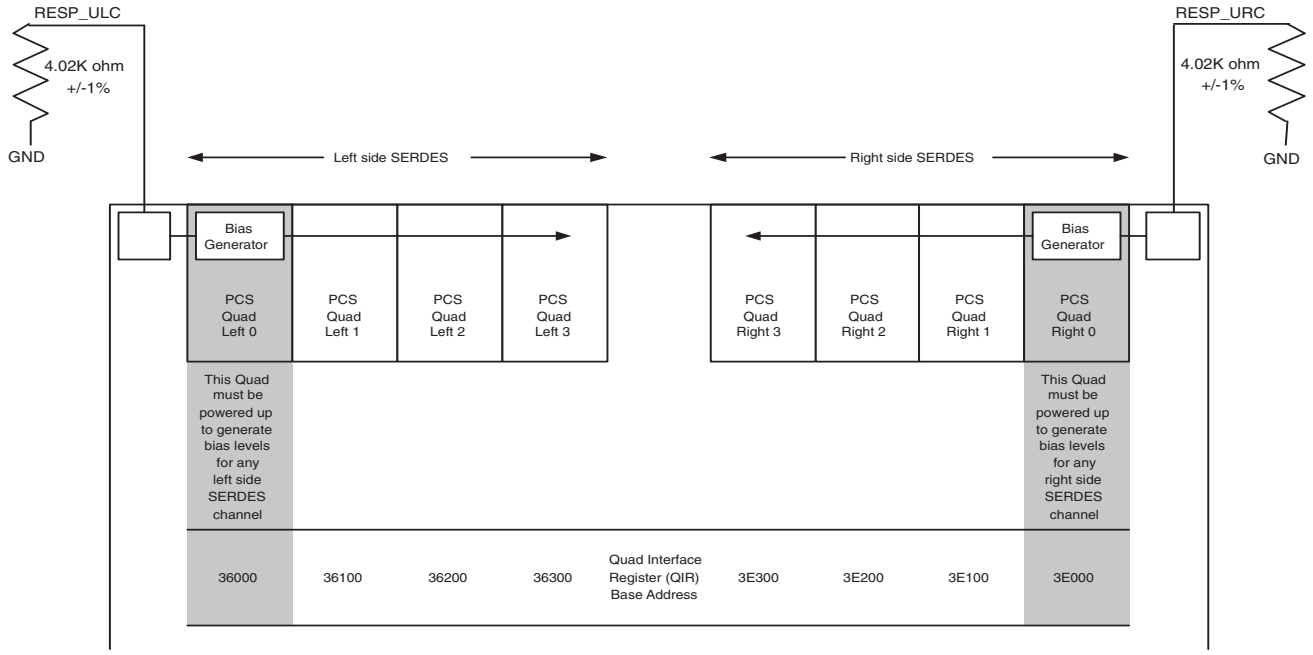
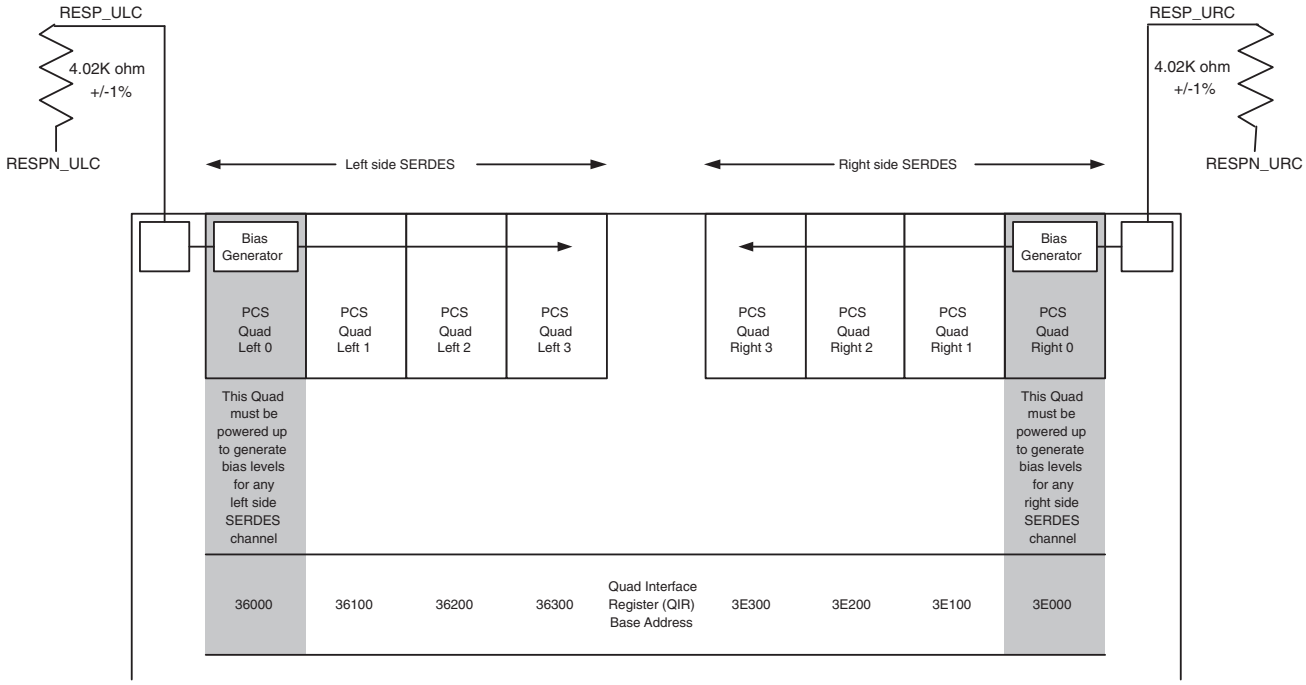


*Figure 2-4. SERDES Internal Bias Generators (for Packages that Include the RESPN_[ULC/URC] Pins)*




## Clocks & Resets

Every LatticeSC PCS quad regardless of selected mode has the same basic clocking and reset architecture. The choice of reference clock frequencies and the selection of PCS/FPGA interface clocks will be determined by the particular application. The following is a description of the functions of the clock and reset ports. For a specific explanation of the clocking strategy for a particular data protocol, refer to the relevant section of the **flexiPCS Data Sheet**.

### Reference Clocks

A reference clock or clocks must be provided to each active quad to provide a synchronization reference for the SERDES PLLs. Reference clock(s) are shared among all four channels of each quad. Reference clocks must be supplied for each active quad in the device. The LatticeSC SERDES provides two options for choosing the source of the reference clock(s) as shown in Table 2-3.

*Table 2-3. Reference Clock Source Selection*

| Quad Interface Register Offset Address = 0x29 | | Reference Clock Source Selection |
|---|---|---|
| **Bit 3** | **Bit 4** | **Description** |
| 0 | 0 | Single external reference clock supplied to **refclkn**/**refclkp** pins. Reference clock used for both transmit and receive directions. (Default mode). |
| 1 | 0 | Internal (from FPGA logic) reference clocks supplied to **refclk** (transmit) and **rxrefclk** (receive). **refclk** and **rxrefclk** can be tied together or connected to separate clocks at the FPGA interface. |

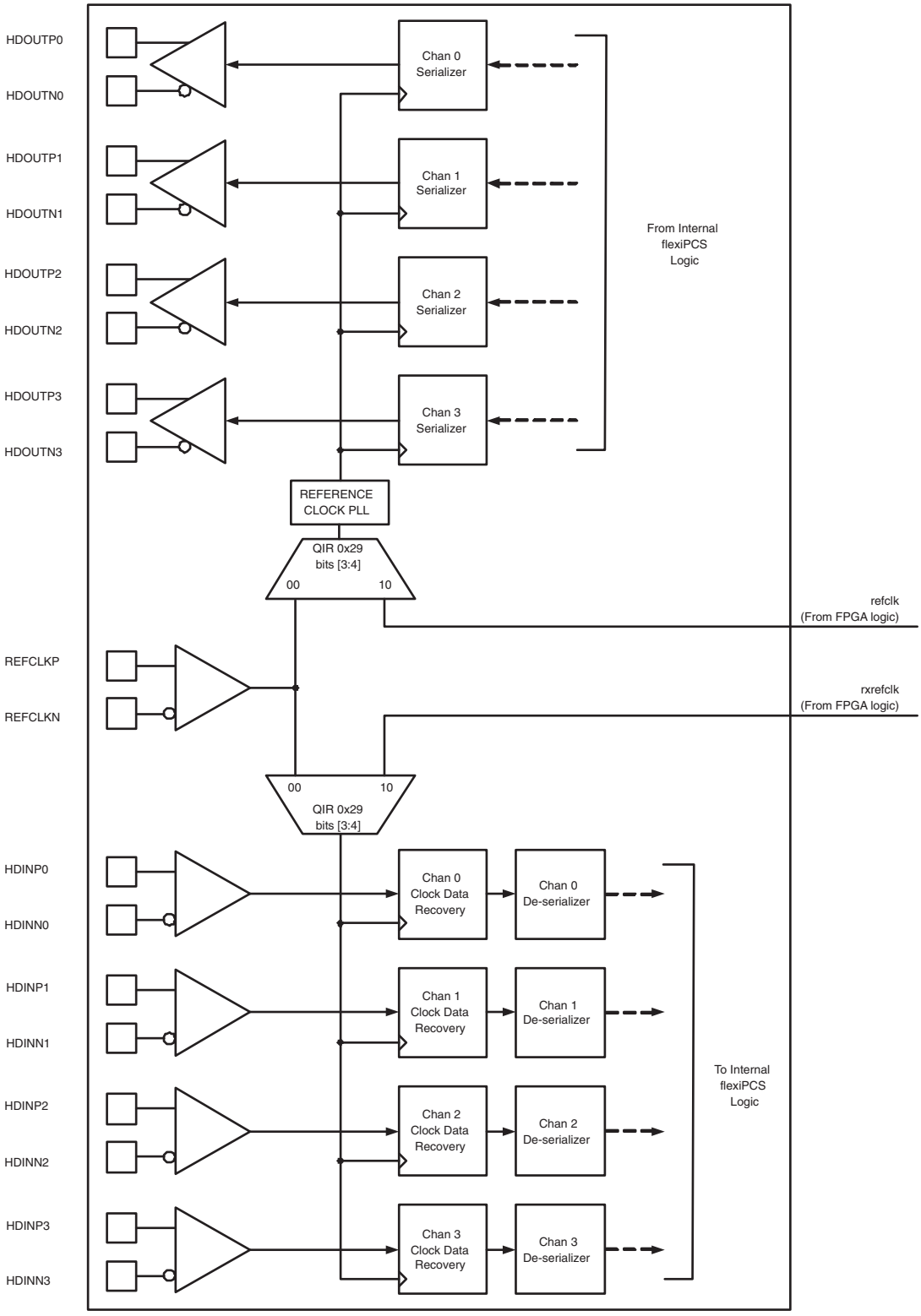
In the default reference clock mode (QIR 0x29, bits [3:4] = “00”), one external reference clock needs to be supplied to pins **refclkn**/**refclkp.** In this mode, the **refclkn**/**refclkp** clock supplies the reference frequency for both transmit and receive data paths. This is appropriate for any applications where the transmit and receive frequencies are

within the specified ppm tolerance for the LatticeSC SERDES (See the **SERDES Electrical & Timing Characteristics** section).

For applications where the transmit and receive frequencies are not the same, two internal reference clocks can be supplied. This mode can be selected by setting the Quad Interface Offset Register 0x29, bits [3:4] = “10”. The transmit reference clock is then supplied to the refclk port at the flexiPCS/FPGA interface. Note that the refclk inputs for all active quads on a device must be connected to the same clock internally. The receive reference clock is supplied to the rxrefclk port at the PCS/FPGA interface. The rxrefclk inputs for all active quads on a device does not have to be connected to the same clock internally. The refclk and rxrefclk reference clock inputs can be tied to the same clock or can be tied to different clocks. In modes which utilize a Clock Tolerance Compensation (CTC) block, these clocks must be the same frequency to within the given protocol' s clock tolerance specification. In non-CTC modes (SERDES Only, Generic 8b10b, and SONET modes) there is no such restriction.

A diagram illustrating the reference clock options for the SERDES is shown in Figure 2-5:

***Figure 2-5. SERDES Reference Clock Selections***



**Reference Clock Control Registers**
Quad Interface Register bits can be used to control other options relevant when using an external reference clock.

By default, the external reference clock inputs **refclkn**/**refclkp** are AC coupled. These reference clock pins can be set to DC coupling by setting Quad Interface Register Offset Address 0x29, bit 1 to '1'.

**Loss of Lock Alarm Registers**
Both the transmit PLL and the individual channel CDRs have 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 reaquires lock.

If a CDR loses lock to the incoming data on the **hdin** inputs, the loss-of-lock for that channel is asserted and the CDR attempts to lock to the reference clock instead. After successfully locking to the reference clock, loss-of-lock for that channel is deasserted and the CDR then attempts to relock to the incoming channel's data. If successful, the CDR will then remain locked to the data. If the CDR cannot relock to the incoming data, then it will once again report an out of lock condition and repeat the training cycle.

The loss of lock alarm for the quad TX PLL can be read at Quad Interface Register Offset Address 0x86, bit 7. The loss of lock alarm for each channel's receive CDR can be read at the appropriate Channel Interface Register Offset Address 0x94, bit 6.

The receiver CDR loss of lock alarm for each channel's receiver input can be read at Channel Interface Register Offset Address 0x94, bit 6.

The time to detect both lock and loss-of-lock for the transmit reference clock PLLs or the individual CDR, when QIR0x2A[7:6]=00 (+/- 600 ppm), is shown in Table 2-4

***Table 2-4. Time to Lock and Loss-of-Lock for TX PLL and CDRs***

| Description | Min. | Typ. | Max. | Units |
|---|---|---|---|---|
| Time to detect locked/unlocked condition for x1 CDR modes | | 655360 | | UI |
| Time to detect locked/unlocked condition for x2 CDR modes | | 1310720 | | UI |

## Rate Mode Registers

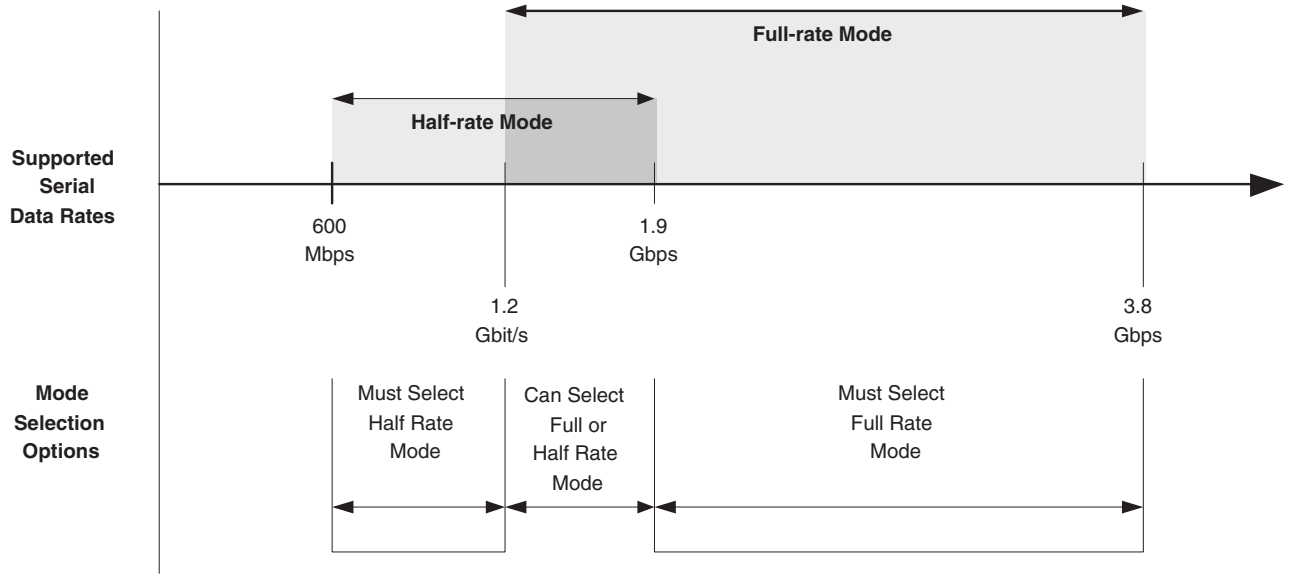
The following section describes how to change the SERDES generated clock rates through register settings. These settings are to be set via an auto-configuration file generated within IPexpress. Reference clock and output clock frequencies can by set by the user in the GUI and the optimal register settings will be calculated and written to the auto-configuration file.

*Note: There are also frequency dependent SERDES performance control bits that are not writable via the System-bus. These control bits are set in the auto-configuration file created within IPexpress and passed into the bitstream through the normal FPGA design flow (map, par, bitgen). To change the bit rate for a SERDES, specify the proper values for the affected flexiPCS quad in IPexpress and regenerate the auto-configuration file. Failure to regenerate the auto-configuration file with the proper value may result in non-optimal SERDES operation.*

The following information is provided for reference. If the registers described in the **Full Rate and Half Rate Modes** section are changed via the Systembus after a bitstream has been loaded, non-optimal SERDES performance may result for the reasons described above.

**Full-Rate and Half-Rate Modes**
The SERDES PLLs support data rates from 1.2 Gbps to 3.8 Gbps. A half-rate mode is available to permit operation of the SERDES at industry standard protocol data rates lower than 1.2 Gbps. The combination of half-rate and full-rate modes allow operation of a single SERDES channel at any data rate from 600 Mbps to 3.8 Gbps as shown in Figure 2-6.

*Figure 2-6. Full-Rate and Half-Rate Mode Selection*




Each of the four receive channels can be independently configured to receive data at either full-rate or half-rate. Similarly each of the four transmit channels can be independently configured to transmit data at either full-rate or half-rate. The truth table shown in Table 2-5 for a transmit channel describes the reference clock and data rate relationship during full and half-rate operation. The same relationship applies to each receive channel as well.

Note that the internal serial (bit) clock multiplier is shown for both 8-bit and 10-bit modes in Table 2-5. The choice of 8-bit or 10-bit mode is made by the user depending on the particular application of the PCS. The control for 8-bit or 10-bit mode is described in each LatticeSC/M Family flexiPCS Data Sheet section.

The FPGA interface clock multiplier value applies to all PCS/FPGA interface locked reference, transmit and receive clocks.

*Table 2-5. Full-Rate and Half-Rate Mode Operations*

| Rate Mode | Reference Clock Mode | Internal Serial (bit) Clock Multiplier[1] | FPGA Interface Clock Multiplier | |
|---|---|---|---|---|
| | | | Quad Interface Register Offset Address 0x19[1] | |
| Channel Interface Register Offset Address = 0x13<br><br>Transmit - Bit 5<br>Receive - Bit 4 | Quad Interface Register Offset Address = 0x28,<br><br>Bits [2:3] | | 8/10 Bit Data Bus Width<br><br>Transmit - Bit 4 = '0'<br>Receive - Bit 5 = '0' | 16/20 Bit Data Bus Width<br><br>Transmit - Bit 4 = '1'<br>Receive - Bit 5 = '1' |
| 0<br><br>Full rate (1.2 Gbps - 3.8 Gbps) | 00 | 16X or 20X | 2X | 1X |
| | 01 | 8X or 10X | 1X | (1/2)X |
| | 10 | 4X or 5X | (1/2)X | (1/4)X |
| 1<br><br>Half rate (600 Mbps - 1.9 Gbps) | 00 | 8X or 10X | 1X | (1/2)X |
| | 01 | 4X or 5X | (1/2)X | (1/4)X |
| | 10 | 2X or (5/2)X | (1/4)X | (1/8)X |

1. All clock multiplier values are with respect to supplied reference clock frequency.

*Note: There are frequency dependent SERDES performance control bits that are not writable via the Systembus. These control bits are set in the auto-configuration file created within IPexpress and passed into the bitstream through the normal FPGA design flow (map, par, bitgen). To change the PLL rate for a SERDES specify the proper*

*value for the affected flexiPCS quad in IPexpress and regenerate the auto-configuration file. Failure to do this may result in non-optimal SERDES operation. These settings do not need to be changed when selecting between full-rate and half-rate operation.*

The registers which change the PLL rate for the transmit direction are: the Reference Clock Mode (QIR 0x28, bits [2:3]). The Rate Mode register (CIR 0x13, bit 5) and the FPGA Interface Clock Multiplier register (QIR0x19, bit 4) for the transmit direction do not affect the transmit PLL rate and can be changed after configuration.

The registers which change the PLL rate for the receive direction are: the Reference Clock Mode (QIR 0x28, bits [2:3]) and the Rate Mode register (CIR 0x13, bit 4). The FPGA Interface Clock Multiplier register (QIR0x19, bit 5) for the receive direction does not affect the receive PLL rate and can be changed after configuration.

**Full Rate Timing**

The relationships between the reference clock, the internal SERDES serial (bit) clock, and the PCS/FPGA interface clocks for a channel set to full rate are shown in the following diagrams. Figure 2-7 shows the timing for a transmit channel and Figure 2-8 shows the timing for a receive channel. Note that each channel in a quad can be independently set to full or half rate mode (although all channels in a given quad share the same reference clock).

Each diagram shows the relationship for all three possible reference clock modes which can be chosen. Note that the reference clock mode applies to all channels of a given quad.

***Figure 2-7. Full-Rate Transmit Path Timing Diagram***

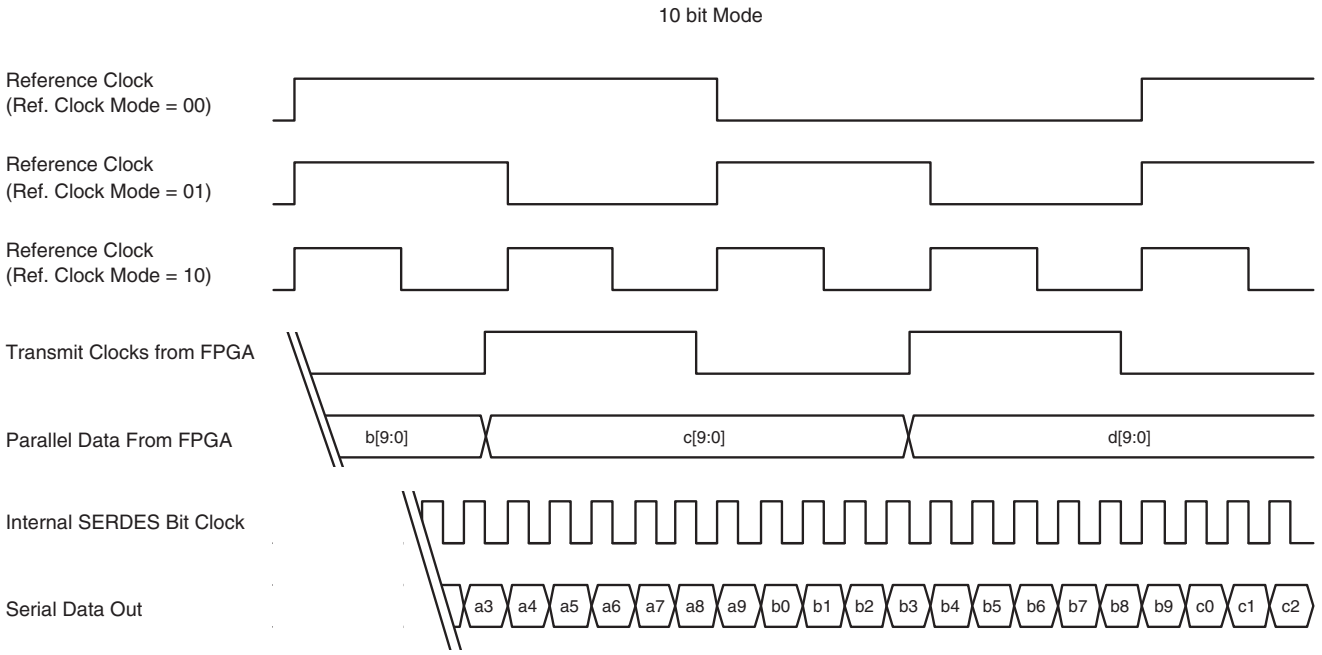
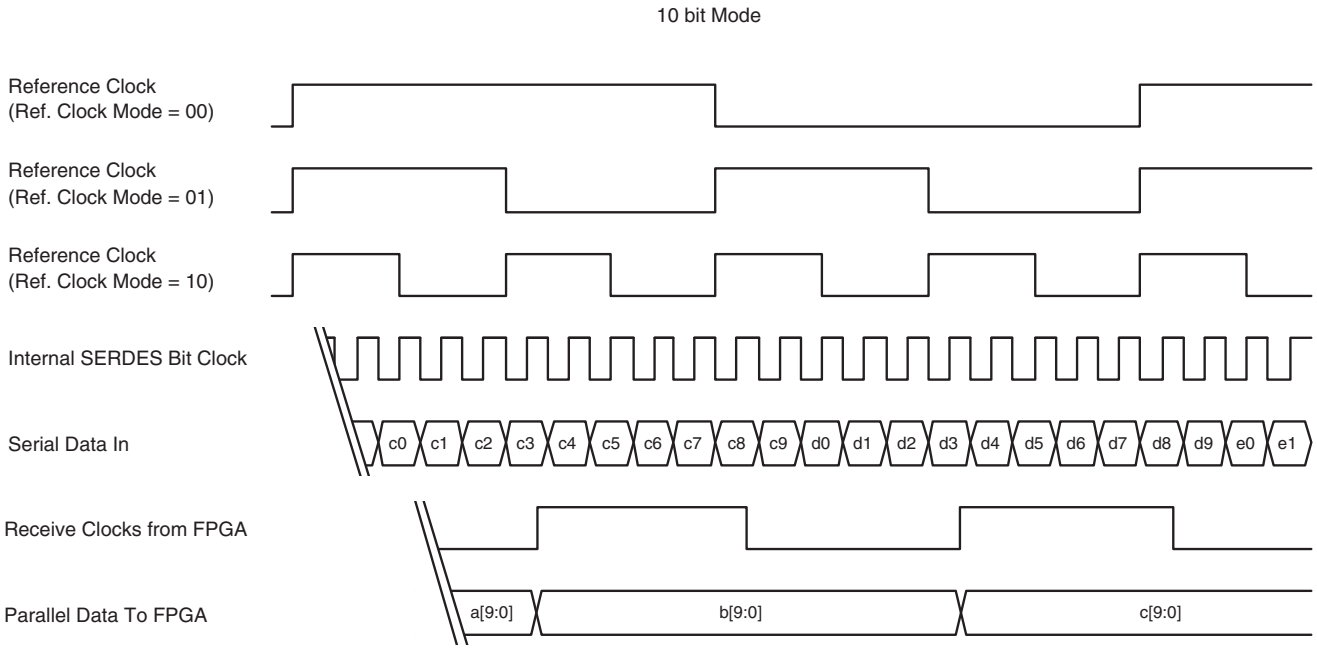


*Figure 2-8. Full-Rate Receive Path Timing Diagram*




**Half Rate Timing**

The relationships between the reference clock, the internal SERDES serial (bit) clock, and the PCS/FPGA interface clocks for a channel set to halfrate are shown in the following diagrams. Figure 2-9 shows the timing for a transmit channel and Figure 2-10 shows the timing for a receive channel. Note that each channel in a quad can be independently set to full or half rate mode (although all channels in a given quad share the same reference clock).

Each diagram shows the relationship for all three possible reference clock modes which can be chosen. Note that the reference clock mode applies to all channels of a given quad.

***Figure 2-9. Half-Rate Transmit Path Timing Diagram***

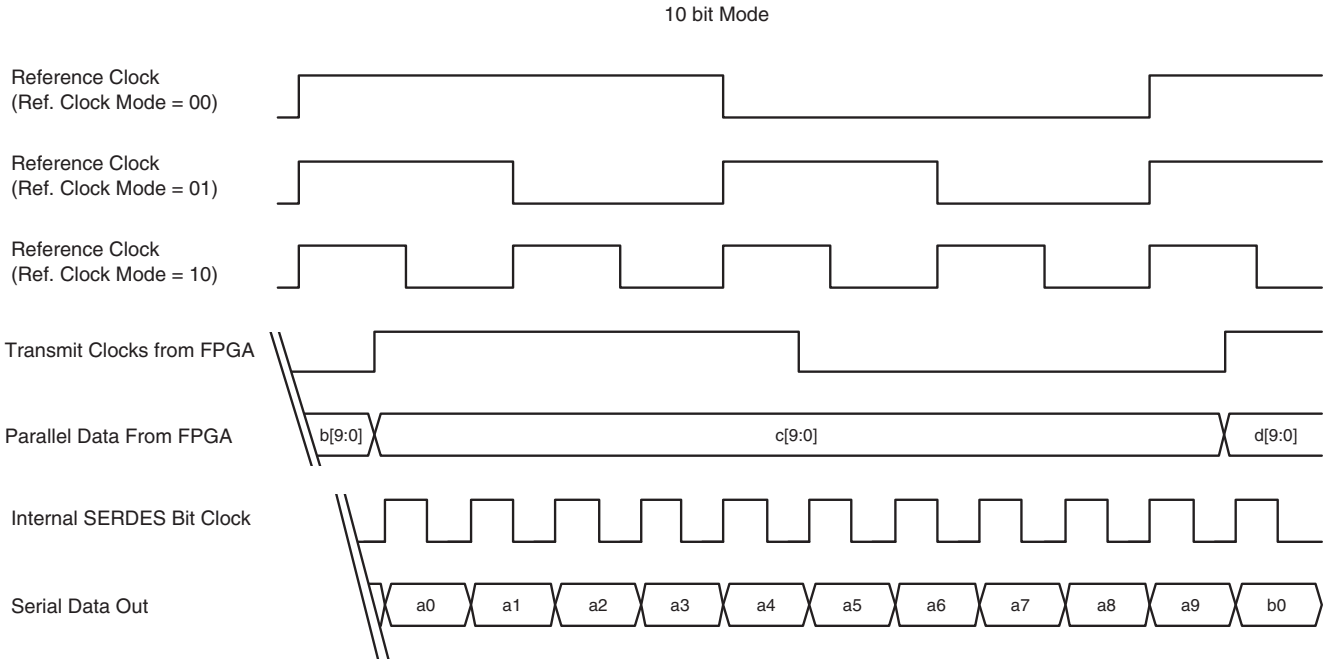
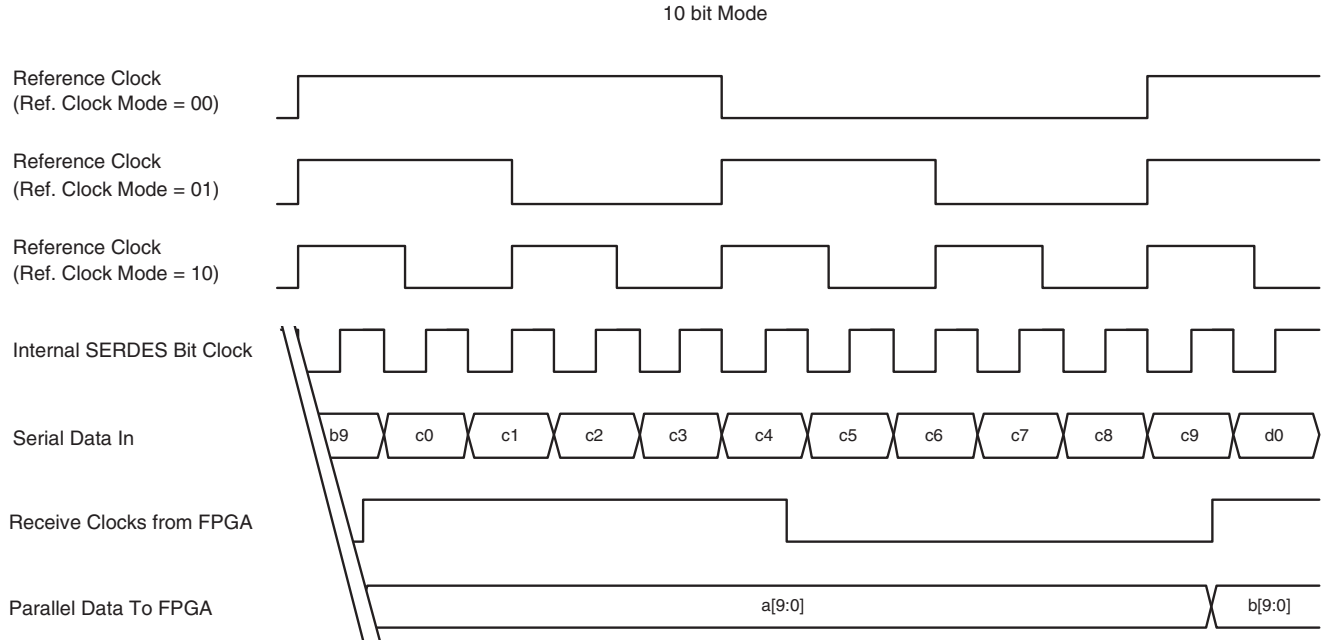


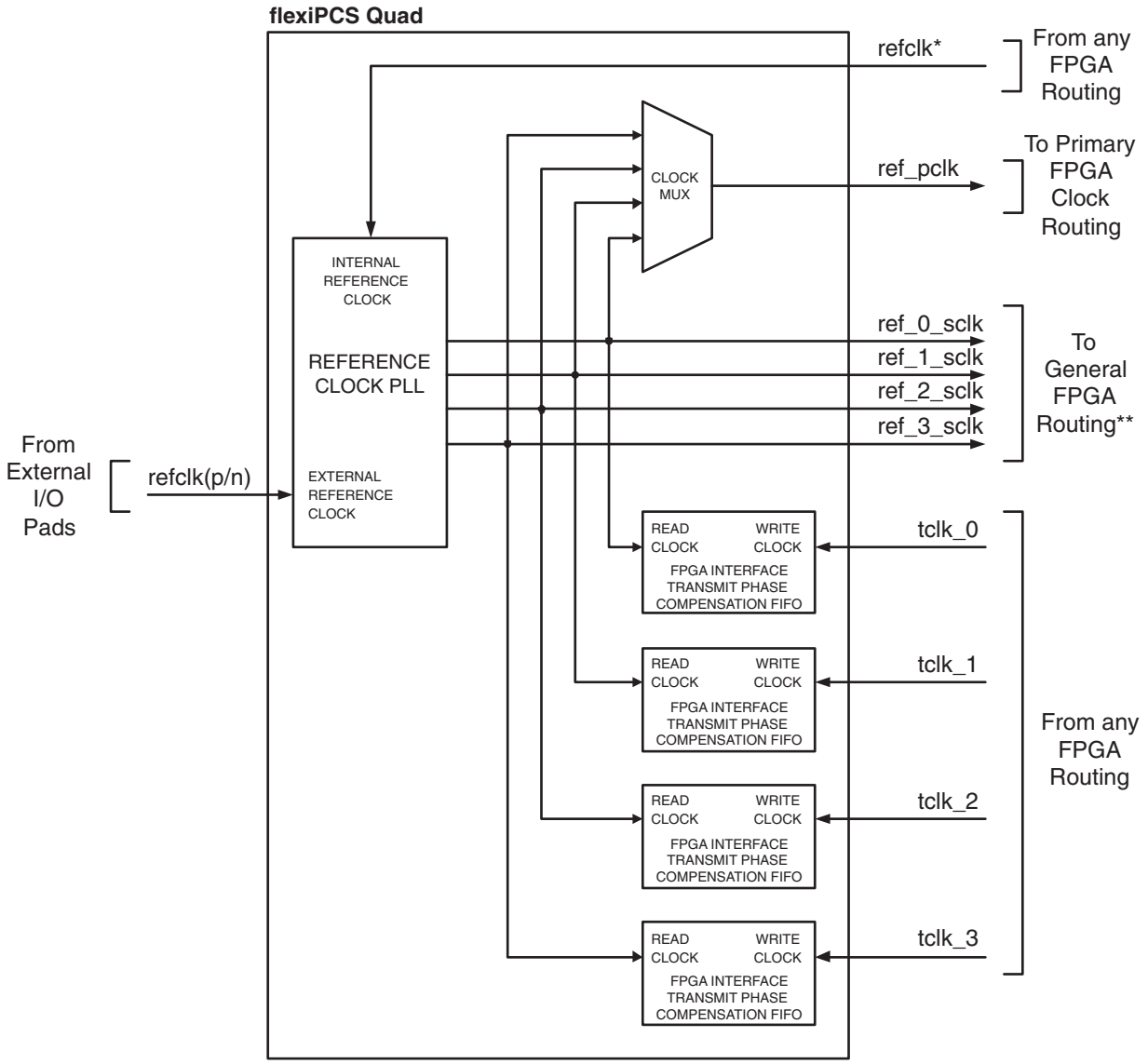

***Figure 2-10. Half-Rate Receive Path Timing Diagram***



## Locked Reference and Transmit Clocks

Figure 2-11 illustrates the relationships between the various PCS/FPGA interface locked reference and transmit clocks.

***Figure 2-11. Locked Reference and Transmit Clocks***




* The refclk inputs for all active quads on a device must be connected to the same clock. The rxrefclk inputs for all active quads on a device do not have to be connected to the same clock.

** Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing.

The reference clock PLL in the SERDES is able to lock to either an external reference clock, or a reference clock provided from the FPGA core.

### Locked Reference Clocks to FPGA Logic

The PCS provides one locked reference clock per channel (**ref_[0-3]_sclk**) to the FPGA. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary

clock routing. A four-way mux of all locked reference channel clocks (**ref_pclk**) is provided on a primary clock. The clock selection for **ref_pclk** is controlled by writing to Quad Interface Register Offset Address 0x02, bits 2 and 3 as shown in Table 2-6. All locked reference clocks are generated from a single PLL.

***Table 2-6. Locked Reference Clock Selection for ref_pclk***

| Quad Interface Register Offset Address = 0x02 | | Locked Reference Clock Selection |
|---|---|---|
| **Bit 2** | **Bit 3** | **Description** |
| 0 | 0 | Selects locked reference clock channel 0 |
| 0 | 1 | Selects locked reference clock channel 1 |
| 1 | 0 | Selects locked reference clock channel 2 |
| 1 | 1 | Selects locked reference clock channel 3 |

## Transmit Clocks

Each respective channel's input transmit clock can be driven from the FPGA logic on pins **tclk_[0-3].** These clocks serve as the write clocks to FPGA interface phase compensation FIFOs embedded in the quad PCS logic as shown in Table 2-9.

### Transmit Clocks from FPGA Logic

The FPGA interface phase compensation FIFOs are intended only to provide a clean timing interface to the PCS logic and cannot be used to arbitrate between different clock frequencies. Therefore the tclk[0-3] ports must be connected to clocks which are frequency locked to the reference clock. Typically, they are fed from the locked reference clock outputs from the PCS to the FPGA (ref_pclk or the appropriate ref_[0-3]_sclk). The phase compensation FIFOs are designed to handle any phase shift of these clocks due to internal clock routing delays. The clocks connected to the tclk[0-3] ports may not be created from a DLL to PLL combination or a PLL to PLL combination as these combinations may create short term frequency deviations that may overflow or underflow the FPGA interface phase compensation FIFOs.

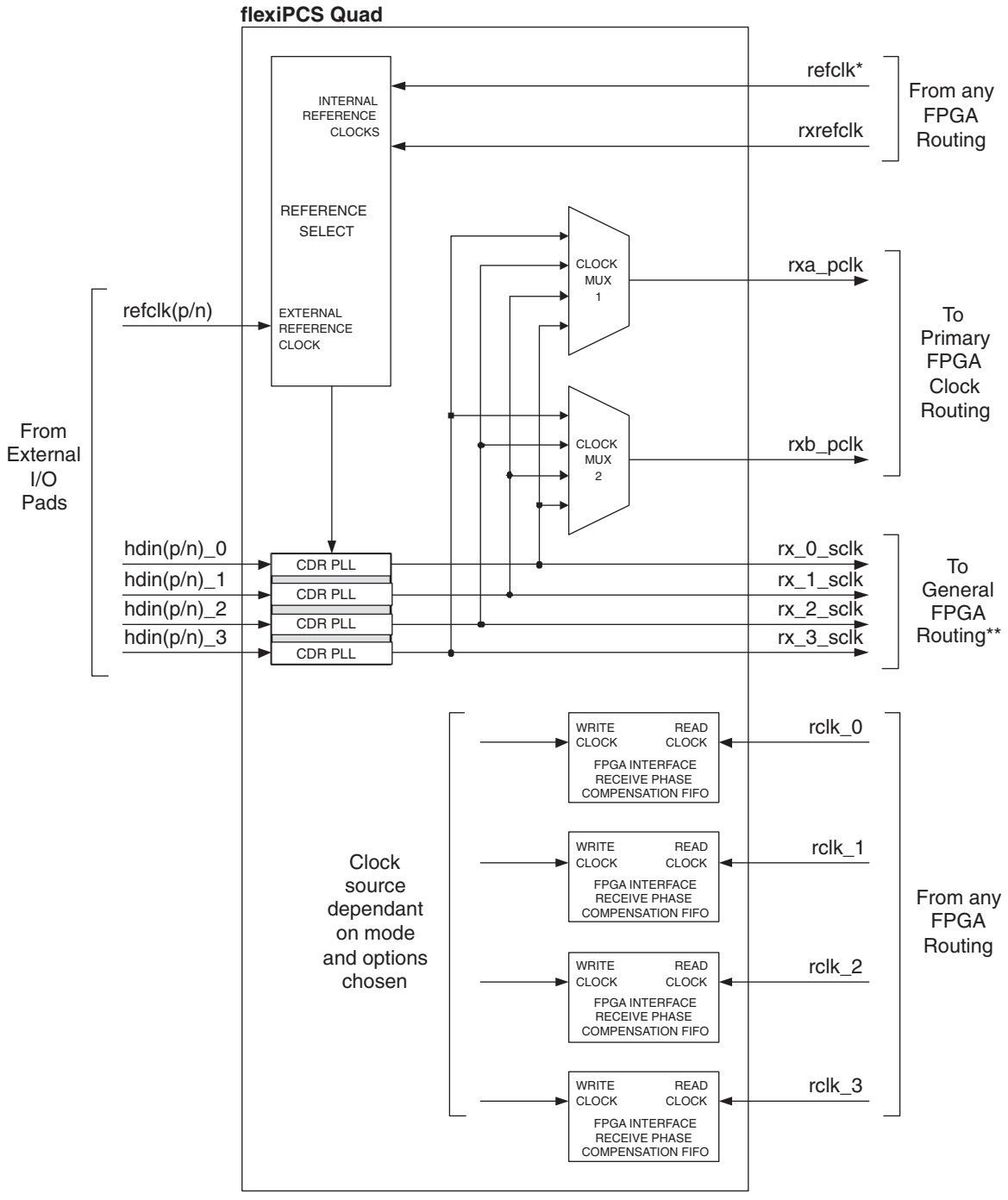
### 16/20 Bit Data Bus Mode Transmit Clock Rates

The above description of locked reference and transmit clock rates assumes that the quad transmit data path is set to “8/10 bit data bus” mode. In this case, one byte of data is to be presented to the PCS every clock cycle at the FPGA interface clock rate shown in Table 2-5 This is the default mode. The transmit data path for a quad can be set to “16/20 bit data bus” mode by writing Quad Interface Register Offset Address 0x19 bit 4 to a “1”. In this mode, two bytes of data are to be presented to the PCS from the FPGA logic at 1/2 of the FPGA interface clock rate as shown in Table 2-5. In “16/20 bit data bus” mode, the byte on the lower bits at the PCS/FPGA interface is transmitted first, then the byte on the upper bits is transmitted second.

## Receive Clocks

Figure 2-12 illustrates the relationships between the various PCS/FPGA interface receive clocks.

***Figure 2-12. Receive Clocks***




* The refclk inputs for all active quads on a device must be connected to the same clock. The rxrefclk inputs for all active quads on a device do not have to be connected to the same clock.

** Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing.

**Receive Clocks to FPGA Logic**

Each of the four receive SERDES in a quad recovers a clock from the incoming data that is then used to clock the associated receive channel flexiPCS logic. The PCS provides a recovered clock per channel (**rx_[0-3]_sclk**) to the FPGA. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. Two clocks are provided to the FPGA which are both a four-way mux of all receive channel clocks (**rxa_pclk** and **rxb_pclk**). These two clocks are provided on primary clocks.

The clock selection for **rxa_pclk** is controlled by writing to Quad Interface Register Offset Address 0x02, bits 4 and 5 as shown in Table 2-7.

***Table 2-7. Receive Clock Selection for rxa_pclk***

| Quad Interface Register Offset Address = 0x02 | | Receive Clock Selection |
|---|---|---|
| **Bit 4** | **Bit 5** | **Description** |
| 0 | 0 | Selects receive clock channel 0 |
| 0 | 1 | Selects receive clock channel 1 |
| 1 | 0 | Selects receive clock channel 2 |
| 1 | 1 | Selects receive clock channel 3 |

Note: Relevant only when multi-channel alignment not selected.

The clock selection for **rxb_pclk** is controlled by writing to Quad Interface Register Offset Address 0x02, bits 6 and 7 as shown in Table 2-8.

***Table 2-8. Receive Clock Selection for rxb_pclk***

| Quad Interface Register Offset Address = 0x02 | | Receive Clock Selection |
|---|---|---|
| **Bit 6** | **Bit 7** | **Description** |
| 0 | 0 | Selects receive clock channel 0 |
| 0 | 1 | Selects receive clock channel 1 |
| 1 | 0 | Selects receive clock channel 2 |
| 1 | 1 | Selects receive clock channel 3 |

Note: Relevant only when multi-channel alignment not selected.

Recovered Clocks are sourced from the VCO of the CDR. During a loss of lock to data, these clocks will be unstable and produce high frequency clocks. Care must be taken when using these clocks that any downstream elements using these clocks can accept a clock with variable frequency and pulse widths.

**Receive Clocks from FPGA Logic**

Each respective channel's input receive clock can be driven from the FPGA logic on pins **rclk[0-3].** These clocks serve as the read clocks to FPGA interface phase compensation FIFOs embedded in the quad PCS logic as shown in Figure 2-12.

The FPGA interface phase compensation FIFOs are intended only to provide a clean timing interface to the PCS logic and cannot be used to arbitrate between different clock frequencies. Therefore the rclk[0-3] ports must be connected to clocks which are frequency locked to the respective channel's FPGA interface receive phase compensation FIFO write clock.

In modes where Multi-channel alignment and Clock Tolerance Compensation features are unused, the input receive clocks to the FPGA Interface Receive phase compensation FIFOs shown in Figure 2-12 are typically fed from the receive clock outputs from the PCS to the FPGA (rxa_pclk, rxb_pclk, or the appropriate rx_[0-3]_sclk). The phase compensation FIFOs are designed to handle any phase shift of these clocks due to internal clock rout-

ing delays. The clocks connected to the rclk[0-3] ports may not be created from a DLL to PLL combination or a PLL to PLL combination as these combinations may create short term frequency deviations that may overflow or underflow the FPGA interface phase compensation FIFOs.

When Multi-channel alignment is active, the write clock to the FPGA interface receive phase compensation FIFOs will be driven from a selection of one of the individual channel CDR clocks. For more information on control and timing issues for Multi-channel alignment, refer to the Multi-channel Alignment section of the LatticeSC/M Family flexiPCS Data Sheet.

When Clock Tolerance Compensation is active, the write clock to the FPGA interface receive phase compensation FIFOs will be driven from the PCS locked reference clock. In this case, the rclk[0-3] must be connected to clocks which are frequency locked to the locked reference clock, specifically the **ref_pclk** or **ref_[0-3]_sclk** ports shown in Figure 2-11. For more information on control and timing issues for Clock Tolerance Compensation, refer to the Detailed Functionality section for each PCS mode.

**16/20 Bit Data Bus Mode Receive Clock Rates**

The above description of receive clock rates assumes that the quad receive data path is set to “8/10 bit data bus” mode. In this case, one byte of data passed from the PCS to the FPGA logic every clock cycle at the FPGA interface clock rate shown in Figure 2-5. This is the default mode. The receive data path for a quad can be set to “16/20 bit data bus” mode by writing Quad Interface Register Offset Address 0x19 bit 5 to a “1”. In this mode, two bytes of data are to provided by the PCS to the FPGA logic at 1/2 of the FPGA interface clock rate as shown in Table 2-5. In “16/20 bit data bus” mode, the first received byte is placed on the lower bits, and the second received byte is placed on the upper bits at the FPGA interface.

**Quad & Channel Resets**

Resets are provided to reset an entire quad (either SERDES logic only or all flexiPCS logic) or each individual channel (all PCS logic per channel). These resets are driven from the FPGA logic or control registers. A summary of the control signals provided for reset are listed below. More detail on recommended initialization and reset sequences for the SERDES is provided in the **flexiPCS Reset Sequences** section.

The following inputs to the PCS from the FPGA are enabled as long as Quad Interface Register Offset Address 0x42, bit 7 is set to ‘1’ (which is the default on powerup). Writing a ‘0’ to this bit will disable these reset inputs.

**quad_rst** - Active high, asynchronous signal from FPGA resets all SERDES and PCS logic in the quad, including all the Quad Interface Registers and the Channel Interface Registers. This reset can also be performed by writing to Quad Interface Register Offset Address 0x43, bit 7 to ‘1’. **quad_rst** also reset all flexiPCS control registers. If these registers had been previously written via the Systembus or set during configuration through an auto-configuration file, they will need to be rewritten following this reset.

**serdes_rst** - Active high, asynchronous signal from FPGA resets all SERDES (but not PCS) logic in the quad. This reset can also be performed by writing to Quad Interface Register Offset Address 0x41, bit 5 to ‘1’. Note that this bit is preset to ‘1’ on powerup and must be written to ‘0’ before operation of the SERDES can commence. **serdes_rst** resets all SERDES PLLs.

**tx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one transmit channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [4:7]. Bit 7 resets transmit channel 0, bit 6 resets transmit channel 1, bit 5 resets transmit channel 2, and bit 4 resets transmit channel 3. **tx_rst** does not reset any flexiPCS control registers.

**rx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one receive channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [0:3]. Bit 3 resets receive channel 0, bit 2 resets receive channel 1, bit 1 resets receive channel 2, and bit 0 resets receive channel 3. **rx_rst** does not reset any flexiPCS control registers.

## Transmit Data

High-speed CML serial outputs are transmitted on external pins **hdoutp[0-3]** and **hdoutn[0-3]**. By default, serial data is transmitted to the SERDES outputs with the least significant bit transmitted first.

**Transmit Data Control Registers**
The order of the transmitted bits can be reversed (most significant bit first) on a per channel basis by setting the appropriate Channel Interface Register Offset Address 0x01, bit 4 to '1'.

All transmitted bits can be inverted on a per channel basis by setting the appropriate Channel Interface Register Offset Address 0x01, bit 5 to '1'.

**Transmit Buffer Pre-emphasis and Control Registers**
Register bits can be used to control other options relevant to the SERDES output buffers.

By default, transmit pre-emphasis is disabled. Pre-emphasis can be enabled on a per-channel basis by setting the appropriate Channel Interface Register Offset Address 0x14, bit 0 to '1'.

Per channel transmit pre-emphasis levels are set by writing to the appropriate Channel Interface Register Offset Address 0x14, bits [1:3] as shown in Table 2-9

***Table 2-9. Transmitter Pre-emphasis Level***

| Channel Interface Register Offset Address = 0x14 | | | | |
|---|---|---|---|---|
| **Bit 0** | **Bit 1** | **Bit 2** | **Bit 3** | **Description** |
| 0 or 1 | 0 | 0 | 0 | Pre-emphasis level = 0% |
| 1 | 0 | 0 | 1 | Pre-emphasis level = 16% |
| 1 | 0 | 1 | 0 | Pre-emphasis level = 32% |
| 1 | 0 | 1 | 1 | Pre-emphasis level = 48% |

By default, each SERDES channel's output driver is set to 1000 mV (typical) differential swing. Per channel transmit output driver levels can be set by writing to the appropriate Channel Interface Register Offset Address 0x14, bits [6:7] as shown in Table 2-10.

***Table 2-10. Transmitter Driver Amplitude***

| Channel Interface Register Offset Address = 0x14 | | |
|---|---|---|
| **Bit 6** | **Bit 7** | **Description** |
| 0 | 0 | 1000 mV differential swing |
| 0 | 1 | -12.5% from default swing |
| 1 | 0 | -25% from default swing |
| 1 | 1 | +12.5% from default swing |

**Transmitter Termination**
By default, each SERDES channel's output transmitter termination is set to 50 ohm (typical). Per channel transmit output termination can be set by writing to the appropriate Channel Interface Register Offset Address 0x014, bits [4:5].

*Table 2-11. Transmitter Termination*

| Channel Interface Register Offset Address = 0x14 | | |
|---|---|---|
| **Bit 4** | **Bit 5** | **Description** |
| 0 | 0 | Sets termination to 50 ohm (default) |
| 0 | 1 | Sets termination to 75 ohm |
| 1 | 0 | Sets termination to 5k ohm |
| 1 | 1 | Reserved |

## Transmit Data Skew

Transmit data skew can be minimized by setting the Quad Interface Register Offset Address 0x30, bit 5 to '1'. This will ensure that the transmit data is aligned within a specified skew across all channels of a quad.

To ensure that the transmit data is aligned within a specified skew across all channels of a device; the PCS SERDES has the option of using an internal reference clock instead of an external reference clock. The internal reference clock comes from the FPGA fabric instead of an external pin. This internal path is designed to provide a balanced reference clock to all of the PCS SERDES quads in a device ensuring that transmit data is aligned within a specified skew across all channels in a device.

Setting Quad Interface Register Offset Address 0x29, bit 3 to '1' and bit 4 to '0' will select the internal reference clock. This setting coupled with setting Quad Interface Register Offset Address 0x30, bit 5 to '1' will minimize the transmit data skew across all channels in a device.

## Receive Data

High-speed CML serial inputs are received on external pins **hdinp[0-3]** and **hdinn[0-3]**. By default, serial data is received by the SERDES outputs with the least significant bit received first.

### Receive Data Control Registers

The order of the received bits can be reversed (most significant bit first) on a per channel basis by setting the appropriate Channel Interface Register Offset Address 0x01, bit 6 to ‘1’.

All receive bits can be inverted on a per channel basis by setting the appropriate Channel Interface Register Offset Address 0x01, bit 7 to ‘1’.

### Receive Loss of Signal Alarm Registers

Each receive channel has a programmable, dual threshold loss-of-signal detector. Both a low threshold and a high threshold value can be programmed for the loss of signal detector by writing to Quad Interface Register Offset Address 0x28, bits [5:7] as shown in Table 2-12.

*Table 2-12. Receiver Loss of Signal Detector Levels*

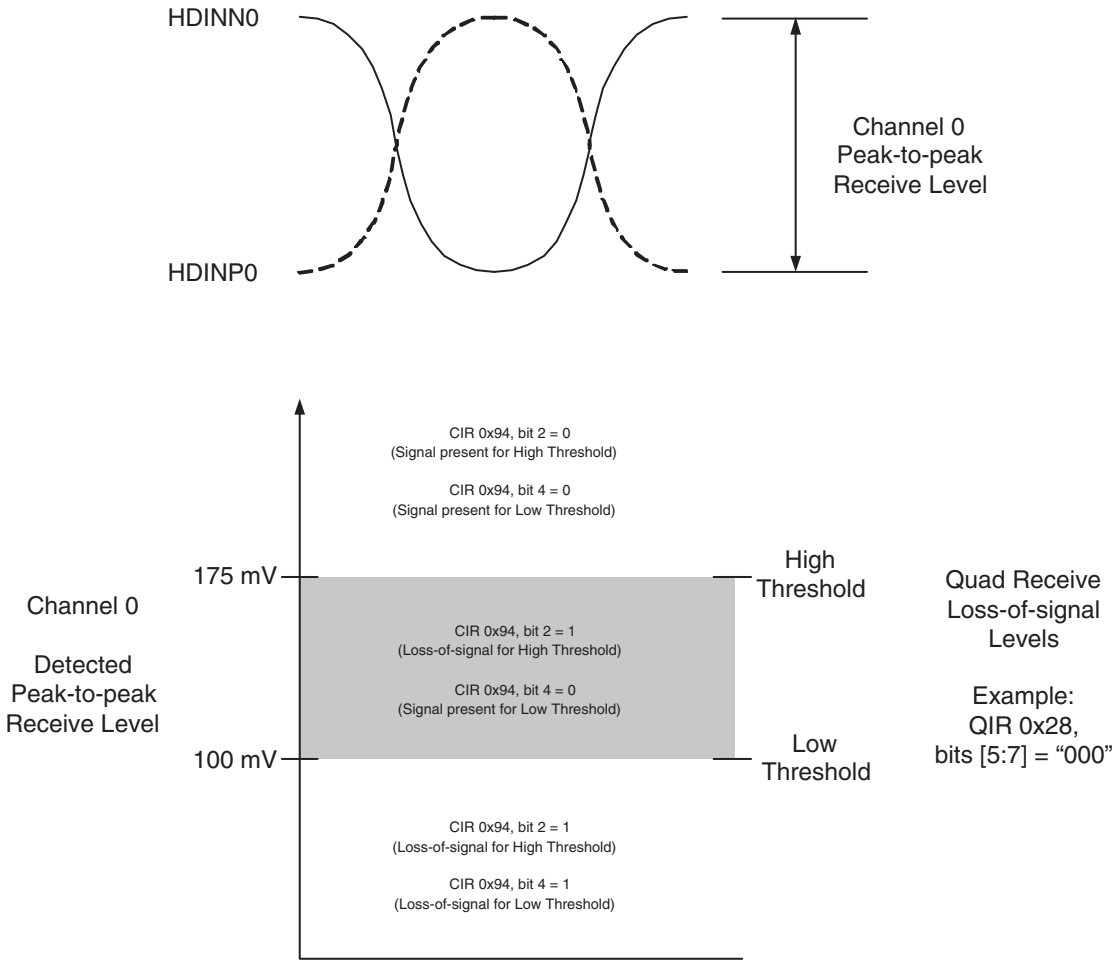
| Quad Interface Register Offset Address = 0x28 | | | Loss of Signal Low Value (Max) | Loss of Signal High Value (Max) | |
|---|---|---|---|---|---|
| Bit 5 | Bit 6 | Bit 7 | Threshold | Threshold | Units |
| 0 | 0 | 0 | 100 | 175 | mV, p-p differential |
| 0 | 0 | 1 | 105 | 186 | mV, p-p differential |
| 0 | 1 | 0 | 110 | 197 | mV, p-p differential |
| 0 | 1 | 1 | 115 | 208 | mV, p-p differential |
| 1 | 0 | 0 | 95 | 164 | mV, p-p differential |
| 1 | 0 | 1 | 90 | 153 | mV, p-p differential |
| 1 | 1 | 0 | 85 | 142 | mV, p-p differential |
| 1 | 1 | 1 | 80 | 131 | mV, p-p differential |

The receiver loss of signal (low) alarm for each channel's receiver input can be read at Channel Interface Register Offset Address 0x94, bit 2. A value of '0' indicates that a signal is present (signal input differential value is greater than the programmed threshold). A value of '1' indicates that a signal is not present (signal input differential value is less than the programmed threshold).

The receiver loss of signal (high) alarm for each channel's receiver input can be read at Channel Interface Register Offset Address 0x94, bit 4. A value of '0' indicates that a signal is present (signal input differential value is greater than the programmed threshold). A value of '1' indicates that a signal is not present (signal input differential value is less than the programmed threshold).

An example of the receiver loss-of-signal for a case where QIR 0x28 is set to "000" and the resulting loss-of-signal alarms for various detected differential peak-to-peak levels is shown in Figure 2-13.

*Figure 2-13. SERDES Receiver Loss-of-signal Detection Example*




**Receive Buffer Equalization and Control Registers**

Quad Interface Register bits can be used to control other options relevant to the SERDES input buffers.

By default, receive equalization is disabled. Receive equalization can be enabled on a per-channel basis by setting the appropriate Channel Interface Register Offset Address 0x15, bit 3 to ‘1’.

Per channel receive equalization settings are selected by writing to the appropriate Channel Interface Register Offset Address 0x15, bit 5.

*Table 2-13. Receiver Equalization Settings*

| Channel Interface Register Offset Address = 0x15 | | |
|---|---|---|
| **Bit 3** | **Bit 5** | **Equalization Level** |
| 0 | 0 or 1 | 0 dB |
| 1 | 0 | +6 dB |
| 1 | 1 | +12 dB |

**Receiver Termination**

By default, each SERDES channel's input receiver termination is set to 50 ohm. Per channel receiver input termination can be set by writing to the appropriate Channel Interface Register Offset Address 0x015, bits [6:7].

***Table 2-14. Receiver Input Termination***

| Channel Interface Register Offset Address = 0x15 | | |
|---|---|---|
| **Bit 6** | **Bit 7** | **Description** |
| 0 | 0 | Sets termination to 50 ohm (default) |
| 0 | 1 | Sets termination to 75 ohm |
| 1 | 0 | Sets termination to 2k ohm |
| 1 | 1 | Sets termination to 60 ohm |

By default, each external receiver data input is AC coupled. The approximate size of the internal coupling capacitor is 5 pf. Each channel's receive input buffer pair can be set to DC coupling by setting the appropriate Channel Interface Register Offset Address 0x15, bit 4 to '1'.

## Low Speed SERDES Receiver Operation

The SERDES Receive buffers can be used to input low speed (< 600 Mbps) by bypassing the receiver CDRs and associated flexiPCS logic. This feature is useful for applications where the same pin is needed for both high speed and low speed data transfers. The Receiver Low Speed Data (rx_lsd) ports at the PCS/FPGA interface are used to input a signal slower than 600 Mbps,. These ports are available when the SERDES Only modes are selected.

The rx_lsd inputs are passed directly from the SERDES input buffers to the FPGA interface and are not locked to the reference clock. The **rx_lsd** inputs do not toggle unless enabled. They can be enabled on a per channel basis by writing the appropriate Channel Interface Register Offset Address 0x15, bit 2 to a '1'. 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 writing the appropriate Channel Interface Register Offset Address 0x15, bit 4 to a '1'.

Figure 2-14 and Figure 2-15 show an example of SERDES Channel 0 receiver operation for both high speed and low speed inputs.

***Figure 2-14. High Speed Operation of SERDES Receiver (Channel 0 Shown)***

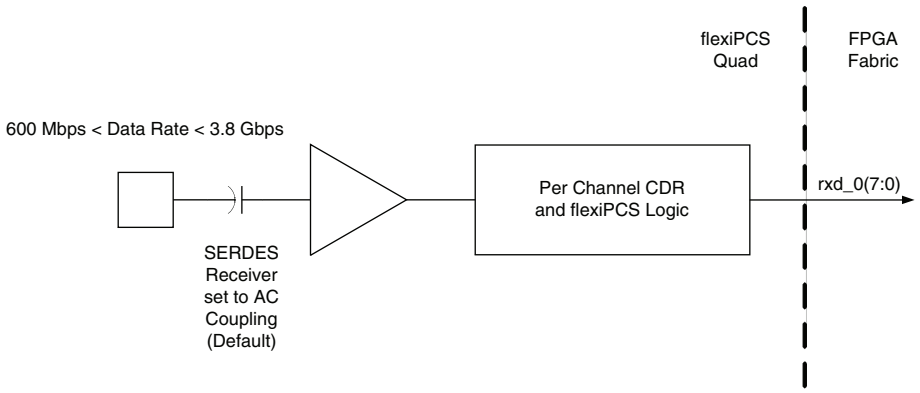
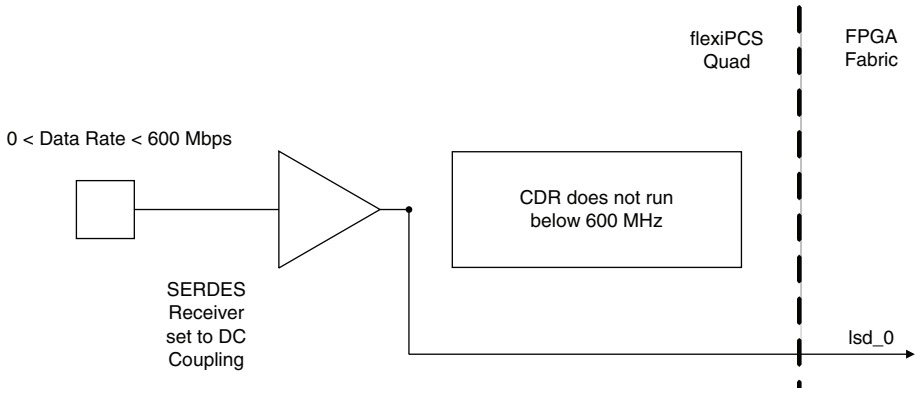


*Figure 2-15. Low Speed Operation of SERDES Receiver (Channel 0 shown)*




## Status Interrupt Registers

Some of the status registers associated with the SERDES operation have an associated status interrupt and status interrupt enable register. Any flexiPCS status interrupt register will generate an interrupt to the Systembus block (if used in the design). For a description of the operation of interrupts with the Systembus, refer to the Systembus section of the LatticeSC/M Family Data Sheet. Operation of the interrupt registers for the flexiPCS is as follows:

User must set the appropriate status interrupt enable bit to a ‘1’

Whenever the associated status bit goes high, its status interrupt bit will also go high. The status interrupt bit will remain high even if the associated status bit goes low.

The status interrupt bit will remain high until it is read, when it will automatically be cleared. If the associated status bit is still high, then the status interrupt bit will go high again (until read the next time).

Table 2-15 shows a listing of the status registers associated with PCI Express operation which have a corresponding status interrupt and status interrupt enable bit.

*Table 2-15. Status Interrupt Register Bits*

| Status Register Bit Function | Status Register Bit Address | Status Interrupt Enable Register Bit Address | Status Interrupt Register Bit Address |
|---|---|---|---|
| Reference Clock PLL Loss of Lock | QIR 0x86, bit 7 | QIR 0x3F, bit 7 | QIR 0x88, bit 7 |
| Receive CDR Loss of Signal High | CIR 0x94, bit 4 | CIR 0x17, bit 4 | CIR 0x97, bit 4 |
| Receive CDR Loss of Signal Low | CIR 0x94, bit 2 | CIR 0x17, bit 2 | CIR 0x97, bit 2 |
| Receive CDR Loss of Lock | CIR 0x94, bit 6 | CIR 0x17, bit 6 | CIR 0x97, bit 6 |
| Receive CDR has Locked to Data | CIR 0x94, bit 7 | CIR 0x17, bit 7 | CIR 0x97, bit 7 |

## Receive CDR Loss of Lock Interrupt Usage

To properly interpret the receive CDR loss of lock interrupts; the following steps should be taken:

1. Disable the receive CDR loss of lock interrupts by setting CIR 0x17, bit 6 and CIR 0x17, bit 7 to a '0' (default condition).

2. Poll register CIR 0x97, bit 6 and CIR 0x97, bit 7 until a '00' condition is achieved. Do not poll register CIR 0x97, bit 6 and CIR 0x97, bit 7 faster than the typical time to detect Loss-of-Lock as shown in Table 2-4.

3. Enable the receive CDR loss of lock interrupts by setting CIR 0x17, bit 6.

The following table decodes the status of the receive CDR loss of lock registers:

| CIR 0x97, bit 6 (rlol) | CIR 0x97, bit 7 (~rlol) | Receive CDR Status |
|---|---|---|
| 0 | 0 | Locked |
| 0 | 1 | Unlocked |
| 1 | 0 | Unlocked |
| 1 | 1 | Unlocked |

Note: Do not poll faster than the lock time indicated in Table 2-4.

## flexiPCS Reset Sequences

The intended usage of the reset controls during initial flexiPCS setup and during subsequent changes to that setup is described below.

**Initial Loading of flexiPCS Quad at Powerup**

1. Power is applied to the device.
2. Power on reset to the PCS quad is asserted automatically at initial power up.
3. The Quad Interface and Channel Interface Register control bits are cleared (take on default value), the SERDES quad is powered down as a result of this.
4. Power on reset is deasserted (waiting for internal power on reset to finish), the SERDES quad remains powered down, the Quad Interface and Channel Interface Register control bits can now be changed.
5. SERDES reset should be left asserted (reset input **serdes_rst** held high and/or leave Quad Interface Register Offset Address 0x41, bit 5 at '1' which is default on power up).
6. The SERDES setup can be loaded into the appropriate Quad Interface and Channel Interface Register control bits, or via the configuration bit stream.
7. SERDES reset is deasserted (reset input **serdes_rst** set low and set Quad Interface Register Offset Address 0x41, bit 5 to '0') and then the quad powers up in 10µs.
8. Lock is automatically acquired with valid input data. All transmit channels of the quad are automatically aligned.

**Subsequent Loading of Entire flexiPCS Quad**

1. Reset to the flexiPCS quad is asserted by driving reset input **quad_rst** high or by writing Quad Interface Register Offset Address 0x43, bit 7 to '1'.
2. The Quad Interface and Channel Interface Register control bits are cleared (take on default value), the SERDES quad is powered down as a result of this.
3. Reset to the flexiPCS quad is deasserted by driving **quad_rst** low and by setting Quad Interface Register Offset Address 0x43, bit 7 to '0'. The SERDES quad remains powered down, the Quad Interface and Channel Interface Register control bits can now be changed.
4. SERDES reset should be left asserted (reset input **serdes_rst** held high and/or leave Quad Interface Register Offset Address 0x41, bit 5 at '1' which is default on power up).
5. The SERDES setup can be loaded into the appropriate Quad Interface and Channel Interface Register control bits.

6. SERDES reset is deasserted (reset input **serdes_rst** set low and set Quad Interface Register Offset Address 0x41, bit 5 to '0') after waiting the specified time for the quad to power up, or after waiting for the minimum allowed duration of SERDES reset, whichever is longer.

7. Lock is automatically acquired with valid input data. All transmit channels of the quad are automatically aligned.

**Subsequent Changes to SERDES Portion of Quad**
Subsequent changes to the SERDES quad setup (not required for PCS register changes) are performed inside an assertion and deassertion cycle of **serdes_rst**:

1. **serdes_rst** is asserted (or Quad Interface Register Offset Address 0x41, bit 5 is set to '1').

2. The SERDES setup is loaded into the appropriate Quad Interface and Channel Interface Register control bits.

3. SERDES reset is deasserted (reset input **serdes_rst** set low and set Quad Interface Register Offset Address 0x41, bit 5 to '0') after waiting the specified time for the quad to power up (if applicable), or after waiting for the minimum allowed duration of SERDES reset, whichever is longer.

4. Lock is automatically acquired, transmit channels are automatically aligned.

**Per Channel PCS Receiver Changes**
Receiver SERDES setup changes for a single lane are performed inside an assertion and deassertion cycle of the corresponding receiver reset signal (**rx_rst_[0-3]**):

1. **rx_rst_[0-3]** is asserted (or Quad Interface Register Offset Address 0x40, bits [3:0] are set to '1').

2. The single channel receiver setup is loaded into the appropriate Channel Interface Register control bits.

3. **rx_rst_[0-3]** is deasserted (and Quad Interface Register Offset Address 0x40, bits [3:0] are set to '0').

**Per Channel PCS Transmitter Changes**
Transmitter SERDES setup changes for a single lane (such as switching between full-rate and half-rate operation) can be achieved "on-the-fly" without requiring application of any reset signals:

1. The single channel transmitter setup is loaded into the appropriate Channel Interface Register control bits.

## flexiPCS Power Down Control Signals

**Quad SERDES Power Down**
The Quad SERDES power down control bit is an active low asynchronous control bit which acts on all four channels of the appropriate quad. This is accessed by writing to Quad Interface Register Offset Address 0x28, bit 1.

When the quad SERDES power down control bit is asserted:

1. Powers down the entire SERDES quad including the transmit PLL.

2. All clocks are stopped and SERDES quad power is minimized.

**Channel Power Down (Transmit)**
The single channel transmit power down control bit is an active low asynchronous control bit which acts on the appropriate transmit channel. This is accessed by writing to Channel Interface Register Offset Address 0x13, bit 7.

When the single channel transmit power down control bit is asserted:

1. Powers down the transmit serializer in the corresponding channel only.

2. Stops the **ref_[0-3]_sclk** transmit clock in the corresponding channel only (the **ref_pclk** clock may also stop if the powered down channel's clock happens to be the clock selected).

**Channel Power Down (Receive)**

The single channel receive power down control bit is an active low asynchronous control bit which acts on the appropriate receive channel. This is accessed by writing to Channel Interface Register Offset Address 0x13, bit 6.

When the single channel receive power down control bit is asserted:

1. Powers down the CDR and deserializer in the corresponding channel only.
2. Stops the **rx_[0-3]_sclk** receive clock in the corresponding channel only (the **rxa_pclk** and **rxb_pclk** clocks may also stop if the powered down channel's clock happens to be the clock selected).

# SERDES Electrical & Timing Characteristics

## SERDES Performance

Table 2-16 specifies SERDES performance measured on devices with worst case process parameters. All SERDES specifications are for $T_J$ = 0°C to 105°C. The SERDES will start up at $T_J$ Š -40°C.

***Table 2-16. SERDES Performance***

| Description | -7 Max. | -6 Max. | -5 Max. | Units |
|---|---|---|---|---|
| Tx synchronization disabled | 3.8 | 3.8 | 3.8 | Gbps |
| Tx synchronization enabled | 3.8 | 3.7 | 3.4 | Gbps |

Note: Tx synchronization is enabled by default in all multi-channel mode applications

## SERDES High Speed Data Transmitter

Table 2-17 specifies serial data output buffer parameters measured on devices with typical and worst case process parameters and over the full range of operation conditions.

***Table 2-17. Serial Output Timing and Levels[2]***

| Parameter | Min. | Typ. | Max. | Units |
|---|---|---|---|---|
| Differential Swing (default setting)[1, 9] | -15% | 1200[6] | +15% | mV, p-p |
| Differential Swing (-12.5% setting)[1, 9] | -15% | 1100[6] | +15% | mV, p-p |
| Differential Swing (-25% setting)[1, 9] | -15% | 950[6] | +15% | mV, p-p |
| Differential Swing (+12.5% setting)[1, 9] | -15% | 1300[6] | +15% | mV, p-p |
| Differential Swing (default setting)[1, 9] | -15% | 1400[7] | +15% | mV, p-p |
| Differential Swing (-12.5% setting)[1, 9] | -15% | 1300[7] | +15% | mV, p-p |
| Differential Swing (-25% setting)[1, 9] | -15% | 1150[7] | +15% | mV, p-p |
| Differential Swing (+12.5% setting)[1, 9] | -15% | 1500[7] | +15% | mV, p-p |
| Output common mode voltage | VDDOB - 0.30 | VDDOB - 0.25 | VDDOB - 0.15 | V |
| Rise Time (20% - 80%) | 80 | 125 | 180 | ps |
| Fall Time (80% - 20%) | 80 | 125 | 180 | ps |
| Output Impedance (single ended)[3] | 45/70/4.5K | 50/75/5K | 55/80/5.5K | ohms |
| Return Loss (without package)[4] | 18 | — | — | dB |
| Full-Rate Skew between Tx channels (all quads)[5] | — | — | 200 | ps |
| Half-Rate Skew between Tx Channels (all quads)[5] | — | — | 1UI + 200ps | — |
| PCI Express Receiver Detect Threshold[8] | 1.2K | — | — | ohms |

1. Output swing measured with differential CML output buffer connected to a CML differential buffer, output buffer and input buffer termination resistors set to 50 ohm mode. Expect 20% - 30% reduction in amplitude when driving into AC coupled terminations.
2. Four transmitter pre-emphasis settings are provided: 0%, 16%, 32% and 48%.
3. The tolerance on these impedances is determined by the return loss specification.
4. Return loss = 18 dB for f <= 1.7 GHz in the 50 ohm configuration. This number allows for 3 dB of additional degradation due to packaging effects.
5. Requires Tx synchronization to be enabled (QIR 0x30 bit 5) and internal reference clock used for multi-quad channels (QIR 0x29 bit 3 = '1' and bit 4 = '0').
6. Amplitude boost mode disabled.
7. Amplitude boost mode enabled.
8. Minimum termination resistor value on the receiving PCI Express device to not be detected by the LatticeSC transmitter.
9. Refer to system level integrity simulations; enhanced HSPICE kit available; contact local sales office.

Transmit output jitter is a critical parameter to systems with high-speed data links. Table 2-18 specifies the transmitter output jitter for typical and worst case devices over the full range of operating conditions.

***Table 2-18. Channel Output Jitter[1]***

| | Description | Typ. | Max. | Units |
|---|---|---|---|---|
| 3.8Gbps[3] | Deterministic | 0.06 | 0.17 | UI, p-p |
| | Random | 0.24 | 0.28 | UI, p-p |
| | Total | 0.30 | 0.39 | UI, p-p |
| 3.125Gbps | Deterministic | 0.04 | 0.09 | UI, p-p |
| | Random | 0.22 | 0.26 | UI, p-p |
| | Total | 0.26 | 0.35 | UI, p-p |
| 2.5Gbps[2] | Deterministic | 0.04 | 0.06 | UI, p-p |
| | Random | 0.19 | 0.25 | UI, p-p |
| | Total | 0.21 | 0.28 | UI, p-p |
| 1.25Gbps | Deterministic | 0.02 | 0.03 | UI, p-p |
| | Random | 0.09 | 0.12 | UI, p-p |
| | Total | 0.10 | 0.14 | UI, p-p |
| 622Mbps | Deterministic | 0.01 | 0.01 | UI, p-p |
| | Random | 0.07 | 0.09 | UI, p-p |
| | Total | 0.08 | 0.09 | UI, p-p |

1. Typical values are measured with PRBS $2^{7-1}$ at the specified rate, FPGA logic active, REFCLK total jitter = 30 ps, x10 reference clock, wirebond packaging, VDDOB = 1.5V, all other voltages are nominal, room temperature, amplitude boost mode disabled.
2. 2.5Gbps is separately tested and passed per PCI Express v.1.0a and v.1.1 requirements.
3. For wire bond packages (256 fpBGA, 900fpBGA), 3.8 Gbps operations should only be used for chip-to-chip applications or single- connector daughter card applications with limited trace length. This restriction is not applicable to flip-chip packages.

*Note: See TN1114, Electrical Recommendations for Lattice SERDES, for information on board layout techniques to reduce SERDES jitter.*

Table 2-19 specifies the end-to-end transmit data latencies through the SERDES and flexiPCS for the specified flexiPCS modes.

***Table 2-19. flexiPCS Transmit Data Latencies***

| | Generic 8b10b | Fibre Channel | PCI Express | Serial Rapid I/O | Gigabit Ethernet | XAUI | SONET | 8b10b SERDES |
|---|---|---|---|---|---|---|---|---|
| Reference clock cycles | 9 | 12 | 12 | 10 | 10 | 10 | 9 | 5 |

## SERDES High Speed Data Receiver

Table 2-20 specifies receiver parameters measured on devices with worst case process parameters and over the full range of operation conditions.

***Table 2-20. Serial Input Data Input Specifications***

| Parameter | Min. | Typ. | Max. | Units |
|---|---|---|---|---|
| Stream of nontransitions (CID = Consecutive identical Digits) @ $10^{-12}$ BER[1] | — | — | 120 | bits |
| Differential Input Sensitivity | 80 | — | — | mV, p-p |
| Input Levels | GND | — | VCC12 + 0.3 | V |
| Input common mode range (DC coupled) | 0.6 | — | VCC12 | V |
| Input common mode range (AC coupled)[2, 3] | Note 2 | — | Note 2 | V |
| CDR lock and re-lock time[4] | — | 3000 | — | bits |
| Input Termination[5] | 45/55/70/1.8K | 50/60/75/2K | 55/65/80/2.2K | ohms |
| Return loss (without package)[6] | 18 | — | — | dB |

1. This is the number of bit times allowed without a transition on the incoming data stream when using internal DC or AC coupling.
2. When AC coupled, the input common mode range is determined by:
   (Min input level) + (Peak-to-peak input swing)/2 <= (Input common mode voltage) <= (Max input level) - (Peak-to-peak input swing)/2
3. AC coupling is used to interface to LVPECL and LVDS.
4. This is the typical number of bit times to re-lock to a new phase or frequency within +/- 300 ppm, assuming 8b10b encoded data
5. The tolerance on these impedances is determined by the return loss specification.
6. Return loss = 18 dB for f <= 1.7 Ghz in the 50 ohm configuration. This number allows for 3 dB of additional degradation due to packaging effects.

*Note: Maximum current during hot socketing is 4mA. See the Hot Socketing section of the LatticeSC/M Family Data Sheet for more details.*

### Input Data Jitter Tolerance

A receiver's ability to tolerate incoming signal jitter is very dependent on jitter type. High speed serial interface standards have recognized the dependency on jitter type and have recently modified specifications to indicate tolerance levels for different jitter types as they relate to specific protocols (e.g XAUI, FC, etc.). Sinusoidal jitter is considered to be a worst case jitter type. Table 2-21 shows receiver specifications with 10 MHz sinusoidal jitter injection.

***Table 2-21. Receiver Total Jitter Tolerance Specification***

| Input Data | Conditions | Min. | Units |
|---|---|---|---|
| Jitter Tolerance @ 3.125 Gbps, Worst Case | CJPAT data pattern | 0.73 | UI, p-p |

Note: With 600 mV differential eye, all channels operating, FPGA logic active, REFCLK jitter of 30 ps, $T_A$ = 0° C to 85° C, 1.425V to 1.575 V supply. Fibre-Channel standard (0.38 UI DDJ, 0.22 UI RJ, 0.20 UI PJ @ 10MHz).

***Table 2-22. Periodic Receiver Jitter Tolerance Specification***

| Description | Frequency (Gbps) | Condition | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|---|
| Periodic | 3.125 | CJPAT data pattern | — | — | 0.22 | UI, p-p |
| Periodic | 2.5 | CJPAT data pattern | — | — | 0.22 | UI, p-p |
| Periodic | 1.25 | CJPAT data pattern | — | — | 0.22 | UI, p-p |
| Periodic | 250 | CJPAT data pattern | — | — | 0.22 | UI, p-p |

Note: With 600 mV differential eye, all channels operating, FPGA logic active, REFCLK jitter of 30 ps, $T_A$ = 0° C to 85° C, 1.425V to 1.575V supply. Fibre-Channel standard (0.38 UI DDJ, 0.22 UI RJ, 0.20 UI PJ at 10 MHz).

Table 2-23 specifies the end-to-end receive data latencies through the SERDES and flexiPCS for the specified flexiPCS modes.

*Table 2-23. flexiPCS Receive Data Latencies*

| | Generic 8b10b | Fibre Channel | PCI Express | Serial Rapid I/O | Gigabit Ethernet | XAUI | SONET | 8b10b SERDES Only |
|---|---|---|---|---|---|---|---|---|
| PCS Rx clock cycles without multi-channel alignment | 11 | 25 | 30 | 25 | 29 | 29 | 16 | 5 |
| PCS Rx clock cycles with multi-channel alignment | 24 | 25 | 39 | 38 | 29 | 42 | 29 | N/A |

Note: In gearing mode (QIR 0x19 bit 5 = '1') add 2 to 3 clock cycles.

**SERDES External CML Reference Clock**

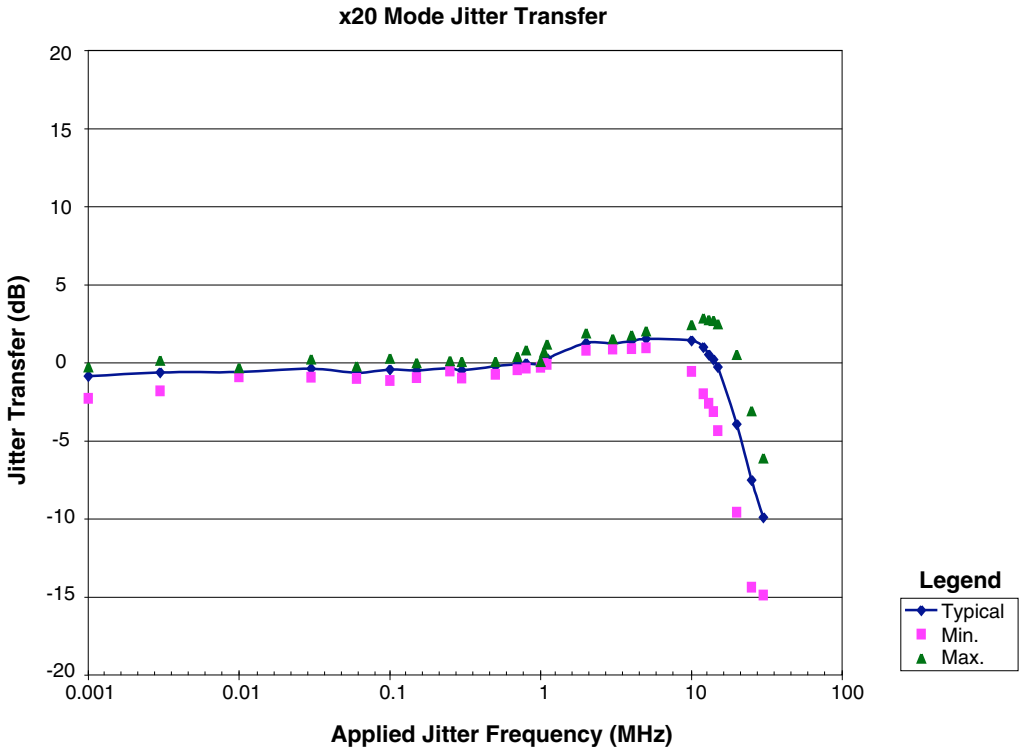
The external reference clock selection and its interface are a critical part of system applications for this product. Table 2-24 specifies reference clock requirements, over the full range of operating conditions.

*Table 2-24. External CML Reference Clock Specification (refclkp/refclkn)*

| Parameter | | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|
| Frequency Range | | 50 | — | 645 | MHz |
| Data/Reference Clock ppm Offset Tolerance[4] | 560 ppm setting | -560 | — | 560 | ppm |
| | 1870 ppm setting | -1870 | — | 1870 | |
| | 2120 ppm setting | -2120 | — | 2120 | |
| | 5900 ppm setting | -5900 | — | 5900 | |
| Input swing, differential clock | | 500 | 800 | 2 * VCC12 | mV, p-p differential |
| Input swing, single-ended clock[1] | | 500 | 800 | VCC12 | mV, p-p |
| Input levels | | 0.0 | — | VCC12 + 0.3 | — |
| Input common mode range (DC coupled) | | 0.6 | — | VCC12 | V |
| Input common mode range (AC coupled) | | Note 2 | — | Note 2 | V |
| Duty Cycle[3] | | 40 | 50 | 60 | % |
| Rise Time (20% - 80%) | | — | 500 | 1000 | ps |
| Fall Time (80% - 20%) | | — | 500 | 1000 | ps |

1. The signal swing for a single-ended input clock must be as large as the p-p differential swing of a differential input clock to get the same gain at the input receiver. Lower swings for the clock may be possible, but will tend to increase jitter.
2. When AC coupled, the input common mode range is determined by:
   (Min input level) + (Peak-to-peak input swing)/2 <= (Input common mode voltage) <= (Max input level) - (Peak-to-peak input swing)/2
3. Measured at 50% amplitude.
4. SERDES external CML reference clock frequency tolerance is register selectable from +/- 560 ppm to +/- 5900 ppm (QIR 30 bits 6 and 7).

*Figure 2-16. Jitter Transfer x20 Mode*



## Interfacing to Reference Clock CML Buffers

Some applications may require a reference clock interface. Interfacing to LVPECL requires external components as shown in the following figures. These show recommended configurations for Differential LVPECL driving CML, Single Ended PECL driving CML (DC coupled), and Single Ended PECL driving CML (AC coupled with either Receiver ended or Source ended termination).

Input reference clock levels should have a 500mV minimum differential input swing to permit the best performance for the SERDES. LVPECL signalling works well as an input reference clock to the SERDES. These levels will achieve full-swing input rise and produce low-jitter. A 100-ohm differential input termination should be located very close to the true (P) and complimentary (N) inputs of the SERDES reference clock. The best practice is to use two center-tapped 50-ohm resistors with an AC-coupled center-tap to ground.

***Figure 2-17. Differential LVPECL Driving CML Reference Clock Buffer (DC)***

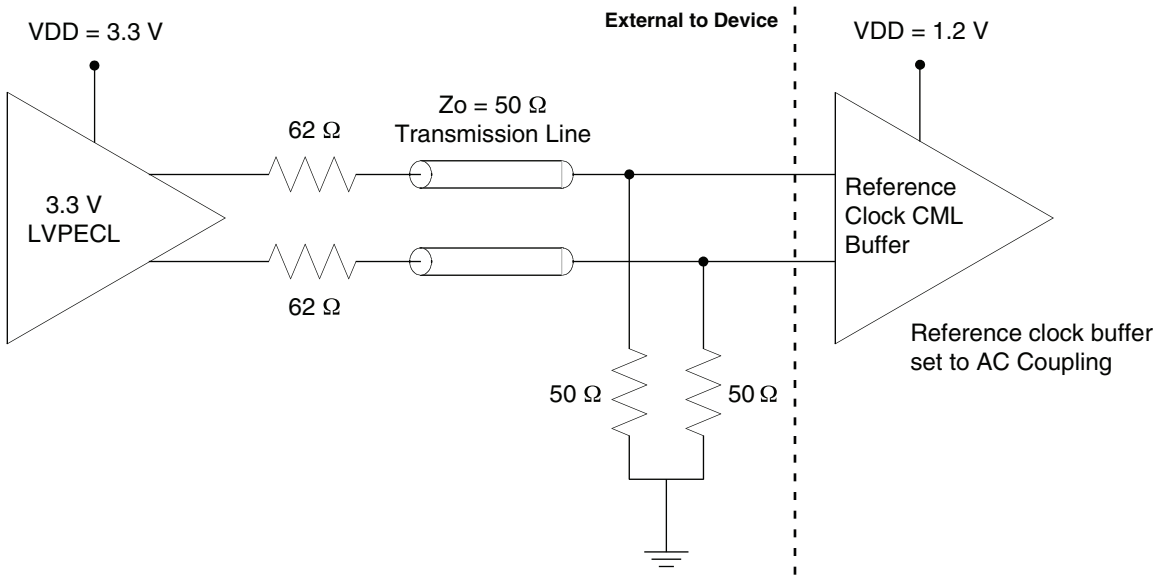



***Figure 2-18. Single Ended PECL Driving CML Reference Clock Buffer (DC)***

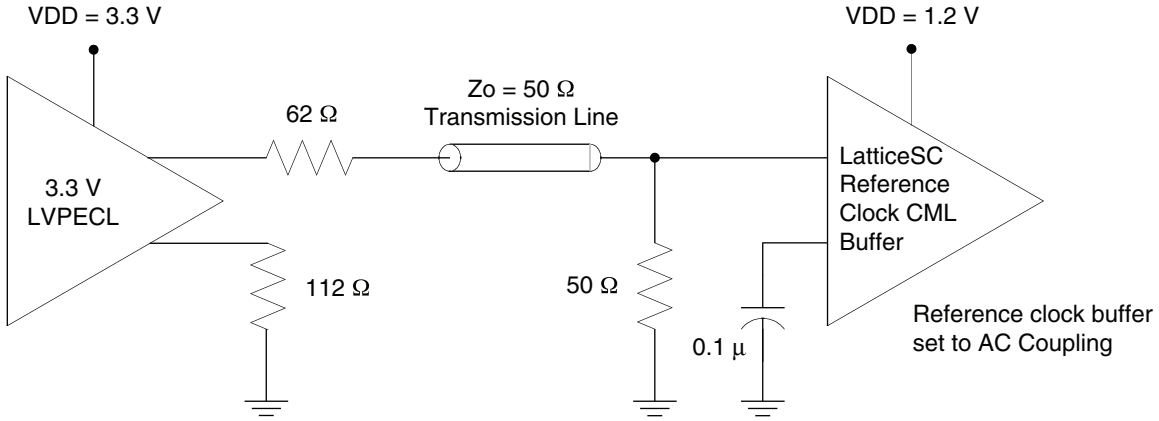


*Figure 2-19. Single Ended PECL Driving CML Reference Clock Buffer (AC, Receiver End Termination)*

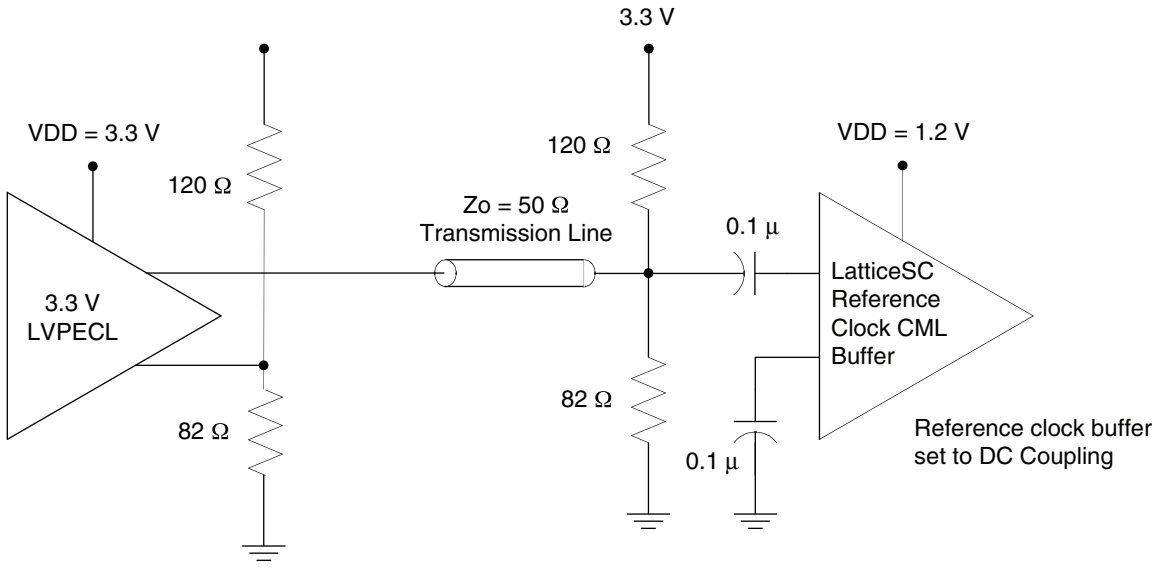
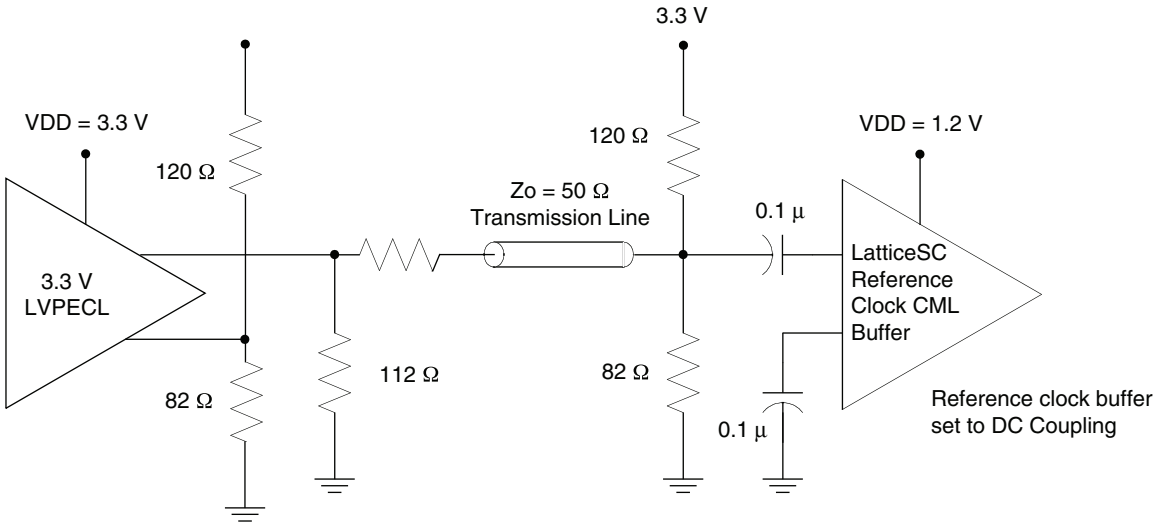



*Figure 2-20. Single Ended PECL Driving CML Reference Clock Buffer (AC, Source End Termination)*




## SERDES Power

Table 2-25 presents the SERDES power for one channel.

***Table 2-25. SERDES Power***

| Parameter | Typ. | Max. | Units |
|---|---|---|---|
| SERDES Power - VCC12 | 91 | 110 | mW |
| SERDES Power - VDDIB | 2 | 8 | mW |
| SERDES Power - VDDOB | 12 | 17 | mW |
| SERDES Power - Total | 105 | 135 | mW |

Note: One channel - one quarter of the total quad power (includes contribution from common circuits), all power supplies = 1.2V, 25°C ambient, all channels in the quad operating at 3.125 Gbps, pre-emphasis disabled, equalization enabled, amplitude boost mode disabled, 50 ohm termination on VDDIB and VDDOB.

### SERDES Power Supply Sequencing Requirements

When using the SERDES with 1.5V VDDIB or VDDOB supplies, 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 supples and voltage regulators is not a concern.

## SERDES Power Supply Requirements

All SERDES standby power is included in the full-chip VCC12 power supply. VDDIB and VDDOB power is negligible when the SERDES is powered down. Table 2-25 shows the typical SERDES power supply requirements. Note that VDDIB and VDDOB power may vary depending on how the SERDES is used in a system.

## SERDES Power Supply Package Requirements

***Table 2-26. SERDES Power Supply Package Requirements***

| Signal Name | Package Ball Requirements |
|---|---|
| VCC12[1] | Individual package balls |
| VDDOB[0-3][2] | Individual package balls, one per channel (four per quad) |
| VDDIB[0-3][2] | Individual package balls, one per channel (four per quad) |
| VDDAX25[3] | One per side, one package ball per corner, two per LatticeSC device |
| RESP | One per side, one package ball per corner, two per LatticeSC device |
| RESPN | One per side on select packages, one package ball per corner, two per LatticeSC device. |

1. All VCC12 pins need to be connected on all devices independent of functionality used on the device. This analog supply is used by both the RX and TX portions of the SERDES and is used to control the core SERDES logic regardless of the SERDES being used in the design.
2. VDDIB and VDDOB are used as supplies for the terminations on the CML input and output buffers. If a particular channel is not used, these can remain UNCONNECTED (floating).
3. VDDAX25 needs to be connected independent of the use of the SERDES. This supply is used to control the SERDES CML I/O regardless of the SERDES being used in the design.

## Power-Down/Power-Up Specification

***Table 2-27. Power-Down and Power-Up Specification***

| Symbol | Description | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|
| $t_{PWRDN}$ | Power-down time after all power down register bits[1] set to '0' | — | — | 10 | us |
| $t_{PWRUP}$ | Power-up time after all power down register bits set to '1' | — | — | 10 | us |

1. SERDES power down register bit is QIR 0x28 bit 1, receive channel power down register bit is CIR 0x13 bit 6, and transmit power down register bit is CIR 0x13 bit 7.

# 8-bit and 10-bit SERDES Only Modes LatticeSC/M flexiPCS Family Data Sheet

December 2008 Data Sheet DS1005

## Introduction

This data sheet describes the operation two modes of the LatticeSC PCS, 8-bit SERDES Only and 10-bit SERDES Only. The SERDES Only modes of the flexiPCS (Physical Coding Sublayer) block are intended for applications requiring access to a high-speed I/O interface without the protocol based manipulation provided in the LatticeSC PCS logic. The LatticeSC flexiPCS can support SERDES rates up to 3.8 Gbps per channel.

The LatticeSC flexiPCS in 8-bit or 10-bit SERDES Only modes supports the following operations:

**Transmit Path (From LatticeSC device to line):**

- Conversion of 8 or 10 bit parallel data to serial at rates from 600 Mbps to 3.8 Gbps per channel

**Receive Path (From line to LatticeSC device):**

- Conversion of serial data to 8 or 10 bit parallel data at rates from 600 Mbps to 3.8 Gbps per channel
- Optional word alignment to user defined alignment pattern (10-bit SERDES operation only)
- Low speed data ports for data inputs ranging from 600 Mbps down to DC levels (no serial to parallel conversion)

### LatticeSC flexiPCS Quad Module

Devices in the LatticeSC family have up to 8 quads of embedded flexiPCS logic. Each quad in turn supports 4 independent full-duplex data channels. A single channel can support a fully compliant SERDES Only data link and each quad can support up to four such channels. Note that mode selection is done on a per quad basis. Therefore, a selection of a SERDES Only mode for a quad dedicates all four channels in that quad to that SERDES Only mode.

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

Operation of the SERDES requires the user to provide a reference clock (or clocks) to each active quad to provide a synchronization reference for the SERDES PLLs. Reference clocks are shared among all four channels of a quad. If the transmit and receive frequencies are within the specified ppm tolerance for the LatticeSC SERDES (See DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet), only one reference clock for both transmit and receive is needed for both transmit and receive directions. If the receive frequency is not within the specified ppm tolerance for the LatticeSC SERDES (See the DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet) of the transmit frequency, then separate reference clocks for the transmit and receive directions must be provided.

Simultaneous support of different (non-synchronous) high-speed data rates is possible with the use of multiple quads. Multiple frequencies that are exact integer multiples can be supported on a per channel basis through use of the full-rate/half-rate mode options (see the **SERDES Functionality section** for more details).

## Quad and Channel Option Control

Although the mode selection and reference frequency covers an entire quad, many options covering clocking and data formatting are available on a per channel basis. These options are detailed in the following SERDES Only Mode description. Some of these options can be controlled directly through the FPGA interface pins made available on the SERDES Only Quad Module.

Other options are available only through dedicated registers that can be written and read via the embedded System Bus Interface (see the System Bus section for more details on the operation of the System Bus), or during device configuration. Depending on the specific option, a register may affect operation of multiple quads, a single

© 2008 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

quad or an individual channel in a quad. Registers dedicated to multiple quads are listed in the Inter-quad Interface Register Map in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet. Registers dedicated to one quad are listed in the Quad Interface Register Map. Registers dedicated to a single channel are listed in the Channel Interface Register Map.

## Register Map Addressing

Figure 3-1 provides a map of base register addresses for any of the flexiPCS quads on a LatticeSC device. Note that different devices may have different numbers of available quads up to a total of 8 quads (or 32 channels) maximum.

Inter-quad Interface, Quad Interface, and Channel Interface Registers are listed by Offset Address. Each Inter-quad Interface Register, Quad Interface Register, and Channel Interface Register has a unique 18-bit address determined by the specific quad or channel targeted. The addressing of Inter-quad Interface Registers, Quad Interface Registers, and Channel Interface Registers is shown Figure 3-1.

Base addresses are dependant on the physical location of a given quad or channel on a LatticeSC device. Each quad and channel has an associated set of control and status registers. These registers are referred throughout this data sheet by their offset addresses. The unique 18-bit address of a control or status register for a specific quad or channel is simply the offset address of that register added to the base address for that specific quad or channel as shown in Figure 3-1.

Channel Interface Registers can be written in one of three methods. One method is to write to one specific register corresponding to one specific channel. The other two methods involve writing to multiple channel registers with just one write operation. This is referred to as a Broadcast write. A Broadcast write is useful when multiple channel registers are to be written with the same value. The first method of Broadcast writing involves writing to all four channel registers in a quad at one time. A second method of Broadcast writing involves writing to all channel registers for all quads on the left or right side of the device at one time. Therefore, a specific Channel Interface Register may be accessed through any one of three channel addresses, depending on the number of channel registers being written to at any one time.

A broadcast write to multiple quads cannot be used to power on SERDES in any device-package combination that does not have all SERDES quads bonded because of excess power on the board and jitter is also affected.

Throughout this document, the byte-wide data at each offset address is listed with a hexadecimal value with bit 0 as the MSB and bit 7 as the LSB as per the PowerPC convention. For example if the value for Quad Interface Register Offset Address 0x02 is stated to be 0x15, the bit pattern defined is as shown in Table 3-1:

***Table 3-1. Control/Status Register Bit Order Example***

| **Quad Interface Register Offset Address 0x02 = 0x15** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Data Bit 0 | Data Bit1 | Data Bit 2 | Data Bit 3 | Data Bit 4 | Data Bit 5 | Data Bit 6 | Data Bit 7 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| **1** | | | | **5** | | | |

A full list of register Offset Addresses is provided in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet.

***Figure 3-1. flexiPCS Inter-quad, Quad, and Channel Interface Register Map Base Addressing***

## Auto-Configuration

Initial register setup for each flexiPCS mode can be performed without accessing the system bus by using the auto-configuration feature in ispLEVER. IPexpress 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 chan-

nel registers will be set to values defined in the auto-configuration file during configuration. The system bus is therefore not needed if all quads are to be set via auto-configuration files. However, the system bus must be included in a design if the user needs to change control registers or monitor status registers during operation. Note that a quad reset (Quad Interface Register 0x43, bit 7 or the **quad_rst** port at the FPGA interface) will reset all registers back to their default (not auto-configuration) values. If a quad reset is issued, the flexiPCS registers need to be rewritten via the Systembus.

## 8-bit SERDES Only Register Settings

The auto-configuration file sets registers that perform the following operations. If the auto-configuration file is not used, the appropriate registers need to be programmed via the Systembus. For more options, refer to the flexiPCS register map in the **Memory Map** section of the **LatticeSC/M Family flexiPCS Data Sheet**.

**A flexiPCS quad can be set to 8-bit SERDES Only mode by writing Quad Interface Register Offset Address 0x18 bits[0:7] to 0x60.**

These register values automatically sets up the following options in the flexiPCS for the given quad. Details on the functionality of each chosen option is provided in the **SERDES Only Mode Detailed Description** section.

**Transmit Path**

- 8-bit parallel to serial conversion is enabled

**Receive Path**

- 8-bit serial to parallel conversion is enabled

Other register settings are required to operate a channel or channels in 8-bit SERDES Only Mode. The following is a list of options that must be set for SERDES Only operation. More detail for each option is included in the **SERDES Only Mode Detailed Description** section.

- Powerup of appropriate quad and channel. Quad powerup is chosen by writing the appropriate Quad Interface Register Offset Address 0x28, bit 1 to '1'. Channel powerup is chosen by writing the appropriate Channel Interface Register Offset Address 0x13, bit 6 (receive direction) and/or bit 7 (transmit direction) to '1'. By default, all channels and quads are powered down.
- Setting SERDES reset low at end of flexiPCS setup. By default, the SERDES reset is set to '1' (SERDES are reset). The SERDES reset can be set low by writing Quad Interface Register Offset Address 0x41, bit 5 to a '0'. For more information on proper flexiPCS powerup and reset sequencing, refer to the **flexiPCS Reset Sequences** and **flexiPCS Power Down Control Signals** descriptions in the LatticeSC/M Family flexiPCS Data Sheet **SERDES Functionality** section.
- Enabling the low speed data port to the FPGA logic.

## 8-bit SERDES Only Auto-Configuration Example

An example of an auto-configuration file for a quad set to 8-bit SERDES Only Mode with only channel 1 enabled is shown in Figure 3-2. Format for the auto-configuration file is

register type (quad=quad register, ch1=channel 1 register), offset address in hex, register value in hex, # comment

***Figure 3-2. 8-bit SERDES Only Auto-Configuration Example File***

```
quad 18 60      #  Set quad to 8-bit SERDES Only Mode
quad 29 01      #  Set reference clock select
quad 28 40      #  Set bit clock multiplier to 8X (for half rate mode), quad powerup set to '1'
quad 02 15      #  Set ref_pclk, rxa_pclk and rxb_pclk to channel 1 clocks
quad 19 00      #  Set FPGA interface data bus width to 8-bit
ch1 13 03       #  Powerup channel 1 transmit and receive directions
ch1 14 90       # 16% pre-emphasis on SERDES output buffer
ch1 15 10       # +6 dB equalization on SERDES input buffer
quad 41 00      # de-assert serdes_rst
quad 40 ff      # assert datapath reset for all channels
quad 40 00      # de-assert datapath reset for all channels
```

*Note: The last three lines must appear last in the autoconfig file. These lines apply the correct reset sequence to the PCS block upon bitstream configuration.*

## 10-bit SERDES Only Register Settings

The auto-configuration file sets registers that perform the following operations. If the auto-configuration file is not used, the appropriate registers need to be programmed via the Systembus. For more options, refer to the flexiPCS register map in the Memory Map section of the LatticeSC/M Family flexiPCS Data Sheet.

A flexiPCS quad can be set to 10-bit SERDES Only mode by writing Quad Interface Register Offset Address 0x18 bits[0:7] to 0x10 and setting the appropriate Channel Interface Register Offset Address 0x05 to 0x03.

These register values automatically set up the following options in the flexiPCS for the given quad. Details on the functionality of each chosen option is provided in the SERDES Only Mode Detailed Description section.

**Transmit Path**

- 10-bit parallel to serial conversion is enabled

**Receive Path**

- 10-bit serial to parallel conversion is enabled
- Optional word alignment capability can be enabled

Other register settings are required to operate a channel or channels in 10-bit SERDES Only Mode. The following is a list of options that must be set for SERDES Only operation. More detail for each option is included in the SERDES Only Mode Detailed Description section.

- Powerup of appropriate quad and channel. Quad powerup is chosen by writing the appropriate Quad Interface Register Offset Address 0x28, bit 1 to ‘1’. Channel powerup is chosen by writing the appropriate Channel Interface Register Offset Address 0x13, bit 6 (receive direction) and/or bit 7 (transmit direction) to ‘1’. By default, all channels and quads are powered down.
- Setting SERDES reset low at end of flexiPCS setup. By default, the SERDES reset is set to ‘1’ (SERDES are reset). The SERDES reset can be set low by writing Quad Interface Register Offset Address 0x41, bit 5 to a ‘0’. For more information on proper flexiPCS powerup and reset sequencing, refer to the flexiPCS Reset Sequences

and flexiPCS Power Down Control Signals descriptions in the LatticeSC/M Family flexiPCS Data Sheet SERDES Functionality section.

- Enabling the low speed data port to the FPGA logic.
- Word alignment character(s), such as a comma if 10-bit SERDES Mode with word alignment mode

### 10-bit SERDES Only Auto-Configuration Example

An example of an auto-configuration file for a quad set to 10-bit SERDES Only Mode with only channel 1 enabled, set to half rate (10X), is shown in Figure 3-3. Format for the auto-configuration file is

register type (quad=quad register, ch1=channel 1 register), offset address in hex, register value in hex, # comment

***Figure 3-3. 10-bit SERDES Only with Word Alignment Auto-Configuration Example File***

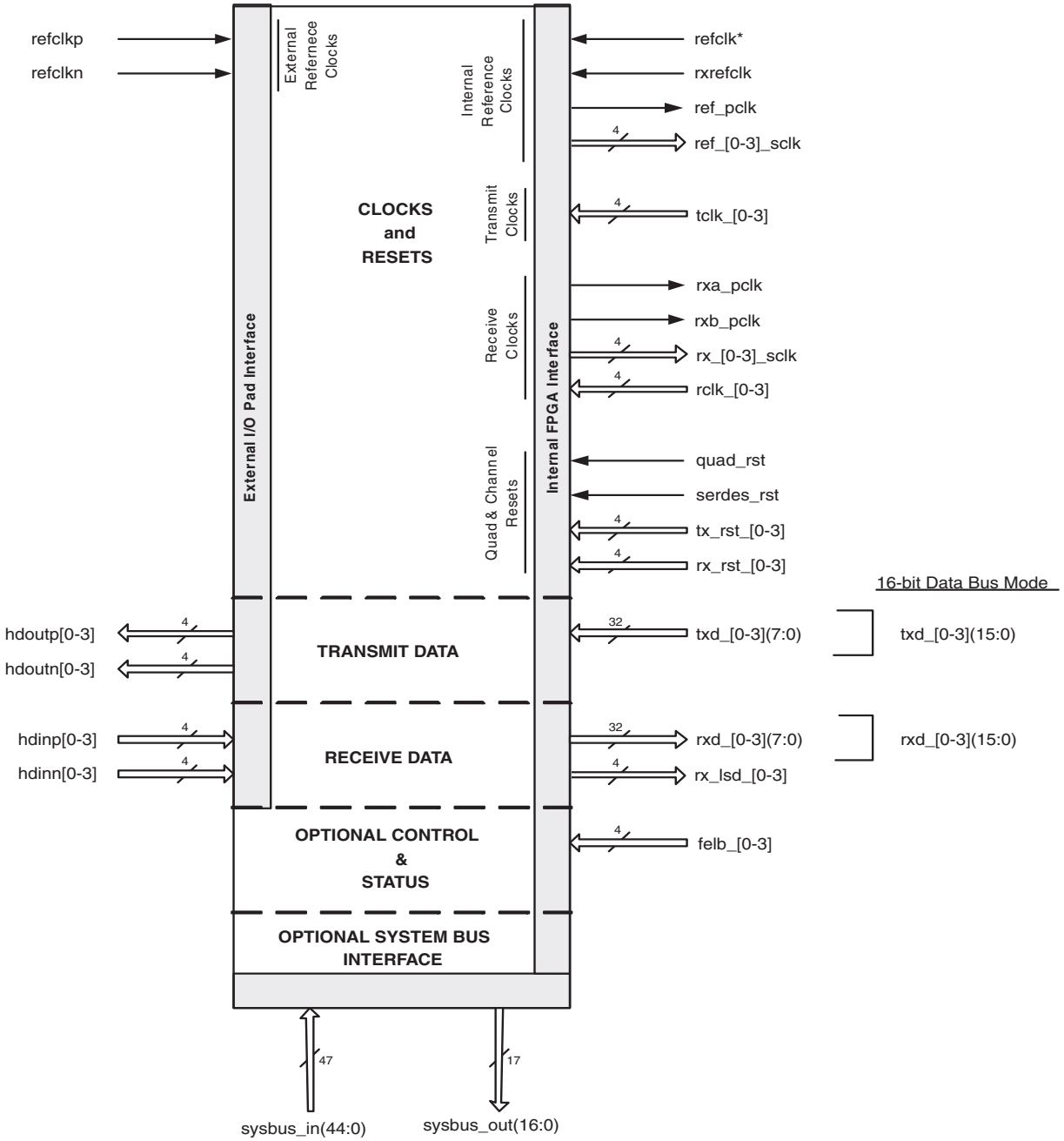
```
quad 18 10      #  Set quad to 8b10 mode (when disabled it will be 10-bit SERDES only)
quad 29 01      #  Set reference clock select
quad 28 40      #  Set bit clock multiplier to 10X (for half rate mode), quad powerup set to '1'
quad 02 15      #  Set ref_pclk, rxa_pclk and rxb_pclk to channel 1 clocks
quad 19 80      #  Set FPGA interface data bus width to 10-bit, enable external word aligner unlock control
quad 14 00      #  Set word alignment mask to always match all word alignment bits
quad 15 03      #  Set positive disparity comma value
quad 16 7C      #  Set negative disparity comma value
quad 41 00      # De-assert serdes_rst
ch1 05 03       # 8b10b disabled
ch1 13 0F       #  Powerup channel 1 to Half Rate
ch1 14 90       # 16% pre-emphasis on SERDES output buffer
ch1 15 10       # +6 dB equalization on SERDES input buffer
quad 41 00      # de-assert serdes_rst
quad 40 ff      # assert datapath reset for all channels
quad 40 00      # de-assert datapath reset for all channels
```

*Note: The last three quad lines must appear last in the autoconfig file. These lines apply the correct reset sequence to the PCS block upon bitstream configuration.*

## 8-bit SERDES Only Mode

IPexpress allows the designer to choose the mode for each PCS quad used in a given design. Figure 3-4 displays the resulting port interface to one PCS quad that has been set to 8-bit SERDES Only mode on all four channels.

***Figure 3-4. 8-bit SERDES Only Mode Pin Diagram***




* The refclk inputs for all active quads on a device must be connected to the same clock. The rxrefclk inputs for all active quads on a device do not have to be connected to the same clock.

## 8-bit SERDES Only Mode Pin Description

Table 3-2 lists all inputs and outputs to/from a PCS quad in 8-bit SERDES Only mode. A brief description for each port is given in the table. A more detailed description of the function of each port is given in the **8-bit SERDES Only Mode Detailed Description**.

***Table 3-2. 8-bit SERDES Only Clocks & Resets***

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Reference Clocks** | | | |
| refclkp | In from I/O pad | | Reference clock input, positive. Dedicated CML input. |
| refclkn | In from I/O pad | | Reference clock input, negative. Dedicated CML input |
| refclk | In from FPGA | | Optional reference clock input from FPGA logic. Can be used instead of I/O pin reference clock.<br><br>The refclk inputs for all active quads on a device must be connected to the same clock. |
| rxrefclk | In from FPGA | | Optional reference clock input from FPGA logic. Can be used instead of I/O pin reference clock. The rxrefclk inputs for all active quads do not have to be connected to the same clock. |
| ref_pclk | Out to FPGA | | Locked reference clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| ref_[0-3]_sclk | Out to FPGA | | Per channel locked reference clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| **Transmit Clocks** | | | |
| tclk_[0-3] | In from FPGA | | Per channel transmit clock inputs from FPGA. Used to clock the TX data phase compensation FIFO with clock synchronous to the reference clock. May also be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Receive Clocks** | | | |
| rxa_pclk | Out to FPGA | | Recovered receive clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| rxb_pclk | Out to FPGA | | Recovered receive clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| rx_[0-3]_sclk | Out to FPGA | | Per channel receive clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| rclk_[0-3] | In from FPGA | | Per channel receive clock inputs from FPGA. Used to clock the RX data phase compensation FIFO with a clock synchronous to the reference and/or receive reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Resets** | | | |
| quad_rst | In from FPGA | | Active high, asynchronous reset for all channels of SERDES and PCS logic. |
| serdes_rst | In from FPGA | | Active high, asynchronous reset for all channels of the SERDES. |
| tx_rst_[0-3] | In from FPGA | | Per channel active high, asynchronous reset of individual transmit channel of PCS logic. |
| rx_rst_[0-3] | In from FPGA | | Per channel active high, asynchronous reset of individual receive channel of PCS logic. |
| **Transmit Data** | | | |
| hdoutp[0-3] | Out to I/O pad | | High-speed CML serial output, positive. |

*Table 3-2. 8-bit SERDES Only Clocks & Resets (Continued)*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| hdoutn[0-3] | Out to I/O pad | | High-speed CML serial output, negative. |
| txd_[0-3](7:0)<br><br>txd_[0-3](15:0)* | In from FPGA | | Per channel parallel transmit data bus from the FPGA. Bus is 8 bits wide if QIR 0x19, bit 4 = '0'.<br><br>*Bus is 16 bits wide if QIR 0x19, bit 4 = '1'. |
| **Receive Data** | | | |
| hdinp[0-3] | In from I/O pad | N/A | High-speed CML serial input, positive. |
| hdinn[0-3] | In from I/O pad | N/A | High-speed CML serial input, negative. |
| rxd_[0-3](7:0)<br><br>rxd_[0-3](15:0)* | Out to FPGA | rclk_[0-3] | Per channel parallel receive data bus to the FPGA. Bus is 8 bits wide if QIR 0x19, bit 5 = '0'.<br><br>*Bus is 16 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_lsd_[0-3] | Out to FPGA | ASYNC | Per channel serial low speed data to the FPGA. |
| **Optional Control & Status** | | | |
| felb_[0-3] | In from FPGA | ASYNC | Per channel active high far end loopback enables. |
| **Optional System Bus Interface** | | | |
| sysbus_in(44:0) | In from FPGA | N/A | Control and data signals from the internal system bus. |
| sysbus_out(16:0) | Out to FPGA | N/A | Control and data signals to the internal system bus. |

# 8-bit SERDES Only Mode Detailed Description

The following section provides a detailed description of the operation of a PCS quad set to 8-bit SERDES Only mode. The functional description is organized according to the port listing shown in Figure 3-4. The System Bus Offset Address for any control and status registers relevant to a given function are provided with the description of the operation of that function.

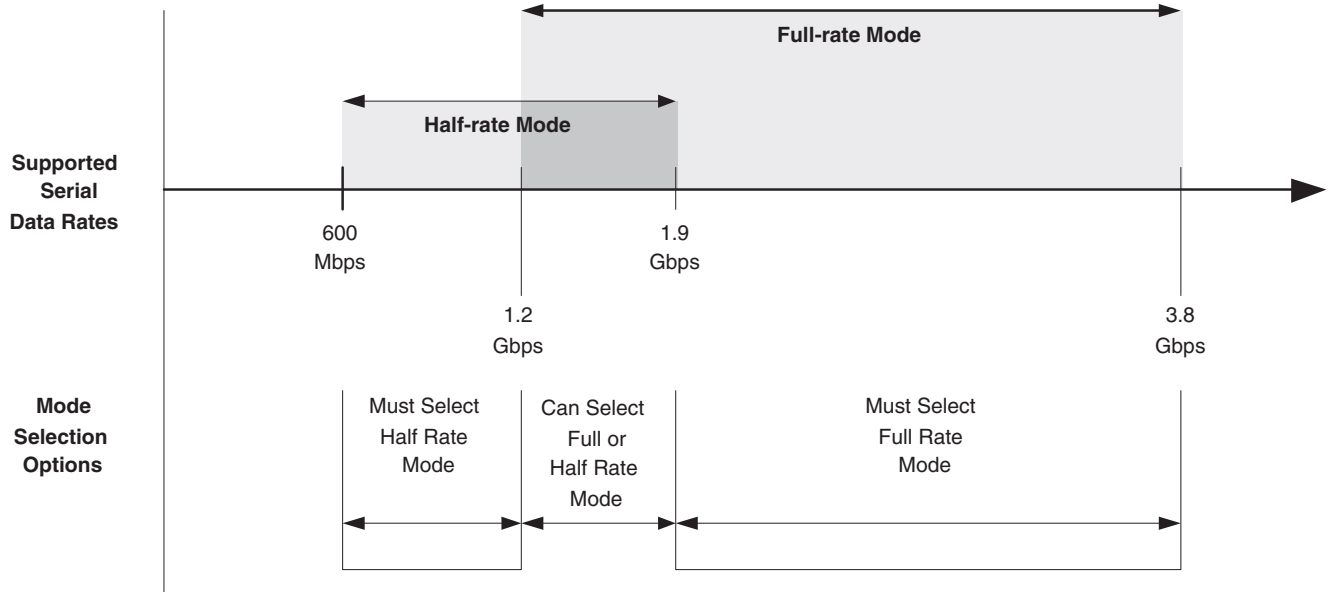
## Clocks & Resets

A detailed description of all SERDES clock, data, and reset ports and recommended reset sequencing is provided in the SERDES Functionality section of the flexiPCS Data Sheet.

### Full-Rate and Half-Rate Modes

The SERDES PLLs support data rates from 1.2 Gbps to 3.8 Gbps. A half-rate mode is available to permit operation of the SERDES at industry standard protocol data rates lower than 1.2 Gbps. The combination of half-rate and full-rate modes allow operation of a single SERDES channel at any data rate from 600 Mbps to 3.8 Gbps as shown in Figure 3-5.

*Figure 3-5. Full-Rate and Half-Rate Mode Selection*




Each of the four receive channels can be independently configured to receive data at either full-rate or half-rate. Similarly each of the four transmit channels can be independently configured to transmit data at either full-rate or half-rate. The truth table shown in Table 3-3 for transmit and receive channels describes the reference clock and data rate relationship during full and half-rate operation.

*Table 3-3. Full-Rate and Half-Rate Mode Operations*

| Rate Mode | Reference Clock Mode | | | |
|---|---|---|---|---|
| Channel Interface Register Offset Address = 0x13<br><br>Transmit - Bit 5<br>Receive - Bit 4 | Quad Interface Register Offset Address = 0x28,<br><br>Bits [2:3] | Internal Serial (bit) Clock Multiplier[1] | FPGA Interface Clock Multiplier<br><br>Quad Interface Register Offset Address 0x19<br><br>(See Note 1) | |
| | | | 8 Bit Data Bus Mode<br><br>Transmit - Bit 4 = '0'<br>Receive - Bit 5 = '0' | 10 Bit Data Bus Mode<br><br>Transmit - Bit 4 = '1'<br>Receive - Bit 5 = '1' |
| 0<br><br>Full rate<br>(1.2 Gbps - 3.8 Gbps) | 00 | 16X | 2X | 1X |
| | 01 | 8X | 1X | (1/2)X |
| | 10 | 4X | (1/2)X | (1/4)X |
| 1<br><br>Half rate<br>(600 Mbps - 1.9 Gbps) | 00 | 8X | 1X | (1/2)X |
| | 01 | 4X | (1/2)X | (1/4)X |
| | 10 | 2X | (1/4)X | (1/8)X |

1. All clock multiplier values are with respect to supplied reference clock frequency.

There are also frequency dependent SERDES performance control bits that are not writable via the Systembus. These control bits are set in the auto-configuration file created within IPexpress and are passed into the bitstream through the normal FPGA design flow (map, par, bitgen). To change the bit rate for a SERDES, specify the proper

values for the affected flexiPCS quad in IPexpress and regenerate the auto-configuration file. Failure to do this may result in non-optimal SERDES operation.

For more information on the selection of full-rate and half-rate modes, refer to the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

**Quad & Channel Resets**

Resets are provided to reset an entire quad (either SERDES logic only or all flexiPCS logic) or each individual channel (all flexiPCS logic per channel). These resets are driven from the FPGA logic or through a Systembus register. A summary of the control signals provided for reset are listed below. More detail on recommended initialization and reset sequences for the SERDES is provided in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

The following inputs to the flexiPCS from the FPGA are enabled as long as Quad Interface Register Offset Address 0x42, bit 7 is set to '1' (which is the default on powerup). Writing a '0' to this bit will disable these reset inputs.

**quad_rst** - Active high, asynchronous signal from FPGA resets all SERDES and PCS logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x43, bit 7 to '1'. quad_rst also resets all flexiPCS control registers. If these registers had been previously written via the Systembus or set during configuration through an auto-configuration file, they will need to be rewritten following this reset.

**serdes_rst** - Active high, asynchronous signal from FPGA resets all SERDES (but not PCS) logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x41, bit 5 to '1'.Note that this bit is preset to '1' on powerup and must be written to '0' before operation of the SERDES can commence.

**tx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one transmit channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [4:7]. Bit 7 resets transmit channel 0, bit 6 resets transmit channel 1, bit 5 resets transmit channel 2, and bit 4 resets transmit channel 3.

**rx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one receive channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [0:3]. Bit 3 resets receive channel 0, bit 2 resets receive channel 1, bit 1 resets receive channel 2, and bit 0 resets receive channel 3.

## Transmit Data

By default, serial data is transmitted to the SERDES outputs with the least significant bit transmitted first. Transmit data from the FPGA logic is passed to the PCS on the **txd_[0-3]** pins. The **txd** bus is 8-bits wide in normal mode and 16-bits wide when in "16-bit data bus mode". High-speed serial outputs are transmitted on external pins **hdoutp[0-3]** and **hdoutn[0-3]**. Timing for transmit data for both Full-Rate and Half-Rate modes is shown in the timing diagrams Figure 3-6 and Figure 3-7 below.

***Figure 3-6. 8-bit SERDES Only Mode Transmit Timing Diagram (Full Rate)***

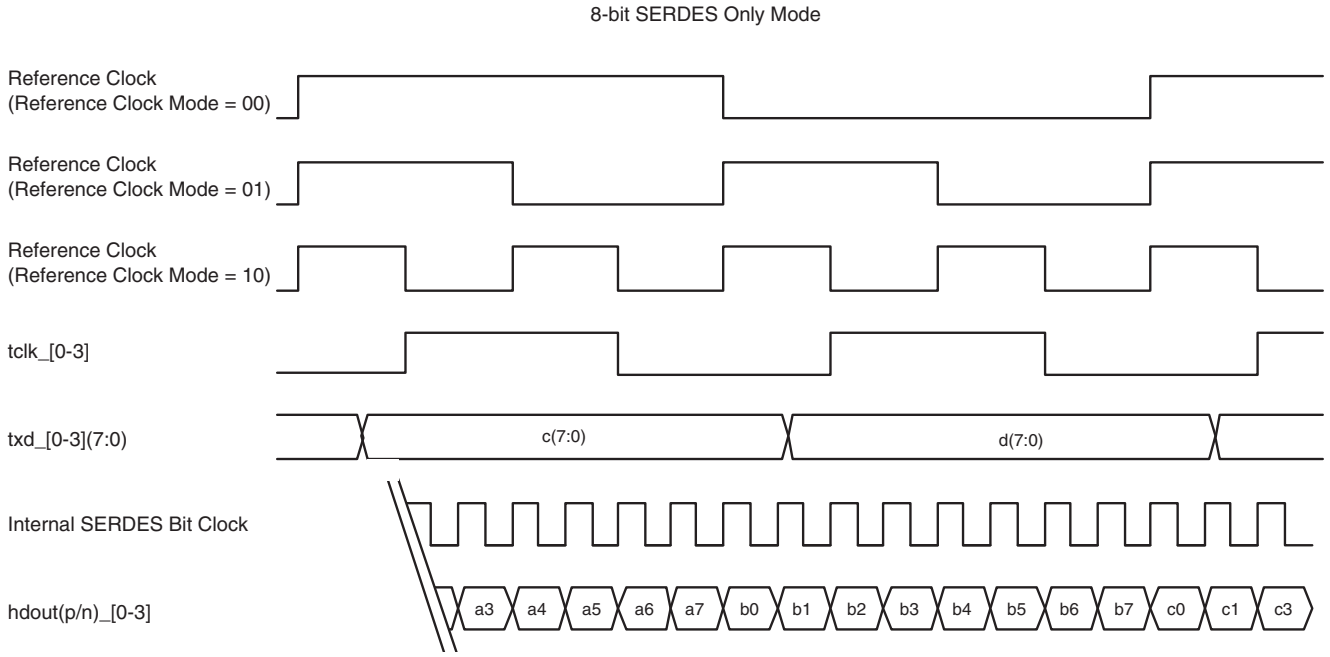
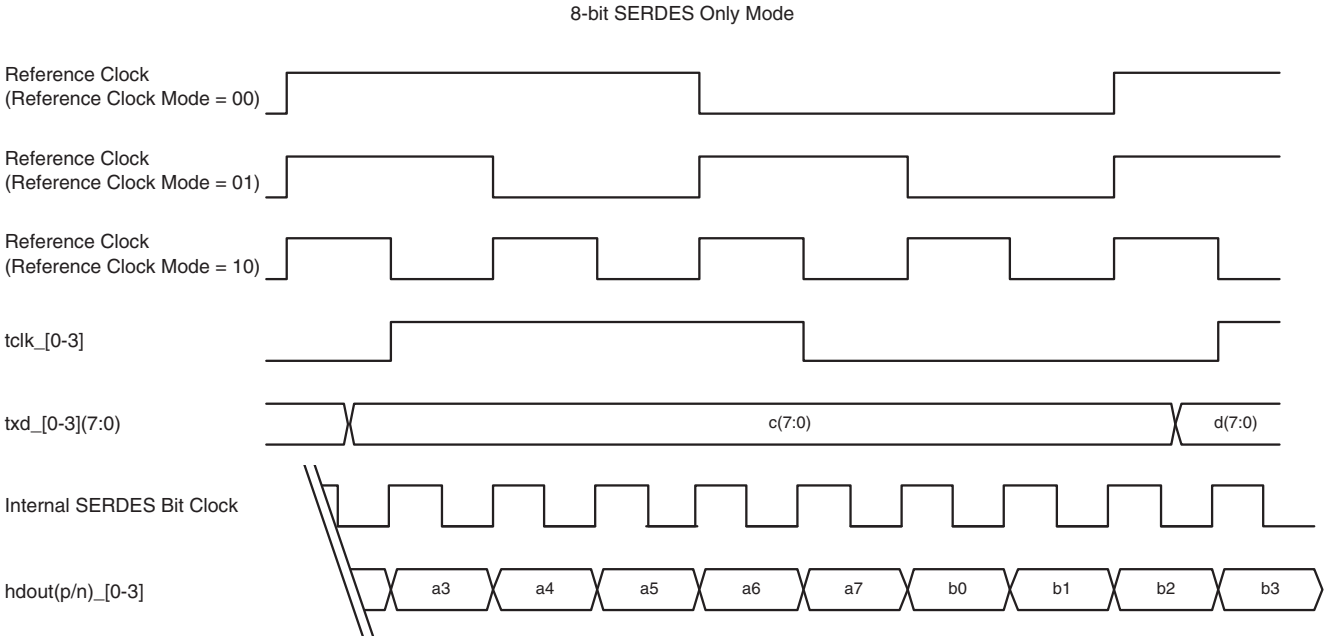


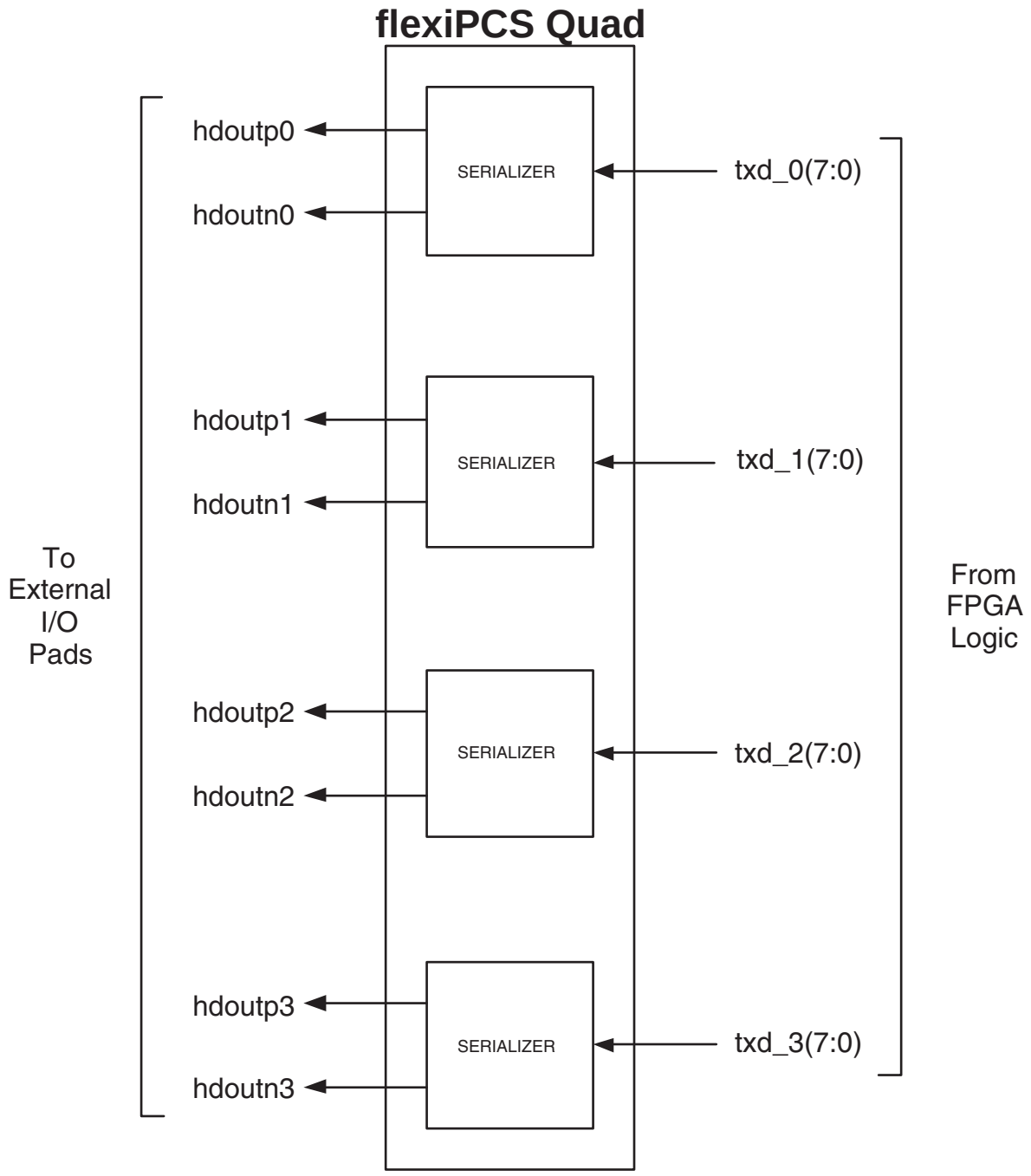

***Figure 3-7. 8-bit SERDES Only Mode Transmit Timing Diagram (Half Rate)***




**txd_[0-3](7:0)** - Per channel transmit data from the PCS/FPGA interface. Each quad supports up to 4 independent channels of 8-bit wide parallel data. Each txd_[0-3][7:0] is an eight bit data bus that is driven synchronously with respect to the corresponding tclk_[0-3].

The quad PCS in 8-bit SERDES Only mode transmit data path consists of the following sub-blocks per channel: Serializer. Figure 3-8 shows the four channels of transmit data paths in a PCS quad set to 8-bit SERDES Only mode.

***Figure 3-8. 8-bit SERDES Only Transmit Path (1 Quad)***




**Transmit Data Control Registers**

The order of the transmitted bits can be reversed (most significant bit first) on a per channel basis by setting the appropriate Channel Interface Register Base Address 0x01, bit 4 to '1'.

All transmitted bits can be inverted on a per channel basis by setting the appropriate Channel Interface Register Base Address 0x01, bit 5 to '1'.

## Receive Data

By default, serial data is received by the SERDES outputs with the least significant bit received first. Timing for receive data for both Full-Rate and Half-Rate modes is shown in the timing diagrams Figure 3-9 and Figure 3-10 below.

***Figure 3-9. 8-bit SERDES Only Mode Receive Timing Diagram (Full Rate)***

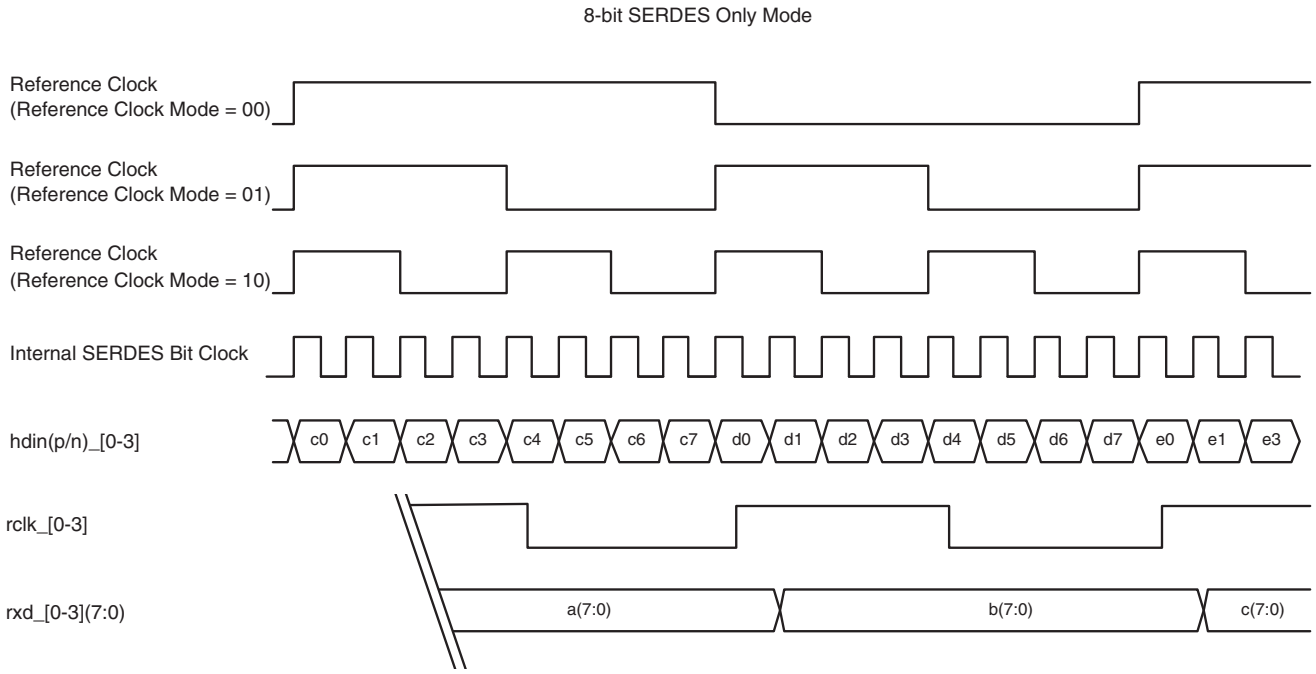
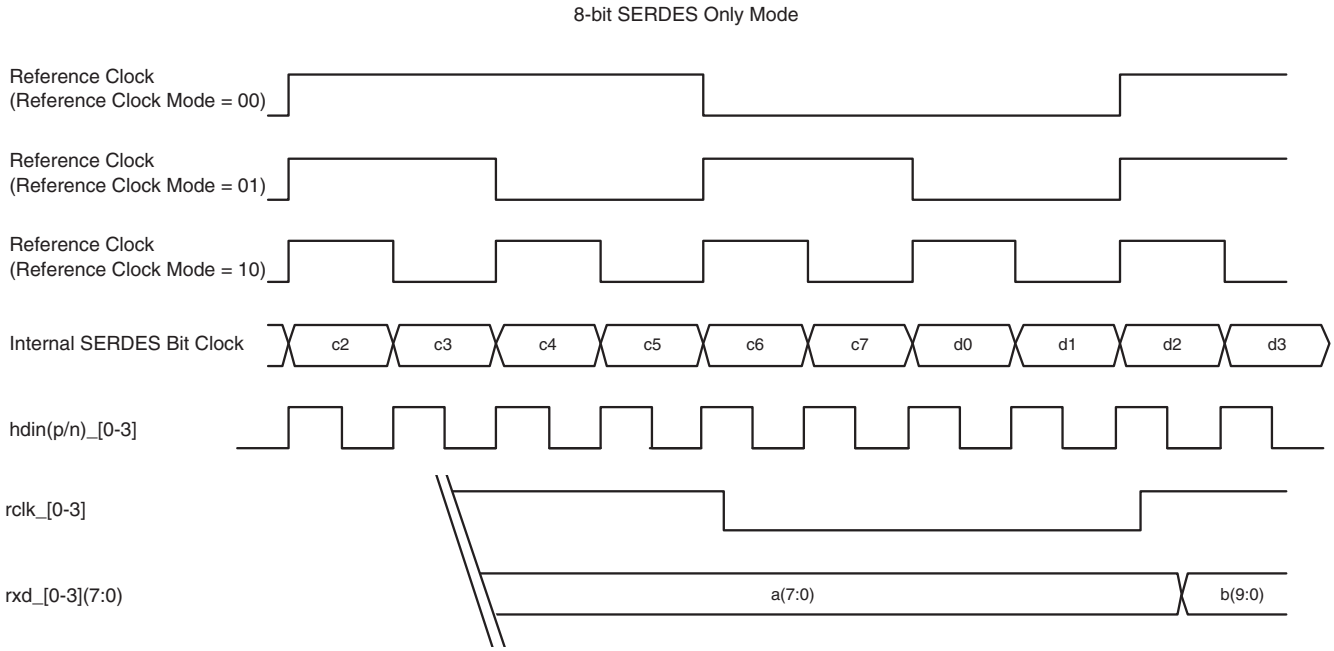
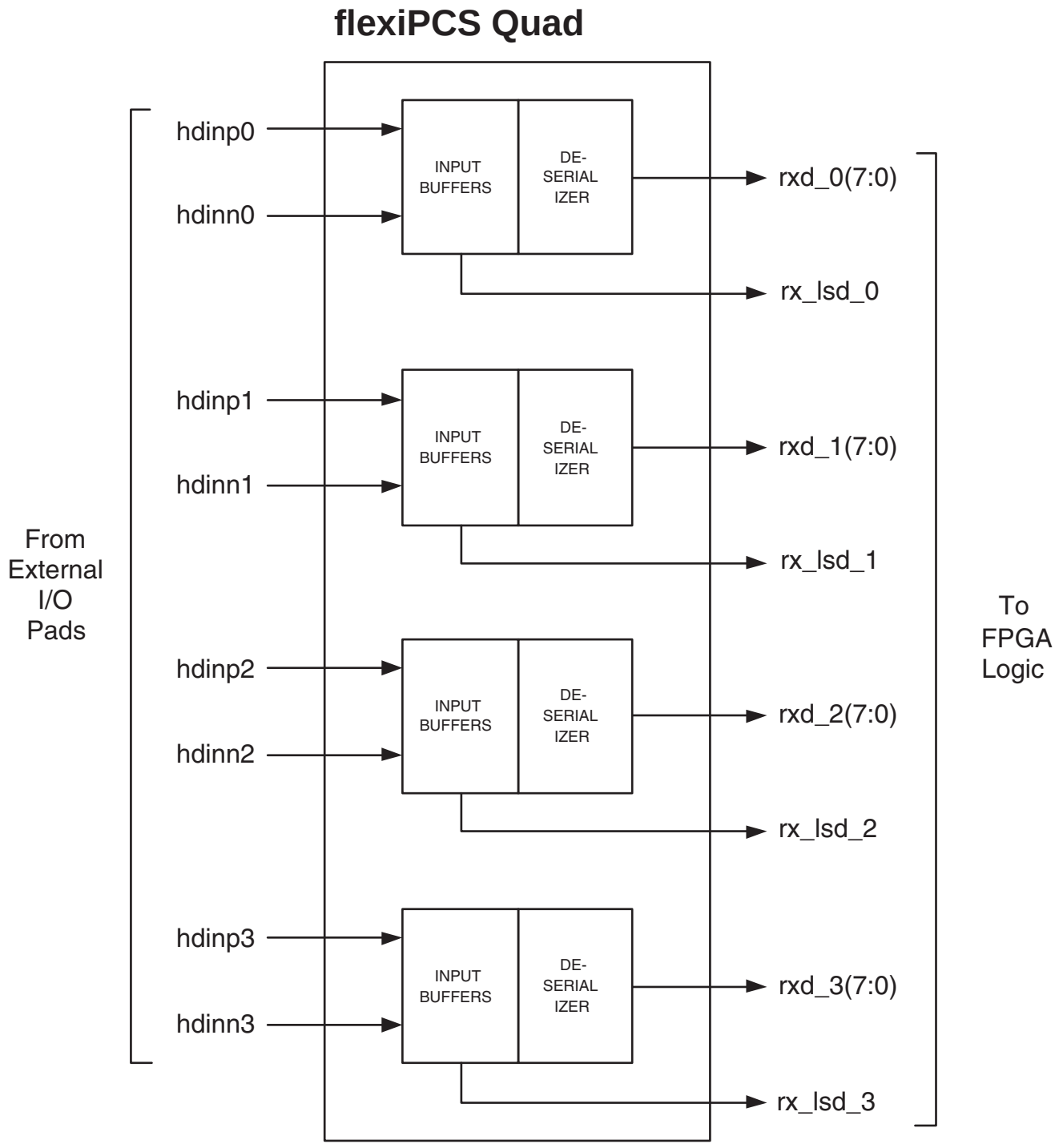



***Figure 3-10. 8-bit SERDES Only Mode Receive Timing Diagram (Half Rate)***



The quad PCS in 8-bit SERDES Only mode receive data path consists of the following sub-blocks per channel: Input Buffers and Deserializer. Figure 3-11 shows the four channels of receive data paths in a PCS quad in 8-bit SERDES Only mode.

***Figure 3-11. 8-bit SERDES Only Receive Data Path (1 Quad)***



**Input Buffers**
Serial data/clock signal is brought on chip to the embedded SERDES buffers. After on-chip buffering, each channel's input is fed directly to the PCS/FPGA interface for use in low speed applications where data rates are under 600 Mbps. The serial signals rx_lsd can be fed to a clock/data recovery circuit in the FPGA logic or off-chip. This allows dual use of a particular SERDES input pin for both high speed (> 600 Mbps using the rxd outputs) and low speed (<600 Mbps using the rx_lsd output) applications. The SERDES input buffers should be set to DC mode when receiving low speed inputs. The SERDES input buffers can be set to DC mode on a per channel basis by writing the appropriate Channel Interface Register Offset Address 0x15, bit 4 to a '1'.

**rx_lsd_[0-3]** - Low speed serial data from the SERDES input buffers. These pins are of interest in an application where the same SERDES input receive buffers are to be driven by either a high speed (> 600 Mbps) signal or a low speed signal. The rx_lsd inputs are passed directly from the SERDES input buffers and are not locked to the reference clock. FPGA logic based or off-chip clock/data recovery circuit would be needed for embedded clock/data input signals. The **rx_lsd** inputs do not toggle unless enabled. They can be enabled on a per channel basis by writing the appropriate Channel Interface Register Offset Address 0x15, bit 2 to a '1'. When driving a SERDES input buffer with a low speed signal, the SERDES input buffer should be set to DC mode, which can be done on a per channel basis by writing the appropriate Channel Interface Register Offset Address 0x15, bit 4 to a '1'.

When the SERDES input buffer data rate is running at greater than 600 Mbps, disable the rx_lsd pins and set the SERDES input buffer to AC mode.

**Deserializer**
Data is sent to the PCS/FPGA interface as 8-bit parallel data updated at 1/8 the internal bit clock rate when the quad is set to "8 bit data bus width" and 16-bit parallel data updated at 1/16 the internal bit clock rate when the quad is set to "16 bit data bus width".

**rxd_[0-3](7:0)** - Per channel parallel receive data to the PCS/FPGA interface. Each quad supports up to 4 independent channels of 8-bit wide parallel data. The rxd_[0-3] signal transitions synchronously with respect to rclk_[0-3].

**Receive Data Control Registers**
The order of the received bits can be reversed (most significant bit first) on a per channel basis by setting the appropriate Channel Interface Register Base Address 0x01, bit 6 to '1'.

All receive bits can be inverted on a per channel basis by setting the appropriate Channel Interface Register Base Address 0x01, bit 7 to '1'.

### Control & Status

The Control & Status pins for the 8-bit SERDES Only mode can be optionally selected on a per quad basis when generating the PCS quad interface files with the ispLEVER tools. Each of these control and status functions are also accessible through equivalent Quad Interface and Channel Interface Registers. The control and status signals allow direct real time access of these functions from FPGA logic.

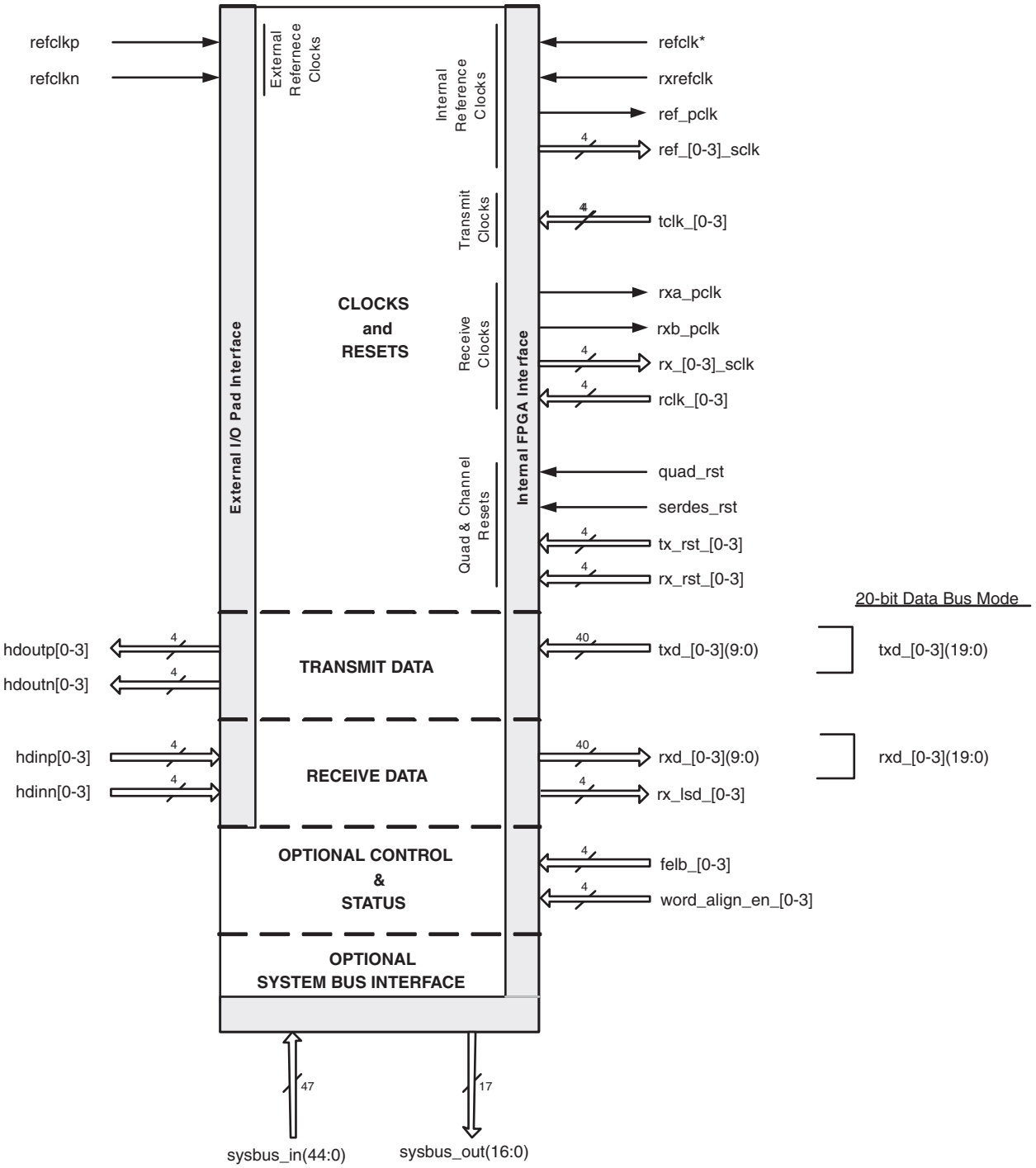
Each of the following functions are accessible on a per channel basis.

**felb_[0-3]** - Active high control signal which sets up the appropriate channel in far-end loopback mode. This mode is generally used for testing the flexiPCS logic, looping receive data back to the transmit direction without crossing the FPGA logic interface. For more information on the details of Far End Loopback mode, see the **flexiPCS Testing** Section of the flexiPCS Data Sheet. The Far End Loopback mode can also be initiated by writing Channel Interface Register Base Address 0x00, bit 5 to '1'.

## 10-bit SERDES Only Mode

IPexpress allows the designer to choose the mode for each PCS quad used in a given design. Figure 3-12 displays the resulting port interface to one PCS quad that has been set to 10-bit SERDES Only mode on all four channels. If the "Optional Direct Control & Status Register Access" box is checked in IPexpress, then word alignment is available and the word aligner can be controlled with the **word_align_en** pins at the PCS/FPGA interface.

***Figure 3-12. 10-bit SERDES Only Mode Pin Diagram***




* The refclk inputs for all active quads on a device must be connected to the same clock . The rxrefclk inputs for all active quads on a device do not have to be connected to the same clock.

## 10-bit SERDES Only Mode Pin Description

Table 3-4 lists all inputs and outputs to/from a PCS quad in 10-bit SERDES Only mode. A brief description for each port is given in the table. A more detailed description of the function of each port is given in the **10-bit SERDES Only Mode Detailed Description**.

***Table 3-4. 10-bit SERDES Only Clocks & Resets***

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Reference Clocks** | | | |
| refclkp | In from I/O pad | | Reference clock input, positive. Dedicated CML input. |
| refclkn | In from I/O pad | | Reference clock input, negative. Dedicated CML input |
| refclk | In from FPGA | | Optional reference clock input from FPGA logic. Can be used instead of I/O pin reference clock.<br><br>The refclk inputs for all active quads on a device must be connected to the same clock. |
| rxrefclk | In from FPGA | | Optional reference clock input from FPGA logic. Can be used instead of I/O pin reference clock. The rxrefclk inputs for all active quads do not have to be connected to the same clock. |
| ref_pclk | Out to FPGA | | Locked reference clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| ref_[0-3]_sclk | Out to FPGA | | Per channel locked reference clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| **Transmit Clocks** | | | |
| tclk_[0-3] | In from FPGA | | Per channel transmit clock inputs from FPGA. Used to clock the TX data phase compensation FIFO with clock synchronous to the reference clock. May also be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Receive Clocks** | | | |
| rxa_pclk | Out to FPGA | | Recovered receive clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| rxb_pclk | Out to FPGA | | Recovered receive clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| rx_[0-3]_sclk | Out to FPGA | | Per channel receive clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| rclk_[0-3] | In from FPGA | | Per channel receive clock inputs from FPGA. Used to clock the RX data phase compensation FIFO with a clock synchronous to the reference and/or receive reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Resets** | | | |
| quad_rst | In from FPGA | | Active high, asynchronous reset for all channels of SERDES and PCS logic. |
| serdes_rst | In from FPGA | | Active high, asynchronous reset for all channels of the SERDES. |
| tx_rst_[0-3] | In from FPGA | | Per channel active high, asynchronous reset of individual transmit channel of PCS logic. |
| rx_rst_[0-3] | In from FPGA | | Per channel active high, asynchronous reset of individual receive channel of PCS logic. |
| **Transmit Data** | | | |
| hdoutp[0-3] | Out to I/O pad | | High-speed CML serial output, positive. |

*Table 3-4. 10-bit SERDES Only Clocks & Resets (Continued)*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| hdoutn[0-3] | Out to I/O pad | | High-speed CML serial output, negative. |
| txd_[0-3](9:0)<br><br>txd_[0-3](19:0)* | In from FPGA | tclk_[0-3] | Per channel parallel transmit data bus from the FPGA. Bus is 10 bits wide if QIR 0x19, bit 4 = '0'.<br><br>*Bus is 20 bits wide if QIR 0x19, bit 4 = '1'. |
| **Receive Data** | | | |
| hdinp[0-3] | In from I/O pad | N/A | High-speed CML serial input, positive. |
| hdinn[0-3] | In from I/O pad | N/A | High-speed CML serial input, negative. |
| rxd_[0-3](9:0)<br><br>rxd_[0-3](19:0)* | Out to FPGA | rclk_[0-3] | Per channel parallel receive data bus to the FPGA. Bus is 10 bits wide if QIR 0x19, bit 5 = '0'.<br><br>*Bus is 20 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_lsd_[0-3] | Out to FPGA | ASYNC | Per channel serial low speed data to the FPGA. |
| **Optional Control & Status** | | | |
| felb_[0-3] | In from FPGA | ASYNC | Per channel active high far end loopback enables. |
| word_align_en_[0-3] | In from FPGA | ASYNC | Per channel active high word align enables. |
| **Optional System Bus Interface** | | | |
| sysbus_in(44:0) | In from FPGA | N/A | Control and data signals from the internal system bus. |
| sysbus_out(16:0) | Out to FPGA | N/A | Control and data signals to the internal system bus. |

# 10-bit SERDES Only Mode Detailed Description

The following section provides a detailed description of the operation of a PCS quad set to 10-bit SERDES Only mode. The functional description is organized according to the port listing shown in Figure 3-12. The System Bus base address for any control and status registers relevant to a given function are provided with the description of the operation of that function.

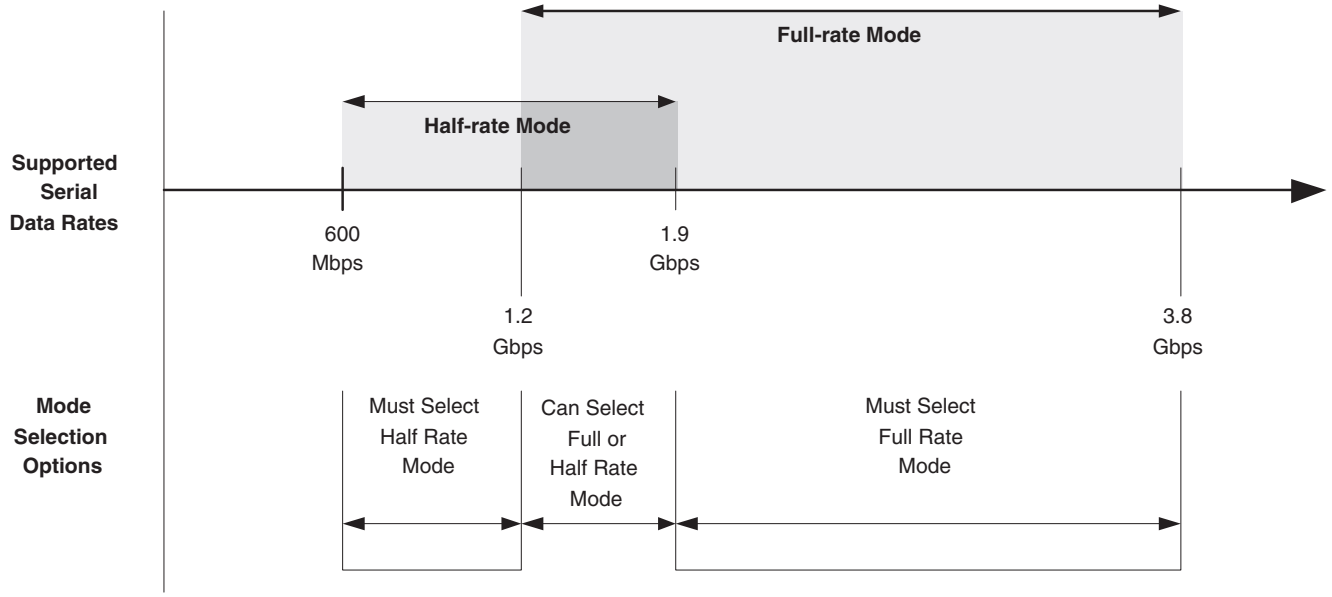
## Clocks & Resets

A detailed description of all SERDES clock, data, and reset ports and recommended reset sequencing is provided in the SERDES Functionality section of the flexiPCS Data Sheet.

### Full-Rate and Half-Rate Modes

The SERDES PLLs support data rates from 1.2 Gbps to 3.8 Gbps. A half-rate mode is available to permit operation of the SERDES at industry standard protocol data rates lower than 1.2 Gbps. The combination of half-rate and full-rate modes allow operation of a single SERDES channel at any data rate from 600 Mbps to 3.8 Gbps as shown in Figure 3-13.

*Figure 3-13. Full-Rate and Half-Rate Mode Selection*




Each of the four receive channels can be independently configured to receive data at either full-rate or half-rate. Similarly each of the four transmit channels can be independently configured to transmit data at either full-rate or half-rate. The truth table shown in Table 3-5 for transmit and receive channels describes the reference clock and data rate relationship during full and half-rate operation.

For more information on the selection of full-rate and half-rate modes, refer to the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

*Table 3-5. Full-Rate and Half-Rate Mode Operations*

| Rate Mode | Reference Clock Mode | | | |
|---|---|---|---|---|
| Channel Interface Register Offset Address = 0x13<br><br>Transmit - Bit 5<br>Receive - Bit 4 | Quad Interface Register Offset Address = 0x28,<br><br>Bits [2:3] | Internal Serial (bit) Clock Multiplier[1] | FPGA Interface Clock Multiplier<br><br>Quad Interface Register Offset Address 0x19<br><br>(See Note 1) | |
| | | | 10 Bit Data Bus Width<br><br>Transmit - Bit 4 = '0'<br>Receive - Bit 5 = '0' | 20 Bit Data Bus Width<br><br>Transmit - Bit 4 = '1'<br>Receive - Bit 5 = '1' |
| 0<br><br>Full rate<br>(1.2 Gbps - 3.8 Gbps) | 00 | 20X | 2X | 1X |
| | 01 | 10X | 1X | (1/2)X |
| | 10 | 5X | (1/2)X | (1/4)X |
| 1<br><br>Half rate<br>(600 Mbps - 1.9 Gbps) | 00 | 10X | 1X | (1/2)X |
| | 01 | 5X | (1/2)X | (1/4)X |
| | 10 | (5/2)X | (1/4)X | (1/8)X |

1. All clock multiplier values are with respect to supplied reference clock frequency.

*Note: There are also frequency dependent SERDES performance control bits that are not writable via the System-bus. These control bits are set in the auto-configuration file generated within IPexpress and passed into the bit-*

*stream through the normal FPGA design flow (map, par, bitgen). To change the bit rate for a SERDES, specify the proper value for the affected flexiPCS quad in IPexpress and regenerate the auto-configuration file. Failure to do this may result in non-optimal SERDES operation.*

For more information on the selection of full-rate and half-rate modes, refer to the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

**Quad & Channel Resets**

Resets are provided to reset an entire quad (either SERDES logic only or all flexiPCS logic) or each individual channel (all flexiPCS logic per channel). These resets are driven from the FPGA logic or through a Systembus register. A summary of the control signals provided for reset are listed below. More detail on recommended initialization and reset sequences for the SERDES is provided in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

The following inputs to the flexiPCS from the FPGA are enabled as long as Quad Interface Register Offset Address 0x42, bit 7 is set to '1' (which is the default on powerup). Writing a '0' to this bit will disable these reset inputs.

**quad_rst** - Active high, asynchronous signal from FPGA resets all SERDES and PCS logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x43, bit 7 to '1'. quad_rst also resets all flexiPCS control registers. If these registers had been previously written via the Systembus or set during configuration through an auto-configuration file, they will need to be rewritten following this reset.

**serdes_rst** - Active high, asynchronous signal from FPGA resets all SERDES (but not PCS) logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x41, bit 5 to '1'.Note that this bit is preset to '1' on powerup and must be written to '0' before operation of the SERDES can commence.

**tx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one transmit channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [4:7]. Bit 7 resets transmit channel 0, bit 6 resets transmit channel 1, bit 5 resets transmit channel 2, and bit 4 resets transmit channel 3.

**rx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one receive channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [0:3]. Bit 3 resets receive channel 0, bit 2 resets receive channel 1, bit 1 resets receive channel 2, and bit 0 resets receive channel 3.

## Transmit Data

By default, serial data is transmitted to the SERDES outputs with the least significant bit transmitted first. Transmit data from the FPGA logic is passed to the PCS on the **txd_[0-3]** pins. The **txd** bus is 10 bits wide when set to "10-bit data bus mode" mode and 20 bits wide when set to "20-bit data bus mode". High-speed serial outputs are transmitted on external pins **hdoutp[0-3]** and **hdoutn[0-3]**. Timing for transmit data for both Full-Rate and Half-Rate modes is shown in the timing diagrams Figure 3-14 and Figure 3-15 below.

***Figure 3-14. 10-bit SERDES Only Mode Transmit Timing Diagram (Full Rate)***

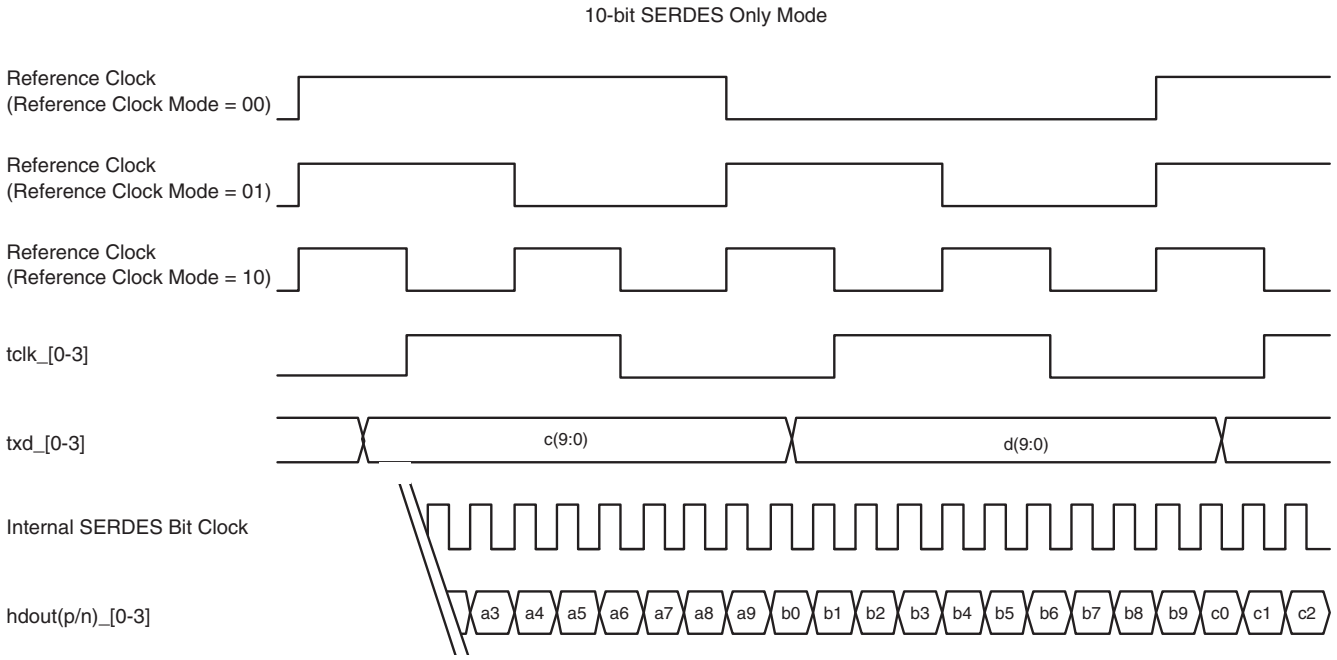
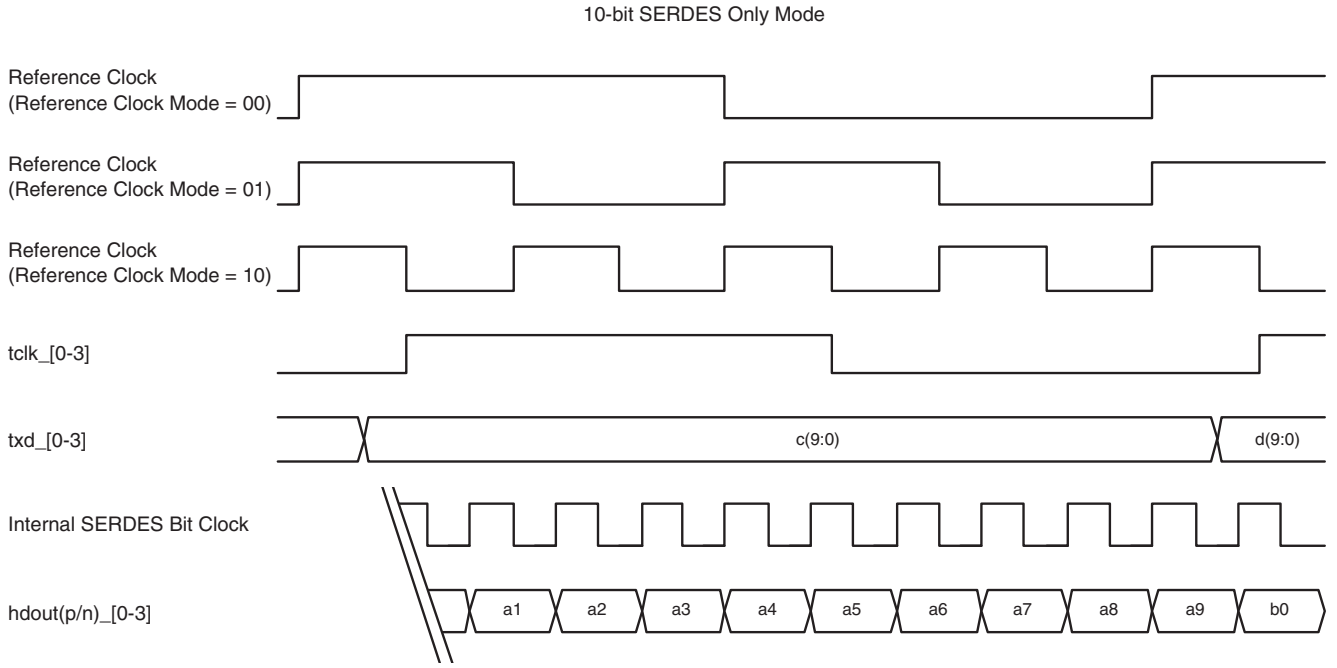


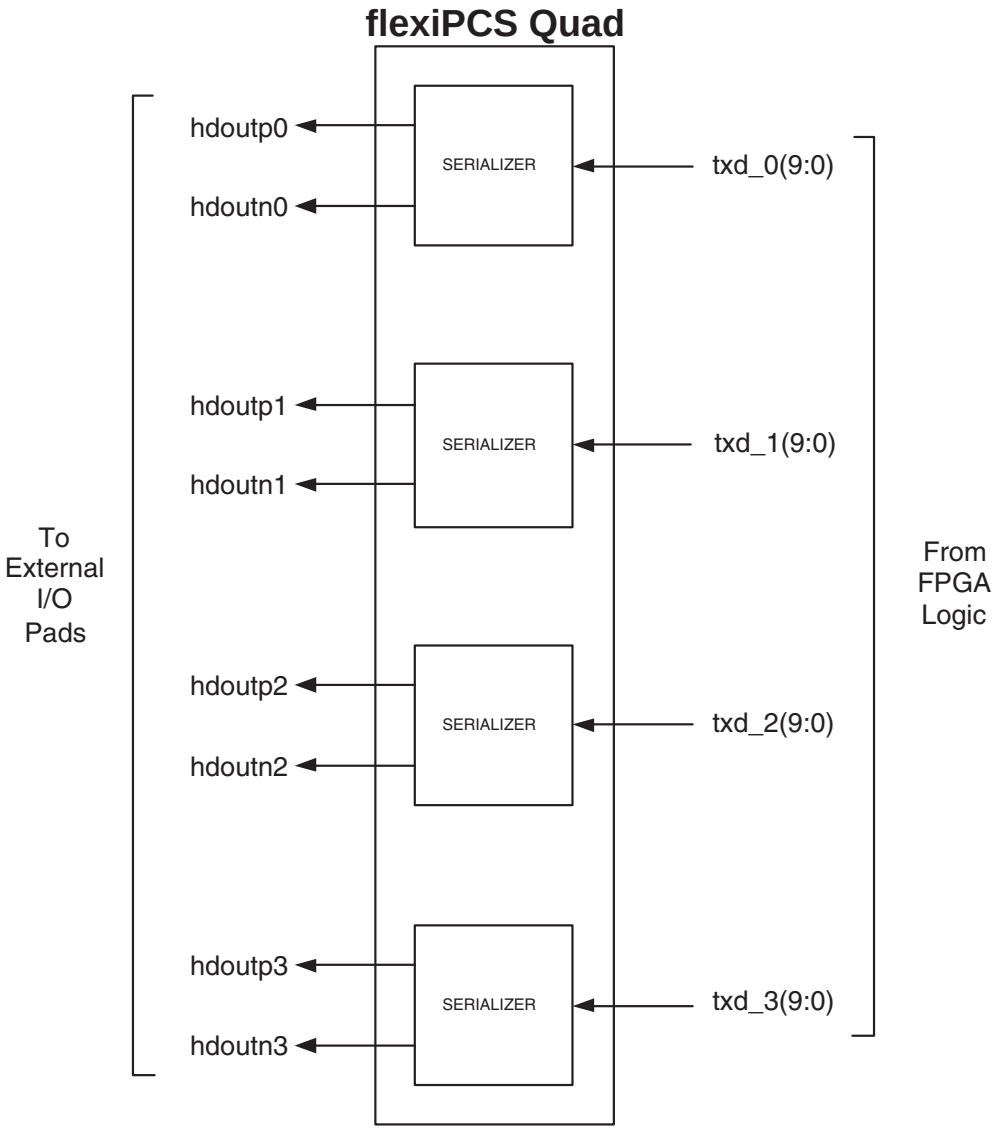

***Figure 3-15. 10-bit SERDES Only Mode Transmit Timing Diagram (Half Rate)***



**txd_[0-3](9:0)** - Per channel transmit data from the PCS/FPGA interface. Each quad supports up to 4 independent channels of 10-bit wide parallel data. Each txd_[0-3][9:0] is an eight bit data bus that is driven synchronously with respect to the corresponding tclk_[0-3].

The quad PCS in 10-bit SERDES Only mode transmit data path consists of the following sub-blocks per channel: Serializer. Figure 3-16 shows the four channels of transmit data paths in a PCS quad set to 10-bit SERDES Only mode.

***Figure 3-16. 10-bit SERDES Only Transmit Path (1 Quad)***




**Transmit Data Control Registers**

The order of the transmitted bits can be reversed (most significant bit first) on a per channel basis by setting the appropriate Channel Interface Register Base Address 0x01, bit 4 to '1'.

All transmitted bits can be inverted on a per channel basis by setting the appropriate Channel Interface Register Base Address 0x01, bit 5 to '1'.

## Receive Data

By default, serial data is received by the SERDES outputs with the least significant bit received first. Timing for receive data for both Full-Rate and Half-Rate modes is shown in the timing diagrams Figure 3-17 and Figure 3-18 below.

***Figure 3-17. 10-bit SERDES Only Mode Receive Timing Diagram (Full Rate)***

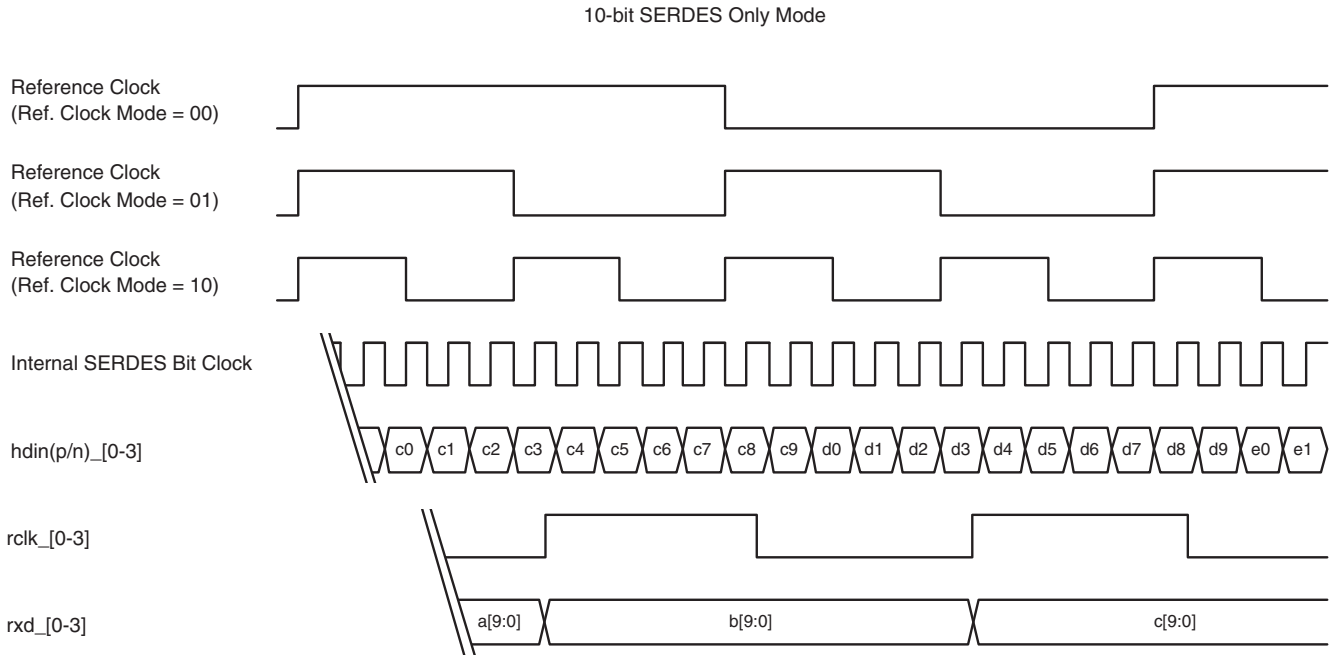
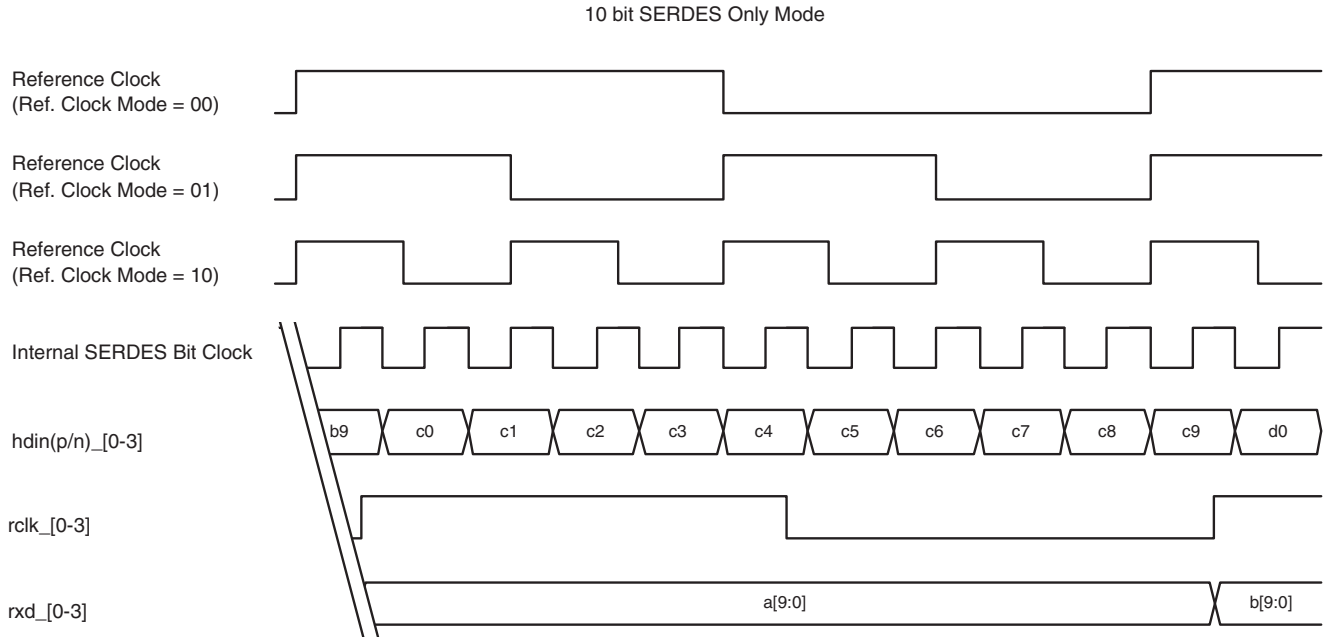
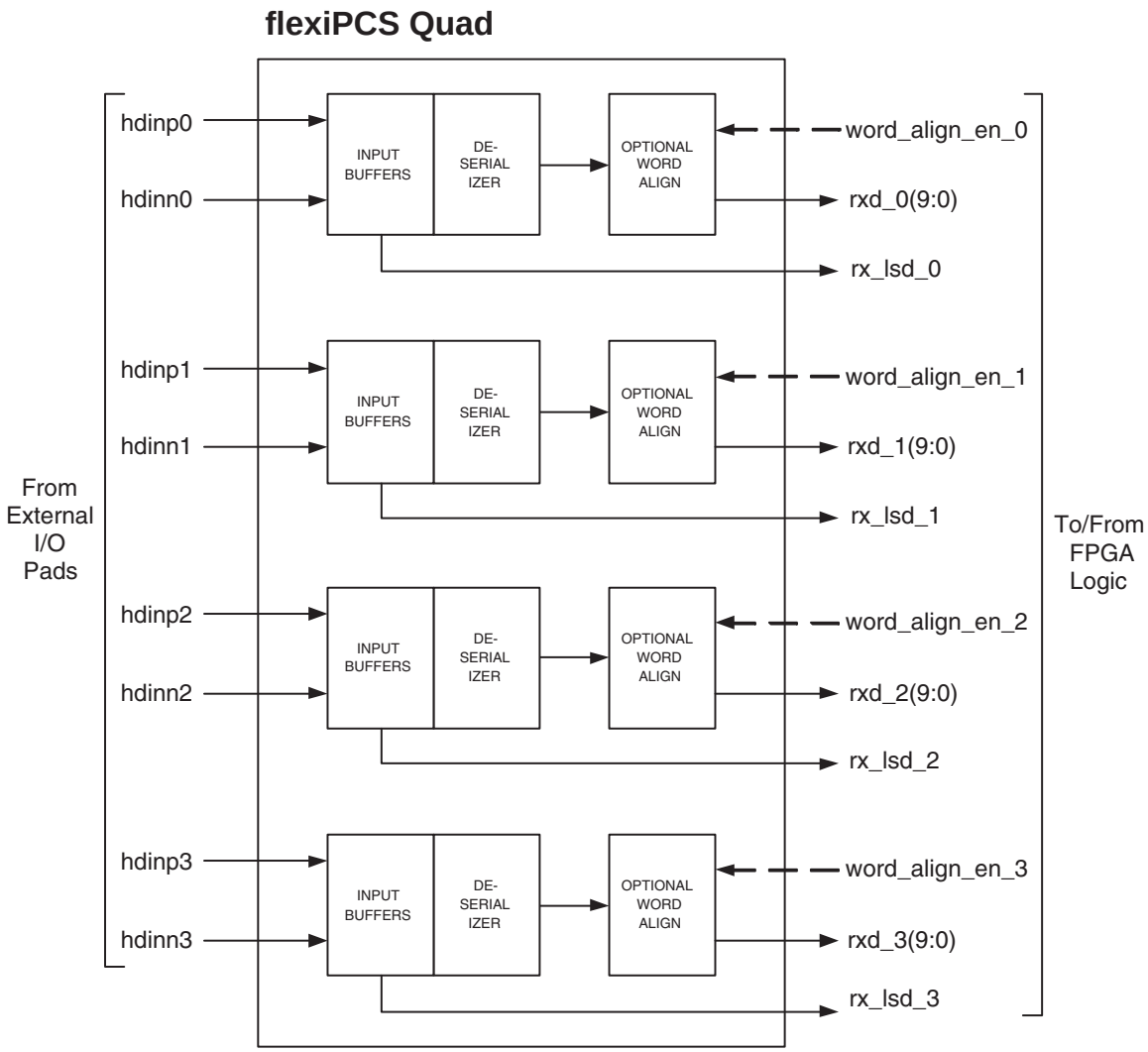



***Figure 3-18. 10-bit SERDES Only Mode Receive Timing Diagram (Half Rate)***



The quad PCS in 10-bit SERDES Only mode receive data path consists of the following sub-blocks per channel: Input Buffers and Deserializer. Figure 3-19 shows the four channels of receive data paths in a PCS quad in 10-bit SERDES Only mode.

***Figure 3-19. 10-bit SERDES Only Receive Data Path (1 Quad)***




**Input Buffers**

Serial data/clock signal is brought on chip to the embedded SERDES buffers. After on-chip buffering, each channel's input is fed directly to the PCS/FPGA interface for use in low speed applications where data rates are under 600 Mbps. The serial signals rx_lsd can be fed to a clock/data recovery circuit in the FPGA logic or off-chip. This allows dual use of a particular SERDES input pin for both high speed (> 600 Mbps using the rxd outputs) and low speed (<600 Mbps using the rx_lsd output) applications. The SERDES input buffers should be set to DC mode when receiving low speed inputs. The SERDES input buffers can be set to DC mode on a per channel basis by writing the appropriate Channel Interface Register Offset Address 0x15, bit 4 to a '1'.

**rx_lsd_[0-3]** - Low speed serial data from the SERDES input buffers. These pins are of interest in an application where the same SERDES input receive buffers are to be driven by either a high speed (> 600 Mbps) signal or a low speed signal. The rx_lsd inputs are passed directly from the SERDES input buffers and are not locked to the reference clock. FPGA logic based or off-chip clock/data recovery circuit would be needed for embedded clock/data input signals. The **rx_lsd** inputs do not toggle unless enabled. They can be enabled on a per channel basis by writ-

ing the appropriate Channel Interface Register Offset Address 0x15, bit 2 to a '1'. When driving a SERDES input buffer with a low speed signal, the SERDES input buffer should be set to DC mode, which can be done on a per channel basis by writing the appropriate Channel Interface Register Offset Address 0x15, bit 4 to a '1'.

When the SERDES input buffer data rate is running at greater than 600 Mbps, disable the rx_lsd pins and set the SERDES input buffer to AC mode.

**Deserializer**
Data is sent to the PCS/FPGA interface as 10-bit parallel data updated at 1/10 the internal bit clock rate when the quad is set to "10 bit data bus width" and 20-bit parallel data updated at 1/20 the internal bit clock rate when the quad is set to "20 bit data bus width".

**rxd_[0-3](9:0)** - Per channel parallel receive data to the PCS/FPGA interface. Each quad supports up to 4 independent channels of 8-bit wide parallel data. The rxd_[0-3] signal transitions synchronously with respect to rclk_[0-3].

**Receive Data Control Registers**
The order of the received bits can be reversed (most significant bit first) on a per channel basis by setting the appropriate Channel Interface Register Base Address 0x01, bit 6 to '1'.

All receive bits can be inverted on a per channel basis by setting the appropriate Channel Interface Register Base Address 0x01, bit 7 to '1'.

## Control & Status

The Control & Status pins for the 10-bit SERDES Only mode can be optionally selected on a per quad basis when generating the PCS quad interface files with the ispLEVER tools. Each of these control and status functions are also accessible through equivalent Quad Interface and Channel Interface Registers. The control and status signals allow direct real time access of these functions from FPGA logic.

Each of the following functions are accessible on a per channel basis.

**felb_[0-3]** - Active high control signal which sets up the appropriate channel in far-end loopback mode. This mode is generally used for testing the flexiPCS logic, looping receive data back to the transmit direction without crossing the FPGA logic interface. For more information on the details of Far End Loopback mode, see the **flexiPCS Testing** Section of the flexiPCS Data Sheet. The Far End Loopback mode can also be initiated by writing Channel Interface Register Base Address 0x00, bit 5 to '1'.

**word_align_en_[0-3]** - Signal which unlocks the word aligner on any minimum one clock cycle wide low pulse for the appropriate receive channel.

The word aligner will lock alignment (it will stop comparing the incoming data to the user-defined word alignment characters and will maintain current alignment) on the first successful compare to either of the two user-defined alignment words after word_align_en is pulsed low for at least one clock cycle. Any subsequent low pulse on the word_align_en pin at the FPGA interface will un-lock the word aligner. The word aligner will then re-lock on the next match to one of the user-defined word alignment characters. If desired, the word_align_en can be controlled by a Link State Machine implemented externally to the flexiPCS quad to allow a change in word alignment only under specific conditions.

The word alignment can also be controlled, as described for word_align_en_[0-3], by writing to the Channel Inteface Register Base Address 0x05, bit '0'. To lock, first write '0' and then '1' to this register bit. To unlock, write a '0' to this register bit.

Two 10-bit user defined alignment words can be set by writing to Quad Interface Registers (the alignment words are shared among all four channels of the quad). User-defined alignment word "A" is written to Quad Interface Register Base Address 0x17, bits 4 and 5 (alignment word "A" bits [9:8]), and Quad Interface Register Base Address 0x15, bits [0:7] (alignment character "A" bits [7:0]). User defined alignment word "B" is written to Quad Interface

Register Base Address 0x17, bits 6 and 7 (alignment word “B” bits [9:8]), and Quad Interface Register Base Address 0x16, bits [0:7] (alignment word “B” bits [7:0]).

If only one alignment word is needed, then the same value should be written for alignment word “A” and alignment word “B”.

The user can also define a 10-bit field alignment mask. When a bit in the alignment mask is set to ‘1’, the corresponding bit location in the user defined alignment word is compared to the incoming data bit. The alignment mask is written to Quad Interface Register Base Address 0x17, bits 2 and 3 (alignment mask bits [9:8]), and Quad Interface Register Base Address 0x14, bits [0:7] (alignment mask bits [7:0]).

# Generic 8b10b Mode
# LatticeSC/M flexiPCS Family Data Sheet

January 2008 Data Sheet DS1005

## Introduction

The Generic 8b10b mode of the flexiPCS (Physical Coding Sublayer) block is intended for applications requiring 8b10b encoding/decoding without the need for additional protocol specific data manipulation. The LatticeSC flexiPCS can support Generic 8b10b applications up to 3.8 Gbps per channel.

The LatticeSC flexiPCS in Generic 8b10b mode supports the following operations:

**Transmit Path (From LatticeSC device to line):**

- Optional Cyclic Redundancy Check (CRC) Generation
- 8b10b Encoding

**Receive Path (From line to LatticeSC device):**

- Word Alignment to a user defined word alignment character.
- 8b10b Decoding
- Optional Link State Machine function.
- Optional Multi-channel Alignment to a user defined alignment character.
- Optional CRC Checker

### LatticeSC flexiPCS Quad Module

Devices in the LatticeSC family have up to 8 quads of embedded flexiPCS logic. Each quad in turn supports 4 independent full-duplex data channels. A single channel can support a fully compliant Generic 8b10b data link and each quad can support up to four such channels. Note that mode selection is done on a per quad basis. Therefore, a selection of Generic 8b10b mode for a quad dedicates all four channels in that quad to Generic 8b10b mode.

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

Operation of the SERDES requires the user to provide a reference clock (or clocks) to each active quad to provide a synchronization reference for the SERDES PLLs. Reference clocks are shared among all four channels of a quad. If the transmit and receive frequencies are within the specified ppm tolerance for the LatticeSC SERDES (See DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet), only one reference clock for both transmit and receive is needed for both transmit and receive directions. If the receive frequency is not within the specified ppm tolerance for the LatticeSC SERDES (See the DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet) of the transmit frequency, then separate reference clocks for the transmit and receive directions must be provided.

Simultaneous support of different (non-synchronous) high-speed data rates is possible with the use of multiple quads. Multiple frequencies that are exact integer multiples can be supported on a per channel basis through use of the full-rate/half-rate mode options (see the **SERDES Functionality section** for more details).

## Quad and Channel Option Control

Although the mode selection and reference frequency covers an entire quad, many options covering clocking and data formatting are available on a per channel basis. These options are detailed in the following Generic 8b10b Mode description. Some of these options can be controlled directly through the FPGA interface pins made available on the Generic 8b10b Quad Module.

© 2008 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Other options are available only through dedicated registers that can be written and read via the embedded System Bus Interface (see the System Bus section for more details on the operation of the System Bus), or during device configuration. Depending on the specific option, a register may affect operation of multiple quads, a single quad or an individual channel in a quad. Registers dedicated to multiple quads are listed in the Inter-quad Interface Register Map in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet. Registers dedicated to one quad are listed in the Quad Interface Register Map. Registers dedicated to a single channel are listed in the Channel Interface Register Map.

## Register Map Addressing

Figure 4-1 provides a map of base register addresses for any of the flexiPCS quads on a LatticeSC device. Note that different devices may have different numbers of available quads up to a total of 8 quads (or 32 channels) maximum.

Inter-quad Interface, Quad Interface, and Channel Interface Registers are listed by Offset Address. Each Inter-quad Interface Register, Quad Interface Register, and Channel Interface Register has a unique 18-bit address determined by the specific quad or channel targeted. The addressing of Inter-quad Interface Registers, Quad Interface Registers, and Channel Interface Registers is shown Figure 4-1.

Base addresses are dependant on the physical location of a given quad or channel on a LatticeSC device. Each quad and channel has an associated set of control and status registers. These registers are referred throughout this data sheet by their offset addresses. The unique 18-bit address of a control or status register for a specific quad or channel is simply the offset address of that register added to the base address for that specific quad or channel as shown in Figure 4-1.

Channel Interface Registers can be written in one of three methods. One method is to write to one specific register corresponding to one specific channel. The other two methods involve writing to multiple channel registers with just one write operation. This is referred to as a Broadcast write. A Broadcast write is useful when multiple channel registers are to be written with the same value. The first method of Broadcast writing involves writing to all four channel registers in a quad at one time. A second method of Broadcast writing involves writing to all channel registers for all quads on the left or right side of the device at one time. Therefore, a specific Channel Interface Register may be accessed through any one of three channel addresses, depending on the number of channel registers being written to at any one time.

A broadcast write to multiple quads cannot be used to power on SERDES in any device-package combination that does not have all SERDES quads bonded because of excess power on the board and jitter is also affected.

Throughout this document, the byte-wide data at each offset address is listed with a hexadecimal value with bit 0 as the MSB and bit 7 as the LSB as per the PowerPC convention. For example if the value for Quad Interface Register Offset Address 0x02 is stated to be 0x15, the bit pattern defined is as shown in Table 4-1:

***Table 4-1. Control/Status Register Bit Order Example***

| **Quad Interface Register Offset Address 0x02 = 0x15** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Data Bit 0 | Data Bit1 | Data Bit 2 | Data Bit 3 | Data Bit 4 | Data Bit 5 | Data Bit 6 | Data Bit 7 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| **1** | | | | **5** | | | |

A full list of register Offset Addresses is provided in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet.

***Figure 4-1. flexiPCS Inter-quad, Quad, and Channel Interface Register Map Base Addressing***

## Auto-Configuration

Initial register setup for each flexiPCS mode can be performed without accessing the system bus by using IPexpress, the auto-configuration tool in ispLEVER. IPexpress generates 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 system bus is therefore not needed if all quads are to be set via auto-configuration files. However, the system bus must be included in a design if the user needs to change control registers or monitor status registers during operation. Note that a quad reset (Quad Interface Register 0x43, bit 7 or the **quad_rst** port at the FPGA interface) will reset all registers back to their default (not auto-configuration) values. If a quad reset is issued, the flexiPCS registers need to be re-written via the System bus.

## Generic 8b10b Register Settings

The auto-configuration file sets registers that perform the following operations. If the auto-configuration file is not used, the appropriate registers need to be programmed via the Systembus. For more options, refer to the flexiPCS register map in the Memory Map section of the LatticeSC/M Family flexiPCS Data Sheet.

A flexiPCS quad can be set to Generic 8b10b mode by writing Quad Interface Register Offset Address 0x18 bits[0:7] to 0x10.

This register value automatically sets up the following options in the flexiPCS for the given quad. Details on the functionality of each chosen option is provided in the Generic 8b10b Mode Detailed Description section.

**Transmit Path**

- 8b10b encoder is enabled

**Receive Path**

- Word aligner is enabled
- 8b10b decoder is enabled

Other register settings are required to operate a channel or channels in Generic 8b10b Mode. The following is a list of options that must be set for Generic 8b10b operation. More detail for each option is included in the **Generic 8b10b Mode Detailed Description** section.

- Powerup of appropriate quad and channel. Quad powerup is chosen by writing the appropriate Quad Interface Register Offset Address 0x28, bit 1 to ‘1’. Channel powerup is chosen by writing the appropriate Channel Interface Register Offset Address 0x13, bit 6 (receive direction) and/or bit 7 (transmit direction) to ‘1’. By default, all channels and quads are powered down.
- Optional Receive Link State Machine enable. By default, all channel Receive Link State Machines are disabled. If use of one of the protocol specific Link State Machines is desired, then the Receive Link State Machine for a given channel can be enabled by writing the appropriate Channel Interface Register Offset Address 0x00, bit 7 to ‘1’. If use of one of the protocol specific Link State Machines is not desired, then the Link State Machines for all channels must be disabled by writing Quad Interface Register Offset Address 0x19, bit 0 to a ‘1’.
- Setting SERDES reset low at end of flexiPCS setup. By default, the SERDES reset is set to ‘1’ (SERDES are reset). The SERDES reset can be set low by writing Quad Interface Register Offset Address 0x41, bit 5 to a ‘0’. For more information on proper flexiPCS powerup and reset sequencing, refer to the **flexiPCS Reset Sequences** and **flexiPCS Power Down Control Signals** descriptions in the LatticeSC/M Family flexiPCS Data Sheet **SERDES Functionality** section.
- Word alignment character(s), such as a comma
- Multi-channel settings including user-defined alignment characters and FIFO settings (if Multi-channel Alignment is desired)
- Cyclic Redundancy Code (CRC) generator/checker enable and FIFO settings (if CRC generator/checker is desired)

## Generic 8b10b Auto-Configuration Example

An example of an auto-configuration file for a quad set to Generic 8b10b Mode with Multi-channel Alignment and CRC generation and checking enabled and with channels 0 and 1 enabled is shown in Figure 4-2. Format for the auto-configuration file is:

register type (quad=quad register, ch1=channel 1 register), offset address in hex, register value in hex, # comment

***Figure 4-2. Generic 8b10b Auto-Configuration Example File***

```
quad 18 10    #  Set quad to Generic 8b10b Mode
quad 29 01    #  Set reference clock select
quad 28 40    #  Set bit clock multiplier to 20X (for full rate mode), quad powerup set to '1'
quad 02 15    #  Set ref_pclk, rxa_pclk and rxb_pclk to channel 1 clocks
quad 14 7F    #  Set word alignment mask to compare lowest 7 LSBs
quad 15 03    #  Set positive disparity comma value
quad 16 7C    #  Set negative disparity comma value
quad 19 90    #  Enable external word aligner unlock control for all channels, select 2 channel (01) alignment
              #  Set FPGA interface data bus width to 8-bit
quad 04 03    #  Set alignment enable for channels 0 and 1
quad 01 00    # Set all multi-channel alignment clocks to be sourced from
              # channel 0 recovered receive clock
quad 07 FF    #  Set multi-channel align character mask to compare bits [7:0]
quad 08 7C    #  Set multi-channel align character “A”, bits [7:0] to 7C (K28.3)
quad 09 7C    #  Set multi-channel align character “B”, bits [7:0] to 7C (K28.3)
quad 0A 15    #  Set multi-channel align character mask to compare bit 8
              #  Set multi-channel align characters “A” and “B” bit 8 (K character) to '1'
quad 05 00    #  Set multi-channel alignment FIFO latency to minimum
quad 06 05    # Set multi-channel alignment FIFO high water mark to 5
quad 1D 30    #  Set CRC generator options (00 disables CRC generator/checker)
quad 1E 00    #  Set CRC checker options, define length of SOP and EOP as 2 bytes
quad 1F FF    #  Set CRC checker Start of Packet character mask to compare bits [0:7]
quad 20 BC    #  Set CRC checker Start of Packet 1st character, bits [7:0] (K28.5)
quad 21 B5    #  Set CRC checker Start of Packet 2nd character, bits [7:0] (D21.5)
quad 22 05    #  Set CRC checker Start of Packet character mask to compare bit 8
              #  Set CRC checker Start of Packet 1st character, bit 8 to '1'
              #  Set CRC checker Start of Packet 2nd character, bit 8 to '0'
quad 23 FF    #  Set CRC checker End of Packet character mask to compare bits [0:7]
quad 24 BC    #  Set CRC checker End of Packet 1st character, bits [7:0] (K28.5)
quad 25 95    #  Set CRC checker End of Packet 2nd character, bits [7:0] (D21.4)
quad 26 05    #  Set CRC checker End of Packet character mask to compare bit 8
              #  Set CRC checker End of Packet 1st character, bit 8 to '1'
              #  Set CRC checker End of Packet 2nd character, bit 8 to '0'
ch0 13 03     #  Powerup channel 0 transmit and receive directions, set rate mode to “full”
ch0 14 90     # 16% pre-emphasis on SERDES output buffer
ch0 15 10     # +6 dB equalization on SERDES input buffer
ch1 13 03     #  Powerup channel 1 transmit and receive directions, set rate mode to “full”
ch1 14 90     # 16% pre-emphasis on SERDES output buffer
ch1 15 10     # +6 dB equalization on SERDES input buffer
```

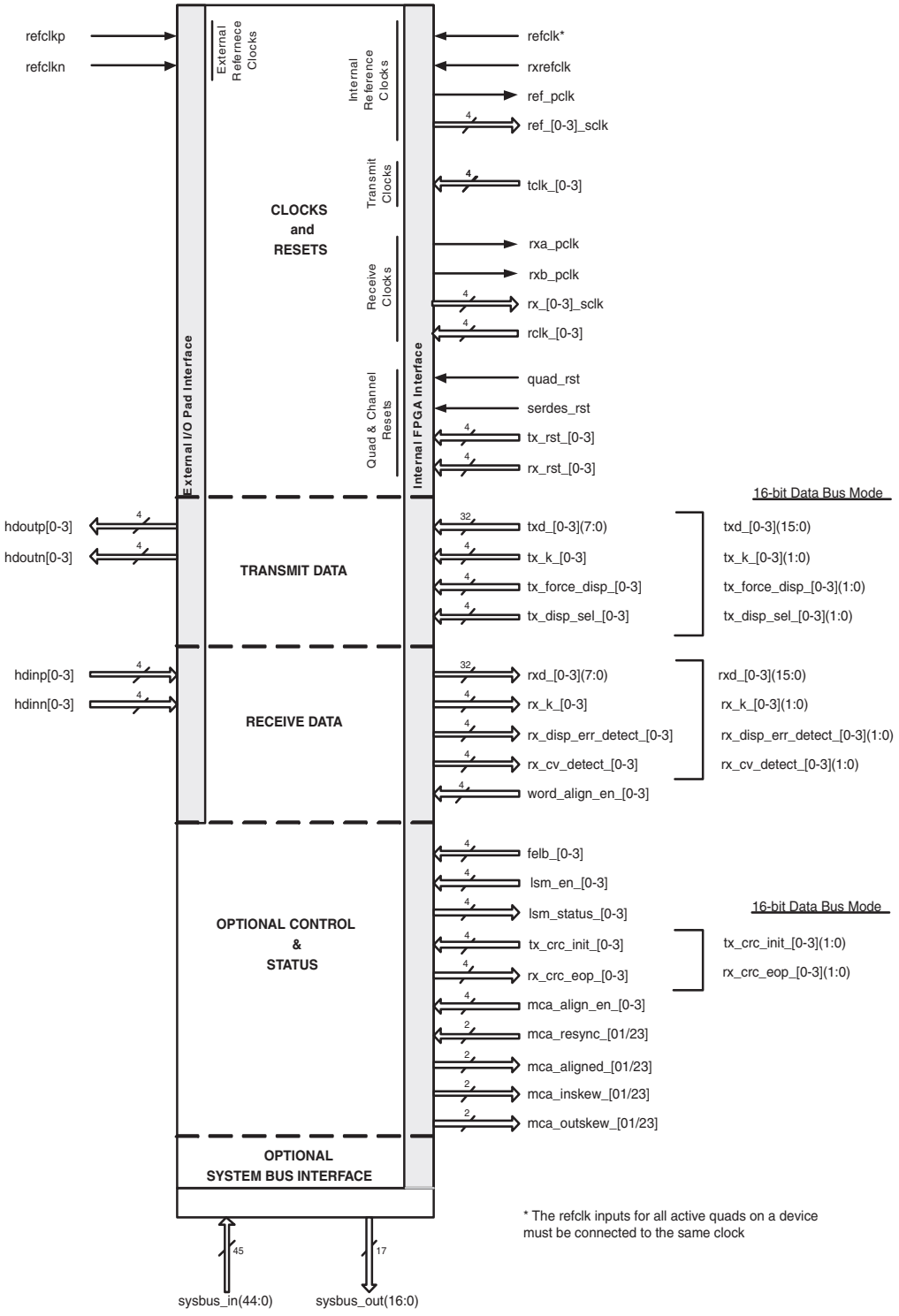
```
quad 41 00     # de-assert serdes_rst
quad 40 ff     # assert datapath reset for all channels
quad 41 03     # assert MCA reset
quad 41 00     # de-assert MCA reset
quad 40 00     # de-assert datapath reset for all channels
```

*Note: The last five lines must appear last in the autoconfig file. These lines apply the correct reset sequence to the PCS block upon bitstream configuration*

## Generic 8b10b Mode

IPexpress allows the designer to choose the mode for each flexiPCS quad used in a given design. Figure 4-3 displays the resulting port interface to one flexiPCS quad that has been set to Generic 8b10b mode on all four channels.

***Figure 4-3. Generic 8b10b Mode Pin Diagram***



## Generic 8b10b Mode Pin Description

Table 4-2 lists all inputs and outputs to/from a flexiPCS quad in Generic 8b10b mode. A brief description for each port is given in the table. A more detailed description of the function of each port is given in the **Generic 8b10b Mode Detailed Description**.

***Table 4-2. Generic 8b10b Mode Pin Description***

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Reference Clocks** | | | |
| refclkp | In from I/O pad | N/A | Reference clock input, positive. Dedicated CML input. |
| refclkn | In from I/O pad | N/A | Reference clock input, negative. Dedicated CML input |
| refclk | In from FPGA | N/A | Optional reference clock input from FPGA logic. Can be used instead of I/O pin reference clock.<br><br>The refclk inputs for all active quads on a device must be connected to the same clock |
| rxrefclk | In from FPGA | N/A | Optional receive reference clock input from FPGA logic. Can be used instead of I/O pin reference clock. The rxrefclk inputs for all active quads do not have to be connected to the same clock. |
| ref_pclk | Out to FPGA | N/A | Locked reference clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| ref_[0-3]_sclk | Out to FPGA | N/A | Per channel locked reference clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| **Transmit Clocks** | | | |
| tclk_[0-3] | In from FPGA | N/A | Per channel transmit clock inputs from FPGA. Used to clock the TX data phase compensation FIFO with clock synchronous to the reference clock. May also be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Receive Clocks** | | | |
| rxa_pclk | Out to FPGA | N/A | Recovered receive clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| rxb_pclk | Out to FPGA | N/A | Recovered receive clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| rx_[0-3]_sclk | Out to FPGA | N/A | Per channel recovered receive clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| rclk_[0-3] | In from FPGA | N/A | Per channel receive clock inputs from FPGA. May be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference and/or receive reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |

*Table 4-2. Generic 8b10b Mode Pin Description (Continued)*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Resets** | | | |
| quad_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of SERDES and PCS logic. |
| serdes_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of the SERDES. |
| tx_rst_[0-3] | In from FPGA | | Per channel active high, asynchronous reset of individual transmit channel of PCS logic. |
| rx_rst_[0-3] | In from FPGA | | Per channel active high, asynchronous reset of individual receive channel of PCS logic. |
| **Transmit Data** | | | |
| hdoutp[0-3] | Out to I/O pad | | High-speed CML serial output, positive. |
| hdoutn[0-3] | Out to I/O pad | | High-speed CML serial output, negative. |
| txd_[0-3](7:0)<br><br>txd_[0-3](15:0)* | In from FPGA | tclk_[0-3] | Per channel parallel transmit data bus from the FPGA. Bus is 8-bits wide if QIR 0x19, bit 4 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 4 = '1'. |
| tx_k_[0-3]<br><br>tx_k_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel transmit control word indicator.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_force_disp_[0-3]<br><br>tx_force_disp_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel active high, force disparity enable.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_disp_sel_[0-3]<br><br>tx_force_disp_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel disparity select. High forces positive disparity, low forces negative disparity.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| **Receive Data** | | | |
| hdinp[0-3] | In from I/O pad | N/A | High-speed CML serial input, positive. |
| hdinn[0-3] | In from I/O pad | N/A | High-speed CML serial input, negative. |
| rxd_[0-3](7:0)<br><br>rxd_[0-3](15:0)* | Out to FPGA | rclk_[0-3] | Per channel parallel receive data bus to the FPGA. Bus is 8-bits wide if QIR 0x19, bit 5 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 5 = '1'. |
| rx_k_[0-3]<br><br>rx_k_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel receive control word indicator.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_disp_err_detect_[0-3]<br><br>rx_disp_err_detect_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high disparity error detection. This signal will go high before the first data cycle to indicate the start of packet (SOP) instead of indicating a code violation if CIR 0x00 bit 4 is set to '1'.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_cv_detect_[0-3]<br><br>rx_cv_detect_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high code violation detection.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| word_align_en_[0-3] | In from FPGA | ASYNC | Per channel word align enables. |
| **Optional Control & Status** | | | |
| felb_[0-3] | In from FPGA | ASYNC | Per channel active high far-end loopback enables. |
| lsm_en_[0-3] | In from FPGA | ASYNC | Receive link state machine enable for all four channels |

*Table 4-2. Generic 8b10b Mode Pin Description (Continued)*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| lsm_status_[0-3] | Out to FPGA | ASYNC | Per channel signal from receive link state machine indicating successful link synchronization |
| tx_crc_init_[0-3]<br><br>tx_crc_init_[0-3](1:0)* | In from FPGA | ASYNC | Per channel CRC initialization.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_crc_eop_[0-3]<br><br>rx_crc_eop_[0-3](1:0)* | Out to FPGA | ASYNC | Per channel active high indicates that an end of packet and/or CRC error was detected. This signal is only valid when CRC is enabled by setting QIR 0x1D bits 2 and 3 = '1'. To prevent this signal from going high on CRC errors, set CIR 0x00 bit 3 = '1'.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| mca_align_en_[0-3] | In from FPGA | ASYNC | Per channel active high multi-channel aligner enables. |
| mca_resync_[01/23] | In from FPGA | ASYNC | Active high asynch reset for multi-channel aligner. |
| mca_aligned_[01/23] | Out to FPGA | ASYNC | Active high indicating successful alignment of channels. |
| mca_inskew_[01/23] | Out to FPGA | ASYNC | Active high indicating alignment skew tolerance is met. |
| mca_outskew_[01/23] | Out to FPGA | ASYNC | Active high indicating alignment skew tolerance is not met. |
| **Optional System Bus Interface** | | | |
| sysbus_in(44:0) | In from FPGA | N/A | Control and data signals from the internal system bus. |
| sysbus_out(16:0) | Out to FPGA | N/A | Control and data signals to the internal system bus. |

# Generic 8b10b Mode Detailed Description

The following section provides a detailed description of the operation of a flexiPCS quad set to Generic 8b10b mode. The functional description is organized according to the port listing shown in Figure 4-3. The System Bus Offset Address for any control and status registers relevant to a given function are provided with the description of the operation of that function.

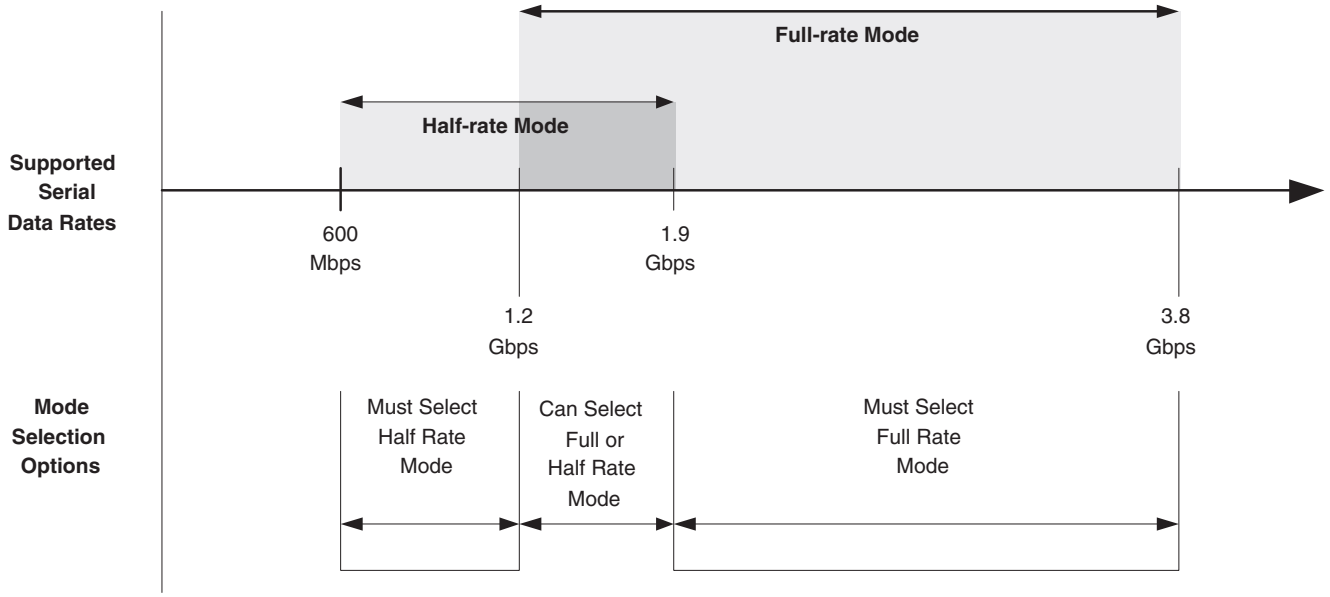
## Clocks & Resets

A detailed description of all SERDES clock, data, and reset ports and recommended reset sequencing is provided in the SERDES Functionality section of the flexiPCS Data Sheet.

### Full-Rate and Half-Rate Modes

The SERDES PLLs support data rates from 1.2 Gbps to 3.8 Gbps. A half-rate mode is available to permit operation of the SERDES at industry standard protocol data rates lower than 1.2 Gbps. The combination of half-rate and full-rate modes allow operation of a single SERDES channel at any data rate from 600 Mbps to 3.8 Gbps as shown in Figure 4-4.

*Figure 4-4. Full-Rate and Half-Rate Mode Selection*




Each of the four receive channels can be independently configured to receive data at either full-rate or half-rate. Similarly each of the four transmit channels can be independently configured to transmit data at either full-rate or half-rate. The truth table shown in Table 4-3 for transmit and receive channels describes the reference clock and data rate relationship during full and half-rate operation.

*Table 4-3. Full-Rate and Half-Rate Mode Operations*

| Rate Mode | Reference Clock Mode | | | |
|---|---|---|---|---|
| Channel Interface Register Offset Address = 0x13<br><br>Transmit - Bit 5<br>Receive - Bit 4 | Quad Interface Register Offset Address = 0x28,<br><br>Bits [2:3] | Internal Serial (bit) Clock Multiplier<br><br>(See Note 1) | FPGA Interface Clock Multiplier<br><br>Quad Interface Register Offset Address 0x19<br><br>(See Note 1) | |
| | | | 10 Bit Data Bus Mode<br><br>Transmit - Bit 4 = '0'<br>Receive - Bit 5 = '0' | 20 Bit Data Bus Mode<br><br>Transmit - Bit 4 = '1'<br>Receive - Bit 5 = '1' |
| 0<br><br>Full rate<br>(1.2 Gbps - 3.8 Gbps) | 00 | 20X | 2X | 1X |
| | 01 | 10X | 1X | (1/2)X |
| | 10 | 5X | (1/2)X | (1/4)X |
| 1<br><br>Half rate<br>(600 Mbps - 1.9 Gbps) | 00 | 10X | 1X | (1/2)X |
| | 01 | 5X | (1/2)X | (1/4)X |
| | 10 | (5/2)X | (1/4)X | (1/8)X |

1. All clock multiplier values are with respect to supplied reference clock frequency.

*Note: There are also frequency dependent SERDES performance control bits that are not writable via the System-bus. These control bits are set in the auto-configuration file generated within IPexpress and passed into the bit-stream through the normal FPGA design flow (map, par, bitgen). To change the bit rate for a SERDES specify the*

*proper values for the affected flexiPCS quad in IPexpress and regenerate the auto-configuration file. Failure to do this may result in non-optimal SERDES operation.*

For more information on the selection of full-rate and half-rate modes, refer to the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

**Quad & Channel Resets**
Resets are provided to reset an entire quad (either SERDES logic only or all flexiPCS logic) or each individual channel (all flexiPCS logic per channel). These resets are driven from the FPGA logic or through a Systembus register. A summary of the control signals provided for reset are listed below. More detail on recommended initialization and reset sequences for the SERDES is provided in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

The following inputs to the flexiPCS from the FPGA are enabled as long as Quad Interface Register Offset Address 0x42, bit 7 is set to '1' (which is the default on powerup). Writing a '0' to this bit will disable these reset inputs.

**quad_rst** - Active high, asynchronous signal from FPGA resets all SERDES and PCS logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x43, bit 7 to '1'. quad_rst also resets all flexiPCS control registers. If these registers had been previously written via the Systembus or set during configuration through an auto-configuration file, they will need to be rewritten following this reset.

**serdes_rst** - Active high, asynchronous signal from FPGA resets all SERDES (but not PCS) logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x41, bit 5 to '1'.Note that this bit is preset to '1' on powerup and must be written to '0' before operation of the SERDES can commence. quad_rst also resets all flexiPCS control registers. If these registers had been previously written via the Systembus or set during configuration through an auto-configuration file, they will need to be rewritten following this reset.

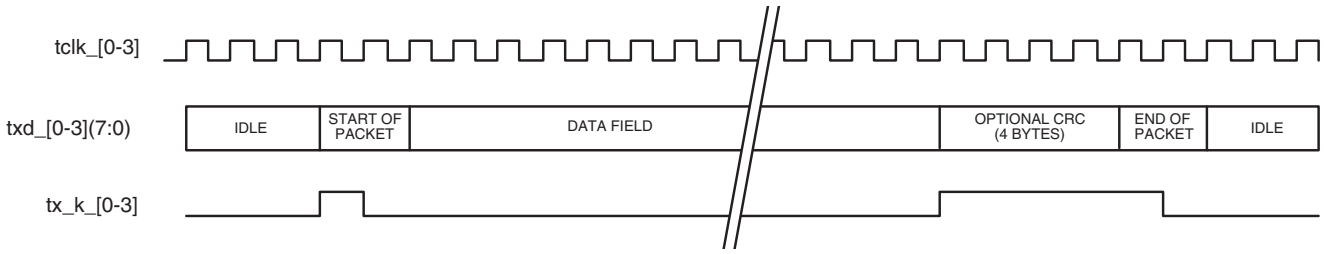
**tx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one transmit channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [4:7]. Bit 7 resets transmit channel 0, bit 6 resets transmit channel 1, bit 5 resets transmit channel 2, and bit 4 resets transmit channel 3.

**rx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one receive channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [0:3]. Bit 3 resets receive channel 0, bit 2 resets receive channel 1, bit 1 resets receive channel 2, and bit 0 resets receive channel 3.

## Transmit Data

When configured into Generic 8b10b mode, a flexiPCS quad provides four transmit data paths from an 8-bit parallel interface at the FPGA logic interface to a high-speed serial line interface at the device outputs. Figure 4-5 shows the typical operation of this interface at the flexiPCS interface. This figure shows an example where the SOP and EOP phases are two characters long with the first character being a control character. The length and value of SOP and EOP is programmable via quad interface registers. The generic 8b10b autoconfig example provides details on setting SOP and EOP length and value.

*Figure 4-5. Generic 8b10b Transmit FPGA Interface Signals*




**txd_[0-3](7:0)** - Per channel transmit data from the FPGA logic interface. Each quad supports up to 4 independent channels of 8-bit wide parallel data by default. Each txd_[0-3][7:0] is an eight bit data bus that is driven synchronously with respect to the corresponding tclk_[0-3].

In 16-Bit Data Bus Mode, this bus is 16-bits wide (**txd_[0-3](15:0)**).

**tx_k_[0-3]** - Per channel, active high control character indicator.

In 16 Bit Data Bus Mode, tx_k is 2 bits wide (**tx_k_[0-3](1:0)**) with tx_k_[0-3](1) associated with txd_[0-3](15:8) and tx_k_[0-3](0) associated with txd_[0-3](7:0).

**tx_force_disp_[0-3]** - Per channel, active high signal which instructs the flexiPCS to accept disparity value from the tx_disp_sel_[0-3] FPGA interface input.

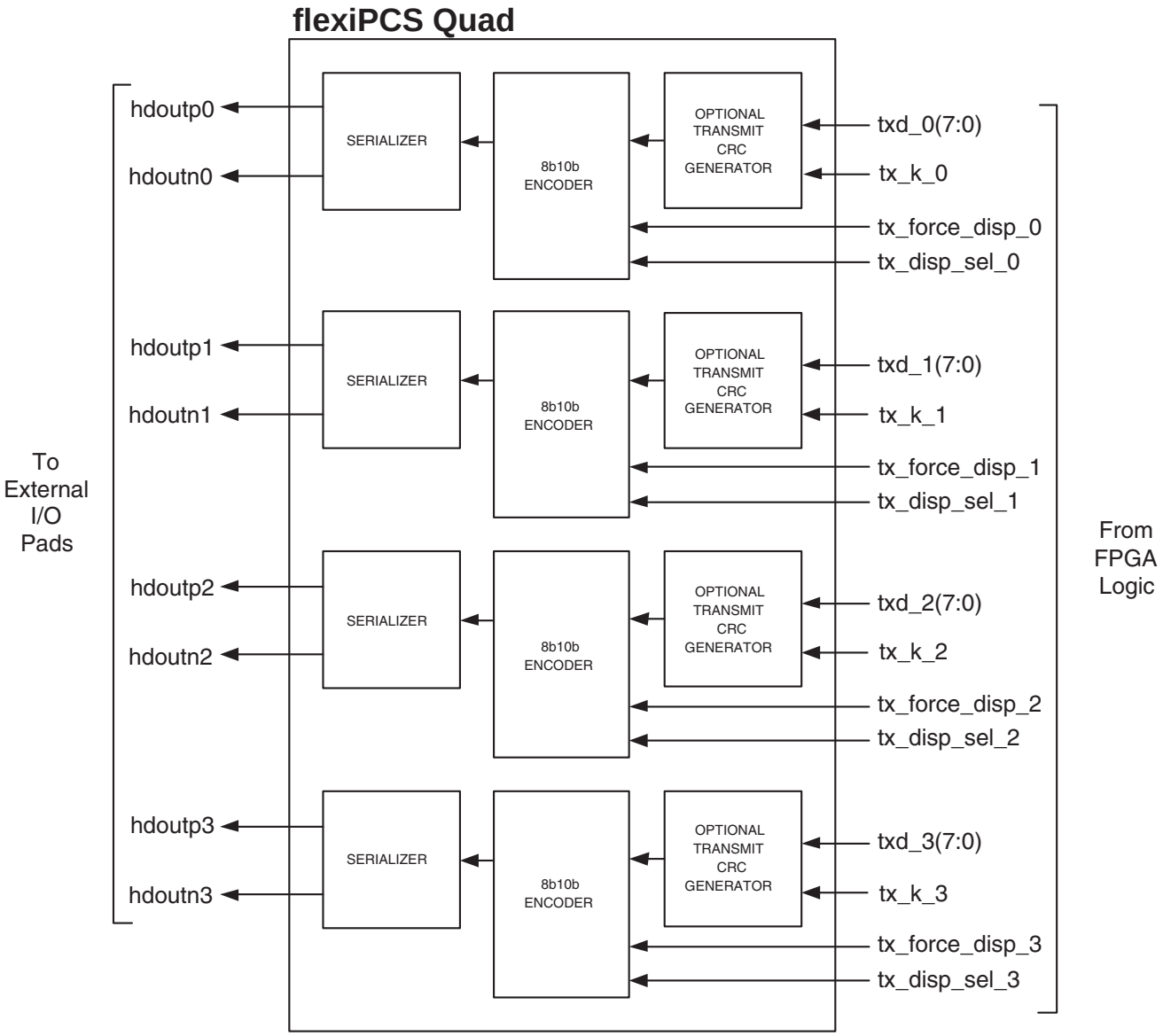
In 16 Bit Data Bus Mode, tx_force_disp is 2 bits wide (**tx_force_disp_[0-3](1:0)**) with tx_force_disp_[0-3](1) associated with txd_[0-3](15:8) and tx_force_disp_[0-3](0) associated with txd_[0-3](7:0).

**tx_disp_sel_[0-3]** - Per channel, disparity value supplied from FPGA logic. It is valid when tx_force_disp_[0-3] is high.

In 16 Bit Data Bus Mode, tx_disp_select is 2 bits wide (**tx_disp_sel_[0-3](1:0)**) with tx_disp_sel_[0-3](1) associated with txd_[0-3](15:8) and tx_disp_sel_[0-3](0) associated with txd_[0-3](7:0).

The flexiPCS quad in Generic 8b10b mode transmit data path consists of the following sub-blocks per channel: Optional Transmit CRC Generator, 8b10b Encoder, and Serializer. Figure 4-6 shows the four channels of transmit data paths in a flexiPCS quad.

*Figure 4-6. Generic 8b10b Transmit Path (1 Quad)*




**Optional CRC Generation**

An optional separate Cyclic Redundancy Code (CRC) generator is provided for all four transmit channels in a flexiPCS quad. To generate a quad which provides access to the CRC control and status pins at the FPGA interface, choose “Optional Direct Control & Status Register Access” when generating a PCS quad with the ispLEVER module generator.

The CRC generator supports the following Generic 8b10b polynomial:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

The CRC generator options are set on a per quad basis. The CRC generator options can be set for Generic 8b10b operation by writing Quad Interface Register Offset Address 0x1D to 0x30 (writing 0x1D to 0x00 disables the CRC generator/checker).

The following signals driven by the FPGA logic are used to control the transmit CRC logic on a per channel basis:

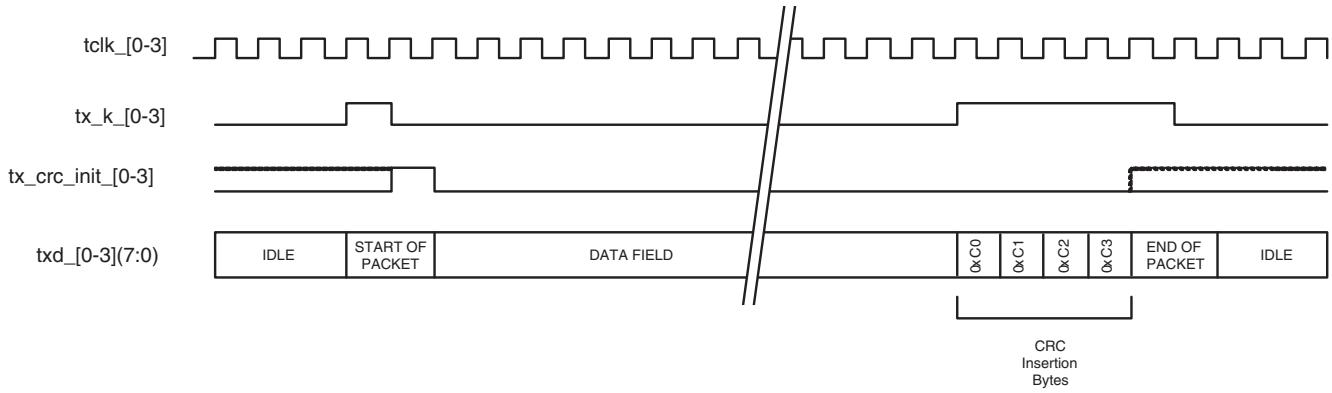
**tx_crc_init_[0-3]** - Per channel CRC initialization. When driven to '1', transmit CRC logic is re-initialized. When driven to '0', CRC is computed on incoming txd_[0-3](7:0) data.

In 16 Bit Data Bus Mode, tx_crc_init is 2 bits wide (**tx_crc_init_[0-3](1:0)**) with tx_crc_init_[0-3](1) associated with txd_[0-3](15:8) and tx_crc_init_[0-3](0) associated with txd_[0-3](7:0).

The tx_crc_init_[0-3] signal should remain low during the CRC calculation and insertion as shown in Figure 4-7.

At the point where the CRC is to be inserted, a K-C0, K-C1, K-C2, or K-C3 character should be placed onto the data bus (K=1, Data = 0xC0, 0xC1, 0xC2, 0xC3 are not valid codes in the 8b10b coding space). Bits 0-7 of the CRC will be substituted for 0xC0; bits 8-15 of the CRC will be substituted for 0xC1; bits 16-23 of the CRC will be substituted for 0xC2; and bits 24-31 of the CRC will be substituted for 0xC3.

***Figure 4-7. Generic 8b10b Transmit CRC Insertion***




Quad Interface Register Offset Address 0x1D controls the CRC generator options. The following options selectable for the CRC generator and the associated register bits are described in Table 4-4 below:

*Table 4-4. flexiPCS CRC Generator Options*

| | Bit | Function |
|---|---|---|
| Quad Interface Register Offset Address 0x1D | 0 | Reserved |
| | 1 | Set 8 bit initial value for CRC generator.<br><br>0 = all zeros (0x00)<br>1 = all ones (0xFF) |
| | [2:3] | Set mode for CRC generator/checker<br><br>00 = Bypass CRC<br>01 = Gigabit Ethernet Mode<br>10 = Fibre Channel Mode<br>11 = Generic 8b10b Mode |
| | 4 | Select/deselect inversion of data (0->1 and 1->0) to CRC generator<br><br>0 = do not invert data input to CRC generator<br>1 = invert data to CRC generator (bitwise inversion) |
| | 5 | Select/deselect bit order swap of 8-bit input data to CRC generator<br><br>0 = do not swap bit order of input data to CRC generator<br>1 = swap bit order (swap bit 7 and bit 0, swap bit 6 and bit 1, and so on) of input data to CRC generator |
| | 6 | Select/deselect inversion of output (0->1 and 1->0) from CRC generator<br><br>0 = do not invert output from CRC generator<br>1 = invert output from CRC generator (bitwise inversion) |
| | 7 | Select/deselect bit order swap of 8-bit output from CRC generator<br><br>0 = do not swap bit order of output data from CRC generator<br>1 = swap bit order (swap bit 7 and bit 0, swap bit 6 and bit 1, and so on) of output data from CRC generator |

### 8b10b Encoding

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 the specification, along with maintaining the running disparity rules as specified.

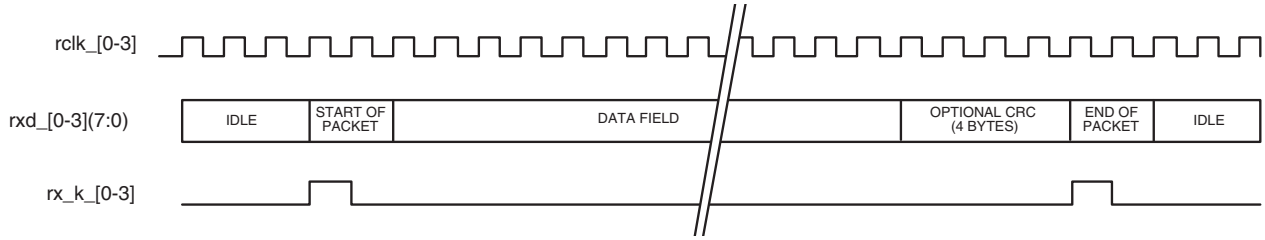
### Serializer

The 8b10b encoded data undergoes parallel to serial conversion and is transmitted off chip via the embedded SERDES. For detailed information on the operation of the LatticeSC SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

## Receive Data

When configured into Generic 8b10b mode, a flexiPCS quad provides four receive data paths from a high-speed serial line interface at the device inputs to an 8-bit parallel interface at the FPGA logic interface as shown in Figure 4-8. This figure shows an example where the SOP and EOP phases are two characters long with the first character being a control character. The length and value of SOP and EOP is programmable via quad interface registers. The generic 8b10b autoconfig example provides details on setting SOP and EOP length and value.

*Figure 4-8. Generic 8b10b Receive FPGA Interface Signals*




**rxd_[0-3](7:0)** - Per channel receive data to the FPGA interface. Each quad supports up to 4 independent channels of 8-bit wide parallel data. The rxd_[0-3] signal transitions synchronously with respect to rclk_[0-3].

In 16-Bit Data Bus Mode, this bus is 16-bits wide (**rxd_[0-3](15:0)**).

**rx_k_[0-3]** - Per channel, active high control character indicator.

In 16 Bit Data Bus Mode, rx_k is 2 bits wide (**rx_k_[0-3](1:0)**) with rx_k_[0-3](1) associated with rxd_[0-3](15:8) and rx_k_[0-3](0) associated with rxd_[0-3](7:0).

**rx_disp_err_detect_[0-3]** - Per channel, active high signal driven by the flexiPCS to indicate a disparity error was detected with the associated data.

In 16 Bit Data Bus Mode, rx_disp_err_detect is 2 bits wide (**rx_disp_err_detect_[0-3](1:0)**) with rx_disp_err_detect_[0-3](1) associated with rxd_[0-3](15:8) and rx_disp_err_detect_[0-3](0) associated with rxd_[0-3](7:0).

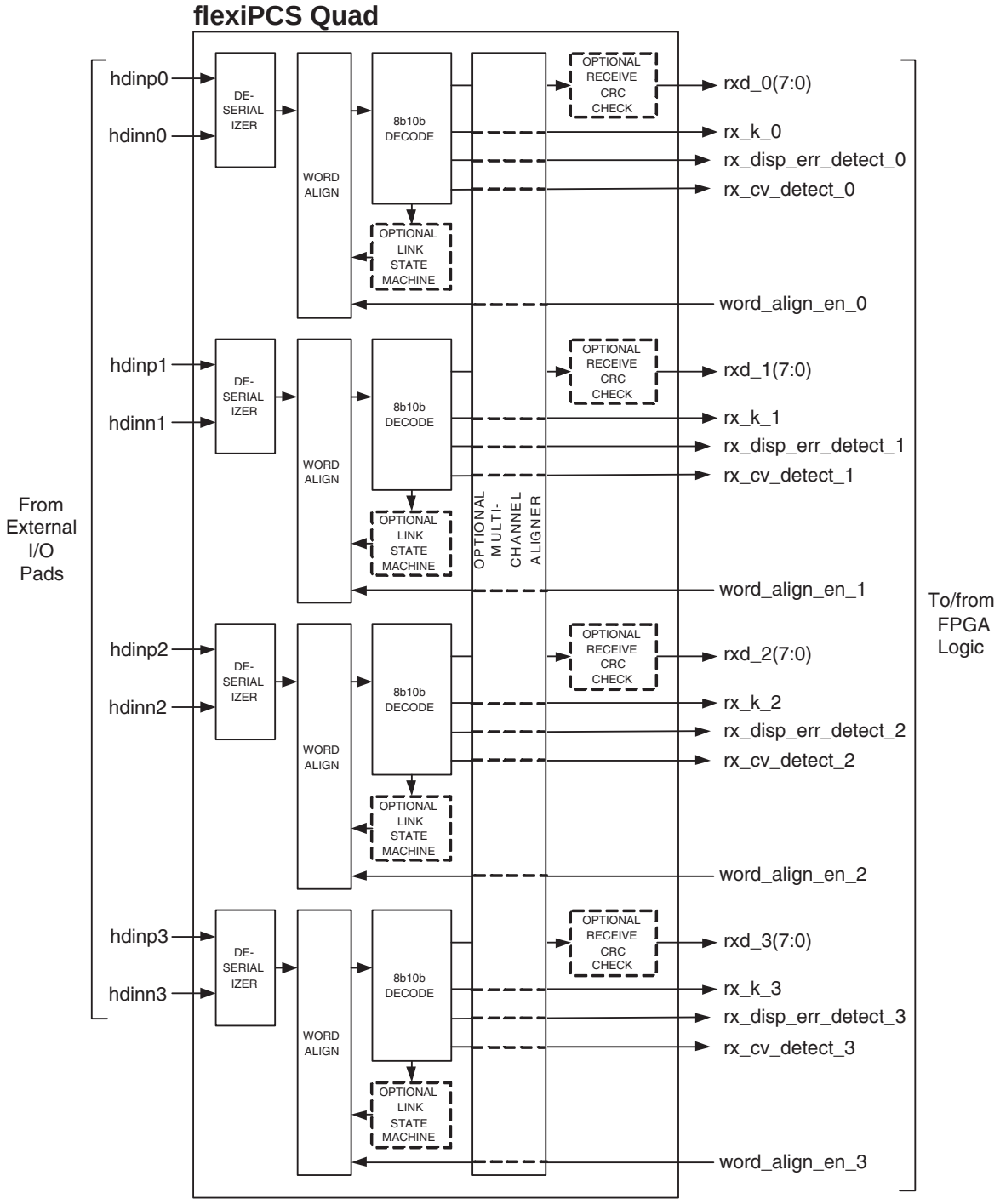
**rx_cv_detect_[0-3]** - Per channel, active high signal driven by the flexiPCS to indicate a code violation was detected with the associated data.

In 16 Bit Data Bus Mode, rx_cv_detect is 2 bits wide (**rx_cv_detect_[0-3](1:0)**) with rx_cv_detect_[0-3](1) associated with rxd_[0-3](15:8) and rx_cv_detect_[0-3](0) associated with rxd_[0-3](7:0).

**word_align_en_[0-3]** - Signal which unlocks the word aligner on any minimum one clock cycle wide low pulse for the appropriate receive channel.

The flexiPCS quad in Generic 8b10b mode receive data path consists of the following sub-blocks per channel: Deserializer, Word Aligner, 8b10b Decoder, Optional Link State Machine, and Optional Receive CRC Checker. In addition, an Optional Multi-channel Aligner allows for 2 or 4 channel alignment within a quad. Figure 4-9 shows the four channels of receive data paths in a flexiPCS quad in Generic 8b10b mode.

*Figure 4-9. Generic 8b10b Receive Data Path (1 Quad)*




**Deserializer**

Data is brought on chip to the embedded SERDES where it undergoes serial to parallel. For detailed information on the operation of the LatticeSC SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

**Word Alignment**

The word aligner module performs the alignment code word detection and alignment operation. The word alignment character is used by the receive logic to perform 10-bit word alignment upon the incoming data stream.

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

- 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, in Generic 8b10b Mode, the user can define any bit pattern up to 10 bits long. Figure 4-5 shows the relationship between the word alignment character bit order and the corresponding encoded 8b10b character bit order.
- The first word alignment character bits 7 to 0 can be set by writing Quad Interface Register Offset Address 0x15 bits [0:7]. The first word alignment character bits 8 and 9 can be set by writing Quad Interface Register Offset Address 0x17 bits 5 and 4 respectively.
- The second word alignment character bits 7 to 0 can be set by writing Quad Interface Register Offset Address 0x16 bits [0:7]. The second word alignment character bits 8 and 9 can be set by writing Quad Interface Register Offset Address 0x17 bits 7 and 6 respectively.
- The mask register can be set to compare word alignment bits 7 to 0 by writing Quad Interface Register Offset Address 0x14 bits [0:7]to 0xFF (a '1' in a bit of the mask register means check the corresponding bit in the word alignment character register). The mask register can be set to compare word alignment bits 8 and 9 by writing Quad Interface Register Offset Address 0x17, bits 3 and 2 respectively to a '1'.

***Table 4-5. Map of Word Alignment Character Register Bit Order to 8b10b Encoded Bit Order***

| Word Character Alignment Register | Word Character Alignment Register Bit | Corresponding 8b10b Encoded Character Bit[1] |
|---|---|---|
| Quad Interface Register Offset Address 0x17 Bits [4:5] (first), Bits [6:7] (second) | 9 | j |
| | 8 | h |
| Quad Interface Register Offset Address 0x15 (First word alignment character) Quad Interface Register Offset Address 0x16 (Second word alignment character) | 7 | g |
| | 6 | f |
| | 5 | i |
| | 4 | e |
| | 3 | d |
| | 2 | c |
| | 1 | b |
| | 0 | a |

1. Bit designations for encoded 8b10b values in Tables 36-1 and 36-2 in the IEEE 802.3ae-2002 1000 BASE-X Specification

It is assumed that most applications using a flexiPCS quad in Generic 8b10b Mode would not make use of a protocol specific Link State Machine in conjunction with word alignment. If this is the case, the Link State Machines for all channels must be disabled by writing Quad Interface Register Offset Address 0x19, bit 0 to a '1'.

If all Link State Machines are deselected (Quad Interface Register Offset Address 0x19, bit 0 set to '1'), the word aligner will lock alignment (it will stop comparing the incoming data to the user-defined word alignment characters and will maintain current alignment) on the first successful compare to either of the two user-defined alignment words after **word_align_en** is pulsed low for at least one clock cycle. Any subsequent low pulse on the **word_align_en** pin at the FPGA interface will un-lock the word aligner. The word aligner will then re-lock on the

next match to one of the user-defined word alignment characters. If desired, the **word_align_en** can be controlled by a Link State Machine implemented externally to the flexiPCS quad to allow a change in word alignment only under specific conditions.

If all Link State Machines are selected (Quad Interface Register Offset Address 0x19, bit 0 set to '0'), and a particular channel's Link State Machine is not enabled (that channel's Channel Interface Register Offset Address 0x00, bit 7 is set to '0'), then the channel's word aligner will align to the on the first successful compare to either the first or second user-defined word alignment character.

If all Link State Machines are selected (Quad Interface Register Offset Address 0x19, bit 0 set to '0'), and a particular channel's Link State Machine is enabled (that channel's Channel Interface Register Offset Address 0x00, bit 7 is set to '1'), then 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 in Generic 8b10b Mode, see the "Optional Link State Machine" description below.

**8b10b Decoder**
The 8b10b decoder implements an 8b10b decoder 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.

When a code violation is detected, the rxd data is set to 0xEE. When a disparity error is detected in a given data channel, the data is passed to that channel's rxd pins and rx_disp_err_detect goes to '1'.

**Optional Link State Machine**
The flexiPCS implements link state machines for various protocols that are normally used in other quad modes. However, one of the link state machines can be enabled for use in Generic 8b10b Mode if desired.

If Quad Interface Register Offset Address 0x19, bit 0 is written to '0' (the default), then one of the protocol based Link State machines will control word alignment. If this is the case, then a particular channel's Link State Machine must be enabled by writing the appropriate Channel Interface Register Offset Address 0x00, bit 7 to '1' for word alignment to occur. Selection of the specific Link State Machine to be enabled is described below and summarized in Figure 4-11.

The Link State Machine for Gigabit Ethernet will be enabled by default when Quad Interface Register Offset Address 0x18 has been set to 0x10. For more information on the operation of the Gigabit Ethernet Link State Machine, refer to the Gigabit Ethernet Mode section of the LatticeSC/M Family flexiPCS Data Sheet.

The Link State Machine for Fibre Channel will be enabled if Quad Interface Register Offset Address 0x18 is set to 0x18. For more information on the operation of the Fibre Channel Link State Machine, refer to the Fibre Channel Mode section of the LatticeSC/M Family flexiPCS Data Sheet.

The Link State Machine for XAUI will be enabled if Quad Interface Register Offset Address 0x18 is set to 0x11. For more information on the operation of the XAUI Link State Machine, refer to the XAUI Mode section of the LatticeSC/M Family flexiPCS Data Sheet.

The Link State Machine for Serial RapidIO will be enabled if Quad Interface Register Offset Address 0x18 is set to 0x12. For more information on the operation of the Serial RapidIO Link State Machine, refer to the Serial RapidIO Mode section of the LatticeSC/M Family flexiPCS Data Sheet. Note that this will set the Multi-channel Alignment state machine to Serial RapidIO mode as well if the Optional Multi-channel Aligner is being used.

Figure 4-10 illustrates the link state machine options.

***Figure 4-10. flexiPCS Word Aligner Control***

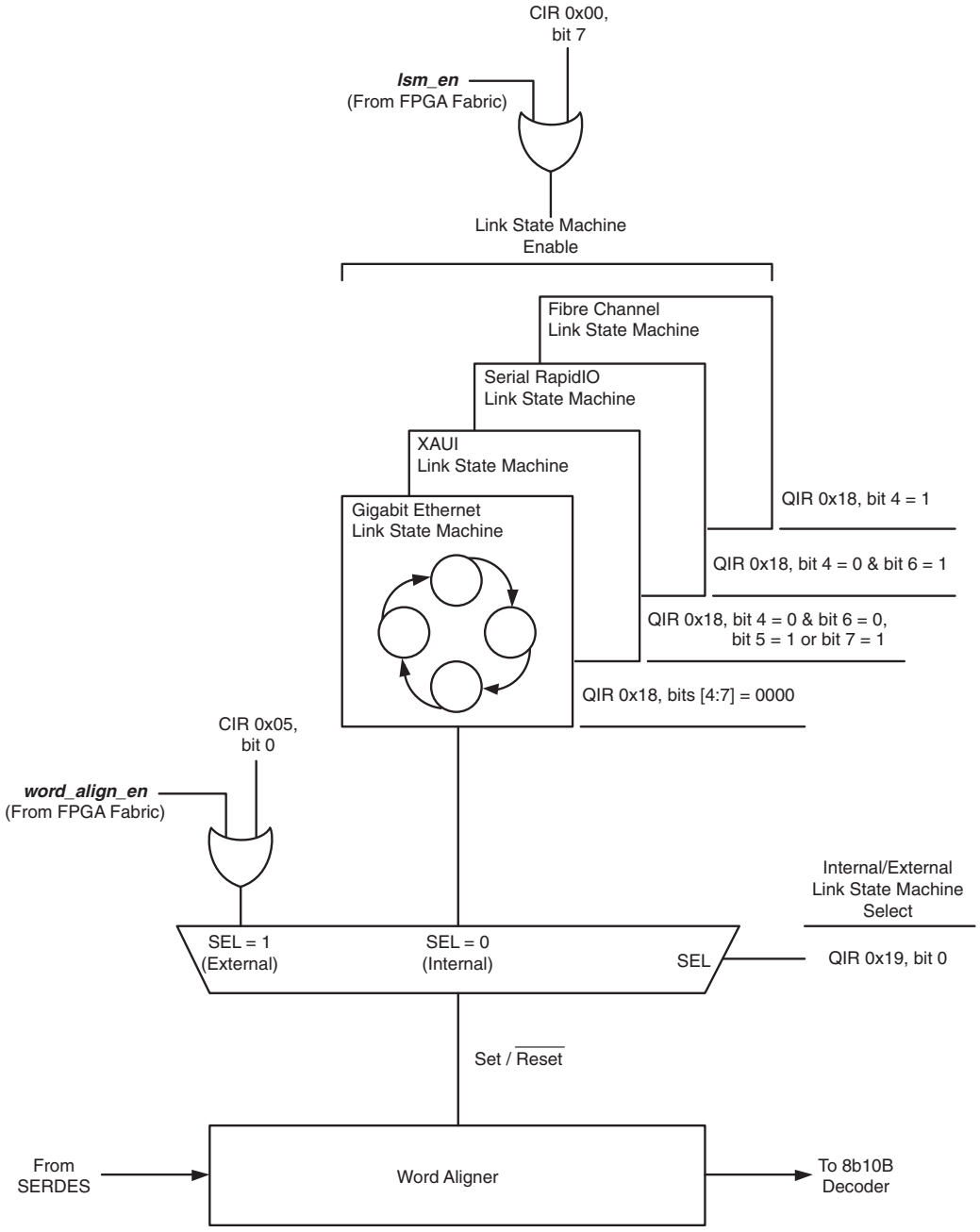


***Figure 4-11. Link State Machine Selection for Generic 8b10b Mode***

| Quad and Channel Interface Register Set-up | | Word Alignment Control |
|---|---|---|
| QIR 0x19, bit 0 = '1' | | External Link State Machines selected<br><br>Word alignment automatically locked on first alignment character match after low pulse on **word_align_en**. Lock/unlock controlled by a change on the per channel **word_align_en** signal at FPGA interface (or CIR 0x05, bit 0). |
| QIR 0x19, bit 0 = '0' (default)<br><br>CIR 0x00, bit 7 = '0' (default) | | Internal Link State Machines selected<br><br>No internal Link State Machines enabled<br><br>Word alignment not locked by a link state machine. Word alignment will occur on any alignment character match. |
| QIR 0x19, bit 0 = '0' (default)<br><br>and<br><br>Appropriate CIR 0x00, bit 7 = '1' | QIR 0x18 = 0x10 | Gigabit Ethernet Link State Machine selected and enabled for appropriate channel<br><br>Word alignment follows Gigabit Ethernet word alignment protocol |
| | QIR 0x18 = 0x18 | Fibre Channel Link State Machine selected and enabled for appropriate channel<br><br>Word alignment follows Fibre Channel word alignment protocol |
| | QIR 0x18 = 0x11 | XAUI Link State Machine selected and enabled for appropriate channel<br><br>Word alignment follows XAUI word alignment protocol |
| | QIR 0x18 = 0x12 | Serial RapidIO Link State Machine selected and enabled for appropriate channel<br><br>Word alignment follows Serial RapidIO word alignment protocol |

When a Link State Machine is selected and enabled, a link synchronization status register bit will go high upon successful link synchronization. This link synchronization status can be read from the Quad Interface Register Offset Address 0x84, bits [4:7] (bit 4 for channel 0, bit 5 for channel 1, bit 4 for channel 2, and bit 7 for channel 3). If the link synchronization state machine is disabled, this status bit will always remain high.

**Optional Multi-channel Aligner**

The Multi-channel Alignment module performs the operations and functions to control the Multi-channel Alignment process. To generate a quad which provides access to the Multi-channel Aligner control and status pins at the FPGA interface, choose "Optional Direct Control & Status Register Access" when generating a PCS quad with IPexpress.

The flexiPCS Multi-channel Aligner allows for alignment of two, three, or four channels in a quad. The Multi-channel Aligner will align channels based on a user-defined alignment character which must be present in the data stream for all receive channels that are to be aligned. Typically this character is inserted simultaneously in all channels during an Idle sequence between packets upon transmission. If arriving data is skewed due to unequal transport delays, the presence of the alignment characters allows the Multi-channel Aligner to re-align the skewed channels.

The following is a procedure for settings registers for Multi-channel Alignment.

**Two Channel Alignment**

A single LatticeSC quad can support up to two aligned links with the use of the embedded multi-channel aligner. This is done by using a flexiPCS quad in Generic 8b10b Mode and setting the Multi-channel Aligner registers appropriately.

For two channel alignment, the following is enabled in Generic 8b10b Mode:

1. For two channel alignment, at least 2 channels must be enabled (Channel 0 and Channel 1 and/or Channel 2 and Channel 3).

2. For two channel alignment, the Multi-channel Aligner is enabled in the Receive direction to provide alignment between two channels as dictated by the COM lane align symbol as defined in the Generic 8b10b Base Specification.

The following is a procedure for settings registers for two channel alignment while in Generic 8b10b Mode.

1. Set the mode for the Multi-channel Aligner and enable/disable the appropriate channels for alignment. The multi-channel aligner can be set to align two channels in a quad by writing to Quad Interface Register Offset Address 0x19.

- Bit 3 controls alignment between Channel 0 and Channel 1. When bit 3 is set to '1', Channels 0 and 1 will be aligned. Figure 4-12 shows an example of the operation of the flexiPCS multi-channel aligner operation for channel 0 and channel 1 alignment. The alignment characters in each channel are lined up by the multi-channel aligner to create an aligned data stream at the PCS/FPGA interface.
- Bit 2 controls alignment between Channel 2 and Channel 3. When bit 2 is set to '1', Channels 2 and 3 will be aligned.
- When both bit 2 and bit 3 are both set to '1', channels 0 and 1 are aligned to each other and channels 2 and 3 are aligned to each other, but channels 0/1 and channels 2/3 are not aligned together. Figure 4-13 shows an example of the operation of the flexiPCS multi-channel aligner operation when channel 0 and 1 alignment and channel 2 and 3 alignment are both selected.
- Each channel to be aligned must also be enabled for alignment by writing to the appropriate bit in Quad Interface Register Offset Address 0x04. Write a '1' to bit 7 to include channel 0 for alignment. Write a '1' to bit 6 to include channel 1 for alignment. Write a '1' to bit 5 to include channel 2 for alignment. Write a '1' to bit 4 to include channel 3 for alignment. Multi-channel alignment can also be enabled by setting the appropriate **mca_align_en_[0-3]** ports at the FPGA interface high.

***Figure 4-12. Generic 8b10b Two Channel Alignment Example (Channels 0 and 1)***

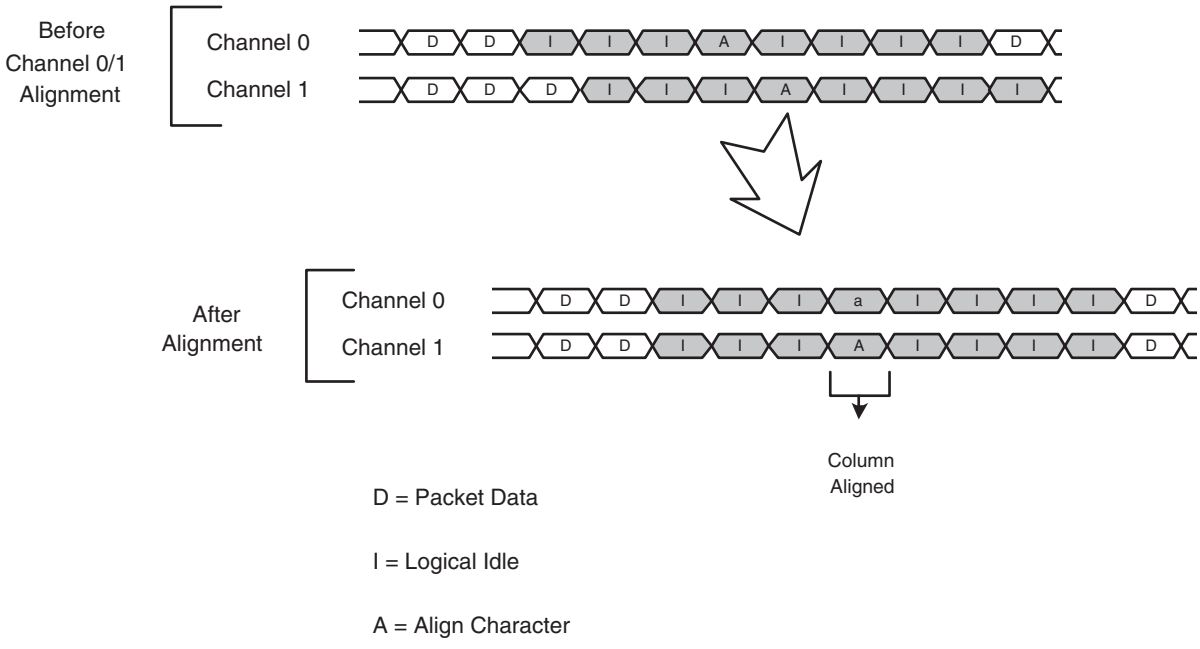
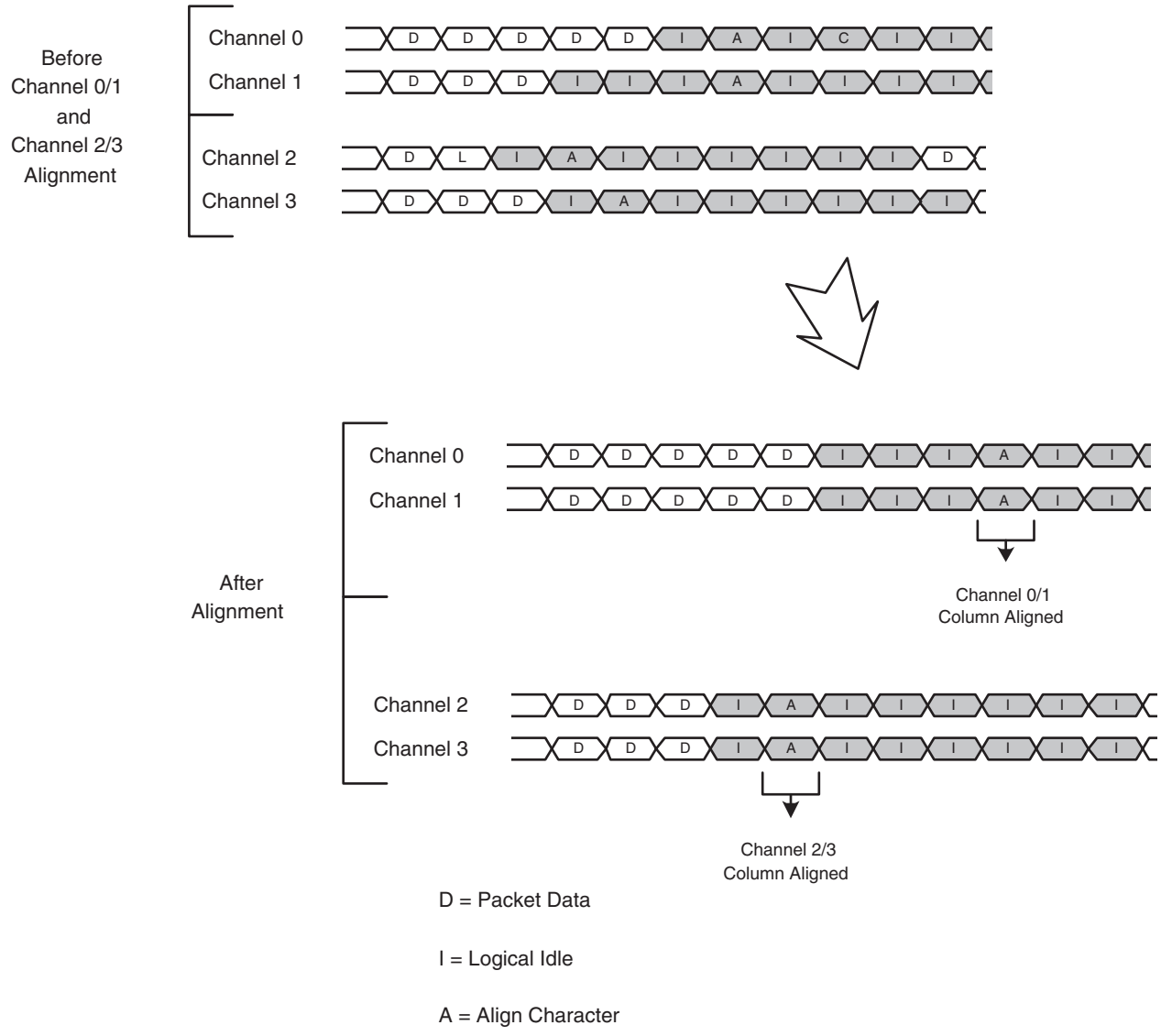


***Figure 4-13. Generic 8b10b Dual Two Channel Alignment Example (channels 0/1 and channels 2/3)***




2. Set the Multi-channel Alignment output clocks. A clock domain transfer occurs between the inputs and outputs of the Multi-channel Aligner. Each channel's receive data is clocked into the Multi-channel Aligner with its own separate recovered receive clock. Each channel's receive data is clocked out of the Multi-channel Aligner on a unique multi-channel receive clock. Each multi-channel receive clock can be programmed to be any of the recovered receive clocks.

It is expected that the user will assign the same recovered receive clock to all the multi-channel output clocks for every channel that is to be aligned. That way, all aligned channels coming out of the Multi-channel Aligner are effectively clocked by the same clock. For more information on setting up a multi-channel receive clock, refer to the **Multi-Channel Alignment** section of the LatticeSC/M Family flexiPCS Data Sheet.

For example, in a 2 channel alignment application (Channels 0/1), in order to set up the Multi-channel Alignment clocks for channels 0 and 1 to be sourced from the channel 0 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xE0. For a 2 channel alignment application (Channels 2/3), in order to set up the Multi-channel Alignment clocks for channels 2 and 3 to be sourced from the channel 2 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xA4. For a dual 2 channel alignment application (Channels 0/1 and

channels 2/3), in order to set up the Multi-channel Alignment clocks for channels 0 and 1 to be sourced from the channel 0 recovered receive clock and the Multi-channel Alignment clocks for channels 2 and 3 to be sourced from the channel 2 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xA0.

3. Set the Multi-channel Alignment characters. The Multi-channel Aligner provides the ability to align channels of incoming data to one of two 10-bit user defined maskable patterns. Both alignment characters should be set to the same value if only one alignment character is defined for an application.

- A user programmable mask word allows the user to set which individual bits are compared to the user defined channel alignment character. When a '1' is present in the user programmable mask, the corresponding bit of the channel alignment characters is compared. When '0' is present in the user programmable mask, the corresponding bit of the channel alignment characters is not compared.
- The lowest 8 significant bits [7:0] of the first multi-channel alignment character are written to Quad Interface Register Offset Address 0x08, bits [0:7]. Bits [9:8] of the first multi-channel alignment character are written to Quad Interface Register Offset Address 0x0A, bits [4:5].
- The lowest 8 significant bits [7:0] of the second multi-channel alignment character are written to Quad Interface Register Offset Address 0x09, bits [0:7]. Bits [9:8] of the second multi-channel alignment character are written to Quad Interface Register Offset Address 0x0A, bits [6:7].

4. Set the Multi-channel Aligner FIFO latency and high-water mark values. Latency values from 0 to 31 can be set by writing to Quad Interface Register Offset Address 0x05, bits [3:7]. Setting a latency value of 0x1F will minimize chance of FIFO overrun or underrun. The FIFO high-water mark values from 0 to 63 can be set by writing to Quad Interface Register Offset Address 0x06, bits [2:7]. The FIFO high-water mark should be set to the maximum the number of bytes of skew allowed for de-skewing. Larger values of FIFO high-water mark increase the chance of FIFO overrun or underrun. For more information on choosing latency and high-water mark values, refer to the Multi-Channel Alignment section of the LatticeSC/M Family flexiPCS Data Sheet.

5. Reset the Multi-channel Aligner state machine and FIFO control logic if desired with the **mca_resync_[01/23]** ports at the FPGA interface. **mca_resync_01** is an active high asynchronous reset which resets the Multi-channel Aligner for channels 0 and 1. **mca_resync_23** is an active high asynchronous reset which resets the Multi-channel Aligner for channels 2 and 3.

**Four Channel Alignment**

A single LatticeSC quad can support a single four channel aligned link with the use of the embedded multi-channel aligner. This is done by using a flexiPCS quad in Generic 8b10b Mode and setting the Multi-channel Aligner block appropriately.

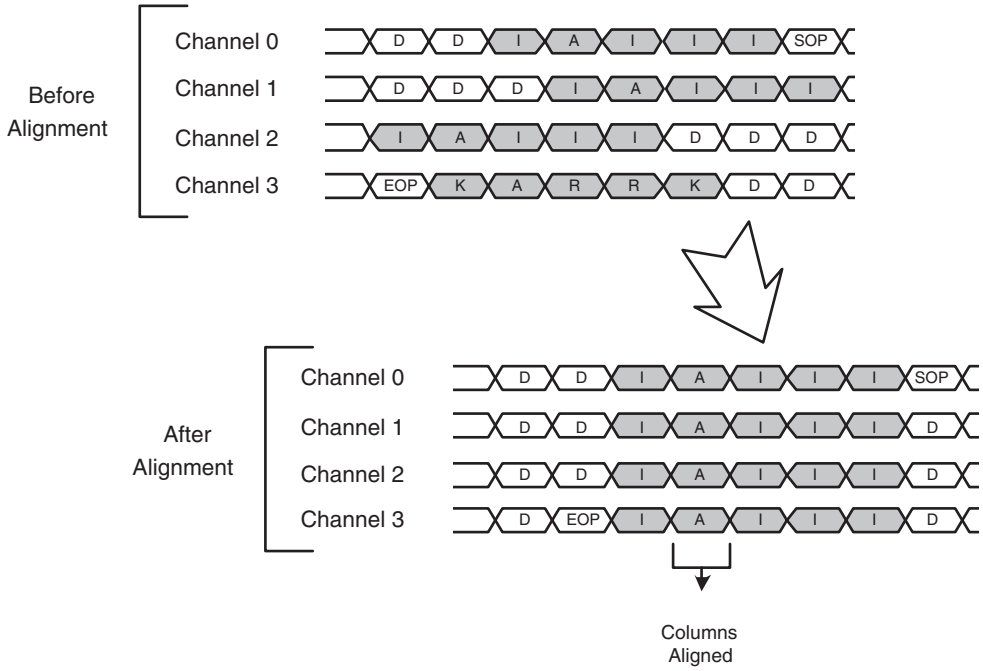
The main differences between two channel alignment and four channel alignment are:

1. For four channel alignment, all four channels in a flexiPCS QUAD must be enabled.

2. For four channel alignment, the Multi-channel Aligner is enabled in the Receive direction to provide alignment between all four channels as dictated by the user-defined alignment character.

Figure 4-14 shows an example of the operation of the flexiPCS multi-channel aligner operation for four channel alignment. The alignment character in each channel are lined up by the multi-channel aligner to create an aligned data stream at the PCS/FPGA interface.

***Figure 4-14. Generic 8b10b Four Channel Alignment Example***




D = Packet Data

SOP = Start of Packet

EOP = End of Packet

I = Logical Idle

A = Align Character

The following is a procedure for settings registers for Multi-channel Alignment.

1. Set the mode for the Multi-channel Aligner and enable/disable the appropriate channels for alignment.

- Set 4 channel alignment for channels 0, 1, 2, and 3 by writing Quad Interface Register Offset Address 0x19, bit 1 to a '1'.
- Each channel to be aligned must also be enabled for alignment by writing to the appropriate bit in Quad Interface Register Offset Address 0x04. Write a '1' to bit 7 to include channel 0 for alignment. Write a '1' to bit 6 to include channel 1 for alignment. Write a '1' to bit 5 to include channel 2 for alignment. Write a '1' to bit 4 to include channel 3 for alignment. Multi-channel alignment can also be enabled by setting the **mca_align_en** pin at the FPGA interface high.

2. Set the Multi-channel Alignment output clocks. A clock domain transfer occurs between the inputs and outputs of the Multi-channel Aligner. Each channel's receive data is clocked into the Multi-channel Aligner with its own separate recovered receive clock. Each channel's receive data is clocked out of the Multi-channel Aligner on a unique multi-channel receive clock. Each multi-channel receive clock can be programmed to be any of the recovered receive clocks.

It is expected that the user will assign the same recovered receive clock to all the multi-channel output clocks for every channel that is to be aligned. That way, all aligned channels coming out of the Multi-channel Aligner are

effectively clocked by the same clock. For more information on setting up a multi-channel receive clock, refer to the **Multi-Channel Alignment** section of the LatticeSC/M Family flexiPCS Data Sheet.

For example, in a 4 channel alignment application, in order to set up all Multi-channel Alignment clocks to be sourced from the channel 0 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0x00. To set all Multi-channel Alignment clocks to be sourced from the channel 1 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0x55. To set all Multi-channel Alignment clocks to be sourced from the channel 2 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xAA. To set all Multi-channel Alignment clocks to be sourced from the channel 3 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xFF.

3. Set the Multi-channel Alignment characters as indicated in step 3 of the **Two Channel Alignment** section.

4. Set the Multi-channel Aligner FIFO latency and high-water mark values as indicated in step 4 of the **Two Channel Alignment** section.

5. Reset the Multi-channel Aligner state machine and FIFO control logic if desired with the **mca_resync** pin at the FPGA interface. **mca_resync** is an active high asynchronous reset which resets the Multi-channel Aligner for all four channels.

When the Multi-channel Aligner is set to 4-channel alignment, the Multi-channel Alignment state machine for all four channels can be disabled by writing Quad Interface Register Offset Address 0x04, bit 3 to a '1'. When the Multi-channel Alignment state machine is disabled, the alignment state machine will be held in the "LOSS_OF_ALIGNMENT" state and no alignment will be performed for the relevant channels.

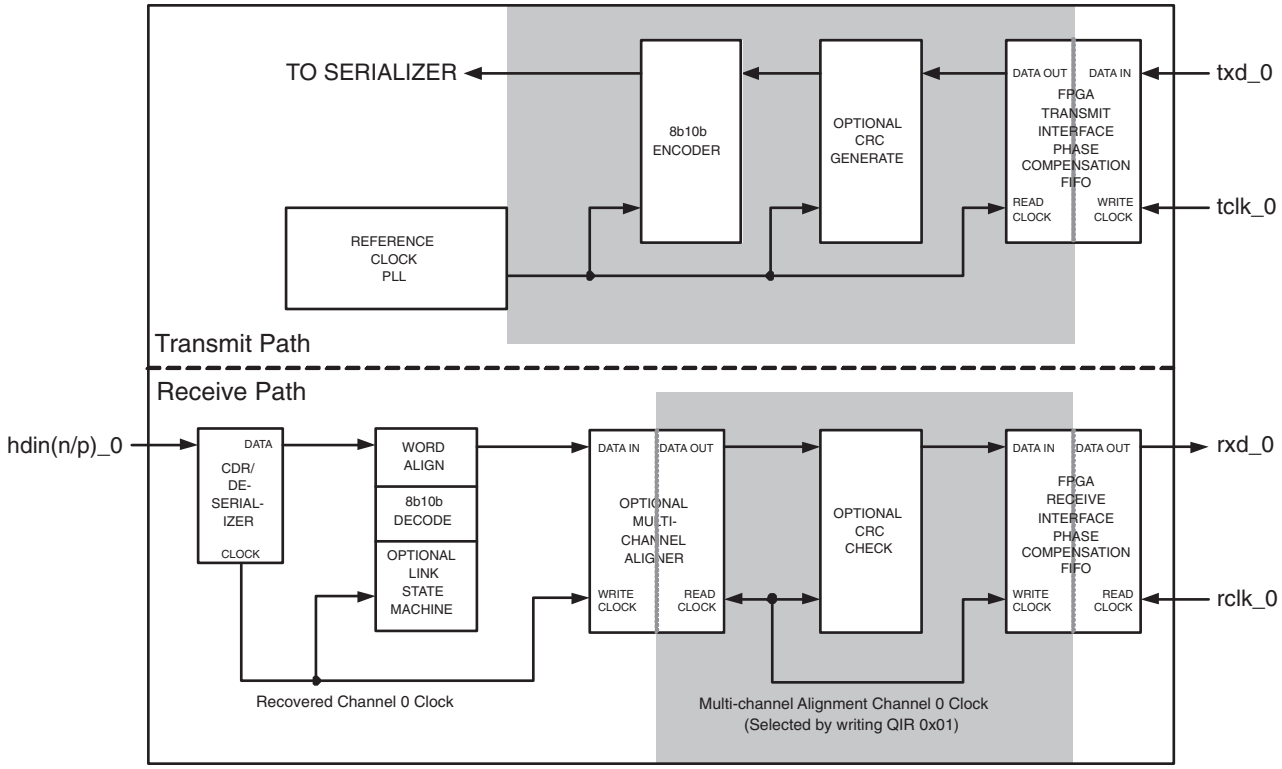
**Multi-quad and Multi-chip Alignment**
More than four channels can be aligned by setting up one or two LatticeSC devices for multi-quad and/of multi-chip alignment. For more information on multi-quad and multi-chip alignment, refer to the **Multi-Channel Alignment** section of the LatticeSC/M Family flexiPCS Data Sheet.

Figure 4-15 shows the clock domains for both transmit and receive directions for a single channel inside the flexiPCS.

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

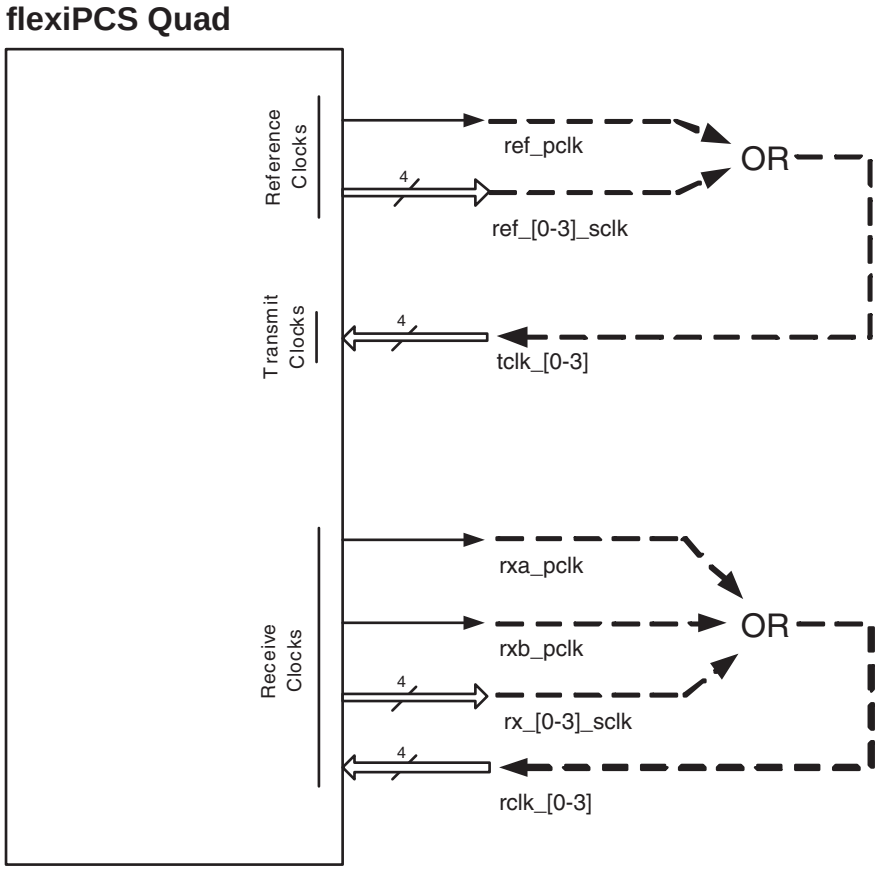
On the receive side, a clock domain transfer from the recovered channel clocks to the Multi-channel Alignment channel clocks occurs at the Multi-channel Aligner. A second clock domain transfer occurs from the Multi-channel Alignment channel clocks to the rclk input at the FPGA interface in the FPGA Receive Interface phase compensation FIFO. The FPGA Receive Interface phase compensation FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. These phase compensation FIFOs (one per channel) cannot compensate for frequency variations.

***Figure 4-15. flexiPCS Clock Domain Transfers for Generic 8b10b Mode***




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 flexiPCS quad set to Generic 8b10b mode. Figure 4-16 illustrates the possible connections that would result in a synchronous interface.

*Figure 4-16. Synchronous input clocks to flexiPCS quad set to Generic 8b10b Mode*




**Optional CRC Checker**

The Generic 8b10b CRC checker automatically checks for user-defined Start-of-Packet ordered sets and begins a CRC calculation immediately after the Start-of-Packet ordered set. The CRC checker also automatically checks for user defined End-of-Packet ordered sets and compares the CRC values calculated on the receive data with the value embedded in the data stream just prior to the End-of-Packet ordered set. Figure 4-17 shows the timing relationships of the CRC checker outputs to the FPGA interface.

The user defined Start-of-Packet (SOP) and End-of-Packet (EOP) ordered sets are set as follows:

- The length of the user defined SOP and EOP ordered sets can be set to 1 or 2 characters long. Both SOP and EOP ordered sets are set to the same length. By default, the length of the SOP and EOP ordered sets is set to 2 characters. The length of the SOP and EOP ordered sets can be set to 1 character for all channels by writing the Quad Interface Register Offset Address 0x1E, bit 7 to a '1'.
- An SOP mask register can be set to compare SOP character bits 7 to 0 by writing Quad Interface Register Offset Address 0x1F bits [0:7] to 0xFF (a '1' in a bit of the mask register means check the corresponding bit in the SOP character registers, a '0' means ignore the corresponding bit in the SOP character registers). The mask register can be set to compare SOP character register bit 8 by writing Quad Interface Register Offset Address 0x22, bit 5 to a '1'.
- Two 9-bit SOP ordered set characters can be set. If the SOP ordered set length has been set to 1 character, then the SOP ordered set character bits 7 to 0 are set by writing to Quad Interface Register Offset Address 0x20, bits [0:7]. SOP ordered set character bit 8 is set by writing to Quad Interface Register Offset Address 0x22, bit 7. If the SOP ordered set length has been set to 2 characters, then these registers hold the value for the first received SOP character. The second received SOP character bits 7 to 0 is then set by writing to Quad Interface Register

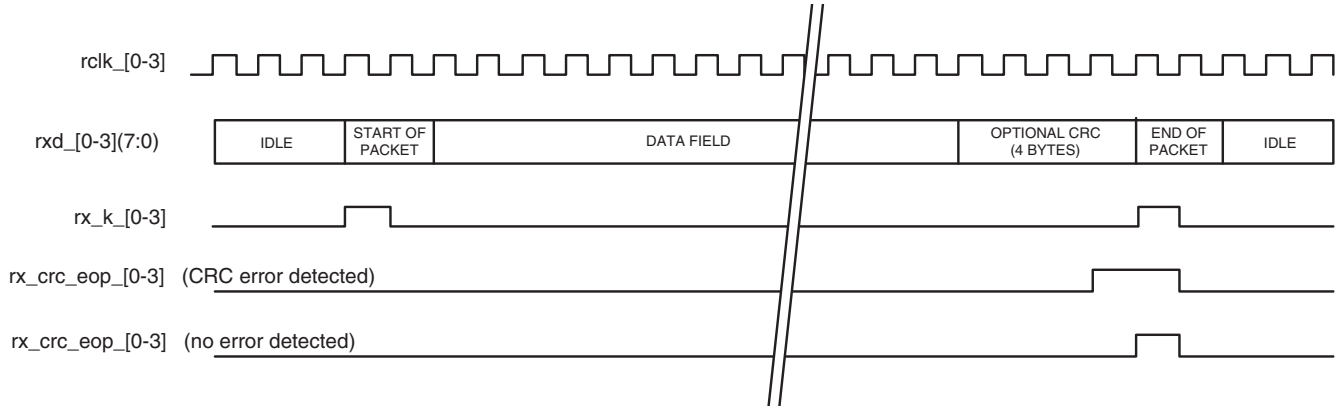
Offset Address 0x21, bits [0:7]. The second SOP ordered set character bit 8 is set by writing to Quad Interface Register Offset Address 0x22, bit 6.

- An EOP mask register can be set to compare EOP character bits 7 to 0 by writing Quad Interface Register Offset Address 0x23 bits [0:7] to 0xFF (a '1' in a bit of the mask register means check the corresponding bit in the EOP character registers, a '0' means ignore the corresponding bit in the EOP character registers). The mask register can be set to compare EOP character register bit 8 by writing Quad Interface Register Offset Address 0x26, bit 5 to a '1'.
- Two 9-bit EOP ordered set characters can be set. If the EOP ordered set length has been set to 1 character, then the EOP ordered set character bits 7 to 0 are set by writing to Quad Interface Register Offset Address 0x24, bits [0:7]. EOP ordered set character bit 8 is set by writing to Quad Interface Register Offset Address 0x26, bit 7. If the EOP ordered set length has been set to 2 characters, then these registers hold the value for the first received EOP character. The second received EOP character bits 7 to 0 are then set by writing to Quad Interface Register Offset Address 0x25, bits [0:7]. The second EOP ordered set character bit 8 is set by writing to Quad Interface Register Offset Address 0x26, bit 6.

**rx_crc_eop_[0-3]** - Per channel active high signal which indicates whether a CRC error was detected. When end of packet is reached, calculated CRC is compared with the expected values. If there is no error, rx_crc_eop_[0-3] acts as an end of packet indicator and goes high for one rclk_[0-3] cycle. If the packet was in error, then rx_crc_eop_[0-3] is asserted for two clock cycles as shown in Figure 4-17.

In 16 Bit Data Bus Mode, rx_crc_eop is 2 bits wide (**rx_crc_eop_[0-3](1:0)**) with rx_crc_eop_[0-3](1) associated with rxd_[0-3](15:8) and rx_crc_eop_[0-3](0) associated with rxd_[0-3](7:0).

***Figure 4-17. Generic 8b10b Receive CRC Error Detection***




The CRC checker options are set on a per quad basis. Quad Interface Register Offset Address 0x1E controls the CRC checker options. The following options selectable for the CRC generator and the associated register bits are described in Table 4-6 below:

*Table 4-6. flexiPCS CRC checker options*

| | Bit | Function |
|---|---|---|
| Quad Interface Register Offset Address 0x1E | [0:1] | Reserved |
| | 2 | Select/deselect inversion of data (0->1 and 1->0) to CRC checker<br><br>0 = do not invert data input to CRC checker<br>1 = invert data to CRC checker (bitwise inversion) |
| | 3 | Select/deselect bit order swap of 8-bit input data to CRC checker<br><br>0 = do not swap bit order of input data to CRC checker<br>1 = swap bit order (swap bit 7 and bit 0, swap bit 6 and bit 1, and so on) of input data to CRC checker |
| | 4 | Select/deselect inversion of output (0->1 and 1->0) from CRC checker as they are checked<br><br>0 = do not invert output from CRC checker<br>1 = invert output from CRC checker (bitwise inversion) |
| | 5 | Select/deselect bit order swap of 8-bit output from CRC checker as they are checked<br><br>0 = do not swap bit order of output data from CRC checker<br>1 = swap bit order (swap bit 7 and bit 0, swap bit 6 and bit 1, and so on) of output data from CRC checker |
| | 6 | Selection of received CRC byte order<br><br>0 = Received CRC byte order is [31:24] first, [23:16] second, [15:8] third, [7:0] fourth<br>1 = Received CRC byte order is [7:0] first, [15:8] second, [23:16] third, [31:24] fourth |
| | 7 | Selection of Start-of-Packet and End-of-Packet ordered set length<br><br>0 = Start-of-Packet and End-of-Packet ordered sets are both set to 2 characters long<br>1 = Start-of-Packet and End-of-Packet ordered sets are both set to 1 character long |

## Control & Status

The Control & Status pins for the Generic 8b10b mode can be optionally selected on a per quad basis when generating the flexiPCS quad interface files with the ispLEVER tools. Each of these control and status functions are also accessible through equivalent Quad Interface and Channel Interface Registers. The control and status signals allow direct real time access of these functions from FPGA logic.

**Control**

**felb_[0-3]** - Per channel, active high control signal which sets up the appropriate channel in Far-End Loopback mode. This mode is generally used for testing the flexiPCS logic, looping back receive data back to the transmit direction without crossing the FPGA logic interface. For more information on the details of Far-End Loopback mode, see the **flexiPCS Testing** Section of the flexiPCS Data Sheet. The Far-End Loopback mode can also be initiated by writing Channel Interface Register Offset Address 0x00, bit 5 to '1'.

**lsm_en_[0-3]** - Per channel Receive Link State Machine Enable. A change (either 0 to 1 or 1 to 0) on the lsm_en_[0-3] resets the Receive Link State Machine into No Sync mode. The Receive Link Machine Enable can also be set by writing Channel Interface Register Offset Address 0x00, bit 7 to '1'.

**tx_crc_init_[0-3]** - Per channel CRC initialization. When driven to '1', transmit CRC logic is re-initialized. When driven to '0', CRC is computed on incoming txd_[0-3](7:0) data.

**mca_align_en_[0-3] -** Per channel active high Multi-channel Align enable. Multi-channel Alignment Enable can also be set by writing the appropriate Quad Interface Register Offset Address 0x04, bits [4:7] to '1' (bit 4 for channel 3, bit 5 for channel 2, bit 4 for channel 1, and bit 7 for channel 0).

**mca_resync_[01/23]** - Active high, asynchronous reset to Multi-channel Aligner state machine and aligner FIFO control logic. **mca_resync_01** resets logic in channels 0 and 1 in two-channel alignment mode and all four channels in four channel alignment mode. **mca_resync_23** resets logic in channels 2 and 3 in two-channel alignment mode (not used in four channel alignment mode).

**Status**

**lsm_status_[0-3]** - Per channel active high signal from the Receive Link State Machine indicating link synchronization successful. The link synchronization status can also be read from the appropriate Quad Interface Register Offset Address 0x84, bits [4:7] (bit 4 for channel 0, bit 5 for channel 1, bit 4 for channel 2, and bit 7 for channel 3).

**rx_crc_eop_[0-3]** - Per channel active high signal which indicates an end-of-packet and/or a CRC error condition. When end of packet is reached, calculated CRC is compared with the expected values. If there is no error, rx_crc_eop_[0-3] acts as an end of packet indicator and goes high for one rclk_[0-3] cycle. If the packet was in error, then rx_crc_eop_[0-3] is asserted for two clock cycles as shown in Figure 4-17.

**mca_aligned_[01/23]** - Active high signal indicating successful alignment of channels 0/1 or channels 2/3. mca_aligned_01 high indicates successful alignment of channels 0 and 1 in two-channel alignment mode and all four channels in four channel alignment mode. mca_aligned_23 high indicates successful alignment of channels 2 and 3 in two-channel alignment mode (Always low in four channel alignment mode). The Multi-channel Alignment status for channels 0/1 (equivalent to the mca_aligned_01 output) can also be read from the Quad Interface Register Offset Address 0x82, bit 2. The Multi-channel Alignment status for channels 2/3 (equivalent to the mca_aligned_23 output) can also be read from the Quad Interface Register Offset Address 0x82, bit 3. *Note: mca_aligned_01 is only valid when performing multichannel alignment within a single quad.*

**mca_inskew_[01/23]** - Active high signal indicating alignment skew tolerance is met for channels 0/1 or 2/3. mca_inskew_01 high indicates that channels 0 and 1 in two-channel alignment mode and all four channels in four channel alignment mode are within specified tolerance for alignment. mca_inskew_23 high indicates that channels 2 and 3 in two-channel alignment mode are within specified tolerance for alignment (Always low in four channel alignment mode). The align in tolerance status for channels 2/3 (equivalent to the mca_inskew_23 output) can also be read from the Quad Interface Register Offset Address 0x82, bit 6. The align in tolerance status for channels 2/3 (equivalent to the mca_inskew_23 output) can also be read from the Quad Interface Register Offset Address 0x82, bit 7.

**mca_outskew_[01/23]** - Active high signal indicating alignment skew tolerance is not met for channels 0/1 or 2/3. mca_inskew_01 high indicates that the skew between channels 0 and 1 in two-channel alignment mode and all four channels in four channel alignment mode is beyond the specified tolerance for alignment and therefore alignment cannot be performed. mca_outskew_23 high indicates that the skew between channels 2 and 3 in two-channel alignment mode is beyond the specified tolerance for alignment (Always low in four channel alignment mode). The align beyond tolerance status for channels 0/1 (equivalent to the mca_outskew_01 output) can also be read from the Quad Interface Register Offset Address 0x82, bit 4. The outskew status for channels 2/3 (equivalent to the mca_outskew_23 output) can also be read from the Quad Interface Register Offset Address 0x82, bit 5.

## Status Interrupt Registers

Some of the status registers associated with Generic 8b10b operation have an associated status interrupt and status interrupt enable register. Any flexiPCS status interrupt register will generate an interrupt to the Systembus block (if used in the design). For a description of the operation of interrupts with the Systembus, refer to the Systembus section of the LatticeSC/M Family Data Sheet. Operation of the interrupt registers for the flexiPCS is as follows:

User must set the appropriate status interrupt enable bit to a ‘1’

Whenever the associated status bit goes high, its status interrupt bit will also go high. The status interrupt bit will remain high even if the associated status bit goes low.

The status interrupt bit will remain high until it is read, when it will automatically be cleared. If the associated status bit is still high, then the status interrupt bit will go high again (until read the next time).

Table 4-7 shows a listing of the status registers associated with Generic 8b10b operation which have a corresponding status interrupt and status interrupt enable bit.

***Table 4-7. Status Interrupt Register Bits***

| Status Register Bit Function | Status Register Bit Address | Status Interrupt Enable Register Bit Address | Status Interrupt Register Bit Address |
|---|---|---|---|
| Multi-channel Alignment State Machine Status (mca_aligned_01)<br><br>Channels 0 & 1 (2-channel alignment mode)<br><br>Channels 0,1,2 & 3 (4-channel alignment mode) | QIR 0x82, bit 2 | QIR 0x0C, bit 2 | QIR 0x83, bit 2 |
| Multi-channel Alignment State Machine Status (mca_aligned_23)<br><br>Channels 2 & 3 (2-channel alignment mode)<br><br>Inactive (4-channel alignment mode) | QIR 0x82, bit 3 | QIR 0x0C, bit 3 | QIR 0x83, bit 3 |
| Multi-channel Alignment Skew Tolerance Exceeded (mca_outskew_01)<br><br>Channels 0 & 1 (2-channel alignment mode)<br><br>Channels 0,1,2 & 3 (4-channel alignment mode) | QIR 0x82, bit 4 | QIR 0x0C, bit 4 | QIR 0x83, bit 4 |
| Multi-channel Alignment Skew Tolerance Exceeded (mca_outskew_23)<br><br>Channels 2 & 3 (2-channel alignment mode)<br><br>Inactive (4-channel alignment mode) | QIR 0x82, bit 5 | QIR 0x0C, bit 5 | QIR 0x83, bit 5 |
| Multi-channel Alignment Skew Tolerance Met (mca_inskew_01)<br><br>Channels 0 & 1 (2-channel alignment mode)<br><br>Channels 0,1,2 & 3 (4-channel alignment mode) | QIR 0x82, bit 6 | QIR 0x0C, bit 6 | QIR 0x83, bit 6 |
| Multi-channel Alignment Skew Tolerance Met (mca_inskew_23)<br><br>Channels 2 & 3 (2-channel alignment mode)<br><br>Inactive (4-channel alignment mode) | QIR 0x82, bit 7 | QIR 0x0C, bit 7 | QIR 0x83, bit 7 |

*Table 4-7. Status Interrupt Register Bits (Continued)*

| | | | |
|---|---|---|---|
| Receive Link State Machine Status (lsm_status) Channel 0 | QIR 0x84, bit 4 | QIR 0x1C, bit 4 | QIR 0x85, bit 4 |
| Receive Link State Machine Status (lsm_status) Channel 1 | QIR 0x84, bit 5 | QIR 0x1C, bit 5 | QIR 0x85, bit 5 |
| Receive Link State Machine Status (lsm_status) Channel 2 | QIR 0x84, bit 6 | QIR 0x1C, bit 6 | QIR 0x85, bit 6 |
| Receive Link State Machine Status (lsm_status) Channel 3 | QIR 0x84, bit 7 | QIR 0x1C, bit 7 | QIR 0x85, bit 7 |

# Gigabit Ethernet Mode
# LatticeSC/M flexiPCS Family Data Sheet

June 2011 Data Sheet DS1005

## Introduction

The Gigabit Ethernet mode of the flexiPCS (Physical Coding Sublayer) block supports full compatibility, from the Serial I/O to the GMII interface of the IEEE 802.3-2002 1000 Base-X Gigabit Ethernet standard.

The LatticeSC flexiPCS in Gigabit Ethernet mode supports the following operations:

**Transmit Path (From LatticeSC device to line):**

- Cyclic Redundancy Check (CRC) generation and insertion into Gigabit Ethernet frame
- Transmit State Machine which performs 8-bit data encapsulation and formatting, including the Auto-Negotiation code word insertion and outputting the correct 8-bit code/data word and k control characters according to the IEEE 802.3-2002 1000 Base-X specification
- 8b10b Encoding

**Receive Path (From line to LatticeSC device):**

- Word Alignment based on IEEE 802.3-2002 1000 Base-X defined alignment characters.
- 8b10b Decoding
- Link State Machine functions compliant with the IEEE 802.3-2002 1000 Base-X specification.
- Clock Tolerance Compensation logic capable of accommodating clock domain differences.
- Receive State Machine including Auto-Negotiation support compliant to the IEEE 802.3-2002 1000 Base-X specification.
- Cyclic Redundancy Code (CRC) checking

### LatticeSC flexiPCS Quad Module

Devices in the LatticeSC family have up to 8 quads of embedded flexiPCS logic. Each quad in turn supports 4 independent full-duplex data channels. A single channel can support a fully compliant Gigabit Ethernet data link and each quad can support up to four such channels. Note that mode selection is done on a per quad basis. Therefore, a selection of Gigabit Ethernet mode for a quad dedicates all four channels in that quad to Gigabit Ethernet mode.

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

Operation of the SERDES requires the user to provide a reference clock (or clocks) to each active quad to provide a synchronization reference for the SERDES PLLs. Reference clocks are shared among all four channels of a quad. If the transmit and receive frequencies are within the specified ppm tolerance for the LatticeSC SERDES (See DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet), only one reference clock for both transmit and receive is needed for both transmit and receive directions. If the receive frequency is not within the specified ppm tolerance for the LatticeSC SERDES (See the DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet) of the transmit frequency, then separate reference clocks for the transmit and receive directions must be provided.

Simultaneous support of different (non-synchronous) high-speed data rates is possible with the use of multiple quads. Multiple frequencies that are exact integer multiples can be supported on a per channel basis through the use of the full-rate/half-rate mode options (see the **SERDES Functionality section** for more details).

© 2011 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

## Quad and Channel Option Control

Although the mode selection and reference frequency covers an entire quad, many options covering clocking and data formatting are available on a per channel basis. These options are detailed in the following Gigabit Ethernet Mode description. Some of these options can be controlled directly through the FPGA interface pins made available on the Gigabit Ethernet Quad Module.

Other options are available only through dedicated registers that can be written and read via the embedded System Bus Interface (see the System Bus section for more details on the operation of the System Bus), or during device configuration. Depending on the specific option, a register may affect operation of multiple quads, a single quad or an individual channel in a quad. Registers dedicated to multiple quads are listed in the Inter-quad Interface Register Map in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet. Registers dedicated to one quad are listed in the Quad Interface Register Map. Registers dedicated to a single channel are listed in the Channel Interface Register Map.

### Register Map Addressing

Figure 5-1 provides a map of base register addresses for any of the flexiPCS quads on a LatticeSC device. Note that different devices may have different numbers of available quads up to a total of 8 quads (or 32 channels) maximum.

Inter-quad Interface, Quad Interface, and Channel Interface Registers are listed by Offset Address. Each Inter-quad Interface Register, Quad Interface Register, and Channel Interface Register has a unique 18-bit address determined by the specific quad or channel targeted. The addressing of Inter-quad Interface Registers, Quad Interface Registers, and Channel Interface Registers is shown Figure 5-1.

Base addresses are dependant on the physical location of a given quad or channel on a LatticeSC device. Each quad and channel has an associated set of control and status registers. These registers are referred throughout this data sheet by their offset addresses. The unique 18-bit address of a control or status register for a specific quad or channel is simply the offset address of that register added to the base address for that specific quad or channel as shown in Figure 5-1.

Channel Interface Registers can be written in one of three methods. One method is to write to one specific register corresponding to one specific channel. The other two methods involve writing to multiple channel registers with just one write operation. This is referred to as a Broadcast write. A Broadcast write is useful when multiple channel registers are to be written with the same value. The first method of Broadcast writing involves writing to all four channel registers in a quad at one time. A second method of Broadcast writing involves writing to all channel registers for all quads on the left or right side of the device at one time. Therefore, a specific Channel Interface Register may be accessed through any one of three channel addresses, depending on the number of channel registers being written to at any one time.

A broadcast write to multiple quads cannot be used to power on SERDES in any device-package combination that does not have all SERDES quads bonded because of excess power on the board and jitter is also affected.

Throughout this document, the byte-wide data at each offset address is listed with a hexadecimal value with bit 0 as the MSB and bit 7 as the LSB as per the PowerPC convention. For example if the value for Quad Interface Register Offset Address 0x02 is stated to be 0x15, the bit pattern defined is as shown in Table 5-1:

***Table 5-1. Control/Status Register Bit Order Example***

| **Quad Interface Register Offset Address 0x02 = 0x15** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Data Bit 0 | Data Bit1 | Data Bit 2 | Data Bit 3 | Data Bit 4 | Data Bit 5 | Data Bit 6 | Data Bit 7 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| **1** | | | | **5** | | | |

A full list of register Offset Addresses is provided in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet.

***Figure 5-1. flexiPCS Inter-quad, Quad, and Channel Interface Register Map Base Addressing***

## Auto-Configuration

Initial register setup for each flexiPCS mode can be performed without accessing the system bus by using the auto-configuration tool, IPexpress, in ispLEVER. tool, IPexpress generates 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 system bus is therefore not needed if all quads are to be set via auto-configuration files. However, the system bus must be included in a design if the user needs to change control registers or monitor status registers during operation. Note that a quad reset (Quad Interface Register 0x43, bit 7 or the **quad_rst** port at the FPGA interface) will reset all registers back to their default (not auto-configuration) values. If a quad reset is issued, the flexiPCS registers need to be rewritten via the Systembus.

## Gigabit Ethernet Register Settings

The auto-configuration file sets registers that perform the following operations. If the auto-configuration file is not used, the appropriate registers need to be programmed via the Systembus. For more options, refer to the flexiPCS register map in the **Memory Map** section of the **LatticeSC/M Family flexiPCS Data Sheet**.

**A flexiPCS quad can be set to Gigabit Ethernet mode by writing Quad Interface Register Offset Address 0x18 bits[0:7] to 0x00.**

This register value automatically sets up the following options in the flexiPCS for the given quad. Details on the functionality of each chosen option is provided in the **Gigabit Ethernet Mode Detailed Description** section.

**Transmit Path**

- Transmit State Machine is set to Gigabit Ethernet Mode
- 8b10b encoder is enabled

**Receive Path**

- Word aligner is enabled
- 8b10b decoder is enabled
- Receive Link State Machine is set to Gigabit Ethernet Mode
- Receive State Machine is set to Gigabit Ethernet Mode

Other register settings are required to operate a channel or channels in Gigabit Ethernet Mode. The following is a list of options that must be set for Gigabit Ethernet operation. More detail for each option is included in the **Gigabit Ethernet Mode Detailed Description** section.

- Powerup of appropriate quad and channel. Quad powerup is chosen by writing the appropriate Quad Interface Register Offset Address 0x28, bit 1 to ‘1’. Channel powerup is chosen by writing the appropriate Channel Interface Register Offset Address 0x13, bit 6 (receive direction) and/or bit 7 (transmit direction) to ‘1’. By default, all channels and quads are powered down.
- Receive Link State Machine enable. By default, all channel Receive Link State Machines are disabled. The Receive Link State Machine for a given channel can be enabled by writing the appropriate Channel Interface Register Offset Address 0x00, bit 7 to ‘1’.
- Setting SERDES reset low at end of flexiPCS setup. By default, the SERDES reset is set to ‘1’ (SERDES are reset). The SERDES reset can be set low by writing Quad Interface Register Offset Address 0x41, bit 5 to a ‘0’. For more information on proper flexiPCS powerup and reset sequencing, refer to the **flexiPCS Reset Sequences** and **flexiPCS Power Down Control Signals** descriptions in the LatticeSC/M Family flexiPCS Data Sheet **SERDES Functionality** section.
- Word alignment character(s), such as a comma

- Clock Tolerance Compensation insertion/deletion ordered set values and FIFO settings
- Auto-Negotiation enable and register definitions
- Cyclic Redundancy Code (CRC) generator/checker enable and settings

## Gigabit Ethernet Auto-Configuration

An example of an auto-configuration file for a quad set to Gigabit Ethernet Mode with no Auto-Negotiation enabled, half-rate mode, and with only channel 1 enabled is shown in Figure 5-2. Format for the auto-configuration file is

register type (quad=quad register, ch1=channel 1 register), offset address in hex, register value in hex, # comment

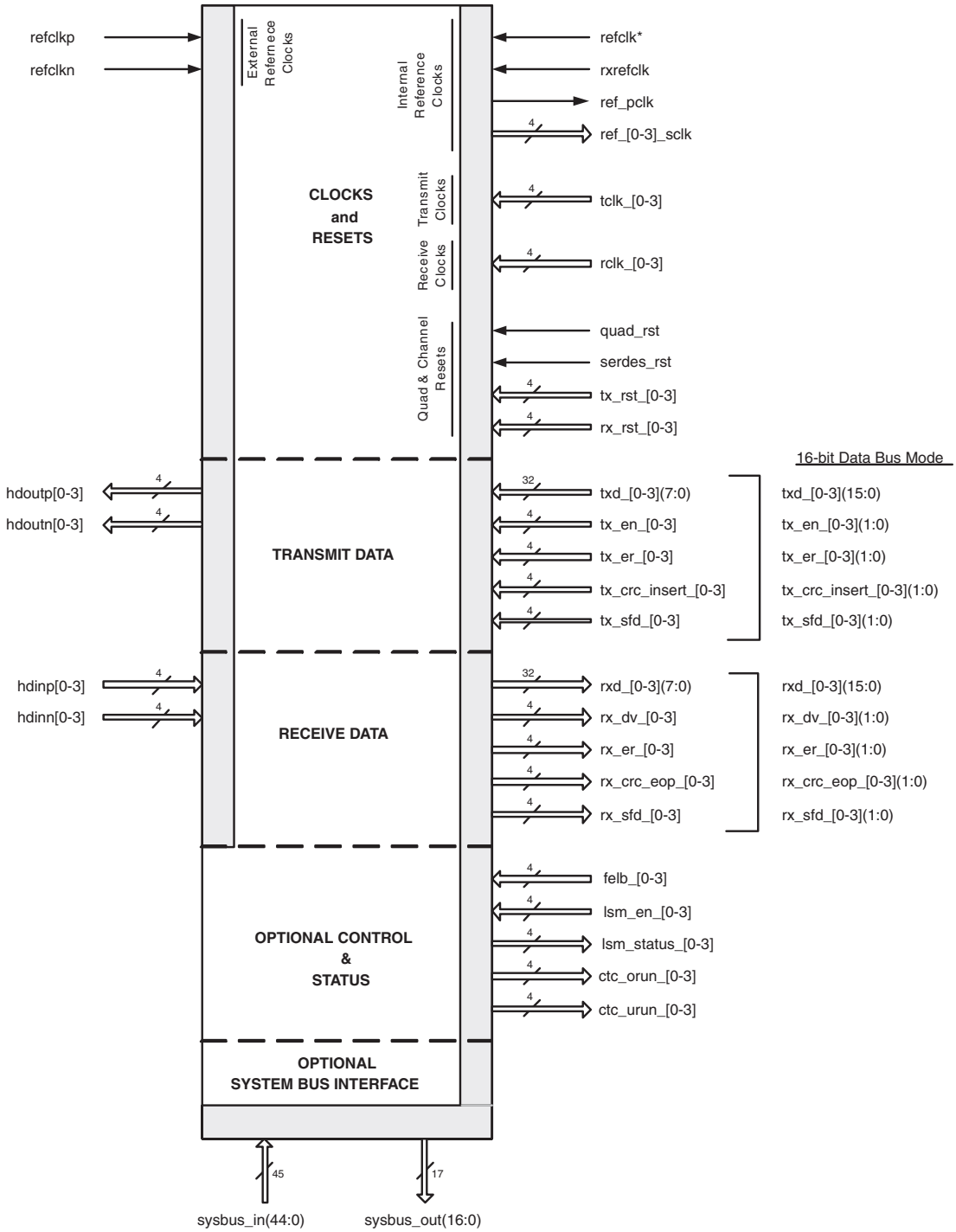
***Figure 5-2. Gigabit Ethernet Auto-Configuration Example File***

```
quad 18 00  #  Set quad to Gigabit Ethernet Mode
quad 29 01  #  Set reference clock select
quad 28 40  #  Set bit clock multiplier to 10X (for half rate mode), quad powerup set to '1'
quad 02 15  #  Set ref_pclk to channel 1 clock
quad 19 00  #  Set FPGA interface data bus width to 8-bit
quad 14 7F  #  Set word alignment mask to compare lowest 7 LSBs
quad 15 03  #  Set positive disparity comma value
quad 16 7C  #  Set negative disparity comma value
quad 0D 97  #  Set CTC FIFO watermarks (high=9, low=7)
quad 0E 0B  #  Set insertion/deletion to 2 bytes for CTC, set maximum inter-packet gap
quad 11 BC  #  1st byte of /I2/ pattern for CTC (K28.5)
quad 12 50  #  2nd byte of /I2/ pattern for CTC (D16.2)
quad 13 04  #  K bits for /I2/ pattern
quad 1D 57  #  Set CRC generator options
quad 1E 1E  #  Set CRC checker options
ch1 00 09    #  Turn on channel 1 link state machine, rx_sfd ports enabled
ch1 13 0F    #  Powerup channel 1 transmit and receive directions, set rate mode to “half”
ch1 14 90    # 16% pre-emphasis on SERDES output buffer
ch1 15 10    # +6 dB equalization on SERDES input buffer
# These lines must appear last in the autoconfig file. These lines apply the
# correct reset sequence to the PCS block upon bitstream configuration
quad 41 00 # de-assert serdes_rst
quad 40 ff   # assert datapath reset for all channels
quad 40 00 # de-assert datapath reset for all channels
```

# Gigabit Ethernet Mode

IPexpress allows the designer to choose the mode for each flexiPCS quad used in a given design. Figure 5-3 displays the resulting port interface to one flexiPCS quad that has been set to Gigabit Ethernet mode on all four channels.

***Figure 5-3. Gigabit Ethernet Mode Pin Diagram***




* The refclk inputs for all active quads on a device must be connected to the same clock . The rxrefclk inputs for all active quads on a device do not have to be connected to the same clock.

## Gigabit Ethernet Mode Pin Description

Table 5-2 lists all inputs and outputs to/from a flexiPCS quad in Gigabit Ethernet mode. A brief description for each port is given in the table. A more detailed description of the function of each port is given in the **Gigabit Ethernet Mode Detailed Description**.

***Table 5-2. Gigabit Ethernet Mode Pin Description***

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Reference Clocks** | | | |
| refclkp | In from I/O pad | N/A | Reference clock input, positive. Dedicated CML input. |
| refclkn | In from I/O pad | N/A | Reference clock input, negative. Dedicated CML input |
| refclk | In from FPGA | N/A | Optional reference clock input from FPGA logic. Can be used instead of I/O pin reference clock.<br><br>The refclk inputs for all active quads on a device must be connected to the same clock. |
| rxrefclk | In from FPGA | N/A | Optional receive reference clock input from FPGA logic. Can be used instead of I/O pin reference clock. The rxrefclk inputs for all active quads do not have to be connected to the same clock. |
| ref_pclk | Out to FPGA | N/A | Locked reference clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| ref_[0-3]_sclk | Out to FPGA | N/A | Per channel locked reference clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| **Transmit Clocks** | | | |
| tclk_[0-3] | In from FPGA | N/A | Per channel transmit clock inputs from FPGA. Used to clock the TX data phase compensation FIFO with clock synchronous to the reference clock. May also be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Receive Clocks** | | | |
| rclk_[0-3] | In from FPGA | N/A | Per channel receive clock inputs from FPGA. May be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference and/or receive reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Resets** | | | |
| quad_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of SERDES and PCS logic. |
| serdes_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of the SERDES. |
| tx_rst_[0-3] | In from FPGA | ASYNC | Per channel active high, asynchronous reset of individual transmit channel of PCS logic. |
| rx_rst_[0-3] | In from FPGA | ASYNC | Per channel active high, asynchronous reset of individual receive channel of PCS logic. |
| **Transmit Data** | | | |
| hdoutp[0-3] | Out to I/O pad | N/A | High-speed CML serial output, positive. |
| hdoutn[0-3] | Out to I/O pad | N/A | High-speed CML serial output, negative. |
| txd_[0-3](7:0)<br><br>txd_[0-3](15:0)* | In from FPGA | tclk_[0-3] | Per channel parallel transmit data bus from the FPGA. Bus is 8-bits wide if QIR 0x19, bit 4 = ‘0’.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 4 = ‘1’. |

*Table 5-2. Gigabit Ethernet Mode Pin Description*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| tx_en_[0-3]<br><br>tx_en_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel active high indicates data is present on the GMII for transmission.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_er_[0-3]<br><br>tx_er_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel active high during the data phase of the Ethernet frame, instructs PCS to insert invalid data.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_crc_insert_[0-3]<br><br>tx_crc_insert_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel active high signal forces CRC into the data bus.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_sfd_[0-3]<br><br>tx_sfd_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel Start of Frame Delimiter indicator. Used for CRC initialization.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| **Receive Data** | | | |
| hdinp[0-3] | In from I/O pad | N/A | High-speed CML serial input, positive. |
| hdinn[0-3] | In from I/O pad | N/A | High-speed CML serial input, negative. |
| rxd_[0-3](7:0)<br><br>rxd_[0-3](15:0)* | Out to FPGA | rclk_[0-3] | Per channel parallel receive data bus to the FPGA. Bus is 8-bits wide if QIR 0x19, bit 5 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 5 = '1'. |
| rx_dv_[0-3]<br><br>rx_dv_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high signal indicates presence of recovered and decoded data on the rxd_[0-3] bus.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_er_[0-3]<br><br>rx_er_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high signal driven by the PCS which indicates invalid data. Transitions in sync with rclk_[0-3].<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_crc_eop_[0-3]<br><br>rx_crc_eop_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high indicates that an end-of-packet and/or CRC error was detected.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_sfd_[0-3]<br><br>rx_sfd_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high Start-of-Frame Delimiter detection if CIR 0x00, bit 4 = '1'. Indicates 8b10b code violation detection otherwise.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| **Optional Control & Status** | | | |
| felb_[0-3] | In from FPGA | ASYNC | Per channel active high far-end loopback enables. |
| lsm_en_[0-3] | In from FPGA | ASYNC | Per channel receive link state machine enable. |
| lsm_status_[0-3] | Out to FPGA | ASYNC | Per channel signal from receive link state machine indicating successful link synchronization. |
| ctc_urun_[0-3] | Out to FPGA | ASYNC | Per channel active high flag indicator that clock tolerance compensation FIFO has underrun. |
| ctc_orun_[0-3] | Out to FPGA | ASYNC | Per channel active high flag indicator that clock tolerance compensation FIFO has overrun. |
| **Optional System Bus Interface** | | | |

*Table 5-2. Gigabit Ethernet Mode Pin Description*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| sysbus_in(44:0) | In from FPGA | N/A | Control and data signals from the internal system bus. |
| sysbus_out(16:0) | Out to FPGA | N/A | Control and data signals to the internal system bus. |

# Gigabit Ethernet Mode Detailed Description

The following section provides a detailed description of the operation of a flexiPCS quad set to Gigabit Ethernet mode. The functional description is organized according to the port listing shown in Figure 5-3. The System Bus Offset Address for any control and status registers relevant to a given function are provided with the description of the operation of that function.

## Clocks & Resets

A detailed description of all SERDES clock, data, and reset ports and recommended reset sequencing is provided in the **SERDES Functionality** section of the flexiPCS Data Sheet. The following is a recommended setup of the SERDES for Gigabit Ethernet applications.

For Gigabit Ethernet applications, the bit clock rate is 1.25 Gbps. This falls into the range defined as “half rate” mode for the SERDES PLLs. Half rate mode is selected separately for each channel and each direction by writing the appropriate Channel Interface Register Offset Address 0x13, bit 5 (transmit) and bit 4 (receive) to a ‘1’.

The reference clock frequency is determined by the reference clock mode chosen. Table 5-3 shows the possible reference clock frequencies for various reference clock modes which result in a 1.25 Gbps internal bit rate.

*Table 5-3. Gigabit Ethernet Reference Clock Frequency Options*

| **Half Rate Mode**<br><br>**Channel Interface Register Offset Address 0x13**<br><br>**Transmit - Bit 5 = ‘1’**<br>**Receive - Bit 4 = ‘1’** | | | | |
|---|---|---|---|---|
| **Reference Clock Mode**<br><br>**Quad Interface Register Offset Address = 0x28, Bits [2:3]** | **Reference Clock Frequency** | **Internal Serial (bit) Clock Frequency** | **FPGA Interface Clock Frequency**<br><br>**Quad Interface Register Offset Address 0x19** | |
| | | | **8-Bit Data Bus Mode**<br><br>**Transmit - Bit 4 = ‘0’**<br>**Receive - Bit 5 = ‘0’** | **16-Bit Data Bus Mode**<br><br>**Transmit - Bit 4 = ‘1’**<br>**Receive - Bit 5 = ‘1’** |
| **00** | **125 MHz** | **1.25 GHz** | **125 MHz** | **62.5 MHz** |
| **01** | **250 MHz** | **1.25 GHz** | **125 MHz** | **62.5 MHz** |
| **10** | **500 MHz** | **1.25 GHz** | **125 MHz** | **62.5 MHz** |

*Note: There are also frequency dependent SERDES performance control bits that are not writable via the System-bus. These control bits are set in the auto-configuration file generated within IPexpress and passed into the bit-stream through the normal FPGA design flow (map, par, bitgen). To change the bit rate for a SERDES, specify the proper values for the affected flexiPCS quad in IPexpress and regenerate the auto-configuration file. Failure to do this may result in non-optimal SERDES operation.*

Additional information on all reference clock rate modes can be found in the **SERDES Functionality** section of the flexiPCS Data Sheet.

**Quad & Channel Resets**

Resets are provided to reset an entire quad (either SERDES logic only or all flexiPCS logic) or each individual channel (all flexiPCS logic per channel). These resets are driven from the FPGA logic or by writing the appropriate Quad or Channel Interface Register. A summary of the control signals provided for reset are listed below. More detail on recommended initialization and reset sequences for the SERDES is provided in the **SERDES Functionality** section of the **LatticeSC/M Family flexiPCS Data Sheet**.

The following inputs to the flexiPCS from the FPGA are enabled as long as Quad Interface Register Offset Address 0x42, bit 7 is set to '1' (which is the default on powerup). Writing a '0' to this bit will disable these reset inputs.

**quad_rst** - Active high, asynchronous signal from FPGA resets all SERDES and PCS logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x43, bit 7 to '1'. quad_rst also resets all flexiPCS control registers. If these registers had been previously written via the Systembus or set during configuration through an auto-configuration file, they will need to be rewritten following this reset.

**serdes_rst** - Active high, asynchronous signal from FPGA resets all SERDES (but not PCS) logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x41, bit 5 to '1'.Note that this bit is preset to '1' on powerup and must be written to '0' before operation of the SERDES can commence.

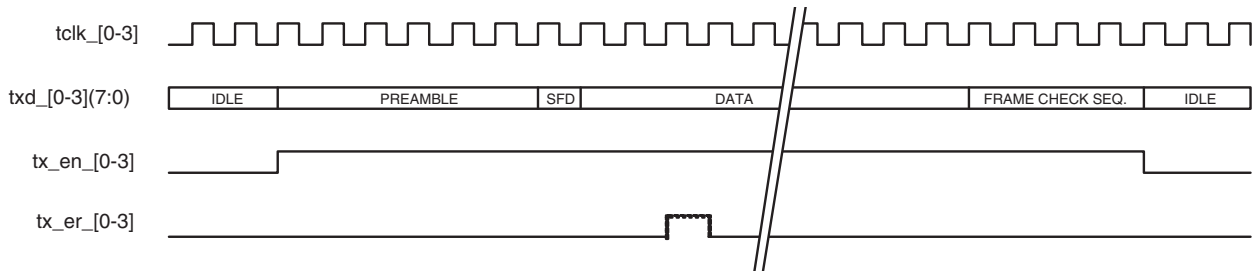
**tx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one transmit channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [4:7]. Bit 7 resets transmit channel 0, bit 6 resets transmit channel 1, bit 5 resets transmit channel 2, and bit 4 resets transmit channel 3.

**rx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one receive channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [0:3]. Bit 3 resets receive channel 0, bit 2 resets receive channel 1, bit 1 resets receive channel 2, and bit 0 resets receive channel 3.

## Transmit Data

When configured into Gigabit Ethernet mode, a flexiPCS quad provides four transmit data paths from a GMII compliant 8-bit parallel interface at the FPGA logic interface to a high-speed serial line interface at the device outputs. Figure 5-4 shows the typical operation of the GMII interface.

***Figure 5-4. Gigabit Ethernet Transmit GMII Interface Signals***




**txd_[0-3](7:0)** - Per channel transmit data from the FPGA GMII interface. Each quad supports up to 4 independent channels of 8-bit wide parallel data by default. Each txd_[0-3][7:0] is an eight bit data bus that is driven synchronously with respect to the corresponding tclk_[0-3]. For each tclk_[0-3] period in which tx_en_[0-3] is asserted and tx_er_[0-3] is de-asserted, data is presented on txd_[0-3] to the flexiPCS for transmission. While tx_en_[0-3] and tx_er_[0-3] are both de-asserted, txd_[0-3] shall have no effect upon the flexiPCS.

During the IDLE phase, the flexiPCS automatically generates IDLE patterns. The user does not need to provide IDLE characters and can typically set the txd_[0-3] to 0x00.

In 16-Bit Data Bus Mode, this bus is 16-bits wide (**txd_[0-3](15:0)**).

**tx_en_[0-3]** - Per channel, active high signal indicating data is present on the GMII for transmission. It is asserted synchronously with the first octet of the preamble and remains asserted while all octets to be transmitted are present on the GMII. tx_en_[0-3] shall be negated prior to the first rising edge of tclk_[0-3] following the final data octet of a frame.

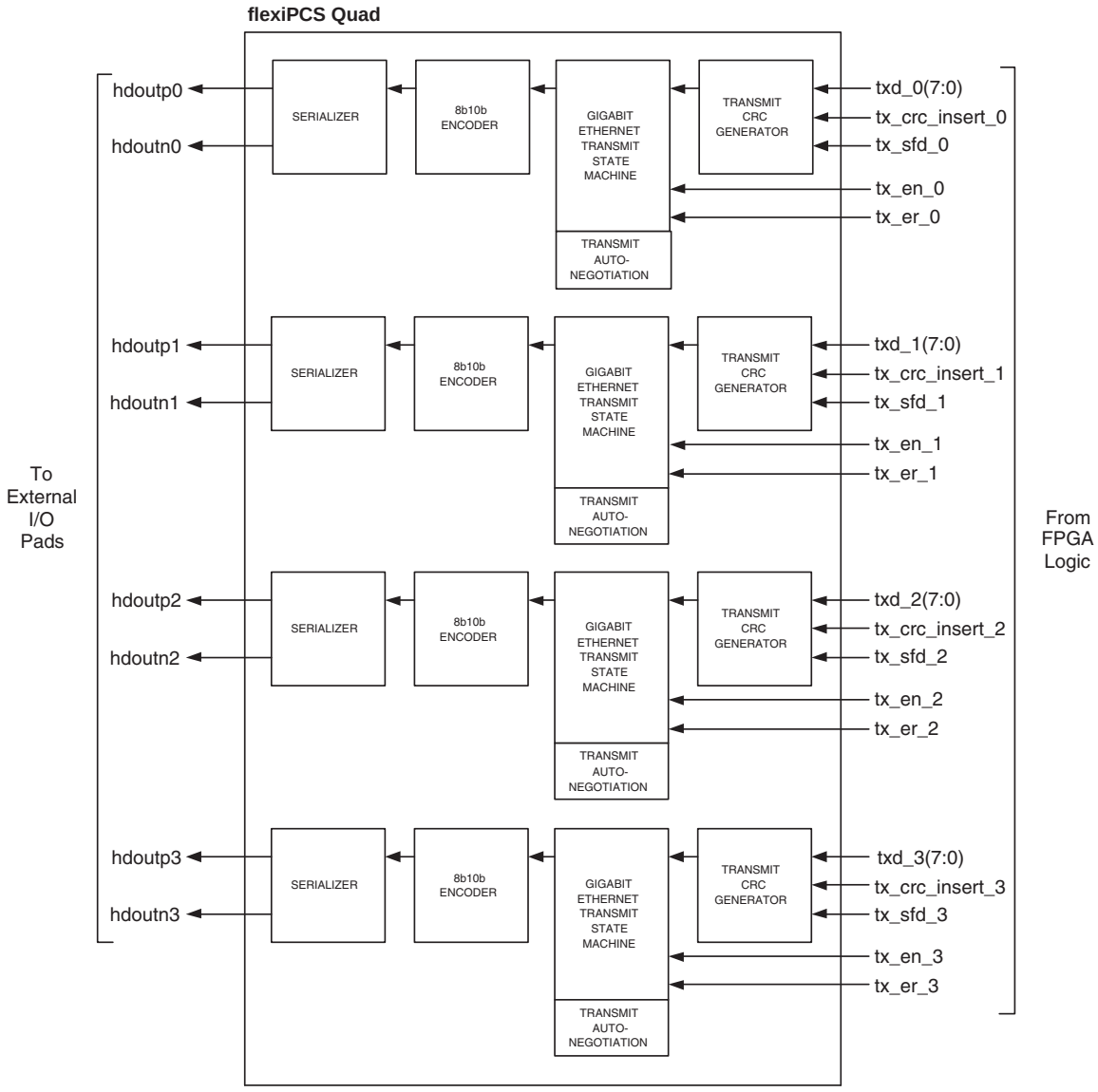
In 16-Bit Data Bus Mode, tx_en is 2 bits wide (**tx_en_[0-3](1:0)**) with tx_en_[0-3](1) associated with txd_[0-3](15:8) and tx_en_[0-3](0) associated with txd_[0-3](7:0).

**tx_er_[0-3]** - Per channel, active high signal which instructs the flexiPCS to insert invalid data during the data phase of the Ethernet frame. When tx_er_[0-3] is asserted for one or more tclk_[0-3] periods while tx_en_[0-3] is also asserted, the flexiPCS will emit one or more code-groups that are not part of the valid data or delimiter set somewhere in the frame being transmitted.

In 16-Bit Data Bus Mode, tx_er is 2 bits wide (**tx_er_[0-3](1:0)**) with tx_er_[0-3](1) associated with txd_[0-3](15:8) and tx_er_[0-3](0) associated with txd_[0-3](7:0).

The flexiPCS quad in Gigabit Ethernet mode transmit data path consists of the following sub-blocks per channel: Transmit CRC Generator, Gigabit Ethernet Transmit State Machine, 8b10b Encoder, and Serializer. Figure 5-5 shows the four channels of transmit data paths in a flexiPCS quad.

*Figure 5-5. Gigabit Ethernet Transmit Path (1 Quad)*




**CRC Generation**

A separate Cyclic Redundancy Code (CRC) generator is provided for all four transmit channels in a flexiPCS quad. The CRC generator supports the following Gigabit Ethernet polynomial:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

The CRC generator options are set on a per quad basis. The CRC generator options can be set for Gigabit Ethernet compliant operation by writing Quad Interface Register Offset Address 0x1D to 0x57 (writing 0x1D to 0x00 disables the CRC generator/checker).

The following signals driven by the FPGA logic are used to control the transmit CRC logic on a per channel basis:

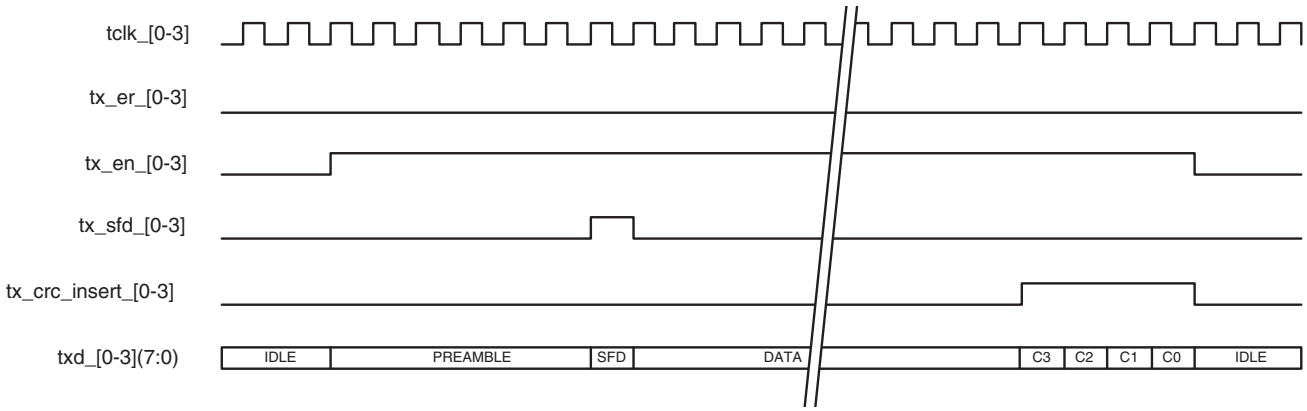
**tx_sfd_[0-3]** - Per channel Start-of Frame Delimiter indicator which is used for CRC initialization. When driven to '1', transmit CRC logic is re-initialized. When driven to '0', CRC is computed on incoming txd_[0-3](7:0) data.

In 16-Bit Data Bus Mode, tx_sfd is 2 bits wide (**tx_sfd_[0-3](1:0)**) with tx_sfd_[0-3](1) associated with txd_[0-3](15:8) and tx_sfd_[0-3](0) associated with txd_[0-3](7:0).

**tx_crc_insert_[0-3]** - Per channel active high signal which forces CRC onto the data bus. The values of 0xC3, 0xC2, 0xC1, and 0xC0 (in that order) should be forced onto the txd bus where the CRC values are to be inserted. When the appropriate tx_crc_insert_[0-3] is driven high, bits 0-7 of the CRC are substituted for 0xC0, bits 8-15 of the CRC are substituted for 0xC1, bits 16-23 of the CRC are substituted for 0xC2, and bits 24-31 of the CRC are substituted for 0xC3. This is shown in the Figure 5-6.

In 16-Bit Data Bus Mode, tx_crc_insert is 2 bits wide (**tx_crc_insert_[0-3](1:0)**) with tx_k_[0-3](1) associated with txd_[0-3](15:8) and tx_crc_insert_[0-3](0) associated with txd_[0-3](7:0).

***Figure 5-6. Gigabit Ethernet Transmit CRC Insertion***




**Transmit State Machine**
The transmit state machine performs 8-bit data encapsulation and formatting, including the Auto-Negotiation code word insertion and outputting the correct 8-bit code/data word and k control characters to the 8b10b encoder. The Transmit State Machine implements both the PCS transmit ordered_set state diagram and the PCS transmit code-group state diagram, found in Clause 36 (Figures 36-5 and 36-7) of the 802.3-2002 1000BASE-X specification.

**Auto-Negotiation**
The LatticeSC flexiPCS implements the Auto-Negotiation state machine shown in the Auto-Negotiation state diagram of the IEEE Standard 802.3-2002 specification. Auto-Negotiation can be enabled by writing the appropriate Channel Interface Register Offset Address 0x05, bit 5 to a '1'. Detailed information on the Auto-Negotiation function in the flexiPCS is provided in the Auto-Negotiation section.

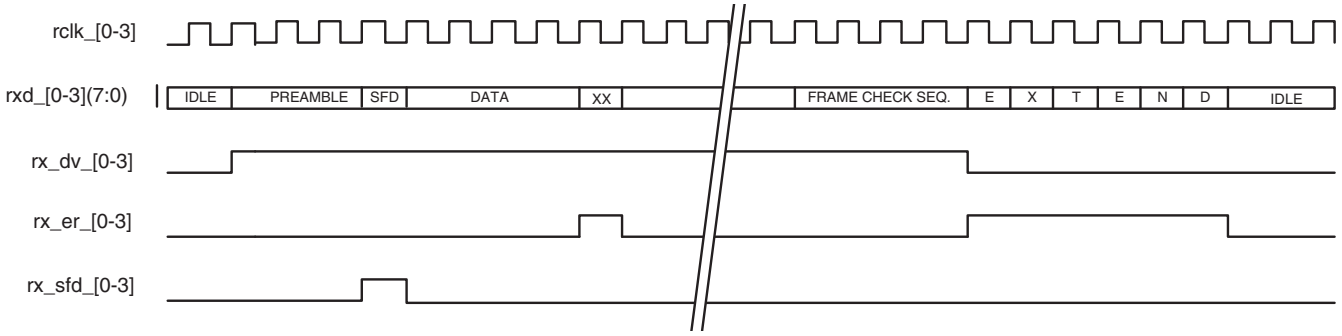
**8b10b Encoding**
This module implements an 8b10b encoder as described within the 802.3-2002 1000BASE-X specification. The encoder performs the 8-bit to 10-bit code conversion as described the specification, along with maintaining the running disparity rules as specified.

**Serializer**
The 8b10b encoded data undergoes parallel to serial conversion and is transmitted off chip via the embedded SERDES. For detailed information on the operation of the LatticeSC SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

## Receive Data

When configured into Gigabit Ethernet mode, a flexiPCS quad provides four receive data paths from a high-speed serial line interface at the device inputs to a GMII compliant 8-bit parallel interface at the FPGA logic interface as shown in Figure 5-7.

***Figure 5-7. Gigabit Ethernet Receive GMII Interface Signals***




**rxd_[0-3](7:0)** - Per channel receive data to the FPGA GMII interface. Each quad supports up to 4 independent channels of 8-bit wide parallel data by default. The rxd_[0-3] signal transitions synchronously with respect to rclk_[0-3]. For each rclk_[0-3] period in which rx_dv_[0-3] is asserted, rxd_[0-3] transfers eight bits of recovered data from the flexiPCS. While rx_dv_[0-3] is de-asserted, the flexiPCS may provide a false carrier indication by asserting the rx_er_[0-3] signal and driving the specific value onto rxd_[0-3]. A carrier extend error is indicated by another specific value of rxd_[0-3].

In 16-Bit Data Bus Mode, this bus is 16-bits wide (**rxd_[0-3](15:0)**).

**rx_dv_[0-3]** - Per channel, active high signal which indicates the presence of recovered and decoded data on the rxd_[0-3] bus. The rx_dv_[0-3] signal transitions synchronously with respect to the rclk_[0-3]. The rx_dv_[0-3] signal is asserted continuously from the first recovered octet of the frame through the final recovered octet and is low prior to the first rising edge of rclk_[0-3] that follows the final octet.

In 16-Bit Data Bus Mode, rx_dv is 2 bits wide (**rx_dv_[0-3](1:0)**) with rx_er_[0-3](1) associated with rxd_[0-3](15:8) and rx_dv_[0-3](0) associated with rxd_[0-3](7:0).

**rx_er_[0-3]** - Per channel, active high signal driven by the flexiPCS which transitions synchronously with respect to rclk_[0-3]. When rx_dv_[0-3] is asserted, rx_er_[0-3] is asserted for one or more rclk_[0-3] periods to indicate that an error was detected in the frame presently being transferred from the flexiPCS. Operation of rx_er_[0-3] follows the 802.3-2002 1000BASE-X specification.

In 16-Bit Data Bus Mode, rx_er is 2 bits wide (**rx_er_[0-3](1:0)**) with rx_er_[0-3](1) associated with rxd_[0-3](15:8) and rx_er_[0-3](0) associated with rxd_[0-3](7:0).

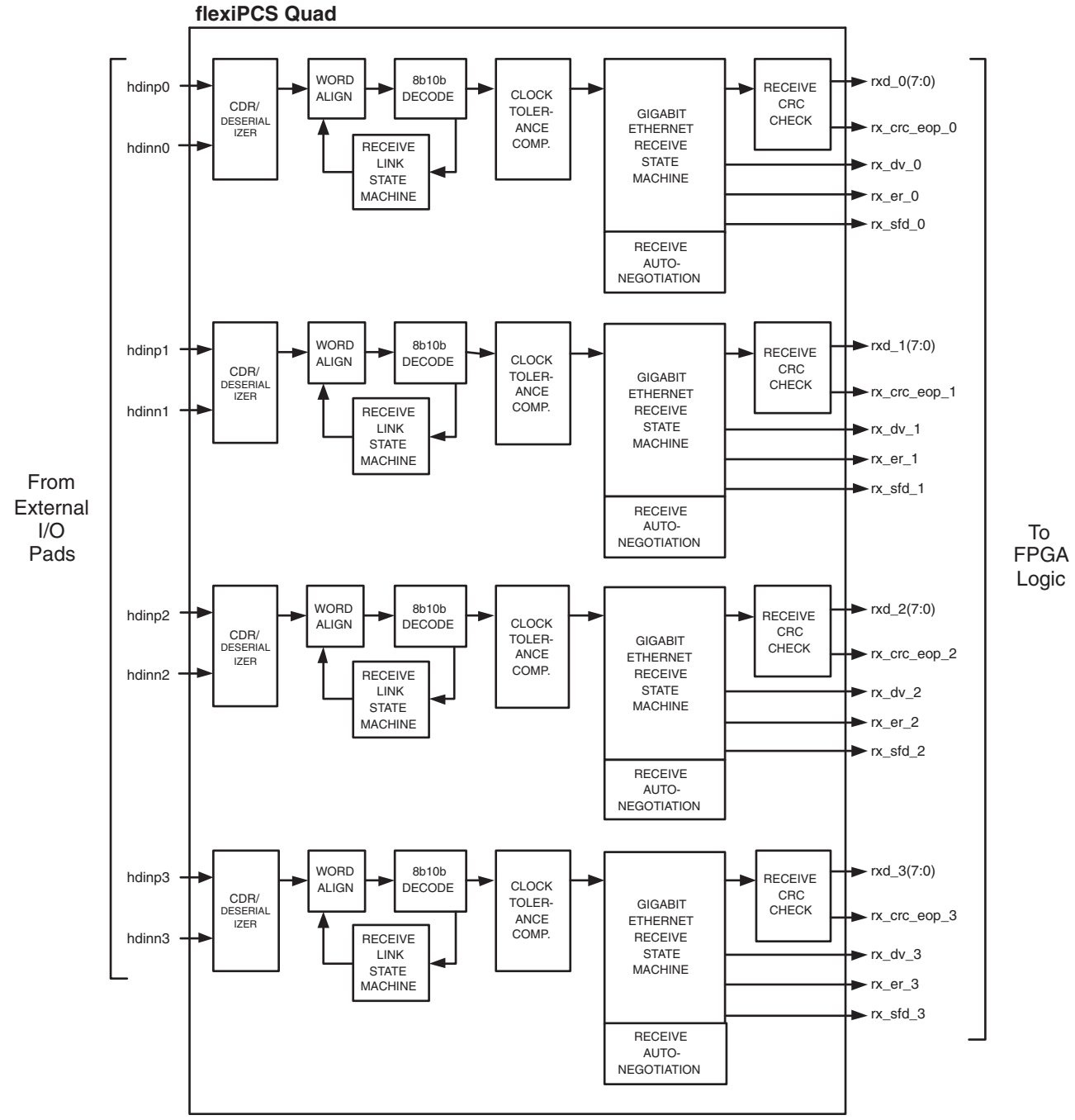
**rx_sfd_[0-3]** - Per channel, active high signal driven by the flexiPCS which transitions indicates the detection of the Start-of-Frame Delimiter byte for that channel. This output will report detection of Start-of-Frame when the appropriate Channel Interface Register Offset Address 0x00, bit 4 is set to '1'. This port will report 8b10b code violations otherwise.

In 16-Bit Data Bus Mode, rx_sfd is 2 bits wide (**rx_sfd_[0-3](1:0)**) with rx_sfd_[0-3](1) associated with rxd_[0-3](15:8) and rx_sfd_[0-3](0) associated with rxd_[0-3](7:0).

The flexiPCS quad in Gigabit Ethernet mode receive data path consists of the following sub-blocks per channel: Deserializer, Word Aligner, 8b10b Decoder, Link State Machine, Clock Tolerance Compensator, Gigabit Ethernet

Receive State Machine, and Receive CRC Checker. Figure 5-8 shows the four channels of receive data paths in a flexiPCS quad in Gigabit Ethernet mode.

***Figure 5-8. Gigabit Ethernet Receive Data Path (1 Quad)***




**Deserializer**

Data is brought on chip to the embedded SERDES where it undergoes serial to parallel. For detailed information on the operation of the LatticeSC SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

**Word Alignment**
The word aligner module performs the word alignment code word detection and alignment operation. The word alignment character is used by the receive logic to perform 10-bit word alignment upon the incoming data stream. The word alignment algorithm is described in Clause 36.2.4.9 in the 802.3-2002 1000BASE-X specification.

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

- 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 word alignment compare. Word alignment characters and the mask register is set on a per quad basis. For Gigabit Ethernet, the word alignment characters should 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).
- The first word alignment character can be set by writing Quad Interface Register Offset Address 0x15 to 0x03.
- The second word alignment character can be set by writing Quad Interface Register Offset Address 0x16 to 0x7C.
- The mask register can be set to compare word alignment bits 6 to 0 by writing Quad Interface Register Offset Address 0x14 to 0x7F (a '1' in a bit of the mask register means compare the corresponding bit in the word alignment character register).

**8b10b Decoder**
The 8b10b decoder implements an 8b10b decoder as described with the 802.3-2002 specification. The decoder performs the 10-bit to 8-bit code conversion along with verifying the running disparity.

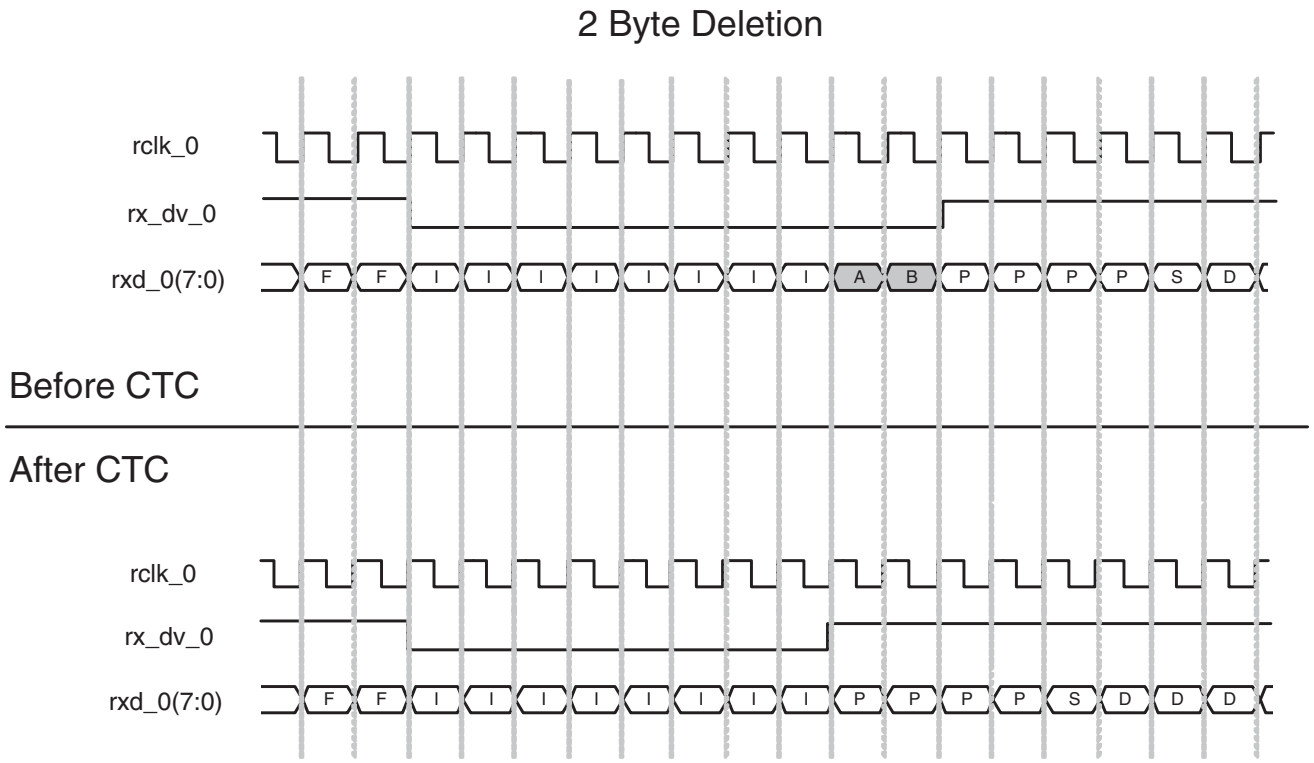
When a code violation, disparity error, or data error is detected, the rxd data is set to 0x00 and rx_er goes to '1'.

**Link State Machine**
The link synchronization state machine is an extension of the word align module. Link synchronization is achieved after the successful detection and alignment of the required number of consecutive aligned code words. The GbE State Machine is compliant to Figure 36-9 (Synchronization state machine) of IEEE 802.3-2002 with one exception. Figure 36-9 requires that four consecutive good code groups be received in order for the LSM to transition from one SYNC_ACQUIRED_{N} (where N=2,3,4) to SYNC_ACQUIRED_{N-1}. Instead, the actual LSM implementation requires five consecutive good code groups to make the transition.

**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 LatticeSC SERDES (See DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet).

A diagram illustrating 4-byte deletion is shown in Figure 5-9:

*Figure 5-9. Gigabit Ethernet Mode Clock Compensation Byte Deletion Example*




F = Frame Check Sequence

I = Idle

A = K28.5, B = D16.2 (IDLE2 Ordered Set)

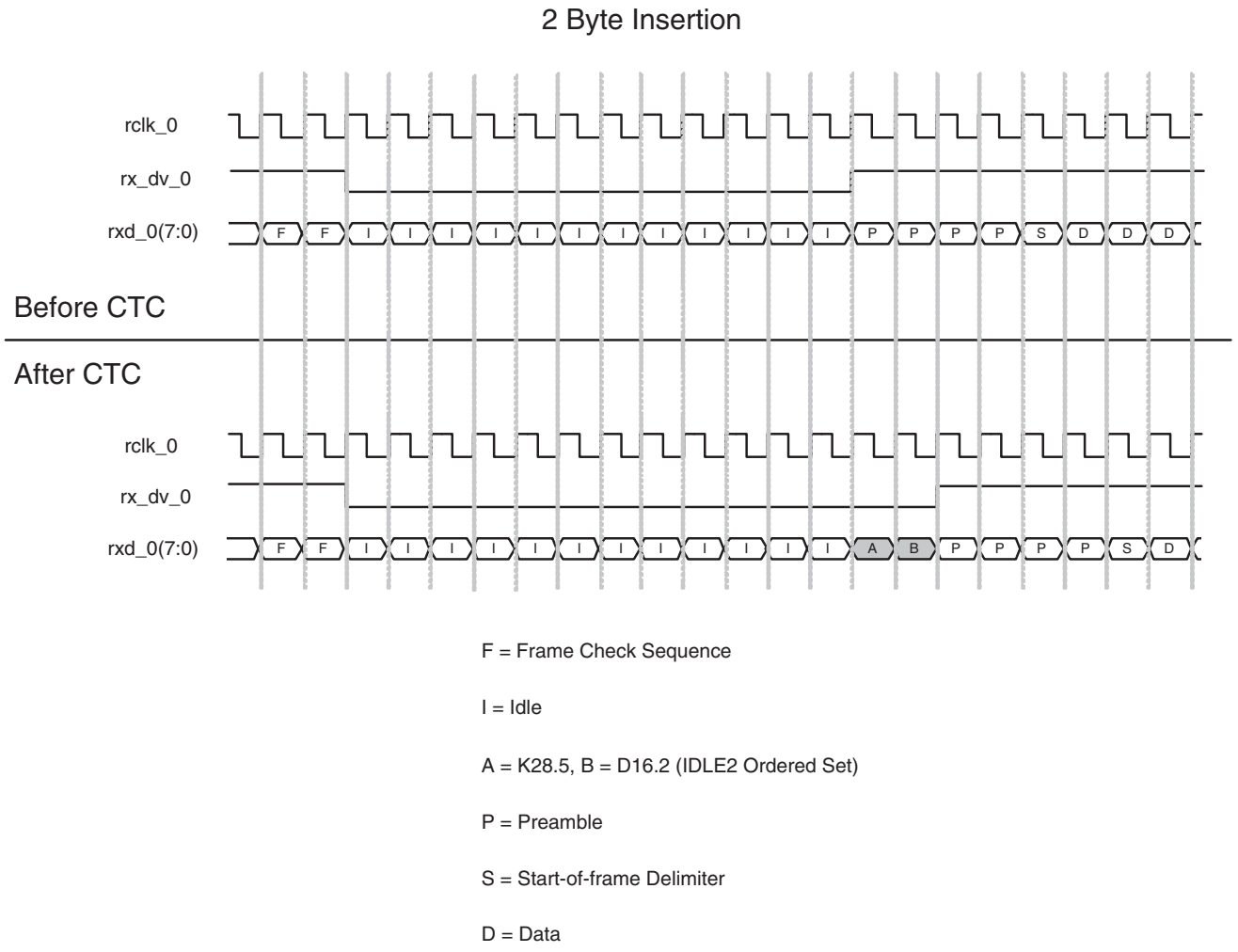
P = Preamble

S = Start-of-frame Delimiter

D = Data

A diagram illustrating 2 byte insertion is shown in Figure 5-10:

***Figure 5-10. Gigabit Ethernet Mode Clock Compensation Byte Insertion Example***




Clock compensation values are set on a quad basis. The following settings for clock compensation should be set for Gigabit Ethernet applications:

- Set the insertion/deletion pattern length to 2 and minimum inter-packet gap to 8 bytes by setting Quad Interface Register Offset Address 0x0E to 0x0B. The minimum allowed inter-packet gap after IDLE2 deletion based on these register settings is described below.
- The Gigabit Ethernet insertion/deletion pattern should be set to the IDLE2 (/I2/) ordered set. The I2 ordered set is defined as /K28.5/D16.2/. To load /K28.5/ set Quad Interface Register Offset Address 0x11 to 0xBC and Quad Interface Register Offset Address 0x13 to 0x04. To load /D16.2/ set Quad Interface Register Offset Address 0x12 to x050.
- Set the clock compensation FIFO high water and low water marks at 9 and 7 respectively (appropriate for insertion/deletion pattern length of 2) by setting Quad Interface Register Offset Address 0x0D to 0x97.
- Clock compensation FIFO underrun can be monitored by reading the appropriate Channel Interface Register Offset Address 0x85, bit 6. This can be also monitored on the flexiPCS interface pin to FPGA logic labeled ctc_urun_[0-3] if “Optional Direct Control & Status Register Access” is selected when generating the flexiPCS block within IPexpress.

- Clock compensation FIFO underrun can be monitored by reading the appropriate Channel Interface Register Offset Address 0x85, bit 7. This can be also monitored on the flexiPCS interface pin to FPGA logic labeled ctc_orun_[0-3] if “Optional Direct Control & Status Register Access” is selected when generating the flexiPCS block within IPexpress.

Table 5-4 shows the relationship between the user-defined values for inter-packet gap (written to Quad Interface Register Offset Address 0x0E, bits 6 and 7), and the guaranteed minimum number of bytes between packets after an IDLE2 deletion from the flexiPCS. 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 Gigabit Ethernet, the number of bytes per insertion/deletion is 2 (Quad Interface Register Offset Address 0x0E, bit 4 is set to ‘1’). If Quad Interface Register Offset Address 0x0E, bits 6 and 7 are set to “11”, then the minimum inter-packet gap is equal to 4 times 2 or 8 bytes. The flexiPCS will not perform an IDLE2 deletion until the minimum number of inter-packet bytes have been detected.

***Table 5-4. Minimum inter-packet gap multiplier***

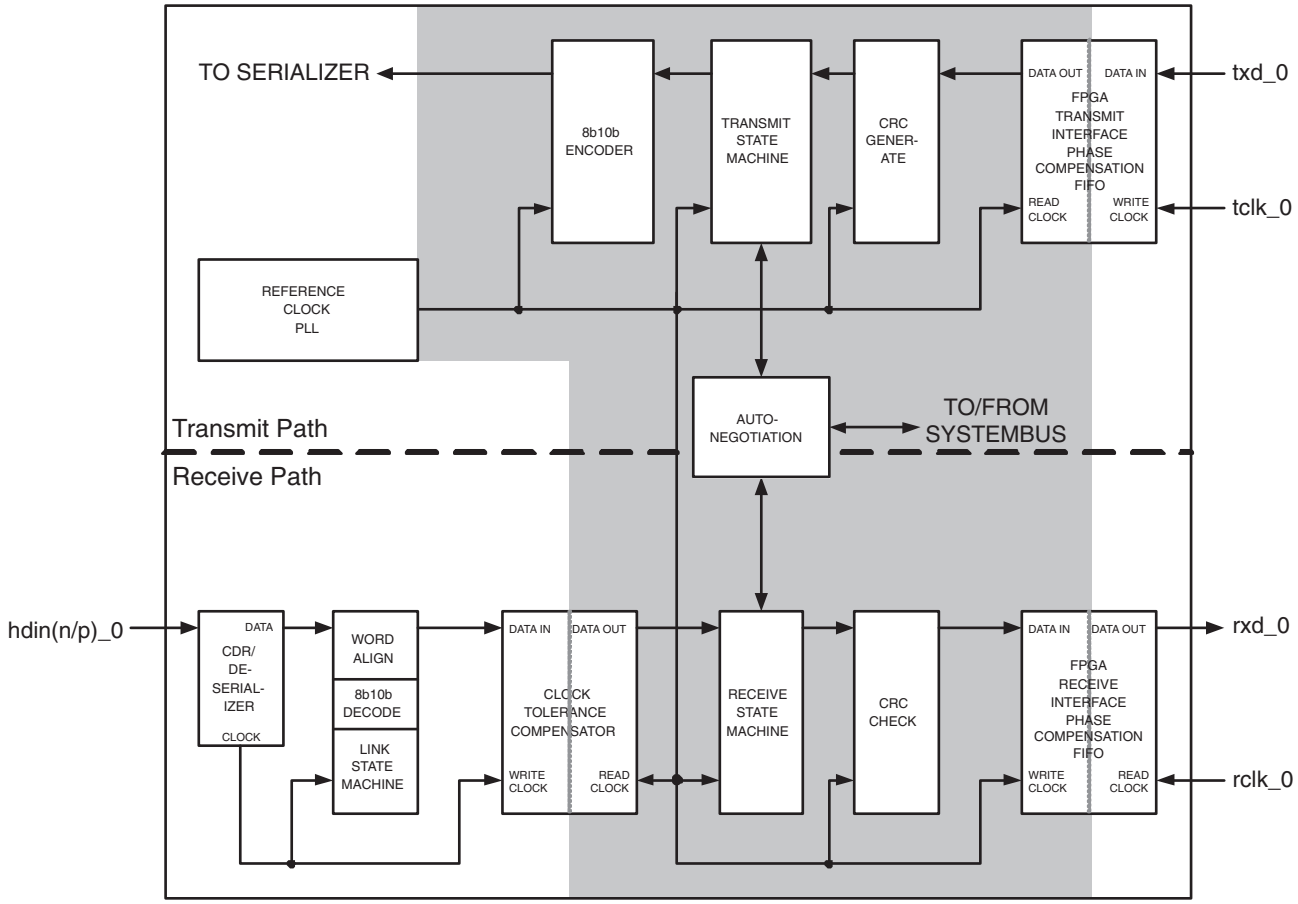
| Quad Interface Register Offset Address 0x0E | | Insertion/ Deletion Multiplier Factor |
|---|---|---|
| Bit 6 | Bit 7 | |
| 0 | 0 | 1 X |
| 0 | 1 | 2 X |
| 1 | 0 | 3 X |
| 1 | 1 | 4 X |

Figure 5-11 shows the clock domains for both transmit and receive directions for a single channel inside the flexiPCS.

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

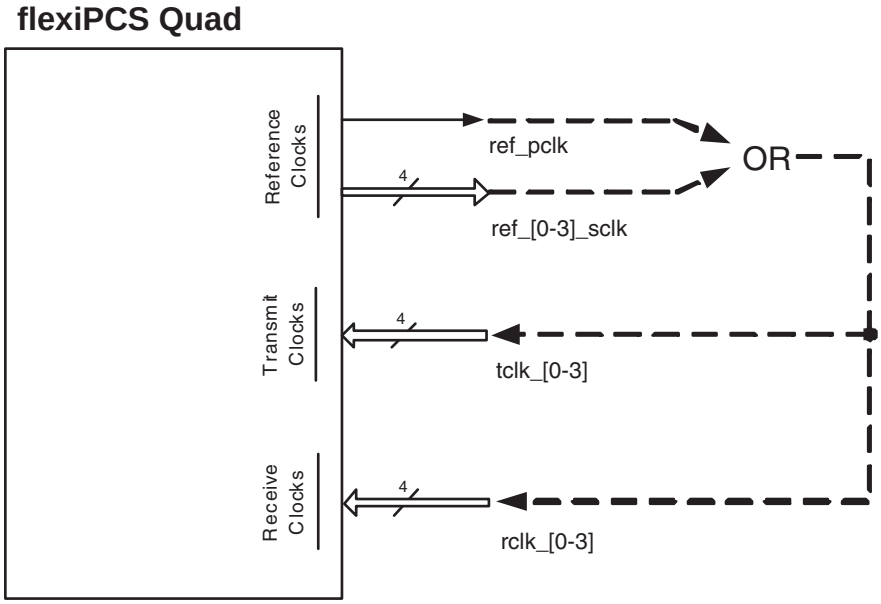
On the receive side, a clock domain transfer from the recovered channel clock to the locked reference clock occurs at the Clock Tolerance Compensator. A second clock domain transfer occurs from the locked reference clock to the rclk input at the FPGA interface at the FPGA Receive Interface phase compensation FIFO. The FPGA Receive Interface phase compensation FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. These phase compensation FIFOs (one per channel) cannot compensate for frequency variations.

***Figure 5-11. flexiPCS Clock Domain Transfers for Gigabit Ethernet Mode***




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 flexiPCS quad set to Gigabit Ethernet mode. Figure 5-12 illustrates the possible connections that would result in a synchronous interface.

*Figure 5-12. Synchronous input clocks to flexiPCS quad set to Gigabit Ethernet Mode*




**Gigabit Ethernet Receive State Machine**

The Gigabit Ethernet receive state machine performs the 8-bit data decapsulation and formatting, including the Auto-Negotiation code word extraction if Auto-Negotiation is enabled. The Gigabit Ethernet receive state machine implements the PCS receive state diagrams in Figure 36- 7a and 36-7b of the 802.3-2002 1000BASE-X specification.

All requirements of Clause 35 and 36 are provided, with the exception of the following:

- Support for half-duplex signalling to MAC
- GMII_COL and GMII_CRS signals and associated logic are not implemented
- In GbE mode, the PCS does NOT conform to the definition of carrier_detect as described in Section 36.2.5.1.4 of IEEE 802.3-2002. Instead, the carrier_detect function is true when ANY bit difference exists between the latched code group and the expected /K28.5/ based on current running disparity.

**Auto-Negotiation**

The LatticeSC flexiPCS implements the Auto-Negotiation state machine shown in the Auto-Negotiation state diagram of the IEEE Standard 802.3-2002 specification. Auto-Negotiation can be enabled by writing the appropriate Channel Interface Register Offset Address 0x05, bit 5 to a '1'. Detailed information on the Auto-Negotiation function in the flexiPCS is given in the Auto-Negotiation section.

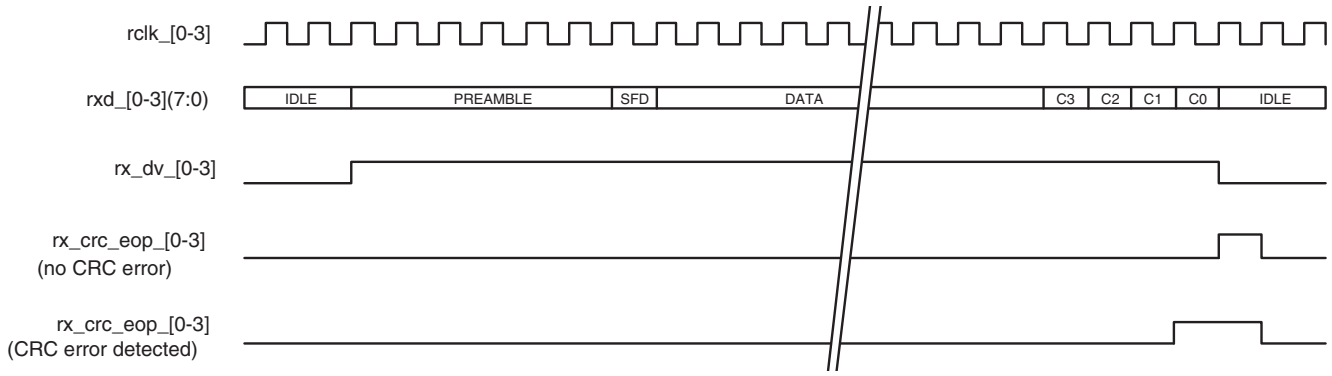
**CRC Checker**

**rx_crc_eop_[0-3]** - Per channel active high signal which indicates an end-of-packet and/or a CRC error condition. When end of packet is reached, calculated CRC is compared with the expected values. If there is no error, rx_crc_eop_[0-3] acts as an end of packet indicator and goes high for one rclk_[0-3] cycle. If the packet was in error, then rx_crc_eop_[0-3] is asserted for two clock cycles as shown in Figure 5-13.

In 16-Bit Data Bus Mode, rx_crc_eop is 2 bits wide (**rx_crc_eop_[0-3](1:0)**) with rx_crc_eop_[0-3](1) associated with rxd_[0-3](15:8) and rx_crc_eop_[0-3](0) associated with rxd_[0-3](7:0).

Note that one way to check for CRC errors is to monitor both the rx_crc_eop and the corresponding channel's rx_dv signals. In normal operation (no CRC errors), the rx_crc_eop and rx_dv will never both be high on the same clock cycle. During an CRC error condition, rx_crc_eop and rx_dv will both be high for one clock cycle.

***Figure 5-13. Gigabit Ethernet Receive CRC Error Detection***




The CRC checker options are set on a per quad basis. The CRC checker options can be set for Gigabit Ethernet compliant operation by writing Quad Interface Register Offset Address 0x1E to 0x1E (writing 0x1D to 0x00 disables the CRC generator/checker).

## Auto-Negotiation

The LatticeSC flexiPCS in Gigabit Ethernet Mode implements the Auto-Negotiation state machine shown in the Auto-Negotiation state diagram shown in Figure 37-6 of the 802.3-2002 specification. This module controls the Auto-Negotiation process and includes the optional next-page implementation.

The Auto-Negotiation logic in the flexiPCS provides the following programmable options:

- Ability to implement the following Clause 22 PHY management registers
  - Register 0: Control
  - Register 1: Status
  - Register 4: Auto-negotiation Advertisement
  - Register 5: Auto-negotiation Link Partner Base Page Ability
  - Register 6: Auto-negotiation Expansion
  - Register 7: Auto-negotiation Next Page Transmit
  - Register 8: Auto-negotiation Link Partner Received Next Page
  - Register 15: Extended Status
- Enabling/disabling Auto-Negotiation function

Auto-negotiation functions can be enabled and set on a per channel basis. Figure 5-5 includes a list of the Clause 22 PHY management register functions supported and the Channel Interface Register bits that correspond to the supported management register bits.

*Table 5-5. Auto-Negotiation Channel Interface Register Bits*

| Clause 22 Management Register Function | Management Register Bit (Register.Bits) | Corresponding Channel Interface Register (CIR) Control Bits | Corresponding Channel Interface Register (CIR) Status Bits |
|---|---|---|---|
| Auto-Negotiation restart | 0.9 | CIR 0x05, bit 4 | |
| Auto-Negotiation enable | 0.12 | CIR 0x05, bit 5 | |
| Loopback | 0.14 | | |
| Reset | 0.15 | See Note 1 | |
| Link status | 1.2 | | CIR 0x85, bit 2 |
| Auto-Negotiation complete | 1.5 | | CIR 0x85, bit 3 |
| Auto-Negotiation advertisement register | 4.15:8 | CIR 0x07, bits [0:7] | |
| Auto-Negotiation advertisement register | 4.7:0 | CIR 0x06, bits [0:7] | |
| Auto-Negotiation link partner ability register | 5.15:8 | | CIR 0x87, bits [0:7] |
| Auto-Negotiation link partner ability register | 5.7:0 | | CIR 0x86, bits [0:7] |
| Page Received | 6.1 | | CIR 0x85, bit 4 |
| Next page able | 6.2 | See Note 2 | |
| Auto-negotiation next page transmit register | 7.15:8 | CIR 0x09, bits [0:7] | |
| Auto-negotiation next page transmit register | 7.7:0 | CIR 0x08, bits [0:7] | |
| Auto-negotiation link partner next page ability register | 8.15:8 | | CIR 0x89, bits [0:7] |
| Auto-negotiation link partner next page ability register | 8.7:0 | | CIR 0x88, bits [0:7] |
| Next page loaded | | | CIR 0x85, bit 5 |

1. Each channel can be reset by writing to the appropriate bits of Quad Interface Register 0x40. Transmit and receive paths should be reset to restart the Auto-Negotiation process. Channel 0: Transmit = bit 0, Receive = bit 4. Channel 1: Transmit = bit 1, Receive = bit 5. Channel 2: Transmit = bit 2, Receive = bit 6. Channel 3: Transmit = bit 3, Receive = bit 7.
2. The LatticeSC has next page capability by default whenever the Auto-Negotiation feature is enabled. This information can be coded to bit 15 of the transmitted base page if use of the next page feature is desired.

The flexiPCS allows transmission and reception of a base page register and subsequent next page registers during a Auto-Negotiation sequence. Auto-Negotiation can be enabled by writing the appropriate Channel Interface Register Offset Address 0x05, bit 5 to a '1'. If enabled, the Auto-Negotiation sequence is started after a quad or channel reset or when the Auto-Negotiation restart bit is enabled by writing Channel Interface Register Offset Address 0x05, bit 4 to a '1'. Once started, the Auto-Negotiation sequence will run for approximately 10 ms. The length of the sequence can be reduced to four byte clock cycles for simulation purposes by writing the Channel Interface Register Offset Address 0x05, bit 3 to '1'.

The flexiPCS performs the exchange of base-page and subsequent next-pages automatically. However, priority resolution, remote fault and dual-acknowledgement processes, as required by Clause 37 must be performed by an external microprocessor or with FPGA logic external to the flexiPCS.

A typical sequence for Auto-Negotiation after reset might look like the following:

1. Enable Auto-Negotiation (CIR 0x05, bit 5)
2. Write the Auto-Negotiation advertisement register (CIR 0x07 & CIR 0x06)
3. Write the Auto-Negotiation next page transmit register (CIR 0x09 & CIR 0x08). This register will be transmitted after the advertisement register if bit 15 of the advertisement register is set to '1' (indicating LatticeSC is next page capable) and bit 15 of the received link partner ability register is also '1' (indicating that link partner is able to receive a next page).
4. Restart the Auto-Negotiation sequence (CIR 0x05, bit 4)

5. Advertisement register will be transmitted and receiver will download link partner ability register to CIR 0x87 & CIR 0x86. Once link partner ability register is loaded the page received status bit (CIR 0x85, bit 4) will go high.

6. If conditions to transmit next page are met (See step 3), next page will be transmitted. Received link partner next page ability register will be downloaded to CIR 0x89 & CIR 0x88. Next page received status bit (CIR 0x85, bit 5) will go high.

7. If bit 15 of the next page transmit register is set to '1' (indicating LatticeSC is next page capable) and bit 15 of the received link partner next page ability register is also '1' (indicating that link partner is ready to receive a next page), then an additional next page can be sent. Additional next page can be written to the Auto-Negotiation next page transmit register (CIR 0x09 & CIR 0x08).

8. Next page transmit/receive procedure repeats until no next pages are requested.

9. Auto-Negotiation process will automatically stop after approximately 10 ms. When sequence is done, the Auto-Negotiation complete status bit (CIR 0x85, bit 3) will go high.

Figure 5-6 maps the Config_Reg base page bit specification to the equivalent LatticeSC flexiPCS Channel Interface Register bits:

***Table 5-6. Base Page Channel Interface Register Bits***

| Config_Reg Bit | Config_Reg Base Page Function | flexiPCS Channel Interface Register Offset Address | flexiPCS Channel Interface Register Bit |
|---|---|---|---|
| 0 | rsvd | CIR 0x06 | 7 |
| 1 | rsvd | | 6 |
| 2 | rsvd | | 5 |
| 3 | rsvd | | 4 |
| 4 | rsvd | | 3 |
| 5 | FD | | 2 |
| 6 | HD | | 1 |
| 7 | PS1 | | 0 |
| 8 | PS2 | CIR 0x07 | 7 |
| 9 | rsvd | | 6 |
| 10 | rsvd | | 5 |
| 11 | rsvd | | 4 |
| 12 | RF1 | | 3 |
| 13 | RF2 | | 2 |
| 14 | Ack | | 1 |
| 15 | NP | | 0 |

Note that bit 5 of the base page register (FD) should be set to '1' to reflect Full Duplex operation of the LatticeSC device.

Figure 5-7 maps the Config_Reg next page bit specification to the equivalent LatticeSC flexiPCS Channel Interface Register bits:

*Table 5-7. Next Page Channel Interface Register Bits*

| Config_Reg Bit | Config_Reg Next Page Function | flexiPCS Channel Interface Register Offset Address | flexiPCS Channel Interface Register Bit |
|---|---|---|---|
| 0 | M0/U0 | CIR 0x86 | 7 |
| 1 | M1/U1 | | 6 |
| 2 | M2/U2 | | 5 |
| 3 | M3/U3 | | 4 |
| 4 | M4/U4 | | 3 |
| 5 | M50/U5 | | 2 |
| 6 | M6/U6 | | 1 |
| 7 | M7/U7 | | 0 |
| 8 | M8/U8 | CIR 0x87 | 7 |
| 9 | M9/U9 | | 6 |
| 10 | M10/U10 | | 5 |
| 11 | T | | 4 |
| 12 | Ack2 | | 3 |
| 13 | MP | | 2 |
| 14 | Ack | | 1 |
| 15 | NP | | 0 |

## Control & Status

The Control & Status pins for the Gigabit Ethernet mode can be optionally selected on a per quad basis when generating the flexiPCS quad interface files with the ispLEVER tools. Each of these control and status functions are also accessible through equivalent Quad Interface and Channel Interface Registers. The control and status signals allow direct real time access of these functions from FPGA logic.

**Control**

**felb_[0-3]** - Per channel, active high control signal which sets up the appropriate channel in Far-End Loopback mode. This mode is generally used for testing the flexiPCS logic, looping back receive data back to the transmit direction without crossing the FPGA logic interface. For more information on the details of Far-End Loopback mode, see the **flexiPCS Testing** Section of the flexiPCS Data Sheet. The Far-End Loopback mode can also be initiated by writing Channel Interface Register Offset Address 0x00, bit 5 to '1'.

**lsm_en_[0-3]** - Per channel Receive Link State Machine Enable. A change (either 0 to 1 or 1 to 0) on the lsm_en_[0-3] resets the Receive Link State Machine into No Sync mode. The Receive Link Machine Enable can also be set by writing Channel Interface Register Offset Address 0x00, bit 7 to '1'.

**Status**

**lsm_status_[0-3]** - Per channel active high signal from the Receive Link State Machine indicating link synchronization successful. The link synchronization status can also be read from the appropriate Quad Interface Register Offset Address 0x84, bits [4:7] (bit 4 for channel 0, bit 5 for channel 1, bit 4 for channel 2, and bit 7 for channel 3).

**ctc_urun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has underrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 6.

**ctc_orun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has overrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 7.

## Status Interrupt Registers

Some of the status registers associated with Gigabit Ethernet operation have an associated status interrupt and status interrupt enable register. Any flexiPCS status interrupt register will generate an interrupt to the Systembus block (if used in the design). For a description of the operation of interrupts with the Systembus, refer to the Systembus section of the LatticeSC/M Family Data Sheet. Operation of the interrupt registers for the flexiPCS is as follows:

User must set the appropriate status interrupt enable bit to a '1'

Whenever the associated status bit goes high, its status interrupt bit will also go high. The status interrupt bit will remain high even if the associated status bit goes low.

The status interrupt bit will remain high until it is read, when it will automatically be cleared. If the associated status bit is still high, then the status interrupt bit will go high again (until read the next time).

Table 5-8 shows a listing of the status registers associated with Gigabit Ethernet operation which have a corresponding status interrupt and status interrupt enable bit.

***Table 5-8. Status Interrupt Register Bits***

| Status Register Bit Function | Status Register Bit Address | Status Interrupt Enable Register Bit Address | Status Interrupt Register Bit Address |
|---|---|---|---|
| Auto-Negotiation Resolve Priority | CIR 0x85, bit 2 | CIR 0x0A, bit 2 | CIR 0x8A, bit 2 |
| Auto-Negotiation Complete | CIR 0x85, bit 3 | CIR 0x0A, bit 3 | CIR 0x8A, bit 3 |
| Auto-Negotiation Page Received | CIR 0x85, bit 4 | CIR 0x0A, bit 4 | CIR 0x8A, bit 4 |
| Auto-Negotiation Next Page Loaded | CIR 0x85, bit 5 | CIR 0x0A, bit 5 | CIR 0x8A, bit 5 |
| Receive Link State Machine Status (lsm_status) Channel 0 | QIR 0x84, bit 4 | QIR 0x1C, bit 4 | QIR 0x85, bit 4 |
| Receive Link State Machine Status (lsm_status) Channel 1 | QIR 0x84, bit 5 | QIR 0x1C, bit 5 | QIR 0x85, bit 5 |
| Receive Link State Machine Status (lsm_status) Channel 2 | QIR 0x84, bit 6 | QIR 0x1C, bit 6 | QIR 0x85, bit 6 |
| Receive Link State Machine Status (lsm_status) Channel 3 | QIR 0x84, bit 7 | QIR 0x1C, bit 7 | QIR 0x85, bit 7 |
| Clock Tolerance Compensation FIFO underrun | CIR 0x85, bit 6 | CIR 0x0A, bit 6 | CIR 0x8A, bit 6 |
| Clock Tolerance Compensation FIFO overrun | CIR 0x85, bit 7 | CIR 0x0A, bit 7 | CIR 0x8A, bit 7 |

# Fibre Channel Mode
# LatticeSC/M Family flexiPCS Data Sheet

January 2008 Data Sheet DS1005

## Introduction

The Fibre Channel mode of the flexiPCS (Physical Coding Sublayer) block supports the 1G and 2G Fibre Channel protocol for FC-0 and FC-1 layers and portions of the FC-2 layer. The Fibre Channel mode is intended for Fibre Channel applications on a single SERDES channel. The LatticeSC SERDES can support 1G (1.0625 Gbps) and 2G (2.125 Gbps) Fibre Channel applications. For 10 gigabit Fibre Channel applications, refer to the XAUI Mode section of the LatticeSC/M Family flexiPCS Data Sheet.

The LatticeSC flexiPCS in Fibre Channel mode supports the following operations:

**Transmit Path (From LatticeSC device to line):**

- Cyclic Redundancy Check (CRC) generation and insertion into Fibre Channel frame
- EOF Disparity Control which supports End of Frame ordered-set insertion
- 8b10b Encoding

**Receive Path (From line to LatticeSC device):**

- Word Alignment based on IEEE 802.3-2002 defined alignment characters.
- 8b10b Decoding
- Link State Machine function.
- Clock Tolerance Compensation logic capable of accommodating clock domain differences.
- Cyclic Redundancy Code (CRC) Checking

### LatticeSC flexiPCS Quad Module

Devices in the LatticeSC family have up to 8 quads of embedded flexiPCS logic. Each quad in turn supports 4 independent full-duplex data channels. A single channel can support a fully compliant Fibre Channel data link and each quad can support up to four such channels. Note that mode selection is done on a per quad basis. Therefore, a selection of Fibre Channel mode for a quad dedicates all four channels in that quad to Fibre Channel mode.

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

Operation of the SERDES requires the user to provide a reference clock (or clocks) to each active quad to provide a synchronization reference for the SERDES PLLs. Reference clocks are shared among all four channels of a quad. If the transmit and receive frequencies are within the specified ppm tolerance for the LatticeSC SERDES (See DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet), only one reference clock for both transmit and receive is needed for both transmit and receive directions. If the receive frequency is not within the specified ppm tolerance for the LatticeSC SERDES (See the DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet) of the transmit frequency, then separate reference clocks for the transmit and receive directions must be provided.

Simultaneous support of different (non-synchronous) high-speed data rates is possible with the use of multiple quads. Multiple frequencies that are exact integer multiples can be supported on a per channel basis through the use of the full-rate/half-rate mode options (see the **SERDES Functionality section** for more details).

© 2008 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

## Quad and Channel Option Control

Although the mode selection and reference frequency covers an entire quad, many options covering clocking and data formatting are available on a per channel basis. These options are detailed in the following Fibre Channel Mode description. Some of these options can be controlled directly through the FPGA interface pins made available on the Fibre Channel Quad Module.

Other options are available only through dedicated registers that can be written and read via the embedded System Bus Interface (see the System Bus section for more details on the operation of the System Bus), or during device configuration. Depending on the specific option, a register may affect operation of multiple quads, a single quad or an individual channel in a quad. Registers dedicated to multiple quads are listed in the Inter-quad Interface Register Map in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet. Registers dedicated to one quad are listed in the Quad Interface Register Map. Registers dedicated to a single channel are listed in the Channel Interface Register Map.

### Register Map Addressing

Figure 6-1 provides a map of base register addresses for any of the flexiPCS quads on a LatticeSC device. Note that different devices may have different numbers of available quads up to a total of 8 quads (or 32 channels) maximum.

Inter-quad Interface, Quad Interface, and Channel Interface Registers are listed by Offset Address. Each Inter-quad Interface Register, Quad Interface Register, and Channel Interface Register has a unique 18-bit address determined by the specific quad or channel targeted. The addressing of Inter-quad Interface Registers, Quad Interface Registers, and Channel Interface Registers is shown Figure 6-1.

Base addresses are dependant on the physical location of a given quad or channel on a LatticeSC device. Each quad and channel has an associated set of control and status registers. These registers are referred throughout this data sheet by their offset addresses. The unique 18-bit address of a control or status register for a specific quad or channel is simply the offset address of that register added to the base address for that specific quad or channel as shown in Figure 6-1.

Channel Interface Registers can be written in one of three methods. One method is to write to one specific register corresponding to one specific channel. The other two methods involve writing to multiple channel registers with just one write operation. This is referred to as a Broadcast write. A Broadcast write is useful when multiple channel registers are to be written with the same value. The first method of Broadcast writing involves writing to all four channel registers in a quad at one time. A second method of Broadcast writing involves writing to all channel registers for all quads on the left or right side of the device at one time. Therefore, a specific Channel Interface Register may be accessed through any one of three channel addresses, depending on the number of channel registers being written to at any one time.

A broadcast write to multiple quads cannot be used to power on SERDES in any device-package combination that does not have all SERDES quads bonded because of excess power on the board and jitter is also affected.

Throughout this document, the byte-wide data at each offset address is listed with a hexadecimal value with bit 0 as the MSB and bit 7 as the LSB as per the PowerPC convention. For example if the value for Quad Interface Register Offset Address 0x02 is stated to be 0x15, the bit pattern defined is as shown in Table 6-1:

***Table 6-1. Control/Status Register Bit Order Example***

| **Quad Interface Register Offset Address 0x02 = 0x15** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Data Bit 0 | Data Bit1 | Data Bit 2 | Data Bit 3 | Data Bit 4 | Data Bit 5 | Data Bit 6 | Data Bit 7 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| **1** | | | | **5** | | | |

A full list of register Offset Addresses is provided in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet.

***Figure 6-1. flexiPCS Inter-quad, Quad, and Channel Interface Register Map Addressing***

## Auto-Configuration

Initial register setup for each flexiPCS mode can be performed without accessing the system bus by using the auto-configuration feature in ispLEVER. IPexpress generates 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 system bus is therefore not needed if all quads are to be set via auto-configuration files. However, the system bus must be included in a design if the user needs to change control registers or monitor status registers during operation. Note that a quad reset (Quad Interface Register 0x43, bit 7 or the **quad_rst** port at the FPGA interface) will reset all registers back to their default (not auto-configuration) values. If a quad reset is issued, the flexiPCS registers need to be rewritten via the Systembus.

## Fibre Channel Register Settings

The auto-configuration file sets registers that perform the following operations. If the auto-configuration file is not used, the appropriate registers need to be programmed via the Systembus. For more options, refer to the flexiPCS register map in the **Memory Map** section of the **LatticeSC/M Family flexiPCS Data Sheet**.

**A flexiPCS quad can be set to Fibre Channel mode by writing Quad Interface Register Offset Address 0x18 bits[0:7] to 0x08.**

This register value automatically sets up the following options in the flexiPCS for the given quad. Details on the functionality of each chosen option is provided in the **Fibre Channel Mode Detailed Description** section.

**Transmit Path**

- EOF Disparity Control is enabled
- 8b10b encoder is enabled

**Receive Path**

- Word aligner is enabled
- 8b10b decoder is enabled
- Receive Link State Machine is set to Fibre Channel Mode

Other register settings are required to operate a channel or channels in Fibre Channel Mode. The following is a list of options that must be set for Fibre Channel operation. More detail for each option is included in the **Fibre Channel Mode Detailed Description** section.

- Powerup of appropriate quad and channel. Quad powerup is chosen by writing the appropriate Quad Interface Register Offset Address 0x28, bit 1 to ‘1’. Channel powerup is chosen by writing the appropriate Channel Interface Register Offset Address 0x13, bit 6 (receive direction) and/or bit 7 (transmit direction) to ‘1’. By default, all channels and quads are powered down.
- Receive Link State Machine enable. By default, all channel Receive Link State Machines are disabled. The Receive Link State Machine for a given channel can be enabled by writing the appropriate Channel Interface Register Offset Address 0x00, bit 7 to ‘1’.
- Setting SERDES reset low at end of flexiPCS setup. By default, the SERDES reset is set to ‘1’ (SERDES are reset). The SERDES reset can be set low by writing Quad Interface Register Offset Address 0x41, bit 5 to a ‘0’. For more information on proper flexiPCS powerup and reset sequencing, refer to the **flexiPCS Reset Sequences** and **flexiPCS Power Down Control Signals** descriptions in the LatticeSC/M Family flexiPCS Data Sheet **SERDES Functionality** section.
- Word alignment character(s), such as a comma
- Clock Tolerance Compensation insertion/deletion ordered set values and FIFO settings

- Cyclic Redundancy Code (CRC) generator/checker enable and settings

## Fibre Channel Auto-Configuration Example

An example of an auto-configuration file for a quad set to Fibre Channel Mode, with no Auto-Negotiation enabled, half-rate mode, and with only channel 1 enabled is shown in Figure 6-2. Format for the auto-configuration file is

register type (quad=quad register, ch1=channel 1 register), offset address in hex, register value in hex, # comment

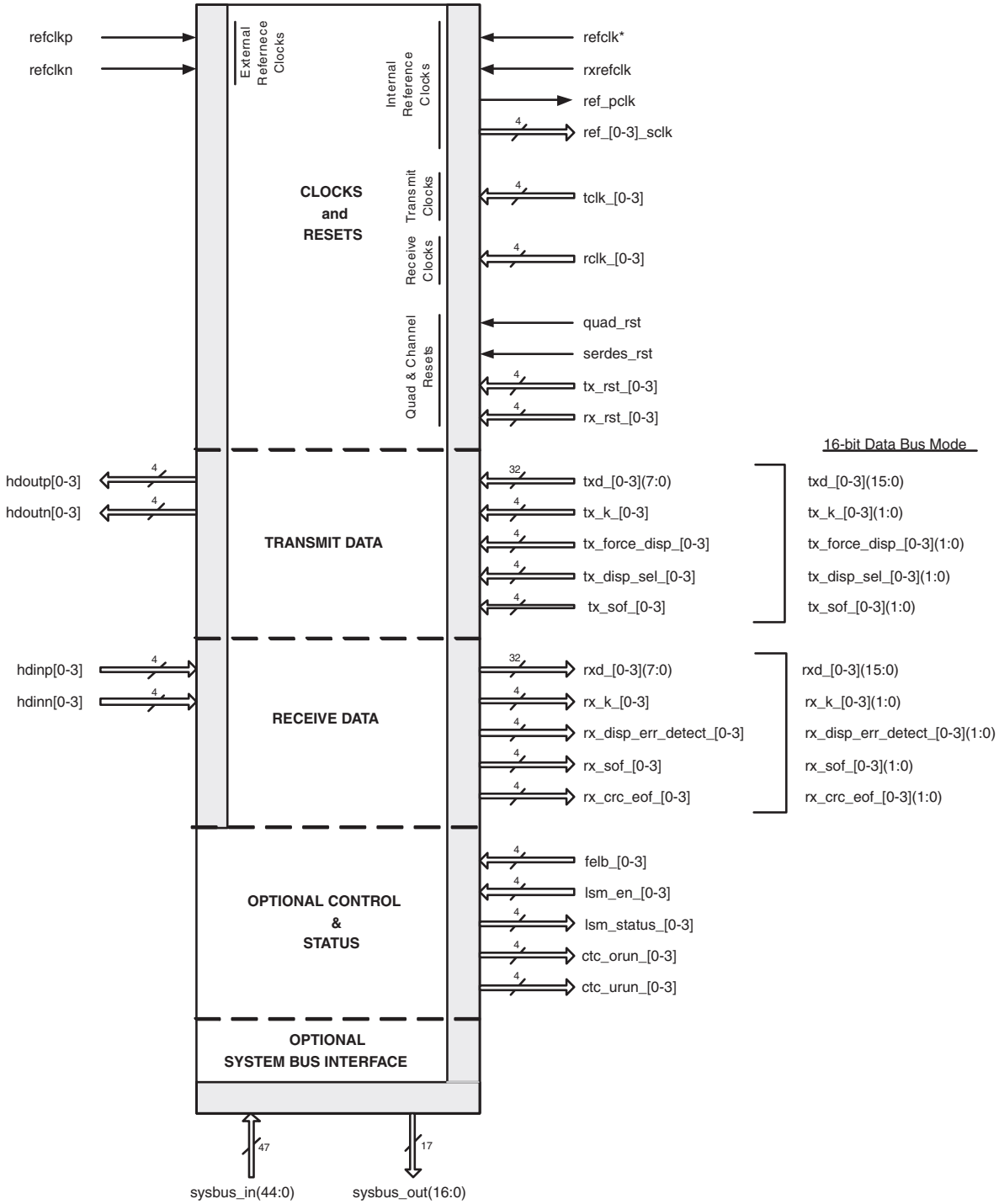
***Figure 6-2. Fibre Channel Auto-Configuration Example File***

```
quad 18 08      #  Set quad to Fibre Channel Mode
quad 29 01      #  Set reference clock select to refclk(p/n) inputs
quad 28 40      #  Set bit clock multiplier to 10X (for half rate mode), quad powerup set to '1'
quad 02 15      #  Set ref_pclk to channel 1 clock
quad 19 00      #  Set FPGA interface data bus width to 8-bit
quad 14 7F      #  Set word alignment mask to compare lowest 7 LSBs
quad 15 03      #  Set positive disparity comma value
quad 16 7C      #  Set negative disparity comma value
quad 0E 05      #  Set insertion/deletion to 4 bytes for CTC, set inter-packet gap
quad 0D A6      #  Set CTC FIFO watermarks (high=10, low=6)
quad 0F BC      #  1st byte of FC Idle pattern for CTC (K28.5)
quad 10 95      #  2nd byte of FC Idle pattern for CTC (D21.4)
quad 11 B5      #  3rd byte of FC Idle pattern for CTC (D21.5)
quad 12 B5      # 4th byte of FC Idle pattern for CTC (D21.5)
quad 13 40      #  K bits for FC Idle pattern
quad 1D 60      #  Set CRC generator options
quad 1E 00      #  Set CRC checker options
ch1 00 09       #  Turn on channel 1 link state machine, rx_sfd ports enabled
ch1 13 0F       #  Powerup channel 1 transmit and receive directions, set rate mode to “half”
ch1 14 90       # 16% pre-emphasis on SERDES output buffer
ch1 15 10       # +6 dB equalization on SERDES input buffer
# These lines must appear last in the autoconfig file. These lines apply the
# correct reset sequence to the PCS block upon bitstream configuration
quad 41 00 # de-assert serdes_rst
quad 40 ff   # assert datapath reset for all channels
quad 40 00 # de-assert datapath reset for all channels
```

# Fibre Channel Mode

IPexpress allows the designer to choose the mode for each flexiPCS quad used in a given design. Figure 6-3 displays the resulting port interface to one flexiPCS quad that has been set to Fibre Channel mode on all four channels.

***Figure 6-3. Fibre Channel Mode Pin Diagram***




* The refclk inputs for all active quads on a device must be connected to the same clock . The rxrefclk inputs for all active quads on a device do not have to be connected to the same clock.

## Fibre Channel Mode Pin Description

Table 6-2 lists all inputs and outputs to/from a flexiPCS quad in Fibre Channel mode. A brief description for each port is given in the table. A more detailed description of the function of each port is given in the **Fibre Channel Mode Detailed Description**.

***Table 6-2. Fibre Channel Mode Pin Description***

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Reference Clocks** | | | |
| refclkp | In from I/O pad | N/A | Reference clock input, positive. Dedicated CML input. |
| refclkn | In from I/O pad | N/A | Reference clock input, negative. Dedicated CML input |
| ff_refclk | In from FPGA | N/A | Optional reference clock input from FPGA logic. Can be used instead of I/O pin reference clock.<br><br>The refclk inputs for all active quads on a device must be connected to the same clock. |
| ff_rxrefclk | In from FPGA | N/A | Optional receive reference clock input from FPGA logic. Can be used instead of I/O pin reference clock. The rxrefclk inputs for all active quads do not have to be connected to the same clock. |
| ref_pclk | Out to FPGA | N/A | Locked reference clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| ref_[0-3]_sclk | Out to FPGA | N/A | Per channel locked reference clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| **Transmit Clocks** | | | |
| tclk_[0-3] | In from FPGA | N/A | Per channel transmit clock inputs from FPGA. Used to clock the TX data phase compensation FIFO with clock synchronous to the reference clock. May also be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Receive Clocks** | | | |
| rclk_[0-3] | In from FPGA | N/A | Per channel receive clock inputs from FPGA. May be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference and/or receive reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Resets** | | | |
| quad_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of SERDES and PCS logic. |
| serdes_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of the SERDES. |
| tx_rst_[0-3] | In from FPGA | ASYNC | Per channel active high, asynchronous reset of individual transmit channel of PCS logic. |
| rx_rst_[0-3] | In from FPGA | ASYNC | Per channel active high, asynchronous reset of individual receive channel of PCS logic. |
| **Transmit Data** | | | |
| hdoutp[0-3] | Out to I/O pad | N/A | High-speed CML serial output, positive. |
| hdoutn[0-3] | Out to I/O pad | N/A | High-speed CML serial output, negative. |

*Table 6-2. Fibre Channel Mode Pin Description*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| txd_[0-3](7:0)<br><br>txd_[0-3](15:0)* | In from FPGA | tclk_[0-3] | Per channel parallel transmit data bus from the FPGA. Bus is 8-bits wide if QIR 0x19, bit 4 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 4 = '1'. |
| tx_k_[0-3]<br><br>tx_k_[0-3]* | In from FPGA | tclk_[0-3] | Per channel transmit control word indicator.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_force_disp_[0-3]<br><br>tx_force_disp_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel active high, force disparity enable.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_disp_sel_[0-3]<br><br>tx_disp_sel_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel disparity select. High forces positive disparity, low forces negative disparity.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_sof_[0-3]<br><br>tx_sof_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel active high Start of Frame indicator. Used for CRC initialization.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| **Receive Data** | | | |
| hdinp[0-3] | In from I/O pad | N/A | High-speed CML serial input, positive. |
| hdinn[0-3] | In from I/O pad | N/A | High-speed CML serial input, negative. |
| rxd_[0-3](7:0)<br><br>rxd_[0-3](15:0)* | Out to FPGA | rclk_[0-3] | Per channel parallel receive data bus to the FPGA. Bus is 8-bits wide if QIR 0x19, bit 5 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 5 = '1'. |
| rx_k_[0-3]<br><br>rx_k_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel receive control word indicator.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_disp_err_detect_[0-3]<br><br>rx_disp_err_detect_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high disparity error detection.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_sof_[0-3]<br><br>rx_sof_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high Start-of-Frame detection if CIR 0x00, bit 4 = '1'. Indicates 8b10b code violation detection otherwise.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_crc_eof_[0-3]<br><br>rx_crc_eof_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high indicates that a end of frame and/or CRC error was detected.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| **Optional Control & Status** | | | |
| felb_[0-3] | In from FPGA | ASYNC | Per channel active high far-end loopback enables. |
| lsm_en_[0-3] | In from FPGA | ASYNC | Per channel receive link state machine enable. |
| lsm_status_[0-3] | Out to FPGA | ASYNC | Per channel signal from receive link state machine indicating successful link synchronization. |
| ctc_urun_[0-3] | Out to FPGA | ASYNC | Per channel active high flag indicator that clock tolerance compensation FIFO has underrun. |
| ctc_orun_[0-3] | Out to FPGA | ASYNC | Per channel active high flag indicator that clock tolerance compensation FIFO has overrun. |
| **Optional System Bus Interface** | | | |

*Table 6-2. Fibre Channel Mode Pin Description*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| sysbus_in(44:0) | In from FPGA | N/A | Control and data signals from the internal system bus. |
| sysbus_out(16:0) | Out to FPGA | N/A | Control and data signals to the internal system bus. |

# Fibre Channel Mode Detailed Description

The following section provides a detailed description of the operation of a flexiPCS quad set to Fibre Channel mode. The functional description is organized according to the port listing shown in Figure 6-3. The System Bus Offset Address for any control and status registers relevant to a given function are provided with the description of the operation of that function.

## Clocks & Resets

A detailed description of all SERDES clock, data, and reset ports and recommended reset sequencing is provided in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet. The following is a recommended setup of the SERDES for Fibre Channel applications.

For 1G Fibre Channel applications, the bit clock rate is 1.0625 Gbps. This falls into the range defined as “half rate” mode for the SERDES PLLs. Full rate mode is selected separately for each channel and each direction by writing the appropriate Channel Interface Register Offset Address 0x13, bit 5 (transmit) and bit 4 (receive) to a ‘1’.

The reference clock frequency is determined by the reference clock mode chosen. Table 6-4 shows the possible reference clock frequencies for various reference clock modes which result in a 1.0625 Gbps internal bit rate.

***Table 6-3. 1G Fibre Channel Reference Clock Frequency Options***

| Half Rate Mode<br><br>Channel Interface Register Offset Address 0x13<br><br>Transmit - Bit 5 = ‘1’<br>Receive - Bit 4 = ‘1’ | | | | |
|---|---|---|---|---|
| Reference Clock Mode<br><br>Quad Interface Register Offset Address = 0x28, Bits [2:3] | Reference Clock Frequency | Internal Serial (bit) Clock Frequency | FPGA Interface Clock Frequency<br><br>Quad Interface Register Offset Address 0x19 | |
| | | | 8 Bit Data Bus Mode<br><br>Transmit - Bit 4 = ‘0’<br>Receive - Bit 5 = ‘0’ | 16 Bit Data Bus Mode<br><br>Transmit - Bit 4 = ‘1’<br>Receive - Bit 5 = ‘1’ |
| 00 | 106.25 MHz | 1.0625 GHz | 106.25 MHz | 53.125 MHz |
| 01 | 212.5 MHz | 1.0625 GHz | 106.25 MHz | 53.125 MHz |
| 10 | 425 MHz | 1.0625 GHz | 106.25 MHz | 53.125 MHz |

For 2G Fibre Channel applications, the bit clock rate is 2.125 Gbps. This falls into the range defined as “full rate” mode for the SERDES PLLs. Full rate mode is selected separately for each channel and each direction by writing the appropriate Channel Interface Register Offset Address 0x13, bit 5 (transmit) and bit 4 (receive) to a ‘0’ (default).

The reference clock frequency is determined by the reference clock mode chosen. Table 6-4 shows the possible reference clock frequencies for various reference clock modes which result in a 2.125 Gbps internal bit rate.

*Table 6-4. 2G Fibre Channel Reference Clock Frequency Options*

| Full Rate Mode<br><br>Channel Interface Register Offset Address 0x13<br><br>Transmit - Bit 5 = '0'<br>Receive - Bit 4 = '0' | | | | |
|---|---|---|---|---|
| Reference Clock Mode<br><br>Quad Interface Register Offset Address = 0x28, Bits [2:3] | Reference Clock Frequency | Internal Serial (bit) Clock Frequency | FPGA Interface Clock Frequency<br><br>Quad Interface Register Offset Address 0x19 | |
| | | | 8 Bit Data Bus Mode<br><br>Transmit - Bit 4 = '0'<br>Receive - Bit 5 = '0' | 16 Bit Data Bus Mode<br><br>Transmit - Bit 4 = '1'<br>Receive - Bit 5 = '1' |
| 00 | 106.25 MHz | 2.125 GHz | 212.5 MHz | 106.25 MHz |
| 01 | 212.5 MHz | 2.125 GHz | 212.5 MHz | 106.25 MHz |
| 10 | 425 MHz | 2.125 GHz | 212.5 MHz | 106.25 MHz |

*Note: There are also frequency dependent SERDES performance control bits that are not writable via the System-bus. These control bits are set in the auto-configuration file generated within IPexpress and passed into the bit-stream through the normal FPGA design flow (map, par, bitgen). To change the bit rate for a SERDES, specify the proper values for the affected flexiPCS quad in IPexpress and regenerate the auto-configuration file. Failure to do this may result in non-optimal SERDES operation.*

Additional information on all reference clock rate modes can be found in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

**Quad & Channel Resets**

Resets are provided to reset an entire quad (either SERDES logic only or all flexiPCS logic) or each individual channel (all flexiPCS logic per channel). These resets are driven from the FPGA logic or by writing the appropriate Quad or Channel Interface Register. A summary of the control signals provided for reset are listed below. More detail on recommended initialization and reset sequences for the SERDES is provided in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

The following inputs to the flexiPCS from the FPGA are enabled as long as Quad Interface Register Offset Address 0x42, bit 7 is set to '1' (which is the default on powerup). Writing a '0' to this bit will disable these reset inputs.

**quad_rst** - Active high, asynchronous signal from FPGA resets all SERDES and PCS logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x43, bit 7 to '1'.

**serdes_rst** - Active high, asynchronous signal from FPGA resets all SERDES (but not PCS) logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x41, bit 5 to '1'.Note that this bit is preset to '1' on powerup and must be written to '0' before operation of the SERDES can commence.

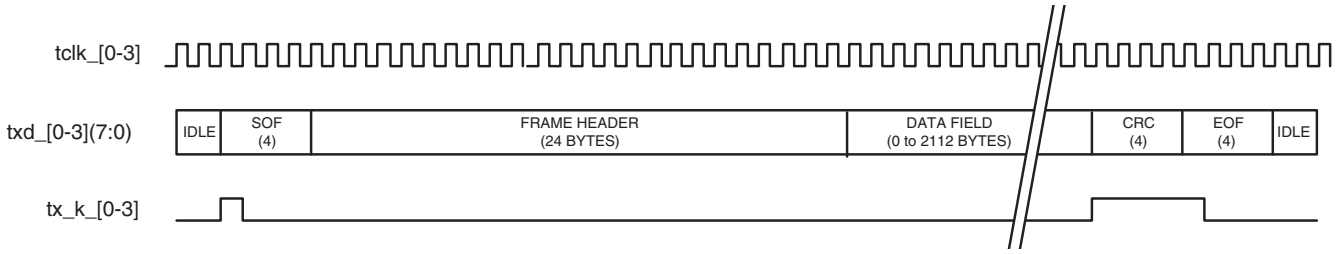
**tx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one transmit channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [4:7]. Bit 7 resets transmit channel 0, bit 6 resets transmit channel 1, bit 5 resets transmit channel 2, and bit 4 resets transmit channel 3.

**rx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one receive channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [0:3]. Bit 3 resets receive channel 0, bit 2 resets receive channel 1, bit 1 resets receive channel 2, and bit 0 resets receive channel 3.

## Transmit Data

When configured into Fibre Channel mode, a flexiPCS quad provides four transmit data paths from an 8-bit parallel interface at the FPGA logic interface to a high-speed serial line interface at the device outputs. Figure 6-4 shows the typical operation of this interface at the flexiPCS interface.

***Figure 6-4. Fibre Channel Transmit FPGA Interface Signals***




**txd_[0-3](7:0)** - Per channel transmit data from the FPGA logic interface. Each quad supports up to 4 independent channels of 8-bit wide parallel data by default. Each txd_[0-3][7:0] is an eight bit data bus that is driven synchronously with respect to the corresponding tclk_[0-3].

In 16-Bit Data Bus Mode, this bus is 16-bits wide (**txd_[0-3](15:0)**).

**tx_k_[0-3]** - Per channel, active high control character indicator.

In 16 Bit Data Bus Mode, tx_k is 2 bits wide (**tx_k_[0-3](1:0)**) with tx_k_[0-3](1) associated with txd_[0-3](15:8) and tx_k_[0-3](0) associated with txd_[0-3](7:0).

*Note: As defined in the Fibre Channel standard, the current running disparity must be forced at some point after reset before the first SOF is sent to guarantee that all non-EOF ordered sets will be sent with negative disparity. The tx_force_disp and tx_disp_sel inputs, defined below, allow this control at the FPGA interface.*

**tx_force_disp_[0-3]** - Per channel, active high signal which instructs the flexiPCS to accept disparity value from the tx_disp_sel_[0-3] FPGA interface input.

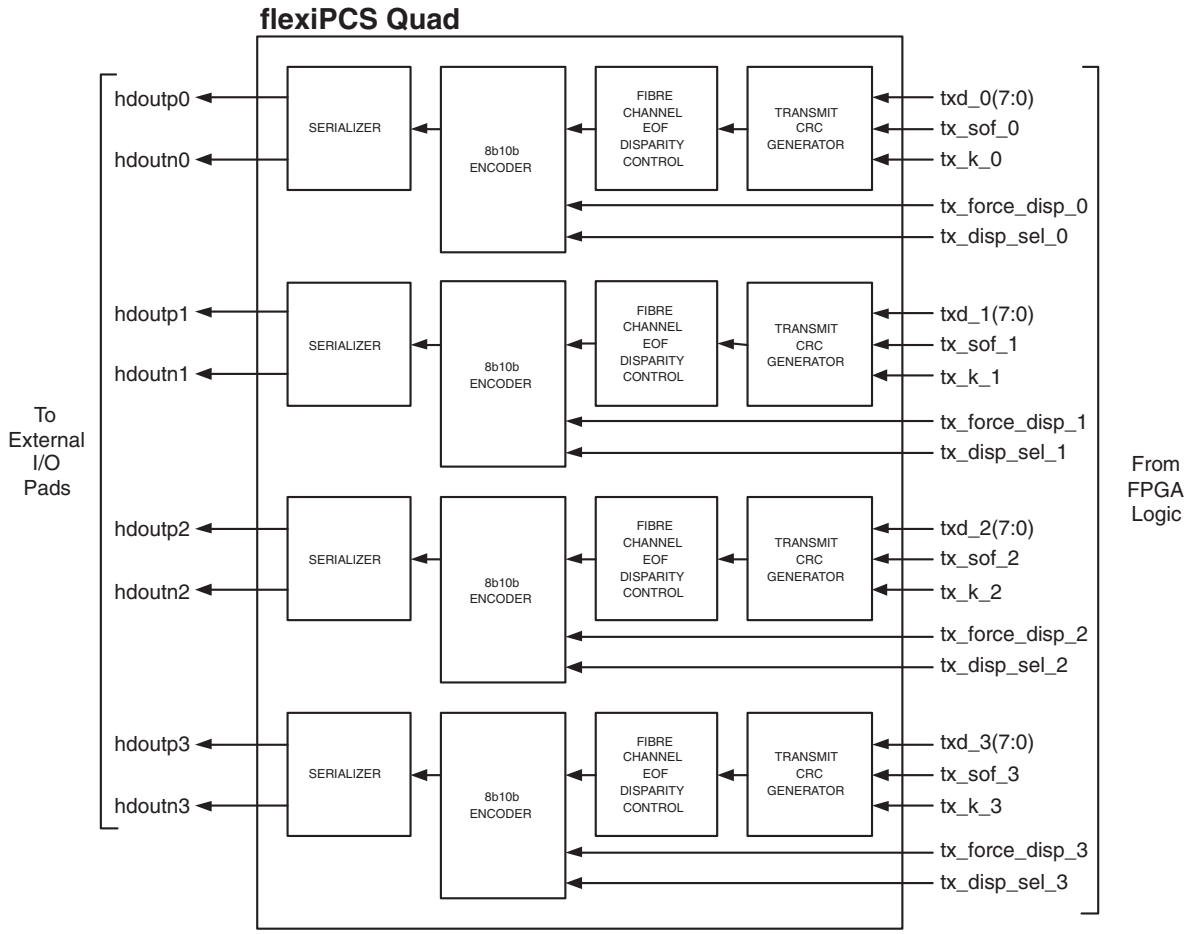
In 16 Bit Data Bus Mode, tx_force_disp is 2 bits wide (**tx_force_disp_[0-3](1:0)**) with tx_force_disp_[0-3](1) associated with txd_[0-3](15:8) and tx_force_disp_[0-3](0) associated with txd_[0-3](7:0).

**tx_disp_sel_[0-3]** - Per channel, disparity value supplied from FPGA logic. It is valid when tx_force_disp_[0-3] is high.

In 16 Bit Data Bus Mode, tx_disp_sel is 2 bits wide (**tx_disp_sel_[0-3](1:0)**) with tx_disp_sel_[0-3](1) associated with txd_[0-3](15:8) and tx_disp_sel_[0-3](0) associated with txd_[0-3](7:0).

The flexiPCS quad in Fibre Channel mode transmit data path consists of the following sub-blocks per channel: Transmit CRC Generator, Fibre Channel EOF Disparity Control, 8b10b Encoder, and Serializer. Figure 6-5 shows the four channels of transmit data paths in a flexiPCS quad.

*Figure 6-5. Fibre Channel Transmit Path (1 Quad)*




**CRC Generation**

A separate Cyclic Redundancy Code (CRC) generator is provided for all four transmit channels in a flexiPCS quad. The CRC generator supports the following Fibre Channel polynomial:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

The CRC generator options are set on a per quad basis. The CRC generator options can be set for Fibre Channel compliant operation by writing Quad Interface Register Offset Address 0x1D to 0x60 (writing 0x1D to 0x00 disables the CRC generator/checker).

The following signals driven by the FPGA logic are used to control the transmit CRC logic on a per channel basis:

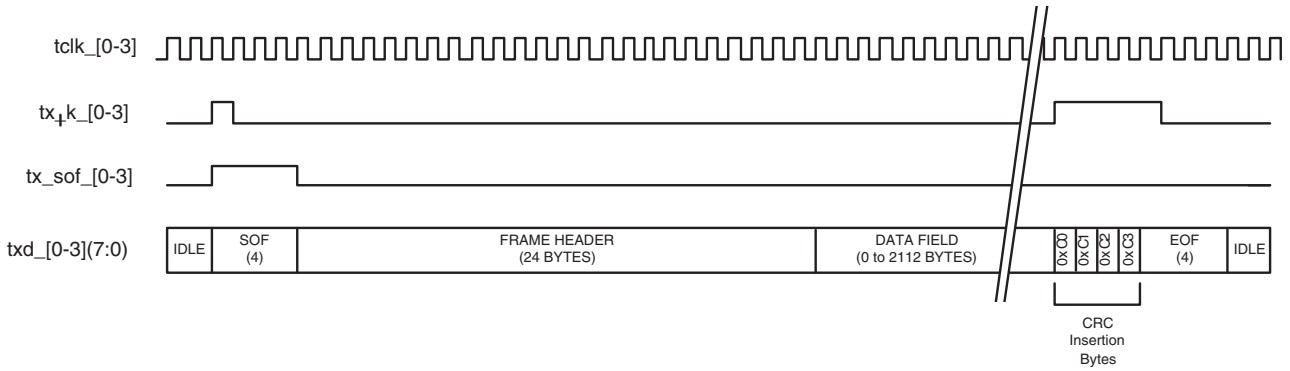
**tx_sof_[0-3]** - Per channel active high Start of Frame indicator which is used for CRC initialization. When driven to '1', transmit CRC logic is re-initialized. When driven to '0', CRC is computed on incoming txd_[0-3](7:0) data.

In 16 Bit Data Bus Mode, tx_sof is 2 bits wide (**tx_sof_[0-3](1:0)**) with tx_sof_[0-3](1) associated with txd_[0-3](15:8) and tx_sof_[0-3](0) associated with txd_[0-3](7:0).

The tx_sof_[0-3] signal should remain low during the CRC calculation and insertion as shown in Figure 6-6.

At the point where the CRC is to be inserted, a K-C0, K-C1, K-C2, or K-C3 character should be placed onto the data bus (K=1, Data = 0xC0, 0xC1, 0xC2, 0xC3 are not valid codes in the 8b10b coding space). Bits 0-7 of the CRC will be substituted for 0xC0; bits 8-15 of the CRC will be substituted for 0xC1; bits 16-23 of the CRC will be substituted for 0xC2; and bits 24-31 of the CRC will be substituted for 0xC3.

*Figure 6-6. Fibre Channel Transmit CRC Insertion*




**EOF Disparity Control**

The Fibre Channel EOF Disparity Control monitors the disparity of the EOF ordered sets as sent to the txd and tx_k inputs of the flexiPCS. The EOF Disparity Control ensures that the transmitted EOF is one that results in negative current running disparity after processing of the final (rightmost) character of the EOF ordered set as required by the Fibre Channel standard.

**8b10b Encoding**

This module implements an 8b10b encoder as described within the 802.3-2002 1000BASE-X specification. The encoder performs the 8-bit to 10-bit code conversion as described the specification, along with maintaining the running disparity rules as specified.

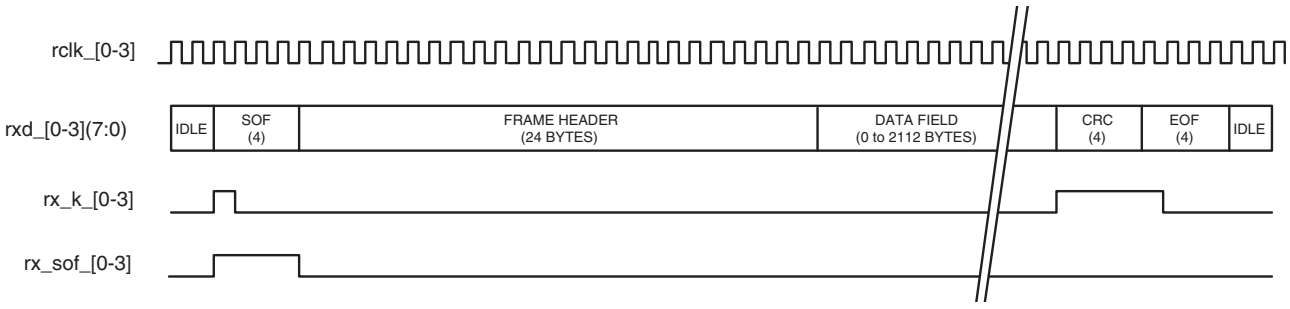
**Serializer**

The 8b10b encoded data undergoes parallel to serial conversion and is transmitted off chip via the embedded SERDES. For detailed information on the operation of the LatticeSC SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

## Receive Data

When configured into Fibre Channel mode, a flexiPCS quad provides four receive data paths from a high-speed serial line interface at the device inputs to an 8-bit parallel interface at the FPGA logic interface as shown in Figure 6-7.

*Figure 6-7. Fibre Channel Receive FPGA Interface Signals*




**rxd_[0-3](7:0)** - Per channel receive data to the FPGA interface. Each quad supports up to 4 independent channels of 8-bit wide parallel data by default. The rxd_[0-3] signal transitions synchronously with respect to rclk_[0-3].

In 16-Bit Data Bus Mode, this bus is 16-bits wide (**rxd_[0-3](15:0)**).

**rx_k_[0-3]** - Per channel, active high control character indicator.

In 16 Bit Data Bus Mode, rx_k is 2 bits wide (**rx_k_[0-3](1:0)**) with rx_k_[0-3](1) associated with rxd_[0-3](15:8) and rx_k_[0-3](0) associated with rxd_[0-3](7:0).

**rx_disp_err_detect_[0-3]** - Per channel, active high signal driven by the flexiPCS to indicate a disparity error was detected with the associated data.

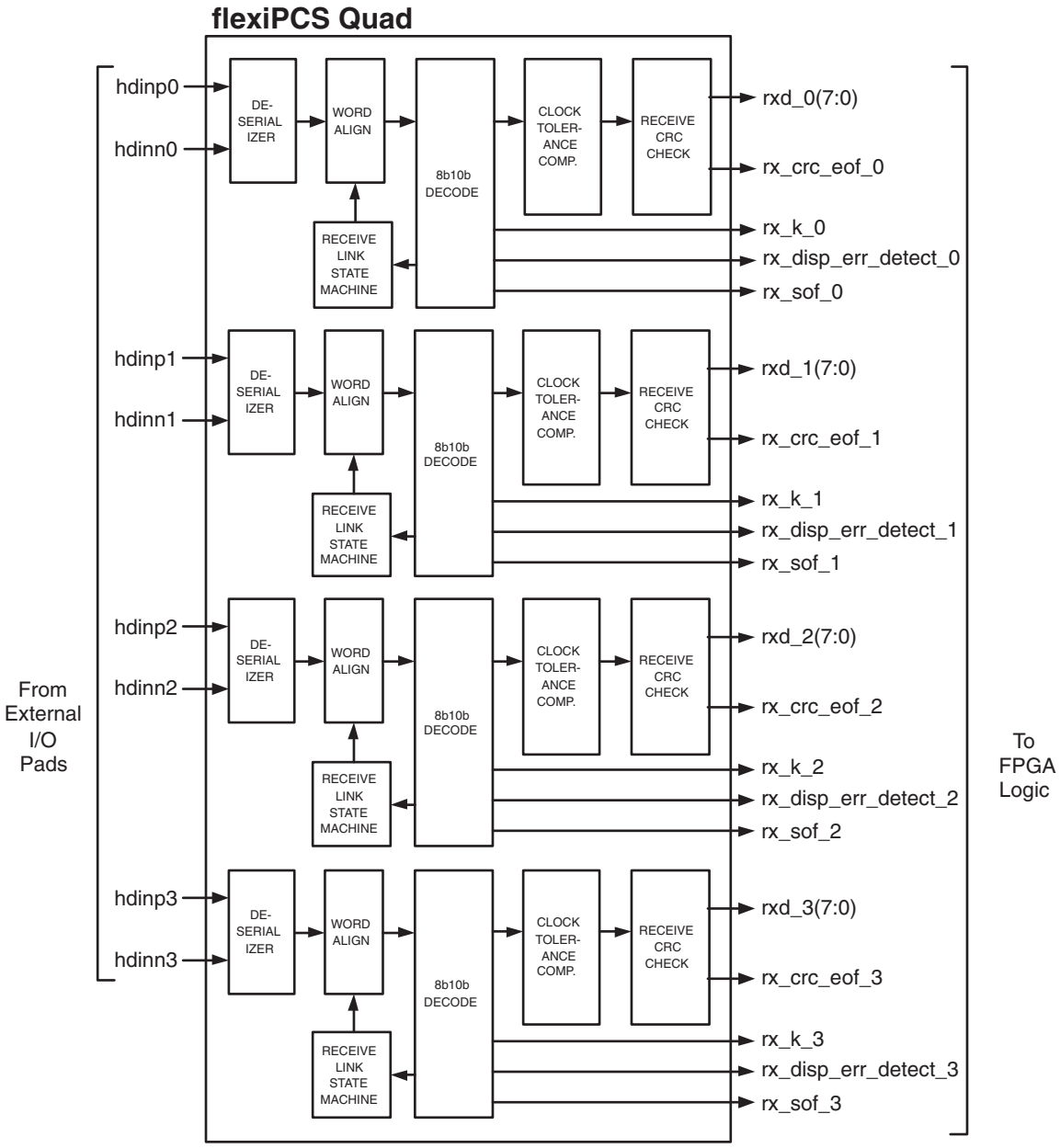
In 16 Bit Data Bus Mode, rx_disp_err_detect is 2 bits wide (**rx_disp_err_detect_[0-3](1:0)**) with rx_disp_err_detect_[0-3](1) associated with rxd_[0-3](15:8) and rx_disp_err_detect_[0-3](0) associated with rxd_[0-3](7:0).

**rx_sof_[0-3]** - Per channel, active high signal driven by the flexiPCS to indicate a Start-of-Frame was detected for the associated channel. This will report a Start-of-Frame when the appropriate Channel Interface Register Offset Address 0x00, bit 4 = '1'. This port will report 8b10b code violations otherwise.

In 16 Bit Data Bus Mode, rx_sof is 2 bits wide (**rx_sof_[0-3](1:0)**) with rx_sof_[0-3](1) associated with rxd_[0-3](15:8) and rx_sof_[0-3](0) associated with rxd_[0-3](7:0).

The flexiPCS quad in Fibre Channel mode receive data path consists of the following sub-blocks per channel: Deserializer, Word Aligner, 8b10b Decoder, Link State Machine, Clock Tolerance Compensator, and Receive CRC Checker. Figure 6-8 shows the four channels of receive data paths in a flexiPCS quad in Fibre Channel mode.

*Figure 6-8. Fibre Channel Receive Data Path (1 Quad)*




**Deserializer**

Data is brought on chip to the embedded SERDES where it undergoes serial to parallel conversion. For detailed information on the operation of the LatticeSC SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

**Word Alignment**

The word aligner module performs the word alignment code word detection and alignment operation. The word alignment character is used by the receive logic to perform 10-bit word alignment upon the incoming data stream. The word alignment algorithm is described in Clause 36.2.4.9 in the 802.3-2002 1000BASE-X specification.

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

- Ability to set two programmable word alignment characters (typically one for positive and one for negative dispar-

ity) and a programmable per bit mask register for word alignment compare. Word alignment characters and the mask register is set on a per quad basis. For Fibre Channel, the word alignment characters should 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).

- The first word alignment character can be set by writing Quad Interface Register Offset Address 0x15 to 0x03.
- The second word alignment character can be set by writing Quad Interface Register Offset Address 0x16 to 0x7C.
- The mask register can be set to compare word alignment bits 6 to 0 by writing Quad Interface Register Offset Address 0x14 to 0x7F (a '1' in a bit of the mask register means check the corresponding bit in the word alignment character register).

**8b10b Decoder**

The 8b10b decoder implements an 8b10b decoder as described with the 802.3-2002 specification. The decoder performs the 10-bit to 8-bit code conversion along with verifying the running disparity.

When a code violation is detected in a given data channel, that channel's rxd data is set to 0xEE and rx_cv_detect goes to '1'.

When a disparity error is detected in a given data channel, the data is passed to that channel's rxd pins and rx_disp_err_detect goes to '1'.

When a data error is detected, that channel's rxd data is set to 0xEE.

**Link State Machine**

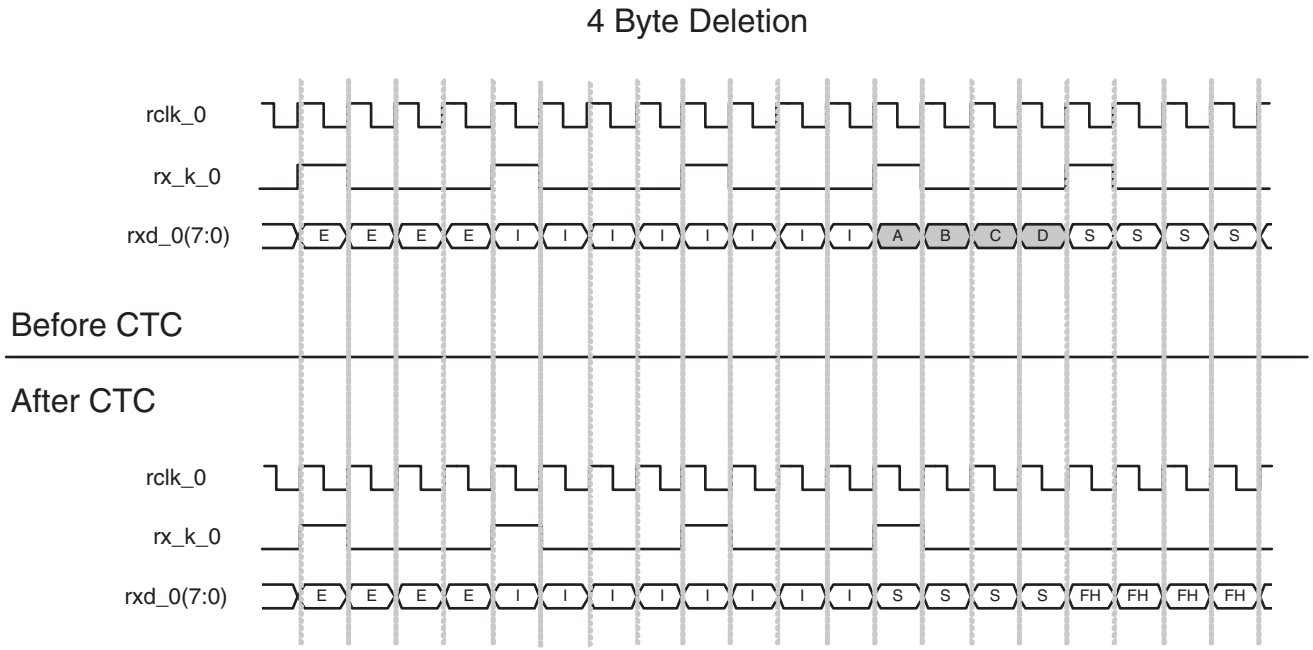
The link synchronization state machine is an extension of the word align module. Link synchronization is achieved after the successful detection and alignment of the required number of consecutive aligned code words. The link synchronization state machine implements a combination of the synchronization state diagrams shown in Figure 36-9 of the 802.3-2002 1000BASE-X specification.

**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 LatticeSC SERDES (See DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet).

A diagram illustrating 4 byte deletion is shown in Figure 6-9:

***Figure 6-9. Fibre Channel Mode Clock Compensation Byte Deletion Example***




E = End-of-frame

I = Idle

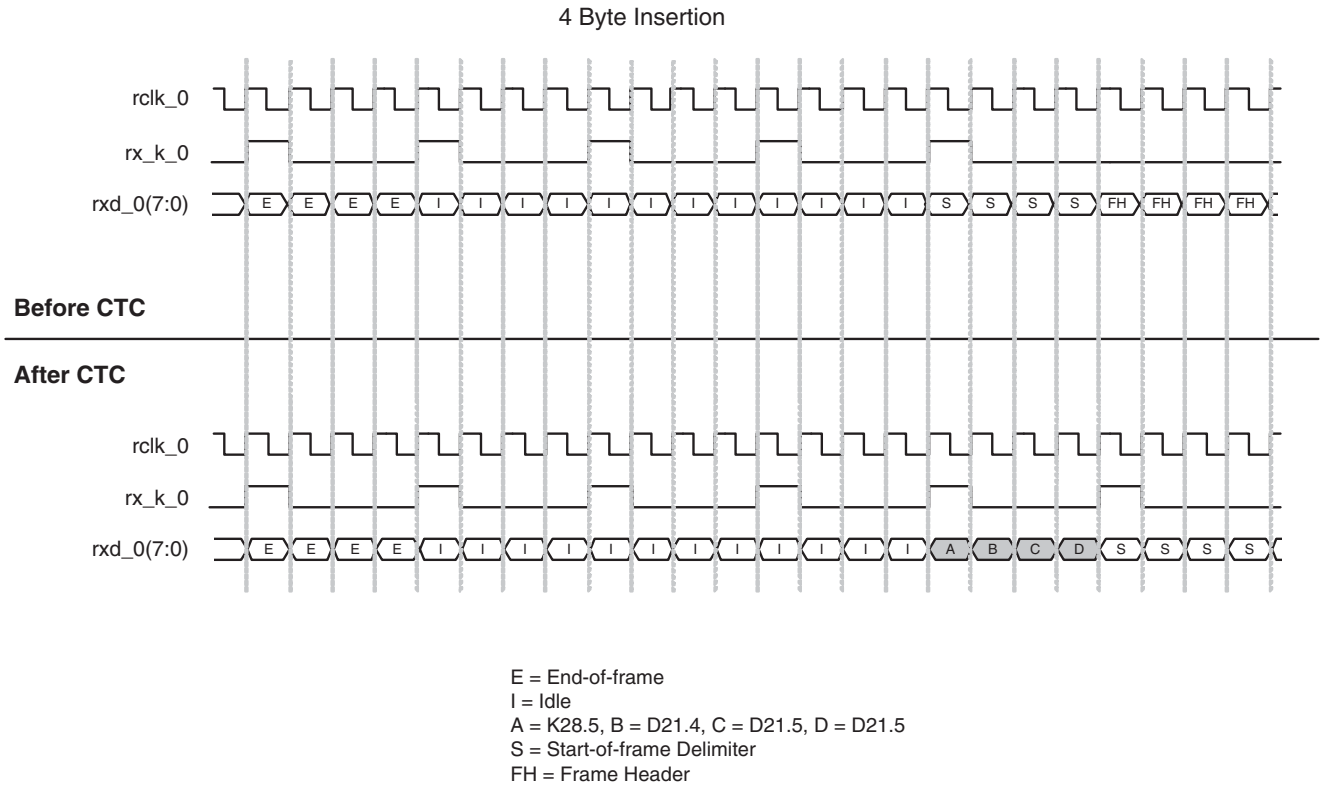
A = K28.5, B = D21.5, C = D21.4, D = D21.5

S = Start-of-frame Delimiter

FH = Frame Header

A diagram illustrating 4 byte insertion is shown in Figure 6-10:

***Figure 6-10. Fibre Channel Mode Clock Compensation Byte Insertion Example***




Clock compensation values are set on a quad basis. The following settings for clock compensation should be set for Fibre Channel applications:

- The Fibre Channel insertion/deletion pattern should be set to the Idle ordered set. The Fibre Channel Idle ordered set is defined as /K28.5/D21.4/D21.5/D21.5/. To load /K28.5/ set Quad Interface Register Offset Address 0x0F to 0xBC and Quad Interface Register Offset Address 0x13 to 0x40. To load /D21.4/ set Quad Interface Register Offset Address 0x10 to 0x95. To load the third byte of the Idle ordered set (/D21.5/) set Quad Interface Register Offset Address 0x11 to 0xB5. To load the fourth byte of the Idle ordered set (/D21.5/) set Quad Interface Register Offset Address 0x12 to 0xB5.
- Set the insertion/deletion pattern length to 4 and minimum inter-packet gap to 8 bytes by setting Quad Interface Register Offset Address 0x0E to 0x05. The minimum allowed inter-packet gap after Idle deletion based on these register settings is described below.
- Set the clock compensation FIFO high water and low water marks at 10 and 6 respectively (appropriate for insertion/deletion pattern length of 4) by setting Quad Interface Register Offset Address 0x0D to 0xA6.
- Clock compensation FIFO underrun can be monitored by reading the appropriate Channel Interface Register Offset Address 0x85, bit 6. This can be also monitored on the flexiPCS interface pin to FPGA logic labeled ctc_urun_[0-3] if “Optional Direct Control & Status Register Access” is selected when generating the flexiPCS block with IPexpress.
- Clock compensation FIFO overrun can be monitored by reading the appropriate Channel Interface Register Offset Address 0x85, bit 7. This can be also monitored on the flexiPCS interface pin to FPGA logic labeled ctc_orun_[0-3] if “Optional Direct Control & Status Register Access” is selected when generating the flexiPCS block with IPexpress.

Table 6-5 shows the relationship between the user-defined values for inter-packet gap (written to Quad Interface Register Offset Address 0x0E, bits 6 and 7), and the guaranteed minimum number of bytes between packets after

an Idle deletion from the flexiPCS. 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 Fibre Channel, the number of bytes per insertion/deletion is 4 (Quad Interface Register Offset Address 0x0E, bit 5 is set to '1'). If Quad Interface Register Offset Address 0x0E, bits 6 and 7 are set to "01", then the minimum inter-packet gap is equal to 2 times 4 or 8 bytes. The flexiPCS will not perform an Idle deletion until the minimum number of inter-packet bytes have been detected.

***Table 6-5. Minimum inter-packet gap multiplier***

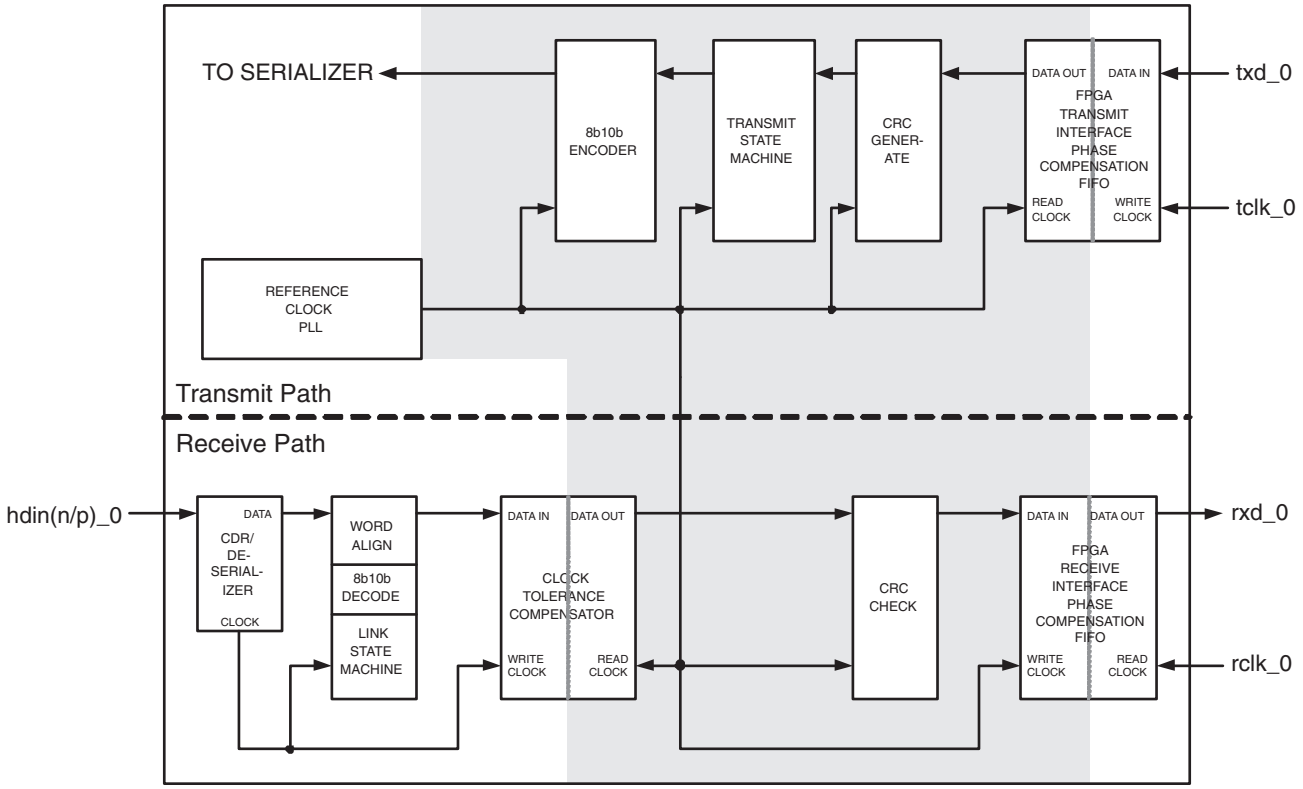
| Quad Interface Register Offset Address 0x0E | | Insertion/Deletion Multiplier Factor |
|---|---|---|
| **Bit 6** | **Bit 7** | |
| 0 | 0 | 1 X |
| 0 | 1 | 2 X |
| 1 | 0 | 3 X |
| 1 | 1 | 4 X |

Figure 6-11 shows the clock domains for both transmit and receive directions for a single channel inside the flexiPCS.

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

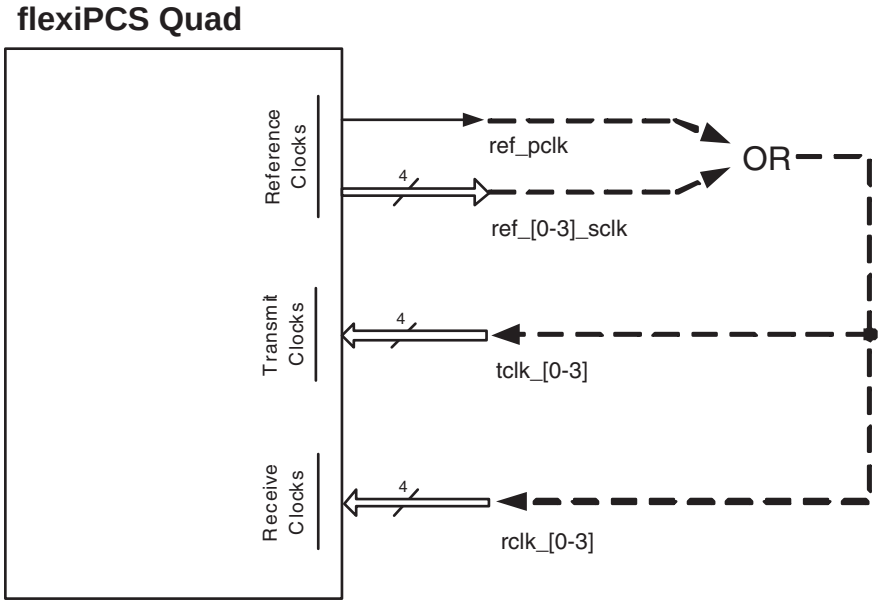
On the receive side, a clock domain transfer from the recovered channel clock to the locked reference clock occurs at the Clock Tolerance Compensator. A second clock domain transfer occurs from the locked reference clock to the rclk input at the FPGA interface at the FPGA Receive Interface phase compensation FIFO. The FPGA Receive Interface phase compensation FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. These phase compensation FIFOs (one per channel) cannot compensate for frequency variations.

***Figure 6-11. flexiPCS Clock Domain Transfers for Fibre Channel Mode***




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 flexiPCS quad set to Fibre Channel mode. Figure 6-12 illustrates the possible connections that would result in a synchronous interface.

***Figure 6-12. Synchronous input clocks to flexiPCS quad set to Fibre Channel Mode***



**CRC Checker**

The Fibre Channel CRC checker automatically checks for any Fibre Channel Start-of-Packet ordered sets and begins a CRC calculation immediately after the Start-of-Packet ordered set. The CRC checker also automatically checks for any Fibre Channel End-of-Packet ordered sets and compares the CRC values calculated on the receive data with the value embedded in the data stream just prior to the End-of-Packet ordered set. Table 6-6 shows the ordered sets defined by the CRC checker for Fibre Channel Start-of-Packet and End-of-Packet. Figure 6-13 shows the timing relationships of the CRC checker outputs to the FPGA interface.

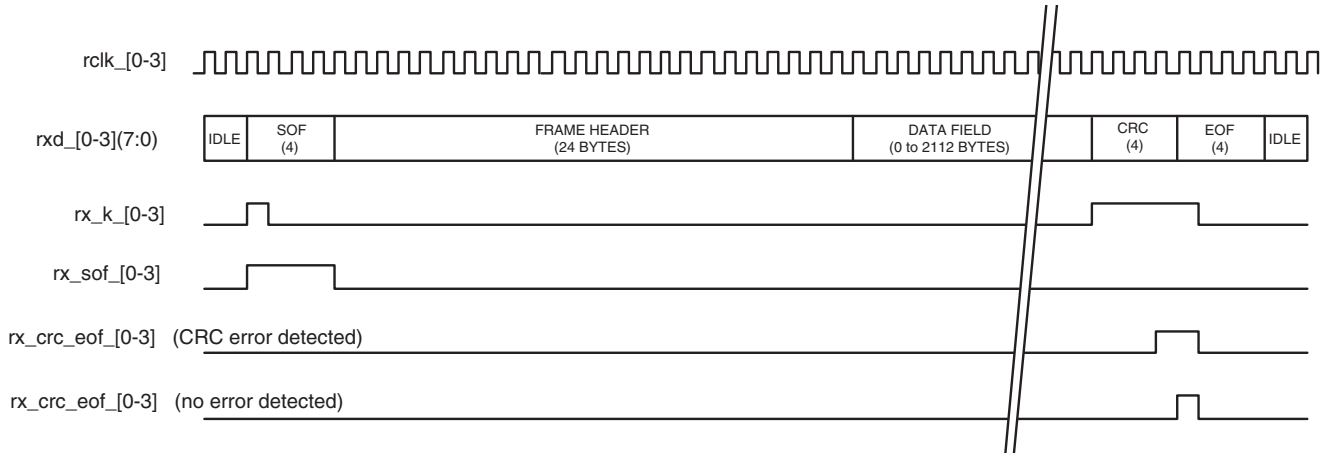
***Table 6-6. CRC Checker Start-of-Packet and End-of-Packet Ordered Sets in Fibre Channel Mode***

| Start-of-Packet Ordered Sets | End-of-Packet Ordered Sets |
|---|---|
| /K28.5/D21.5/D23.0/D23.0/ | /K28.5/D21.4/D21.3/D21.3/ |
| /K28.5/D21.5/D23.2/D23.2/ | /K28.5/D21.4/D21.4/D21.4/ |
| /K28.5/D21.5/D23.1/D23.1/ | /K28.5/D21.4/D21.6/D21.6/ |
| /K28.5/D21.5/D21.2/D21.2/ | /K28.5/D21.4/D21.7/D21.7/ |
| /K28.5/D21.5/D21.1/D21.1/ | /K28.5/D21.4/D25.4/D25.4/ |
| /K28.5/D21.5/D22.2/D22.2/ | /K28.5/D21.5/D21.3/D21.3/ |
| /K28.5/D21.5/D22.1/D22.1/ | /K28.5/D21.5/D21.4/D21.4/ |
| /K28.5/D21.5/D25.0/D25.0/ | /K28.5/D21.5/D21.6/D21.6/ |
| /K28.5/D21.5/D25.2/D25.2/ | /K28.5/D21.5/D21.7/D21.7/ |
| /K28.5/D21.5/D25.1/D25.1/ | /K28.5/D21.5/D25.4/D25.4/ |
| /K28.5/D21.5/D24.2/D24.2/ | /K28.5/D10.4/D21.4/D21.4/ |
| | /K28.5/D10.4/D21.6/D21.6/ |
| | /K28.5/D10.4/D25.4/D25.4/ |
| | /K28.5/D10.5/D21.4/D21.4/ |
| | /K28.5/D10.5/D21.6/D21.6/ |
| | /K28.5/D10.5/D25.4/D25.4/ |

**rx_crc_eof_[0-3]** - Per channel active high signal which indicates whether a CRC error was detected. When end of packet is reached, calculated CRC is compared with the expected values. If there is no error, rx_crc_eof_[0-3] acts as an end of packet indicator and goes high for one rclk_[0-3] cycle. If the packet was in error, then rx_crc_sof_[0-3] is asserted for two clock cycles as shown in Figure 6-13.

In 16 Bit Data Bus Mode, rx_crc_eof is 2 bits wide (**rx_crc_eof_[0-3](1:0)**) with rx_crc_eof_[0-3](1) associated with rxd_[0-3](15:8) and rx_crc_eof_[0-3](0) associated with rxd_[0-3](7:0).

*Figure 6-13. Fibre Channel Receive CRC Error Detection*




The CRC checker options are set on a per quad basis. The CRC checker options can be set for Fibre Channel compliant operation by writing Quad Interface Register Offset Address 0x1E to 0x00 (writing 0x1D to 0x00 disables the CRC generator/checker).

## Control & Status

The Control & Status pins for the Fibre Channel mode can be optionally selected on a per quad basis when generating the flexiPCS quad interface files with the ispLEVER tools. Each of these control and status functions are also accessible through equivalent Quad Interface and Channel Interface Registers. The control and status signals allow direct real time access of these functions from FPGA logic.

### Control

**felb_[0-3]** - Per channel, active high control signal which sets up the appropriate channel in Far-End Loopback mode. This mode is generally used for testing the flexiPCS logic, looping back receive data back to the transmit direction without crossing the FPGA logic interface. For more information on the details of Far-End Loopback mode, see the **flexiPCS Testing** Section of the flexiPCS Data Sheet. The Far-End Loopback mode can also be initiated by writing Channel Interface Register Offset Address 0x00, bit 5 to '1'.

**lsm_en_[0-3]** - Per channel Receive Link State Machine Enable. A change (either 0 to 1 or 1 to 0) on the lsm_en_[0-3] resets the Receive Link State Machine into No Sync mode. The Receive Link Machine Enable can also be set by writing Channel Interface Register Offset Address 0x00, bit 7 to '1'.

### Status

**lsm_status_[0-3]** - Per channel active high signal from the Receive Link State Machine indicating link synchronization successful. The link synchronization status can also be read from the appropriate Quad Interface Register Offset Address 0x84, bits [4:7] (bit 4 for channel 0, bit 5 for channel 1, bit 4 for channel 2, and bit 7 for channel 3).

**ctc_urun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has underrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 6.

**ctc_orun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has overrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 7.

## Status Interrupt Registers

Some of the status registers associated with Fibre Channel operation have an associated status interrupt and status interrupt enable register. Any flexiPCS status interrupt register will generate an interrupt to the Systembus block

(if used in the design). For a description of the operation of interrupts with the Systembus, refer to the Systembus section of the LatticeSC/M Family Data Sheet. Operation of the interrupt registers for the flexiPCS is as follows:

User must set the appropriate status interrupt enable bit to a '1'

Whenever the associated status bit goes high, its status interrupt bit will also go high. The status interrupt bit will remain high even if the associated status bit goes low.

The status interrupt bit will remain high until it is read, when it will automatically be cleared. If the associated status bit is still high, then the status interrupt bit will go high again (until read the next time).

Table 6-7 shows a listing of the status registers associated with Fibre Channel operation which have a corresponding status interrupt and status interrupt enable bit.

***Table 6-7. Status Interrupt Register Bits***

| Status Register Bit Function | Status Register Bit Address | Status Interrupt Enable Register Bit Address | Status Interrupt Register Bit Address |
|---|---|---|---|
| Receive Link State Machine Status (lsm_status) Channel 0 | QIR 0x84, bit 4 | QIR 0x1C, bit 4 | QIR 0x85, bit 4 |
| Receive Link State Machine Status (lsm_status) Channel 1 | QIR 0x84, bit 5 | QIR 0x1C, bit 5 | QIR 0x85, bit 5 |
| Receive Link State Machine Status (lsm_status) Channel 2 | QIR 0x84, bit 6 | QIR 0x1C, bit 6 | QIR 0x85, bit 6 |
| Receive Link State Machine Status (lsm_status) Channel 3 | QIR 0x84, bit 7 | QIR 0x1C, bit 7 | QIR 0x85, bit 7 |
| Clock Tolerance Compensation FIFO underrun | CIR 0x85, bit 6 | CIR 0x0A, bit 6 | CIR 0x8A, bit 6 |
| Clock Tolerance Compensation FIFO overrun | CIR 0x85, bit 7 | CIR 0x0A, bit 7 | CIR 0x8A, bit 7 |

# XAUI Mode
# LatticeSC/M flexiPCS Family Data Sheet

June 2011 Data Sheet DS1005

## Introduction

The XAUI mode of the flexiPCS (Physical Coding Sublayer) block supports full compatibility from Serial I/O to the XGMII interface of the IEEE 802.3-2002 XAUI standard. XAUI Mode supports 10 Gigabit Ethernet as well as 10 Gigabit Fibre Channel applications.

The LatticeSC flexiPCS in XAUI mode supports the following operations:

**Transmit Path (From LatticeSC device to line):**

- Transmit State Machine which performs translation of XGMII idles to proper ||A||, ||K||, ||R|| characters according to the IEEE 802.3ae-2002 specification
- 8b10b Encoding

**Receive Path (From line to LatticeSC device):**

- Word Alignment based on IEEE 802.3-2002 defined alignment characters.
- 8b10b Decoding
- Link State Machine functions incorporating operations defined in PCS Synchronization State Diagram of the IEEE 802.3ae-2002 specification.
- Clock Tolerance Compensation logic capable of accommodating clock domain differences.
- Receive State Machine compliant to the IEEE 802ae.3-2002 specification.

### LatticeSC flexiPCS Quad Module

Devices in the LatticeSC family have up to 8 quads of embedded flexiPCS logic. Each quad in turn supports 4 independent full-duplex data channels. A single channel can support a fully compliant XAUI data link and each quad can support up to four such channels. Note that mode selection is done on a per quad basis. Therefore, a selection of XAUI mode for a quad dedicates all four channels in that quad to XAUI mode.

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

Operation of the SERDES requires the user to provide a reference clock (or clocks) to each active quad to provide a synchronization reference for the SERDES PLLs. Reference clocks are shared among all four channels of a quad. If the transmit and receive frequencies are within the specified ppm tolerance for the LatticeSC SERDES (See DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet), only one reference clock for both transmit and receive is needed for both transmit and receive directions. If the receive frequency is not within the specified ppm tolerance for the LatticeSC SERDES (See the DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet) of the transmit frequency, then separate reference clocks for the transmit and receive directions must be provided.

Simultaneous support of different (non-synchronous) high-speed data rates is possible with the use of multiple quads. Multiple frequencies that are exact integer multiples can be supported on a per channel basis through use of the full-rate/half-rate mode options (see the **SERDES Functionality section** for more details).

## Quad and Channel Option Control

Although the mode selection and reference frequency covers an entire quad, many options covering clocking and data formatting are available on a per channel basis. These options are detailed in the following XAUI Mode

© 2011 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

description. Some of these options can be controlled directly through the FPGA interface pins made available on the XAUI Quad Module.

Other options are available only through dedicated registers that can be written and read via the embedded System Bus Interface (see the System Bus section for more details on the operation of the System Bus), or during device configuration. Depending on the specific option, a register may affect operation of multiple quads, a single quad or an individual channel in a quad. Registers dedicated to multiple quads are listed in the Inter-quad Interface Register Map in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet. Registers dedicated to one quad are listed in the Quad Interface Register Map. Registers dedicated to a single channel are listed in the Channel Interface Register Map.

## Register Map Addressing

Figure 7-1 provides a map of base register addresses for any of the flexiPCS quads on a LatticeSC device. Note that different devices may have different numbers of available quads up to a total of 8 quads (or 32 channels) maximum.

Inter-quad Interface, Quad Interface, and Channel Interface Registers are listed by Offset Address. Each Inter-quad Interface Register, Quad Interface Register, and Channel Interface Register has a unique 18-bit address determined by the specific quad or channel targeted. The addressing of Inter-quad Interface Registers, Quad Interface Registers, and Channel Interface Registers is shown Figure 7-1.

Base addresses are dependant on the physical location of a given quad or channel on a LatticeSC device. Each quad and channel has an associated set of control and status registers. These registers are referred throughout this data sheet by their offset addresses. The unique 18-bit address of a control or status register for a specific quad or channel is simply the offset address of that register added to the base address for that specific quad or channel as shown in Figure 7-1.

Channel Interface Registers can be written in one of three methods. One method is to write to one specific register corresponding to one specific channel. The other two methods involve writing to multiple channel registers with just one write operation. This is referred to as a Broadcast write. A Broadcast write is useful when multiple channel registers are to be written with the same value. The first method of Broadcast writing involves writing to all four channel registers in a quad at one time. A second method of Broadcast writing involves writing to all channel registers for all quads on the left or right side of the device at one time. Therefore, a specific Channel Interface Register may be accessed through any one of three channel addresses, depending on the number of channel registers being written to at any one time.

A broadcast write to multiple quads cannot be used to power on SERDES in any device-package combination that does not have all SERDES quads bonded because of excess power on the board and jitter is also affected.

Throughout this document, the byte-wide data at each offset address is listed with a hexadecimal value with bit 0 as the MSb and bit 7 as the LSb as per the PowerPC convention. For example if the value for Quad Interface Register Offset Address 0x02 is stated to be 0x15, the bit pattern defined is as shown in Table 7-1:

***Table 7-1. Control/Status Register Bit Order Example***

| **Quad Interface Register Offset Address 0x02 = 0x15** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Data Bit 0 | Data Bit1 | Data Bit 2 | Data Bit 3 | Data Bit 4 | Data Bit 5 | Data Bit 6 | Data Bit 7 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| **1** | | | | **5** | | | |

A full list of register Offset Addresses is provided in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet.

***Figure 7-1. flexiPCS Inter-quad, Quad, and Channel Interface Register Map Base Addressing***

## Auto-Configuration

Initial register setup for each flexiPCS mode can be performed without accessing the system bus by using the auto-configuration feature in ispLEVER. IPexpress generates 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 chan-

nel registers will be set to values defined in the auto-configuration file during configuration. The system bus is therefore not needed if all quads are to be set via auto-configuration files. However, the system bus must be included in a design if the user needs to change control registers or monitor status registers during operation. Note that a quad reset (Quad Interface Register 0x43, bit 7 or the **quad_rst** port at the FPGA interface) will reset all registers back to their default (not auto-configuration) values. If a quad reset is issued, the flexiPCS registers need to be rewritten via the Systembus.

## XAUI Register Settings

The auto-configuration file sets registers that perform the following operations. If the auto-configuration file is not used, the appropriate registers need to be programmed via the Systembus. For more options, refer to the flexiPCS register map in the **Memory Map** section of the **LatticeSC/M Family flexiPCS Data Sheet**.

**A flexiPCS quad can be set to XAUI mode by writing Quad Interface Register Offset Address 0x18 bits[0:7] to 0x01.**

This register value automatically sets up the following options in the flexiPCS for the given quad. Details on the functionality of each chosen option is provided in the **XAUI Mode Detailed Description** section.

**Transmit Path**

- The four internal flexiPCS Tx clocks (one per channel) are synchronized to align the four data lanes to the same clock edge
- Transmit State Machine is set to XAUI Mode
- 8b10b encoder is enabled

**Receive Path**

- Word aligner is enabled
- 8b10b decoder is enabled
- Receive Link State Machine is set to XAUI Mode
- Receive State Machine is set to XAUI Mode

Other register settings are required to operate a channel or channels in XAUI Mode. The following is a list of options that must be set for XAUI operation. More detail for each option is included in the **XAUI Mode Detailed Description** section.

- Powerup of appropriate quad and channel. Quad powerup is chosen by writing the appropriate Quad Interface Register Offset Address 0x28, bit 1 to '1'. Channel powerup is chosen by writing the appropriate Channel Interface Register Offset Address 0x13, bit 6 (receive direction) and/or bit 7 (transmit direction) to '1'. By default, all channels and quads are powered down.
- Receive Link State Machine enable. By default, all channel Receive Link State Machines are disabled. The Receive Link State Machine for a given channel can be enabled by writing the appropriate Channel Interface Register Offset Address 0x00, bit 7 to '1'.
- Setting SERDES reset low at end of flexiPCS setup. By default, the SERDES reset is set to '1' (SERDES are reset). The SERDES reset can be set low by writing Quad Interface Register Offset Address 0x41, bit 5 to a '0'. For more information on proper flexiPCS powerup and reset sequencing, refer to the **flexiPCS Reset Sequences** and **flexiPCS Power Down Control Signals** descriptions in the LatticeSC/M Family flexiPCS Data Sheet **SERDES Functionality** section.
- Word alignment character(s), such as a comma
- Clock Tolerance Compensation insertion/deletion ordered set values and FIFO settings
- Multi-channel settings including user-defined alignment characters and FIFO settings

## XAUI Auto-Configuration Example

An example of an auto-configuration file for a quad set to XAUI Mode with Multi-channel Alignment and Clock Tolerance Compensation (CTC) enabled and with all four channels enabled is shown in Figure 7-2. Format for the auto-configuration file is

register type (quad=quad register, ch1=channel 1 register), offset address in hex, register value in hex, # comment

***Figure 7-2. XAUI Auto-Configuration Example File***

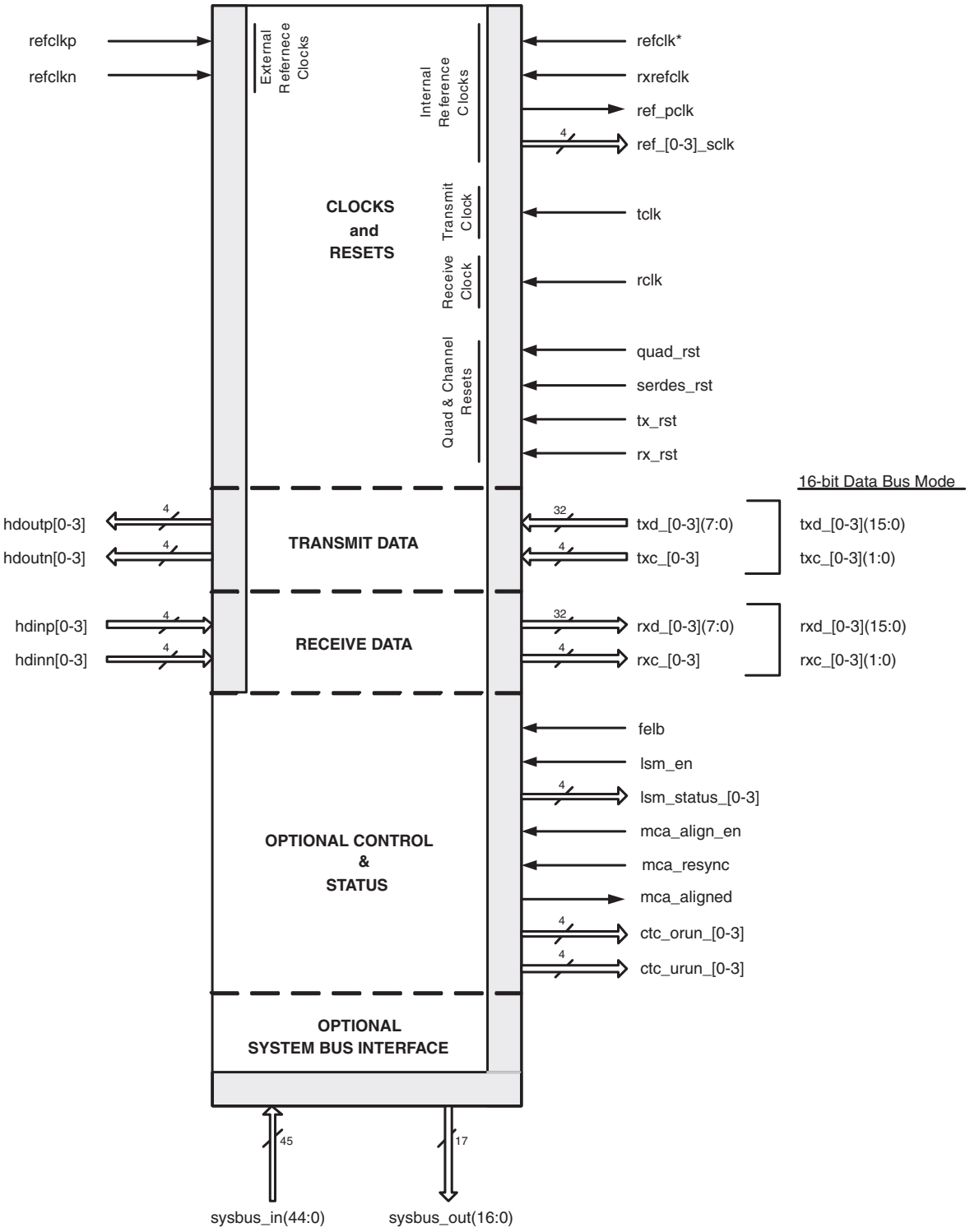
```
quad 18 01     #  Set quad to XAUI Mode
quad 30 04     #  Set Tx Synchronization bit
quad 29 01     #  Set reference clock select to refclk(p/n) inputs
quad 28 40     #  Set bit clock multiplier to 20X (for full rate mode), quad powerup set to '1'
quad 02 00     #  Set ref_pclk to channel 0 clock
quad 14 7F     #  Set word alignment mask to compare lowest 7 LSBs
quad 15 03     #  Set positive disparity comma value
quad 16 7C     #  Set negative disparity comma value
quad 19 40     #  Select 4 channel alignment, set FPGA interface data bus width to 8-bit
quad 04 0F     #  Set alignment enable for all four channels
quad 01 00     # Set all multi-channel alignment clocks to be sourced from
# the channel 0 recovered receive clock
quad 07 FF     #  Set multi-channel align character mask to compare bits [7:0]
quad 08 7C     #  Set multi-channel align character “A”, bits [7:0] to 7C (K28.3)
quad 09 7C     #  Set multi-channel align character “B”, bits [7:0] to 7C (K28.3)
quad 0A 15     #  Set multi-channel align character mask to compare bit 8
               #  Set multi-channel align characters “A” and “B” bit 8 (K character) to '1'
quad 05 00     #  Set multi-channel alignment FIFO latency to minimum
quad 06 05     #  Set multi-channel alignment FIFO high water mark to 5
quad 0E 00     #  Set insertion/deletion to 1 byte for CTC, set minimum inter-packet gap
quad 0D 97     #  Set CTC FIFO watermarks (high=9, low=7)
quad 12 1C     #  1st byte of ||R|| (skip) pattern for CTC (K28.0)
quad 13 01     #  K bit for ||R|| pattern
ch0 00 01      #  Turn on channel 0 link state machine
ch0 13 03      #  Powerup channel 0 transmit and receive directions, set rate mode to “full”
ch0 14 90      # 16% pre-emphasis on SERDES output buffer
ch0 15 10      # +6 dB equalization on SERDES input buffer
ch1 00 01      #  Turn on channel 1 link state machine
ch1 13 03      #  Powerup channel 1 transmit and receive directions, set rate mode to “full”
ch1 14 90      # 16% pre-emphasis on SERDES output buffer
ch1 15 10      # +6 dB equalization on SERDES input buffer
ch2 00 01      #  Turn on channel 2 link state machine
ch2 13 03      #  Powerup channel 2 transmit and receive directions, set rate mode to “full”
ch2 14 90      # 16% pre-emphasis on SERDES output buffer
ch2 15 10      # +6 dB equalization on SERDES input buffer
ch3 00 01      #  Turn on channel 3 link state machine
```

```
ch3 13 03       #  Powerup channel 3 transmit and receive directions, set rate mode to “full”
ch3 14 90       # 16% pre-emphasis on SERDES output buffer
ch3 15 10       # +6 dB equalization on SERDES input buffer

# These lines must appear last in the autoconfig file. These lines apply the

# correct reset sequence to the PCS block upon bitstream configuration

quad 41 00 # de-assert serdes_rst

quad 40 ff   # assert datapath reset for all channels

quad 41 03 # assert MCA reset

quad 41 00 # de-assert MCA reset

quad 40 00 # de-assert datapath reset for all channels
```

## XAUI Mode

IPexpress allows the designer to choose the mode for each flexiPCS quad used in a given design. Figure 7-3 displays the resulting port interface to one flexiPCS quad that has been set to XAUI mode.

***Figure 7-3. XAUI Mode Pin Diagram***




* The refclk inputs for all active quads on a device must be connected to the same clock . The rxrefclk inputs for all active quads on a device do not have to be connected to the same clock.

## XAUI Mode Pin Descriptions

Table 7-2 lists all inputs and outputs to/from a flexiPCS quad in XAUI mode. A brief description for each port is given in the table. A more detailed description of the function of each port is given in the **XAUI Mode Detailed Description**.

***Table 7-2. XAUI Mode Pin Description***

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Reference Clocks** | | | |
| refclkp | In from I/O pad | N/A | Reference clock input, positive. Dedicated CML input. |
| refclkn | In from I/O pad | N/A | Reference clock input, negative. Dedicated CML input |
| refclk | In from FPGA | N/A | Optional reference clock input from FPGA logic. Can be used instead of per quad reference clock.<br><br>The refclk inputs for all active quads on a device must be connected to the same clock |
| rxrefclk | In from FPGA | N/A | Optional receive reference clock input from FPGA logic. Can be used instead of per quad reference clock. The rxrefclk inputs for all active quads do not have to be connected to the same clock. |
| ref_pclk | Out to FPGA | N/A | Locked reference clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| ref_[0-3]_sclk | Out to FPGA | N/A | Per channel locked reference clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| **Transmit Clock** | | | |
| tclk | In from FPGA | N/A | Transmit clock input from FPGA. Used to clock the TX data phase compensation FIFO with clock synchronous to the reference clock. May also be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Receive Clock** | | | |
| rclk | In from FPGA | N/A | Receive clock input from FPGA. May be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference and/or receive reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Resets** | | | |
| quad_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of SERDES and PCS logic. |
| serdes_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of the SERDES. |
| tx_rst | In from FPGA | ASYNC | Active high, asynchronous reset of all four transmit channels of PCS logic. |
| rx_rst | In from FPGA | ASYNC | Active high, asynchronous reset of all four receive channels of PCS logic. |
| **Transmit Data** | | | |
| hdoutp[0-3] | Out to I/O pad | N/A | High-speed CML serial output, positive. |
| hdoutn[0-3] | Out to I/O pad | N/A | High-speed CML serial output, negative. |
| txd_[0-3](7:0)<br><br>txd_[0-3](15:0)* | In from FPGA | tclk_[0-3] | Per channel parallel transmit data bus from the FPGA. Bus is 8-bits wide if QIR 0x19, bit 4 = ‘0’.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 4 = ‘1’. |
| txc_[0-3]<br><br>txc_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel transmit control character indication.<br><br>*Bus is 2 bits wide if QIR 0x19, bit 4 = ‘1’. |
| **Receive Data** | | | |

*Table 7-2. XAUI Mode Pin Description*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| hdinp[0-3] | In from I/O pad | N/A | High-speed CML serial input, positive. |
| hdinn[0-3] | In from I/O pad | N/A | High-speed CML serial input, negative. |
| rxd_[0-3](7:0)<br><br>rxd_[0-3](15:0)* | Out to FPGA | rclk_0-3] | Per channel parallel receive data bus to the FPGA. Bus is 8-bits wide if QIR 0x19, bit 5 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 5 = '1'. |
| rxc_[0-3]<br><br>rxc_[0-3](1:0)* | Out to FPGA | rclk_0-3] | Per channel receive control character indication.<br><br>*Bus is 2 bits wide if QIR 0x19, bit 5 = '1'. |
| **Optional Control & Status** | | | |
| felb | In from FPGA | ASYNC | Active high far-end loopback enable for all four channels. |
| lsm_en | In from FPGA | ASYNC | Receive link state machine enable for all four channels. |
| lsm_status_[0-3] | Out to FPGA | ASYNC | Per channel signal from receive link state machine indicating successful link synchronization. |
| mca_align_en | In from FPGA | ASYNC | Active high multi-channel aligner enable for all four channels. |
| mca_resync | In from FPGA | ASYNC | Active high async multi-channel aligner resynchronization signal. |
| mca_aligned | Out to FPGA | ASYNC | Active high indicating successful alignment of channels. |
| ctc_urun_[0-3] | Out to FPGA | ASYNC | Per channel active high flag indicator that clock tolerance compensation FIFO has underrun. |
| ctc_orun_[0-3] | Out to FPGA | ASYNC | Per channel active high flog indicator that clock tolerance compensation FIFO has overrun. |
| **Optional System Bus Interface** | | | |
| sysbus_in(44:0) | In from FPGA | N/A | Control and data signals from the internal system bus. |
| sysbus_out(16:0) | Out to FPGA | N/A | Control and data signals to the internal system bus. |

# XAUI Mode Detailed Description

The following section provides a detailed description of the operation of a flexiPCS quad set to XAUI mode. The functional description is organized according to the port listing shown in Figure 7-3. The System Bus Offset Address for any control and status registers relevant to a given function are provided with the description of the operation of that function.

## Clocks & Resets

A detailed description of all SERDES clock, data, and reset ports and recommended reset sequencing is provided in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet. The following is a recommended setup of the SERDES for XAUI applications.

For XAUI applications, the bit clock rate is 3.125 Gbps. This falls into the range defined as "full rate" mode for the SERDES PLLs. Full rate mode is selected separately for each channel and each direction by writing the appropriate Channel Interface Register Offset Address 0x13, bit 5 (transmit) and bit 4 (receive) to a '0' (default).

The reference clock frequency is determined by the reference clock mode chosen. Table 7-3 shows the possible reference clock frequencies for various reference clock modes which result in a 3.125 Gbps internal bit rate.

*Table 7-3. XAUI Reference Clock Frequency Options*

| Full Rate Mode<br><br>Channel Interface Register Offset Address 0x13<br><br>Transmit - Bit 5 = '0'<br>Receive - Bit 4 = '0' | | | | |
|---|---|---|---|---|
| Reference Clock Mode<br><br>Quad Interface Register Offset Address = 0x28, Bits [2:3] | Reference Clock Frequency | Internal Serial (bit) Clock Frequency | FPGA Interface Clock Frequency<br><br>Quad Interface Register Offset Address 0x19 | |
| | | | 8 Bit Data Bus Mode<br><br>Transmit - Bit 4 = '0'<br>Receive - Bit 5 = '0' | 16 Bit Data Bus Mode<br><br>Transmit - Bit 4 = '1'<br>Receive - Bit 5 = '1' |
| 00 | 156.25 MHz | 3.125 GHz | 312.5 MHz | 156.25 MHz |
| 01 | 312.5 MHz | 3.125 GHz | 312.5 MHz | 156.25 MHz |
| 10 | 625 MHz | 3.125 GHz | 312.5 MHz | 156.25 MHz |

*Note: There are also frequency dependent SERDES performance control bits that are not writable via the Systembus. These control bits are set in the auto-configuration file generated within IPexpress and passed into the bitstream through the normal FPGA design flow (map, par, bitgen). To change the bit rate for a SERDES, specify the proper values for the affected flexiPCS quad in IPexpress and regenerate the auto-configuration file. Failure to do this may result in non-optimal SERDES operation.*

Additional information on all reference clock rate modes can be found in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

**Quad & Channel Resets**

Resets are provided to reset an entire quad (either SERDES logic only or all flexiPCS logic) or each individual channel (all flexiPCS logic per channel). These resets are driven from the FPGA logic. A summary of the control signals provided for reset are listed below. More detail on recommended initialization and reset sequences for the SERDES is provided in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

The following inputs to the flexiPCS from the FPGA are enabled as long as Quad Interface Register Offset Address 0x42, bit 7 is set to '1' (which is the default on powerup). Writing a '0' to this bit will disable these reset inputs.

**quad_rst** - Active high, asynchronous signal from FPGA resets all SERDES and PCS logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x43, bit 7 to '1'. quad_rst also resets all flexiPCS control registers. If these registers had been previously written via the Systembus or set during configuration through an auto configuration file, they will need to be rewritten following this reset.

**serdes_rst** - Active high, asynchronous signal from FPGA resets all SERDES (but not PCS) logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x41, bit 5 to '1'.Note that this bit is preset to '1' on powerup and must be written to '0' before operation of the SERDES can commence.

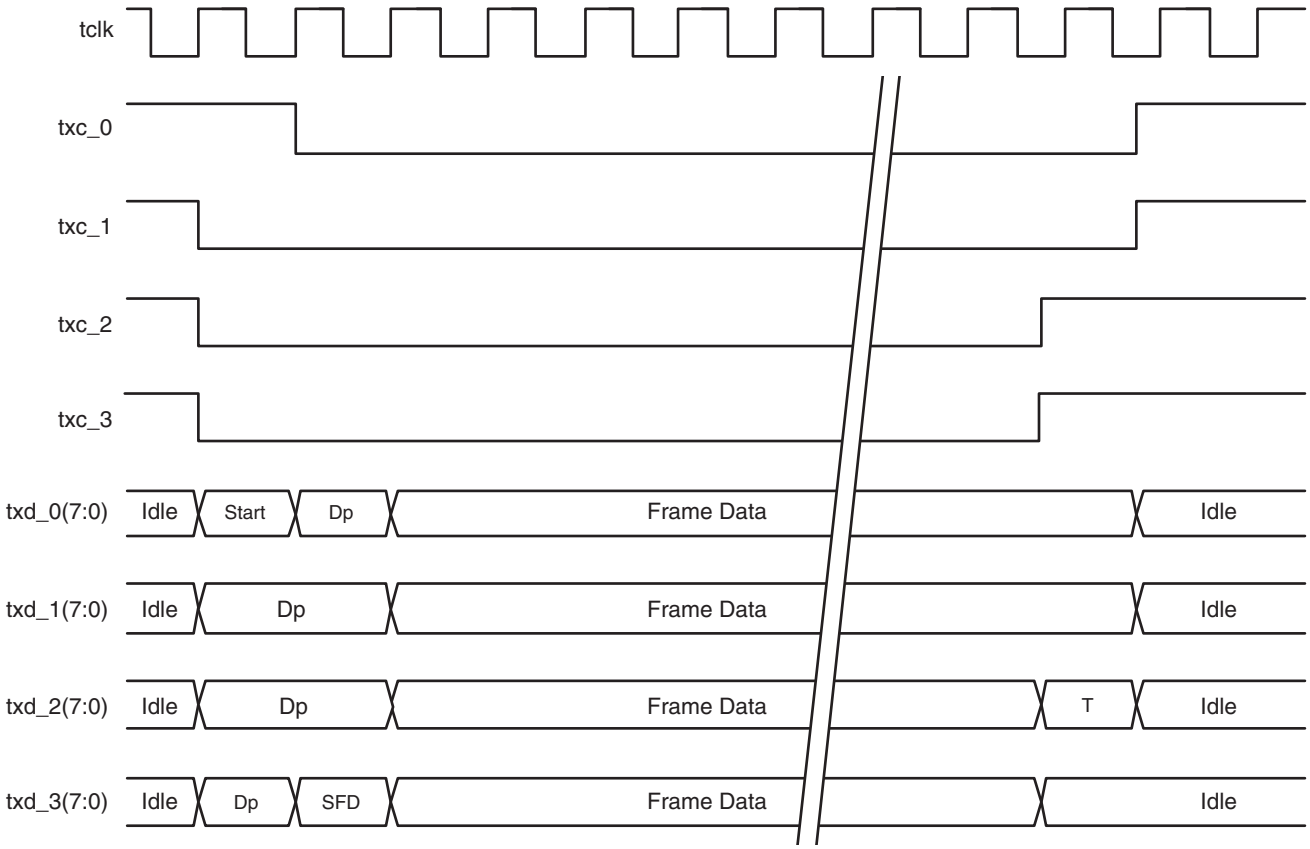
**tx_rst** - Active high, asynchronous signal from FPGA resets all four transmit channels of PCS logic. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [4:7]. Bit 7 resets transmit channel 0, bit 6 resets transmit channel 1, bit 5 resets transmit channel 2, and bit 4 resets transmit channel 3.

**rx_rst** - Active high, asynchronous signals from FPGA resets all four receive channels of PCS logic. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [0:3]. Bit 3 resets receive channel 0, bit 2 resets receive channel 1, bit 1 resets receive channel 2, and bit 0 resets receive channel 3.

## Transmit Data

When configured into XAUI mode, a flexiPCS quad provides four transmit data paths from an XGMII compliant 32-bit parallel interface at the FPGA logic interface to a high-speed serial line interface at the device outputs. Figure 7-4 shows the typical operation of this interface at the flexiPCS interface.

***Figure 7-4. XAUI Transmit XGMII Interface Signals***



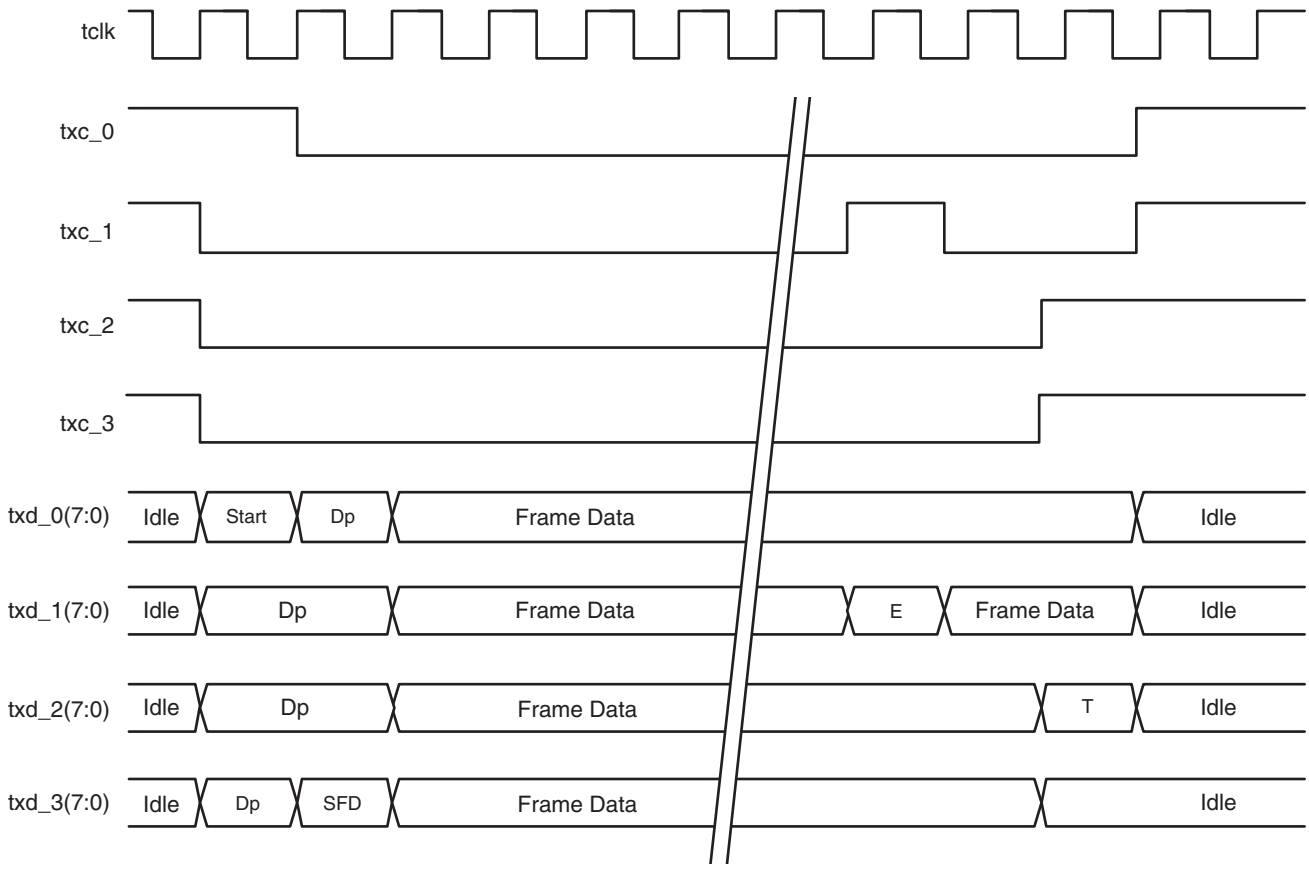

Dp = Preamble data octet

SFD = Start Frame Delimiter

T = Terminate control character (Can occur on any channel)

Figure 7-5 shows operation when an error control character replaces a field data octet.

*Figure 7-5. XAUI Transmit XGMII Interface Signals (Data Error Condition)*




Dp = Preamble data octet

SFD = Start Frame Delimiter

E = Error control character

T = Terminate control character (Can occur on any channel)

**txd_[0-3](7:0)** - 32-bit transmit data from the FPGA XGMII interface. The 32-bit bus is divided into 4 8-bit data busses at the PCS/FPGA logic interface by default. Each channel's txd is sampled on the rising edges of the input transmit clock tclk.

In 16 Bit Data Bus Mode, this bus is 16-bits wide (**txd_[0-3](15:0)**).

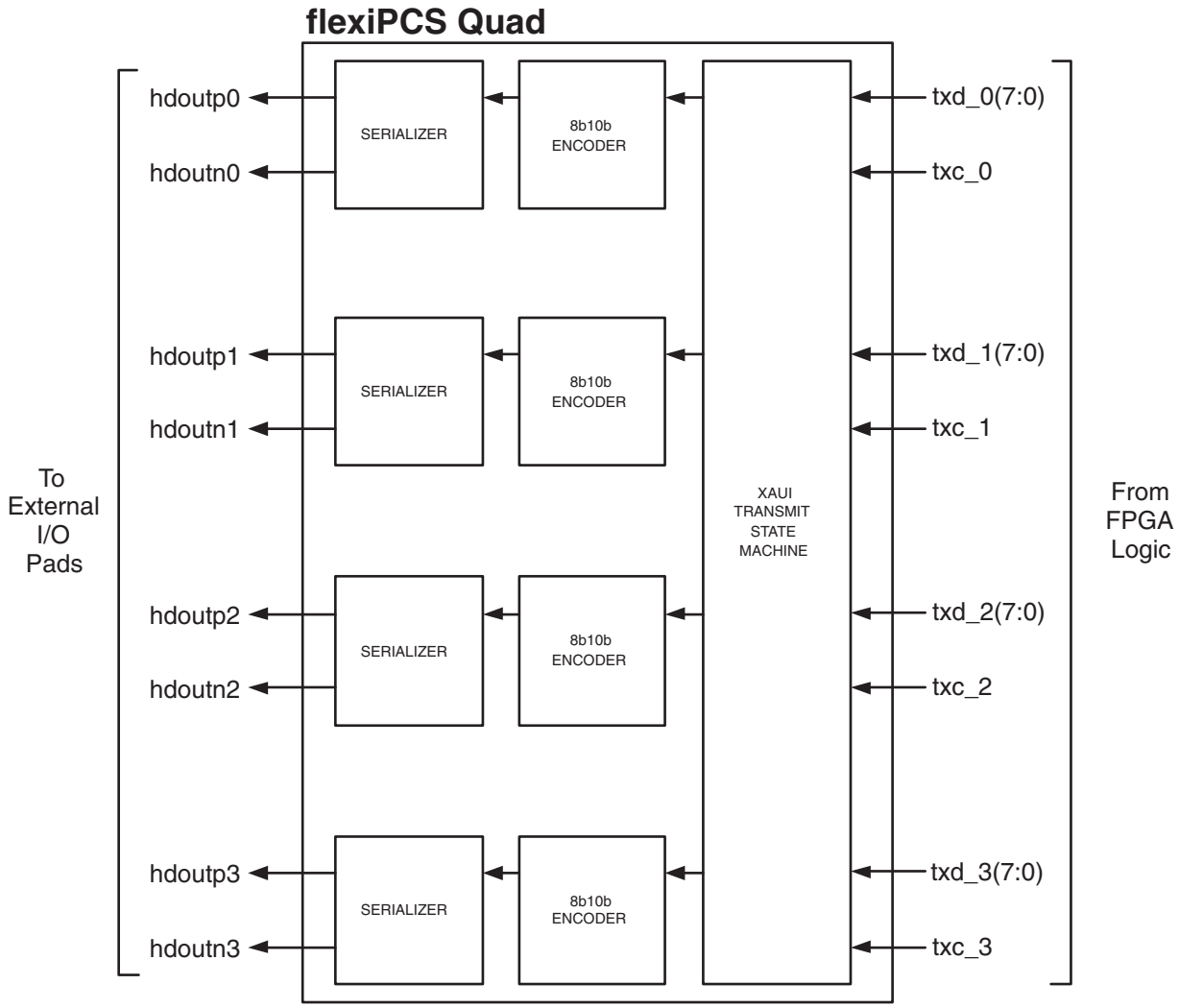
**txc_[0-3]** - Per channel transmit control character indication. txc is high when a control character is present on the corresponding channel's txd inputs. txc is low when a data byte is present on the corresponding channel's txd inputs. When the txc of a channel is asserted, the flexiPCS generates code groups associated with either idle, start, terminate, sequence or error control characters. When the txc of a channel is de-asserted, the flexiPCS generates code-groups corresponding to the data octet value of the corresponding channel's txd. Each channel's txc is sampled on the rising edges of the input transmit clock tclk.

In 16 Bit Data Bus Mode, txc is 2 bits wide (**txc_[0-3](1:0)**) with txc_[0-3](1) associated with txd_[0-3](15:8) and txc_[0-3](0) associated with txd_[0-3](7:0).

Note that the Tx Synchronization bit (Quad Interface Register Offset Address 0x30, bit 5) needs to be set to ensure that all four channels are aligned going into the transmit state machine. Since each Tx channel is clocked by its own transmit clock in the flexiPCS, setting the Tx Synchronization bit ensures these clocks are synchronized to the same clock edges.

The flexiPCS quad in XAUI mode transmit data path consists of the following sub-blocks per channel: XAUI Transmit State Machine, 8b10b Encoder, and Serializer. Figure 7-6 shows the four channels of transmit data paths in a flexiPCS quad.

***Figure 7-6. XAUI Transmit Path (1 Quad)***

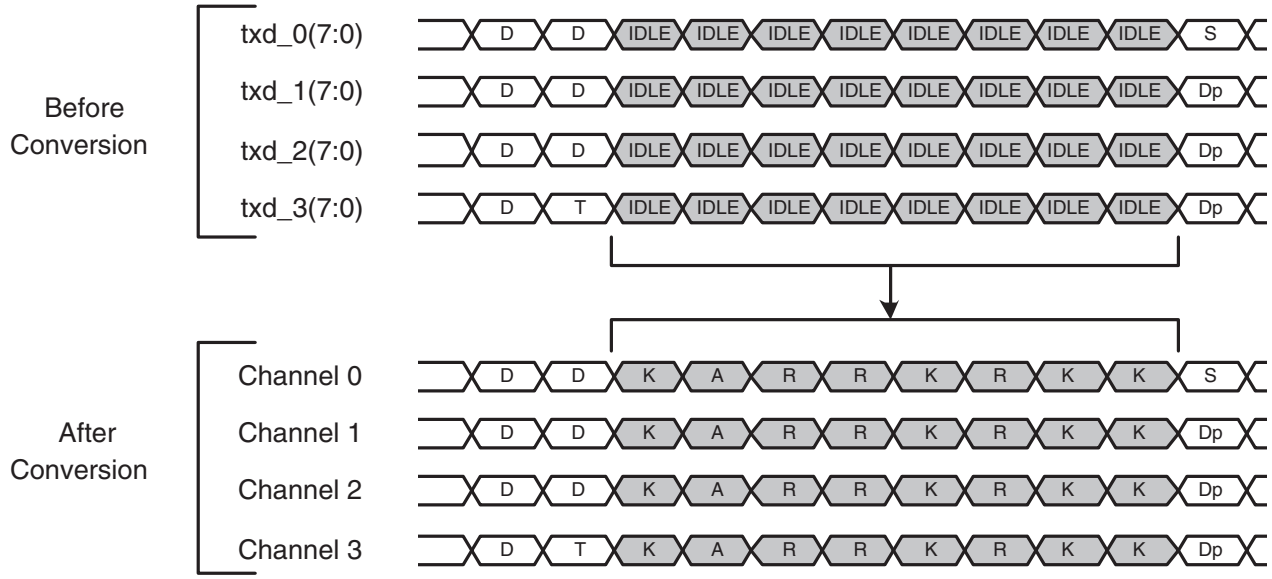



**Transmit State Machine**

The XAUI Transmit State Machine performs the conversion of XGMII idle control characters to a randomized sequence of /A/, /K/, /R/, or /Q/ code-groups to enable lane synchronization, clock compensation and lane-to-lane alignment as described in the flexiPCS Transmit State Diagram (Figure 48-6) in the IEEE 802.3-2002 10GBASE-X specification. This includes a PRBS counter as described in the PCS Idle Randomizer diagram in the 10GBASE-X specification to provide the random code selection.

An example of idle control conversion across the four channels of a quad in XAUI mode is shown in Figure 7-7 below:

***Figure 7-7. XAUI Transmit State Machine Idle Conversion Example***




Detection of txd = 0x9C, txc = 1 (K28.4) on channel 0 at the XGMII interface during an Idle sequence indicates a request to the Transmit State Machine to transmit a Sequence ordered set (||Q||), a Local Fault signal (||LF||), or a Remote Fault signal (||RF||) as defined in Table 7-4. The Transmit State Machine will transmit the appropriate ordered set only after an align ordered set ||A||, as defined in Section 48 of the IEEE 802.3ae-2002 Specification.

***Table 7-4. Transmit defined ordered-sets and special code-groups***

| Code | Ordered_Set | Number of code-groups | Encoding |
|---|---|---|---|
| ||I|| | Idle | | Substitute for XGMII Idle |
| ||K|| | Sync column | 4 | /K28.5/K28.5/K28.5/K28.5/ |
| ||R|| | Skip column | 4 | /K28.0/K28.0/K28.0/K28.0/ |
| ||A|| | Align column | 4 | /K28.3/K28.3/K28.3/K28.3/ |
| | **Encapsulation** | | |
| ||S|| | Start column | 4 | /K27.7/Dx.y/Dx.y/Dx.y/[a] |
| ||T|| | Terminate column | 4 | Terminate code-group in any lane |
| ||T0|| | Terminate in Lane 0 | 4 | /K29.7/K28.5/K28.5/K28.5/ |
| ||T1|| | Terminate in Lane 1 | 4 | /Dx.y/K29.7/K28.5/K28.5/[a] |
| ||T2|| | Terminate in Lane 2 | 4 | /Dx.y/Dx.y/K29.7/K28.5/[a] |
| ||T3|| | Terminate in Lane 3 | 4 | /Dx.y/Dx.y/Dx.y/K29.7/[a] |
| | **Control** | | |
| /E/ | Error code-group | 1 | /K30.7/ |
| | **Link Status** | | |
| ||Q|| | Sequence ordered_set | 4 | /K28.4/Dx.y/Dx.y/Dx.y/[a] |
| ||LF|| | Local Fault signal | 4 | /K28.4/D0.0/D0.0/D1.0/ |
| ||RF|| | Remote Fault signal | 4 | /K28.4/D0.0/D0.0/D2.0/ |
| ||Qrsvd|| | Reserved | 4 | !||LF|| and !||RF|| |
| | **Reserved** | | |
| ||Fsig|| | Signal ordered_set | 4 | /K28.2/Dx.y/Dx.y/Dx.y/[a,b] |

[a]/Dx.y/ indicates any data code-group.
[b]Reserved for INCITS T11

**8b10b Encoding**

This module implements an 8b10b encoder as described within the 802.3-2002 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.

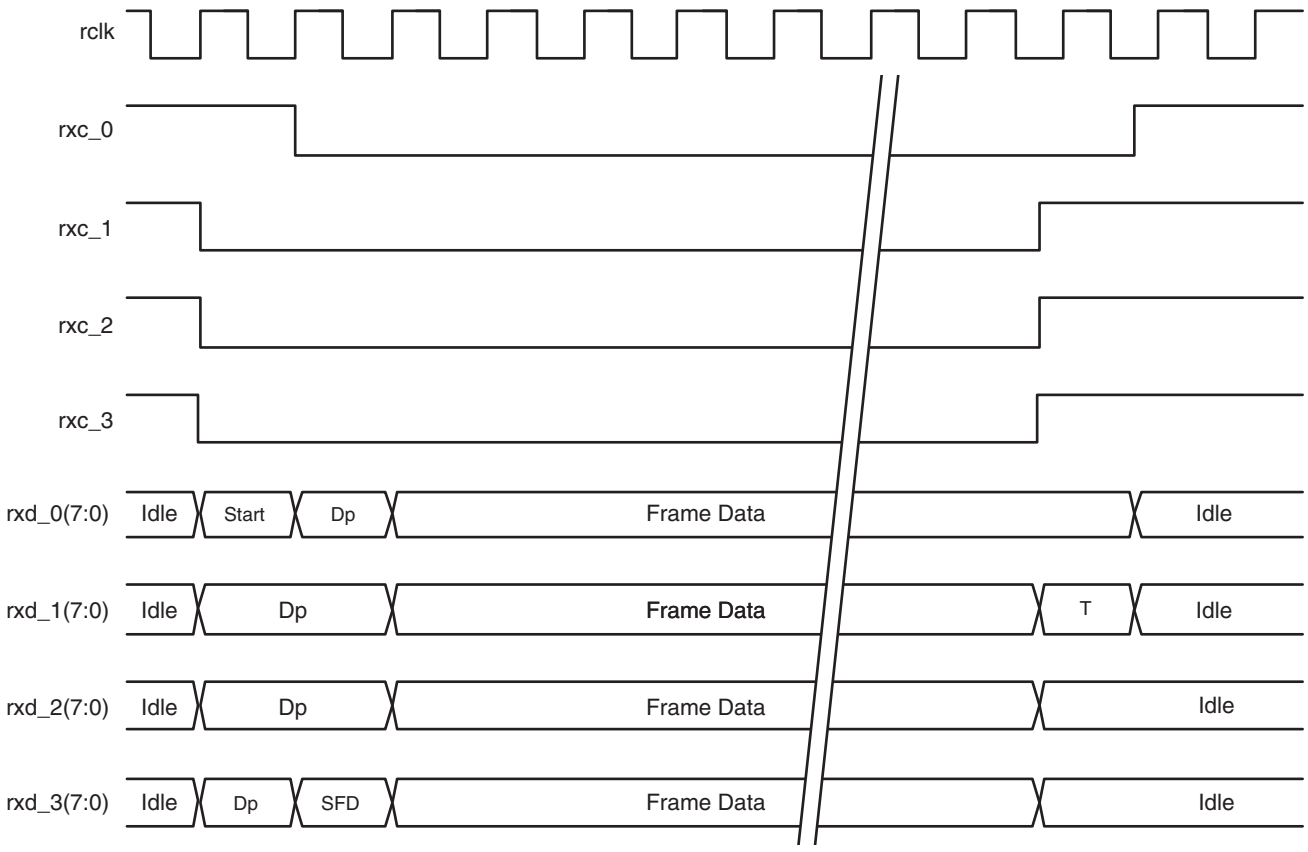
**Serializer**

The 8b10b encoded data undergoes parallel to serial conversion and is transmitted off chip via the embedded SERDES. For detailed information on the operation of the LatticeSC PCS SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

## Receive Data

When configured into XAUI mode, a flexiPCS quad provides four receive data paths from a high-speed serial line interface at the device inputs to a XGMII compliant 32-bit parallel interface at the FPGA logic interface as shown in Figure 7-8.

***Figure 7-8. XAUI Receive XGMII Interface Signals***



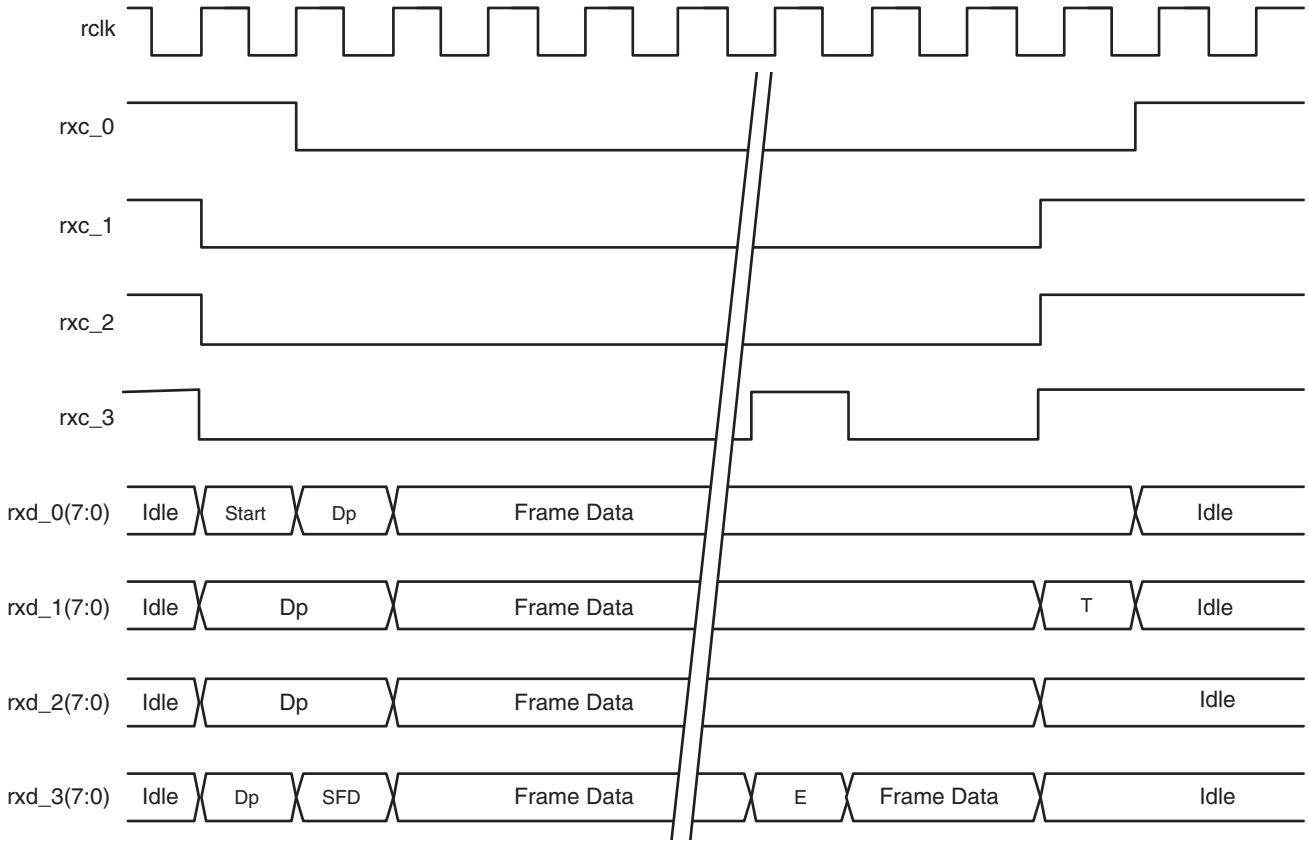

Dp = Preamble data octet

SFD = Start Frame Delimiter

T = Terminate control character (Can occur in any channel)

Figure 7-9 shows operation when an error control character replaces a field data octet.

*Figure 7-9. XAUI Receive XGMII Interface Signals (Data Error Condition)*




Dp = Preamble data octet

SFD = Start Frame Delimiter

E = Error control character

T = Terminate control character (Can occur in any channel)

**rxd_[0-3](7:0)** - 32-bit receive data to the FPGA XGMII interface. The 32-bit bus is divided into four 8-bit data busses at the PCS/FPGA logic interface by default. Each channel's rxd transitions synchronously with respect to that the input receive clock rclk.

In 16 Bit Data Bus Mode, this bus is 16-bits wide (**rxd_[0-3](15:0)**).

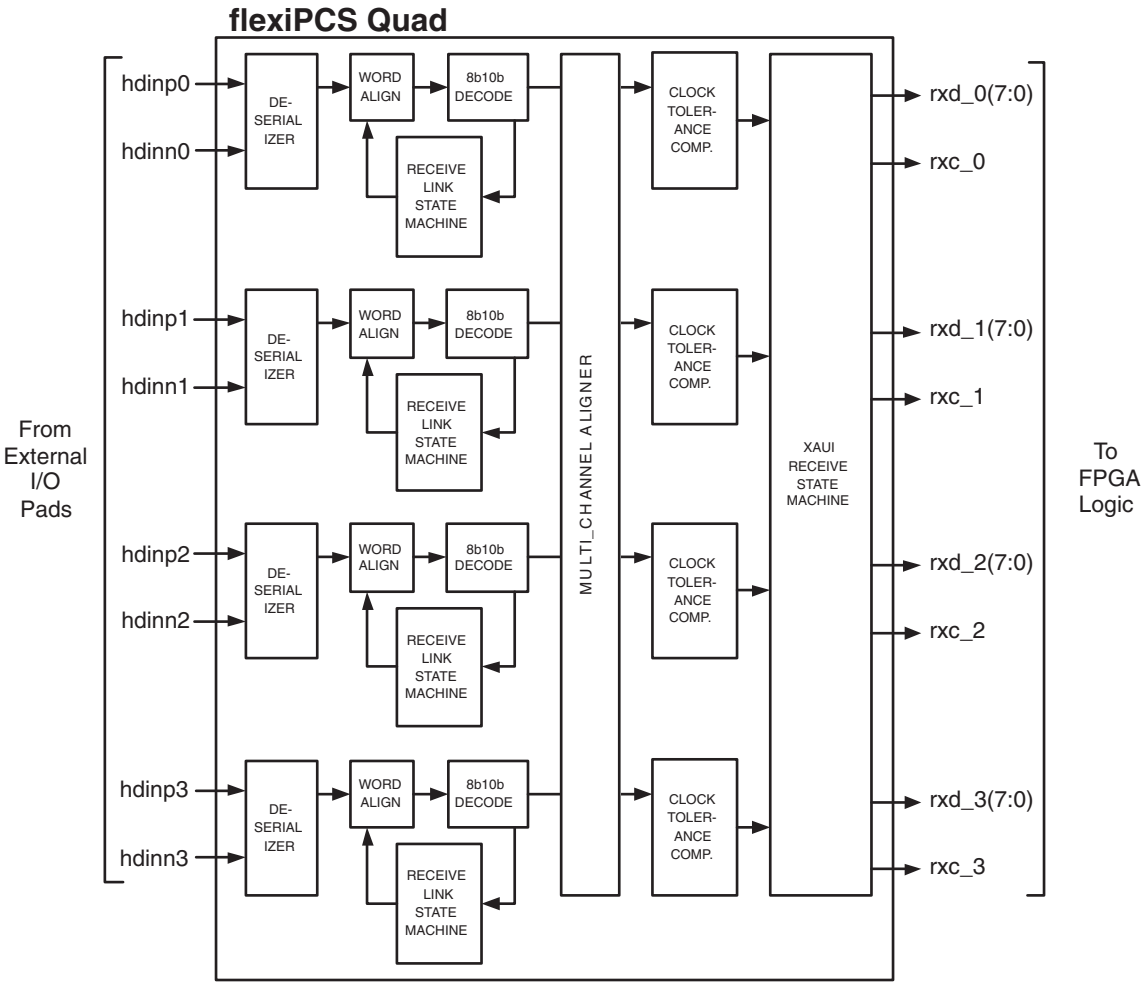
**rxc_[0-3]** - Per channel receive control character indication. rxc is high when a control character is present on the corresponding channel's rxd outputs. rxc is low when a data byte is present on the corresponding channel's rxd outputs. Each channel's rxc transitions synchronously with respect to that channel's input receive clock rclk.

In 16 Bit Data Bus Mode, rxc is 2 bits wide (**rxc_[0-3](1:0)**) with txc_[0-3](1) associated with rxd_[0-3](15:8) and rxc_[0-3](0) associated with rxd_[0-3](7:0).

The flexiPCS quad in XAUI mode receive data path consists of the following sub-blocks per channel: Deserializer, Word Aligner, 8b10b Decoder, & Link State Machine, Multi-channel Aligner, Clock Tolerance Compensator, and

XAUI Receive State Machine. Figure 7-10 shows the four channels of receive data in a flexiPCS quad set to XAUI mode.

***Figure 7-10. XAUI Receive Data Path (1 Quad)***




**Deserializer**

Data is brought on chip to the embedded SERDES where it undergoes serial to parallel conversion. For detailed information on the operation of the LatticeSC PCS SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

**Word Alignment**

The word aligner module performs the word alignment code word detection and alignment operation. The word alignment character is used by the receive logic to perform 10-bit word alignment upon the incoming data stream. The word alignment algorithm is described in Clause 36.2.4.9 in the 802.3-2002 1000BASE-X specification.

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

- 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 word alignment compare. Word alignment characters and the mask register are set on a per quad basis. For XAUI, the word alignment characters should 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).

- The first word alignment character can be set by writing Quad Interface Register Offset Address 0x15 to 0x03.
- The second word alignment character can be set by writing Quad Interface Register Offset Address 0x16 to 0x7C.
- The mask register can be set to compare word alignment bits 6 to 0 by writing Quad Interface Register Offset Address 0x14 to 0x7F (a '1' in a bit of the mask register means check the corresponding bit in the word alignment character register).

**8b10b Decoder**
The 8b10b decoder implements an 8b10b decoder as described with the 802.3-2002 specification. The decoder performs the 10-bit to 8-bit code conversion along with verifying the running disparity.

When a code violation, disparity error, or data error is detected, the rxd data is set to 0xFE and all rxc control bits are set to '1' (corresponding to the Error code group K30.7).

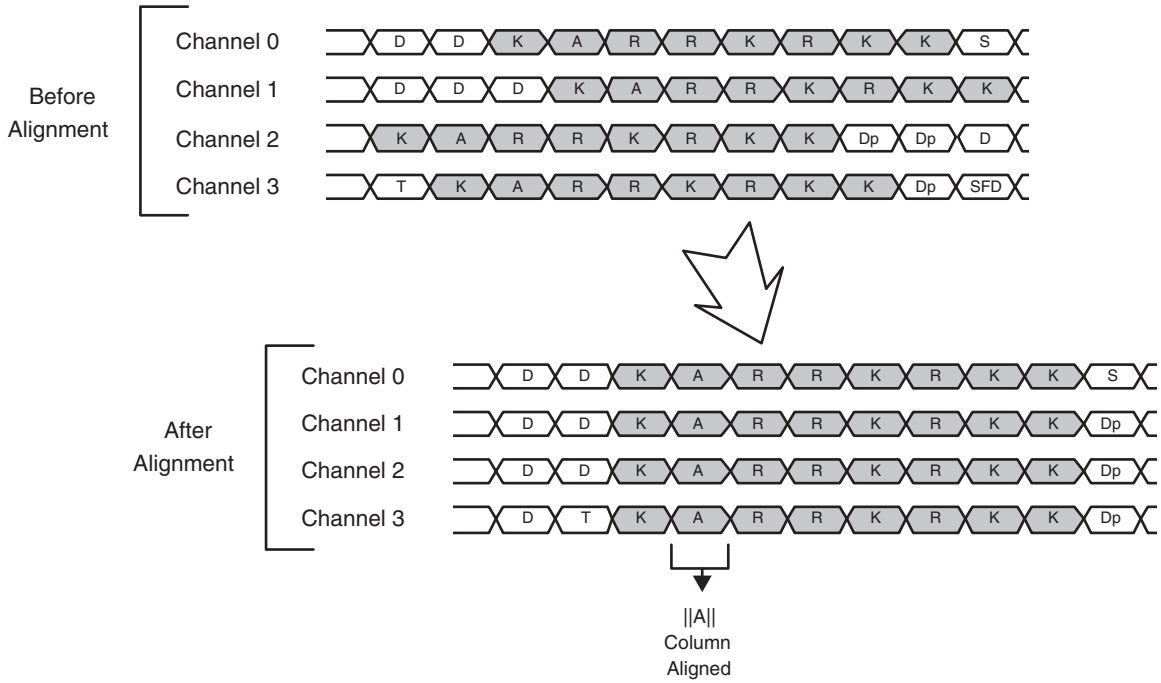
**Link State Machine**
The link synchronization state machine is an extension of the word align module. Link synchronization is achieved after the successful detection and alignment of the required number of consecutive code words. The XAUI Link State Machine is compliant to Figure 48-7 (PCS synchronization state diagram) of IEEE 802.3ae-2002 with one exception. Figure 48-7 requires that four consecutive good code groups be received in order for the LSM to transition from one SYNC_ACQUIRED_{N} (where N=2,3,4) to SYNC_ACQUIRED_{N-1}. Instead, the actual LSM implementation requires five consecutive good code groups to make the transition.

**Multi-channel Aligner**
The Multi-channel alignment module performs the operations and functions to control the multi-channel alignment process. This includes the ||A|| detect and the alignment status signal generation required within the implementation of the alignment state machine as described in the PCS Deskew State Diagram (Figure 48-8) in the IEEE 802ae.3-2002 10GBASE-X standard.

The flexiPCS Multi-channel Aligner allows for alignment of two, three, or four channels in a quad. For XAUI operation, alignment must be performed for all four channels in a quad. The Multi-channel Aligner will align channels based on a user-defined alignment character (referred to as /A/) which must be present in the data stream for all receive channels that are to be aligned. Typically this character is inserted simultaneously in all channels during an Idle sequence between packets upon transmission. If arriving data is skewed due to unequal transport delays, the presence of the alignment characters allows the Multi-channel Aligner to re-align the skewed channels.

Figure 7-11 shows an example of the operation of the flexiPCS multi-channel aligner operation for XAUI applications. The /A/ Align code-groups in each channel are lined up by the multi-channel aligner to create an aligned XGMII compliant data stream at the PCS/FPGA interface.

***Figure 7-11. XAUI Four Channel Alignment Example***




The following is a procedure for settings registers for Multi-channel Alignment.

1. Set the mode for the Multi-channel Aligner and enable/disable the appropriate channels for alignment.

- For XAUI operation, set 4 channel alignment for channels 0, 1, 2, and 3 by writing Quad Interface Register Offset Address 0x19, bit 1 to a '1'.
- Each channel to be aligned must also be enabled for alignment by writing to the appropriate bit in Quad Interface Register Offset Address 0x04. Write a '1' to bit 7 to include channel 0 for alignment. Write a '1' to bit 6 to include channel 1 for alignment. Write a '1' to bit 5 to include channel 2 for alignment. Write a '1' to bit 4 to include channel 3 for alignment. Multi-channel alignment can also be enabled by setting the **mca_align_en** pin at the FPGA interface high.

2. Set the Multi-channel Alignment output clocks. A clock domain transfer occurs between the inputs and outputs of the Multi-channel Aligner. Each channel's receive data is clocked into the Multi-channel Aligner with its own separate recovered receive clock. Each channel's receive data is clocked out of the Multi-channel Aligner on a unique multi-channel receive clock. Each multi-channel receive clock can be programmed to be any of the recovered receive clocks.

It is expected that the user will assign the same recovered receive clock to all the multi-channel output clocks for every channel that is to be aligned. That way, all aligned channels coming out of the Multi-channel Aligner are effectively clocked by the same clock. For more information on setting up a multi-channel receive clock, refer to the **Multi-Channel Alignment** section of the LatticeSC/M Family flexiPCS Data Sheet.

For example, in a 4 channel alignment application, in order to set up all Multi-channel Alignment clocks to be sourced from the channel 0 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0x00. To set all Multi-channel Alignment clocks to be sourced from the channel 1 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0x55. To set all Multi-channel Alignment clocks to be sourced from the channel 2 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xAA. To set all Multi-channel Alignment clocks to be sourced from the channel 3 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xFF.

3. Set the Multi-channel Alignment characters. The Multi-channel Aligner provides the ability to align channels of incoming data to one of two 10-bit user defined maskable patterns. Both alignment characters should be set to the same value if only one alignment character is defined for an application. For XAUI operation, the alignment character should be set to ||A|| (K28.3).

- A user programmable mask word allows the user to set which individual bits are compared to the user defined channel alignment character. When a '1' is present in the user programmable mask, the corresponding bit of the channel alignment characters is compared. When '0' is present in the user programmable mask, the corresponding bit of the channel alignment characters is not compared. For XAUI operation, set the lowest 9 significant bits of the user programmable mask by writing Quad Interface Register Offset Address 0x07 to 0xFF and Quad Interface Register Offset Address 0x0A, bit 3 to '1'.
- The first channel alignment character can be set to the XAUI Align column value ||A|| (K28.3) by writing Quad Interface Register Offset Address 0x08 to 0x7C. The first channel K character can be set by writing Quad Interface Register Offset Address 0x0A, bit 5 to '1'.
- The second channel alignment character can be set to the XAUI Align column value ||A|| (K28.3) by writing Quad Interface Register Offset Address 0x09 to 0x7C. The second channel K character can be set by writing Quad Interface Register Offset Address 0x0A, bit 7 to '1'.

4. Set the Multi-channel Aligner FIFO latency and high-water mark values. Latency values from 0 to 31 can be set by writing to Quad Interface Register Offset Address 0x05, bits [3:7]. The FIFO high-water mark values from 0 to 63 can be set by writing to Quad Interface Register Offset Address 0x06, bits [2:7]. The FIFO high-water mark should be set to the maximum the number of bytes of skew allowed for de-skewing. Larger values of FIFO high-water mark increase the chance of FIFO overrun or underrun. For more information on choosing latency and high-water mark values, refer to the Multi-Channel Alignment section of the LatticeSC/M Family flexiPCS Data Sheet.

5. Reset the Multi-channel Aligner state machine and FIFO control logic if desired with the **mca_resync** pin at the FPGA interface. **mca_resync** is an active high asynchronous reset which resets the Multi-channel Aligner for all four channels.

When the Multi-channel Aligner is set to 4-channel alignment, the Multi-channel Alignment state machine for all four channels can be disabled by writing Quad Interface Register Offset Address 0x04, bit 3 to a '1'. When the Multi-channel Alignment state machine is disabled, the alignment state machine will be held in the "LOSS_OF_ALIGNMENT" state and no alignment will be performed for the relevant channels.

**Clock Tolerance Compensation**

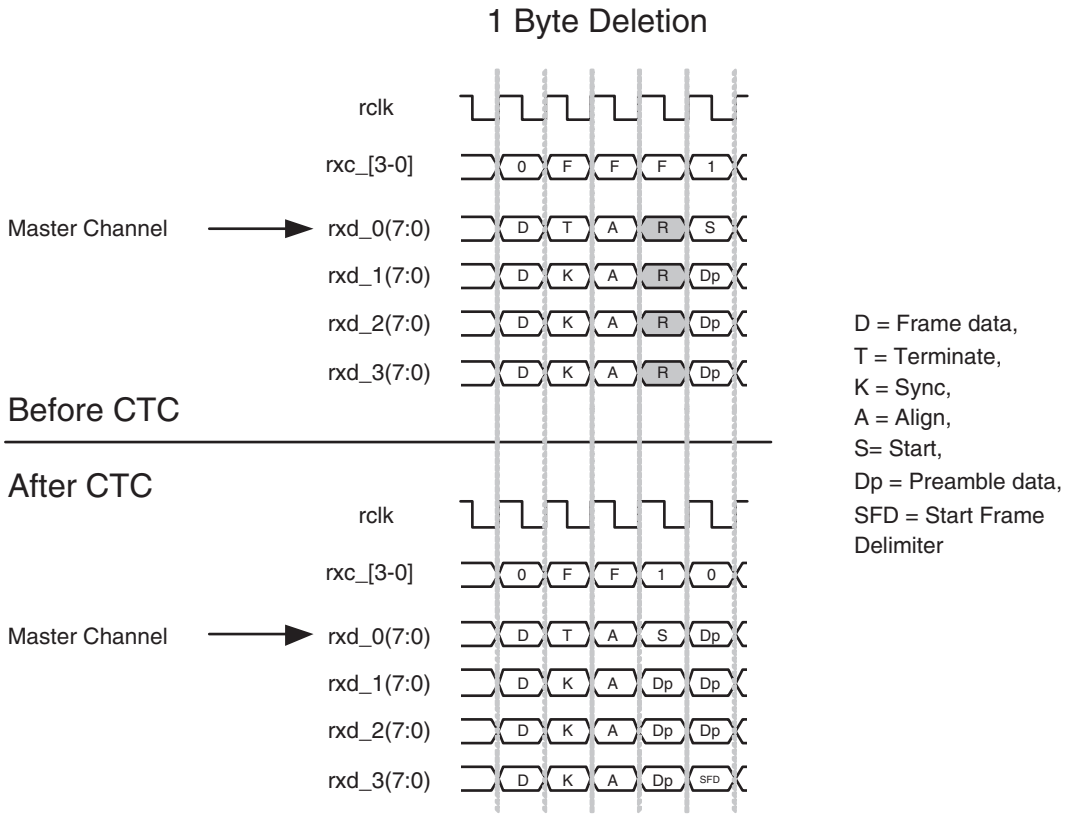
The Clock Tolerance Compensation module performs clock rate adjustment between the recovered receive clocks and the locked reference clock. Clock compensation is performed by inserting or deleting bytes at pre-defined positions, without causing loss of packet data. A 16-Byte elasticity FIFO is used to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

When insertion/deletion is performed by the clock tolerance compensation logic for XAUI operation, insertion or deletion is performed across all aligned channels simultaneously (to preserve the multi-channel alignment).

Note that when four channel alignment is selected, channel 0 is selected as the master channel for performing clock tolerance compensation. This means that only channel 0 is monitored for the presence of the SKIP character. Insertion or deletion is performed on all four channels simultaneously based on detection of the SKIP character in channel 0. This means that channel 0 must be active and transmitting XAUI compliant data for the clock tolerance compensation function to work in four channel alignment mode. If channel 0 is not active, receive data will not be present on the rxd outputs of the flexiPCS.

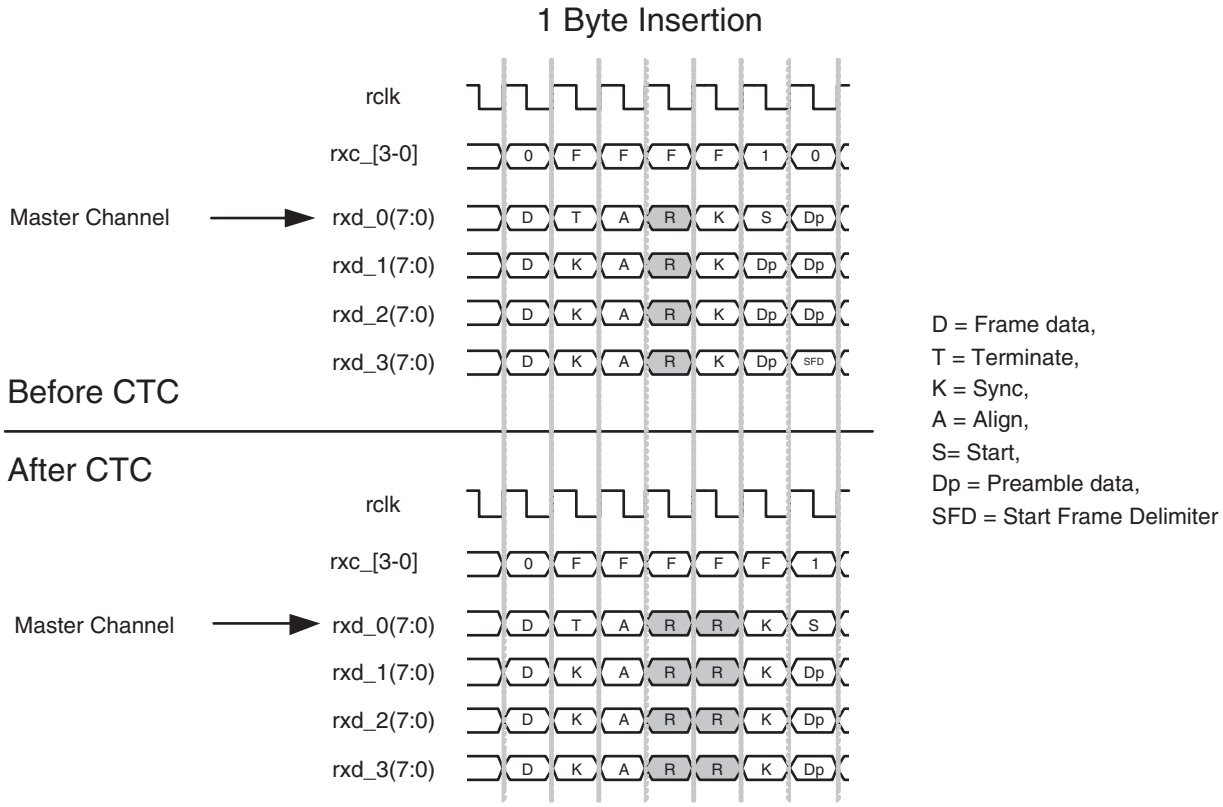
A diagram illustrating 1 byte deletion for XAUI operation is shown in Figure 7-12:

***Figure 7-12. XAUI Operation Clock Compensation Byte Deletion Example***




A diagram illustrating 1 byte insertion for XAUI operation is shown in Figure 7-13:

***Figure 7-13. XAUI Operation Clock Compensation Byte Insertion Example***




Clock compensation values are set on a quad basis. The following settings for clock compensation should be set for XAUI applications:

- Set the insertion/deletion pattern length to 1 and minimum inter-packet gap to 1 bytes by setting Quad Interface Register Offset Address 0x0E to 0x00. The minimum allowed inter-packet gap after skip character deletion based on these register settings is described below.
- The XAUI insertion/deletion pattern should be set to the skip character. For XAUI mode, the pattern should be set to the Skip column code group ||R|| (K28.0). When 1 byte insertion/deletion is selected, the ||R|| code group can be selected by writing Quad Interface Register Offset Address 0x12 to 0x1C and Quad Interface Register Offset Address 0x13 to 0x01.
- Set the clock compensation FIFO high water and low water marks at 9 and 7 respectively (appropriate for insertion/deletion pattern length of 1) by setting Quad Interface Register Offset Address 0x0D to 0x97.
- Clock compensation FIFO underrun can be monitored by reading the appropriate Channel Interface Register Offset Address 0x85, bit 6. This can be also monitored on the flexiPCS interface pin to FPGA logic labeled ctc_urun_[0-3] if “Optional Direct Control & Status Register Access” is selected when generating the flexiPCS block with IPexpress.
- Clock compensation FIFO overrun can be monitored by reading the appropriate Channel Interface Register Offset Address 0x85, bit 7. This can be also monitored on the flexiPCS interface pin to FPGA logic labeled ctc_orun_[0-3] if “Optional Direct Control & Status Register Access” is selected when generating the flexiPCS block with IPexpress.

Table 7-5 shows the relationship between the user-defined values for inter-packet gap (written to Quad Interface Register Offset Address 0x0E, bits 6 and 7), and the guaranteed minimum number of bytes between packets after a skip character deletion from the flexiPCS. The table shows the inter-packet gap as a multiplier number. The mini-

mum number of bytes between packets is equal to the number of bytes per insertion/deletion times the multiplier number shown in the table. For XAUI, the number of bytes per insertion/deletion is 1 (Quad Interface Register Offset Address 0x0E, bits 4 and 5 are both set to '1'). If Quad Interface Register Offset Address 0x0E, bits 6 and 7 are set to "00", then the minimum inter-packet gap is equal to 1 times 1 or 1 bytes. The flexiPCS will not perform a skip character deletion until the minimum number of inter-packet bytes have been detected.

***Table 7-5. Minimum inter-packet gap multiplier***

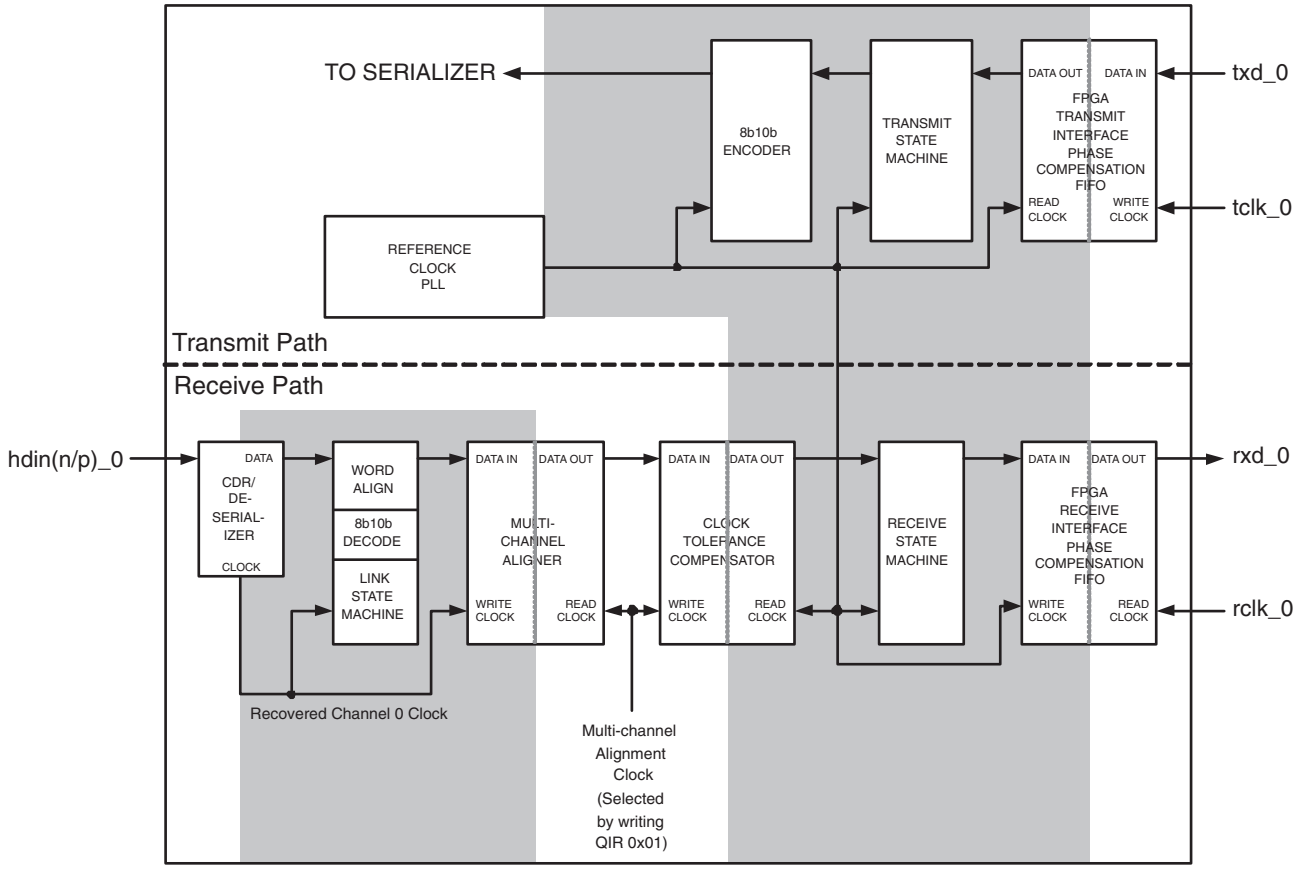
| Quad Interface Register Offset Address 0x0E | | Insertion/Deletion Multiplier Factor |
|---|---|---|
| Bit 6 | Bit 7 | |
| 0 | 0 | 1 X |
| 0 | 1 | 2 X |
| 1 | 0 | 3 X |
| 1 | 1 | 4 X |

Figure 7-14 shows the clock domains for both transmit and receive directions for a single channel inside the flexiPCS.

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

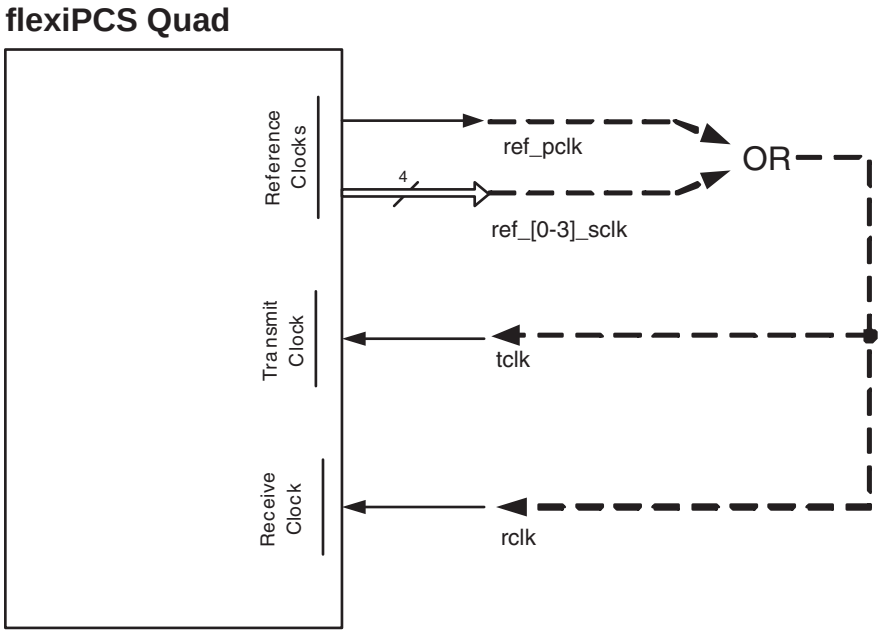
On the receive side, a clock domain transfer from the recovered channel clock to the Multi-channel Alignment channel clock occurs at the Multi-channel Aligner. A second clock domain transfer from the Multi-channel Alignment channel clock to the locked reference clock occurs at the Clock Tolerance Compensator. A third clock domain transfer occurs from the locked reference clock to the rclk input at the FPGA interface at the FPGA Receive Interface phase compensation FIFO. The FPGA Receive Interface phase compensation FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. These phase compensation FIFOs (one per channel) cannot compensate for frequency variations.

***Figure 7-14. flexiPCS Clock Domain Transfers for XAUI Mode***




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 flexiPCS quad set to XAUI mode. Figure 7-15 illustrates the possible connections that would result in a synchronous interface.

*Figure 7-15. Synchronous input clocks to flexiPCS quad set to XAUI Mode*




**XAUI Receive State Machine**

The XAUI receive state machine converts flexiPCS characters to XGMII data and control according to the 802.3ae-2002 10GBASE-X specification. The XAUI receive state machine reverses the operation performed by the XAUI transmit state machine by converting the randomized /A/, /K/, and /R/ PCS code groups to XGMII idle character. In accordance with the 802ae.3-2002 10GBASE-X specification, the XAUI receive state machine also performs the check_end and crvx_terminate operations:

- If channels are not aligned, rxd data is set to the Local Fault group, ||LF|| (/K28.4/D0.0/D0.0/D1.0/). rxd_0(7:0) = 0x9C and rxc_0 = '1' (K28.4), rxd_1(7:0) = 0x00 and rxc_1 = '0' (D0.0), rxd_2(7:0) = 0x00 and rxc_2 = '0' (D0.0), and rxd_3(7:0) = 0x01 and rxc_3 = '0' (D1.0).
- If improper Terminate code group sequence is present as determined by the crvx_terminate protocol in the 802.3-2002 1000BASE-X specification, then all rxd_[0-3](7:0) data is set to 0xFE and all rxc_[0-3] control bits are set to '1' (corresponding to the Error code group K30.7).

## Control & Status

The Control & Status pins for the XAUI mode can be optionally selected on a per quad basis when generating the flexiPCS quad interface files with the ispLEVER tools. Each of these control and status functions are also accessible through equivalent Quad Interface and Channel Interface Registers. The control and status signals allow direct real time access of these functions from FPGA logic.

**Control**

**felb** - Active high control signal which sets up all four channels in Far-End Loopback mode. This mode is generally used for testing the flexiPCS logic, looping back receive data back to the transmit direction without crossing the FPGA logic interface. For more information on the details of Far-End Loopback mode, see the **flexiPCS Testing** Section of the flexiPCS Data Sheet. The Far-End Loopback mode can also be initiated by writing Channel Interface Register Offset Address 0x00, bit 5 to '1'.

**lsm_en** - Active high Receive Link State Machine Enable for all four channels. A change on lsm_en resets the Receive Link State Machines into No Sync mode. The Receive Link Machine Enable can also be set by writing Channel Interface Register Offset Address 0x00, bit 7 to '1'.

**mca_align_en -** Active high multi-channel align enable. Multi-channel Alignment Enable can also be set by writing the appropriate Quad Interface Register Offset Address 0x04, bits [4:7] to '1' (bit 4 for channel 3, bit 5 for channel 2, bit 4 for channel 1, and bit 7 for channel 0).

**mca_resync** - Active high, asynchronous reset to multi-channel aligner state machine and aligner FIFO control logic. mca_resync resets logic in all four channels in four channel alignment mode.

**Status**

**lsm_status_[0-3]** - Per channel signal from the Receive Link State Machine indicating link synchronization successful. The link synchronization status can also be read from the appropriate Quad Interface Register Offset Address 0x84, bits [4:7] (bit 4 for channel 0, bit 5 for channel 1, bit 4 for channel 2, and bit 7 for channel 3).

**mca_aligned** - Active high signal indicating successful alignment of all four channels in four channel alignment mode. The multi-channel alignment status can also be read from the Quad Interface Register Offset Address 0x82, bit 2.

**ctc_urun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has underrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 6.

**ctc_orun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has overrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 7

## Status Interrupt Registers

Some of the status registers associated with XAUI operation have an associated status interrupt and status interrupt enable register. Any flexiPCS status interrupt register will generate an interrupt to the Systembus block (if used in the design). For a description of the operation of interrupts with the Systembus, refer to the Systembus section of the LatticeSC/M Family Data Sheet. Operation of the interrupt registers for the flexiPCS is as follows:

User must set the appropriate status interrupt enable bit to a '1'

Whenever the associated status bit goes high, its status interrupt bit will also go high. The status interrupt bit will remain high even if the associated status bit goes low.

The status interrupt bit will remain high until it is read, when it will automatically be cleared. If the associated status bit is still high, then the status interrupt bit will go high again (until read the next time).

Table 7-6 shows a listing of the status registers associated with XAUI operation which have a corresponding status interrupt and status interrupt enable bit.

***Table 7-6. Status Interrupt Register Bits***

| Status Register Bit Function | Status Register Bit Address | Status Interrupt Enable Register Bit Address | Status Interrupt Register Bit Address |
|---|---|---|---|
| Receive Link State Machine Status (lsm_status) Channel 0 | QIR 0x84, bit 4 | QIR 0x1C, bit 4 | QIR 0x85, bit 4 |
| Receive Link State Machine Status (lsm_status) Channel 1 | QIR 0x84, bit 5 | QIR 0x1C, bit 5 | QIR 0x85, bit 5 |
| Receive Link State Machine Status (lsm_status) Channel 2 | QIR 0x84, bit 6 | QIR 0x1C, bit 6 | QIR 0x85, bit 6 |
| Receive Link State Machine Status (lsm_status) Channel 3 | QIR 0x84, bit 7 | QIR 0x1C, bit 7 | QIR 0x85, bit 7 |

*Table 7-6. Status Interrupt Register Bits (Continued)*

| Status Register Bit Function | Status Register Bit Address | Status Interrupt Enable Register Bit Address | Status Interrupt Register Bit Address |
|---|---|---|---|
| Multi-channel Alignment State Machine Status (mca_aligned) | QIR 0x82, bit 2 | QIR 0x0C, bit 2 | QIR 0x83, bit 2 |
| Clock Tolerance Compensation FIFO underrun | CIR 0x85, bit 6 | CIR 0x0A, bit 6 | CIR 0x8A, bit 6 |
| Clock Tolerance Compensation FIFO Overrun | CIR 0x85, bit 7 | CIR 0x0A, bit 7 | CIR 0x8A, bit 7 |

# PCI Express Mode
# LatticeSC/M flexiPCS Family Data Sheet

December 2008 Data Sheet DS1005

## Introduction

This data sheet describes the operation of the PCI Express mode of the LatticeSC flexiPCS. The LatticeSC can support x1, x2, and x4 PCI Express applications with one flexiPCS quad. PCI Express applications of widths x8, x16, and x32 can be realized on one device through the use of multiple flexiPCS quads (limited by the size of the LatticeSC device chosen).

The LatticeSC flexiPCS in PCI Express Mode supports the following operations:

**Transmit Path (From LatticeSC device to line):**

- PCI Express Scrambler compliant to either Specification Version 1.0 ($X^{16} + X^{15} + X^{13} + X^4 + 1$) and Specification Version 1.0a ($X^{16} + X^5 + X^4 + X^3 + 1$).
- 8b10b Encoding
- Receiver Detection
- Electrical Idle

**Receive Path (From line to LatticeSC device):**

- Word Alignment based on the Sync Code Group as defined in the IEEE 802.3-2002 specification.
- 8b10b Decoding
- Link Synchronization State Machine functions incorporating operations defined in the PCS Synchronization State Machine (Figure 48-7) of the IEEE 802.3ae-2002 10GBASE-X Specification.
- x2 or x4 PCI Express operation with one PCS quad set to PCI Express mode
- x8 PCI Express operation with two PCS quads set to PCI Express mode (Up to x32 operation on one device)
- Clock Tolerance Compensation logic capable of accommodating clock domain differences.
- PCI Express Descrambler compliant to either Specification Version 1.0 ($X^{16} + X^{15} + X^{13} + X^4 + 1$) and Specification Version 1.0a ($X^{16} + X^5 + X^4 + X^3 + 1$).

x2 PCI Express operation can be achieved by selecting a flexiPCS quad in PCI Express mode and enabling two channels which can be subsequently aligned using the multi-channel aligner. Channels enabled for x2 PCI Express operation can be channels 0 and 1 and/or channels 2 and 3 of a given flexiPCS quad.

x4 PCI Express operation can be achieved by selecting a flexiPCS quad in PCI Express mode and enabling all four channels which can be subsequently aligned using the multi-channel aligner. IPexpress will produce a minimized port list for a flexiPCS quad set to PCI Express Mode with “Align channels 0, 1, 2, and 3” selected.

x8 PCI Express operation can be achieved by selecting PCI Express mode for two quads and using the Multi-channel Aligner to align all 8 channels in the two quads. As such, x8 PCI Express operation is considered a special application of x4 PCI Express operation and therefore is described as part of the x4 PCI Express Operation Detailed Description in this data sheet.

### LatticeSC flexiPCS Quad Module

Devices in the LatticeSC family have up to 8 quads of embedded flexiPCS logic. Each quad in turn supports 4 independent full-duplex data channels. A single channel can support a fully compliant PCI Express data link and each

© 2008 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

quad can support up to four such channels. Note that mode selection is done on a per quad basis. Therefore, a selection of PCI Express mode for a quad dedicates all four channels in that quad to PCI Express mode.

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

Operation of the SERDES requires the user to provide a reference clock (or clocks) to each active quad to provide a synchronization reference for the SERDES PLLs. Reference clocks are shared among all four channels of a quad. If the transmit and receive frequencies are within the specified ppm tolerance for the LatticeSC SERDES (See DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet), only one reference clock for both transmit and receive is needed for both transmit and receive directions. If the receive frequency is not within the specified ppm tolerance for the LatticeSC SERDES (See the DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet) of the transmit frequency, then separate reference clocks for the transmit and receive directions must be provided.

Simultaneous support of different (non-synchronous) high-speed data rates is possible with the use of multiple quads. Multiple frequencies that are exact integer multiples can be supported on a per channel basis through use of the full-rate/half-rate mode options (see the **SERDES Functionality section** for more details).

## Quad and Channel Option Control

Although the mode selection and reference frequency covers an entire quad, many options covering clocking and data formatting are available on a per channel basis. These options are detailed in the following PCI Express Mode description. Some of these options can be controlled directly through the FPGA interface pins made available on the PCI Express Quad Module.

Other options are available only through dedicated registers that can be written and read via the embedded System Bus Interface (see the System Bus section for more details on the operation of the System Bus), or during device configuration. Depending on the specific option, a register may affect operation of multiple quads, a single quad or an individual channel in a quad. Registers dedicated to multiple quads are listed in the Inter-quad Interface Register Map in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet. Registers dedicated to one quad are listed in the Quad Interface Register Map. Registers dedicated to a single channel are listed in the Channel Interface Register Map.

### Register Map Addressing

Figure 8-1 provides a map of base register addresses for any of the flexiPCS quads on a LatticeSC device. Note that different devices may have different numbers of available quads up to a total of 8 quads (or 32 channels) maximum.

Inter-quad Interface, Quad Interface, and Channel Interface Registers are listed by Offset Address. Each Inter-quad Interface Register, Quad Interface Register, and Channel Interface Register has a unique 18-bit address determined by the specific quad or channel targeted. The addressing of Inter-quad Interface Registers, Quad Interface Registers, and Channel Interface Registers is shown Figure 8-1.

Base addresses are dependant on the physical location of a given quad or channel on a LatticeSC device. Each quad and channel has an associated set of control and status registers. These registers are referred throughout this data sheet by their offset addresses. The unique 18-bit address of a control or status register for a specific quad or channel is simply the offset address of that register added to the base address for that specific quad or channel as shown in Figure 8-1.

Channel Interface Registers can be written in one of three methods. One method is to write to one specific register corresponding to one specific channel. The other two methods involve writing to multiple channel registers with just one write operation. This is referred to as a Broadcast write. A Broadcast write is useful when multiple channel registers are to be written with the same value. The first method of Broadcast writing involves writing to all four channel registers in a quad at one time. A second method of Broadcast writing involves writing to all channel registers for all quads on the left or right side of the device at one time. Therefore, a specific Channel Interface Register may be

accessed through any one of three channel addresses, depending on the number of channel registers being written to at any one time.

A broadcast write to multiple quads cannot be used to power on SERDES in any device-package combination that does not have all SERDES quads bonded because of excess power on the board and jitter is also affected.

Throughout this document, the byte-wide data at each offset address is listed with a hexadecimal value with bit 0 as the MSb and bit 7 as the LSb as per the PowerPC convention. For example if the value for Quad Interface Register Offset Address 0x02 is stated to be 0x15, the bit pattern defined is as shown in Table 8-1:

***Table 8-1. Control/Status Register Bit Order Example***

| **Quad Interface Register Offset Address 0x02 = 0x15** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Data Bit 0 | Data Bit1 | Data Bit 2 | Data Bit 3 | Data Bit 4 | Data Bit 5 | Data Bit 6 | Data Bit 7 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| **1** | | | | **5** | | | |

A full list of register Offset Addresses is provided in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet.

***Figure 8-1. flexiPCS Inter-quad, Quad, and Channel Interface Register Base Map Addressing***

## Auto-Configuration

Initial register setup for each flexiPCS mode can be performed without accessing the system bus by using the auto-configuration tool, IPexpress, in ispLEVER. IPexpress generates 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 system bus is therefore not needed if all quads are to be set via auto-configuration files. However, the system bus must be included in a design if the user needs to change control registers or monitor status registers during operation. Note that a quad reset (Quad Interface Register 0x43, bit 7 or the **quad_rst** port at the FPGA interface) will reset all registers back to their default (not auto-configuration) values. If a quad reset is issued, the flexiPCS registers need to be rewritten via the Systembus.

## PCI Express Register Settings

The auto-configuration file sets registers that perform the following operations. If the auto-configuration file is not used, the appropriate registers need to be programmed via the Systembus. For more options, refer to the flexiPCS register map in the **Memory Map** section of the **LatticeSC/M Family flexiPCS Data Sheet**.

**A flexiPCS quad can be set to PCI Express mode by writing Quad Interface Register Offset Address 0x18 bits[0:7] to 0x04.**

This register value automatically sets up the following options in the flexiPCS for the given quad. Details on the functionality of each chosen option is provided in the **PCI Express Mode Detailed Description** section.

**Transmit Path**

- PCI Express Scrambler is enabled
- 8b10b encoder is enabled

**Receive Path**

- Word aligner is enabled
- 8b10b decoder is enabled
- Receive Link State Machine is set to the XAUI Link State Machine
- PCI Express Descrambler is enabled

Other register settings are required to operate a channel or channels in PCI Express Mode. The following is a list of options that must be set for PCI Express operation. More detail for each option is included in the **PCI Express Mode Detailed Description** section.

- Powerup of appropriate quad and channel. Quad powerup is chosen by writing the appropriate Quad Interface Register Offset Address 0x28, bit 1 to ‘1’. Channel powerup is chosen by writing the appropriate Channel Interface Register Offset Address 0x13, bit 6 (receive direction) and/or bit 7 (transmit direction) to ‘1’. By default, all channels and quads are powered down.
- Receive Link State Machine enable. By default, all channel Receive Link State Machines are disabled. The Receive Link State Machine for a given channel can be enabled by writing the appropriate Channel Interface Register Offset Address 0x00, bit 7 to ‘1’.
- Setting SERDES reset low at end of flexiPCS setup. By default, the SERDES reset is set to ‘1’ (SERDES are reset). The SERDES reset can be set low by writing Quad Interface Register Offset Address 0x41, bit 5 to a ‘0’. For more information on proper flexiPCS powerup and reset sequencing, refer to the **flexiPCS Reset Sequences** and **flexiPCS Power Down Control Signals** descriptions in the LatticeSC/M Family flexiPCS Data Sheet **SERDES Functionality** section.
- Word alignment character(s), such as a comma
- Clock Tolerance Compensation insertion/deletion ordered set values and FIFO settings
- Multi-channel settings including user-defined alignment characters and FIFO settings for x2, x4, and x8 PCI Express operation.

## x4 PCI Express Auto-Configuration Example

An example of an auto-configuration file for a quad set to PCI Express Mode with Multi-channel Alignment and Clock Tolerance Compensation (CTC) enabled and with all four channels enabled and aligned is shown in Figure 8-2. Format for the auto-configuration file is

register type (quad=quad register, ch1=channel 1 register), offset address in hex, register value in hex, # comment

***Figure 8-2. x4 PCI Express Auto-Configuration Example File***

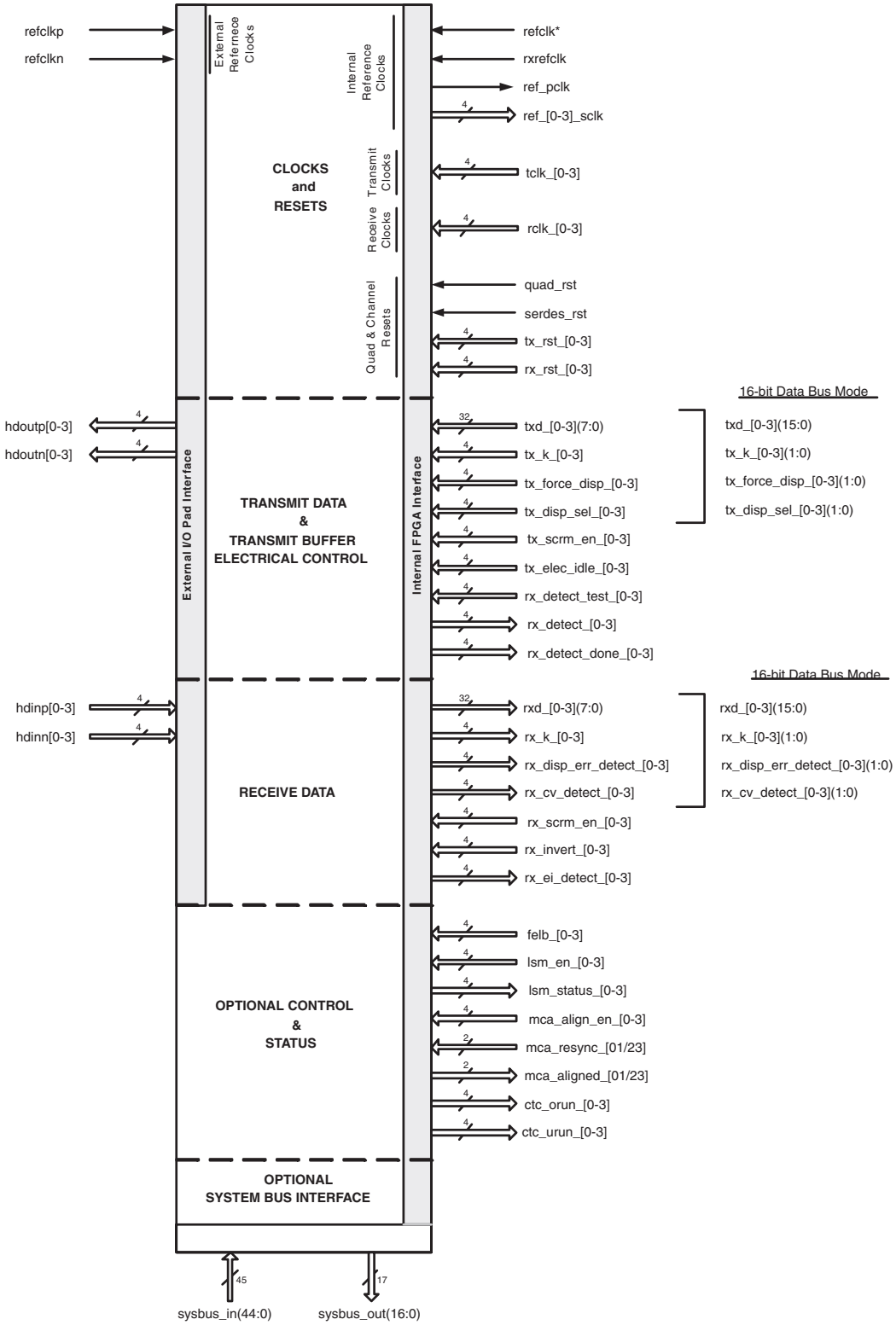
```
quad 18 04    #  Set quad to PCI Express Mode
quad 30 04    #  Set Tx Synchronization bit
quad 29 01    #  Set reference clock select to refclk(p/n) inputs
quad 28 40    #  Set bit clock multiplier to 20X (for full rate mode), quad powerup set to '1'
quad 02 00    #  Set ref_pclk to channel 0 clock
quad 14 7F    #  Set word alignment mask to compare lowest 7 LSBs
quad 15 03    #  Set positive disparity comma value
quad 16 7C    #  Set negative disparity comma value
quad 19 40    #  Select 4 channel alignment (x4 PCI Express), FPGA interface data bus width 8-bits
quad 04 0F    #  Set alignment enable for all four channels
quad 01 00    #  Set all multi-channel alignment clocks to be sourced from
              #  the channel 0 recovered receive clock
quad 07 FF    #  Set multi-channel align character mask to compare bits [7:0]
quad 08 BC    #  Set multi-channel align character “A”, bits [7:0] to BC (K28.5)
quad 09 BC    #  Set multi-channel align character “B”, bits [7:0] to BC (K28.5)
quad 0A 15    #  Set multi-channel align character mask to compare bit 8
              #  Set multi-channel align characters “A” and “B” bit 8 (K character) to '1'
quad 05 00    #  Set multi-channel alignment FIFO latency to minimum
quad 06 03    #  Set multi-channel alignment FIFO high water mark to 3
quad 0E 00    #  Set insertion/deletion to 1 byte for CTC, set minimum inter-packet gap
quad 0D 97    #  Set CTC FIFO watermarks (high=9, low=7)
quad 12 1C    #  1st byte of /SKP/ (skip) pattern for CTC (K28.0)
quad 13 01    #  K bit for /SKP/ pattern
ch0 00 01     #  Turn on channel 0 link state machine
ch0 13 03     #  Powerup channel 0 transmit and receive directions, set rate mode to “full”
ch0 14 90     # 16% pre-emphasis on SERDES output buffer
ch0 15 10     # +6 dB equalization on SERDES input buffer
ch1 00 01     #  Turn on channel 1 link state machine
ch1 13 03     #  Powerup channel 1 transmit and receive directions, set rate mode to “full”
ch1 14 90     # 16% pre-emphasis and on SERDES output buffer
ch1 15 10     # +6 dB equalization on SERDES input buffer
ch2 00 01     #  Turn on channel 2 link state machine
ch2 13 03     #  Powerup channel 2 transmit and receive directions, set rate mode to “full”
ch2 14 90     # 16% pre-emphasis on SERDES output buffer
ch2 15 10     # +6 dB equalization on SERDES input buffer
ch3 00 01     #  Turn on channel 3 link state machine
```

```
ch3 13 03       #  Powerup channel 3 transmit and receive directions, set rate mode to “full”
ch3 14 90       # 16% pre-emphasis on SERDES output buffer
ch3 15 10       # +6 dB equalization on SERDES input buffer

# These lines must appear last in the autoconfig file. These lines apply the

# correct reset sequence to the PCS block upon bitstream configuration

quad 41 00 # de-assert serdes_rst

quad 40 ff   # assert datapath reset for all channels

quad 41 03 # assert MCA reset

quad 41 00 # de-assert MCA reset

quad 40 00 # de-assert datapath reset for all channels
```

## PCI Express Mode

IPexpress allows the designer to choose the mode for each flexiPCS quad used in a given design. Figure 8-3 displays the resulting port interface to one flexiPCS quad that has been set to PCI Express mode with either no alignment options selected or 2 channel alignment selected with the ispLEVER module generator.

***Figure 8-3. PCI Express Mode Pin Diagram (Four channel alignment not selected)***




* The refclk inputs for all active quads on a devicemust be connected to the same clock. The rxrefclk inputs for all active quads on a device do not have to be connected to the same clock.

## PCI Express Mode Pin Description

Table 8-2 lists all inputs and outputs to/from a flexiPCS quad in PCI Express mode (four channel alignment not selected). A brief description for each port is given in the table. A more detailed description of the function of each port is given in the **PCI Express Mode Detailed Description**.

***Table 8-2. PCI Express Mode Pin Description***

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Reference Clocks** | | | |
| refclkp | In from I/O pad | N/A | Reference clock input, positive. Dedicated CML input. |
| refclkn | In from I/O pad | N/A | Reference clock input, negative. Dedicated CML input |
| refclk | In from FPGA | N/A | Optional reference clock input from FPGA logic. Can be used instead of per quad reference clock.<br><br>The refclk inputs for all active quads on a device must be connected to the same clock. |
| frxrefclk | In from FPGA | N/A | Optional receive reference clock input from FPGA logic. Can be used instead of per quad reference clock. The rxrefclk inputs for all active quads do not have to be connected to the same clock. |
| ref_pclk | Out to FPGA | N/A | Locked reference clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| ref_[0-3]_sclk | Out to FPGA | N/A | Per channel locked reference clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| **Transmit Clocks** | | | |
| tclk_[0-3] | In from FPGA | N/A | Per channel transmit clock inputs from FPGA. Used to clock the TX data phase compensation FIFO with clock synchronous to the reference clock. May also be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Receive Clocks** | | | |
| rclk_[0-3] | In from FPGA | N/A | Per channel receive clock inputs from FPGA. May be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference and/or receive reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Resets** | | | |
| quad_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of SERDES and PCS logic. |
| serdes_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of the SERDES. |
| tx_rst_[0-3] | In from FPGA | ASYNC | Per channel active high, asynchronous reset of individual transmit channel of PCS logic. |
| rx_rst_[0-3] | In from FPGA | ASYNC | Per channel active high, asynchronous reset of individual receive channel of PCS logic. |
| **Transmit Data & Transmit Buffer Electrical Control** | | | |
| hdoutp[0-3] | Out to I/O pad | N/A | High-speed CML serial output, positive. |
| hdoutn[0-3] | Out to I/O pad | N/A | High-speed CML serial output, negative. |

*Table 8-2. PCI Express Mode Pin Description*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| txd_[0-3](7:0)<br><br>txd_[0-3](15:0)* | In from FPGA | tclk_[0-3] | Per channel parallel transmit data bus from the FPGA. Bus is 8-bits wide if QIR 0x19, bit 4 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 4 = '1'. |
| tx_k_[0-3]<br><br>tx_k_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel transmit control word indicator.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_force_disp_[0-3]<br><br>tx_force_disp_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel active high, force disparity enable.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_disp_sel_[0-3]<br><br>tx_disp_sel_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel disparity select. High forces positive disparity, low forces negative disparity.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_scrm_en_[0-3] | In from FPGA | ASYNC | Per channel, active high PCI Express scrambler enable. |
| tx_elec_idle_[0-3] | In from FPGA | ASYNC | Per channel, active high electrical idle control. |
| rx_detect_test_[0-3] | In from FPGA | ASYNC | Per channel, active high Receiver Detection test input. |
| rx_detect_[0-3] | Out to FPGA | ASYNC | Per channel, Receiver Detect test result. |
| rx_detect_done_[0-3] | Out to FPGA | ASYNC | Per channel, Receiver Detect test done indicator. |
| **Receive Data** | | | |
| hdinp[0-3] | In from I/O pad | N/A | High-speed CML serial input, positive. |
| hdinn[0-3] | In from I/O pad | N/A | High-speed CML serial input, negative. |
| rxd_[0-3](7:0)<br><br>rxd_[0-3](15:0)* | Out to FPGA | rclk_[0-3] | Per channel parallel receive data bus to the FPGA. Bus is 8-bits wide if QIR 0x19, bit 5 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 5 = '1'. |
| rx_k_[0-3]<br><br>rx_k_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel receive control word indicator.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_disp_err_detect_[0-3]<br><br>rx_disp_err_detect_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high disparity error detection.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_cv_detect_[0-3]<br><br>rx_cv_detect_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high code violation detection.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_scrm_en_[0-3] | In from FPGA | ASYNC | Per channel, active high PCI Express descrambler enable. |
| rx_invert_[0-3] | In from FPGA | ASYNC | Per channel, receive data invert. |
| rx_ei_detect_[0-3] | Out to FPGA | ASYNC | Per channel, Electrical Idle condition detect. |
| **Optional Control & Status** | | | |
| felb_[0-3] | In from FPGA | ASYNC | Per channel active high far-end loopback enables. |
| lsm_en_[0-3] | In from FPGA | ASYNC | Per channel receive link state machine enable. |
| lsm_status_[0-3] | Out to FPGA | ASYNC | Per channel signal from receive link state machine indicating successful link synchronization. |
| mca_align_en_[0-3] | In from FPGA | ASYNC | Per channel active high multi-channel aligner enables. |
| mca_resync_[01/23] | In from FPGA | ASYNC | Active high async multi-channel aligner resynchronization signal |
| mca_aligned_[01/23] | Out to FPGA | ASYNC | Active high indicating successful alignment of channels. |
| ctc_urun_[0-3] | Out to FPGA | ASYNC | Per channel active high flag indicator that clock tolerance compensation FIFO has underrun. |

*Table 8-2. PCI Express Mode Pin Description*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| ctc_orun_[0-3] | Out to FPGA | ASYNC | Per channel active high flag indicator that clock tolerance compensation FIFO has overrun. |
| **Optional System Bus Interface** | | | |
| sysbus_in(44:0) | In from FPGA | N/A | Control and data signals from the internal system bus. |
| sysbus_out(16:0) | Out to FPGA | N/A | Control and data signals to the internal system bus. |

# PCI Express Mode Detailed Description

The following section provides a detailed description of the operation of a flexiPCS quad set to PCI Express mode. The functional description is organized according to the port listing shown in Figure 8-3. The System Bus Offset Address for any control and status registers relevant to a given function are provided with the description of the operation of that function.

## Clocks & Resets

A detailed description of all SERDES clock, data, and reset ports and recommended reset sequencing is provided in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet. The following is a recommended setup of the SERDES for PCI Express applications.

For PCI Express applications, the bit clock rate is typically 2.5 Gbps. This bit clock rate falls into the range defined as “full rate” mode for the SERDES PLLs. Full rate mode is selected separately for each channel and each direction by writing the appropriate Channel Interface Register Offset Address 0x13, bit 5 (transmit) and bit 4 (receive) to a ‘0’ (default).

The reference clock frequency is determined by the reference clock mode chosen. Table 8-3 shows the possible reference clock frequencies which result in a 2.5 Gbps internal bit rate.

*Table 8-3. PCI Express Reference Clock Frequency Options for 2.5 Gbps Bit Clock Rate*

| **Full Rate Mode**<br><br>**Channel Interface Register Offset Address 0x13**<br><br>**Transmit - Bit 5 = ‘0’**<br>**Receive - Bit 4 = ‘0’** | | | | |
|---|---|---|---|---|
| **Reference Clock Mode**<br><br>**Quad Interface Register Offset Address = 0x28, Bits [2:3]** | **Reference Clock Frequency** | **Internal Serial (bit) Clock Frequency** | **FPGA Interface Clock Frequency**<br><br>**Quad Interface Register Offset Address 0x19** | |
| | | | **8 Bit Data Bus Mode**<br><br>**Transmit - Bit 4 = ‘0’**<br>**Receive - Bit 5 = ‘0’** | **16 Bit Data Bus Mode**<br><br>**Transmit - Bit 4 = ‘1’**<br>**Receive - Bit 5 = ‘1’** |
| **00** | **125 MHz** | **2.5 GHz** | **250 MHz** | **125 MHz** |
| **01** | **250 MHz** | **2.5 GHz** | **250 MHz** | **125 MHz** |
| **10** | **500 MHz** | **2.5 GHz** | **250 MHz** | **125 MHz** |

*Note: There are also frequency dependent SERDES performance control bits that are not writable via the Systembus. These control bits are set in the auto-configuration file generated within IPexpress and passed into the bitstream through the normal FPGA design flow (map, par, bitgen). To change the bit rate for a SERDES, specify the proper values for the affected flexiPCS quad in IPexpress and regenerate the auto-configuration file. Failure to do this may result in non-optimal SERDES operation.*

Additional information on all reference clock rate modes can be found in the **SERDES Functionality** section of the flexiPCS Data Sheet.

**Quad & Channel Resets**
Resets are provided to reset an entire quad (either SERDES logic only or all flexiPCS logic) or each individual channel (all flexiPCS logic per channel). These resets are driven from the FPGA logic. A summary of the control signals provided for reset are listed below. More detail on recommended initialization and reset sequences for the SERDES is provided in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

The following inputs to the flexiPCS from the FPGA are enabled as long as Quad Interface Register Offset Address 0x42, bit 7 is set to '1' (which is the default on powerup). Writing a '0' to this bit will disable these reset inputs.

**quad_rst** - Active high, asynchronous signal from FPGA resets all SERDES and PCS logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x43, bit 7 to '1'. quad_rst also resets all flexiPCS control registers. If these registers had been previously written via the Systembus or set during configuration through an auto-configuration file, they will need to be rewritten following this reset.

**serdes_rst** - Active high, asynchronous signal from FPGA resets all SERDES (but not PCS) logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x41, bit 5 to '1'.Note that this bit is preset to '1' on powerup and must be written to '0' before operation of the SERDES can commence.

**tx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one transmit channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [4:7]. Bit 7 resets transmit channel 0, bit 6 resets transmit channel 1, bit 5 resets transmit channel 2, and bit 4 resets transmit channel 3.

**rx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one receive channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [0:3]. Bit 3 resets receive channel 0, bit 2 resets receive channel 1, bit 1 resets receive channel 2, and bit 0 resets receive channel 3.

## Transmit Data

When configured into PCI Express mode, a flexiPCS quad provides up to four transmit data paths. Each transmit data path converts a 8-bit wide PCI Express compliant data stream at the FPGA logic interface to a high-speed serial line interface at the device outputs.

The following signals are the transmit path inputs from the FPGA logic to a single flexiPCS set to PCI Express Mode with all 4 channels enabled:

**txd_[0-3](7:0)** - Per channel transmit data from the FPGA logic interface. Each quad supports up to 4 independent channels of 8-bit wide parallel data by default. Each txd_[0-3][7:0] is an eight bit data bus that is driven synchronously with respect to the corresponding tclk_[0-3].

In 16-Bit Data Bus Mode, this bus is 16-bits wide (**txd_[0-3](15:0)**).

**tx_k_[0-3]** - Per channel, active high control character indicator.

In 16 Bit Data Bus Mode, tx_k is 2 bits wide (**tx_k_[0-3](1:0)**) with tx_k_[0-3](1) associated with txd_[0-3](15:8) and tx_k_[0-3](0) associated with txd_[0-3](7:0).

**tx_force_disp_[0-3]** - Per channel, active high signal which instructs flexiPCS to accept disparity value from the tx_disp_sel_[0-3] FPGA interface input.

In 16 Bit Data Bus Mode, tx_force_disp is 2 bits wide (**tx_force_disp_[0-3](1:0)**) with tx_force_disp_[0-3](1) associated with txd_[0-3](15:8) and tx_force_disp_[0-3](0) associated with txd_[0-3](7:0).

**tx_disp_sel_[0-3]** - Per channel, disparity value supplied from FPGA logic. It is valid when tx_force_disp_[0-3] is high. A '1' value for **tx_disp_sel** forces positive disparity, and a '0' value for **tx_disp_sel** forces negative disparity.

In 16 Bit Data Bus Mode, tx_disp_sel is 2 bits wide (**tx_disp_sel_[0-3](1:0)**) with tx_disp_sel_[0-3](1) associated with txd_[0-3](15:8) and tx_disp_sel_[0-3](0) associated with txd_[0-3](7:0).

**tx_scrm_en_[0-3]** - Per channel, active high PCI Express scrambler enable. Enables the scrambler on the first clock cycle after **tx_scrm_en** goes high. Disables the scrambler on the first clock cycle after **tx_scrm_en** goes low.

**tx_elec_idle_[0-3]** - Per channel, active high electrical idle control. When held high, sets the corresponding channel's SERDES transmit buffers into the PCI Express Electrical Idle state.

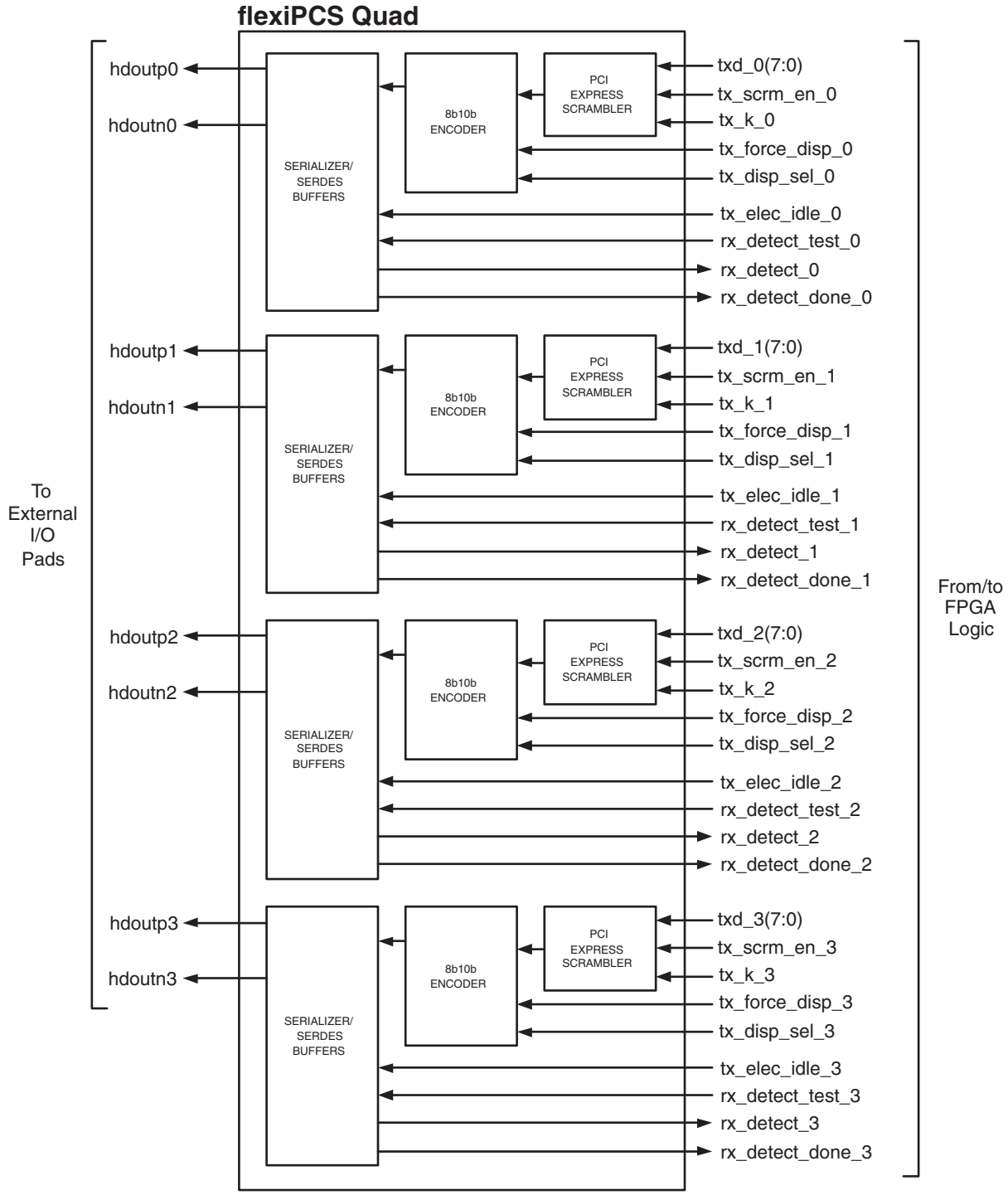
**rx_detect_test_[0-3]** - Per channel, active high Receiver Detection test input. When set high, starts the Receiver Detection test sequence for the corresponding channel's SERDES transmit buffers.

**rx_detect_[0-3]** - Per channel, Receiver Detect test result. When high, indicates that a far end receiver has been detected. If low after the **tx_rcv_detect_done** signal goes high, indicates that a far end receiver has not been detected. **tx_rcv_detect** is cleared when the corresponding channel's **tx_rcv_detect_test** is set high.

**rx_detect_done_[0-3]** - Per channel, Receiver Detect test done indicator. Goes high when time for expected time for completion of Receiver Detect test has expired. Status of Receiver Detect test can then be read at **tx_rcv_detect** pin.

The flexiPCS quad in PCI Express mode transmit data path consists of the following sub-blocks per channel: PCI Express Scrambler, 8b10b Encoder, and Serializer. Figure 8-4 shows the four channels of transmit data paths in a flexiPCS quad.

*Figure 8-4. PCI Express Transmit Path (1 Quad)*




**PCI Express Scrambler**

The PCI Express Scrambler performs a PCI Express compliant scrambling function on the txd data. The PCI Express scrambler supports both the scrambler functions described in Specification Version 1.0 ($X^{16} + X^{15} + X^{13} + X^4 + 1$) and Specification Version 1.0a ($X^{16} + X^5 + X^4 + X^3 + 1$). The specific scrambler function is chosen on a per quad basis by writing to Quad Interface Register Offset Address 0x19, bit 7. Bit 7 = '0' selects Version 1.0a (default) and bit 7 = '1' selects Version 1.0.

The PCI Express Scrambler performs data scrambling when the **tx_scrm_en** input is low except when a K control word is detected or during the training sequences TS1, and TS2 as defined in the PCI Express Base Specification 1.0a Section 4.2.4.1.

The PCI Express Scrambler is designed to halt the scrambling operation when the /COM/ symbol (K28.5) is detected (txd = 0xBC, tx_k = '1'). The training sequences TS1 and TS2 both start with the /COM/ symbol. The Scrambler has a byte counter which starts upon detection of /COM/. The Scrambler then restarts when either of the following conditions occur, whichever comes first:

- The byte counter reaches 16 (The TS1 and TS2 training sequences are defined to be 16 bytes in length)

**OR**

- The Scrambler detects another K control word before the counter reaches 16 UNLESS the K control word is a /PAD/ symbol (K23.7, txd = 0xF7, tx_k ='1'). /PAD/ is the only K control word expected in a TS1 or TS2 training sequence. Other K control words after the /COM/ indicate that the sequence is not a TS1 or TS2 sequence and scrambling commences on the next non-K control word.

The PCI Express scrambler is automatically initialized by the /COM/ character (K28.5) as per the PCI Express Data Scrambling specification (Version 1.0a). The scrambler LFSR value is advanced 8 serial shifts for each character except for the /SKP/ character (K28.0) in accordance with the Data Scrambling specification.

Each channel's PCI Express Scrambler/Descrambler can be disabled by writing the appropriate Channel Interface Offset Register 0x05, bit 1 to a '1'. If Channel Interface Offset Register 0x05, bit 1 is set to a '1', that channel's scrambler/descrambler will be disabled regardless of the state of **tx_scrm_en.** If Channel Interface Offset Register 0x05, bit 1 is set to a '0' (default), then the operation of the scrambler/descrambler is controlled by the **tx_scrm_en** port.

**8b10b Encoding**
This module implements an 8b10b encoder as described in Appendix B of the PCI Express Base Specification, Revision 1.0a. The encoder performs the 8-bit to 10-bit code conversion as described the specification, along with maintaining the running disparity rules as specified.

**Serializer**
The 8b10b encoded data undergoes parallel to serial conversion and is transmitted off chip via the embedded SERDES. For detailed information on the operation of the LatticeSC PCS SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

Note that the Tx Synchronization bit (Quad Interface Register Offset Address 0x30, bit 5) needs to be set to ensure that all four channels are aligned and meet PCI Express transmit skew specifications.

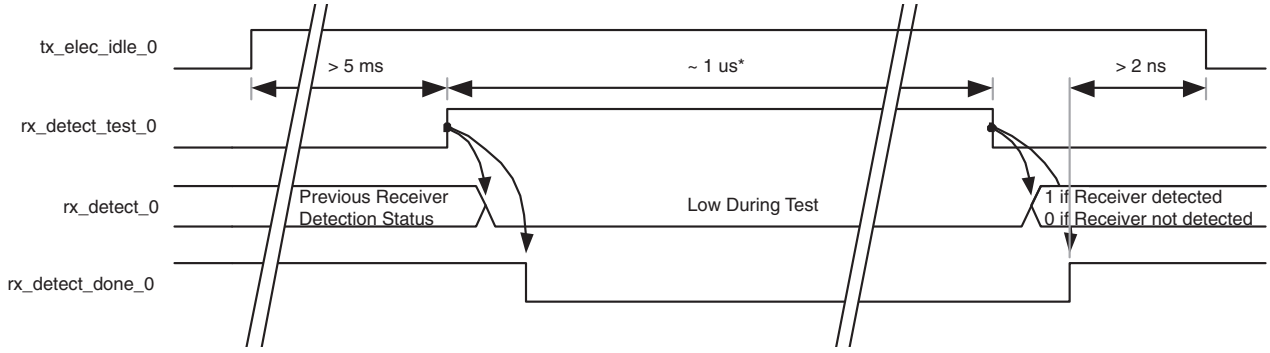
**Electrical Idle**
Each channel's SERDES transmit buffers can independently be set to Electrical Idle mode by setting the appropriate channel's **tx_elec_idle_[0-3]** input to the flexiPCS to a '1'. The SERDES transmit buffers will go into the Electrical Idle state within 120 ns after **tx_elec_idle** goes high**.** The SERDES transmit buffers will exit Electrical Idle within xx ns after **tx_elec_idle** goes low.

Note that the flexiPCS does not automatically transmit the Electrical Idle ordered set (/K28.5/K28.3/K28.3/K28.3/) when **tx_elec_idle** is asserted high. The Electrical Idle ordered set must be sent to the flexiPCS transmit inputs (txd, txc) and transmitted before the transmit buffers are put into the Electrical Idle state to meet the Electrical Idle transmission requirements as defined in Section 4.13.1.9 of the PCI Express Base Specification 1.0a. In order to meet this condition, the last byte of the Electrical Idle ordered set must be clocked into the flexiPCS 10 **ref_pclk** cycles before **tx_elec_idle** is asserted (in 8 Bit Data Bus Mode).

**Receiver Detection**
Figure 8-5 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 **tx_elec_idle** input high. The Receiver Detection test can begin 5 ms after tx_elec_idle is set high by driving the appropriate **rx_detect_test_[0-3]** high. The corresponding channel's **rx_detect_done** is then cleared asynchronously. The **rx_detect_test** input can then be deasserted (driven low) after enough time to complete a Receiver Detection test has passed, nominally 1 us. Once the **rx_detect_test** input is deasserted, the **rx_detect_done** port will go high and the Receiver Detect status will appear at the **rx_detect** port. If at that time the **rx_detect** port is high, then a receiver has been detected on that channel. If, however, the **rx_detect** port is low, then no receiver has been detected for that channel. Once the Receiver Detect test is complete, **tx_elec_idle** can be deasserted.

***Figure 8-5. PCI Express Mode Receiver Detection Sequence (Example for Channel 0)***




* Optimal detect_test high time of 1 us +/- 0.5 us. With 250 Mhz ref_pclk, optimal high time corresponds to 256 clock cycles. With 125 Mhz ref_pclk (16 Bit Data Bus Mode), optimal high time corresponds to 128 clock cycles.

## Receive Data

When configured into PCI Express mode, a flexiPCS quad provides up to four receive data paths from a high-speed serial line interface at the device inputs to 8-bit parallel interface at the FPGA logic interface.

The following signals are the receive path outputs to the FPGA logic from a single quad flexiPCS set to PCI Express Mode with all 4 channels enabled:

**rxd_[0-3](7:0)** - 4 separate 8-bit wide receive data bus outputs to the FPGA interface by default. Each channel's rxd transitions synchronously with respect to that channel's corresponding rclk.

In 16 Bit Data Bus Mode, this bus is 16-bits wide (**rxd_[0-3](15:0)**).

**rx_k_[0-3]** - Per channel, active high control character indicator.

In 16 Bit Data Bus Mode, rx_k is 2 bits wide (**rx_k_[0-3](1:0)**) with rx_k_[0-3](1) associated with rxd_[0-3](15:8) and rx_k_[0-3](0) associated with rxd_[0-3](7:0).

**rx_disp_err_detect_[0-3]** - Per channel, active high signal driven by the flexiPCS to indicate a disparity error was detected with the associated data.

In 16 Bit Data Bus Mode, rx_disp_err_detect is 2 bits wide (**rx_disp_err_detect_[0-3](1:0)**) with rx_disp_err_detect_[0-3](1) associated with rxd_[0-3](15:8) and rx_disp_err-detect_[0-3](0) associated with rxd_[0-3](7:0).

**rx_cv_detect_[0-3]** - Per channel, active high signal driven by the flexiPCS to indicate a code violation was detected with the associated data.

In 16 Bit Data Bus Mode, rx_cv_detect is 2 bits wide (**rx_cv_detect_[0-3](1:0)**) with rx_cv_detect_[0-3](1) associated with rxd_[0-3](15:8) and rx_cv_detect_[0-3](0) associated with rxd_[0-3](7:0).

**rx_scrm_en_[0-3]** - Per channel, active high PCI Express descrambler enable. Enables the descrambler on the first clock cycle after **rx_scrm_en** goes high. Disables the descrambler on the first clock cycle after **rx_scrm_en** goes low.

**rx_invert_[0-3]** - Per channel, active high signal that inverts the incoming data from the flexiPCS. Can be used to handle Lane Polarity Inversion as specified in the PCI Express Base Specification.

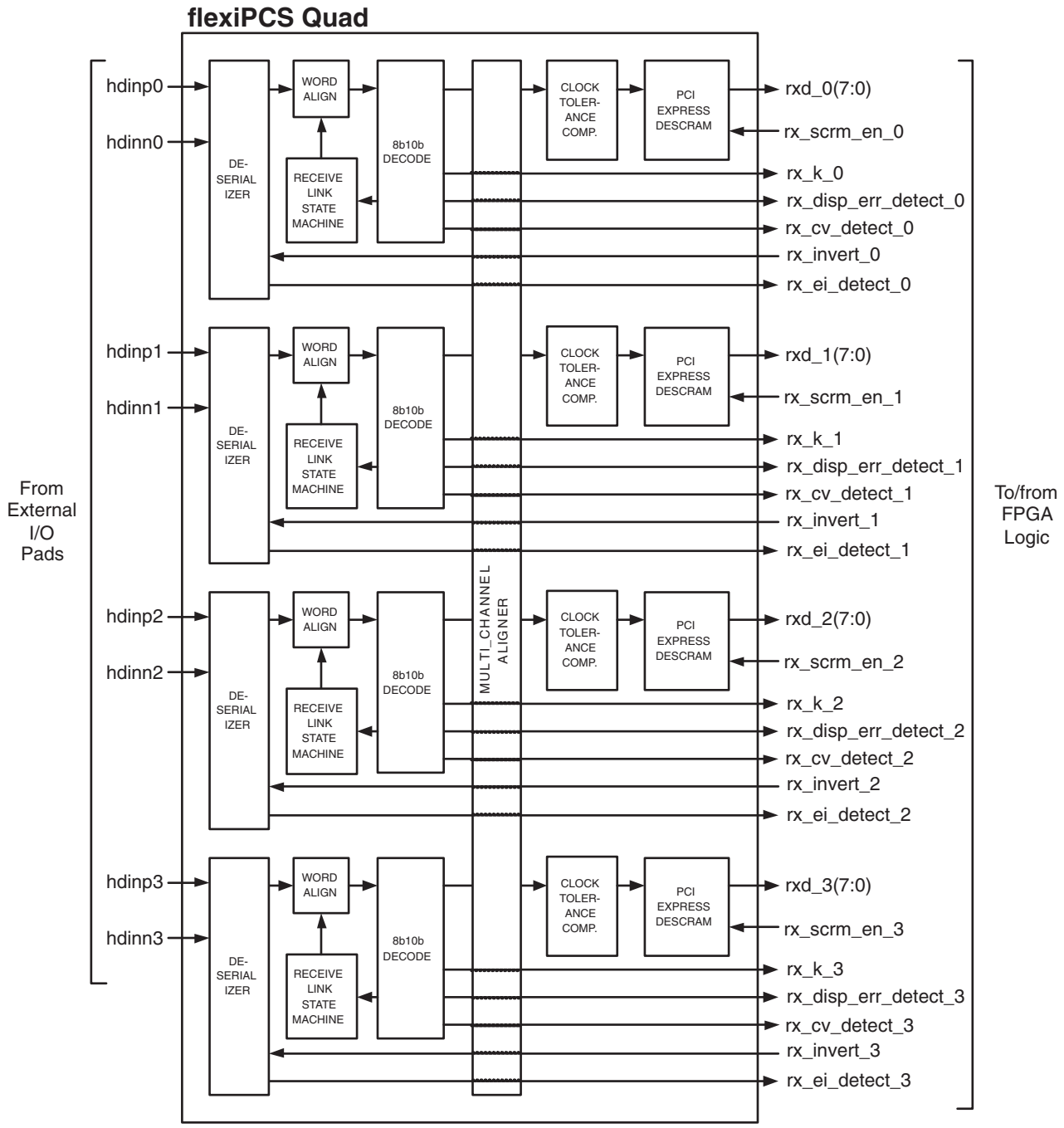
**rx_ei_detect_[0-3]** - Per channel, active high PCI Express Electrical Idle condition detection. This signal will go high when a loss of signal condition is detected on the hdin(p/n)_[0-3] I/O pad receiver inputs. The **rx_ei_detect** is derived from each channel's receive loss of signal (low value) flag. For correct operation of this signal, the SERDES receive data loss of signal levels must be set to the proper values. This is done by writing to the Quad Interface Register Offset Address 0x28, bits [5:7] as shown in Table 8-4.

***Table 8-4. Receiver Loss of Signal Detector Levels for Electrical Idle Detect***

| Quad Interface Register Offset Address = 0x28 | | | Loss of Signal Low Value (Max) | |
|---|---|---|---|---|
| **Bit 5** | **Bit 6** | **Bit 7** | **Threshold** | **Units** |
| 0 | 0 | 0 | 100 | mV, p-p differential |

The flexiPCS quads in PCI Express mode receive data path consists of the following sub-blocks per channel: Deserializer, Word Align, 8b10b Decode, & Link State Machine, Multi-channel Aligner, Clock Tolerance Compensator, and PCI Express Descrambler. Figure 8-6 shows the four channels of receive data in a flexiPCS quad set to PCI Express mode.

*Figure 8-6. PCI Express Receive Data Path (1 Quad)*




**Deserializer**

Data is brought on chip to the embedded SERDES where it undergoes serial to parallel conversion. For detailed information on the operation of the LatticeSC PCS SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

**Word Alignment**

The word aligner module performs the word alignment code word detection and alignment operation. The word alignment character is used by the receive logic to perform 10-bit word alignment upon the incoming data stream. The word alignment algorithm is described in Clause 36.2.4.9 in the 802.3-2002 1000BASE-X specification.

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

- 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 word alignment compare. Word alignment characters and the mask register are set on a per quad basis. For PCI Express, the word alignment characters should 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).
- The first word alignment character can be set by writing Quad Interface Register Offset Address 0x15 to 0x03.
- The second word alignment character can be set by writing Quad Interface Register Offset Address 0x16 to 0x7C.
- The mask register can be set to compare word alignment bits 6 to 0 by writing Quad Interface Register Offset Address 0x14 to 0x7F (a '1' in a bit of the mask register means check the corresponding bit in the word alignment character register).

**8b10b Decoder**
This module implements an 8b10b decoder as described in Appendix B of the PCI Express Base Specification, Revision 1.0a. The decoder performs the 10-bit to 8-bit code conversion as described the specification, along with verifying the running disparity rules as specified.

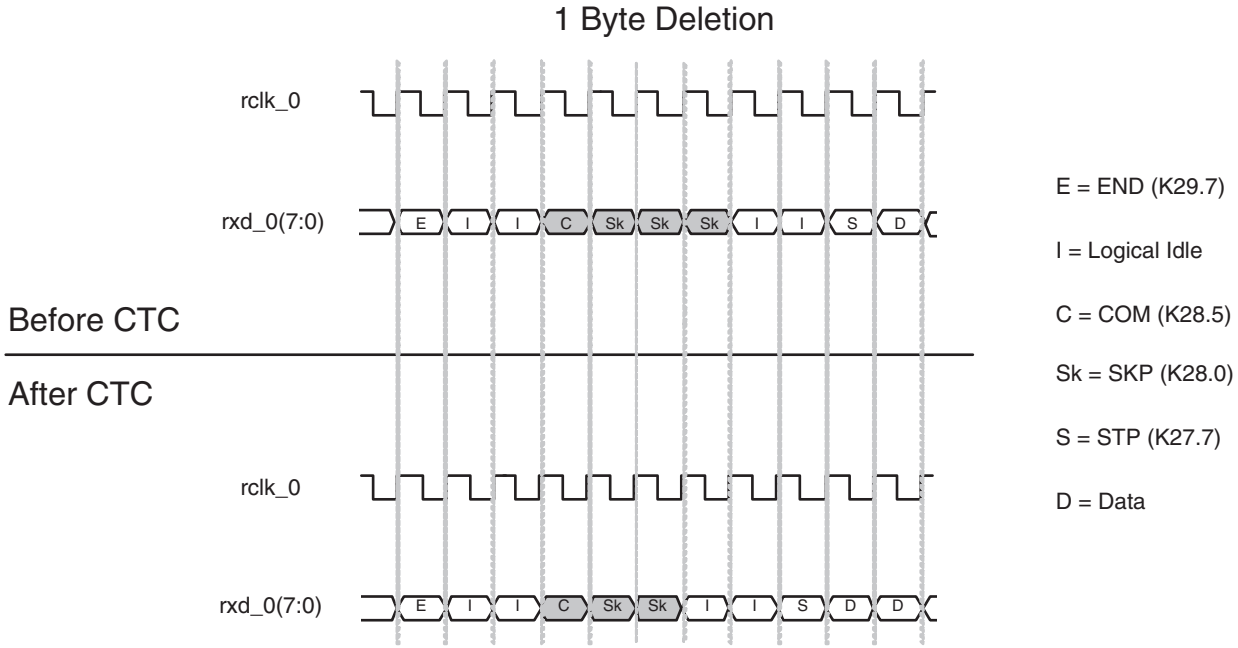
**Link Synchronization State Machine**
The link synchronization state machine is an extension of the word align module. Link synchronization is achieved after the successful detection and alignment of the required number of consecutive code words. The link synchronization state machine implements the PCS Synchronization State Diagram (Figure 48-7) of the 802ae.3-2002 specification.

**Multi-Channel Aligner**
The multi-channel aligner can be used to align two or more PCI Express receive channels. When a LatticeSC flexiPCS quad is set to PCI Express Mode, the multi-channel aligner can be used to create up to two independent x2 PCI Express links or one x4 PCI Express link. Channels in multiple quads can be aligned to create one or more x8 PCI Express links. More information can be found in the **x2 PCI Express Operation, x4 PCI Express Operation, or x8 PCI Express Operations** sections. Additional information on multi-quad alignment for creating x8 PCI Express (and wider) links can be found in the **Multi-channel Alignment** section of the **LatticeSC/M Family flexiPCS Data Sheet**.

**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 LatticeSC SERDES (See DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet).

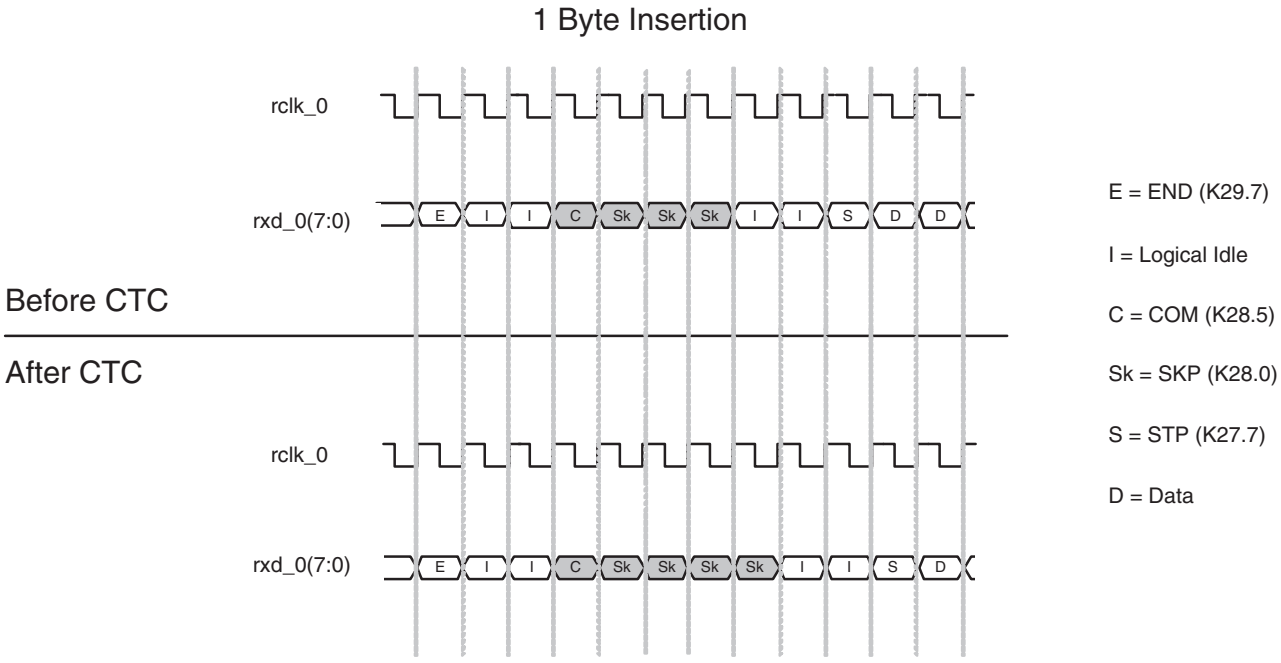
A diagram illustrating 1 byte deletion is shown in Figure 8-7:

***Figure 8-7. PCI Express Mode Clock Compensation Byte Deletion Example***




A diagram illustrating 1 byte insertion is shown in Figure 8-8:

***Figure 8-8. PCI Express Mode Clock Compensation Byte Insertion Example***




Clock compensation values are set on a quad basis. The following settings for clock compensation should be set for PCI Express applications:

- Set the insertion/deletion pattern length to 1 and minimum inter-packet gap to 1 bytes by setting Quad Interface Register Offset Address 0x0E to 0x00. The minimum allowed inter-packet gap after skip character deletion

based on these register settings is described below.

- The PCI Express insertion/deletion pattern should be set to the /SKP/ character (K28.0). When 1 byte insertion/deletion is selected, the SKP symbol can be set by writing Quad Interface Register Offset Address 0x12 to 0x1C and Quad Interface Register Offset Address 0x13 to 0x01.
- Set the clock compensation FIFO high water and low water marks to 9 and 7 respectively (appropriate for insertion/deletion pattern length of 1) by setting Quad Interface Register Offset Address 0x0D to 0x97.
- Clock compensation FIFO underrun can be monitored by reading the appropriate Channel Interface Register Offset Address 0x85, bit 6. This can be also monitored on the flexiPCS interface pin to FPGA logic labeled ctc_urun_[0-3] if “Optional Direct Control & Status Register Access” is selected when generating the flexiPCS block with IPexpress.
- Clock compensation FIFO overrun can be monitored by reading the appropriate Channel Interface Register Offset Address 0x85, bit 7. This can be also monitored on the flexiPCS interface pin to FPGA logic labeled ctc_orun_[0-3] if “Optional Direct Control & Status Register Access” is selected when generating the flexiPCS block with IPexpress.

**Calculating Minimum Inter-packet Gap**

Table 8-5 shows the relationship between the user-defined values for inter-packet gap (written to Quad Interface Register Offset Address 0x0E, bits 6 and 7), and the guaranteed minimum number of bytes between packets after a skip character deletion from the flexiPCS. 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 PCI Express, the number of bytes per insertion/deletion is 1 (Quad Interface Register Offset Address 0x0E, bits 4 and 5 are both set to ‘0’). If Quad Interface Register Offset Address 0x0E, bits 6 and 7 are set to “00”, then the minimum inter-packet gap is equal to 1 times 1 or 1 byte. The flexiPCS will not perform a skip character deletion until the minimum number of inter-packet bytes have been detected.

***Table 8-5. Minimum inter-packet gap multiplier***

| Quad Interface Register Offset Address 0x0E | | Insertion/ Deletion Multiplier Factor |
|---|---|---|
| Bit 6 | Bit 7 | |
| 0 | 0 | 1 X |
| 0 | 1 | 2 X |
| 1 | 0 | 3 X |
| 1 | 1 | 4 X |

**PCI Express Descrambler**

The PCI Express Descrambler reverses the operation performed by the PCI Express scrambler.

## Control & Status

The Control & Status pins for the PCI Express mode can be optionally selected on a per quad basis when generating the flexiPCS quad interface files with the ispLEVER tools. Each of these control and status functions are also accessible through equivalent Quad Interface and Channel Interface Registers. The control and status signals allow direct real time access of these functions from FPGA logic.

**Control**

**felb_[0-3]** - Per channel, active high control signal which sets up the appropriate channel in Far-End Loopback mode. This mode is generally used for testing the flexiPCS logic, looping back receive data back to the transmit direction without crossing the FPGA logic interface. For more information on the details of Far-End Loopback mode, see the **flexiPCS Testing** Section of the flexiPCS Data Sheet. The Far-End Loopback mode can also be initiated by writing Channel Interface Register Offset Address 0x00, bit 5 to ‘1’.

**lsm_en_[0-3]** - Per channel Receive Link State Machine Enable. A change (either 0 to 1 or 1 to 0) on the lsm_en_[0-3] resets the Receive Link State Machine into No Sync mode. The Receive Link Machine Enable can also be set by writing Channel Interface Register Offset Address 0x00, bit 7 to '1'.

**mca_align_en_[0-3] -** Per channel active high Multi-channel Align enable. Multi-channel Alignment Enable can also be set by writing the appropriate Quad Interface Register Offset Address 0x04, bits [4:7] to '1' (bit 4 for channel 3, bit 5 for channel 2, bit 6 for channel 1, and bit 7 for channel 0).

**mca_resync_[01/23]** - Active high, asynchronous reset to Multi-channel Aligner state machine and aligner FIFO control logic. **mca_resync_01** resets logic in channels 0 and 1 in two-channel alignment mode and all four channels in four channel alignment mode. **mca_resync_23** resets logic in channels 2 and 3 in two-channel alignment mode (not used in four channel alignment mode).

**Status**

**lsm_status_[0-3]** - Per channel active high signal from the Receive Link State Machine indicating link synchronization successful. The link synchronization status can also be read from the appropriate Quad Interface Register Offset Address 0x84, bits [4:7] (bit 4 for channel 0, bit 5 for channel 1, bit 4 for channel 2, and bit 7 for channel 3).

**mca_aligned_[01/23]** - Active high signal indicating successful alignment of channels 0/1 or channels 2/3. mca_aligned_01 high indicates successful alignment of channels 0 and 1 in two-channel alignment mode and all four channels in four channel alignment mode. mca_aligned_23 high indicates successful alignment of channels 2 and 3 in two-channel alignment mode (Always low in four channel alignment mode). The Multi-channel Alignment status for channels 0/1 (equivalent to the mca_aligned_01 output) can also be read from the Quad Interface Register Offset Address 0x82, bit 2. The Multi-channel Alignment status for channels 2/3 (equivalent to the mca_aligned_23 output) can also be read from the Quad Interface Register Offset Address 0x82, bit 3. mca_aligned_01 is only valid when performing multichannel alignment within a single quad.

**ctc_urun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has underrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 6.

**ctc_orun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has overrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 7.

## x2 PCI Express Operation

A single LatticeSC quad can support up to two aligned x2 PCI Express links with the use of the embedded multi-channel aligner. This is done by using a flexiPCS quad in PCI Express Mode and setting the Multi-channel Aligner registers appropriately.

**Multi-channel Aligner**

The Multi-channel alignment module performs the operations and functions to control the multi-channel alignment process. This includes the /A/ detect and the alignment status signal generation required within the implementation of the alignment state machine as described in the PCS Deskew State Diagram (Figure 48-8) in the IEEE 802ae.3-2002 10GBASE-X standard.

The flexiPCS Multi-channel Aligner allows for alignment of two or four channels in a quad. For x2 PCI Express operation, alignment is performed for two channels in a quad. The Multi-channel Aligner will align channels based on a user-defined alignment character (COM for PCI Express applications) which must be present in the data stream for all receive channels that are to be aligned. This character is inserted simultaneously in all channels to be aligned during an Idle sequence between packets upon transmission. If arriving data is skewed due to unequal transport delays, the presence of the alignment characters allows the Multi-channel Aligner to re-align the skewed channels.

When alignment is enabled for PCI Express operation:

1. For x2 PCI Express operation, at least 2 channels must be enabled (Channel 0 and Channel 1 and/or Channel 2 and Channel 3).

2. For x2 PCI Express operation, the Multi-channel Aligner is enabled in the Receive direction to provide alignment between two channels as dictated by the COM lane align symbol as defined in the PCI Express Base Specification.

3. For x2 PCI Express operation, the clock tolerance compensation logic always inserts or deletes across the appropriate two columns (0 and 1 and/or 2 and 3) simultaneously to preserve multi-channel alignment.

The following is a procedure for settings registers for Multi-channel Alignment for x2 PCI Express.

1. Set the mode for the Multi-channel Aligner and enable/disable the appropriate channels for alignment. The multi-channel aligner can be set to align two channels in a quad by writing to Quad Interface Register Offset Address 0x19.

- Bit 3 controls alignment between Channel 0 and Channel 1. When bit 3 is set to '1', Channels 0 and 1 will be aligned. Figure 8-9 shows an example of the operation of the flexiPCS multi-channel aligner operation for channel 0 and channel 1 alignment. The COM code-groups in each channel are lined up by the multi-channel aligner to create an aligned data stream at the PCS/FPGA interface.
- Bit 2 controls alignment between Channel 2 and Channel 3. When bit 2 is set to '1', Channels 2 and 3 will be aligned.
- When both bit 2 and bit 3 are both set to '1', channels 0 and 1 are aligned to each other and channels 2 and 3 are aligned to each other, but channels 0/1 and channels 2/3 are not aligned together. Figure 8-10 shows an example of the operation of the flexiPCS multi-channel aligner operation when channel 0 and 1 alignment and channel 2 and 3 alignment are both selected.
- Each channel to be aligned must also be enabled for alignment by writing to the appropriate bit in Quad Interface Register Offset Address 0x04. Write a '1' to bit 7 to include channel 0 for alignment. Write a '1' to bit 6 to include channel 1 for alignment. Write a '1' to bit 5 to include channel 2 for alignment. Write a '1' to bit 4 to include channel 3 for alignment. Multi-channel alignment can also be enabled by setting the appropriate **mca_align_en-[0-3]** ports at the FPGA interface high.

***Figure 8-9. x2 PCI Express Alignment Example (Channels 0 and 1)***

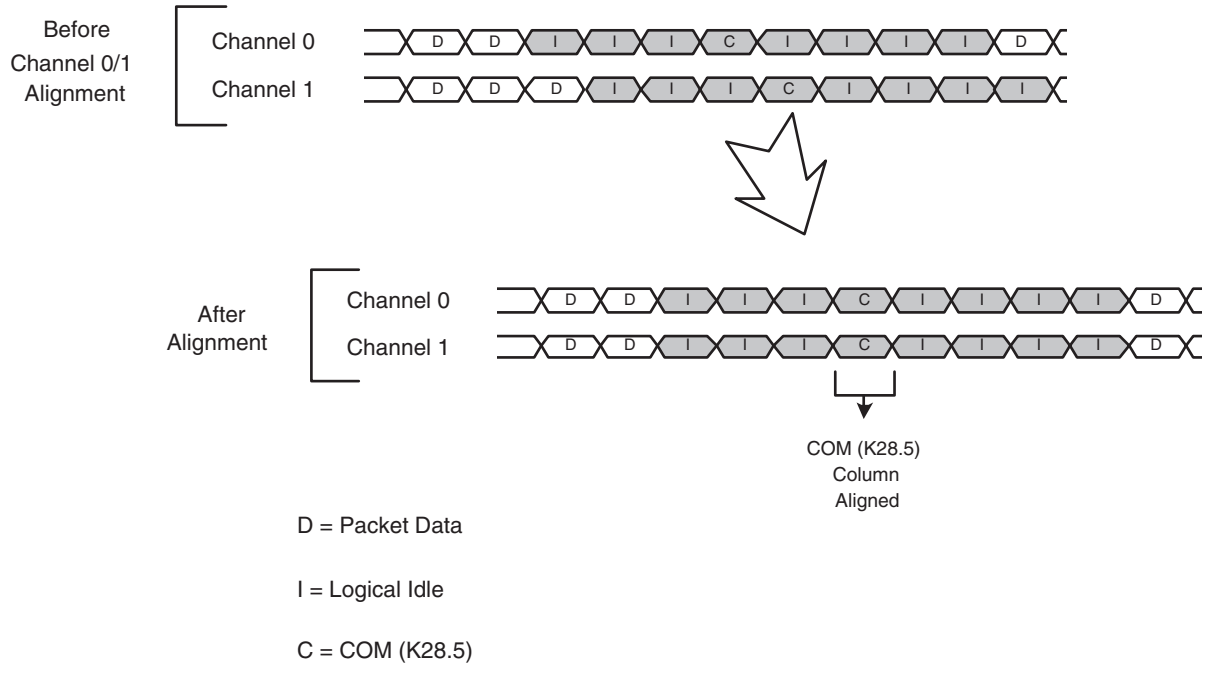
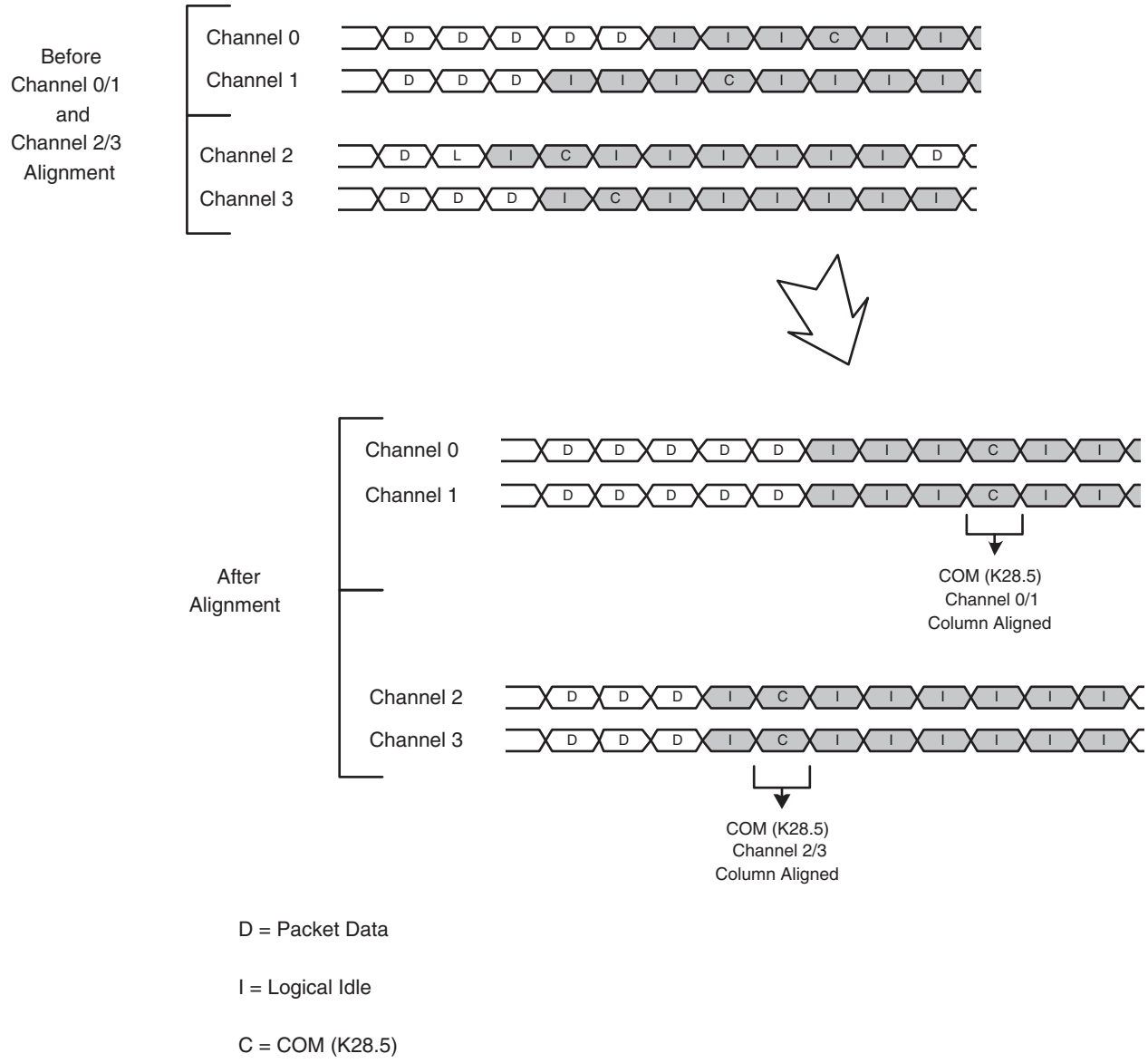


***Figure 8-10. Dual x2 PCI Express Alignment Example (channels 0/1 and channels 2/3)***




2. Set the Multi-channel Alignment output clocks. A clock domain transfer occurs between the inputs and outputs of the Multi-channel Aligner. Each channel's receive data is clocked into the Multi-channel Aligner with its own separate recovered receive clock. Each channel's receive data is clocked out of the Multi-channel Aligner on a unique multi-channel receive clock. Each multi-channel receive clock can be programmed to be any of the recovered receive clocks.

It is expected that the user will assign the same recovered receive clock to all the multi-channel output clocks for every channel that is to be aligned. That way, all aligned channels coming out of the Multi-channel Aligner are effectively clocked by the same clock. For more information on setting up a multi-channel receive clock, refer to the **Multi-Channel Alignment** section of the LatticeSC/M Family flexiPCS Data Sheet.

For example, in a 2 channel alignment application (Channels 0/1), in order to set up the Multi-channel Alignment clocks for channels 0 and 1 to be sourced from the channel 0 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xE0. For a 2 channel alignment application (Channels 2/3), in order to set up the Multi-channel Alignment clocks for channels 2 and 3 to be sourced from the channel 2 recovered receive clock, set Quad

Interface Register Offset Address 0x01 to 0xA4. For a dual 2 channel alignment application (Channels 0/1 and channels 2/3), in order to set up the Multi-channel Alignment clocks for channels 0 and 1 to be sourced from the channel 0 recovered receive clock and the Multi-channel Alignment clocks for channels 2 and 3 to be sourced from the channel 2 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xA0.

3. Set the Multi-channel Alignment characters. The Multi-channel Aligner provides the ability to align channels of incoming data to one of two 10-bit user defined maskable patterns. Both alignment characters should be set to the same value if only one alignment character is defined for an application. For x2 PCI Express operation, the alignment character should be set to COM (K28.5).

- A user programmable mask word allows the user to set which individual bits are compared to the user defined channel alignment character. When a '1' is present in the user programmable mask, the corresponding bit of the channel alignment character is compared. When '0' is present in the user programmable mask, the corresponding bit of the channel alignment character is not compared. For x2 PCI Express operation, set the lowest 9 significant bits of the user programmable mask by writing Quad Interface Register Offset Address 0x07 to 0xFF and Quad Interface Register Offset Address 0x0A, bit 3 to '1'.
- The first channel alignment character can be set to the PCI Express Align column value COM (K28.5) by writing Quad Interface Register Offset Address 0x08 to 0xBC. The first channel K character can be set by writing Quad Interface Register Offset Address 0x0A, bit 7 to '1'.
- The second channel alignment character can be set to the PCI Express Align column value COM (K28.5) by writing Quad Interface Register Offset Address 0x09 to 0xBC. The second channel K character can be set by writing Quad Interface Register Offset Address 0x0A, bit 5 to '1'.

4. Set the Multi-channel Aligner FIFO latency and high-water mark values. Latency values from 0 to 31 can be set by writing to Quad Interface Register Offset Address 0x05, bits [3:7]. The FIFO high-water mark values from 0 to 63 can be set by writing to Quad Interface Register Offset Address 0x06, bits [2:7]. The FIFO high-water mark should be set to the maximum the number of bytes of skew allowed for de-skewing. Larger values of FIFO high-water mark increase the chance of FIFO overrun or underrun. For more information on choosing latency and high-water mark values, refer to the Multi-Channel Alignment section of the LatticeSC/M Family flexiPCS Data Sheet.

5. Reset the Multi-channel Aligner state machine and FIFO control logic if desired with the **mca_resync_[01/23]** ports at the FPGA interface. **mca_resync_01** is an active high asynchronous reset which resets the Multi-channel Aligner for channels 0 and 1. **mca_resync_23** is an active high asynchronous reset which resets the Multi-channel Aligner for channels 2 and 3.

**Clock Tolerance Compensation**

When insertion/deletion is performed by the clock tolerance compensation logic for x2 PCI Express operation, insertion or deletion is performed across both aligned channels simultaneously (to preserve the multi-channel alignment).

Note that when two channel alignment for channels 0 and 1 is selected, channel 0 is selected as the master channel for performing clock tolerance compensation. This means that only channel 0 is monitored for the presence of the SKIP character. Insertion or deletion is performed on channels 0 and 1 simultaneously based on detection of the SKIP character in channel 0. This means that channel 0 must be active and transmitting PCI Express compliant data for the clock tolerance compensation function to work in two channel alignment mode for channels 0 and 1. If channel 0 is not active, receive data will not be present on the rxd_0 and rxd_1 outputs of the flexiPCS.

When two channel alignment for channels 2 and 3 is selected, channel 2 is selected as the master channel for performing clock tolerance compensation. This means that only channel 2 is monitored for the presence of the SKIP character. Insertion or deletion is performed on channels 2 and 3 simultaneously based on detection of the SKIP character in channel 2. This means that channel 2 must be active and transmitting PCI Express compliant data for the clock tolerance compensation function to work in two channel alignment mode for channels 2 and 3. If channel 2 is not active, receive data will not be present on the rxd_2 and rxd_3 outputs of the flexiPCS.

## x4 PCI Express Operation

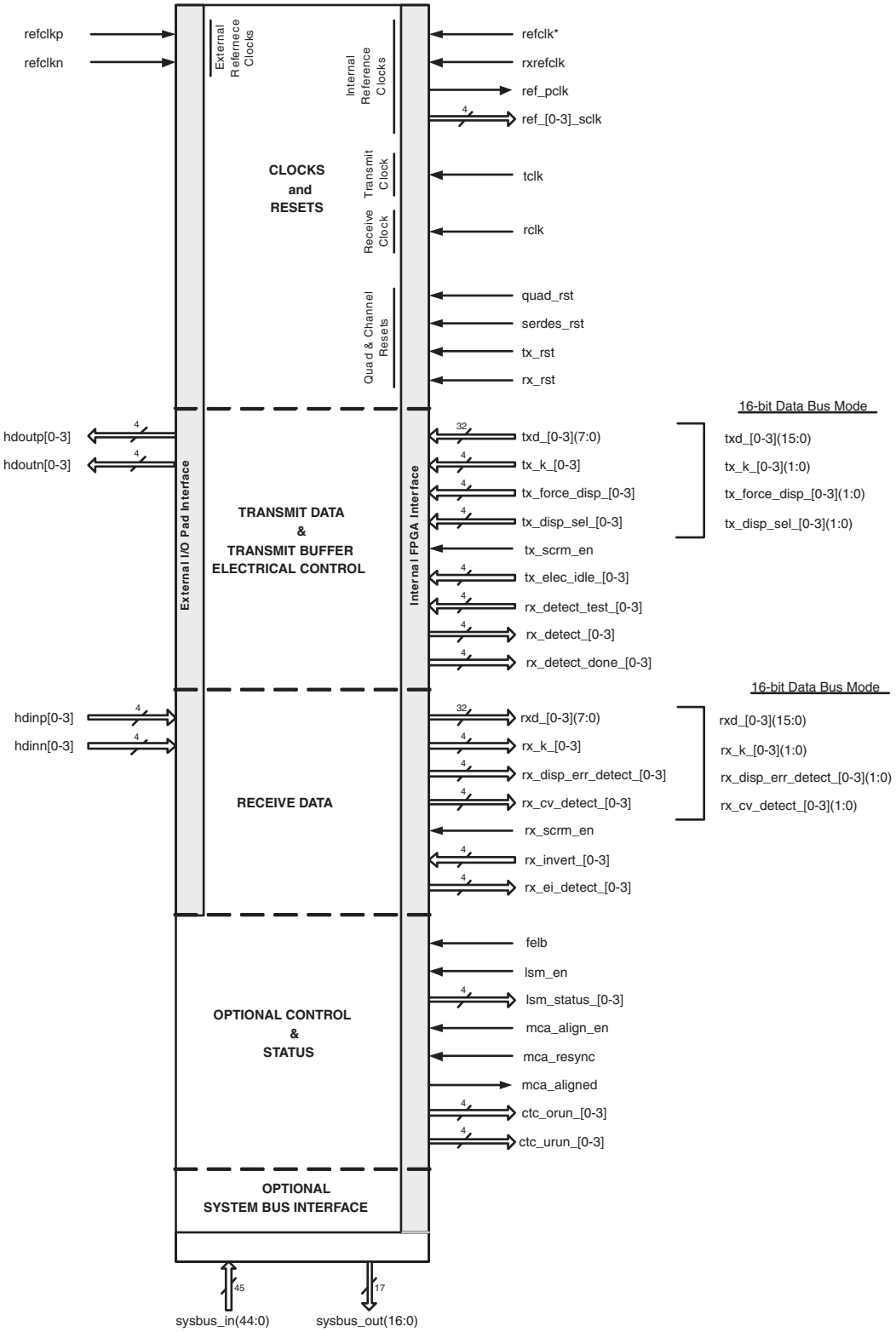
A single LatticeSC quad can support a single x4 PCI Express link with the use of the embedded multi-channel aligner. This is done by using a flexiPCS quad in PCI Express Mode and setting the Multi-channel Aligner block appropriately.

The main differences between x2 and x4 PCI Express operation are:

1. For x4 PCI Express operation, all four channels in a flexiPCS QUAD must be enabled.
2. For x4 PCI Express operation, the Multi-channel Aligner is enabled in the Receive direction to provide alignment between all four channels as dictated by the Align column symbol (COM) as defined in the PCI Express specification.
3. For x4 PCI Express operation, the clock tolerance compensation logic always inserts or deletes across all four columns simultaneously to preserve multi-channel alignment.

IPexpress allows the designer to choose the mode for each flexiPCS quad used in a given design. A minimized pinout for x4 PCI Express operation is provided by IPexpress. In general, this means that all per channel input pins at the FPGA interface are tied together in the flexiPCS and presented as one pin. Figure 8-11 displays the resulting port interface to one flexiPCS quad that has been set to PCI Express mode with “Align channels 0,1,2, and 3” selected with IPexpress.

***Figure 8-11. PCI Express Mode Pin Diagram (Four channel alignment selected)***




* The refclk inputs for all active quads on a device must be connected to the same clock. The rxrefclk inputs for all active quads on a device do not have to be connected to the same clock.

## x4 PCI Express Operation Pin Description

Table 8-6 lists the inputs and outputs to/from a flexiPCS quad targeted to x4 PCI Express operation by selecting "Align channels 0, 1, 2, and 3" in IPexpress. A more detailed description of the function of each additional port is given in the **x4 PCI Express Operation Detailed Description**.

***Table 8-6. PCI Express Mode Pin Description (Four channel alignment selected)***

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Reference Clocks** | | | |
| refclkp | In from I/O pad | N/A | Reference clock input, positive. Dedicated CML input. |
| refclkn | In from I/O pad | N/A | Reference clock input, negative. Dedicated CML input |
| refclk | In from FPGA | N/A | Optional reference clock input from FPGA logic. Can be used instead of per quad reference clock.<br><br>The refclk inputs for all active quads on a device must be connected to the same clock. |
| rxrefclk | In from FPGA | N/A | Optional receive reference clock input from FPGA logic. Can be used instead of per quad reference clock. The rxrefclk inputs for all active quads do not have to be connected to the same clock. |
| ref_pclk | Out to FPGA | N/A | Locked reference clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| ref_[0-3]_sclk | Out to FPGA | N/A | Per channel locked reference clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| **Transmit Clock** | | | |
| tclk | In from FPGA | N/A | Transmit clock input from FPGA. Used to clock the TX data phase compensation FIFO with clock synchronous to the clock. May also be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Receive Clock** | | | |
| rclk | In from FPGA | N/A | Receive clock input from FPGA. May be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference and/or receive reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Resets** | | | |
| quad_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of SERDES and PCS logic. |
| serdes_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of the SERDES. |
| tx_rst | In from FPGA | ASYNC | Active high, asynchronous reset of all four transmit channels of PCS logic. |
| rx_rst | In from FPGA | ASYNC | Active high, asynchronous reset of all four receive channels of PCS logic. |
| **Transmit Data & Transmit Buffer Electrical Control** | | | |
| hdoutp[0-3] | Out to I/O pad | N/A | High-speed CML serial output, positive. |
| hdoutn[0-3] | Out to I/O pad | N/A | High-speed CML serial output, negative. |
| txd_[0-3](7:0)<br><br>txd_[0-3](15:0)* | In from FPGA | tclk_[0-3] | Per channel parallel transmit data bus from the FPGA. Bus is 8-bits wide if QIR 0x19, bit 4 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 4 = '1'. |

*Table 8-6. PCI Express Mode Pin Description (Four channel alignment selected)*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| tx_k_[0-3]<br><br>tx_k_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel transmit control word indicator.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_force_disp_[0-3]<br><br>tx_force_disp_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel active high, force disparity enable.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_disp_sel_[0-3]<br><br>tx_disp_sel_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel disparity select. High forces positive disparity, low forces negative disparity.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| tx_scrm_en | In from FPGA | ASYNC | Active high PCI Express scrambler enable for all four channels of txd data. |
| tx_elec_idle_[0-3] | In from FPGA | ASYNC | Per channel, active high electrical idle control. |
| rx_detect_test_[0-3] | In from FPGA | ASYNC | Per channel, active high Receiver Detection test input. |
| rx_detect_[0-3] | Out to FPGA | ASYNC | Per channel, Receiver Detect test result. |
| rx_detect_done_[0-3] | Out to FPGA | ASYNC | Per channel, Receiver Detect test done indicator. |
| **Receive Data** | | | |
| hdinp[0-3] | In from I/O pad | N/A | High-speed CML serial input, positive. |
| hdinn[0-3] | In from I/O pad | N/A | High-speed CML serial input, negative. |
| rxd_[0-3](7:0)<br><br>rxd_[0-3](15:0)* | Out to FPGA | rclk_[0-3] | Per channel parallel receive data bus to the FPGA. Bus is 8-bits wide if QIR 0x19, bit 5 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 5 = '1'. |
| rx_k_[0-3]<br><br>rx_k_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel receive control word indicator.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_disp_err_detect_[0-3]<br><br>rx_disp_err_detect_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high disparity error detection.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_cv_detect_[0-3]<br><br>rx_cv_detect_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high code violation detection.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_scrm_en | In from FPGA | ASYNC | Active high PCI Express descrambler enable for all four channels for rxd data. |
| rx_invert_[0-3] | In from FPGA | ASYNC | Per channel, receive data invert. |
| rx_ei_detect_[0-3] | Out to FPGA | ASYNC | Per channel, Electrical Idle condition detect. |
| **Optional Control & Status** | | | |
| felb | In from FPGA | ASYNC | Active high far-end loopback enable for all four channels. |
| lsm_en | In from FPGA | ASYNC | Receive link state machine enable for all four channels. |
| lsm_status_[0-3] | Out to FPGA | ASYNC | Per channel signal from receive link state machine indicating successful link synchronization. |
| mca_align_en | In from FPGA | ASYNC | Active high multi-channel aligner enable for all four channels. |
| mca_resync | In from FPGA | ASYNC | Active high async multi-channel aligner resynchronization signal. |
| mca_aligned | Out to FPGA | ASYNC | Active high indicating successful alignment of channels. |
| ctc_urun_[0-3] | Out to FPGA | ASYNC | Per channel active high flag indicator that clock tolerance compensation FIFO has underrun. |
| ctc_orun_[0-3] | Out to FPGA | ASYNC | Per channel active high flog indicator that clock tolerance compensation FIFO has overrun. |
| **Optional System Bus Interface** | | | |

*Table 8-6. PCI Express Mode Pin Description (Four channel alignment selected)*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| sysbus_in(44:0) | In from FPGA | N/A | Control and data signals from the internal system bus. |
| sysbus_out(16:0) | Out to FPGA | N/A | Control and data signals to the internal system bus. |

# x4 PCI Express Operation Detailed Description

The functionality of a LatticeSC flexiPCS quad targeted to x4 PCI Express operation is identical in most respects to that of a flexiPCS quad set to PCI Express Mode. See the PCI Express Mode Detailed Description for more information regarding the operation of the LatticeSC flexiPCS in PCI Express Mode. x4 PCI Express operation is different in the use of the Multi-channel aligner and the Clock Tolerance Compensation logic. The operation of these blocks for x4 PCI Express operation is described below.

## Clocks & Resets

Clocking setup for a flexiPCS quad targeted to x4 PCI Express operation is generally the same as that of a quad set to PCI Express mode. Refer to the **Clocks & Resets** description in the PCI Express **Mode Detailed Functional Description** section. A summary of the clocks that are different for x4 PCI Express operation are listed below:

**tclk** - One transmit clock input from the FPGA logic to the flexiPCS is provided for all four channels of data (rather than a separate clock for each channel).

**rx_pclk** - One receive clock output from the flexiPCS to the FPGA logic is provided for connection to a primary clock route. This clock is derived from the recovered clock on Channel 0 by default.

**rclk** - One receive clock input from the FPGA logic to the flexiPCS is provided for all four channels of data (rather than a separate clock for each channel).

For x4 PCI Express applications, the bit clock rate is typically 2.5 Gbps. This bit clock rate falls into the range defined as “full rate” mode for the SERDES PLLs. Full rate mode is selected separately for each channel and each direction by writing the appropriate Channel Interface Register Offset Address 0x13, bit 5 (transmit) and bit 6 (receive) to a ‘0’ (default).

Table 8-3 in the **PCI Express Mode Detailed Description** section shows the possible reference clock frequencies for various reference clock modes which result in a 2.5 Gbps internal bit rate.

Additional information on all reference clock rate modes can be found in the **SERDES Functionality** section of the flexiPCS Data Sheet.

### Quad & Channel Resets

Resets for a flexiPCS quad targeted to x4 PCI Express operation are generally the same as that of a quad set to PCI Express mode. Refer to the **Clocks & Resets** description in the PCI Express **Mode Detailed Functional Description** section. A summary of the channel resets that are different for x4 PCI Express operation are listed below:

**tx_rst** - Active high, asynchronous signal from FPGA resets all four transmit channels of PCS logic. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [4:7]. Bit 7 resets transmit channel 0, bit 6 resets transmit channel 1, bit 5 resets transmit channel 2, and bit 4 resets transmit channel 3.

**rx_rst** - Active high, asynchronous signals from FPGA resets all four receive channels of PCS logic. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [0:3]. Bit 3 resets receive channel 0, bit 2 resets receive channel 1, bit 1 resets receive channel 2, and bit 0 resets receive channel 3.

## Transmit Data

When configured for x4 PCI Express operation, a flexiPCS quad provides up to four transmit data paths. Each transmit data path converts a 32-bit wide PCI Express compliant data stream at the FPGA logic interface to a high-speed serial line interface at the device outputs.

## Receive Data

When configured for x4 PCI Express operation, a flexiPCS quad provides a receive data path from a high-speed serial line interface at the device inputs to four 8-bit parallel interfaces at the FPGA logic interface by default.

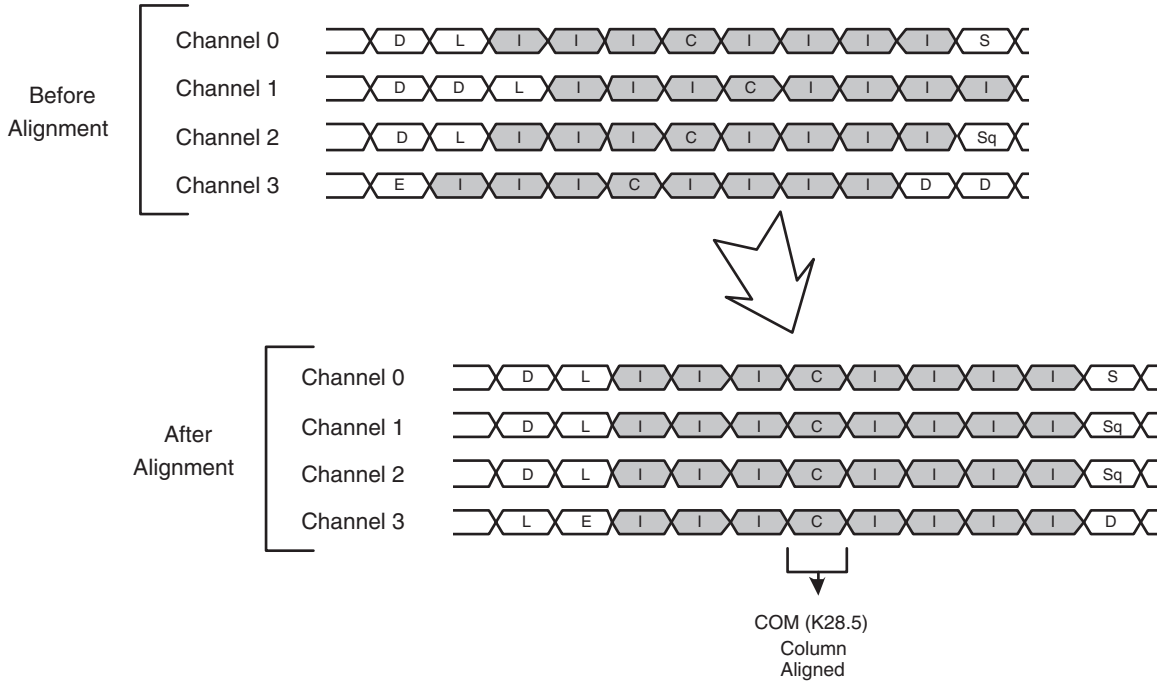
**Multi-channel Aligner**

The Multi-channel alignment module performs the operations and functions to control the multi-channel alignment process. This includes the ||A|| detect and the alignment status signal generation required within the implementation of the alignment state machine as described in the PCS Deskew State Diagram (Figure 48-8) in the IEEE 802ae.3-2002 10GBASE-X standard.

The flexiPCS Multi-channel Aligner allows for alignment of two, three, or four channels in a quad. For x4 PCI Express operation, alignment must be performed for all four channels in a quad. The Multi-channel Aligner will align channels based on a user-defined alignment character (referred to as /A/) which must be present in the data stream for all receive channels that are to be aligned. Typically this character is inserted simultaneously in all channels during an Idle sequence between packets upon transmission. If arriving data is skewed due to unequal transport delays, the presence of the alignment characters allows the Multi-channel Aligner to re-align the skewed channels.

Figure 8-12 shows an example of the operation of the flexiPCS multi-channel aligner operation for x4 PCI Express applications. The /A/ Align code-groups in each channel are lined up by the multi-channel aligner to create an aligned data stream at the PCS/FPGA interface.

***Figure 8-12. x4 PCI Express Four Channel Alignment Example***




D = Data

L = LCRC

E = END (K29.7)

I = Logical Idle

C = COM (K28.5)

S = STP (K27.7)

Sq = Sequence Number

The following is a procedure for settings registers for Multi-channel Alignment.

1. Set the mode for the Multi-channel Aligner and enable/disable the appropriate channels for alignment.

- For x4 PCI Express operation, set 4 channel alignment for channels 0, 1, 2, and 3 by writing Quad Interface Register Offset Address 0x19, bit 1 to a '1'.
- Each channel to be aligned must also be enabled for alignment by writing to the appropriate bit in Quad Interface Register Offset Address 0x04. Write a '1' to bit 7 to include channel 0 for alignment. Write a '1' to bit 6 to include channel 1 for alignment. Write a '1' to bit 5 to include channel 2 for alignment. Write a '1' to bit 4 to include channel 3 for alignment. Multi-channel alignment can also be enabled by setting the **mca_align_en** pin at the FPGA interface high.

2. Set the Multi-channel Alignment output clocks. A clock domain transfer occurs between the inputs and outputs of the Multi-channel Aligner. Each channel's receive data is clocked into the Multi-channel Aligner with its own separate recovered receive clock. Each channel's receive data is clocked out of the Multi-channel Aligner on a unique multi-channel receive clock. Each multi-channel receive clock can be programmed to be any of the recovered receive clocks.

It is expected that the user will assign the same recovered receive clock to all the multi-channel output clocks for every channel that is to be aligned. That way, all aligned channels coming out of the Multi-channel Aligner are effectively clocked by the same clock. For more information on setting up a multi-channel receive clock, refer to the **Multi-Channel Alignment** section of the LatticeSC/M Family flexiPCS Data Sheet.

For example, in a 4 channel alignment application, in order to set up all Multi-channel Alignment clocks to be sourced from the channel 0 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0x00. To set all Multi-channel Alignment clocks to be sourced from the channel 1 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0x55. To set all Multi-channel Alignment clocks to be sourced from the channel 2 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xAA. To set all Multi-channel Alignment clocks to be sourced from the channel 3 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xFF.

3. Set the Multi-channel Alignment characters as indicated in step 3 of the **x2 PCI Express Operation** section.

4. Set the Multi-channel Aligner FIFO latency and high-water mark values as indicated in step 4 of the **x2 PCI Express Operation** section.

5. Reset the Multi-channel Aligner state machine and FIFO control logic if desired with the **mca_resync** pin at the FPGA interface. **mca_resync** is an active high asynchronous reset which resets the Multi-channel Aligner for all four channels.

When the Multi-channel Aligner is set to 4-channel alignment, the Multi-channel Alignment state machine for all four channels can be disabled by writing Quad Interface Register Offset Address 0x04, bit 3 to a '1'. When the Multi-channel Alignment state machine is disabled, the alignment state machine will be held in the "LOSS_OF_ALIGNMENT" state and no alignment will be performed for the relevant channels.

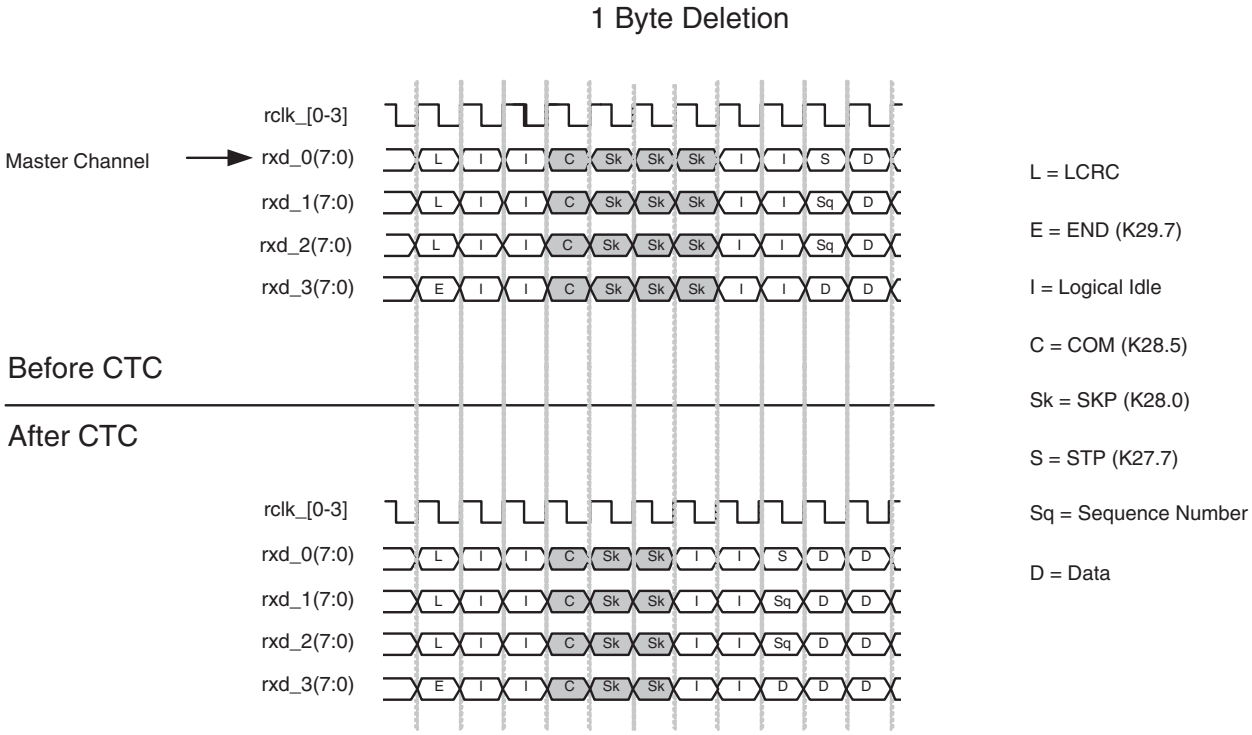
**Clock Tolerance Compensation**

When insertion/deletion is performed by the clock tolerance compensation logic for x4 PCI Express operation, insertion or deletion is performed across all aligned channels simultaneously (to preserve the multi-channel alignment).

Note that when four channel alignment is selected, channel 0 is selected as the master channel for performing clock tolerance compensation. This means that only channel 0 is monitored for the presence of the SKIP character. Insertion or deletion is performed on all four channels simultaneously based on detection of the SKIP character in channel 0. This means that channel 0 must be active and transmitting PCI Express compliant data for the clock tolerance compensation function to work in four channel alignment mode. If channel 0 is not active, receive data will not be present on the rxd outputs of the flexiPCS.

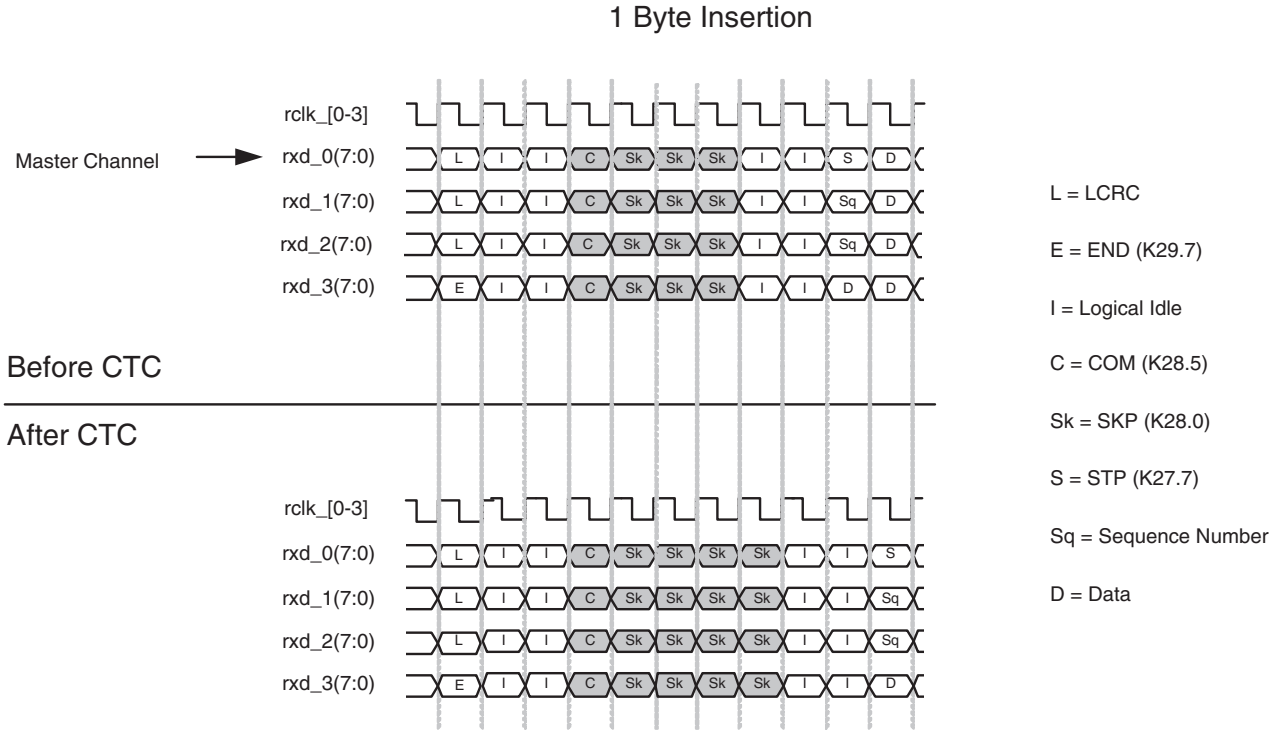
A diagram illustrating 1 byte deletion for x4 PCI Express operation is shown in Figure 8-13:

***Figure 8-13. x4 PCI Express Operation Clock Compensation Byte Deletion Example***




A diagram illustrating 1 byte insertion for x4 PCI Express operation is shown in Figure 8-14:

***Figure 8-14. x4 PCI Express Operation Clock Compensation Byte Insertion Example***




## x8 PCI Express Operation

Two LatticeSC flexiPCS quads can support one x8 PCI Express link with the use of the embedded multi-channel aligner. The following procedure can be used to create a x8 PCI Express link:

1. Instantiate two flexiPCS quads in a design with both set to PCI Express Mode and targeted to x4 PCI Express operation.
2. Choose registers settings as defined for PCI Express Mode (for all blocks except the Multi-channel Aligner)
3. Set the registers settings for the Multi-channel Aligners for both quads as defined for x4 PCI Express operation. Both Multi-channel Aligners must have the same register settings.
4. Set up alignment between the two quads by setting the appropriate Quad Interface Registers and Inter-Quad Interface Registers. For more details on setting up multi-quad and multi-chip alignment, refer to the Multi-Channel Alignment section of the LatticeSC/M Family flexiPCS Data Sheet.
5. Incorporate Multi-quad Clock Tolerance Compensation correction logic in the FPGA fabric. This logic corrects for any times that the CTC modules in two quads that are being used together as part of a 8 channel link do not simultaneously insert and/or delete /SKP/ bytes at the same time. This may occur occasionally as the CTC modules in separate quads are not linked. The CTC correction logic detects when insertion and/or deletion has occurred in only one of the two quads and adjusts the data streams so that channel alignment is maintained across all 8 channels. This logic effectively delays the Clock Tolerance Compensation insertion and deletion operation for the brief time that the CTC operation in the two quads are not synchronized.

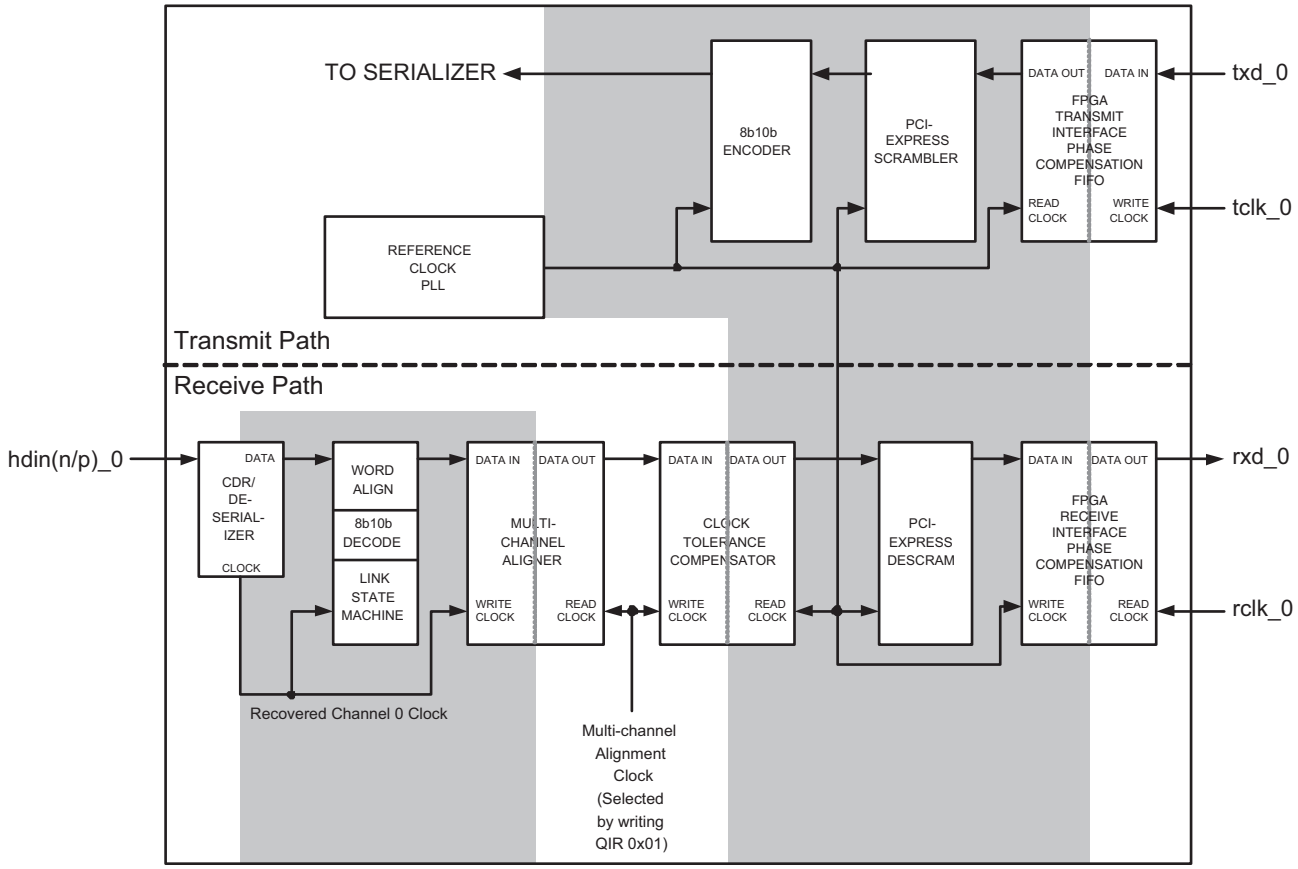
Figure 8-15 shows the clock domains for both transmit and receive directions for a single channel inside the flexiPCS.

On the transmit side, a clock domain transfer from the tclk input at the FPGA interface to the locked reference clock occurs at the FPGA Transmit Interface phase compensation FIFO. The FPGA Transmit Interface phase compensa-

tion FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. These phase compensation FIFOs (one per channel) cannot compensate for frequency variations.

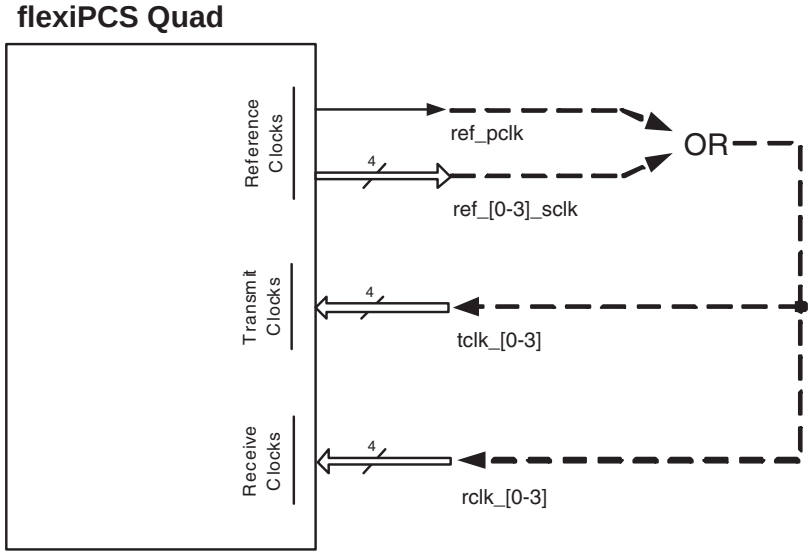
On the receive side, a clock domain transfer from the recovered channel clock to the Multi-channel Alignment channel clock occurs at the Multi-channel Aligner. A second clock domain transfer from the Multi-channel Alignment channel clock to the locked reference clock occurs at the Clock Tolerance Compensator. A third clock domain transfer occurs from the locked reference clock to the rclk input at the FPGA interface at the FPGA Receive Interface phase compensation FIFO. The FPGA Receive Interface phase compensation FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. These phase compensation FIFOs (one per channel) cannot compensate for frequency variations.

***Figure 8-15. flexiPCS Clock Domain Transfers for x4 PCI Express Operation***




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 flexiPCS quad set to PCI Express mode. Figure 8-16 illustrates the possible connections that would result in a synchronous interface.

*Figure 8-16. Synchronous input clocks to flexiPCS quad set to PCI Express mode*




## Control & Status

The Control & Status pins for x4 PCI Express operation can be optionally selected on a per quad basis when generating the flexiPCS quad interface files with the ispLEVER tools. Each of these control and status functions are also accessible through equivalent Quad Interface and Channel Interface Registers. The control and status signals allow direct real time access of these functions from FPGA logic. In addition to the Control and Status pins common to the PCI Express Mode, a flexiPCS quad targeted to x4 PCI Express operation has additional Control & Status pins related to control and monitoring of the multi-channel aligner.

**Control**

**felb** - Active high control signal which sets up all four channels in Far-End Loopback mode. This mode is generally used for testing the flexiPCS logic, looping back receive data back to the transmit direction without crossing the FPGA logic interface. For more information on the details of Far-End Loopback mode, see the **flexiPCS Testing** Section of the flexiPCS Data Sheet. The Far-End Loopback mode can also be initiated by writing Channel Interface Register Offset Address 0x00, bit 5 to '1'.

**lsm_en** - Active high Receive Link State Machine Enable for all four channels. A change on lsm_en resets the Receive Link State Machines into No Sync mode. The Receive Link Machine Enable can also be set by writing Channel Interface Register Offset Address 0x00, bit 7 to '1'.

**mca_align_en -** Active high multi-channel align enable. Multi-channel Alignment Enable can also be set by writing the appropriate Quad Interface Register Offset Address 0x04, bits [4:7] to '1' (bit 4 for channel 3, bit 5 for channel 2, bit 4 for channel 1, and bit 7 for channel 0).

**mca_resync** - Active high, asynchronous reset to multi-channel aligner state machine and aligner FIFO control logic. mca_resync resets logic in all four channels in four channel alignment mode.

**Status**

**lsm_status_[0-3]** - Per channel signal from the Receive Link State Machine indicating link synchronization successful. The link synchronization status can also be read from the appropriate Quad Interface Register Offset Address 0x84, bits [4:7] (bit 4 for channel 0, bit 5 for channel 1, bit 4 for channel 2, and bit 7 for channel 3).

**mca_aligned** - Active high signal indicating successful alignment of all four channels in four channel alignment mode. The multi-channel alignment status can also be read from the Quad Interface Register Offset Address 0x82, bit 2.

**ctc_urun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has underrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 6.

**ctc_orun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has overrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 7.

## Status Interrupt Registers

Some of the status registers associated with PCI Express operation have an associated status interrupt and status interrupt enable register. Any flexiPCS status interrupt register will generate an interrupt to the Systembus block (if used in the design). For a description of the operation of interrupts with the Systembus, refer to the Systembus section of the LatticeSC/M Family Data Sheet. Operation of the interrupt registers for the flexiPCS is as follows:

User must set the appropriate status interrupt enable bit to a '1'

Whenever the associated status bit goes high, its status interrupt bit will also go high. The status interrupt bit will remain high even if the associated status bit goes low.

The status interrupt bit will remain high until it is read, when it will automatically be cleared. If the associated status bit is still high, then the status interrupt bit will go high again (until read the next time).

Table 8-7 shows a listing of the status registers associated with PCI Express operation which have a corresponding status interrupt and status interrupt enable bit.

***Table 8-7. Status Interrupt Register Bits***

| Status Register Bit Function | Status Register Bit Address | Status Interrupt Enable Register Bit Address | Status Interrupt Register Bit Address |
|---|---|---|---|
| Receive Link State Machine Status (lsm_status) Channel 0 | QIR 0x84, bit 4 | QIR 0x1C, bit 4 | QIR 0x85, bit 4 |
| Receive Link State Machine Status (lsm_status) Channel 1 | QIR 0x84, bit 5 | QIR 0x1C, bit 5 | QIR 0x85, bit 5 |
| Receive Link State Machine Status (lsm_status) Channel 2 | QIR 0x84, bit 6 | QIR 0x1C, bit 6 | QIR 0x85, bit 6 |
| Receive Link State Machine Status (lsm_status) Channel 3 | QIR 0x84, bit 7 | QIR 0x1C, bit 7 | QIR 0x85, bit 7 |
| Multi-channel Alignment State Machine Status (mca_aligned_01 in x1 mode or mca_aligned in x4 mode)<br><br>Channels 0 & 1 (2-channel alignment mode)<br><br>Channels 0,1,2 & 3 (4-channel alignment mode) | QIR 0x82, bit 2 | QIR 0x0C, bit 2 | QIR 0x83, bit 2 |

*Table 8-7. Status Interrupt Register Bits (Continued)*

| | | | |
|---|---|---|---|
| Multi-channel Alignment State Machine Status (mca_aligned_23)<br><br>Channels 2 & 3 (2-channel alignment mode)<br><br>Inactive (4-channel alignment mode) | QIR 0x82, bit 3 | QIR 0x0C, bit 3 | QIR 0x83, bit 3 |
| Clock Tolerance Compensation FIFO underrun | CIR 0x85, bit 6 | CIR 0x0A, bit 6 | CIR 0x8A, bit 6 |
| Clock Tolerance Compensation FIFO overrun | CIR 0x85, bit 7 | CIR 0x0A, bit 7 | CIR 0x8A, bit 7 |

# Serial RapidIO Mode
# LatticeSC/M Family flexiPCS Data Sheet

January 2008 Data Sheet DS1005

## Introduction

This data sheet describes the operation of the Serial RapidIO mode of the LatticeSC flexiPCS. The LatticeSC can support 1x and 4x Serial RapidIO applications with one flexiPCS quad. Serial RapidIO applications of widths 8x, 16x, and 32x can be realized on one device through the use of multiple flexiPCS quads (limited by the size of the LatticeSC device chosen).

The LatticeSC flexiPCS in Serial RapidIO Mode supports the following operations:

**Transmit Path (From LatticeSC device to line):**

- Transmit State Machine with randomized A, K, R Idle generation according to the RapidIO Physical Layer 1x/4x LP-Serial specification.
- 8b10b Encoding

**Receive Path (From line to LatticeSC device):**

- Word Alignment based on the Sync Code Group as defined in the RapidIO Physical Layer 1x/4x LP-Serial specification.
- 8b10b Decoding
- Link State Machine functions incorporating operations defined in the Lane_Synchronization state diagram of the RapidIO Physical Layer 1x/4x LP-Serial specification.
- Clock Tolerance Compensation logic capable of accommodating clock domain differences.
- Receive State Machine compliant to the RapidIO Physical Layer 1x/4x LP-Serial specification.

### LatticeSC flexiPCS Quad Module

Devices in the LatticeSC family have up to 8 quads of embedded flexiPCS logic. Each quad in turn supports 4 independent full-duplex data channels. A single channel can support a fully compliant Serial RapidIO data link and each quad can support up to four such channels. Note that mode selection is done on a per quad basis. Therefore, a selection of Serial RapidIO mode for a quad dedicates all four channels in that quad to Serial RapidIO mode.

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

Operation of the SERDES requires the user to provide a reference clock (or clocks) to each active quad to provide a synchronization reference for the SERDES PLLs. Reference clocks are shared among all four channels of a quad. If the transmit and receive frequencies are within the specified ppm tolerance for the LatticeSC SERDES (See DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet), only one reference clock for both transmit and receive is needed for both transmit and receive directions. If the receive frequency is not within the specified ppm tolerance for the LatticeSC SERDES (See the DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet) of the transmit frequency, then separate reference clocks for the transmit and receive directions must be provided.

A broadcast write to multiple quads cannot be used to power on SERDES in any device-package combination that does not have all SERDES quads bonded because of excess power on the board and jitter is also affected.

Simultaneous support of different (non-synchronous) high-speed data rates is possible with the use of multiple quads. Multiple frequencies that are exact integer multiples can be supported on a per channel basis through use of the full-rate/half-rate mode options (see the **SERDES Functionality section** for more details).

© 2008 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

## Quad and Channel Option Control

Although the mode selection and reference frequency covers an entire quad, many options covering clocking and data formatting are available on a per channel basis. These options are detailed in the following Serial RapidIO Mode description. Some of these options can be controlled directly through the FPGA interface pins made available on the Serial RapidIO Quad Module.

Other options are available only through dedicated registers that can be written and read via the embedded System Bus Interface (see the System Bus section for more details on the operation of the System Bus), or during device configuration. Depending on the specific option, a register may affect operation of multiple quads, a single quad or an individual channel in a quad. Registers dedicated to multiple quads are listed in the Inter-quad Interface Register Map in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet. Registers dedicated to one quad are listed in the Quad Interface Register Map. Registers dedicated to a single channel are listed in the Channel Interface Register Map.

### Register Map Addressing

Figure 9-1 provides a map of base register addresses for any of the flexiPCS quads on a LatticeSC device. Note that different devices may have different numbers of available quads up to a total of 8 quads (or 32 channels) maximum.

Inter-quad Interface, Quad Interface, and Channel Interface Registers are listed by Offset Address. Each Inter-quad Interface Register, Quad Interface Register, and Channel Interface Register has a unique 18-bit address determined by the specific quad or channel targeted. The addressing of Inter-quad Interface Registers, Quad Interface Registers, and Channel Interface Registers is shown Figure 9-1.

Base addresses are dependant on the physical location of a given quad or channel on a LatticeSC device. Each quad and channel has an associated set of control and status registers. These registers are referred throughout this data sheet by their offset addresses. The unique 18-bit address of a control or status register for a specific quad or channel is simply the offset address of that register added to the base address for that specific quad or channel as shown in Figure 9-1.

Channel Interface Registers can be written in one of three methods. One method is to write to one specific register corresponding to one specific channel. The other two methods involve writing to multiple channel registers with just one write operation. This is referred to as a Broadcast write. A Broadcast write is useful when multiple channel registers are to be written with the same value. The first method of Broadcast writing involves writing to all four channel registers in a quad at one time. A second method of Broadcast writing involves writing to all channel registers for all quads on the left or right side of the device at one time. Therefore, a specific Channel Interface Register may be accessed through any one of three channel addresses, depending on the number of channel registers being written to at any one time.

Throughout this document, the byte-wide data at each offset address is listed with a hexadecimal value with bit 0 as the MSb and bit 7 as the LSb as per the PowerPC convention. For example if the value for Quad Interface Register Offset Address 0x02 is stated to be 0x15, the bit pattern defined is as shown in Table 9-1:

***Table 9-1. Control/Status Register Bit Order Example***

| **Quad Interface Register Offset Address 0x02 = 0x15** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Data Bit 0 | Data Bit1 | Data Bit 2 | Data Bit 3 | Data Bit 4 | Data Bit 5 | Data Bit 6 | Data Bit 7 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| **1** | | | | **5** | | | |

A full list of register Offset Addresses is provided in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet.

***Figure 9-1. flexiPCS Inter-quad, Quad, and Channel Interface Register Base Map Addressing***

## Auto-Configuration

Initial register setup for each flexiPCS mode can be performed without accessing the system bus by using the auto-configuration tool, IPexpress. IPexpress generates 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 system bus is therefore not needed if all quads are to be set via auto-configuration files. However, the system bus must be included in a design if the user needs to change control registers or monitor status registers during operation. Note that a quad reset (Quad Interface Register 0x43, bit 7 or the **quad_rst** port at the FPGA interface) will reset all registers back to their default (not auto-configuration) values. If a quad reset is issued, the flexiPCS registers need to be rewritten via the Systembus.

## Serial RapidIO Register Settings

The auto-configuration file sets registers that perform the following operations. If the auto-configuration file is not used, the appropriate registers need to be programmed via the Systembus. For more options, refer to the flexiPCS register map in the **Memory Map** section of the **LatticeSC/M Family flexiPCS Data Sheet**.

**A flexiPCS quad can be set to Serial RapidIO mode by writing Quad Interface Register Offset Address 0x18 bits[0:7] to 0x02.**

This register value automatically sets up the following options in the flexiPCS for the given quad. Details on the functionality of each chosen option is provided in the **Serial RapidIO Mode Detailed Description** section.

**Transmit Path**

- The four internal flexiPCS Tx clocks (one per channel) are synchronized to align the four data lanes to the same clock edge
- Serial RapidIO Transmit State Machine is enabled
- 8b10b encoder is enabled

**Receive Path**

- Word aligner is enabled
- 8b10b decoder is enabled
- Receive Link State Machine is set to Serial RapidIO Mode
- Receive State Machine is set to Serial RapidIO Mode

Other register settings are required to operate a channel or channels in Serial RapidIO Mode. The following is a list of options that must be set for Serial RapidIO operation. More detail for each option is included in the **Serial RapidIO Mode Detailed Description** section.

- Powerup of appropriate quad and channel. Quad powerup is chosen by writing the appropriate Quad Interface Register Offset Address 0x28, bit 1 to ‘1’. Channel powerup is chosen by writing the appropriate Channel Interface Register Offset Address 0x13, bit 6 (receive direction) and/or bit 7 (transmit direction) to ‘1’. By default, all channels and quads are powered down.
- Receive Link State Machine enable. By default, all channel Receive Link State Machines are disabled. The Receive Link State Machine for a given channel can be enabled by writing the appropriate Channel Interface Register Offset Address 0x00, bit 7 to ‘1’.
- Setting SERDES reset low at end of flexiPCS setup. By default, the SERDES reset is set to ‘1’ (SERDES are reset). The SERDES reset can be set low by writing Quad Interface Register Offset Address 041x, bit 5 to a ‘0’. For more information on proper flexiPCS powerup and reset sequencing, refer to the **flexiPCS Reset Sequences** and **flexiPCS Power Down Control Signals** descriptions in the LatticeSC/M Family flexiPCS Data Sheet **SERDES Functionality** section.
- Word alignment character(s), such as a comma
- Clock Tolerance Compensation insertion/deletion ordered set values and FIFO settings
- Multi-channel settings including user-defined alignment characters and FIFO settings for 4x Serial RapidIO operation.

## Serial RapidIO Auto-Configuration Example

An example of an auto-configuration file for a quad set to Serial RapidIO Mode for a 4x Serial RapidIO application with Multi-channel Alignment and Clock Tolerance Compensation (CTC) enabled and with all four channels enabled is shown in Figure 9-2. Format for the auto-configuration file is register type (quad=quad register, ch1=channel 1 register), offset address in hex, register value in hex, # comment

***Figure 9-2. Serial RapidIO Auto-Configuration Example File***

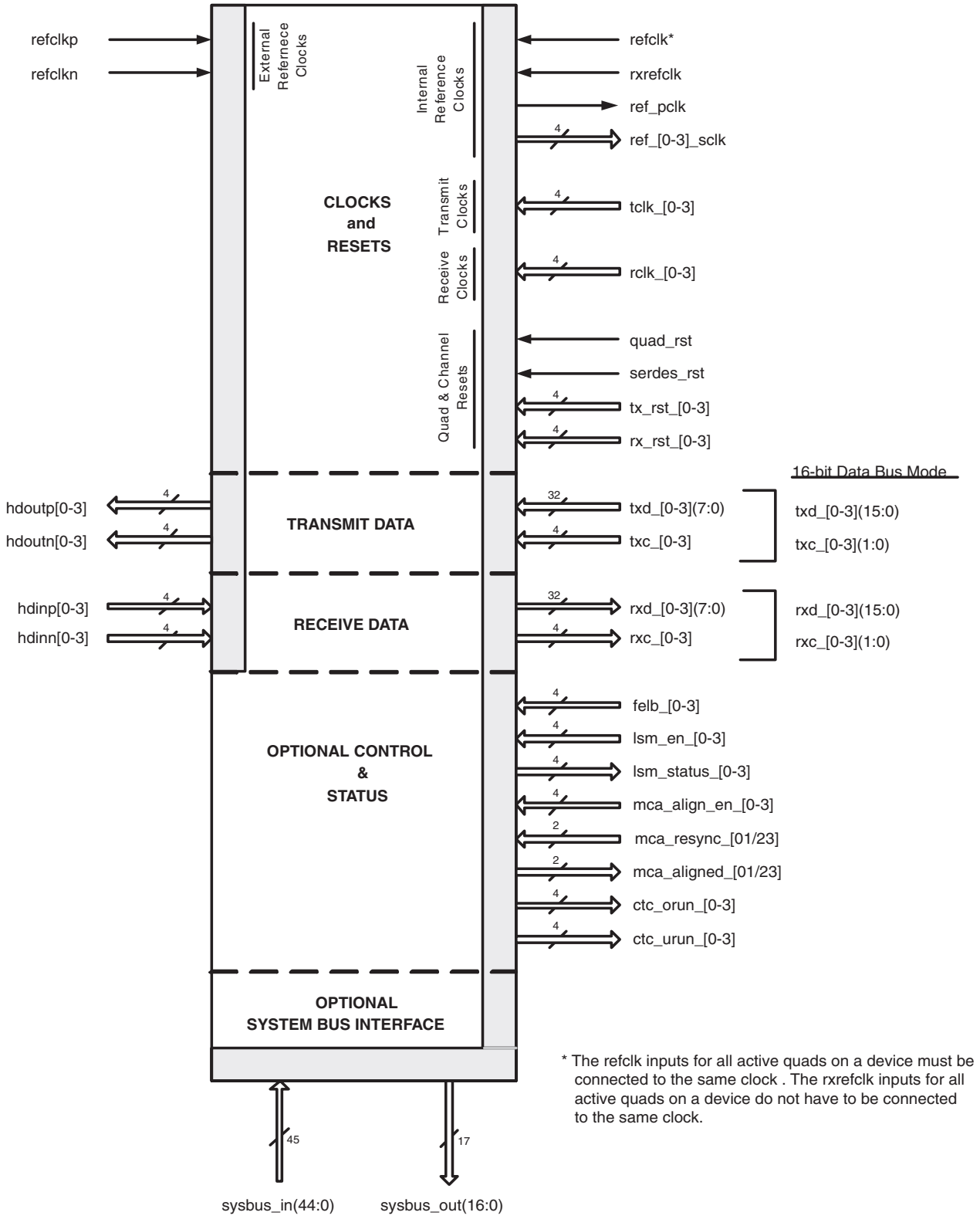
```
quad 18 02     #  Set quad to Serial RapidIO Mode
quad 30 04     #  Set Tx Synchronization bit
quad 29 00     #  Set reference clock select to refclk(p/n) inputs
quad 28 40     #  Set bit clock multiplier to 20X (for full rate mode), quad powerup set to '1'
quad 02 00     #  Set ref_pclk to channel 0 clock
quad 14 7F     #  Set word alignment mask to compare lowest 7 LSBs
quad 15 03     #  Set positive disparity comma value
quad 16 7C     #  Set negative disparity comma value
quad 19 40     #  Select 4 channel alignment (4x Serial RapidIO), FPGA interface data bus width 8-bits
quad 04 0F     #  Set alignment enable for all four channels
quad 01 00     #  Set all multi-channel alignment clocks to be sourced from
#  the channel 0 recovered receive clock
quad 07 FF     #  Set multi-channel align character mask to compare bits [7:0]
quad 08 FB     #  Set multi-channel align character “A”, bits [7:0] to FB (K27.7)
quad 09 FB     #  Set multi-channel align character “B”, bits [7:0] to FB (K27.7)
quad 0A 15     #  Set multi-channel align character mask to compare bit 8
               #  Set multi-channel align characters “A” and “B” bit 8 (K character) to '1'
quad 05 00     #  Set multi-channel alignment FIFO latency to minimum
quad 06 03     #  Set multi-channel alignment FIFO high water mark to 3
quad 0E 00     #  Set insertion/deletion to 1 byte for CTC, set minimum inter-packet gap
quad 0D 97     #  Set CTC FIFO watermarks (high=9, low=7)
quad 12 FD     #  1st byte of ||R|| (skip) pattern for CTC (K29.7)
quad 13 01     #  K bit for ||R|| pattern
ch0 00 01      #  Turn on channel 0 link state machine
ch0 13 03      #  Powerup channel 0 transmit and receive directions, set rate mode to “full”
ch0 14 90      # 16% pre-emphasis on SERDES output buffer
ch0 15 10      # +6 dB equalization on SERDES input buffer
ch1 00 01      #  Turn on channel 1 link state machine
ch1 13 03      #  Powerup channel 1 transmit and receive directions, set rate mode to “full”
ch1 14 90      # 16% pre-emphasis and on SERDES output buffer
ch1 15 10      # +6 dB equalization on SERDES input buffer
ch2 00 01      #  Turn on channel 2 link state machine
ch2 13 03      #  Powerup channel 2 transmit and receive directions, set rate mode to “full”
ch2 14 90      # 16% pre-emphasis and on SERDES output buffer
ch2 15 10      # +6 dB equalization on SERDES input buffer
```

```
ch3 00 01      #  Turn on channel 3 link state machine
ch3 13 03      #  Powerup channel 3 transmit and receive directions, set rate mode to “full”
ch3 14 90      # 16% pre-emphasis and on SERDES output buffer
ch3 15 10      # +6 dB equalization on SERDES input buffer
quad 41 00     # de-assert serdes_rst
quad 40 ff     # assert datapath reset for all channels
quad 41 03     # assert MCA reset
quad 41 00     # de-assert MCA reset
quad 40 00     # de-assert datapath reset for all channels
```

## Serial RapidIO Mode

IPexpress allows the designer to choose the mode for each flexiPCS quad used in a given design. Figure 9-3 displays the resulting port interface to one flexiPCS quad that has been set to Serial RapidIO mode with either no alignment options selected or two channel alignment selected in IPexpress.

***Figure 9-3. Serial RapidIO Mode Pin Diagram (Four channel alignment not selected)***




* The refclk inputs for all active quads on a device must be connected to the same clock . The rxrefclk inputs for all active quads on a device do not have to be connected to the same clock.

## 1x Serial RapidIO Mode Pin Descriptions

Table 9-2 lists all inputs and outputs to/from a flexiPCS quad in Serial RapidIO mode (four channel alignment not selected). A brief description for each port is given in the table. A more detailed description of the function of each port is given in the **Serial RapidIO Mode Detailed Description**.

***Table 9-2. Serial RapidIO Mode Pin Description***

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Reference Clocks** | | | |
| refclkp | In from I/O pad | N/A | Reference clock input, positive. Dedicated CML input. |
| refclkn | In from I/O pad | N/A | Reference clock input, negative. Dedicated CML input |
| refclk | In from FPGA | N/A | Optional reference clock input from FPGA logic. Can be used instead of per quad reference clock.<br><br>The refclk inputs for all active quads on a device must be connected to the same clock. |
| rxrefclk | In from FPGA | N/A | Optional receive reference clock input from FPGA logic. Can be used instead of per quad reference clock. The rxrefclk inputs for all active quads do not have to be connected to the same clock. |
| ref_pclk | Out to FPGA | N/A | Locked reference clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| ref_[0-3]_sclk | Out to FPGA | N/A | Per channel locked reference clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| **Transmit Clocks** | | | |
| tclk_[0-3] | In from FPGA | N/A | Per channel transmit clock inputs from FPGA. Used to clock the TX data phase compensation FIFO with clock synchronous to the reference clock. May also be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Receive Clocks** | | | |
| rclk_[0-3] | In from FPGA | N/A | Per channel receive clock inputs from FPGA. May be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference and/or receive reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Resets** | | | |
| quad_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of SERDES and PCS logic. |
| serdes_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of the SERDES. |
| tx_rst_[0-3] | In from FPGA | ASYNC | Per channel active high, asynchronous reset of individual transmit channel of PCS logic. |
| rx_rst_[0-3] | In from FPGA | ASYNC | Per channel active high, asynchronous reset of individual receive channel of PCS logic. |
| **Transmit Data** | | | |
| hdoutp[0-3] | Out to I/O pad | N/A | High-speed CML serial output, positive. |
| hdoutn[0-3] | Out to I/O pad | N/A | High-speed CML serial output, negative. |
| txd_[0-3](7:0)<br><br>txd_[0-3](15:0)* | In from FPGA | tclk_[0-3] | Per channel parallel transmit data bus from the FPGA. Bus is 8-bits wide if QIR 0x19, bit 4 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 4 = '1'. |
| txc_[0-3]<br><br>txc_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel transmit control character indication.<br><br>*Bus is 2 bits wide if QIR 0x19, bit 4 = '1'. |

*Table 9-2. Serial RapidIO Mode Pin Description*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Receive Data** | | | |
| hdinp[0-3] | In from I/O pad | N/A | High-speed CML serial input, positive. |
| hdinn[0-3] | In from I/O pad | N/A | High-speed CML serial input, negative. |
| rxd_[0-3](7:0)<br><br>rxd_[0-3](15:0)* | Out to FPGA | rclk_[0-3] | Per channel parallel receive data bus to the FPGA. Bus is 8-bits wide if QIR 0x19, bit 5 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 5 = '1'. |
| rxc_[0-3]<br><br>rxc_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel receive control character indication.<br><br>*Bus is 2 bits wide if QIR 0x19, bit 5 = '1'. |
| **Optional Control & Status** | | | |
| felb_[0-3] | In from FPGA | ASYNC | Per channel active high far-end loopback enables. |
| lsm_en_[0-3] | In from FPGA | ASYNC | Per channel receive link state machine enable. |
| lsm_status_[0-3] | Out to FPGA | ASYNC | Per channel signal from receive link state machine indicating successful link synchronization. |
| mca_align_en_[0-3] | In from FPGA | ASYNC | Per channel active high multi-channel aligner enables. |
| mca_resync_[01/23] | In from FPGA | ASYNC | Active high async multi-channel aligner resynchronization signal |
| mca_aligned_[01/23] | Out to FPGA | ASYNC | Active high indicating successful alignment of channels. |
| ctc_urun_[0-3] | Out to FPGA | ASYNC | Per channel active high flag indicator that clock tolerance compensation FIFO has underrun. |
| ctc_orun_[0-3] | Out to FPGA | ASYNC | Per channel active high flag indicator that clock tolerance compensation FIFO has overrun. |
| felb_[0-3] | In from FPGA | ASYNC | Per channel active high far-end loopback enables. |
| lsm_en_[0-3] | In from FPGA | ASYNC | Per channel receive link state machine enable. |
| **Optional System Bus Interface** | | | |
| sysbus_in(44:0) | In from FPGA | N/A | Control and data signals from the internal system bus. |
| sysbus_out(16:0) | Out to FPGA | N/A | Control and data signals to the internal system bus. |

# Serial RapidIO Mode Detailed Description

The following section provides a detailed description of the operation of a flexiPCS quad set to Serial RapidIO mode. The functional description is organized according to the port listing shown in Figure 9-3. The System Bus base address for any control and status registers relevant to a given function are provided with the description of the operation of that function.

## Clocks & Resets

A detailed description of all SERDES clock, data, and reset ports and recommended reset sequencing is provided in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet. The following is a recommended setup of the SERDES for Serial RapidIO applications.

For 1x Serial RapidIO applications, the bit clock rate is typically either 1.25 Gbps, 2.5 Gbps, or 3.125 Gbps. The 1.25 Gbps bit clock rate falls into the range defined as "half rate" mode for the SERDES PLLs. Full rate mode is selected separately for each channel and each direction by writing the appropriate Channel Interface Register Base Address 0x13, bit 5 (transmit) and bit 4 (receive) to a '1' (default).

The 2.5 Gbps and 3.125 Gbps bit clock rates fall into the range defined as "full rate" mode for the SERDES PLLs. Full rate mode is selected separately for each channel and each direction by writing the appropriate Channel Interface Register Base Address 0x13, bit 5 (transmit) and bit 4 (receive) to a '0' (default).

The reference clock frequency is determined by the reference clock mode chosen. Table 9-3, Table 9-4, and Table 9-5 show the possible reference clock frequencies for various reference clock modes which result in a 1.25 Gbps, 2.5 Gbps, and 3.125 Gbps internal bit rate respectively.

***Table 9-3. Serial RapidIO Reference Clock Frequency Options for 1.25 Gbps Bit Clock Rate***

| Half Rate Mode<br><br>Channel Interface Register Base Address 0x13<br><br>Transmit - Bit 5 = '1'<br>Receive - Bit 4 = '1' | | | | |
|---|---|---|---|---|
| Reference Clock Mode<br><br>Quad Interface Register Base Address = 0x28, Bits [2:3] | Reference Clock Frequency | Internal Serial (bit) Clock Frequency | FPGA Interface Clock Frequency<br><br>Quad Interface Register Base Address 0x19 | |
| | | | 8 Bit Data Bus Width<br><br>Transmit - Bit 4 = '0'<br>Receive - Bit 5 = '0' | 16 Bit Data Bus Width<br><br>Transmit - Bit 4 = '1'<br>Receive - Bit 5 = '1' |
| 00 | 125 MHz | 1.25 GHz | 125 MHz | 62.5 MHz |
| 01 | 250 MHz | 1.25 GHz | 125 MHz | 62.5 MHz |
| 10 | 500 MHz | 1.25 GHz | 125 MHz | 62.5 MHz |

***Table 9-4. Serial RapidIO Reference Clock Frequency Options for 2.5 Gbps Bit Clock Rate***

| Full Rate Mode<br><br>Channel Interface Register Base Address 0x13<br><br>Transmit - Bit 5 = '0'<br>Receive - Bit 4 = '0' | | | | |
|---|---|---|---|---|
| Reference Clock Mode<br><br>Quad Interface Register Base Address = 0x28, Bits [2:3] | Reference Clock Frequency | Internal Serial (bit) Clock Frequency | FPGA Interface Clock Frequency<br><br>Quad Interface Register Base Address 0x19 | |
| | | | 8 Bit Data Bus Width<br><br>Transmit - Bit 4 = '0'<br>Receive - Bit 5 = '0' | 16 Bit Data Bus Width<br><br>Transmit - Bit 4 = '1'<br>Receive - Bit 5 = '1' |
| 00 | 125 MHz | 2.5 GHz | 250 MHz | 125 MHz |
| 01 | 250 MHz | 2.5 GHz | 250 MHz | 125 MHz |
| 10 | 500 MHz | 2.5 GHz | 250 MHz | 125 MHz |

*Table 9-5. Serial RapidIO Reference Clock Frequency Options for 3.125 Gbps Bit Clock Rate*

| Full Rate Mode<br><br>Channel Interface Register Base Address 0x13<br><br>Transmit - Bit 5 = '0'<br>Receive - Bit 4 = '0' | | | | |
|---|---|---|---|---|
| Reference Clock Mode<br><br>Quad Interface Register Base Address = 0x28, Bits [2:3] | Reference Clock Frequency | Internal Serial (bit) Clock Frequency | FPGA Interface Clock Frequency<br><br>Quad Interface Register Base Address 0x19 | |
| | | | 8 Bit Data Bus Width<br><br>Transmit - Bit 4 = '0'<br>Receive - Bit 5 = '0' | 16 Bit Data Bus Width<br><br>Transmit - Bit 4 = '1'<br>Receive - Bit 5 = '1' |
| 00 | 156.25 MHz | 3.125 GHz | 312.5 MHz | 156.25 MHz |
| 01 | 312.5 MHz | 3.125 GHz | 312.5 MHz | 156.25 MHz |
| 10 | 625 MHz | 3.125 GHz | 312.5 MHz | 156.25 MHz |

*Note: There are also frequency dependent SERDES performance control bits that are not writable via the Systembus. These control bits are set in the auto-configuration file generated within IPexpress and passed into the bitstream through the normal FPGA design flow (map, par, bitgen). To change the bit rate for a SERDES, specify the proper values for the affected flexiPCS quad in IPexpress and regenerate the auto-configuration file. Failure to do this may result in non-optimal SERDES operation.*

Additional information on all reference clock rate modes can be found in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

**Quad & Channel Resets**

Resets are provided to reset an entire quad (either SERDES logic only or all flexiPCS logic) or each individual channel (all flexiPCS logic per channel). These resets are driven from the FPGA logic. A summary of the control signals provided for reset are listed below. More detail on recommended initialization and reset sequences for the SERDES is provided in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

The following inputs to the flexiPCS from the FPGA are enabled as long as Quad Interface Register Base Address 0x42, bit 7 is set to '1' (which is the default on powerup). Writing a '0' to this bit will disable these reset inputs.

**quad_rst** - Active high, asynchronous signal from FPGA resets all SERDES and PCS logic in the quad. This reset can also be performed by writing Quad Interface Register Base Address 0x43, bit 7 to '1'. quad_rst also resets all flexiPCS control registers. If these registers had been previously written via the Systembus or set during configuration through an auto-configuration file, they will need to be rewritten following this reset.

**serdes_rst** - Active high, asynchronous signal from FPGA resets all SERDES (but not PCS) logic in the quad. This reset can also be performed by writing Quad Interface Register Base Address 041x, bit 5 to '1'.Note that this bit is preset to '1' on powerup and must be written to '0' before operation of the SERDES can commence.

**tx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one transmit channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Base Address 0x40, bits [4:7]. Bit 7 resets transmit channel 0, bit 6 resets transmit channel 1, bit 5 resets transmit channel 2, and bit 4 resets transmit channel 3.

**rx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one receive channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Base Address 0x40, bits [0:3]. Bit 3 resets receive channel 0, bit 2 resets receive channel 1, bit 1 resets receive channel 2, and bit 0 resets receive channel 3.

## Transmit Data

When configured into Serial RapidIO mode, a flexiPCS quad provides up to four transmit data paths. Each transmit data path converts a 8-bit wide Serial RapidIO compliant data stream at the FPGA logic interface to a high-speed serial line interface at the device outputs.

The following signals are the transmit path inputs from the FPGA logic to a single flexiPCS quad set to 1x Serial RapidIO Mode with all 4 channels enabled:

**txd_[0-3](7:0)** - Four separate 8-bit wide transmit data bus inputs from the FPGA interface by default. Each channel's txd is sampled on the rising edges of the corresponding tclk.

In 16 Bit Data Bus Mode, this bus is 16-bits wide (**rxd_[0-3](15:0)**).

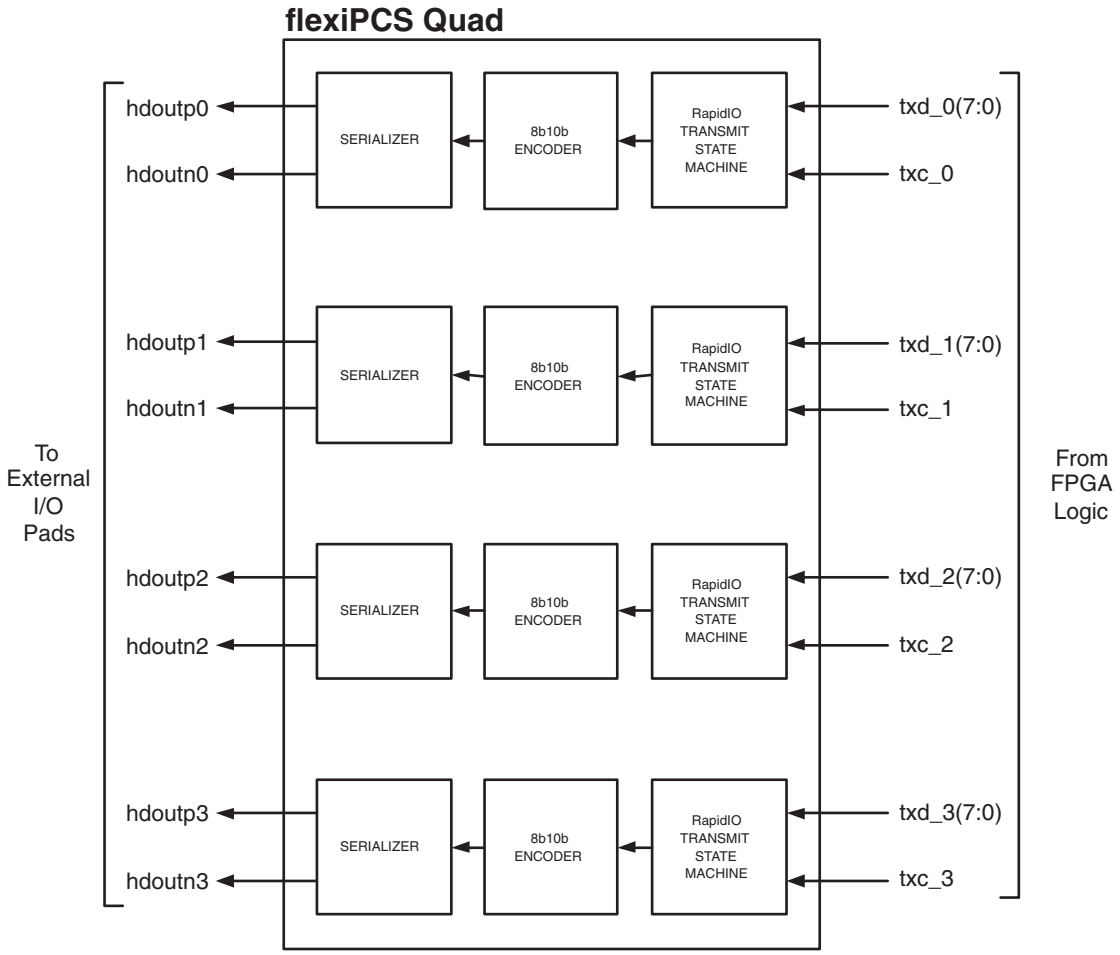
**txc_[0-3]** - Per channel, active high control character indicator.

In 16 Bit Data Bus Mode, txc is 2 bits wide (**txc_[0-3](1:0)**) with txc_[0-3](1) associated with txd_[0-3](15:8) and txc_[0-3](0) associated with txd_[0-3](7:0).

Note that the Tx Synchronization bit (Quad Interface Register Offset Address 0x30, bit 5) needs to be set to ensure that all four channels are aligned going into the transmit state machine. Since each Tx channel is clocked by its own transmit clock in the flexiPCS, setting the Tx Synchronization bit ensures these clocks are synchronized to the same clock edges.

The flexiPCS quad in Serial RapidIO mode transmit data path consists of the following sub-blocks per channel: Serial RapidIO Transmit State Machine, 8b10b Encoder, and Serializer. Figure 9-4 shows the four channels of transmit data paths in a flexiPCS quad.

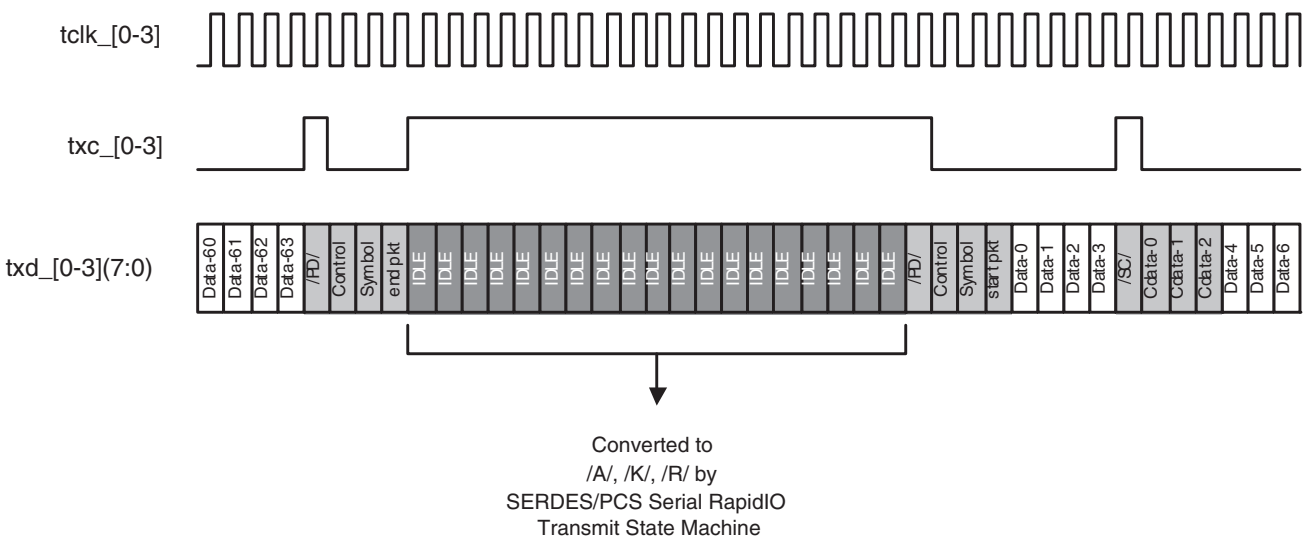
*Figure 9-4. Serial RapidIO Transmit Path (1 Quad)*




**Transmit State Machine**

The Serial RapidIO Transmit State Machine performs the conversion of Serial RapidIO idle control character, /I/ (txd = 0x07, txc=1), to a randomized sequence of /A/, /K/ or /R/ code-groups to enable lane synchronization, clock compensation and lane-to-lane alignment as described in the Physical Layer 1x/4x LP_Serial RapidIO specification. This includes a PRBS counter ($X^7 + X^6 + 1$) as described in the Pseudo-Random Idle Code-Group Generator diagram in the Serial RapidIO specification to provide the random code selection. Figure 9-5 shows an example of 1x Serial RapidIO data flow with idles:

***Figure 9-5. 1x Serial RapidIO Compliant Data Stream Example with Idles***




A list of 1x Serial RapidIO mode special characters is shown in Table 9-6 below:

***Table 9-6. 1x Serial RapidIO Transmit path ordered-sets and special code-groups***

| Code-Group Column Designation | Code-Group Column Use | Encoding |
|---|---|---|
| /PD/ | Packet_Delimiter Control Symbol | /K28.3/ |
| /SC/ | Start_of_Control Symbol | /K28.0/ |
| /I/ | Idle | |
| /K/ | 1x Sync | /K28.5/ |
| /R/ | 1x Skip | /K29.7/ |
| /A/ | 1x Align | /K27.7/ |

The flexiPCS Serial RapidIO Transmit State Machine converts Serial RapidIO compliant Idles at the txd inputs to a randomized stream of /A/, /K/, /R/ code groups according to the Physical Layer 1x LP-Serial RapidIO specification:

1. The first code-group of an idle sequence generated by a port operating in 1x mode shall be a /K/. The first code-group shall be transmitted immediately following the last code-group of a packet or delimited control symbol.

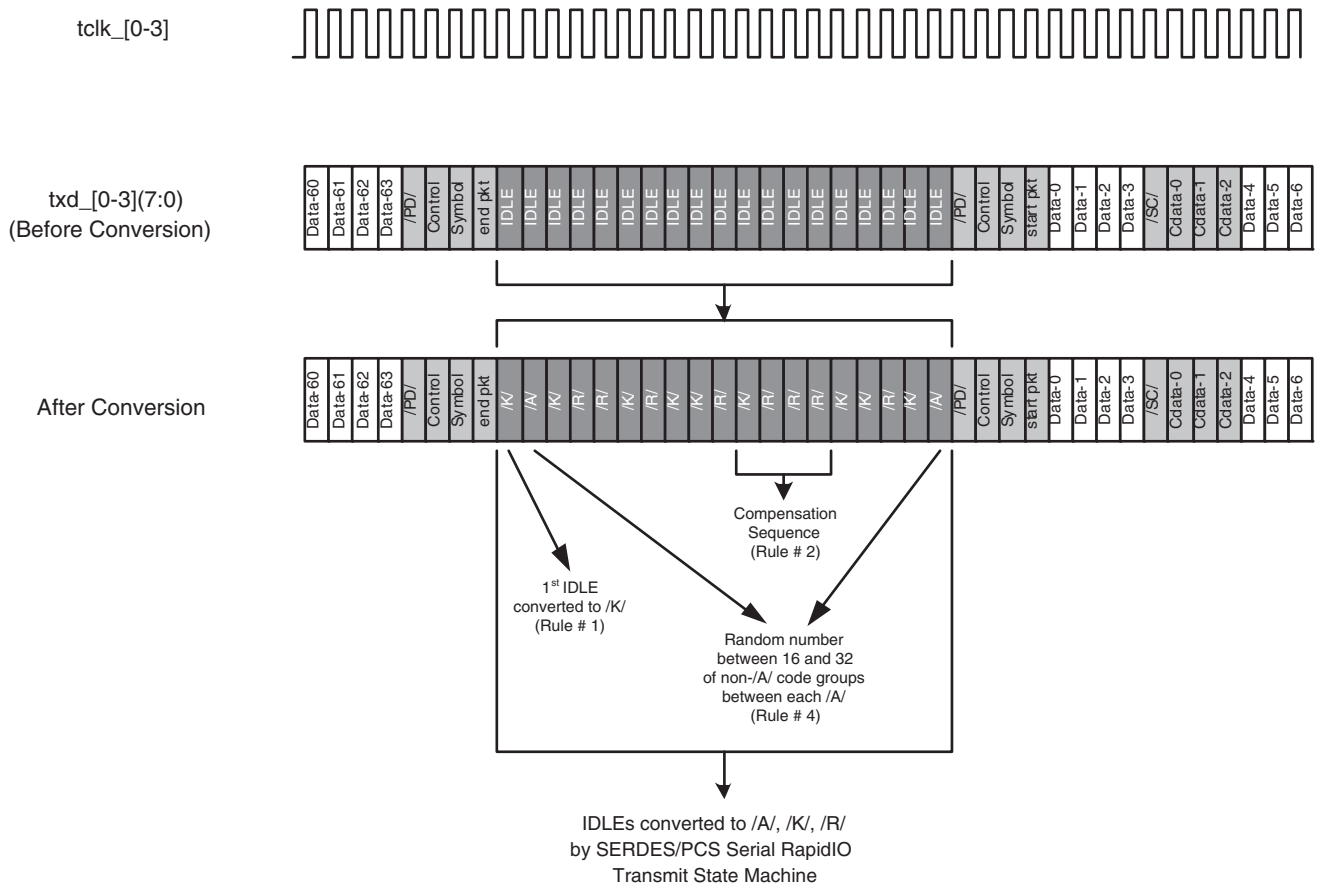
2. At least once every 5000 code-groups transmitted by a port operating in 1x mode, an idle sequence containing the /K/R/R/R/ code-group sequence shall be transmitted by the port. This sequence is referred to as the “compensation sequence”.

3. When not transmitting the compensation sequence, all code-groups following the first code-group of an idle sequence generated by a port operating in 1x mode shall be a pseudo-randomly selected sequence of /A/, /K/, and /R/ based on a pseudo-random sequence generator ($X^7 + X^6 + 1$ in the LatticeSC flexiPCS) and subject to the minimum and maximum /A/ spacing requirements.

4. The number of non /A/ code-groups between /A/ code-groups in the idle sequence of a port operating in 1x mode shall be no less than 16 and no more than 32. The number shall be pseudo-randomly selected, uniformly distributed across the range and based on a pseudo-random sequence ($X^7 + X^6 + 1$ in the LatticeSC flexiPCS).

Figure 9-6 shows an example of 1x Serial RapidIO data flow with idle code groups converted by the flexiPCS Serial RapidIO Transmit State Machine:

***Figure 9-6. 1x Serial RapidIO Idle Code Group Conversion Example***




**8b10b Encoding**

This module implements an 8b10b encoder as described in Section 4.4 of the RapidIO Physical Layer 1x/4x LP_Serial Specification. The encoder performs the 8-bit to 10-bit code conversion as described the specification, along with maintaining the running disparity rules as specified.

When a code violation is detected, the rxd data is set to 0xEE. When a disparity error is detected in a given data channel, the data is passed to that channel's rxd pins and rx_disp_err_detect goes to '1'.

**Serializer**

The 8b10b encoded data undergoes parallel to serial conversion and is transmitted off chip via the embedded SERDES. For detailed information on the operation of the LatticeSC PCS SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

## Receive Data

When configured into Serial RapidIO mode, a flexiPCS quad provides up to four 1x receive data paths from a high-speed serial line interface at the device inputs to 8-bit parallel interface at the FPGA logic interface.

The following signals are the receive path outputs to the FPGA logic from a single flexiPCS quad set to Serial RapidIO Mode with all 4 channels enabled:

**rxd_[0-3](7:0)** - Four separate 8-bit wide receive data bus outputs to the FPGA interface by default. Each channel's rxd transitions synchronously with respect to that channel's corresponding rclk.

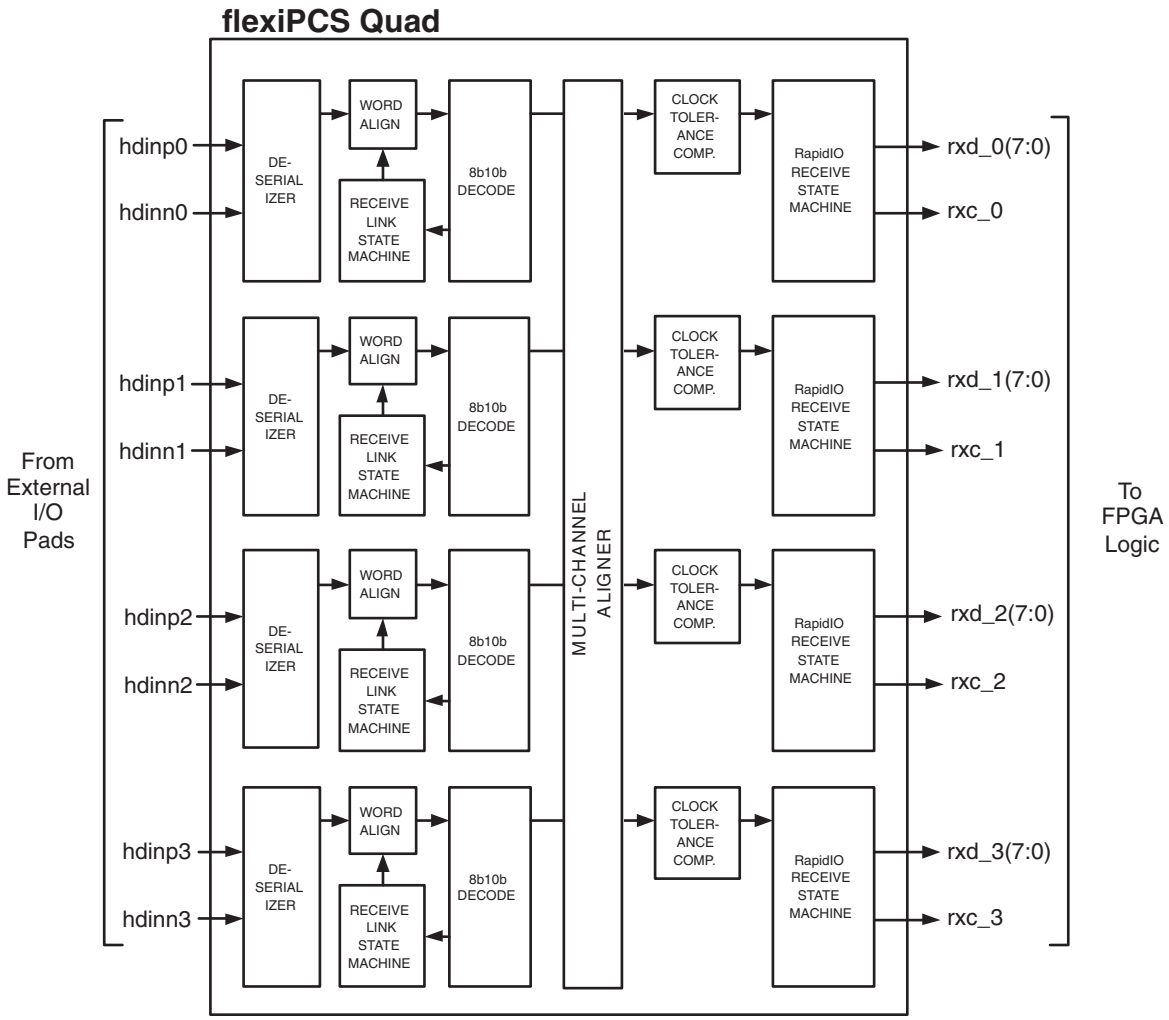
In 16 Bit Data Bus Mode, this bus is 16-bits wide (**rxd_[0-3](15:0)**).

**rxc_[0-3]** - Per channel receive control character indication. rxc is high when a control character is present on the corresponding channel's rxd inputs. rxc is low when a data byte is present on the corresponding channel's rxd inputs. Each channel's rxc transitions synchronously with respect to that channel's corresponding rclk.

In 16 Bit Data Bus Mode, rxc is 2 bits wide (**rxc_[0-3](1:0)**) with rxc_[0-3](1) associated with rxd_[0-3](15:8) and rxc_[0-3](0) associated with rxd_[0-3](7:0).

The flexiPCS quad in Serial RapidIO mode receive data path consists of the following sub-blocks per channel: Deserializer, Word Align, 8b10b Decode, & Link State Machine, Multi-channel Aligner, Clock Tolerance Compensator, and Serial RapidIO Receive State Machine. Figure 9-7 shows the four channels of receive data in a flexiPCS quad set to Serial RapidIO mode.

***Figure 9-7. Serial RapidIO Receive Data Path (1 Quad)***




**Deserializer**

Data is brought on chip to the embedded SERDES where it undergoes serial to parallel conversion. For detailed information on the operation of the LatticeSC PCS SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

**Word Alignment**
The word aligner module performs the word alignment code word detection and alignment operation. The word alignment character is used by the receive logic to perform 10-bit word alignment upon the incoming data stream. The word alignment algorithm is described in Clause 36.2.4.9 in the 802.3-2002 1000BASE-X specification.

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

- 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 word alignment compare. Word alignment characters and the mask register are set on a per quad basis. For Serial RapidIO, the word alignment characters should be set to "XXX0000011" (jhgfiedcba bits for positive running disparity comma character matching code groups K28.1, K28.5, and K28.7) and "XX1x111100" (jhgfiedcba bits for negative running disparity comma character matching code groups K28.1, K28.5, and K28.7).
- The first word alignment character can be set by writing Quad Interface Register Offset Address 0x15 to 0x03.
- The second word alignment character can be set by writing Quad Interface Register Offset Address 0x16 to 0x7C.
- The mask register can be set to compare word alignment bits 6 to 0 by writing Quad Interface Register Offset Address 0x14 to 0x7F (a '1' in a bit of the mask register means check the corresponding bit in the word alignment character register).

**8b10b Decoder**
The 8b10b decoder implements an 8b10b decoder as described with the 802.3-2002 specification. The decoder performs the 10-bit to 8-bit code conversion along with verifying the running disparity.

When a code violation is detected, the 8b10b decoder sets the data rxd to 0xEE and rxc to '1'.

**Link State Machine**
The LatticeSC flexiPCS Serial RapidIO link synchronization state machine monitors the bit synchronization and code-group boundary alignment for a lane receiver. Each channel has its own independent link synchronization state machine.

The link synchronization state machine determines the bit synchronization and code-group boundary alignment state of a channel by monitoring the received code-groups and looking for /K28.5/s, other valid code-groups and invalid code-groups. The code-group /K28.5/ contains the "comma" bit sequence that is used to establish code-group boundary alignment. When a lane is error free, the "comma" pattern occurs only in the /K28.5/ code-group.

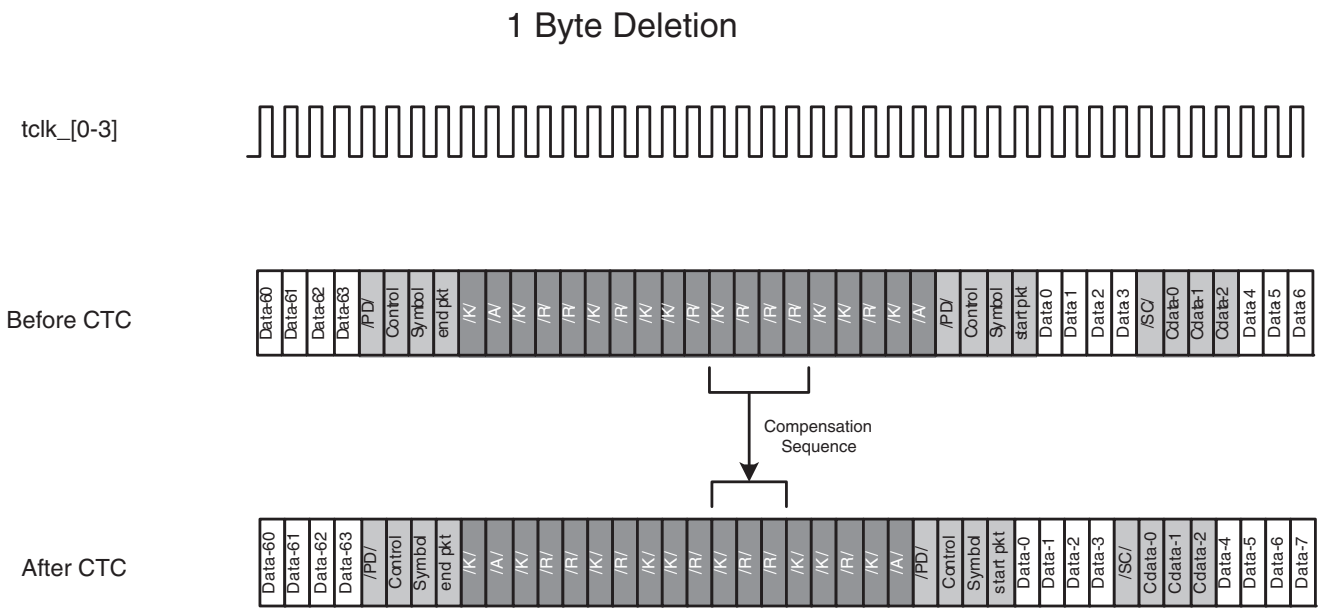
The link synchronization state machine implements the function described by the Lane_Synchronization State diagram in section VI of the Physical Layer 1x LP-Serial RapidIO specification.

**Multi-Channel Aligner**
The multi-channel aligner can be used to align two Serial RapidIO receive channels. When the LatticeSC flexiPCS is set to Serial RapidIO Mode, the multi-channel aligner can be used to create a 2x or 4x Serial RapidIO link. More information can be found in the **2x Serial RapidIO Application** and **4x Serial RapidIO Application** sections.

**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 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 LatticeSC SERDES (See DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet).

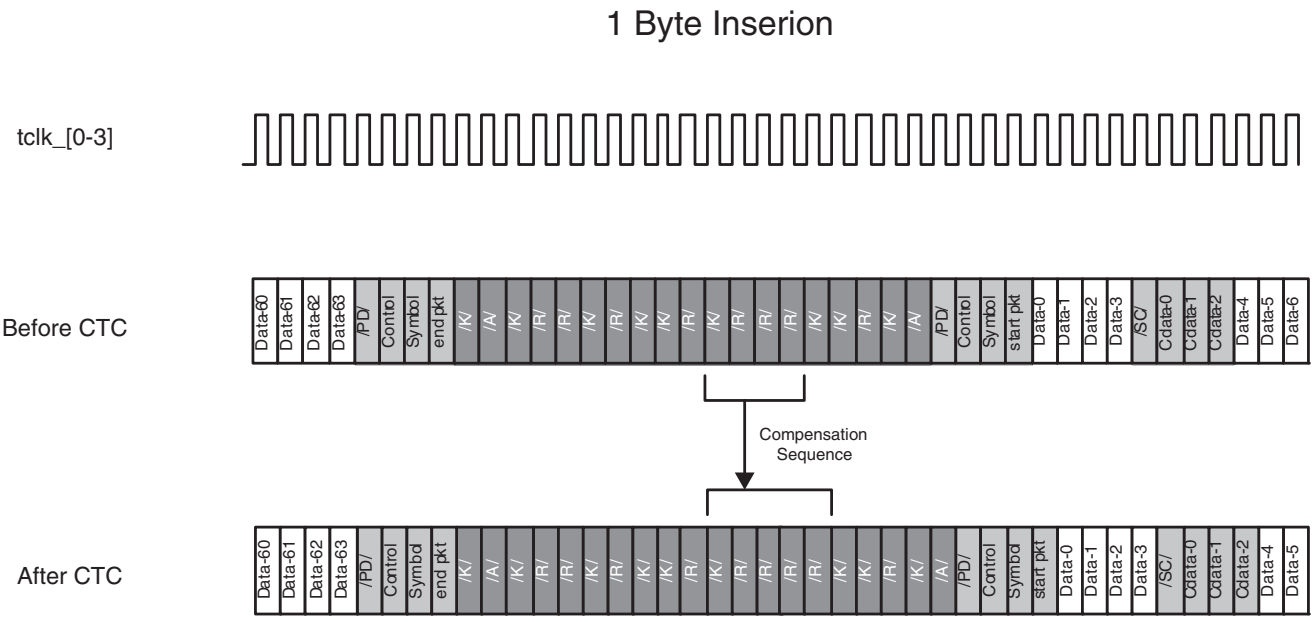
A diagram illustrating 1 byte deletion is shown in Figure 9-8:

***Figure 9-8. Serial RapidIO Mode Clock Compensation Byte Deletion Example***




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

***Figure 9-9. Serial RapidIO Mode Clock Compensation Byte Insertion Example***




Clock compensation values are set on a quad basis. The following settings for clock compensation should be set for Serial RapidIO applications:

- Set the insertion/deletion pattern length to 1 and minimum inter-packet gap to 1 bytes by setting Quad Interface

Register Offset Address 0x0E to 0x00. The minimum allowed inter-packet gap after skip character deletion based on these register settings is described below.

- The Serial RapidIO insertion/deletion pattern should be set to the /R/ character (K29.7). When 1 byte insertion/deletion is selected, the /R/ symbol can be set by writing Quad Interface Register Offset Address 0x12 to 0xFD and Quad Interface Register Offset Address 0x13 to 0x01.
- Set the clock compensation FIFO high water and low water marks to 9 and 7 respectively (appropriate for insertion/deletion pattern length of 1) by setting Quad Interface Register Offset Address 0x0D to 0x97.
- Clock compensation FIFO underrun can be monitored by reading the appropriate Channel Interface Register Offset Address 0x85, bit 6. This can be also monitored on the flexiPCS interface pin to FPGA logic labeled ctc_urun_[0-3] if “Optional Direct Control & Status Register Access” is selected when generating the flexiPCS block with IPexpress.
- Clock compensation FIFO overrun can be monitored by reading the appropriate Channel Interface Register Offset Address 0x85, bit7. This can be also monitored on the flexiPCS interface pin to FPGA logic labeled ctc_orun_[0-3] if “Optional Direct Control & Status Register Access” is selected when generating the flexiPCS block with IPexpress.

Table 9-7 shows the relationship between the user-defined values for inter-packet gap (written to Quad Interface Register Offset Address 0x0E, bits 6 and 7), and the guaranteed minimum number of bytes between packets after a skip character deletion from the flexiPCS. 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 Serial RapidIO, the number of bytes per insertion/deletion is 1 (Quad Interface Register Offset Address 0x0E, bits 4 and 5 are both set to ‘1’). If Quad Interface Register Offset Address 0x0E, bits 6 and 7 are set to “00”, then the minimum inter-packet gap is equal to 1 times 1 or 1 byte. The flexiPCS will not perform a skip character deletion until the minimum number of inter-packet bytes have been detected.

***Table 9-7. Minimum Inter-Packet Gap Multiplier***

| Quad Interface Register Offset Address 0x0E | | Insertion/Deletion Multiplier Factor |
|---|---|---|
| Bit 6 | Bit 7 | |
| 0 | 0 | 1 X |
| 0 | 1 | 2 X |
| 1 | 0 | 3 X |
| 1 | 1 | 4 X |

**Serial RapidIO Receive State Machine**

The Serial RapidIO receive state machine reverses the operation performed by the Serial RapidIO transmit state machine by converting the randomized /A/, /K/, and /R/ PCS code groups to Serial RapidIO idle characters (rxd=0x07, rxc=1).

## Control & Status

The Control & Status pins for the 1x Serial RapidIO mode can be optionally selected on a per quad basis when generating the flexiPCS quad interface files with the ispLEVER tools. Each of these control and status functions are also accessible through equivalent Quad Interface and Channel Interface Registers. The control and status signals allow direct real time access of these functions from FPGA logic.

**Control**

**felb_[0-3]** - Per channel, active high control signal which sets up the appropriate channel in Far-End Loopback mode. This mode is generally used for testing the flexiPCS logic, looping back receive data back to the transmit direction without crossing the FPGA logic interface. For more information on the details of Far-End Loopback mode, see the **flexiPCS Testing** Section of the flexiPCS Data Sheet. The Far-End Loopback mode can also be initiated by writing Channel Interface Register Offset Address 0x00, bit 5 to ‘1’.

**lsm_en_[0-3]** - Per channel Receive Link State Machine Enable. A change (either 0 to 1 or 1 to 0) on the lsm_en_[0-3] resets the Receive Link State Machine into No Sync mode. The Receive Link Machine Enable can also be set by writing Channel Interface Register Offset Address 0x00, bit 7 to '1'.

**mca_align_en_[0-3] -** Per channel active high Multi-channel Align enable. Multi-channel Alignment Enable can also be set by writing the appropriate Quad Interface Register Offset Address 0x04, bits [4:7] to '1' (bit 4 for channel 3, bit 5 for channel 2, bit 6 for channel 1, and bit 7 for channel 0).

**mca_resync_[01/23]** - Active high, asynchronous reset to Multi-channel Aligner state machine and aligner FIFO control logic. **mca_resync_01** resets logic in channels 0 and 1 in two-channel alignment mode and all four channels in four channel alignment mode. **mca_resync_23** resets logic in channels 2 and 3 in two-channel alignment mode (not used in four channel alignment mode).

**Status**

**lsm_status_[0-3]** - Per channel active high signal from the Receive Link State Machine indicating link synchronization successful. The link synchronization status can also be read from the appropriate Quad Interface Register Offset Address 0x84, bits [4:7] (bit 4 for channel 0, bit 5 for channel 1, bit 4 for channel 2, and bit 7 for channel 3).

**mca_aligned_[01/23]** - Active high signal indicating successful alignment of channels 0/1 or channels 2/3. mca_aligned_01 high indicates successful alignment of channels 0 and 1 in two-channel alignment mode and all four channels in four channel alignment mode. mca_aligned_23 high indicates successful alignment of channels 2 and 3 in two-channel alignment mode (Always low in four channel alignment mode). The Multi-channel Alignment status for channels 0/1 (equivalent to the mca_aligned_01 output) can also be read from the Quad Interface Register Offset Address 0x82, bit 2. The Multi-channel Alignment status for channels 2/3 (equivalent to the mca_aligned_23 output) can also be read from the Quad Interface Register Offset Address 0x82, bit 3. mca_aligned_01 is only valid when performing multichannel alignment within a single quad.

**ctc_urun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has underrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 6.

**ctc_orun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has overrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 7.

## 4x Serial RapidIO Operation

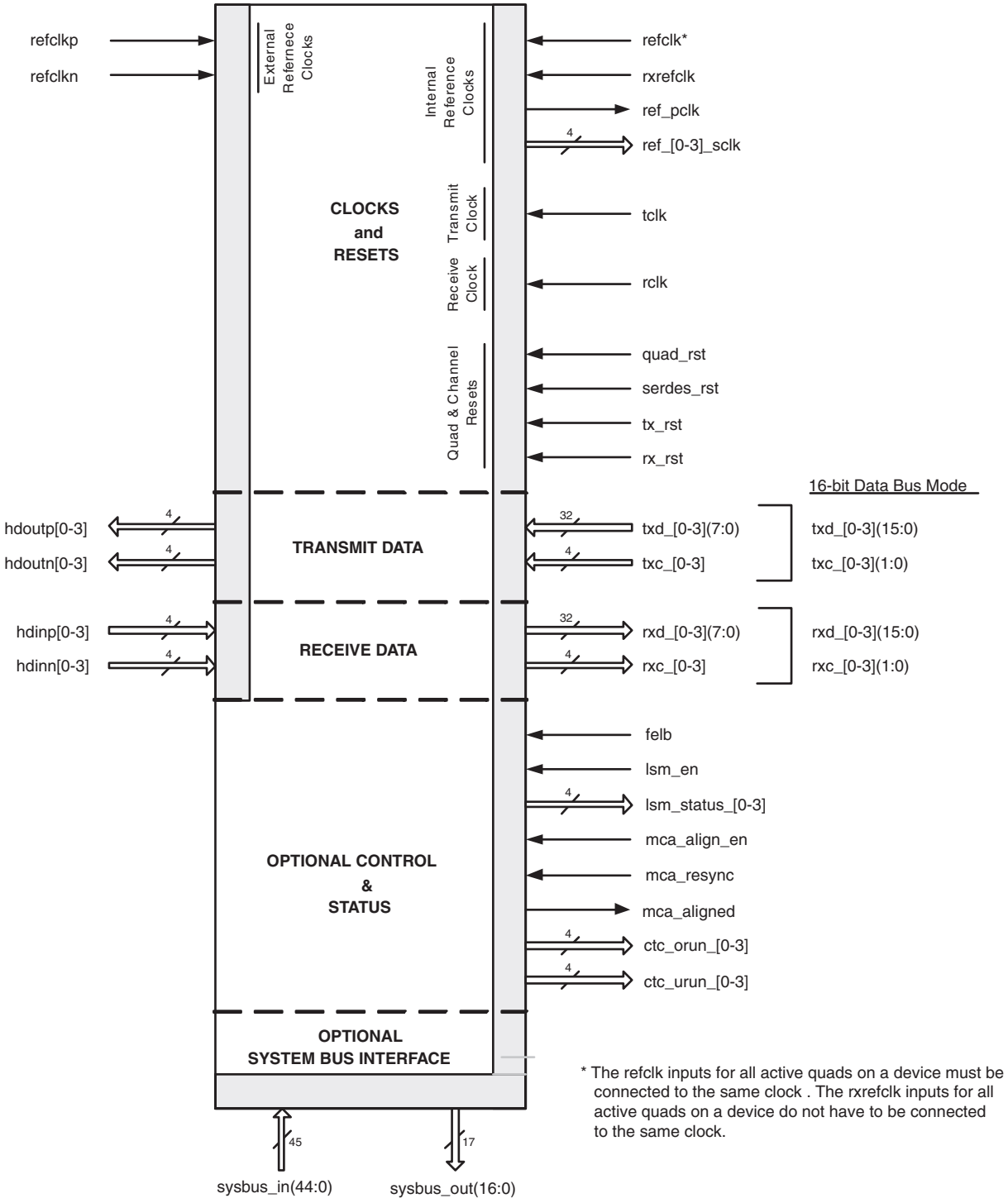
A single LatticeSC quad can support a single 4x Serial RapidIO link with the use of the embedded multi-channel aligner. This is done by using a flexiPCS quad in Serial RapidIO Mode and setting the Multi-channel Aligner block appropriately.

The main differences between 1x and 4x Serial RapidIO operation are:

1. For 4x Serial RapidIO operation, all four channels in a flexiPCS QUAD must be enabled.
2. For 4x Serial RapidIO operation, the Multi-channel Aligner is enabled in the Receive direction to provide alignment between all four channels as dictated by the Align column symbol ||A|| as defined in the 1x/4x Serial RapidIO Specification.
3. For 4x Serial RapidIO operation, the clock tolerance compensation logic always inserts or deletes across all four columns simultaneously to preserve multi-channel alignment.

IPexpress allows the designer to choose the mode for each flexiPCS quad used in a given design. A minimized pinout for 4x Serial RapidIO operation is provided by IPexpress. In general, this means that all per channel input pins at the FPGA interface are tied together in the flexiPCS and presented as one pin. Figure 9-10 displays the resulting port interface to one flexiPCS quad that has been set to Serial RapidIO mode with "Align channels 0,1,2, and 3" selected within IPexpress.

***Figure 9-10. Serial RapidIO Mode Pin Diagram (Four channel alignment selected)***



## 4x Serial RapidIO Operation Pin Description

Table 9-8 lists the inputs and outputs to/from a flexiPCS quad targeted to 4x Serial RapidIO operation by selecting "Align channels 0, 1, 2, and 3" in IPexpress. A more detailed description of the function of each additional port is given in the **4x Serial RapidIO Operation Detailed Description**.

***Table 9-8. Serial RapidIO Mode Pin Description (Four channel alignment selected)***

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Reference Clocks** | | | |
| refclkp | In from I/O pad | N/A | Reference clock input, positive. Dedicated CML input. |
| refclkn | In from I/O pad | N/A | Reference clock input, negative. Dedicated CML input |
| refclk | In from FPGA | N/A | Optional reference clock input from FPGA logic. Can be used instead of per quad reference clock.<br><br>The refclk inputs for all active quads on a device must be connected to the same clock. |
| rxrefclk | In from FPGA | N/A | Optional receive reference clock input from FPGA logic. Can be used instead of per quad reference clock. The rxrefclk inputs for all active quads do not have to be connected to the same clock. |
| ref_pclk | Out to FPGA | N/A | Locked reference clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| ref_[0-3]_sclk | Out to FPGA | N/A | Per channel locked reference clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| **Transmit Clock** | | | |
| tclk | In from FPGA | N/A | Transmit clock input from FPGA. Used to clock the TX data phase compensation FIFO with clock synchronous to the clock. May also be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Receive Clock** | | | |
| rclk | In from FPGA | N/A | Receive clock input from FPGA. May be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference and/or receive reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Resets** | | | |
| quad_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of SERDES and PCS logic. |
| serdes_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of the SERDES. |
| tx_rst | In from FPGA | ASYNC | Active high, asynchronous reset of all four transmit channels of PCS logic. |
| rx_rst | In from FPGA | ASYNC | Active high, asynchronous reset of all four receive channels of PCS logic. |
| **Transmit Data** | | | |
| hdoutp[0-3] | Out to I/O pad | N/A | High-speed CML serial output, positive. |
| hdoutn[0-3] | Out to I/O pad | N/A | High-speed CML serial output, negative. |
| txd_[0-3](7:0)<br><br>txd_[0-3](15:0)* | In from FPGA | tclk_[0-3] | Per channel parallel transmit data bus from the FPGA. Bus is 8-bits wide if QIR 0x19, bit 4 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 4 = '1'. |
| txc_[0-3]<br><br>txc_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel transmit control character indication.<br><br>*Bus is 2 bits wide if QIR 0x19, bit 4 = '1'. |
| **Receive Data** | | | |
| hdinp[0-3] | In from I/O pad | N/A | High-speed CML serial input, positive. |

*Table 9-8. Serial RapidIO Mode Pin Description (Four channel alignment selected)*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| hdinn[0-3] | In from I/O pad | N/A | High-speed CML serial input, negative. |
| rxd_[0-3](7:0)<br><br>rxd_[0-3](15:0)* | Out to FPGA | rclk_[0-3] | Per channel parallel receive data bus to the FPGA. Bus is 8-bits wide if QIR 0x19, bit 5 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 5 = '1'. |
| rxc_[0-3]<br><br>rxc_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel receive control character indication.<br><br>*Bus is 2 bits wide if QIR 0x19, bit 5 = '1'. |
| **Optional Control & Status** | | | |
| felb | In from FPGA | ASYNC | Active high far-end loopback enable for all four channels. |
| lsm_en | In from FPGA | ASYNC | Receive link state machine enable for all four channels. |
| lsm_status_[0-3] | Out to FPGA | ASYNC | Per channel signal from receive link state machine indicating successful link synchronization. |
| mca_align_en | In from FPGA | ASYNC | Active high multi-channel aligner enable for all four channels. |
| mca_resync | In from FPGA | ASYNC | Active high async multi-channel aligner resynchronization signal. |
| mca_aligned | Out to FPGA | ASYNC | Active high indicating successful alignment of channels. |
| ctc_urun_[0-3] | Out to FPGA | ASYNC | Per channel active high flag indicator that clock tolerance compensation FIFO has underrun. |
| ctc_orun_[0-3] | Out to FPGA | ASYNC | Per channel active high flog indicator that clock tolerance compensation FIFO has overrun. |
| **Optional System Bus Interface** | | | |
| sysbus_in(44:0) | In from FPGA | N/A | Control and data signals from the internal system bus. |
| sysbus_out(16:0) | Out to FPGA | N/A | Control and data signals to the internal system bus. |

# 4x Serial RapidIO Operation Detailed Description

The following section provides a detailed description of the operation of a flexiPCS quad targeted to 4x Serial RapidIO operation. The functional description is organized according to the port listing shown in Figure 9-10. The System Bus base address for any control and status registers relevant to a given function are provided with the description of the operation of that function.

## Clocks & Resets

Clocking setup for a flexiPCS quad set to 4x Serial RapidIO operation is generally the same as that of a quad set to Serial RapidIO mode. Refer to the **Clocks & Resets** description in the **Serial RapidIO Mode Detailed Functional Description** section. A summary of the clocks that are different for 4x Serial RapidIO operation are listed below:

**tclk** - One transmit clock input from the FPGA logic to the flexiPCS is provided for all four channels of data (rather than a separate clock for each channel).

**rx_pclk** - One receive clock output from the flexiPCS to the FPGA logic is provided for connection to a primary clock route. This clock is derived from the recovered clock on Channel 0 by default.

**rclk** - One receive clock input from the FPGA logic to the flexiPCS is provided for all four channels of data for 4x Serial RapidIO operation (rather than a separate clock for each channel).

For 4x Serial RapidIO applications, the bit clock rate is typically 3.125 Gbps. This bit clock rate falls into the range defined as "full rate" mode for the SERDES PLLs. Full rate mode is selected separately for each channel and each direction by writing the appropriate Channel Interface Register Base Address 0x13, bit 5 (transmit) and bit 6 (receive) to a '0' (default).

Table 9-5 in the **Serial RapidIO Mode Detailed Description** section shows the possible reference clock frequencies for various reference clock modes which result in a 3.125 Gbps internal bit rate.

Additional information on all reference clock rate modes can be found in the **SERDES Functionality** section of the flexiPCS Data Sheet.

**Quad & Channel Resets**
Resets for a flexiPCS quad targeted to 4x Serial RapidIO operation are generally the same as that of a quad set to 1x Serial RapidIO mode. Refer to the **Clocks & Resets** description in the **Serial RapidIO Mode Detailed Functional Description** section. A summary of the channel resets that are different for 4x Serial RapidIO operation are listed below:

**tx_rst** - Active high, asynchronous signal from FPGA resets all four transmit channels of PCS logic. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [4:7]. Bit 7 resets transmit channel 0, bit 6 resets transmit channel 1, bit 5 resets transmit channel 2, and bit 4 resets transmit channel 3.

**rx_rst** - Active high, asynchronous signals from FPGA resets all four receive channels of PCS logic. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [0:3]. Bit 3 resets receive channel 0, bit 2 resets receive channel 1, bit 1 resets receive channel 2, and bit 0 resets receive channel 3.

## Transmit Data

When configured for 4x Serial RapidIO operation, a flexiPCS quad converts a 32-bit wide 4x Serial RapidIO compliant data stream at the FPGA logic interface to a high-speed serial line interface at the device outputs.

**Transmit State Machine**
The Serial RapidIO Transmit State Machine performs the conversion of Serial RapidIO idle control column, ||I|| (txd = 0x07, txc=1), to a randomized sequence of ||A||, ||K||, ||R|| code-group columns to enable lane synchronization, clock compensation and lane-to-lane alignment as described in the Physical Layer 1x/4x LP_Serial RapidIO specification. This includes a PRBS counter ($X^7 + X^6 + 1$) as described in the Pseudo-Random Idle Code-Group Generator diagram in the Serial RapidIO specification to provide the random code selection. Figure 9-11 shows an example of 4x Serial RapidIO data flow with Idle:

***Figure 9-11. 4x Serial RapidIO Compliant Data Stream Example with Idles***

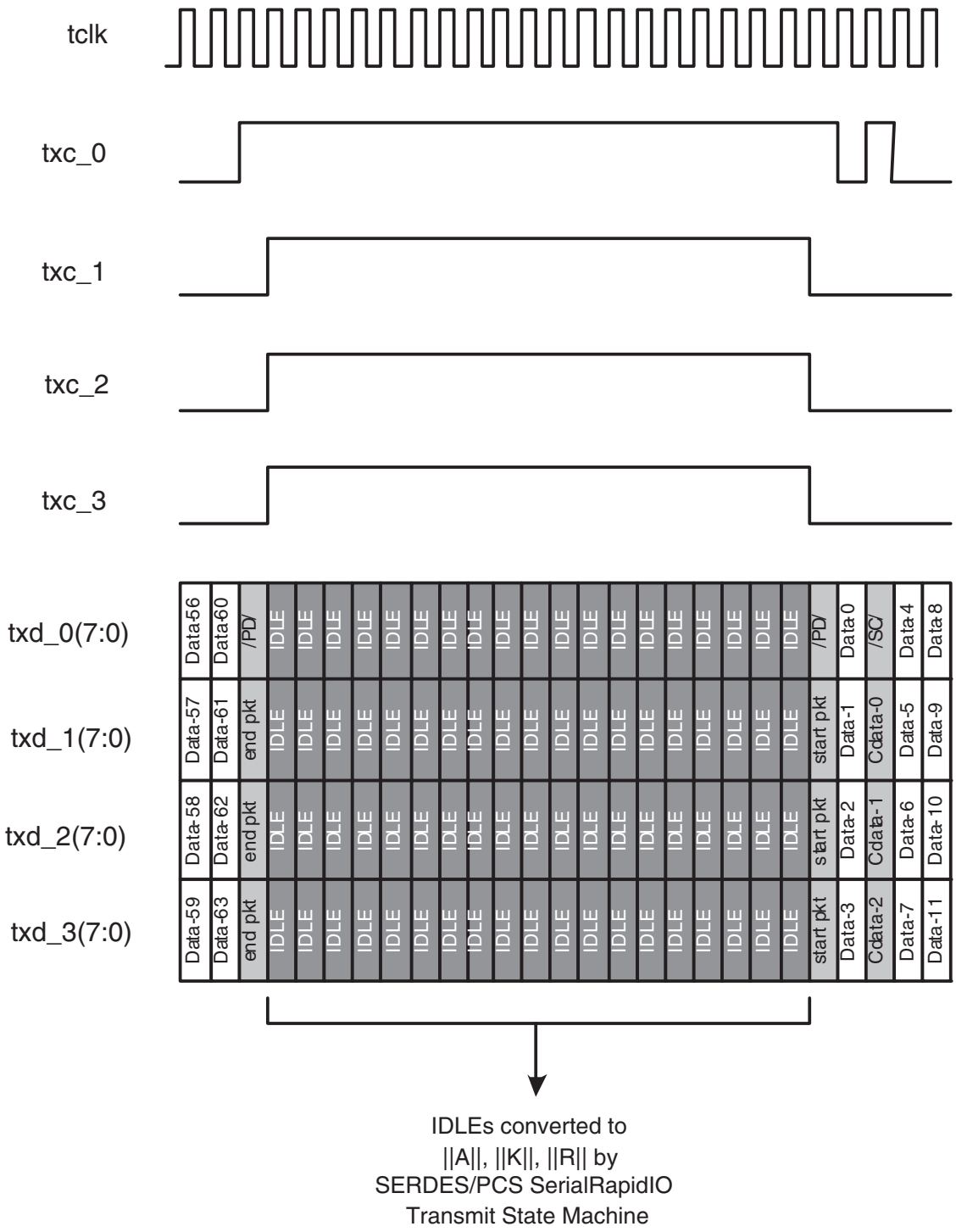
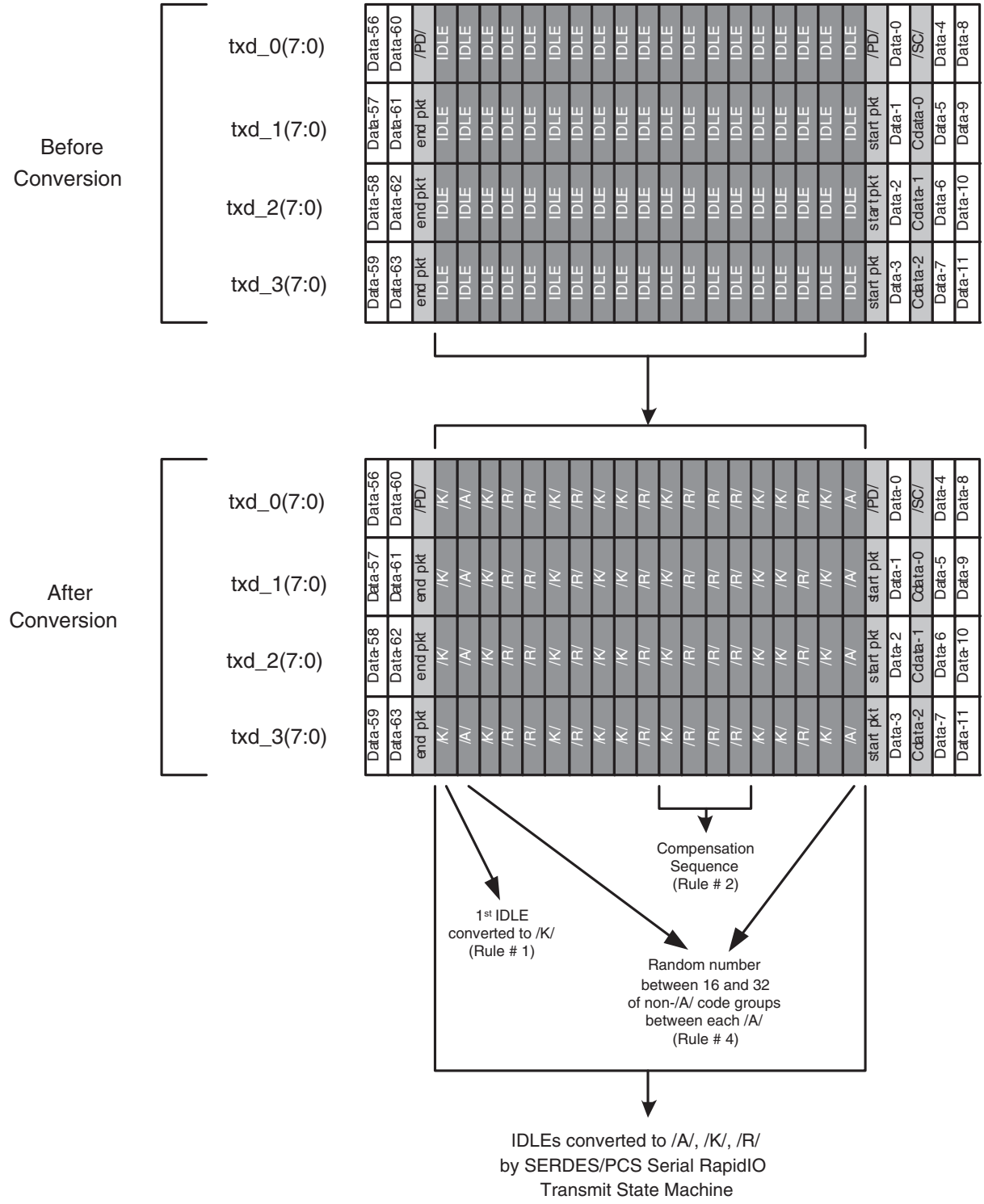


Figure 9-12 shows an example of 4x Serial RapidIO data flow with idle code groups converted by the flexiPCS Serial RapidIO Transmit State Machine:

***Figure 9-12. 4x Serial RapidIO Idle Code Group Conversion Example***



## Receive Data

When configured into Serial RapidIO mode with four channel alignment, a flexiPCS quad provides a receive data path from a high-speed serial line interface at the device inputs to 4 8-bit parallel interfaces at the FPGA logic interface.

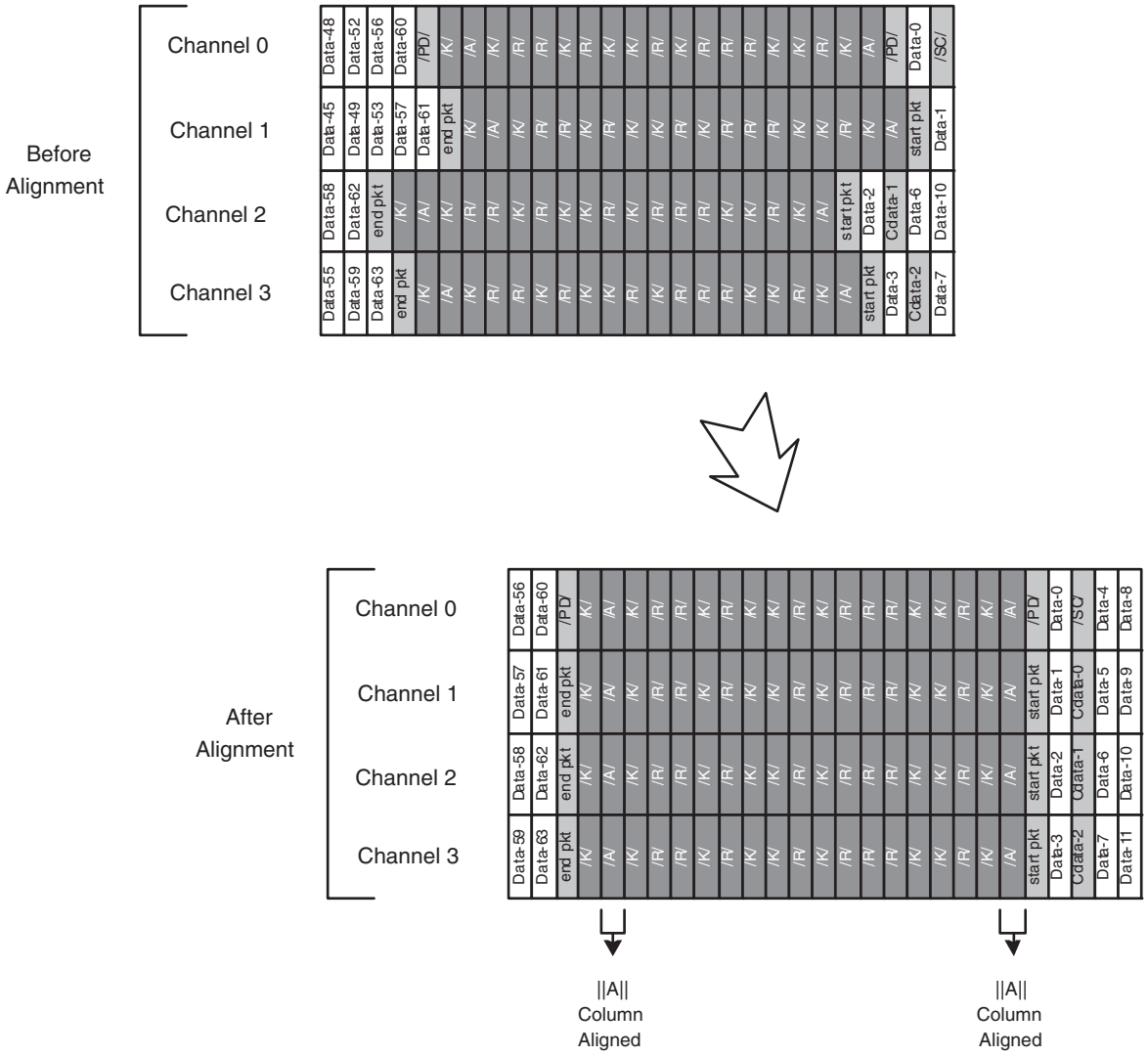
**Multi-channel Aligner**

The Multi-channel alignment module performs the operations and functions to control the multi-channel alignment process. This includes the ||A|| detect and the alignment status signal generation required within the implementation of the alignment state machine as described in the Lane Alignment State Machine Diagram (Figure 4-10) in Part VI of the Physical Layer 1x/4x Serial RapidIO Specification.

The flexiPCS Multi-channel Aligner allows for alignment of two or four channels in a quad. For 2x Serial RapidIO operation, alignment is performed for two channels in a quad. The Multi-channel Aligner will align channels based on a user-defined alignment character (referred to as ||A|| for Serial RapidIO applications) which must be present in the data stream for all receive channels that are to be aligned. This character is inserted simultaneously in all channels to be aligned during an Idle sequence between packets upon transmission. If arriving data is skewed due to unequal transport delays, the presence of the alignment characters allows the Multi-channel Aligner to re-align the skewed channels.

Figure 9-13 shows an example of the operation of the flexiPCS multi-channel aligner operation for 4x Serial RapidIO applications. The ||A|| Align code-groups in each channel are lined up by the multi-channel aligner to create an aligned data stream at the PCS/FPGA interface.

***Figure 9-13. x4 Serial RapidIO Four Channel Alignment Example***




The following is a procedure for settings registers for Multi-channel Alignment.

1. Set the mode for the Multi-channel Aligner and enable/disable the appropriate channels for alignment.

- For 4x Serial RapidIO operation, set 4 channel alignment for channels 0, 1, 2, and 3 by writing Quad Interface Register Offset Address 0x19, bit 1 to a '1'.
- Each channel to be aligned must also be enabled for alignment by writing to the appropriate bit in Quad Interface Register Offset Address 0x04. Write a '1' to bit 7 to include channel 0 for alignment. Write a '1' to bit 6 to include channel 1 for alignment. Write a '1' to bit 5 to include channel 2 for alignment. Write a '1' to bit 4 to include channel 3 for alignment. Multi-channel alignment can also be enabled by setting the **mca_align_en** pin at the FPGA interface high.

2. Set the Multi-channel Alignment output clocks. A clock domain transfer occurs between the inputs and outputs of the Multi-channel Aligner. Each channel's receive data is clocked into the Multi-channel Aligner with its own separate recovered receive clock. Each channel's receive data is clocked out of the Multi-channel Aligner on a unique multi-channel receive clock. Each multi-channel receive clock can be programmed to be any of the recovered receive clocks.

It is expected that the user will assign the same recovered receive clock to all the multi-channel output clocks for every channel that is to be aligned. That way, all aligned channels coming out of the Multi-channel Aligner are effectively clocked by the same clock. For more information on setting up a multi-channel receive clock, refer to the **Multi-Channel Alignment** section of the LatticeSC/M Family flexiPCS Data Sheet.

For example, in a 4 channel alignment application, in order to set up all Multi-channel Alignment clocks to be sourced from the channel 0 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0x00. To set all Multi-channel Alignment clocks to be sourced from the channel 1 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0x55. To set all Multi-channel Alignment clocks to be sourced from the channel 2 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xAA. To set all Multi-channel Alignment clocks to be sourced from the channel 3 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xFF.

3. Set the Multi-channel Alignment characters. The Multi-channel Aligner provides the ability to align channels of incoming data to one of two 10-bit user defined maskable patterns. Both alignment characters should be set to the same value if only one alignment character is defined for an application. For 4x Serial RapidIO operation, the alignment character should be set to ||A|| (K27.7).

- A user programmable mask word allows the user to set which individual bits are compared to the user defined channel alignment character. When a '1' is present in the user programmable mask, the corresponding bit of the channel alignment character is compared. When '0' is present in the user programmable mask, the corresponding bit of the channel alignment character is not compared. For 4x Serial RapidIO operation, set the lowest 9 significant bits of the user programmable mask by writing Quad Interface Register Offset Address 0x07 to 0xFF and Quad Interface Register Offset Address 0x0A, bit 3 to '1'.
- The first channel alignment character can be set to the Serial RapidIO Align column value ||A|| (K27.7) by writing Quad Interface Register Offset Address 0x08 to 0xFB. The first channel K character can be set by writing Quad Interface Register Offset Address 0x0A, bit 5 to '1'.
- The second channel alignment character can be set to the Serial RapidIO Align column value ||A|| (K27.7) by writing Quad Interface Register Offset Address 0x09 to 0xFB. The second channel K character can be set by writing Quad Interface Register Offset Address 0x0A, bit 7 to '1'.

4. Set the Multi-channel Aligner FIFO latency and high-water mark values. Latency values from 0 to 31 can be set by writing to Quad Interface Register Offset Address 0x05, bits [3:7]. The FIFO high-water mark values from 0 to 63 can be set by writing to Quad Interface Register Offset Address 0x06, bits [2:7]. The FIFO high-water mark should be set to the maximum the number of bytes of skew allowed for de-skewing. Larger values of FIFO high-water mark increase the chance of FIFO overrun or underrun. For more information on choosing latency and high-water mark values, refer to the Multi-Channel Alignment section of the LatticeSC/M Family flexiPCS Data Sheet.

5. Reset the Multi-channel Aligner state machine and FIFO control logic if desired with the **mca_resync** pin at the FPGA interface. **mca_resync** is an active high asynchronous reset which resets the Multi-channel Aligner for all four channels.

When the Multi-channel Aligner is set to 4-channel alignment, the Multi-channel Alignment state machine for all four channels can be disabled by writing Quad Interface Register Offset Address 0x04, bit 3 to a '1'. When the Multi-channel Alignment state machine is disabled, the alignment state machine will be held in the "LOSS_OF_ALIGNMENT" state and no alignment will be performed for the relevant channels.

**Clock Tolerance Compensation**

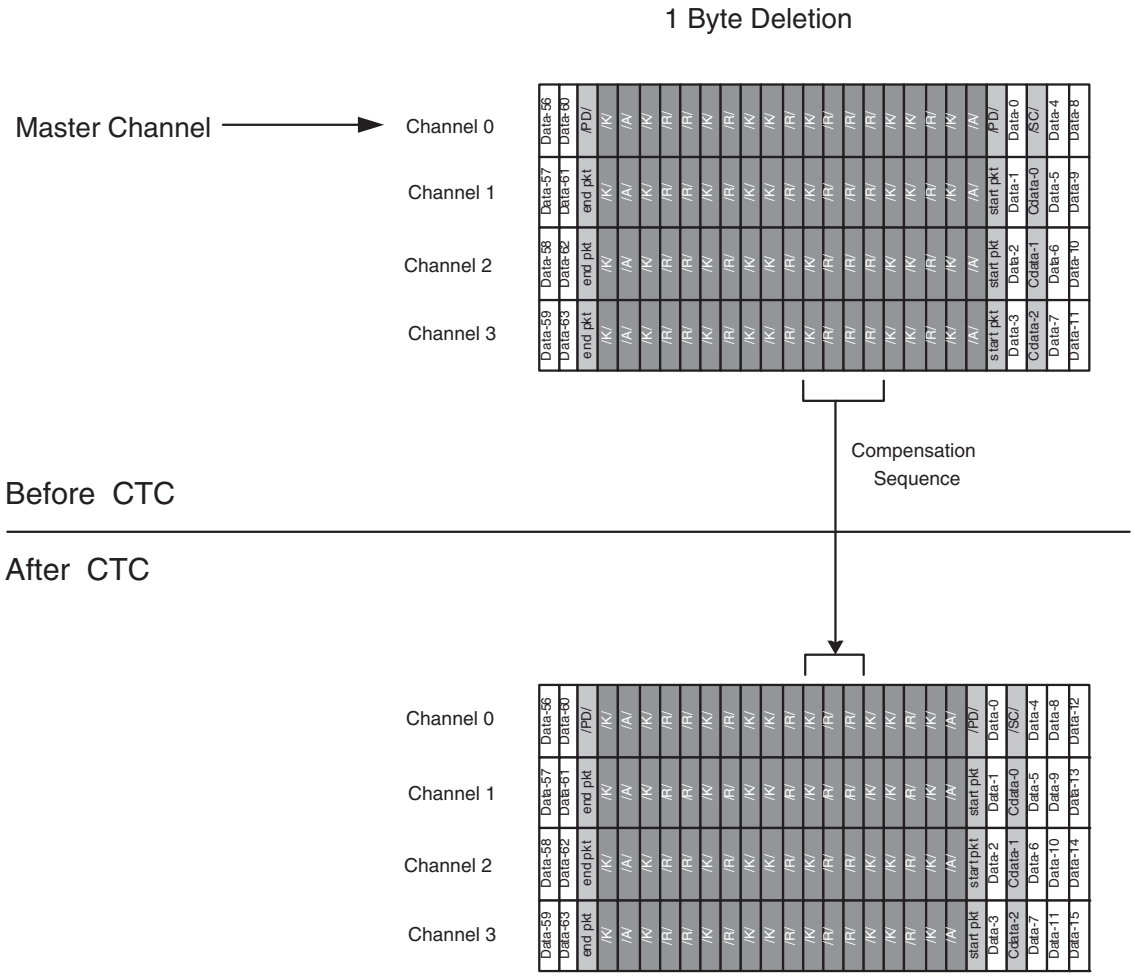
When insertion/deletion is performed by the clock tolerance compensation logic for 4x Serial RapidIO operation, insertion or deletion is performed across all aligned channels simultaneously (to preserve the multi-channel alignment).

Note that when four channel alignment is selected, channel 0 is selected as the master channel for performing clock tolerance compensation. This means that only channel 0 is monitored for the presence of the SKIP character.

Insertion or deletion is performed on all four channels simultaneously based on detection of the SKIP character in channel 0. This means that channel 0 must be active and transmitting Serial RapidIO compliant data for the clock tolerance compensation function to work in four channel alignment mode. If channel 0 is not active, receive data will not be present on the rxd outputs of the flexiPCS.

A diagram illustrating 1 byte deletion for 4x Serial RapidIO operation is shown in Figure 9-14:

***Figure 9-14. 4x Serial RapidIO Operation Clock Compensation Byte Deletion Example***




A diagram illustrating 1 byte insertion for 4x Serial RapidIO operation is shown in Figure 9-15:

***Figure 9-15. 4x Serial RapidIO Operation Clock Compensation Byte Insertion Example***

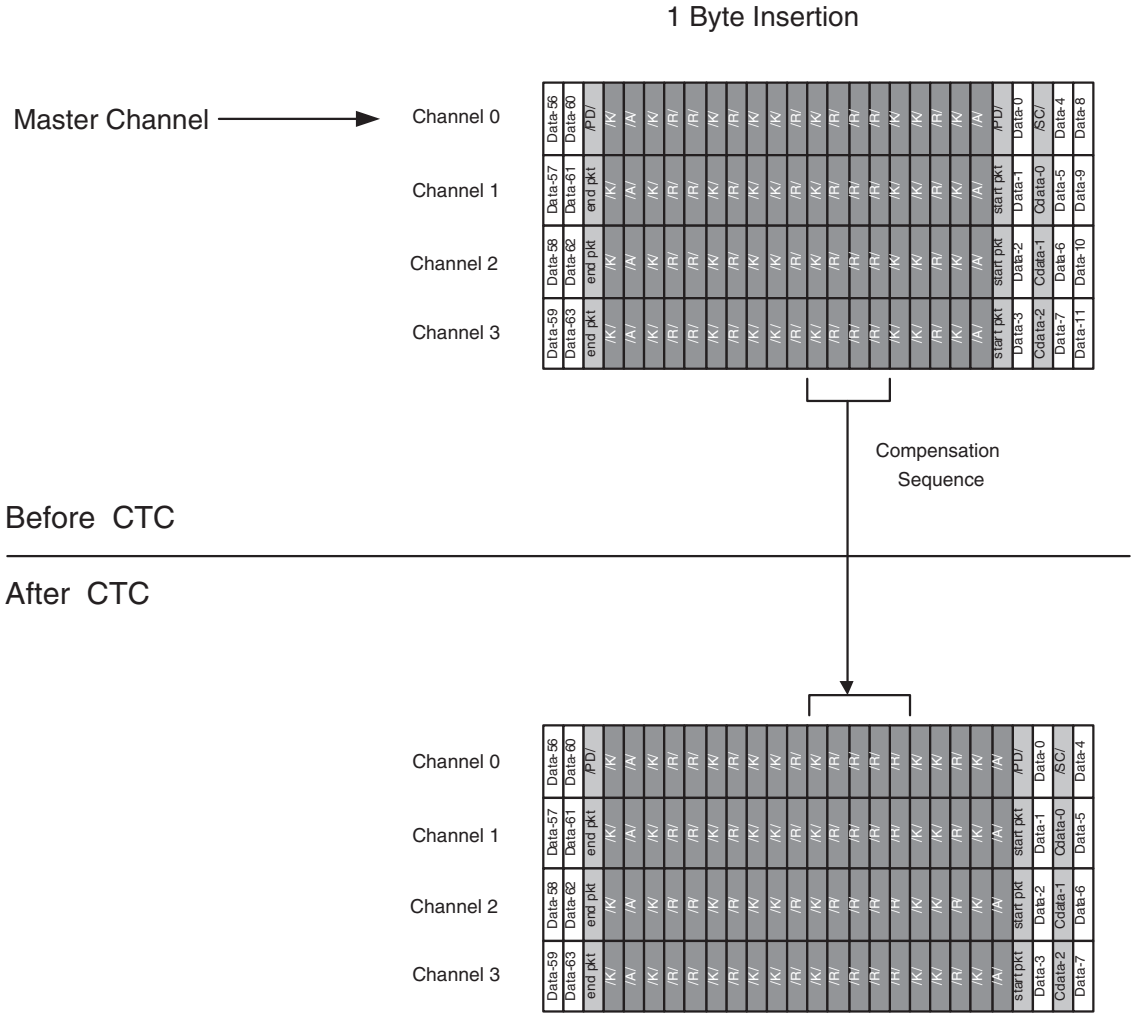
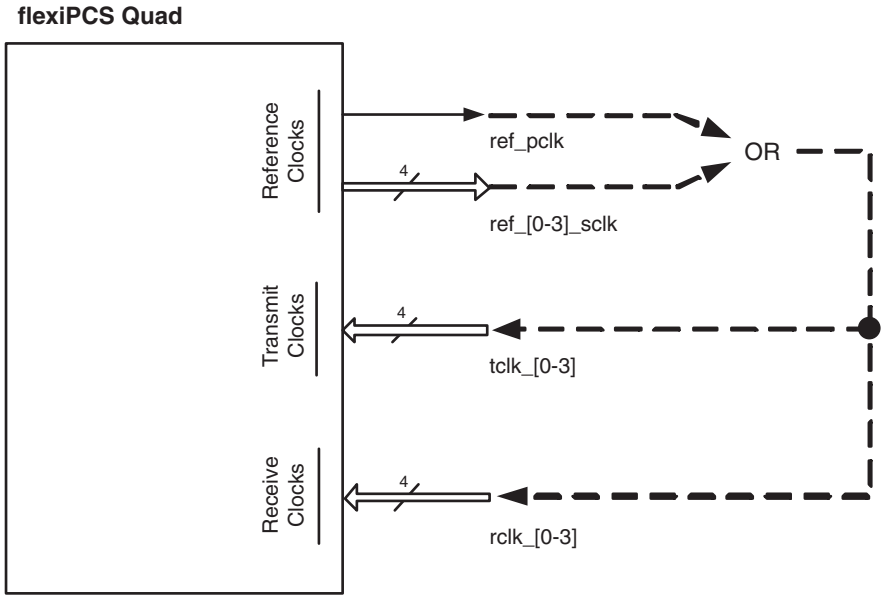



Figure 9-16 shows the clock domains for both transmit and receive directions for a single channel inside the flexiPCS.

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

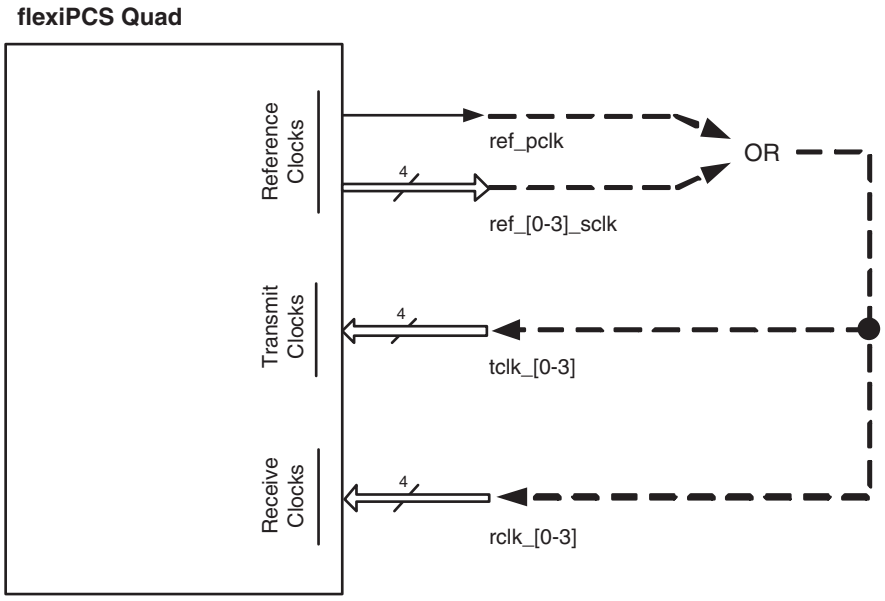
On the receive side, a clock domain transfer from the recovered channel clock to the Multi-channel Alignment channel clock occurs at the Multi-channel Aligner. A second clock domain transfer from the Multi-channel Alignment channel clock to the locked reference clock occurs at the Clock Tolerance Compensator. A third clock domain transfer occurs from the locked reference clock to the rclk input at the FPGA interface at the FPGA Receive Interface phase compensation FIFO. The FPGA Receive Interface phase compensation FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. These phase compensation FIFOs (one per channel) cannot compensate for frequency variations.

***Figure 9-16. flexiPCS Clock Domain Transfers for 4x Serial RapidIO Operation***




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 flexiPCS quad set to Serial RapidIO mode. Figure 9-17 illustrates the possible connections that would result in a synchronous interface.

***Figure 9-17. Synchronous Input Clocks to flexiPCS Quad Set to Serial RapidIO Mode***




## Control & Status

The Control & Status pins for the 4x Serial RapidIO operation can be optionally selected on a per quad basis when generating the flexiPCS quad interface files with the ispLEVER tools. Each of these control and status functions are also accessible through equivalent Quad Interface and Channel Interface Registers. The control and status signals allow direct real time access of these functions from FPGA logic. In addition to the Control and Status pins common to the Serial RapidIO Mode, a flexiPCS quad targeted to 4x Serial RapidIO operation has additional Control & Status pins related to control and monitoring of the multi-channel aligner.

**Control**

**felb** - Active high control signal which sets up all four channels in Far-End Loopback mode. This mode is generally used for testing the flexiPCS logic, looping back receive data back to the transmit direction without crossing the FPGA logic interface. For more information on the details of Far-End Loopback mode, see the **flexiPCS Testing** Section of the flexiPCS Data Sheet. The Far-End Loopback mode can also be initiated by writing Channel Interface Register Offset Address 0x00, bit 5 to '1'.

**lsm_en** - Active high Receive Link State Machine Enable for all four channels. A change on lsm_en resets the Receive Link State Machines into No Sync mode. The Receive Link Machine Enable can also be set by writing Channel Interface Register Offset Address 0x00, bit 7 to '1'.

**mca_align_en -** Active high multi-channel align enable. Multi-channel Alignment Enable can also be set by writing the appropriate Quad Interface Register Offset Address 0x04, bits [4:7] to '1' (bit 4 for channel 3, bit 5 for channel 2, bit 4 for channel 1, and bit 7 for channel 0).

**mca_resync** - Active high, asynchronous reset to multi-channel aligner state machine and aligner FIFO control logic. mca_resync resets logic in all four channels in four channel alignment mode.

**Status**

**lsm_status_[0-3]** - Per channel signal from the Receive Link State Machine indicating link synchronization successful. The link synchronization status can also be read from the appropriate Quad Interface Register Offset Address 0x84, bits [4:7] (bit 4 for channel 0, bit 5 for channel 1, bit 4 for channel 2, and bit 7 for channel 3).

**mca_aligned** - Active high signal indicating successful alignment of all four channels in four channel alignment mode. The multi-channel alignment status can also be read from the Quad Interface Register Offset Address 0x82, bit 2.

**ctc_urun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has underrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 6.

**ctc_orun_[0-3]** - Per channel active high flag indication that the clock tolerance compensation FIFO has overrun. These flags can also be read from the appropriate Channel Interface Register Offset Address 0x85, bit 7.

## Status Interrupt Registers

Some of the status registers associated with Serial RapidIO operation have an associated status interrupt and status interrupt enable register. Any flexiPCS status interrupt register will generate an interrupt to the Systembus block (if used in the design). For a description of the operation of interrupts with the Systembus, refer to the Systembus section of the LatticeSC/M Family Data Sheet. Operation of the interrupt registers for the flexiPCS is as follows:

User must set the appropriate status interrupt enable bit to a '1'

Whenever the associated status bit goes high, its status interrupt bit will also go high. The status interrupt bit will remain high even if the associated status bit goes low.

The status interrupt bit will remain high until it is read, when it will automatically be cleared. If the associated status bit is still high, then the status interrupt bit will go high again (until read the next time).

Table 9-9 shows a listing of the status registers associated with Serial RapidIO operation which have a corresponding status interrupt and status interrupt enable bit.

*Table 9-9. Status Interrupt Register Bits*

| Status Register Bit Function | Status Register Bit Address | Status Interrupt Enable Register Bit Address | Status Interrupt Register Bit Address |
|---|---|---|---|
| Receive Link State Machine Status (lsm_status) Channel 0 | QIR 0x84, bit 4 | QIR 0x1C, bit 4 | QIR 0x85, bit 4 |
| Receive Link State Machine Status (lsm_status) Channel 1 | QIR 0x84, bit 5 | QIR 0x1C, bit 5 | QIR 0x85, bit 5 |
| Receive Link State Machine Status (lsm_status) Channel 2 | QIR 0x84, bit 6 | QIR 0x1C, bit 6 | QIR 0x85, bit 6 |
| Receive Link State Machine Status (lsm_status) Channel 3 | QIR 0x84, bit 7 | QIR 0x1C, bit 7 | QIR 0x85, bit 7 |
| Multi-channel Alignment State Machine Status (mca_aligned_01 in x1 mode or mca_aligned in x4 mode)<br><br>Channels 0 & 1 (2-channel alignment mode)<br><br>Channels 0,1,2 & 3 (4-channel alignment mode) | QIR 0x82, bit 2 | QIR 0x0C, bit 2 | QIR 0x83, bit 2 |
| Multi-channel Alignment State Machine Status (mca_aligned_23)<br><br>Channels 2 & 3 (2-channel alignment mode)<br><br>Inactive (4-channel alignment mode) | QIR 0x82, bit 3 | QIR 0x0C, bit 3 | QIR 0x83, bit 3 |
| Clock Tolerance Compensation FIFO underrun | CIR 0x85, bit 6 | CIR 0x0A, bit 6 | CIR 0x8A, bit 6 |
| Clock Tolerance Compensation FIFO overrun | CIR 0x85, bit 7 | CIR 0x0A, bit 7 | CIR 0x8A, bit 7 |

# SONET Mode
# LatticeSC/M flexiPCS Family Data Sheet

January 2008 Data Sheet DS1005

## Introduction

The SONET mode of the flexiPCS (Physical Coding Sublayer) block supports compatibility to the SONET standard for STS-12 and STS-48. It also includes options used to implement the TFI-5 standard.

The LatticeSC flexiPCS in SONET mode supports the following operations for STS-12and STS-48 frames:

**Transmit (From LatticeSC device to line) Path:**

- TOH byte generation of A1, A2, and Section B1 bytes
- SONET compatible data scrambling before transmission

**Receive (From line to LatticeSC device) Path:**

- STS-12 or STS-48 SONET framer including frame boundary detection and frame pulse generation.
- SONET compatible data descrambling
- AIS-L insertion
- RDI-L detection
- Section B1 BIP error checking and reporting
- Monitors and interprets the SONET pointers (H1 and H2 bytes)

### LatticeSC flexiPCS Quad Module

Devices in the LatticeSC family have up to 8 quads of embedded flexiPCS logic. Each quad in turn supports 4 independent full-duplex data channels. A single channel can support a SONET data link and each quad can support up to four such channels. Note that mode selection is done on a per quad basis. Therefore, a selection of SONET mode for a quad dedicates all four channels in that quad to SONET mode.

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

Operation of the SERDES requires the user to provide a reference clock (or clocks) to each active quad to provide a synchronization reference for the SERDES PLLs. Reference clocks are shared among all four channels of a quad. If the transmit and receive frequencies are within the specified ppm tolerance for the LatticeSC SERDES (See DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet), only one reference clock for both transmit and receive is needed for both transmit and receive directions. If the receive frequency is not within the specified ppm tolerance for the LatticeSC SERDES (See the DC and Switching Characteristics section of the LatticeSC/M Family Data Sheet) of the transmit frequency, then separate reference clocks for the transmit and receive directions must be provided.

Simultaneous support of different (non-synchronous) high-speed data rates is possible with the use of multiple quads. Multiple frequencies that are exact integer multiples can be supported on a per channel basis through the use of the full-rate/half-rate mode options (see the **SERDES Functionality section** for more details).

## Quad and Channel Option Control

Although the mode selection and reference frequency covers an entire quad, many options covering clocking and data formatting are available on a per channel basis. These options are detailed in the following SONET Mode description. Some of these options can be controlled directly through the FPGA interface pins made available on the SONET Quad Module.

© 2008 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Other options are available only through dedicated registers that can be written and read via the embedded System Bus Interface (see the System Bus section for more details on the operation of the System Bus), or during device configuration. Depending on the specific option, a register may affect operation of multiple quads, a single quad or an individual channel in a quad. Registers dedicated to multiple quads are listed in the Inter-quad Interface Register Map in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet. Registers dedicated to one quad are listed in the Quad Interface Register Map. Registers dedicated to a single channel are listed in the Channel Interface Register Map.

## Register Map Addressing

Figure 10-1 provides a map of base register addresses for any of the flexiPCS quads on a LatticeSC device. Note that different devices may have different numbers of available quads up to a total of 8 quads (or 32 channels) maximum.

Inter-quad Interface, Quad Interface, and Channel Interface Registers are listed by Offset Address. Each Inter-quad Interface Register, Quad Interface Register, and Channel Interface Register has a unique 18-bit address determined by the specific quad or channel targeted. The addressing of Inter-quad Interface Registers, Quad Interface Registers, and Channel Interface Registers is shown Figure 10-1.

Base addresses are dependant on the physical location of a given quad or channel on a LatticeSC device. Each quad and channel has an associated set of control and status registers. These registers are referred throughout this data sheet by their offset addresses. The unique 18-bit address of a control or status register for a specific quad or channel is simply the offset address of that register added to the base address for that specific quad or channel as shown in Figure 10-1.

Channel Interface Registers can be written in one of three methods. One method is to write to one specific register corresponding to one specific channel. The other two methods involve writing to multiple channel registers with just one write operation. This is referred to as a Broadcast write. A Broadcast write is useful when multiple channel registers are to be written with the same value. The first method of Broadcast writing involves writing to all four channel registers in a quad at one time. A second method of Broadcast writing involves writing to all channel registers for all quads on the left or right side of the device at one time. Therefore, a specific Channel Interface Register may be accessed through any one of three channel addresses, depending on the number of channel registers being written to at any one time.

A broadcast write to multiple quads cannot be used to power on SERDES in any device-package combination that does not have all SERDES quads bonded because of excess power on the board and jitter is also affected.

Throughout this document, the byte-wide data at each offset address is listed with a hexadecimal value with bit 0 as the MSB and bit 7 as the LSB as per the PowerPC convention. For example if the value for Quad Interface Register Offset Address 0x02 is stated to be 0x15, the bit pattern defined is as shown in Table 10-1:

***Table 10-1. Control/Status Register Bit Order Example***

| **Quad Interface Register Offset Address 0x02 = 0x15** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Data Bit 0 | Data Bit1 | Data Bit 2 | Data Bit 3 | Data Bit 4 | Data Bit 5 | Data Bit 6 | Data Bit 7 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| **1** | | | | **5** | | | |

A full list of register Offset Addresses is provided in the **LatticeSC flexiPCS Memory Map** section of the flexiPCS Data Sheet.

***Figure 10-1. flexiPCS Inter-quad, Quad, and Channel Interface Register Map Base Addressing***

## Auto-Configuration

Initial register setup for each flexiPCS mode can be performed without accessing the system bus by using the auto-configuration tool, IPexpress. IPexpress generates 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 system bus is therefore not needed if all quads are to be set via auto-configuration files. However, the system bus must be included in a design if the user needs to change control registers or monitor status registers during operation. Note that a quad reset (Quad Interface Register 0x43, bit 7 or the **quad_rst** port at the FPGA interface) will reset all registers back to their default (not auto-configuration) values. If a quad reset is issued, the flexiPCS registers need to be rewritten via the Systembus.

## SONET Register Settings

The auto-configuration file sets registers that perform the following operations. If the auto-configuration file is not used, the appropriate registers need to be programmed via the Systembus. For more options, refer to the flexiPCS register map in the **Memory Map** section of the **LatticeSC/M Family flexiPCS Data Sheet**.

**A flexiPCS quad can be set to SONET mode by writing Quad Interface Register Offset Address 0x18 bits[0:7] to 0x20.**

**By default, a flexiPCS quad in SONET mode is configured for STS-48 SONET (1 STS-48 link per channel). Each channel in the flexiPCS quad can be independently set to STS-12 SONET by writing the appropriate Channel Interface Register Offset Address 0x0B, bit 7 to '1'. Note that both STS-12 and STS-48 modes cannot be supported in the same quad because they require different reference clock rates (there is one reference clock per LatticeSC flexiPCS quad).**

**Each channel in the flexiPCS quad can be independently set to TFI-5 mode by writing the appropriate Channel Interface Register Offset Address 0x0C, bit 3 to '1'.**

This register value automatically sets up the following options in the flexiPCS for the given quad. Details on the functionality of each chosen option is provided in the **SONET Mode Detailed Description** section.

**Transmit Path**

- SONET Transmit Scrambler is enabled

**Receive Path**

- SONET framer enabled (default is STS-48 framer)
- SONET Receive Descrambler is enabled
- AIS-L Insert enabled (default is AIS-L insert during out of frame condition)
- RDI-L checker enabled
- Section B1 calculation and check enabled
- SONET Pointer Interpreter enabled

Other register settings are required to operate a channel or channels in SONET Mode. The following is a list of options that must be set for SONET operation. More detail for each option is included in the **SONET Mode Detailed Description** section.

- Powerup of appropriate quad and channel. Quad powerup is chosen by writing the appropriate Quad Interface Register Offset Address 0x28, bit 1 to '1'. Channel powerup is chosen by writing the appropriate Channel Interface Register Offset Address 0x13, bit 6 (receive direction) and/or bit 7 (transmit direction) to '1'. By default, all channels and quads are powered down.
- Setting SERDES reset low at end of flexiPCS setup. By default, the SERDES reset is set to '1' (SERDES are reset). The SERDES reset can be set low by writing Quad Interface Register Offset Address 0x41, bit 5 to a '0'. For more information on proper flexiPCS powerup and reset sequencing, refer to the **flexiPCS Reset Sequences** and **flexiPCS Power Down Control Signals** descriptions in the LatticeSC/M Family flexiPCS Data Sheet **SERDES Functionality** section.

- Enabling of Transmit Auto A1/A2 insertion, Auto B1 insertion, Auto Section B1 byte generation, Auto Frame Pulse Monitoring, and Auto RDI-L insertion
- Multi-channel alignment within a quad, between quads, or between two LatticeSC devices

## SONET Auto-Configuration

An example of an auto-configuration file for a flexiPCS quad set to STS-12 SONET Mode, with only channel 1 enabled, half-rate mode, is shown in Figure 10-2. Format for the auto-configuration file is

register type (quad=quad register, ch1=channel 1 register), offset address in hex, register value in hex, # comment

***Figure 10-2. STS-12 SONET Auto-Configuration Example File***

```
quad 18 20     #  Set quad to SONET Mode
quad 29 00     #  Set reference clock select to refclk(p/n) inputs
quad 28 40     #  Set bit clock multiplier to 8X (for half rate mode), quad powerup set to '1'
quad 02 15     #  Set ref_pclk, rxa_pclk and rxb_pclk to channel 1 clocks
quad 19 00     #  Set FPGA interface data bus width to 8-bit
ch1 0B 0D      #  Set channel 1 to STS-12 mode, auto b1, auto a1a2 selected
ch1 01 0A      #  Set SERDES to transmit MSB first, receive MSB first
ch1 13 0F      #  Powerup channel 1 transmit and receive directions, set rate mode to “half”
ch1 14 90      # 16% pre-emphasis on SERDES output buffer
ch1 15 10      # +6 dB equalization on SERDES input buffer
quad 41 00     # de-assert serdes_rst
quad 40 ff     # assert datapath reset for all channels
quad 40 00     #de-assert datapath reset for all channels
```

An example of an auto-configuration file for a flexiPCS quad set to STS-48 SONET Mode with all four channels enabled and configured for 4 channel alignment is shown in Figure 10-3.

***Figure 10-3. STS-48 SONET Auto-Configuration Example File***

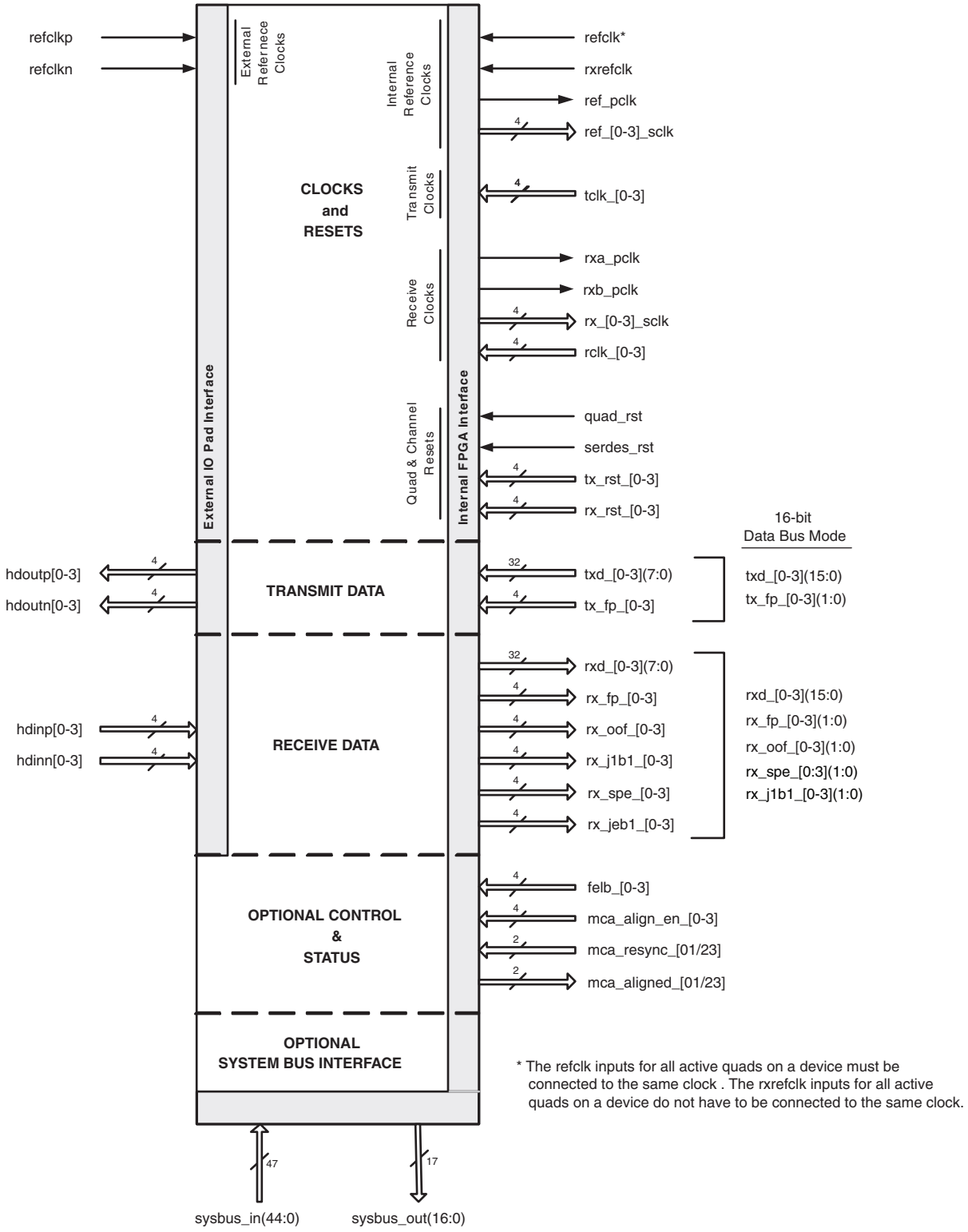
```
quad 18 20     #  Set quad to SONET Mode
quad 29 00     #  Set reference clock select to refclk(p/n) inputs
quad 28 40     #  Set bit clock multiplier to 16X (for full rate mode), quad powerup set to '1'
quad 02 00     #  Set ref_pclk, rxa_pclk and rxb_pclk to channel 0 clock
quad 19 40     #  Set FPGA interface data bus width to 8-bit, enable 4 channel alignment
quad 04 0F     #  Enable alignment on all channels
quad 01 00     #  Set multi-channel alignment clock to channel 0 recovered clock
ch0 0B 0C      #  Set channel 0 to STS-48 mode, auto b1, auto a1a2 selected
ch0 01 0A      #  Set SERDES to transmit MSB first, receive MSB first
```

```
ch0 13 03     #  Powerup channel 0 transmit and receive directions, set rate mode to “full”
ch0 14 90     # 16% pre-emphasis
ch0 15 10     # +6 dB equalization on SERDES input buffer
ch1 0B 0C     #  Set channel 1 to STS-48 mode, auto b1, auto a1a2 selected
ch1 01 0A     #  Set SERDES to transmit MSB first, receive MSB first
ch1 13 03     #  Powerup channel 1 transmit and receive directions, set rate mode to “full”
ch1 14 90     # 16% pre-emphasis
ch1 15 10     # +6 dB equalization on SERDES input buffer
ch2 0B 0C     #  Set channel 2 to STS-48 mode, auto b1, auto a1a2 selected
ch2 01 0A     #  Set SERDES to transmit MSB first, receive MSB first
ch2 13 03     #  Powerup channel 2 transmit and receive directions, set rate mode to “full”
ch2 14 90     # 16% pre-emphasis
ch2 15 10     # +6 dB equalization on SERDES input buffer
ch3 0B 0C     #  Set channel 3 to STS-48 mode, auto b1, auto a1a2 selected
ch3 01 0A     #  Set SERDES to transmit MSB first, receive MSB first
ch3 13 03     #  Powerup channel 3 transmit and receive directions, set rate mode to “full”
ch3 14 90     # 16% pre-emphasis on SERDES output buffer
ch3 15 10     # +6 dB equalization on SERDES input buffer
quad 41 00    # de-assert serdes_rst
quad 40 ff    # assert datapath reset for all channels
quad 41 03    # assert MCA reset
quad 41 00    # de-assert MCA reset
quad 40 00    # de-assert datapath reset for all channels
```

## SONET Mode

IPexpress allows the designer to choose the mode for each PCS quad used in a given design. Figure 10-4 displays the resulting port interface to one PCS quad that has been set to SONET mode.

***Figure 10-4. SONET Mode Pin Diagram***



## SONET Mode Pin Description

Table 10-2 lists all inputs and outputs to/from a PCS quad in SONET mode. A brief description for each port is given in the table. A more detailed description of the function of each port is given in the **SONET Mode Detailed Description**.

***Table 10-2. SONET Mode Pin Description***

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| **Reference Clocks** | | | |
| refclkp | In from I/O pad | N/A | Reference clock input, positive. Dedicated CML input. |
| refclkn | In from I/O pad | N/A | Reference clock input, negative. Dedicated CML input |
| refclk | In from FPGA | N/A | Reference clock input from FPGA logic. Can be used instead of external reference clock.<br><br>The refclk inputs for all active quads on a device must be connected to the same clock. |
| rxrefclk | In from FPGA | N/A | Reference clock input from FPGA logic. Can be used instead of external reference clock. The rxrefclk inputs for all active quads do not have to be connected to the same clock. |
| ref_pclk | Out to FPGA | N/A | Locked reference clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| ref_[0-3]_sclk | Out to FPGA | N/A | Per channel locked reference clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| **Transmit Clocks** | | | |
| tclk_[0-3] | In from FPGA | N/A | Per channel transmit clock inputs from FPGA. Used to clock the TX data phase compensation FIFO with clock synchronous to the transmit reference clock. May also be used to clock the RX data phase compensation FIFO with a clock synchronous to the reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Receive Clocks** | | | |
| rxa_pclk | Out to FPGA | N/A | Recovered receive clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| rxb_pclk | Out to FPGA | N/A | Recovered receive clock selection from one of the four channels. Selection made through register setting. Output to primary clock routing. |
| rx_[0-3]_sclk | Out to FPGA | N/A | Per channel receive clocks. Each channel's locked reference clock is connected to FPGA general routing. Clocks connected to general FPGA routing can route to either primary or secondary clocks with a larger clock injection delay than clock ports dedicated solely to primary clock routing. |
| rclk_[0-3] | In from FPGA | N/A | Per channel receive clock inputs from FPGA. Used to clock the RX data phase compensation FIFO with a clock synchronous to the reference and/or receive reference clock. May not be created from a DLL to PLL combination or a PLL to PLL combination. |
| **Resets** | | | |
| quad_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of SERDES and PCS logic. |
| serdes_rst | In from FPGA | ASYNC | Active high, asynchronous reset for all channels of the SERDES. |
| tx_rst_[0-3] | In from FPGA | ASYNC | Per channel active high, asynchronous reset of individual transmit channel of PCS logic. |
| rx_rst_[0-3] | In from FPGA | ASYNC | Per channel active high, asynchronous reset of individual receive channel of PCS logic. |
| **Transmit Data** | | | |
| hdoutp[0-3] | Out to I/O pad | N/A | High-speed CML serial output, positive. |

*Table 10-2. SONET Mode Pin Description (Continued)*

| Symbol | Direction/ Interface | Clock | Description |
|---|---|---|---|
| hdoutn[0-3] | Out to I/O pad | N/A | High-speed CML serial output, negative. |
| txd_[0-3](7:0)<br><br>txd_[0-3](15:0)* | In from FPGA | tclk_[0-3] | Per channel parallel transmit data bus from the FPGA. Bus is 8-bits wide if QIR 0x19, bit 4 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 4 = '1'. |
| tx_fp_[0-3}<br><br>tx_fp_[0-3](1:0)* | In from FPGA | tclk_[0-3] | Per channel active high signal indicating start of STS-48 or STS-12 SONET frame.<br><br>*2 bits wide if QIR 0x19, bit 4 = '1'. |
| **Receive Data** | | | |
| hdinp[0-3] | In from I/O pad | N/A | High-speed CML serial input, positive. |
| hdinn[0-3] | In from I/O pad | N/A | High-speed CML serial input, negative. |
| rxd_[0-3](7:0)<br><br>rxd_[0-3](15:0)* | Out to FPGA | rclk_[0-3] | Per channel parallel receive data bus to the FPGA. Bus is 8-bits wide if QIR 0x19, bit 5 = '0'.<br><br>*Bus is 16-bits wide if QIR 0x19, bit 5 = '1'. |
| rx_fp_[0-3]<br><br>rx_fp_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high indicating first A1 byte.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_oof_[0-3]<br><br>rx_oof_[0-3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high signaling out-of-frame.<br><br>*2 bits wide if QIR 0x19, bit 5 = '1'. |
| rx_spe_[0-3]<br><br>rx_spe_[0:3](1:0)* | Out to FPGA | rclk_[0-3] | Per channel active high indicating valid SPE data.<br><br>*2 bits wide if QIR 0x19, bit 5='1'. |
| rx_j1b1_[0-3] | Out to FPGA | rclk_[0-3] | Per channel active high indicating J1 byte on rxd data bus or per channel active high indicating B1 BIP error. |
| **Optional Control & Status** | | | |
| felb_[0-3] | In from FPGA | ASYNC | Per channel active high far end loopback enables. |
| mca_align_en_[0-3] | In from FPGA | ASYNC | Per channel active high multi-channel aligner enables. |
| mca_resync_[01/23] | In from FPGA | ASYNC | Active high async. multi-channel aligner resynchronization signal. |
| mca_aligned_[01/23] | Out to FPGA | ASYNC | Active high indicating successful alignment of channels |
| **Optional System Bus Interface** | | | |
| sysbus_in(44:0) | In from FPGA | N/A | Control and data signals from the internal system bus. |
| sysbus_out(16:0) | Out to FPGA | N/A | Control and data signals to the internal system bus. |

# SONET Mode Detailed Description

The following section provides a detailed description of the operation of a PCS quad set to SONET mode. The functional description is organized according to the port listing shown in Figure 10-4. The System Bus Offset Address for any control and status registers relevant to a given function are provided with the description of the operation of that function.

## Clocks & Resets

A detailed description of all SERDES clock, data, and reset ports and recommended reset sequencing is provided in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet. The following is a recommended setup of the SERDES for SONET applications.

For STS-12 SONET applications, the bit clock rate is 622 Mbps. This falls into the range defined as “half rate” mode for the SERDES PLLs. Half rate mode is selected separately for each channel and each direction by writing the appropriate Channel Interface Register Offset Address 0x13, bit 5 (transmit) and bit 4 (receive) to a ‘1’.

The reference clock frequency is determined by the reference clock mode chosen. Table 10-3 shows the possible reference clock frequencies for various reference clock modes which result in a 622 Mbps internal bit rate.

***Table 10-3. STS-12 SONET Reference Clock Frequency Options***

| Half Rate Mode<br><br>Channel Interface Register Offset Address 0x13<br><br>Transmit - Bit 5 = ‘1’<br>Receive - Bit 4 = ‘1’ | | | | |
|---|---|---|---|---|
| Reference Clock Mode<br><br>Quad Interface Register Offset Address = 0x28, Bits [2:3] | Reference Clock Frequency | Internal Serial (bit) Clock Frequency | FPGA Interface Clock Frequency<br><br>Quad Interface Register Offset Address 0x19 | |
| | | | 8 Bit Data Bus Width<br><br>Transmit - Bit 4 = ‘0’<br>Receive - Bit 5 = ‘0’ | 16 Bit Data Bus Width<br><br>Transmit - Bit 4 = ‘1’<br>Receive - Bit 5 = ‘1’ |
| 00 | 77.75 MHz | 622 MHz | 77.75 MHz | 38.875 MHz |
| 01 | 155.5 MHz | 622 MHz | 77.75 MHz | 38.875 MHz |
| 10 | 311 MHz | 622 MHz | 77.75 MHz | 38.875 MHz |

For STS-48 SONET applications, the bit clock rate is 2.488 Gbps. This falls into the range defined as “full rate” mode for the SERDES PLLs. Full rate mode is selected separately for each channel and each direction by writing the appropriate Channel Interface Register Offset Address 0x13, bit 5 (transmit) and bit 4 (receive) to a ‘0’ (default).

The reference clock frequency is determined by the reference clock mode chosen. Table 10-4 shows the possible reference clock frequencies for various reference clock modes which result in a 2.488 Gbps internal bit rate.

***Table 10-4. STS-48 SONET Reference Clock Frequency Options***

| Full Rate Mode<br><br>Channel Interface Register Offset Address 0x13<br><br>Transmit - Bit 5 = ‘0’<br>Receive - Bit 4 = ‘0’ | | | | |
|---|---|---|---|---|
| Reference Clock Mode<br><br>Quad Interface Register Offset Address = 0x28, Bits [2:3] | Reference Clock Frequency | Internal Serial (bit) Clock Frequency | FPGA Interface Clock Frequency<br><br>Quad Interface Register Offset Address 0x19 | |
| | | | 8 Bit Data Bus Width<br><br>Transmit - Bit 4 = ‘0’<br>Receive - Bit 5 = ‘0’ | 16 Bit Data Bus Width<br><br>Transmit - Bit 4 = ‘1’<br>Receive - Bit 5 = ‘1’ |
| 00 | 155.5 MHz | 2.488 GHz | 311 MHz | 155.5 MHz |
| 01 | 311 MHz | 2.488GHz | 311 MHz | 155.5 MHz |
| 10 | 622 MHz | 2.488 GHz | 311 MHz | 155.5 MHz |

*Note: There are also frequency dependent SERDES performance control bits that are not writable via the System-bus. These control bits are set in the auto-configuration file generated within IPexpress and passed into the bit-*

*stream through the normal FPGA design flow (map, par, bitgen). To change the bit rate for a SERDES, specify the proper values for the affected flexiPCS quad in IPexpress and regenerate the auto-configuration file. Failure to do this may result in non-optimal SERDES operation.*

Additional information on all reference clock rate modes can be found in the **SERDES Functionality** section of the Lattice flexiPCS Data Sheet.

**Quad & Channel Resets**
Resets are provided to reset an entire quad (either SERDES logic only or all flexiPCS logic) or each individual channel (all flexiPCS logic per channel). These resets are driven from the FPGA logic. A summary of the control signals provided for reset are listed below. More detail on recommended initialization and reset sequences for the SERDES is provided in the **SERDES Functionality** section of the LatticeSC/M Family flexiPCS Data Sheet.

The following inputs to the PCS from the FPGA are enabled as long as Quad Interface Register Offset Address 0x42, bit 7 is set to '1' (which is the default on powerup). Writing a '0' to this bit will disable these reset inputs.

**quad_rst** - Active high, asynchronous signal from FPGA resets all SERDES and PCS logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x43, bit 7 to '1'. quad_rst also resets all flexiPCS control registers. If these registers had been previously written via the Systembus or set during configuration through an auto-configuration file, they will need to be rewritten following this reset.

**serdes_rst** - Active high, asynchronous signal from FPGA resets all SERDES (but not PCS) logic in the quad. This reset can also be performed by writing Quad Interface Register Offset Address 0x41, bit 5 to '1'.Note that this bit is preset to '1' on powerup and must be written to '0' before operation of the SERDES can commence.

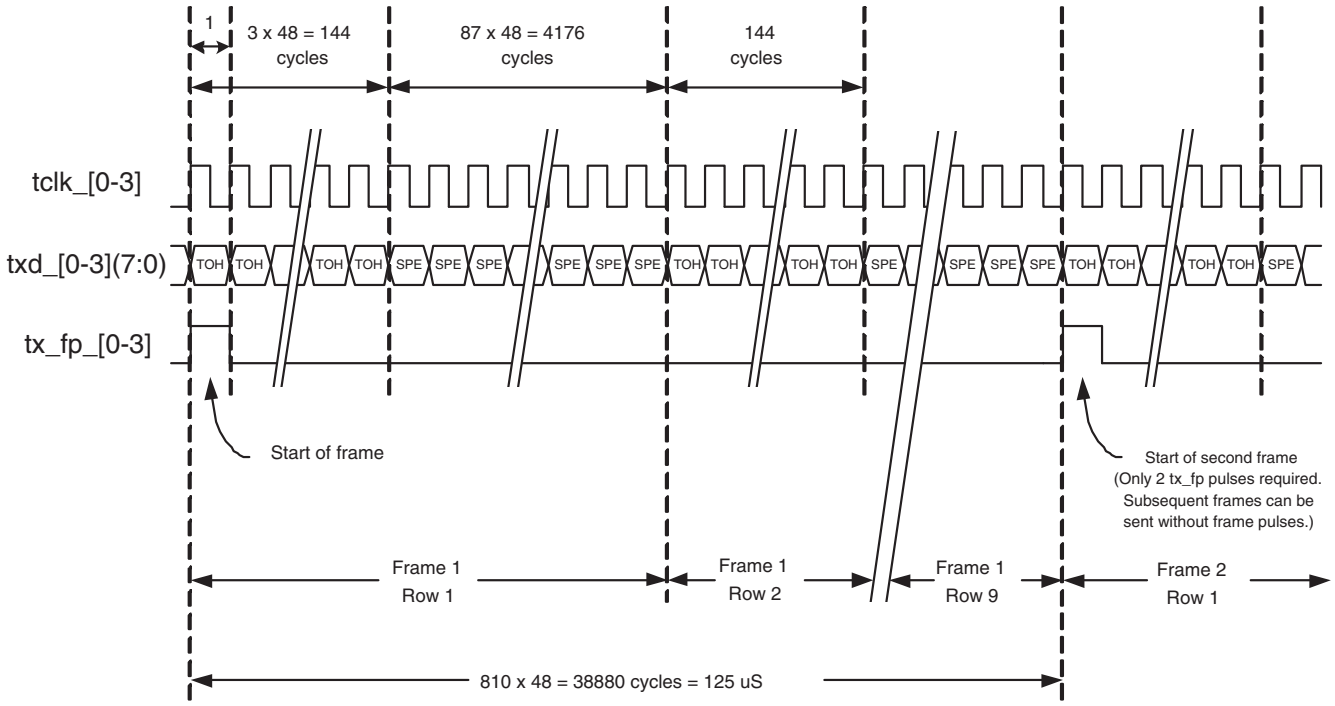
**tx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one transmit channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [4:7]. Bit 7 resets transmit channel 0, bit 6 resets transmit channel 1, bit 5 resets transmit channel 2, and bit 4 resets transmit channel 3.

**rx_rst_[0-3]** - Active high, asynchronous signals from FPGA reset one receive channel of PCS logic each. This reset can also be performed by writing to Quad Interface Register Offset Address 0x40, bits [0:3]. Bit 3 resets receive channel 0, bit 2 resets receive channel 1, bit 1 resets receive channel 2, and bit 0 resets receive channel 3.

## Transmit Data

When configured into SONET mode, a PCS quad supports up to four STS-48 or STS-12 transmit data paths. Figure 10-5 shows the transmit signals in a STS-48 application.

***Figure 10-5. SONET Mode STS-48/STS48c Transmit Timing Diagram***

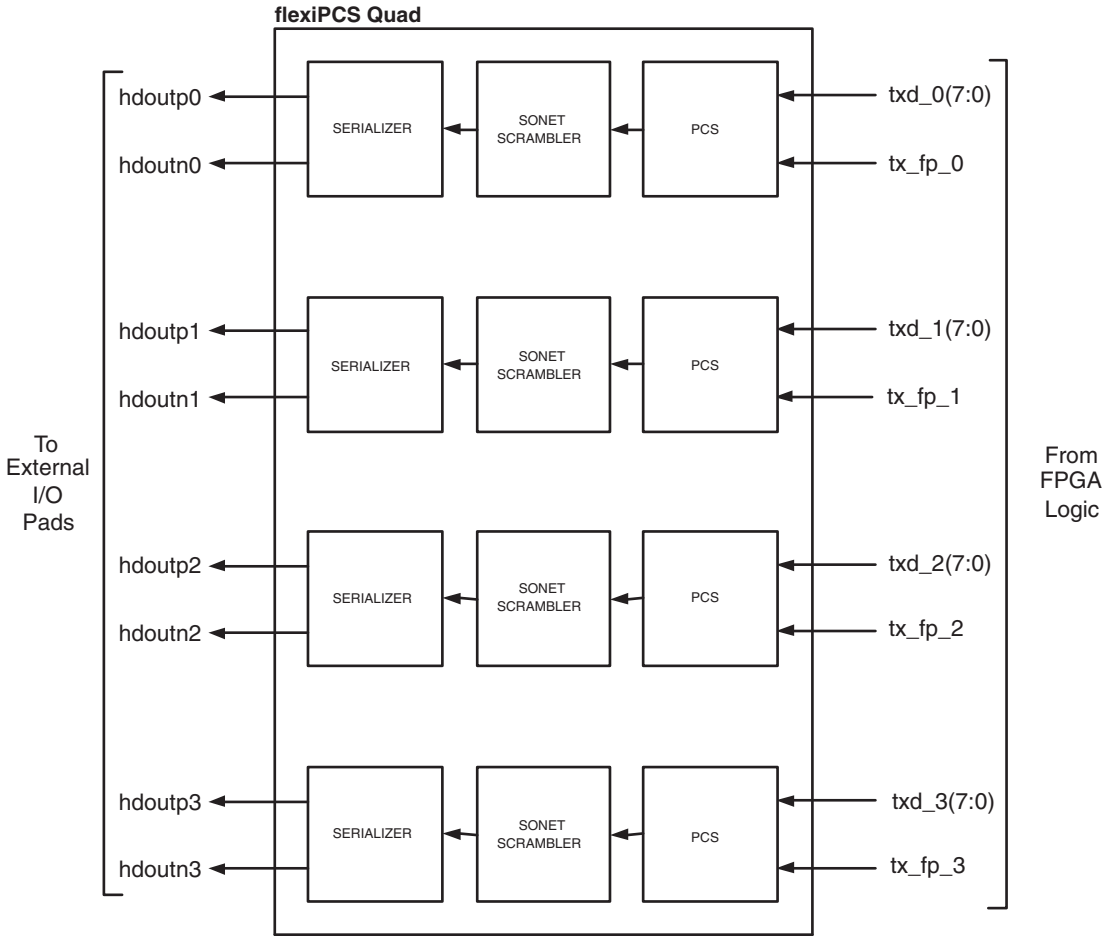



**txd_[0-3](7:0)** - Per channel transmit data from the FPGA interface. Each quad supports up to 4 independent channels of 8-bit wide parallel data. Each txd_[0-3] is an eight bit data bus that is driven synchronously with respect to the corresponding tclk_[0-3].

**tx_fp_[0-3]** - Per channel, active high signal indicating the beginning of a STS-48 or STS-12 SONET frame. It is to be asserted synchronously for one clock cycle concurrent with the corresponding channel's first TOH A1 byte (default) or A2 byte of a SONET frame. The frame pulse can be set to align with the first A2 byte by writing to the appropriate Channel Interface Register Offset Address 0x0C, bit 4 to '1'.

The quad PCS in SONET mode transmit data path consists of the following sub-blocks per channel: PCS, SONET Scrambler, and Serializer. Figure 10-6 shows the four channels of transmit data paths in a PCS quad in SONET mode.

*Figure 10-6. SONET Transmit Path (Single Quad)*




**The flexiPCS automatically** performs transmit frame pulse monitoring when configured in SONET mode. A clock-wide frame pulse (tx_fp) must be provided during the first A1 byte for a minimum of 2 frames after a PCS reset. SONET row, column, and frame counters are synchronized at the second frame pulse. Subsequent frames do not require a frame pulse. If subsequent frame pulses are present, they reset the internal row, column and STS counters to synchronize to the user-provided frame pulse.

The flexiPCS performs the following operations on the transmitted data for that channel:

**A1/A2 Insertion**: The flexiPCS inserts the A1 (0xF6) and A2 (0x28) framing bytes in the Section Overhead for all transmitted SONET frames when Register Offset Address 0x0B, bit 5, is set to '1'.

A1/A2 framing bytes can be delayed for 2 clock cycles by writing the appropriate Channel Interface Register Offset Address 0x0B, bit 3 to ‘1’.

In the TFI-5 mode (Selected by writing appropriate Channel Interface Register Offset Address 0x0C, bit 2 to ‘1’), the A1 bytes are only inserted in the STS-1#10, STS-1#11 and STS-1#12 (STS-12 Mode) or the STS-1#46, STS-1#47 and STS-1#48 (STS-48 Mode) positions and the A2 bytes inserted only in the STS-1#1, STS-1#2 and STS-1#3 positions (both STS-12 and STS-48 Modes).

**A1/A2 Corruption:** A preset number of A1/A2 framing bytes can be intentionally corrupted by writing the appropriate Channel Interface Register Offset Address 0x11, bit 5 to ‘1’. A1/A2 corruption is accomplished by inverting the A1 and A2 bytes (A1 set to 0x09 and A2 set to 0xD7). If A1/A2 corruption is selected, the number of A1/A2 byte errors to be transmitted is set by writing the appropriate Channel Interface Register Offset Address 0x10, bits [2:7]

to the value corresponding to the desired number of A1/A2 errors to be transmitted. A1/A2 framing bytes can be persistently corrupted by writing the appropriate Channel Interface Register Offset Address 0x10, bit 1 to '1'.

**Section B1 Byte Generation:** B1 byte is the Bit Interleaved Parity (BIP-8) using even parity calculated over all bits of the previous SONET frame after scrambling. Calculated B1 is inserted in the first STS-1 position of a STS-12 or STS-48 frame. All other B1 bytes are set to 0x00.

**Section B1 Byte Corruption:** A preset number of B1 bytes can be intentionally corrupted (by inverting the B1 bytes) by writing the appropriate Channel Interface Register Offset Address 0x11, bit 6 to '1'. If B1 corruption is selected, the number of B1 byte errors to be transmitted is set by writing the appropriate Channel Interface Register Offset Address 0x0F, bits [2:7] to the value corresponding to the desired number of B1 errors to be transmitted. B1 framing bytes can be persistently corrupted (by inverting the B1 bytes) by writing the appropriate Channel Interface Register Offset Address 0x0F, bit 1 to '1'.

**Scrambler**

The scrambler polynomial used is the GR-253 standard polynomial $x^7 + x^6 + 1$. All bytes except the first row of section overhead (A1, A2 and J0) will be scrambled. The 7-bit scrambler sequence is reset to "1111_111"on the first byte following the last TOH byte in the first row (after J0). The scrambler function can be disabled on a per channel basis by writing the Channel Interface Register Offset Address 0xOC, bit 1 to "1" (Bit 1 = '0' is scrambler enable which is the default when in SONET mode). Note that the transmit scrambler can be enabled or disabled for each channel independently.

**Serializer**

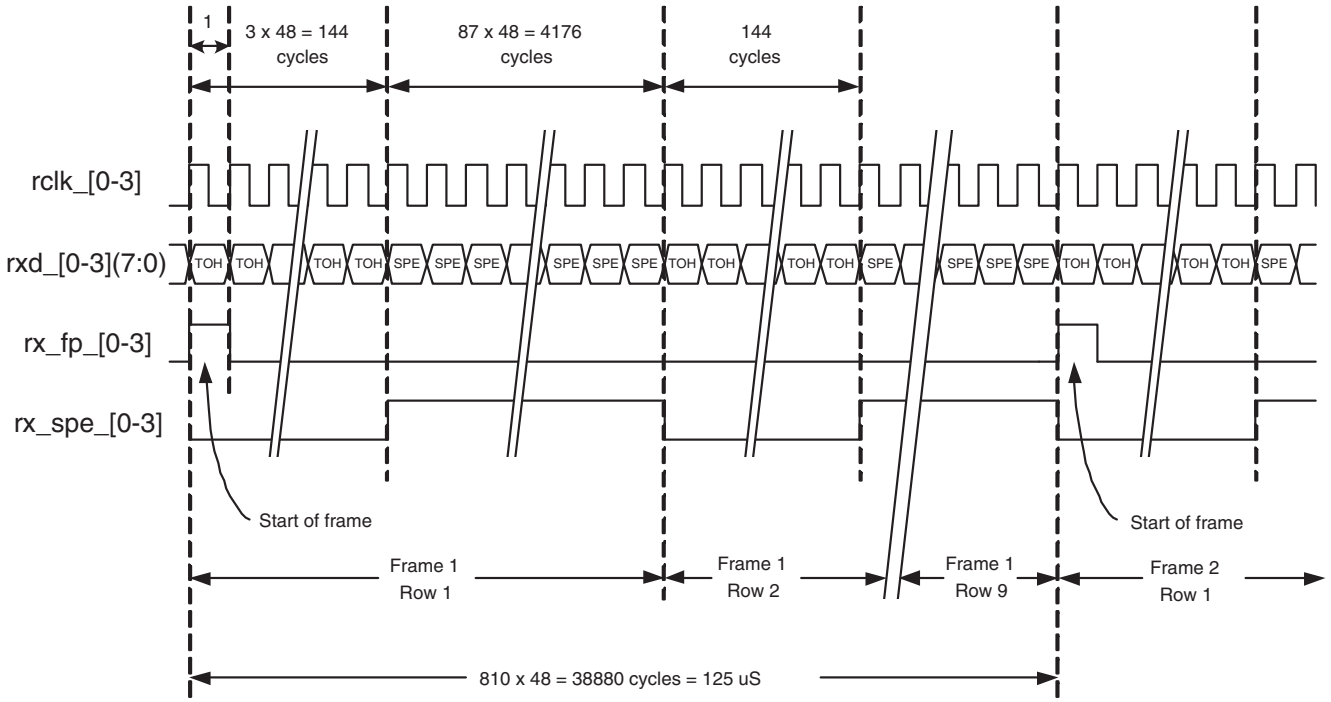
The data undergoes parallel to serial conversion and is transmitted off chip via the embedded SERDES. Note that by default, the SERDES transmits the LSb first. For SONET compliance, the MSb must be transmitted first. This can be set on a per channel basis by writing the appropriate Channel Interface Register Offset Address 0x01, bit 4 to '1'.

For detailed information on the operation of the LatticeSC PCS SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

## Receive Data

When configured into SONET mode, a PCS quad supports up to four STS-48 or STS-12 receive data paths. Figure 10-7 shows the receive signals in a STS-48 application.

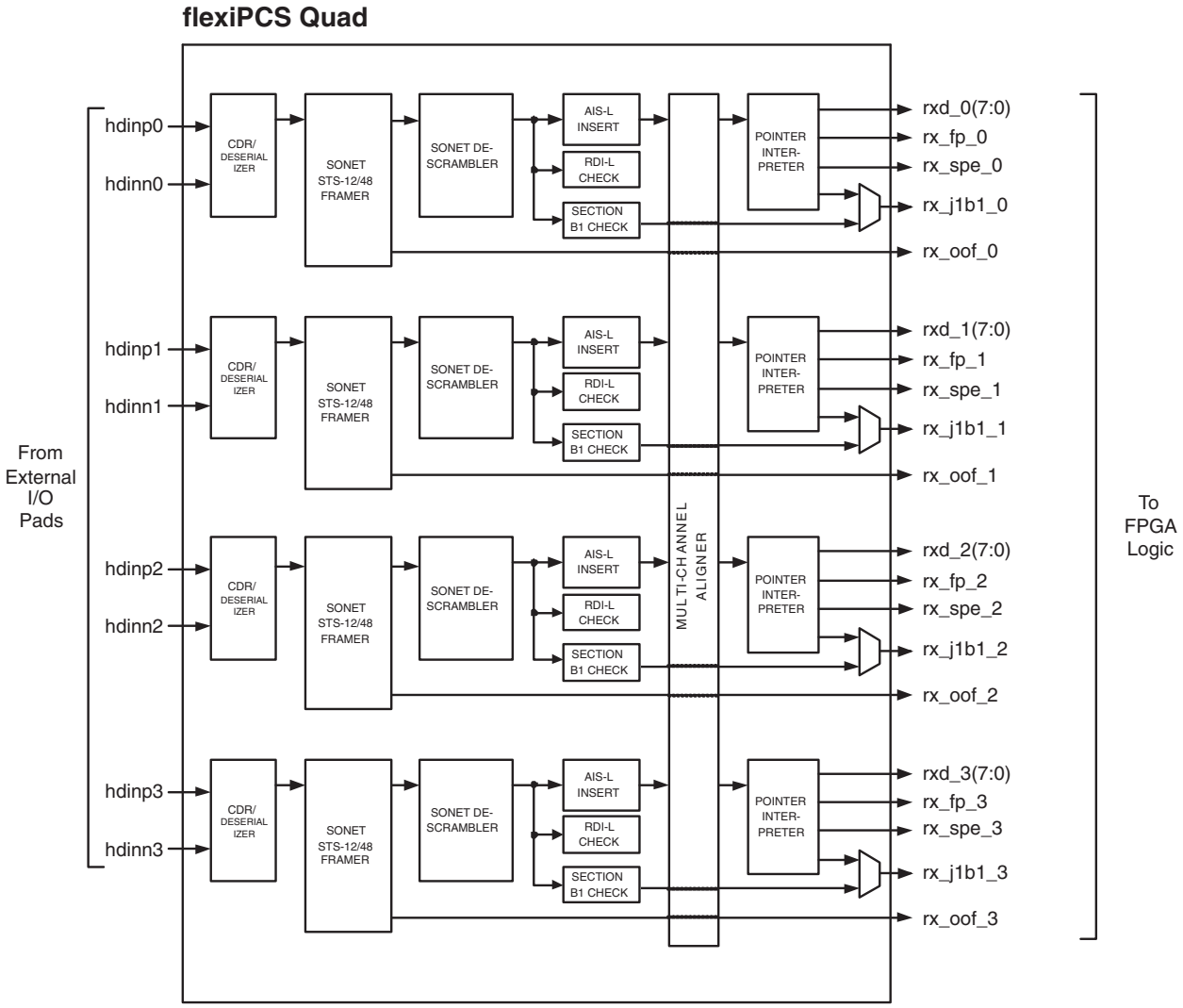
***Figure 10-7. SONET Mode STS-48/STS48c Receive Timing Diagram***




**rxd_[0-3](7:0)** - Per channel receive data to the FPGA interface. Each quad supports up to 4 independent channels of 8 bit wide parallel data. rxd_[0-3] transitions synchronously with respect to rclk_[0-3].

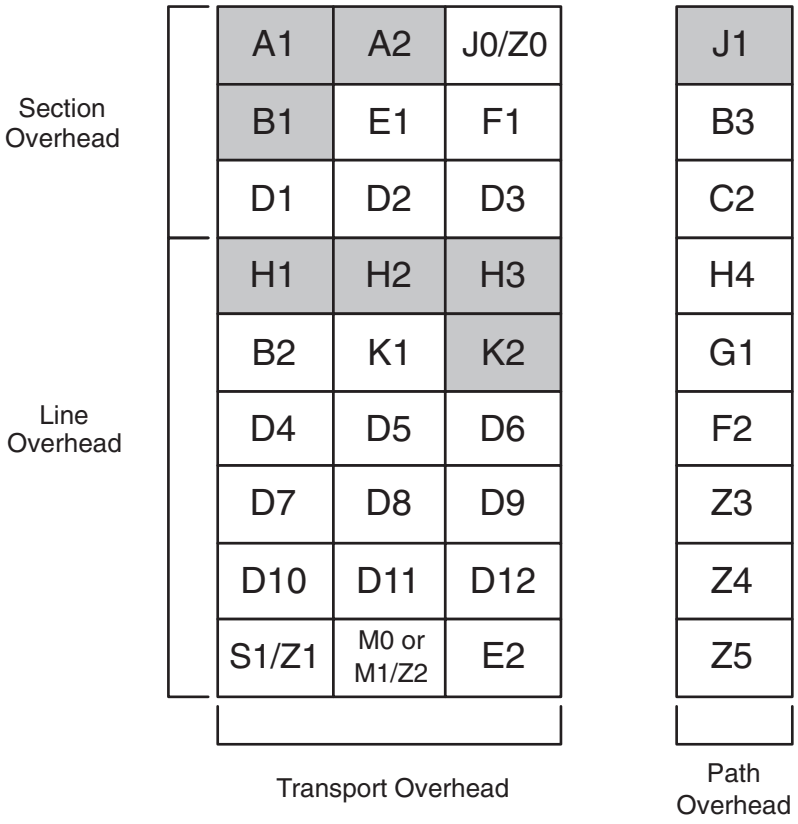
The quad PCS in SONET mode receive data path consists of the following sub-blocks per channel: Deserializer, SONET STS-12/48 Framer, SONET Descrambler, AIS-L Insert, and Pointer Interpreter. In addition, RDI-L Check and Section B1 Check is performed on the descrambled data. An optional multi-channel aligner block allows alignment of multiple SONET channels if desired. Figure 10-8 shows the four channels of transmit data paths in a PCS quad in SONET mode.

*Figure 10-8. SONET Receive Data Path (Single Quad)*




The Receive Path of the flexiPCS in SONET Mode monitors the Transport and Path Overhead bytes shown in Figure 10-9:

*Figure 10-9. SONET Transport and Path Overhead Bytes Monitored in flexiPCS Receive Path*




**Deserializer**
Data is brought on chip to the embedded SERDES where it undergoes serial to parallel.

Note that by default, the SERDES interprets the first bit in the incoming serial stream as the LSb. In the SONET specification, the first receive bit in the serial stream is assumed to be the MSb. This can be set on a per channel basis by writing the appropriate Channel Interface Register Offset Address 0x01, bit 6 to '1'.

For detailed information on the operation of the LatticeSC PCS SERDES, refer to the SERDES Functionality section of the LatticeSC/M Family flexiPCS Data Sheet.

**SONET STS-12/48 Framer**
The SONET Framer is responsible for detecting the in-frame and out-of-frame status of the incoming data and sends out alarms on OOF (Out Of Frame).

The framer supports both STS-12/STS-12c and STS-48/STS-48c frames.

**rx_fp_[0-3]** - Per channel, active high frame pulse indicating the first A1 byte of each SONET frame. SONET frame pattern detection is done in two stages:

1. Byte alignment is performed by searching for the A1 byte (0xF6). Once an A1 byte is found in the data stream, the 8-bit parallel data is shifted such that all subsequent words are byte aligned.

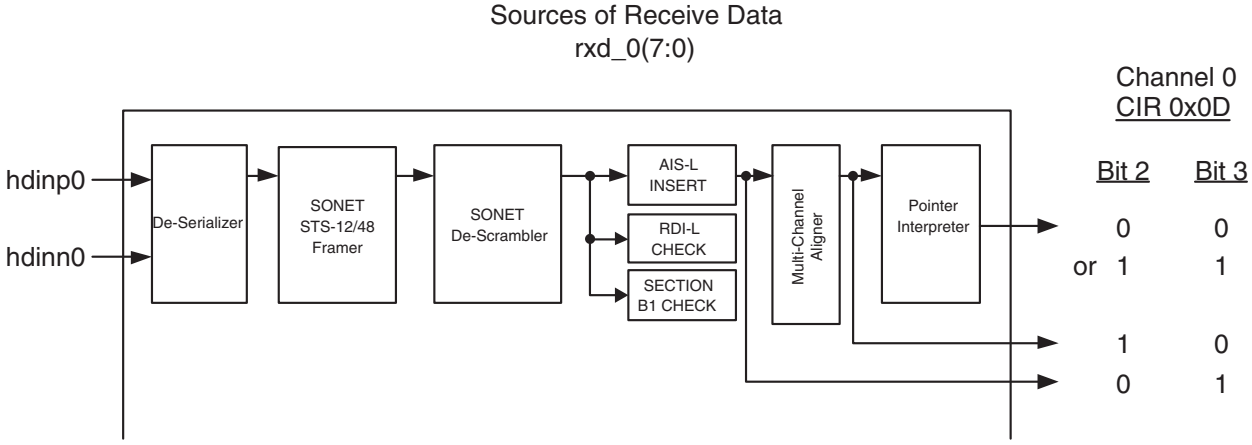
2. The SONET framing state machine locates a A1A1A1A2A2A2 transition (F6F6F6282828). When this transition is found, the row, column and STS counters are set and the frame pulse generator is synchronized to the frame. The occurrences of the A1A1A1A2A2A2 bytes happen at the appropriate location as shown below:

A1A1A1 = row 1, column 1, STS#46, STS#47, STS#48 (for a STS-48/STS-48c frame)

A1A1A1 = row 1, column 1, STS#10, STS#11, STS#12 (for a STS-12/STS-12c frame)

A2A2A2 = row 1, column 2, STS#1, STS#2, STS#3 (for STS-48/STS-48c or STS-12/STS-12c)

***Figure 10-10. Frame Pulse Generation on A1 or A2***




**rx_oof_[0-3]** - Per channel, active high signal indicating an out-of-frame condition. This signal is generated by the SONET receive framer. The in-frame/out-of-frame determination is defined in the SONET framer description below. The operation of rx_oof_[0-3] is as follows:

- The SONET framer comes out of reset in the out-of-frame state with rx_oof_[0-3] high.
- The framer goes in-frame if it finds two consecutive frames with the desired framing bytes. This corresponds to SONET GR-253 specification that it should take two consecutive valid framing patterns to frame to an incoming signal. This also satisfies the OIF standard.
- The framer goes out of frame if it finds four consecutive frames with at least one framing bit error in each frame. as specified in GR-253. In TFI-5 mode, (Selected by writing appropriate Channel Interface Register Offset Address 0x0C, bit 2 to '1'), the number of consecutive frames with a framing bit error can be set by writing the value to the appropriate Channel Interface Register Offset Address 0x0C, bits [5:7].
- The framer also has a fast frame mode where a single good framing pattern can cause the framer state machine to go the "in frame" state and a single bad frame can cause the state machine to declare "OOF". The fast frame mode can be set by writing the appropriate Channel Interface Register Offset Address 0x0D, bit 7 to '1'.

**SONET Descrambler**

Data from the framer block is descrambled using the GR-253 standard polynomial $x^7 + x^6 + 1$. The descrambling is done on each incoming byte except the overhead bytes A1, A2 (framing bytes) and J0. The descrambling can be disabled by writing the appropriate Channel Interface Register Offset Address 0x0C, bit 1 to '1'.

**AIS-L Insert**

AIS-L insertion implies that all bytes except the Section Overhead bytes in a SONET frame will be set to all 1s. This is compatible to GR-253. There are two conditions under which AIS-L insertion can occur, both of which are software controlled:

- During an out-of-frame condition. This can be selected by writing the appropriate Channel Interface Register Offset Address 0x0D, bit 5 to '1'.
- Force AIS-L on all SONET frames regardless of out-of-frame condition. This can be selected by writing the appropriate Channel Interface Register Offset Address 0x0D, bit 6 to '1'.

**RDI-L Check**
The RDI-L (Remote Defect Indicator-Line) is monitored through bits 2-0 of the K2 byte (value should be "110" to indicate an RDI-L status). This is checked in the first STS-1 only. RDI-L must be detected in two consecutive frames before the RDI-L software alarm register bit is set. In fast-frame mode, RDI-L software alarm bit will be set if detected in one frame.

**Section B1 Check**
The section BIP-8 B1 byte in the first STS-1 position of a SONET STS-N/STS-Nc frame contains the scrambled BIP value for all the scrambled bytes of the previous frame. Except for the A1, A2 and J0 bytes, all bytes in a SONET frame are scrambled. The Section B1 byte is calculated as the even parity of all bits (from the framer) in the current STS-48/STS-48c frame or STS-12/STS-12c frame. This value is compared to the descrambled Section B1 byte of the next frame.

**rx_j1b1_[0-3]** - Per channel, active high signal indicating that a B1 BIP error has been detected; valid only when Channel Interface Register Offset Address 0x0D bits [2:3] are set to 10 or 01.

The result of the B1 byte comparison is stored in an alarm register (Channel Interface Register Offset Address 0x83). A '1' on a particular bit indicates an error in the corresponding bit position within the B1 byte.

An 8-bit B1 byte error counter (one count per errored bit) is also provided as a status register (Channel Interface Register Offset Address 0x8C). Upon saturation, this counter will hold at its maximum value until cleared on a read.

**Pointer Interpreter**
The pointer interpreter receives the incoming SONET or SDH STS-48/STS-48c, STS-12/STS-12c frame and interprets the pointer bytes H1 and H2.

The pointer interpreter does the following:

- Monitors the H1, H2 bytes for each incoming STS-1.
- Generates NDF when a valid pointer is received three consecutive times after the AIS state. Note that the consecutive pointer offsets need not be the same value. The offset is then adjusted to the third received valid pointer. For example:
  - STS-1#1 for frame 1 has an AIS pointer
  - STS-1#1 for frame 2 has an offset of 10
  - STS-1#1 for frame 3 has an offset of 10
  - STS-1#1 for frame 4 has an offset of 20
  - The J1 marker for STS-1#1 will be reset to the offset of 20 for Frame 4.
- Provides the J1 marker for every STS-1 for a STS-48 frame or the first STS-1 of a concatenated (STS-48c/STS-12c) frame.
- If an increment or decrement operation is detected, the pointer offset is adjusted and the J1 byte marker is adjusted to reflect the new offset (offset is incremented by 1 for increment and decremented by 1 for decrement operation). For example:
  - Start with pointer offset for STS-1#1 in Frame 1 = 89:
  - In frame 2, a positive frequency justification is detected for STS-1#1 by inversion of the five bits.
  - In frame 3, the pointer offset is incremented to 90 and the J1 byte marker will be reset to location 90.

Any invalid pointer or a value of 0xFF in the pointer fields H1 and H2 will result in AIS-P. During AIS-P, the pointer bytes H1, H2, and H3 and the entire SPE will be set to 1s. If the incoming STS-1 is part of a concatenated group, only the head STS-1 will be set to FF during AIS-P.

**rx_spe_[0-3]** - Per channel, active high signal indicating valid SPE data available on corresponding channel's rxd receive data bus; valid only when Channel Interface Register Offset Address 0x0D bits [2:3] are set to 00 or 11.

**rx_j1b1_[0-3]** - Per channel, active high signal indicating J1 byte present on the corresponding channel rxd receive data bus; valid only when Channel Interface Register Offset Address 0x0D bits [2:3] are set to 00 or 11.

The SPE and J1 indicator provides the SPE and J1 indicator that delineates payload and transport overhead in an STS-48/STS-48c or STS12/STS12c frame.

- During positive pointer justification, the SPE indicator will be low during H3 byte and the byte following it. The J1 indicator will be incremented by 1 for the following frame.
- During negative pointer justification, the SPE indicator will be high during H3 byte. The J1 indicator will be decremented by 1 for the following frame.
- During no justification, the SPE indicator will be low during H3 byte. The J1 indicator will continue to indicate the start of the payload pointed by the offset value in the H2 byte.
- During concatenation, the J1 indicator for a concatenated group will follow the offset for the head of that group.
- During AIS-P state, the J1 indicator should be ignored by the user.

The pointer interpreter will handle any length of concatenation within an STS-12/STS-12c or STS-48/STS-48c frame. The incoming STS-12/STS-12c or STS-48/STS-48c frames have to follow the byte ordering scheme specified in Section 6 of the GR-253 spec. This requires up to 12 different J1 indicators in STS12/STS12c, or up to 48 different indicators in STS48/STS48C mode to be tracked.

**Multi-Channel Alignment**
Two channel alignment of channels 0 and 1 in a flexiPCS quad in SONET Mode is selected by setting Quad Interface Register Offset Address 0x19, bit 3 to '1'. Two channel alignment of channels 2 and 3 in a flexiPCS quad in SONET Mode is selected by setting Quad Interface Register Offset Address 0x19, bit 2 to '1'. Four channel alignment of channels 0, 1, 2 and 3 in a flexiPCS quad in SONET Mode is selected by setting Quad Interface Register Offset Address 0x19, bit 1 to '1'.

**Multi-channel Aligner**
The Multi-channel alignment module performs the operations and functions to control the multi-channel alignment process.

The flexiPCS Multi-channel Aligner allows for alignment of two, three, or four channels in a quad. In any of the SONET multi-channel alignment modes, channels are aligned by the frame pulse extracted from the incoming receive data stream.

The following is a procedure for settings registers for Multi-channel Alignment.

1. Set the mode for the Multi-channel Aligner and enable/disable the appropriate channels for alignment.
2. Each channel to be aligned must also be enabled for alignment by writing to the appropriate bit in Quad Interface Register Offset Address 0x04. Write a '1' to bit 7 to include channel 0 for alignment. Write a '1' to bit 6 to include channel 1 for alignment. Write a '1' to bit 5 to include channel 2 for alignment. Write a '1' to bit 4 to include channel 3 for alignment. Multi-channel alignment can also be enabled by setting the **mca_align_en** pin at the FPGA interface high.
3. Set the Multi-channel Alignment output clocks. A clock domain transfer occurs between the inputs and outputs of the Multi-channel Aligner. Each channel's receive data is clocked into the Multi-channel Aligner with its own separate recovered receive clock. Each channel's receive data is clocked out of the Multi-channel Aligner on a unique multi-channel receive clock. Each multi-channel receive clock can be programmed to be any of the recovered receive clocks.

It is expected that the user will assign the same recovered receive clock to all the multi-channel output clocks for every channel that is to be aligned. That way, all aligned channels coming out of the Multi-channel Aligner are effectively clocked by the same clock. For more information on setting up a multi-channel receive clock, refer to the **Multi-Channel Alignment** section of the LatticeSC/M Family flexiPCS Data Sheet.

For example, in a 4 channel alignment application, in order to set up all Multi-channel Alignment clocks to be sourced from the channel 0 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0x00. To

set all Multi-channel Alignment clocks to be sourced from the channel 1 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0x55. To set all Multi-channel Alignment clocks to be sourced from the channel 2 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xAA. To set all Multi-channel Alignment clocks to be sourced from the channel 3 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xFF.

Figure 10-11 shows all clock domains for both transmit and receive directions for a single channel inside the flexiPCS.

4. Set the Multi-channel Aligner FIFO latency and high-water mark values. Latency values from 0 to 31 can be set by writing to Quad Interface Register Offset Address 0x05, bits [3:7]. Setting a latency value of 0x1F will minimize chance of FIFO overrun or underrun. The FIFO high-water mark values from 0 to 63 can be set by writing to Quad Interface Register Offset Address 0x06, bits [2:7]. The FIFO high-water mark should be set to the maximum the number of bytes of skew allowed for de-skewing. Larger values of FIFO high-water mark increase the chance of FIFO overrun or underrun. For more information on choosing latency and high-water mark values, refer to the Multi-channel Alignment section of the LatticeSC/M Family flexiPCS Data Sheet.

5. Reset the Multi-channel Aligner state machine and FIFO control logic if desired with the **mca_resync** pin at the FPGA interface. **mca_resync** is an active high asynchronous reset which resets the Multi-channel Aligner for all four channels.

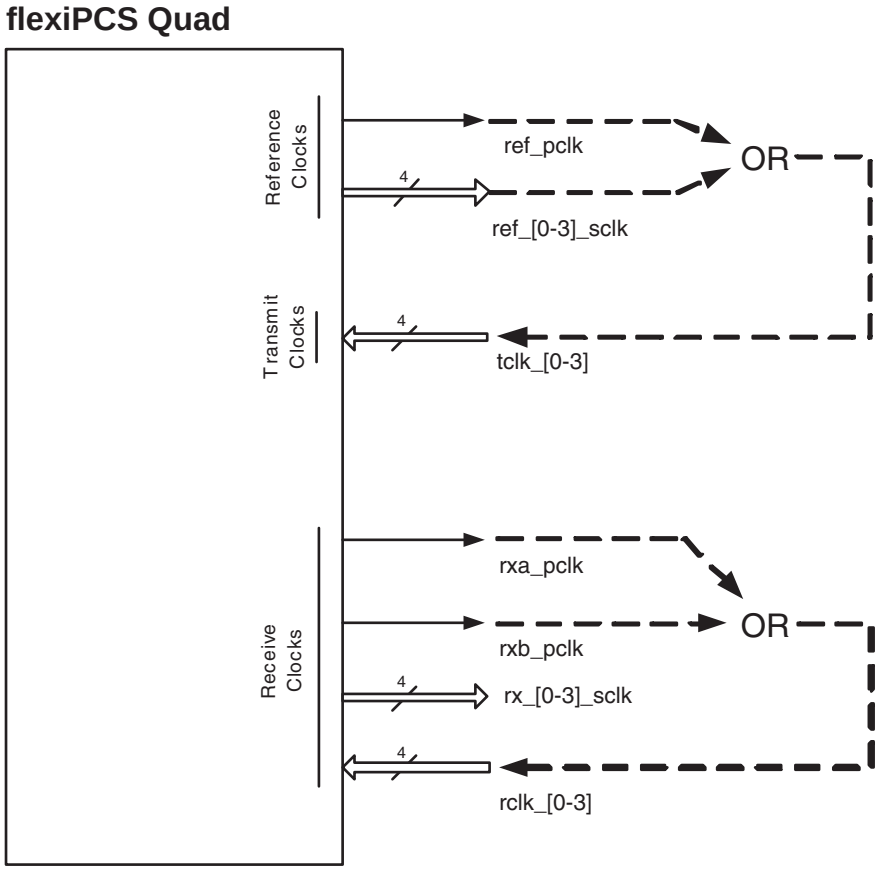
Channels can also be aligned between two different quads on the same chip of between quads on two chips. More information about Multi-channel Alignment is available in the **Multi-Channel Alignment** section of the flexiPCS Data Sheet.

Figure 10-11 shows the clock domains for both transmit and receive directions for a single channel inside the flexiPCS.

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

On the receive side, a clock domain transfer from the recovered channel clock to the Multi-channel Alignment channel clock occurs at the Multi-channel Aligner. A second clock domain transfer from the Multi-channel Alignment channel clock to the locked reference clock occurs at the Clock Tolerance Compensator. A third clock domain transfer occurs from the locked reference clock to the rclk input at the FPGA interface at the FPGA Receive Interface phase compensation FIFO. The FPGA Receive Interface phase compensation FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. These phase compensation FIFOs (one per channel) cannot compensate for frequency variations.

*Figure 10-11. flexiPCS Clock Domain Transfers for SONET Operation*




**Bypass Modes**

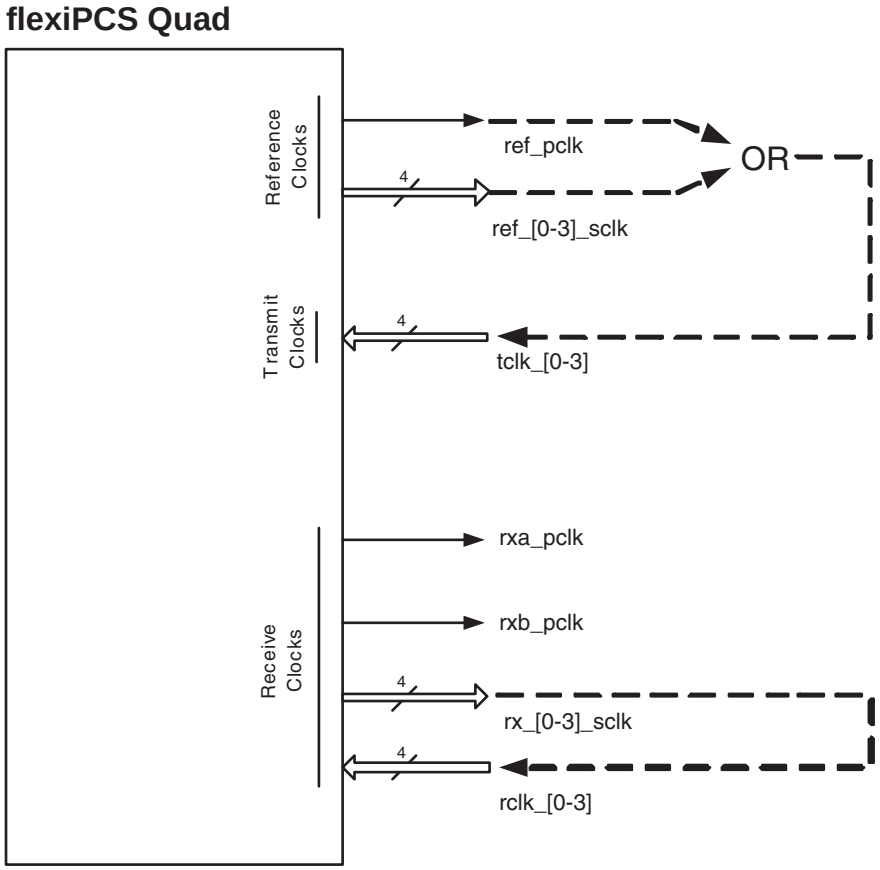
The above receive data path description is the default path for SONET mode. However several bypass modes for receive rxd data are available through a control register. Bypass modes are controlled on a per channel basis by writing to the appropriate Channel Interface Register Offset Address 0x0D, bits [2:3] as shown in Table 10-5.

*Table 10-5. Bypass Mode Controls*

| Channel Interface Register Offset Address = 0xOD | | Reference Clock Selection |
|---|---|---|
| **Bit 2** | **Bit 3** | **Description** |
| 0 | 0 | RXD data comes from the Pointer Interpreter (default) |
| 0 | 1 | RXD data comes from AIS-L block output before the Aligner |
| 1 | 0 | RXD data comes from Multi-channel Aligner |
| 1 | 1 | RXD data comes from Pointer Interpreter |

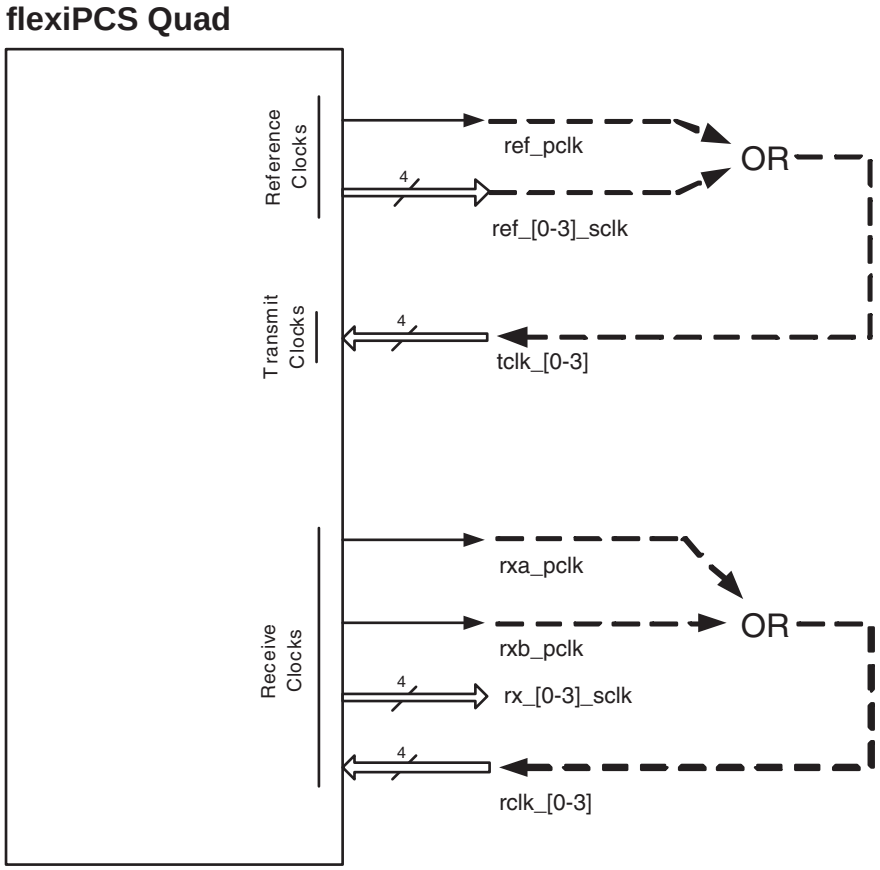
The bypass paths for receive data in the flexiPCS is shown in Figure 10-12:

*Figure 10-12. Bypass Modes for flexiPCS Receive Path in SONET Mode*



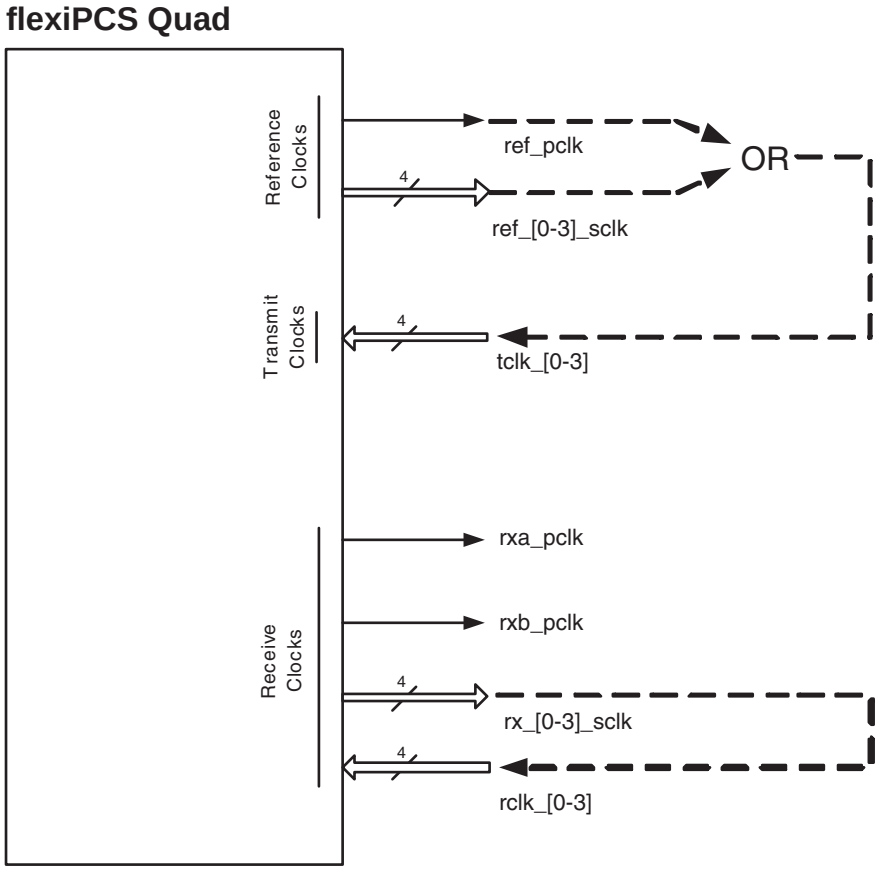

To guarantee a synchronous interface, both the input transmit clocks and input receive clock should be driven from one of the output reference clocks for a flexiPCS quad set to SONET mode. The user can Figure 10-13 illustrates the possible connections that would result in a synchronous interface for bypass modes CIR 0x0D, bits [2:3] “00” and “11”.

*Figure 10-13. Synchronous input clocks to flexiPCS quad for bypass modes "00" and "11"*




The user can Figure 10-14 illustrates the possible connections that would result in a synchronous interface for bypass modes CIR 0x0D, bits [2:3] "01" and "10".

*Figure 10-14. Synchronous input clocks to flexiPCS quad for bypass modes "01" and "10"*




## Control & Status

The Control & Status pins for the SONET mode can be optionally selected on a per quad basis when generating the PCS quad interface files with the ispLEVER tools. Each of these control and status functions are also accessible through equivalent Quad Interface and Channel Interface Registers. The control and status signals allow direct real time access of these functions from FPGA logic. In addition to the Control and Status pins common to SONET Mode, a flexiPCS quad with 2 or 4 channel alignment selected has additional Control & Status pins related to control and monitoring of the multi-channel aligner.

**Control**

**felb** - Active high control signal which sets up all four channels in Far-End Loopback mode. This mode is generally used for testing the flexiPCS logic, looping back receive data back to the transmit direction without crossing the FPGA logic interface. For more information on the details of Far-End Loopback mode, see the **flexiPCS Testing** Section of the flexiPCS Data Sheet. The Far-End Loopback mode can also be initiated by writing Channel Interface Register Offset Address 0x00, bit 5 to '1'.

**mca_align_en -** Active high multi-channel align enable. Multi-channel Alignment Enable can also be set by writing the appropriate Quad Interface Register Offset Address 0x04, bits [4:7] to '1' (bit 4 for channel 3, bit 5 for channel 2, bit 4 for channel 1, and bit 7 for channel 0).

**mca_resync** - Active high, asynchronous reset to multi-channel aligner state machine and aligner FIFO control logic. mca_resync resets logic in all four channels in four channel alignment mode.

**Status**

**mca_aligned** - Active high signal indicating successful alignment of all four channels in four channel alignment mode. The multi-channel alignment status can also be read from the Quad Interface Register Offset Address 0x82, bit 2.

## Status Interrupt Registers

Some of the status registers associated with SONET operation have an associated status interrupt and status interrupt enable register. Any flexiPCS status interrupt register will generate an interrupt to the Systembus block (if used in the design). For a description of the operation of interrupts with the Systembus, refer to the Systembus section of the LatticeSC/M Family Data Sheet. Operation of the interrupt registers for the flexiPCS is as follows:

User must set the appropriate status interrupt enable bit to a '1'

Whenever the associated status bit goes high, its status interrupt bit will also go high. The status interrupt bit will remain high even if the associated status bit goes low.

The status interrupt bit will remain high until it is read, when it will automatically be cleared. If the associated status bit is still high, then the status interrupt bit will go high again (until read the next time).

Table 10-6 shows a listing of the status registers associated with PCI Express operation which have a corresponding status interrupt and status interrupt enable bit.

***Table 10-6. Status Interrupt Register Bits***

| Status Register Bit Function | Status Register Bit Address | Status Interrupt Enable Register Bit Address | Status Interrupt Register Bit Address |
|---|---|---|---|
| B1 Byte BIP Error Flag | CIR 0x83, bits [0:7] | CIR 0x03, bits [0:7] | CIR 0x98, bits [0:7] |
| RDI-L Detect Flag | CIR 0x8B, bit 5 | CIR 0x12, bit 5 | CIR 0x93, bit 5 |
| Out-of-frame Detection Flag | CIR 0x8B, bit 6 | CIR 0x12, bit 6 | CIR 0x93, bit 6 |
| Out-of-frame to In-frame Detection Flag | CIR 0x8B, bit 7 | CIR 0x12, bit 7 | CIR 0x93, bit 7 |
| Multi-channel Alignment State Machine Status (mca_aligned_01 in x1 mode or mca_aligned in x4 mode)<br><br>Channels 0 & 1 (2-channel alignment mode)<br><br>Channels 0,1,2 & 3 (4-channel alignment mode) | QIR 0x82, bit 2 | QIR 0x0C, bit 2 | QIR 0x83, bit 2 |
| Multi-channel Alignment State Machine Status (mca_aligned_23)<br><br>Channels 2 & 3 (2-channel alignment mode)<br><br>Inactive (4-channel alignment mode) | QIR 0x82, bit 3 | QIR 0x0C, bit 3 | QIR 0x83, bit 3 |

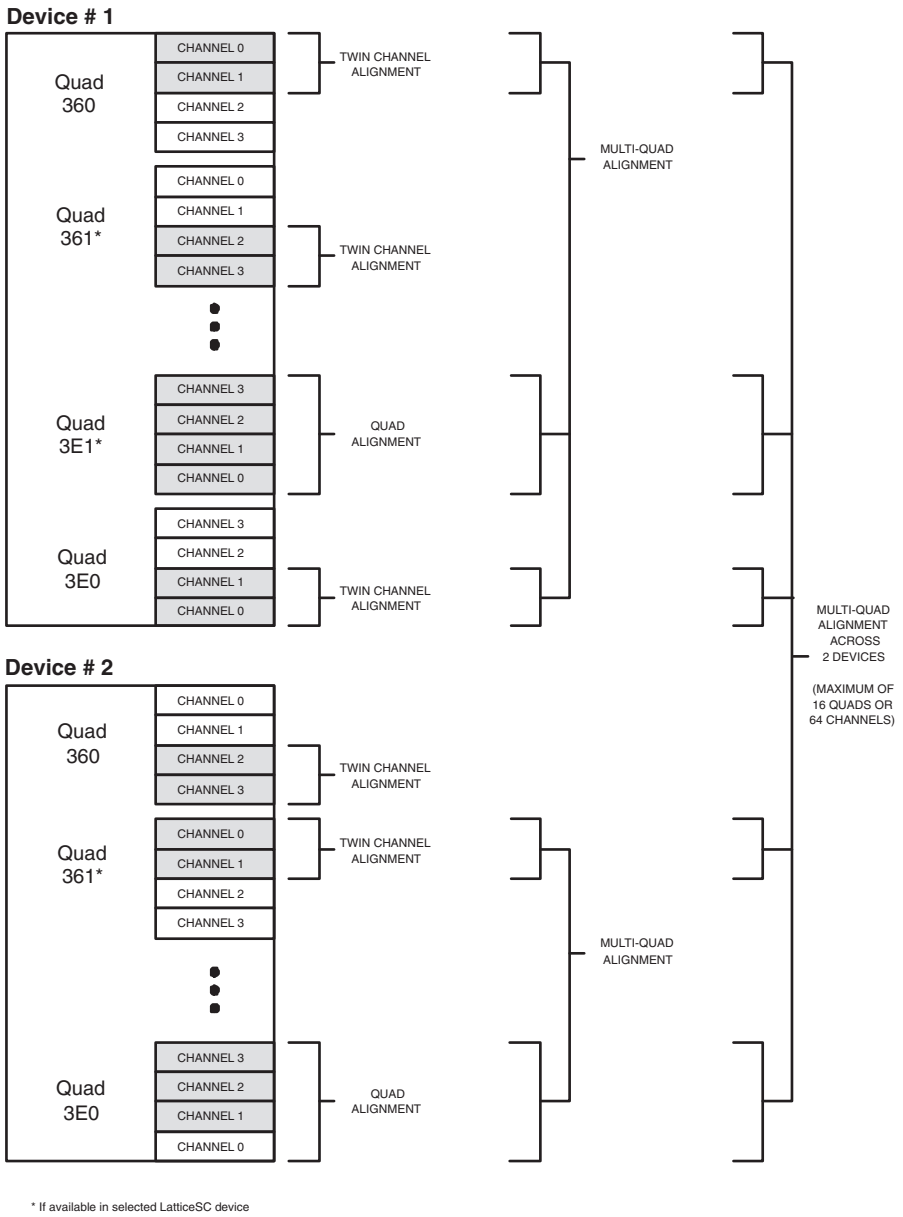
# Multi-Channel Alignment
# LatticeSC/M Family flexiPCS Data Sheet

**December 2008** **Data Sheet DS1005**

## Introduction

The Multi-channel Aligner allows the user to align as many as 64 receive data channels across two LatticeSC devices. The multi-channel aligner can align data between two channels in a quad (twin alignment), all four channels in a quad (quad alignment), channels between quads on a device, or quads between two devices. Up to four separate alignment group of two or more customer selected quads can be aligned per device, with one of these groups extendable across two devices. Examples of supported multi-channel alignment are shown in Figure 11-1.

***Figure 11-1. Multi-channel Alignment Examples***




* If available in selected LatticeSC device

© 2008 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

The Multi-channel Aligner for packet based protocols works by automatically detecting alignment bytes in the receive data streams and buffering the appropriate channels such that the alignment bytes in each channel are lined up on the same clock edge at the output of the multi-channel aligner.

The alignment bytes usually occur during the Idle sequence in packet based protocols. This is not a requirement for the multi-channel aligner. The user specifies up to two alignment bytes and the multi-channel aligner will align to those bytes regardless of where they appear in the incoming receive data streams. It is up to the user to specify the proper alignment byte for a given packet based protocol.

For SONET mode, channels are aligned by buffering the receive data such that the start of SONET frames are lined up at the output of the Multi-channel aligner. For SONET mode, no alignment byte needs to be specified as the multi-channel aligner aligns SONET start-of-frame pulses that are generated by the embedded flexiPCS receive SONET framers.

## Supported Protocols

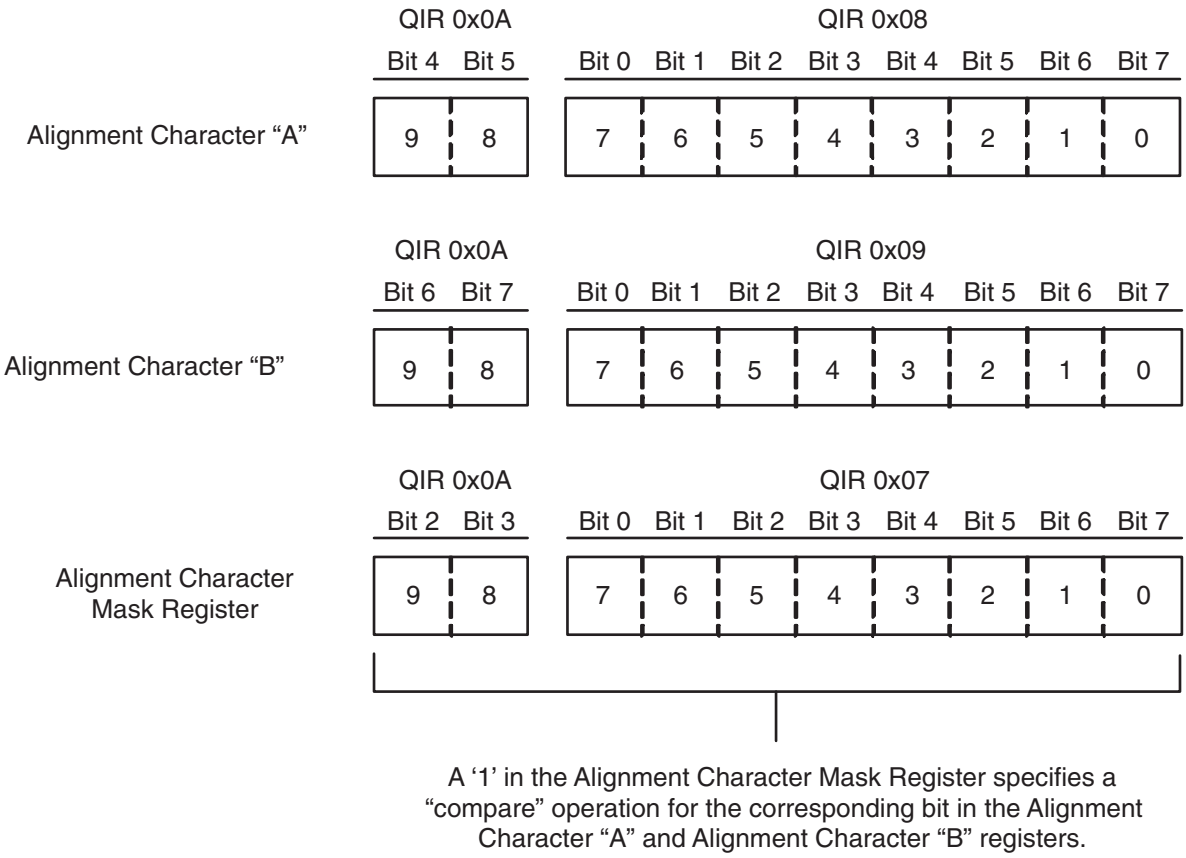
The multi-channel aligner can support many industry standard data protocols. Alignment is performed according to the requirements of the particular specification. IPexpress will generate an Auto-configuration file which will set up multi-channel and multi-quad alignment registers appropriately for the chosen mode. Multi-quad alignment also requires the instantiation of the Systembus. In that case, a separate Auto-configuration file for setting up the multi-channel Systembus registers is supplied by IPexpress for the Systembus. The details provided in this section are for users who wish to understand the operation of the multi-channel and multi-quad alignment operation in detail or wish to set up their own non-standard alignment configuration. Table 11-1 shows the alignment conditions for several supported protocols.

***Table 11-1. Multi-Channel Alignment Protocol Settings.***

| Standard | Alignment Condition |
|---|---|
| SONET | A1/A2 frame pulse (rx_fp signal) |
| XAUI | K control bit = 1, data[7:0] = 0x7C |
| Multi-Channel PCI Express | data[8] = 1, data[7:0] = 0xBC |
| Multi-Channel RapidIO | data[8] = 1, data[7:0] = 0xFB |

In addition to the standard protocol alignment support, the multi-channel aligner also supports user defined alignment. For each quad, the user can define up to two 10-bit alignment characters. The user can also define a 10-bit mask register as shown in Figure 11-2.

***Figure 11-2. Multi-channel Alignment Registers***



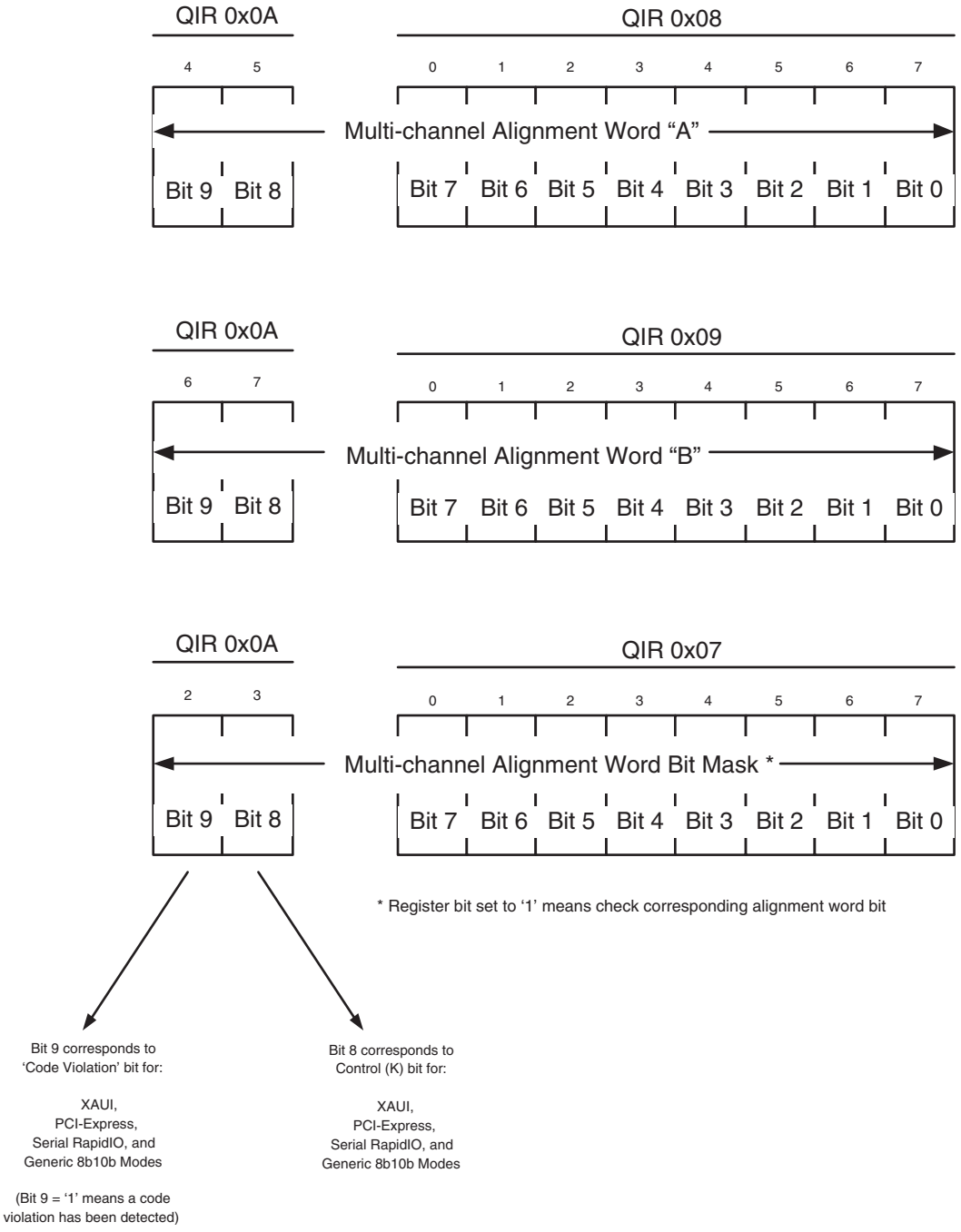

User defined alignment character "A" can be set by writing to Quad Interface Register Base Address 0x0A, bits [4:5] (for bits [9:8] of the "A" alignment character) and Quad Interface Register Base Address 0x08, bits [0:7] (for bits [7:0] of the "A" alignment character).

User defined alignment character "B" can be set by writing to Quad Interface Register Base Address 0x0A, bits [6:7] (for bits [9:8] of the "B" alignment character) and Quad Interface Register Base Address 0x09, bits [0:7] (for bits [7:0] of the "B" alignment character).

The user defined alignment character mask register can be set by writing to Quad Interface Register Base Address 0x0A, bits [2:3] (for bits [9:8] of the alignment character mask register) and Quad Interface Register Base Address 0x07, bits [0:7] (for bits [7:0] of the alignment character mask register).

Figure 11-3 illustrates the multi-channel alignment registers. Note that the multi-channel aligner will always align to both multi-channel alignment words "A" and "B". If a specific application calls for only one multi-channel alignment word, both alignment words "A" and "B" must be set to the same value.

***Figure 11-3. Multi-channel Word Alignment Registers for Packet Protocols***

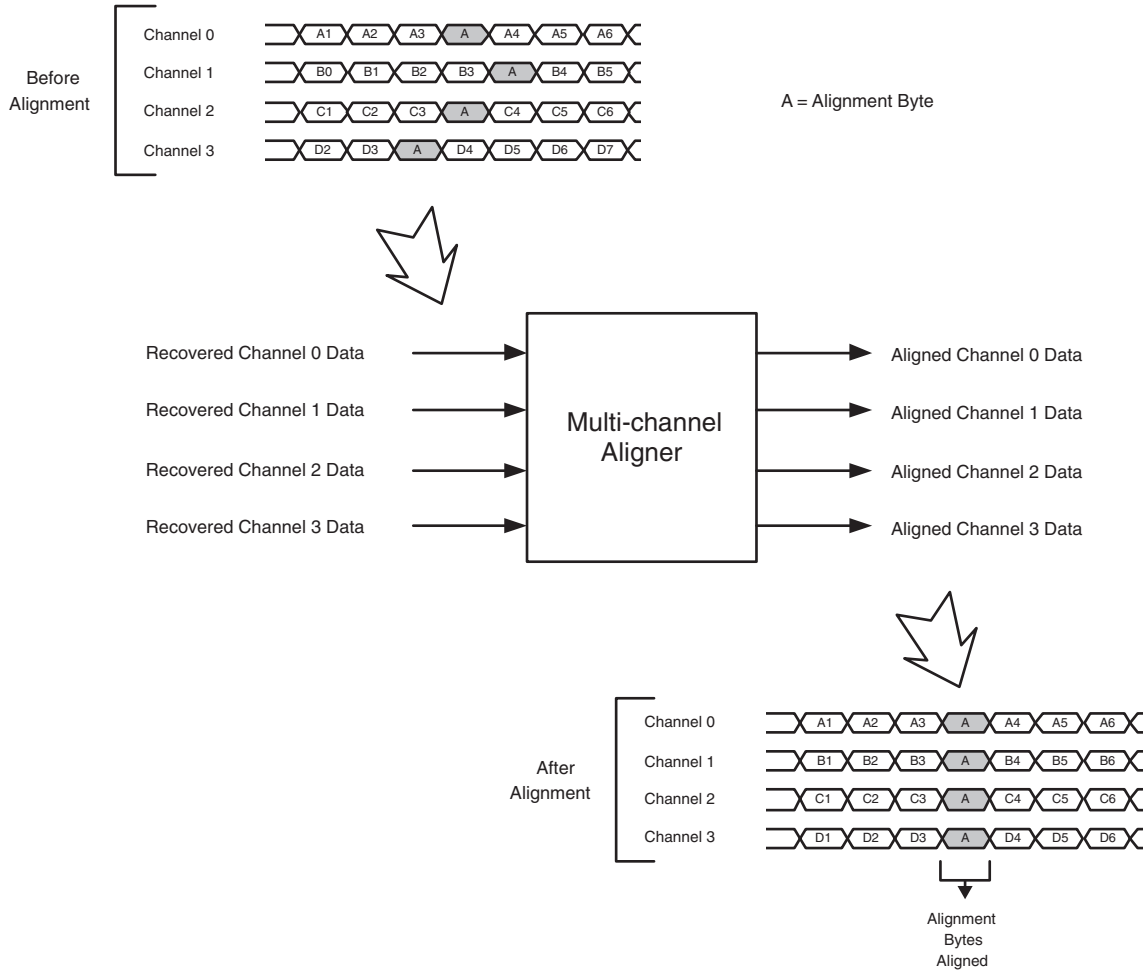



## Alignment Within A Quad

The flexiPCS Multi-channel Aligner allows for alignment of two or four channels within a quad. In packet based protocols, the Multi-channel Aligner will align channels based on a user-defined alignment character which must be present in the data stream for all receive channels that are to be aligned. Typically this character is inserted simultaneously in all channels during an Idle sequence between packets upon transmission. If arriving data is skewed due to unequal transport delays, the presence of the alignment characters allows the Multi-channel Aligner to re-align the skewed channels.

Figure 11-4 shows an example of the operation of the flexiPCS multi-channel aligner operation for 4 channel alignment for a packet based protocol. The Alignment bytes in each channel are lined up by the multi-channel aligner to create an aligned data stream at the PCS/FPGA interface.

***Figure 11-4. Four Channel Alignment Example for Packet Protocols***




The following is a procedure for settings registers for Multi-channel Alignment.

1. Set the mode for the Multi-channel Aligner and enable/disable the appropriate channels for alignment.

The multi-channel aligner can be set to align two or four channels in a quad by writing to Quad Interface Register Offset Address 0x19.

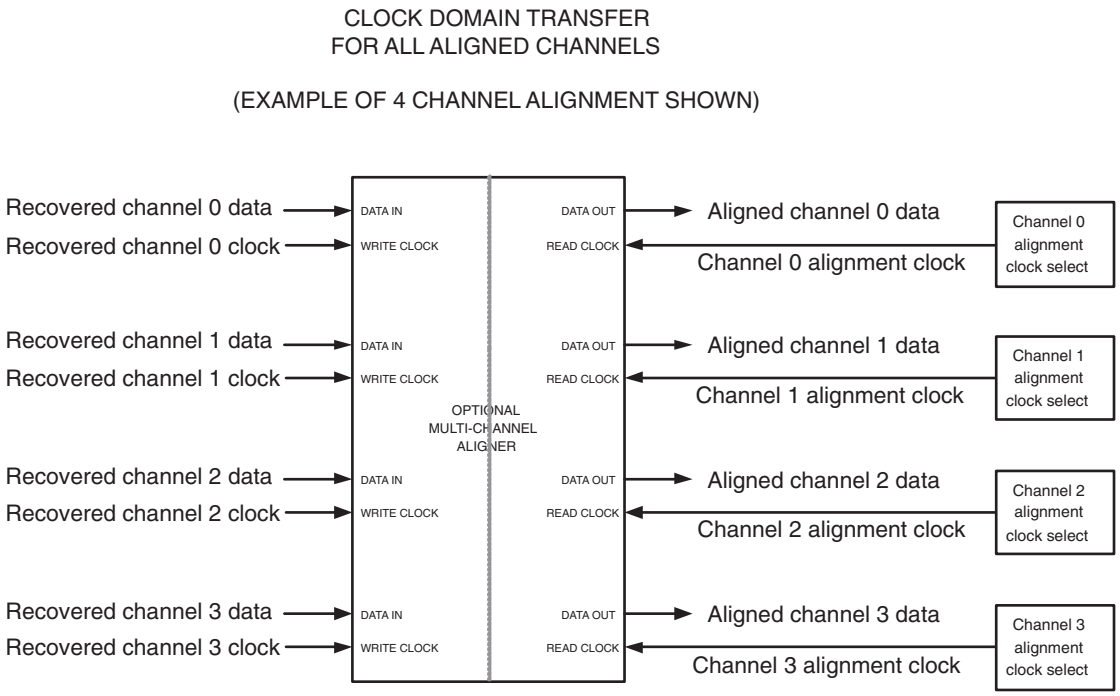
- Bit 3 controls alignment between Channel 0 and Channel 1. When bit 3 is set to ‘1’, Channels 0 and 1 will be aligned.
- Bit 2 controls alignment between Channel 2 and Channel 3. When bit 2 is set to ‘1’, Channels 2 and 3 will be aligned.
- If Bit 2 and bit 3 are set to ‘1’ simultaneously, then Channel 0 will be aligned to Channel 1 and Channel 2 will be aligned to Channel 3, but Channels 0/1 will act independently of Channels 2/3.
- Bit 1 controls alignment between Channels 0, 1, 2, and 3. When bit 1 is set to ‘1’, Channels 0, 1, 2, and 3 will be aligned.

- Each channel to be aligned must also be enabled for alignment by writing to the appropriate bit in Quad Interface Register Offset Address 0x04. Write a '1' to bit 7 to include channel 0 for alignment. Write a '1' to bit 6 to include channel 1 for alignment. Write a '1' to bit 5 to include channel 2 for alignment. Write a '1' to bit 4 to include channel 3 for alignment. Multi-channel alignment can also be enabled by setting the **mca_align_en** port at the FPGA interface high.

2. Set the Multi-channel Alignment output clocks. A clock domain transfer occurs between the inputs and outputs of the Multi-channel Aligner. Each channel's receive data is clocked into the Multi-channel Aligner with its own separate recovered receive clock. Each channel's receive data is clocked out of the Multi-channel Aligner on a unique multi-channel alignment clock. Each multi-channel alignment clock can be programmed to be any of the recovered receive clocks. For all channels to be aligned, their alignment clock must be frequency locked (0 ppm frequency difference) to the corresponding recovered receive clocks. The alignment clock can be of arbitrary phase with respect to the corresponding recovered receive clocks.

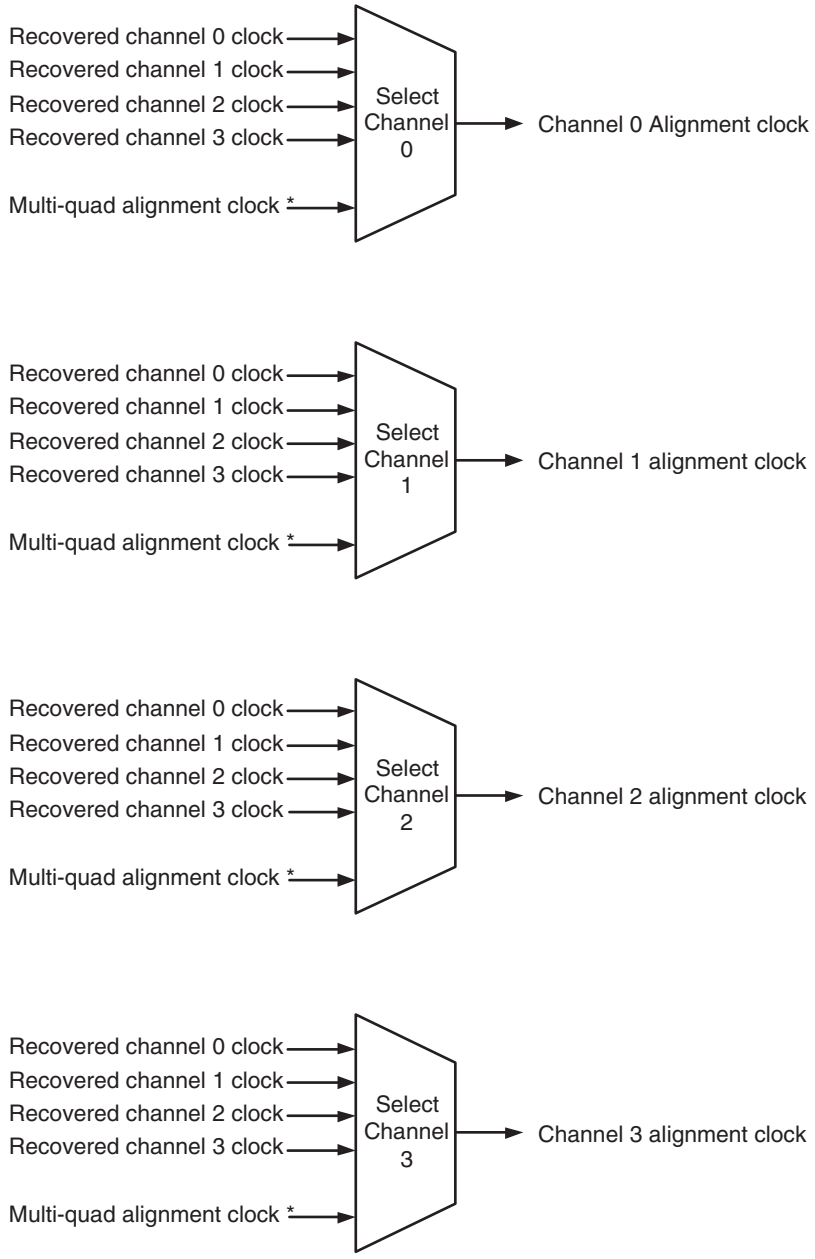
Figure 11-5 shows this clock domain transfer at the Multi-channel Aligner.

***Figure 11-5. Multi-channel Aligner Input and Output Clocks***




Each channel's Multi-channel Alignment output clock can be driven from any of the recovered channel input clocks. This relationship is shown in Figure 11-6. It is expected that the user will assign the same recovered receive clock to all the multi-channel output clocks for all channels that are to be aligned to each other. That way, all aligned channels coming out of the Multi-channel Aligner are phase aligned to the same clock. Note that these clocks are driven by a common multi-quad alignment group clock when multi-quad alignment is enabled (see **Alignment Between Quads** section):

***Figure 11-6. Multi-channel Aligner Output Clock Selection***




* Multi-quad alignment clock used for all channels assigned to a multi-channel alignment group clock when multi-quad alignment is enabled

The Multi-channel Alignment clock assignments are set by writing to Quad Interface Register Offset Address 0x01. Table 11-2 shows the recovered channel clock selection for all values of the selection bits:

*Table 11-2. Alignment Channel Clock Selection*

| Quad Interface Register 0x01 Bits | Value | Recovered Channel Clock Selection |
|---|---|---|
| Channel 0 Alignment Clock Selection: bits [6:7] | 00 | Recovered channel 0 clock |
| Channel 1 Alignment Clock Selection: bits [4:5] | 01 | Recovered channel 1 clock |
| Channel 2 Alignment Clock Selection: bits [2:3] | 10 | Recovered channel 2 clock |
| Channel 3 Alignment Clock Selection: bits [0:1] | 11 | Recovered channel 3 clock |

For example, in a 4 channel alignment application, in order to set up all Multi-channel Alignment clocks to be sourced from the channel 0 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0x00. To set all Multi-channel Alignment clocks to be sourced from the channel 1 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0x55. To set all Multi-channel Alignment clocks to be sourced from the channel 2 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xAA. To set all Multi-channel Alignment clocks to be sourced from the channel 3 recovered receive clock, set Quad Interface Register Offset Address 0x01 to 0xFF.

For modes which require Clock Tolerance Compensation (Gigabit Ethernet, Fibre Channel, XAUI, PCI Express, Serial RapidIO), the operation of the Clock Tolerance Compensation block may determine the choice of recovered receive clock. The Clock Tolerance Compensation block must perform insertion or deletion of the defined SKIP byte across all aligned channels simultaneously in order to preserve the multi-channel alignment. This places the following restrictions on choice of alignment clocks for the selected alignment mode.

When two channel alignment for channels 0 and 1 is selected, channel 0 is selected as the master channel for performing clock tolerance compensation. This means that only channel 0 is monitored for the presence of the SKIP character. Insertion or deletion is performed on channels 0 and 1 simultaneously based on detection of the SKIP character in channel 0. This means that channel 0 must be active and transmitting data for the clock tolerance compensation function to work in two channel alignment mode for channels 0 and 1. If channel 0 is not active, receive data will not be present on the rxd_0 and rxd_1 outputs of the flexiPCS.

When two channel alignment for channels 2 and 3 is selected, channel 2 is selected as the master channel for performing clock tolerance compensation. This means that only channel 2 is monitored for the presence of the SKIP character. Insertion or deletion is performed on channels 2 and 3 simultaneously based on detection of the SKIP character in channel 2. This means that channel 2 must be active and transmitting data for the clock tolerance compensation function to work in two channel alignment mode for channels 2 and 3. If channel 2 is not active, receive data will not be present on the rxd_2 and rxd_3 outputs of the flexiPCS.

When four channel alignment is selected, channel 0 is selected as the master channel for performing clock tolerance compensation. This means that only channel 0 is monitored for the presence of the SKIP character. Insertion or deletion is performed on all four channels simultaneously based on detection of the SKIP character in channel 0. This means that channel 0 must be active and transmitting data for the clock tolerance compensation function to work in four channel alignment mode. If channel 0 is not active, receive data will not be present on the rxd outputs of the flexiPCS.

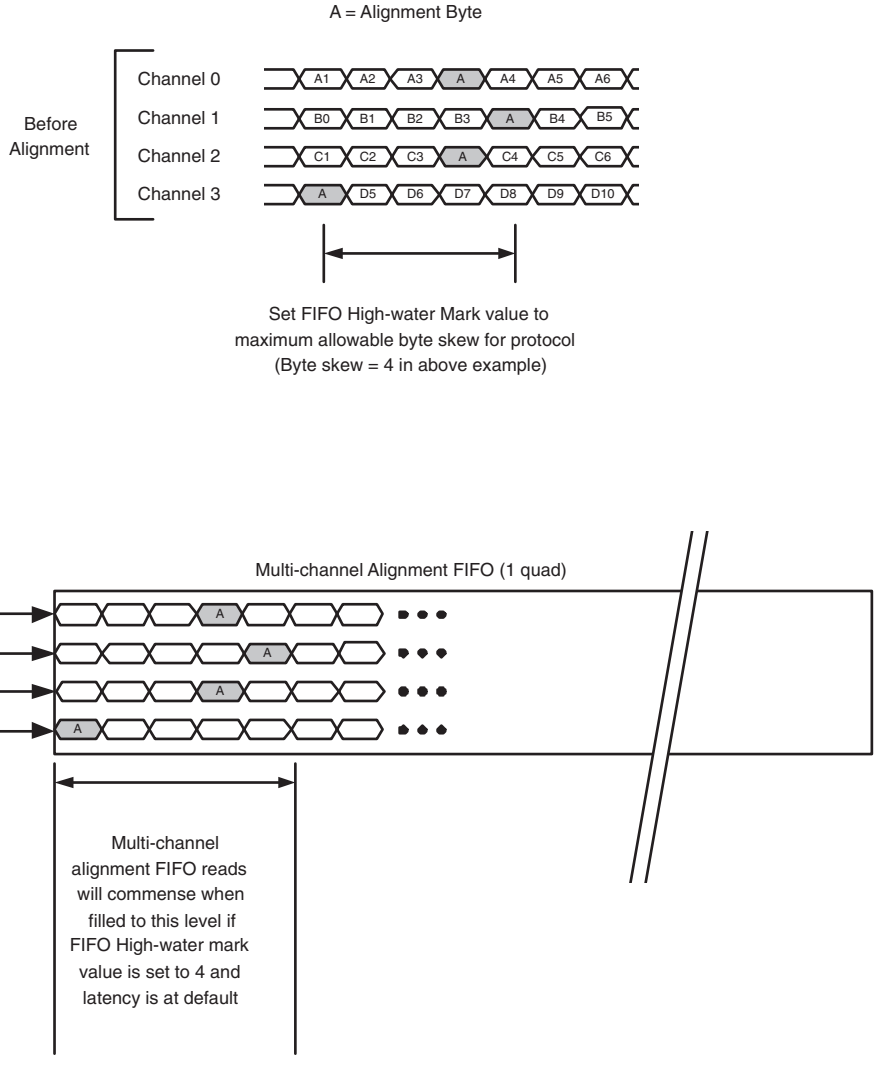
3. For packet protocol applications (any mode except SONET mode), the user must set values for the Multi-channel Alignment characters. The Multi-channel Aligner provides the ability to align channels of incoming data to two 10-bit user defined maskable patterns. Both alignment characters must be set to the same value if only one alignment character is defined for an application.

- A user programmable mask word allows the user to set which individual bits are compared to the user defined channel alignment character. When a ‘1’ is present in the user programmable mask, the corresponding bit of the channel alignment character is compared.

- The lowest 8 significant bits [7:0] of the user programmable mask can be written to Quad Interface Register Offset Address 0x07, bits [0:7]. Bits [9:8] of the user programmable mask can be written to Quad Interface Register Offset Address 0x0A, bits [2:3].
- The lowest 8 significant bits [7:0] of the first channel alignment character can be written to Quad Interface Register Offset Address 0x08, bits [0:7]. Bits [9:8] of the first channel alignment character can be written to Quad Interface Register Offset Address 0x0A, bits [4:5].
- The lowest 8 significant bits [7:0] of the second channel alignment character can be written to Quad Interface Register Offset Address 0x09, bits [0:7]. Bits [9:8] of the second channel alignment character can be written to Quad Interface Register Offset Address 0x0A, bits [6:7].

4. Set the Multi-channel Aligner FIFO high-water mark and latency values for a quad. The FIFO high-water mark values from 0 to 63 can be set by writing to Quad Interface Register Offset Address 0x06, bits [2:7]. The FIFO high-water mark should be set to the maximum the number of bytes of skew allowed for de-skewing. Figure 11-7 shows an example of how alignment byte skew between channels relates to the FIFO high-water mark value for a four channel alignment application.

***Figure 11-7. Multi-channel alignment FIFO high-water mark example***




Latency values from 0 to 31 can be set by writing to Quad Interface Register Offset Address 0x05, bits [3:7]. Latency sets an upper limit on the number of bytes after the last channel alignment byte has been loaded into the

multi-channel alignment FIFO before the alignment data is read out of the FIFO. The default latency is 32 bytes. If the skew between the first and last alignment byte is less than the programmed High Water Mark, the multi-channel alignment FIFO will continue to load for (High Water Mark - Skew)/2 bytes, or (Latency) bytes, whichever is less. This provides a buffer in case an individual channel's multi-channel aligner write clock drifts with respect to the read clock which could result in a FIFO underflow after multi-channel alignment has been achieved. Figure 11-8 illustrates an example for when (High Water Mark - Skew)/2 is less than the programmed latency.

***Figure 11-8. Multi-channel Alignment FIFO High-water Mark and Latency Value Example***

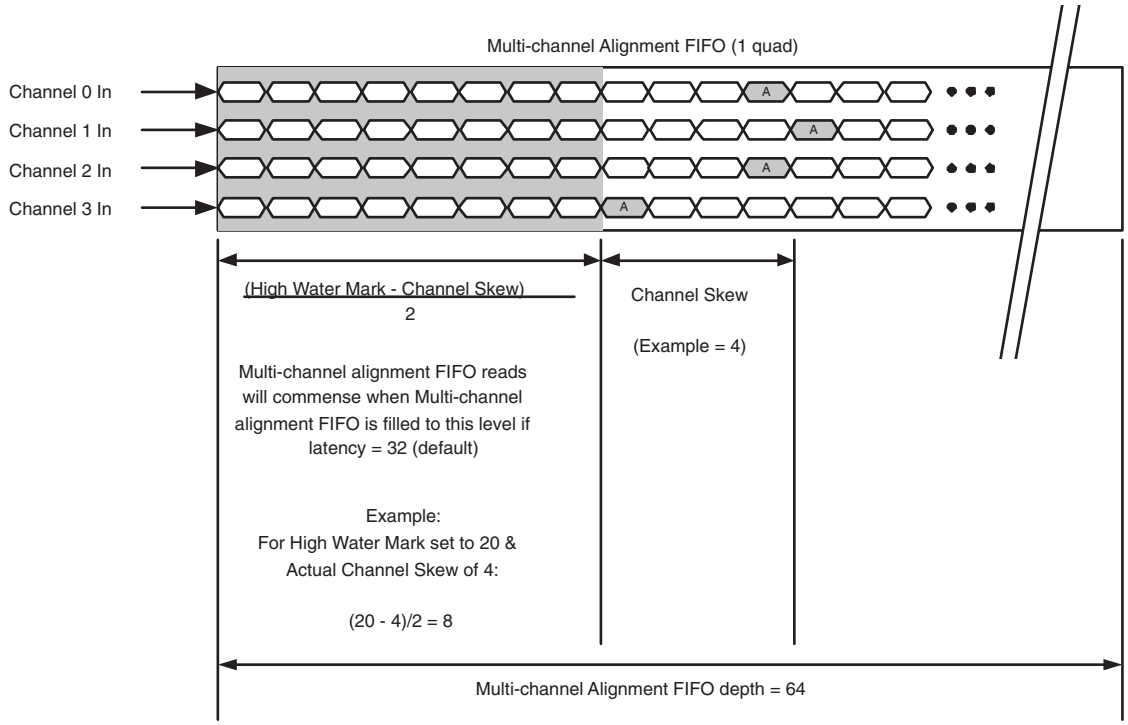
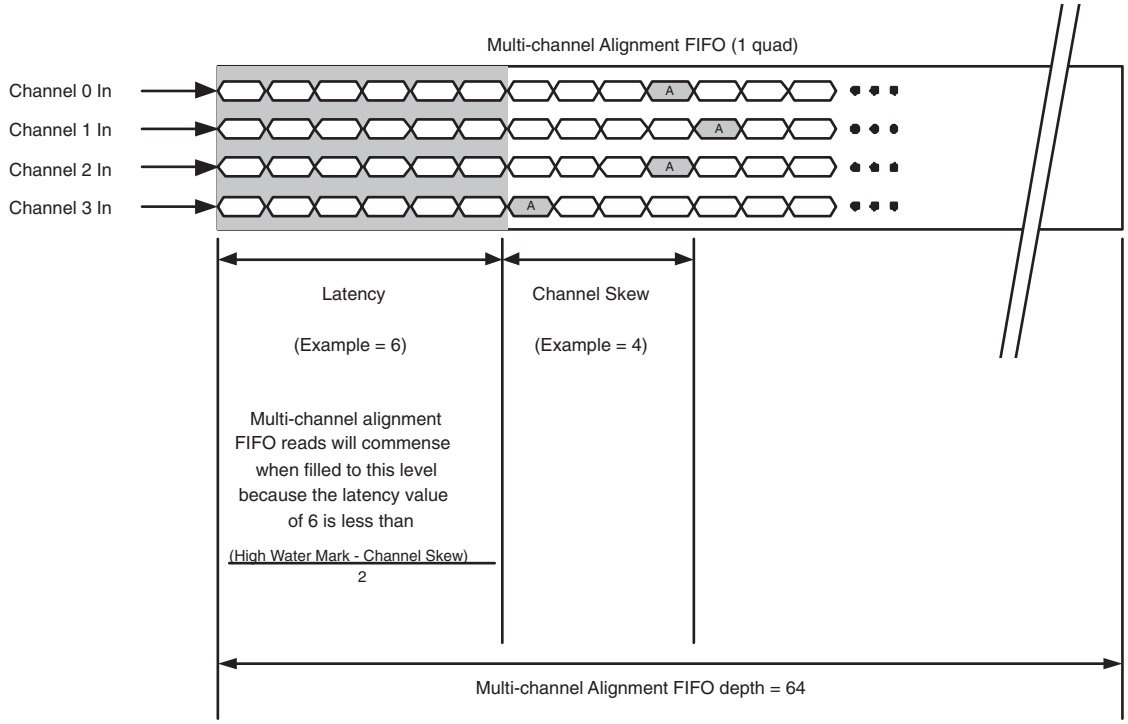



Figure 11-9 illustrates an example when the programmed latency is less than (High Water Mark - Skew)/2.

***Figure 11-9. Multi-channel Alignment FIFO Optimal Latency Settings***

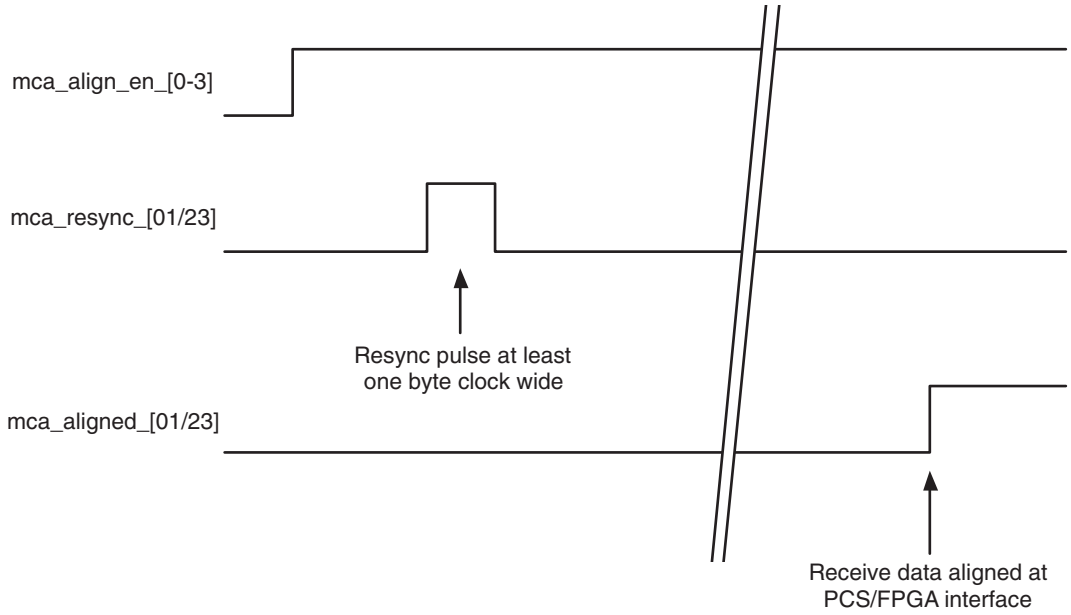



5. Reset the Multi-channel Aligner state machine and FIFO control logic with the **mca_resync_01** and/or **mca_resync_23** port(s) at the FPGA interface.

All flexiPCS modes which support multi-channel alignment also provide multi-channel alignment control and monitoring ports at the PCS/FPGA interface (usually when the “Optional Direct Status & Control Register Access” selection is made in IPexpress. Control ports include mca_align_en_[0-3], mca_resync_[01/23], and mca_aligned_[01/23]. Each of these ports has an equivalent control or status register bit (written or read from the Systembus) which controls/monitors the same function. The user’s particular application will determine if these functions need to be accessed directly via these optional ports.

The relationship of these signals during multi-channel alignment is shown in Figure 11-10.

*Figure 11-10. Multi-channel Alignment Control and Monitoring Signals for Multi-channel Alignment (1 Quad)*




Unless the channel alignment state machines are disabled, channel alignment synchronization is controlled by one of two embedded channel alignment state machines. For all modes except Serial RapidIO, the XAUI alignment state machine is used (PCS Deskew State Diagram in the IEEE803.2ae-2002 Specification). For Serial RapidIO Mode (QIR 0x18, bit 6 = '1'), the Serial RapidIO alignment state machine is used (Lane Alignment State Machine in the RapidIO Physical Layer 1x/4x LP-Serial Specification, Revision 1.2). In cases where the channel alignment state machines are enabled (default), channel resynchronization is automatically initiated whenever alignment is lost, according to the appropriate specification.

If the alignment state machine is enabled (QIR 0x04 bit 3=0, for 01 or 0123 alignment), the MCA continuously resets the MCA alignment until the MCA finds the set of alignment markers within HIGH_WATERMARK cycles. In this mode of usage, the mca_aligned_01/23 (or QIR 0x82 bit 0) high indicates the alignment state machine observed good alignment at the MCA output. When mca_aligned_01/23 or (QIR 0x82 bit 0) goes low, it indicates there is a problem that results in the MCA being unable to deskew the lanes.

The channel alignment state machines can be disabled by writing QIR 0x04, bit 3 to a '1' (for Channels 0 and 1 in 0/1 two channel alignment mode or Channels 0, 1, 2, and 3 in four channel alignment mode), or by writing QIR 0x04, bit 2 to a '1' (for Channels 2 and 3 in 2/3 two channel alignment mode). Note that if the state machines are disabled, channel re-synchronization will only be initiated by the appropriate user controlled **mca_resync** inputs (see below).

**mca_align_en_[0-3]** - Each channel has a separate multi-channel alignment enable which must be active (high) for any channel to be aligned. Equivalent to control register at Quad Interface Register Offset Address 0x 04, bit 4 (channel 3), bit 5 (channel 2), bit 6 (channel 1), and bit 7 (channel 0).

**mca_resync_[01/23]** - Resynchronization signal which resets the multi-channel alignment for the appropriate channels. When in one of the two channel alignment modes, mca_resync_01 resets alignment for channels 0 and 1, and mca_resync_23 resets alignment for channels 2 and 3. When in four channel alignment mode, mca_resync_01 resets alignment for channels 0, 1, 2, and 3, and mca_resync_23 provides no function. Equivalent to control register at Quad Interface Register Offset Address 0x 41, bit 7 (mca_resync_01), and bit 6 (mca_resync_23).

**mca_aligned_[01/23]** - Status monitor signal indicating (when high) that multi-channel alignment for the appropriate channels have been achieved. When in one of the two channel alignment modes, mca_aligned_01 reports alignment status for channels 0 and 1, and mca_aligned_23 reports alignment status for channels 2 and 3. When in four channel alignment mode, mca_aligned_01 reports alignment status for channels 0, 1, 2, and 3, and mca_aligned_23 provides no function. Equivalent to status register at Quad Interface Register Offset Address 0x 82, bit 2 (mca_aligned_01), and bit 3 (mca_aligned_23). mca_aligned_01 is only valid when performing multichannel alignment within a single quad. mca_aligned_[01/23] is not valid in SONET mode. In SONET mode, when all channels of data are aligned, all rx_fp will be aligned and come out at the same time.

The following status register bits are useful for multichannel alignment.

1. QIR 0x82, bit 6 (for 01and 0123 lane alignment) is an inskew indicator, which means that all alignment characters across all channels were received within a required window. This bit indicates that the most recently encountered group of alignment markers are within the required window. The window starts at the first encountered alignment marker and ends HIGH_WATERMARK (QIR 0x05 bits [2:7} clocks later. The window then restarts on the next encountered alignment marker on any enabled lane.

2. QIR 0x82, bit 4 (for 01 and 0123 lane alignment) is the outskew bit. This bit indicates that the most recently encountered group of alignment markers was outside the required window. The window starts at the first encountered alignment marker and ends HIGH_WATERMARK (QIR 0x05 bits [2:7] clocks later. The window then restarts on the next encountered alignment marker on any enabled lane.

## Alignment Between Quads

Channels in different quads on the same device can also be aligned. Depending on the LatticeSC device chosen, this will allow up to 32 channel alignment on one chip. Note that mca_aligned_01 is not valid when aligning between quads because the signal comes from the multichannel alignment state machine which is disabled in this mode. On one device, up to 4 multi-quad alignment groups can be defined that can be aligned independently of each other. Each flexiPCS quad can be assigned to any of the four multi-quad alignment groups. Figure 11-11 shows an example where channels 0 and 1 in quad 360 are to be aligned with all four channels in quad 3E1. In this example, the following steps set up this multi-quad alignment:

1. Set Quad 360 to x2 channel alignment mode for channels 0 and 1 (QIR 0x19, bit 3 = ‘1’).

2. Set Quad 3E1 to x4 channel alignment mode (QIR 0x19, bit 1 = ‘1’).

3. Enable all channels to be aligned in each quad for multi-channel alignment (QIR 0x04, bit [4:7]). For Quad 360, QIR 0x04, bits [4:7] = “03.” For Quad 3E1, QIR 0x04, bits [4:7] = “0F.”

4. Configure Quads 360 and 3E1 for multi-quad alignment (QIR 0x00, bit 3 = ‘1’ and IIR 0x00 and IIR 0x05, bit 3 =’1’).

5. Select one of the four multi-quad alignment group clocks as the multi-quad alignment group clock for this set of six channels. A source clock for that multi-quad alignment group can be selected from either quad 360 or 3E1 in this example (QIR 0x00, bit [4:5] and either IIR 0x08 or IIR 0x09, depending on the source quad chosen). The source clock for the multi-quad alignment group should be a recovered receive clocks from one of the six channels to be aligned.

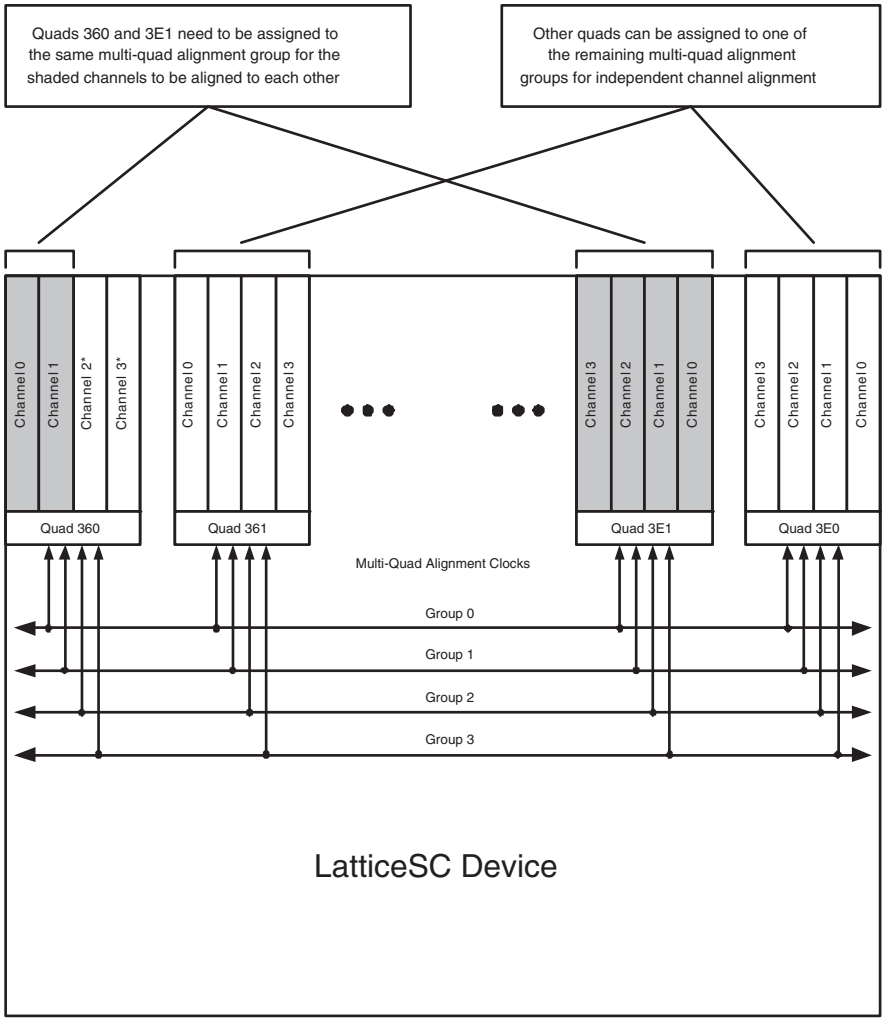
6. Disable alignment state machine for all quads to be aligned (QIR 0x04, bit 2 ='1' when aligning channels 2 and 3, bit 3='1' when aligning channels 0 and 1 or channels 0,1,2 and 3 in x4 mode).

7. Select the chosen multi-quad alignment group clock as input to both quads 360 and 3E1 (QIR 0x00, bits [6:7] and IIR 0x00 and IIR 0x05, bits [6:7]).

8. Resynchronize the multi-channel aligners in both quads simultaneously by routing the same signal to their respective **mca_resync_01** inputs. Note that while resynchronization can also be performed by writing to QIR

0x41, bit 7, this could not be done simultaneously for multiple quads and so the PCS/FPGA interface port must be used.

***Figure 11-11. Multi-quad alignment example***




* Channels 2 and 3 in this example can be aligned to each other, but cannot be part of a multi-quad group

When configured for multi-quad alignment, a flexiPCS quad can be set to any of the multi-channel alignment modes:

Two channel alignment for channels 0 & 1 (by writing Quad Interface Register Offset Address 0x19, bit 3 to '1')

Two channel alignment for channels 2 & 3 (by writing Quad Interface Register Offset Address 0x19, bit 2 to '1')

Both two channel alignment modes simultaneously (Channels 0 & 1 together and Channels 2 & 3 together by writing Quad Interface Register Offset Address 0x19, bits [2:3] to "11")

Four channel alignment (by writing Quad Interface Register Offset Address 0x 19, bit 1 to '1')

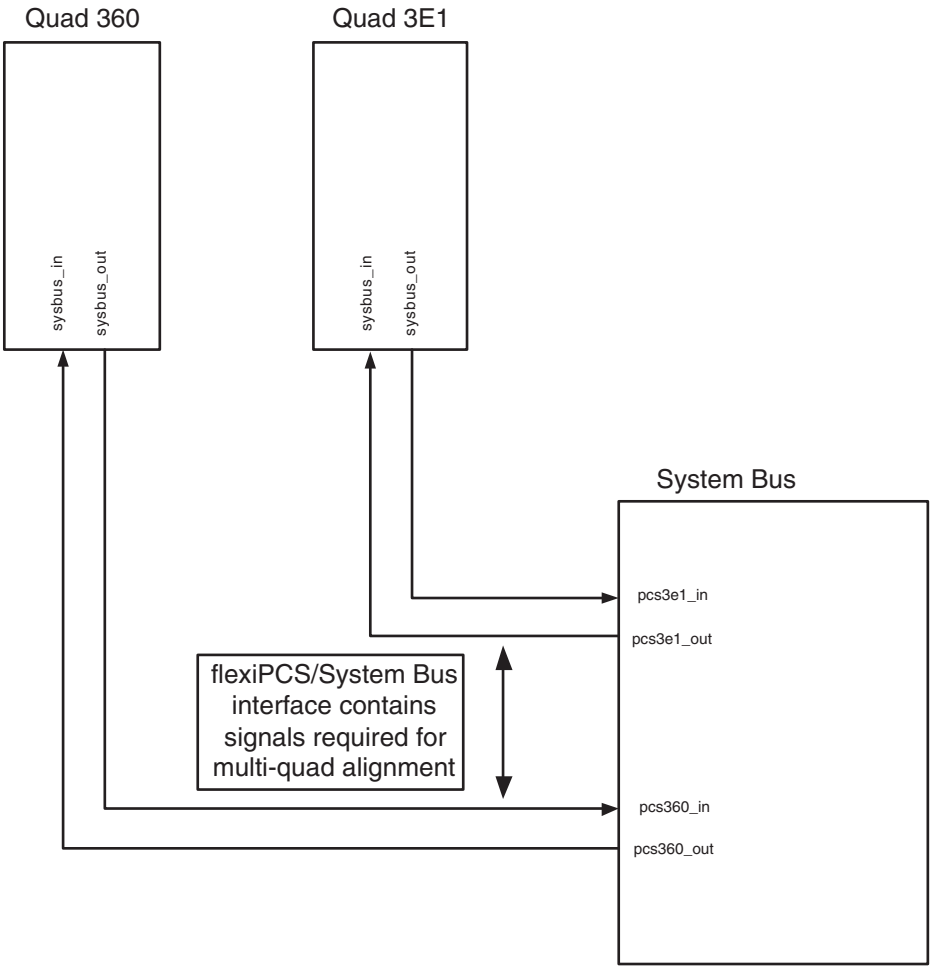
Only channels 0 and 1 in two channel alignment mode or channels 0, 1, 2, and 3 in four channel alignment mode can be assigned to a multi-quad alignment group. Channels 2 and 3 in two channel alignment mode cannot be assigned to a multi-quad alignment group. In the case where channels 0 and 1 in two channel alignment mode are

configured for multi-quad alignment with channels in another quad, channels 2 and 3, when configured for channel 2/3 two channel alignment mode, can be aligned to each other independently of channels 0 and 1, if desired.

All quads assigned to the same multi-quad alignment group must have their FIFO high-water marks set to the same value. Likewise, all quads assigned to the same multi-quad alignment group must have their latency value set to the same value.

Multi-quad alignment requires the instantiation of the Systembus block, which passes alignment control signals and clock between all the flexiPCS quad on a LatticeSC device. An example of the interface connection of two flexiPCS and the Systembus is shown in Figure 11-12 below. In this example, quads 360 and 3E1 could be configured for multi-quad alignment if desired.

***Figure 11-12. Multiple flexiPCS/Systembus Connections***

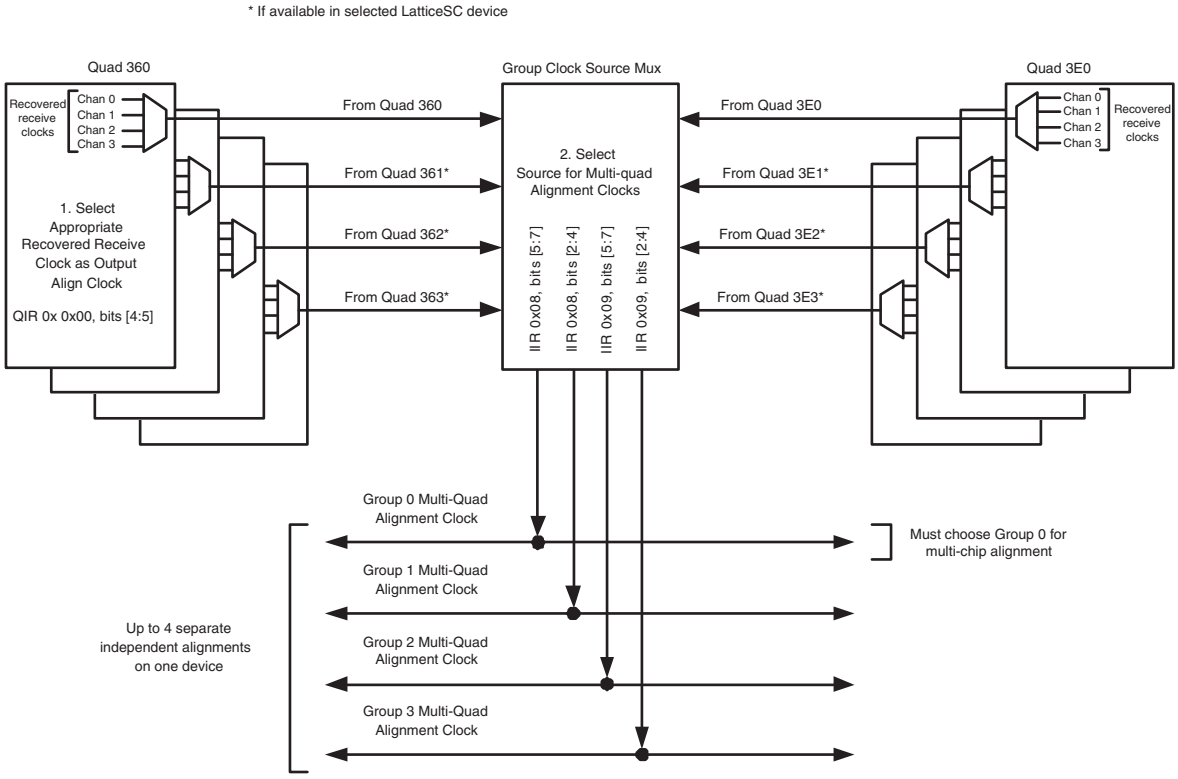



**Assigning Quads to Multi-Quad Group Clocks**

When a set of flexiPCS quads are connected to the Systembus block, multi-quad alignment can be set up be writing appropriate registers to perform the following operations:

1. A source for a common multi-quad alignment clock must be selected. A diagram of the Multi-quad Alignment Group clock source setup is shown in Figure 11-13 below:

***Figure 11-13. Multi-quad Alignment Group Clock Source Setup***




The LatticeSC device has four independent clocks available for multi-quad alignment (referred to as group 0, group 1, group2, and group 3). The source for each of these four multi-quad clocks can come from any of the available flexiPCS quads (up to a maximum of 8) on the chosen LatticeSC device. The source for the group 0 and group 1 multi-quad alignment clocks is selected by writing to the Inter-quad Interface Register Offset Address 0x08, bits [5:7] (group 0), and bits [2:4] (group 1). The source for the group 2 and group 3 multi-quad alignment clocks is selected by writing to the Inter-quad Interface Register Offset Address 0x09, bits [5:7] (group 2), and bits [2:4] (group 3). Table 11-3 shows the bit settings for all possible multi-quad alignment clock sources.

***Table 11-3. Multi-quad Alignment Group Clock Source Register Settings***

| Inter-quad Interface Register Bits | Bit settings | Group Clock Source Quad |
|---|---|---|
| Group 0 Source Clock Register: IIR 0x08, bit [5:7]<br>Group 1 Source Clock Register: IIR 0x08, bit [2:4]<br>Group 2 Source Clock Register: IIR 0x09, bit [5:7]<br>Group 3 Source Clock Register: IIR 0x09, bit [2:4] | 000 | 360 |
| | 001 | 361 |
| | 010 | 362 |
| | 011 | 363 |
| | 100 | 3E0 |
| | 101 | 3E1 |
| | 110 | 3E2 |
| | 111 | 3E3 |

The Source Quad Group Clock is selected among the four recovered receive clocks, one per receive channel. The selected recovered receive clock must match the clock chosen as the multi-channel aligner output clock for the channel to be aligned (See Table 11-2 from “Alignment Within A Quad” above). This recovered receive clock is selected by writing the appropriate Quad Interface Register Offset Address 0x00, bits [4:5]. The source quad clock selection is shown in Table 11-4.

*Table 11-4. Source Quad Clock Selection*

| Quad Interface Register 0x00 | | |
|---|---|---|
| Bit 4 | Bit 5 | Output Quad Clock Source Selection |
| 0 | 0 | Recovered Receive Channel 0 Clock |
| 0 | 1 | Recovered Receive Channel 1 Clock |
| 1 | 0 | Recovered Receive Channel 2 Clock |
| 1 | 1 | Recovered Receive Channel 3 Clock |

2. The quads to be aligned must be enabled for multi-quad alignment. This is done by writing the appropriate Quad Interface Register Offset Address 0x00, bit 3 to a '1'. In addition, Inter-quad Interface Register bits must also be set for multi-quad alignment enable. Table 11-5 shows which Inter-quad Interface Registers must be set for each potential quad to be aligned.

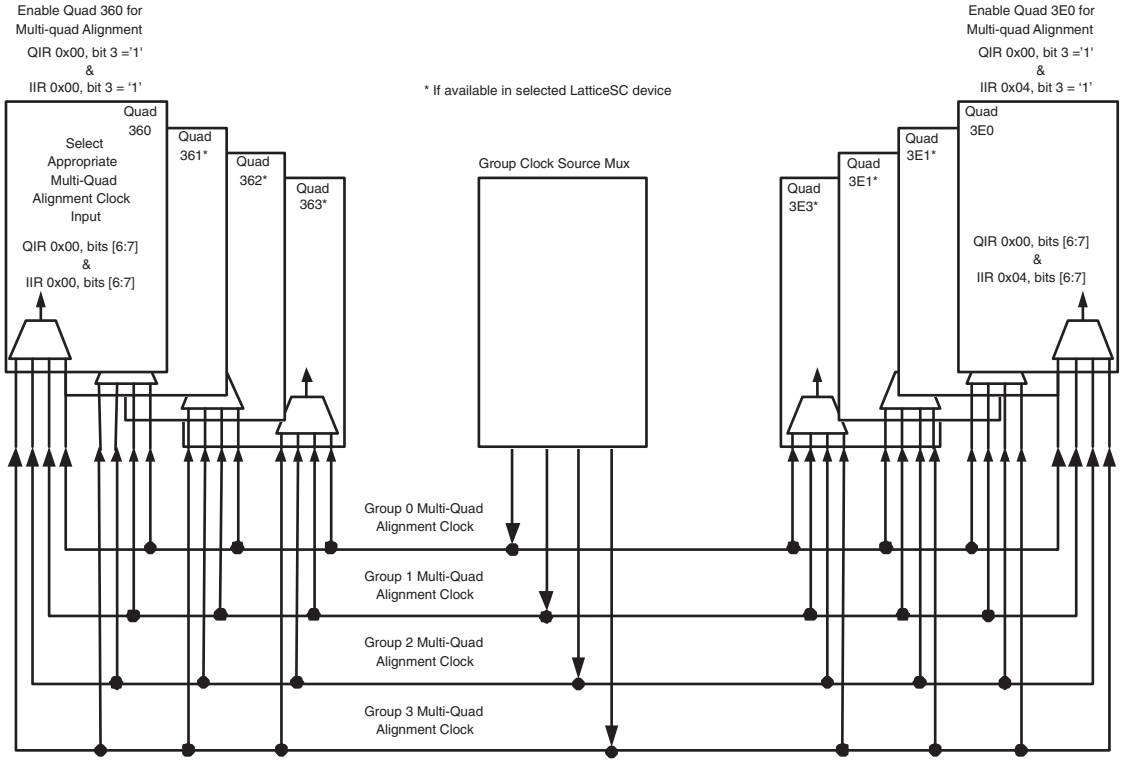
*Table 11-5. Inter-quad Interface Register Multi-quad Alignment Enable Bits*

| Quad | Inter-quad Interface Register Multi-quad Enable Bit |
|---|---|
| 360 | IIR 0x00, bit 3 |
| 361 | IIR 0x01, bit 3 |
| 362 | IIR 0x02, bit 3 |
| 363 | IIR 0x03, bit 3 |
| 3E0 | IIR 0x04, bit 3 |
| 3E1 | IIR 0x05, bit 3 |
| 3E2 | IIR 0x06, bit 3 |
| 3E3 | IIR 0x07, bit 3 |

3. Assignment of the Multi-quad alignment group clocks to the inputs of quads to be aligned together is done by writing to both Inter-quad Interface and Quad Interface Registers.

Quad setup for Multi-quad Alignment and Group Clock input selection is shown in Figure 11-14 below:

***Figure 11-14. Multi-quad Alignment Clock Input Selection***




This information must be registered in both the appropriate Quad Interface Register and in the Inter-quad Interface Register as shown in Table 11-6.

***Table 11-6. Multi-quad Alignment Group Clock Source Register Settings***

| Quad Interface Register Bits | Inter-quad Interface Register Bits | Bit Settings | Input Group Clock Selection |
|---|---|---|---|
| All quads: QIR 0x00, bits [6:7] | Quad 360: IIR 0x00, bits [6:7]<br>Quad 361: IIR 0x01, bits [6:7] | 00 | Group 0 Clock |
| | Quad 362: IIR 0x02, bits [6:7]<br>Quad 363: IIR 0x03, bits [6:7] | 01 | Group 1 Clock |
| | Quad 3E0: IIR 0x04, bits [6:7]<br>Quad 3E1: IIR 0x05, bits [6:7] | 10 | Group 2 Clock |
| | Quad 3E2: IIR 0x06, bits [6:7]<br>Quad 3E3: IIR 0x07, bits [6:7] | 11 | Group 3 Clock |

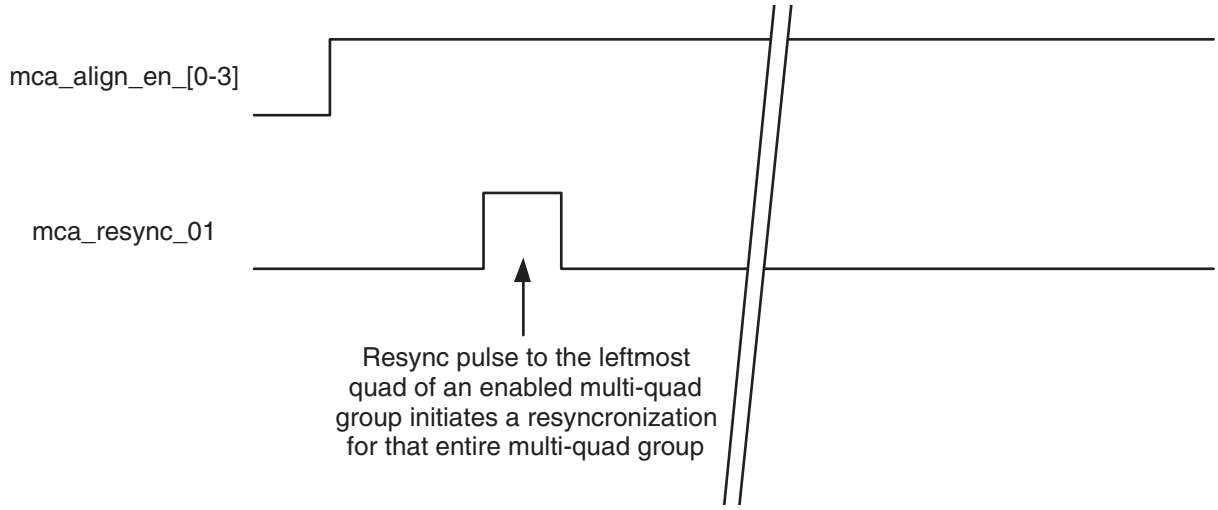
The function of the multi-channel alignment PCS/FPGA interface ports which are available when the “Optional Direct Status & Control Register Access” selection is made in IPexpress during multi-quad alignment is similar to the multi-channel alignment (1 quad) case.

**mca_align_en_[0-3]** - All channels to be aligned must be enabled for multi-channel alignment either through this PCS/FPGA interface port of by writing the appropriate QIR 0x04 bits to a '1'. As this is usually a static setting for a particular application, using the registers for control for this function is acceptable.

**mca_resync_01** - Since only channels 0 and 1 in 0/1 two channel mode or all four channels in four channel mode can be assigned to a multi-quad alignment group, only the mca_resync_01 control signal is relevant for multi-quad alignment (mca_resync_23 can be used to resynchronize channels 2 and 3 independently if desired). Because both multi-channel aligners in both quads must be resynchronized simultaneously, the same FPGA signal must be routed to this input for all quads assigned to the same multi-quad alignment group. Note that although resynchronization for one quad can also be performed by writing to QIR 0x41, bit 7, this could not be done simultaneously for multiple quads and so the mca_resync_01 PCS/FPGA interface port must be used instead.

The relationship of these signals during multi-quad alignment is shown in Figure 11-10.

***Figure 11-15. Multi-channel alignment control and monitoring signals for multi-quad alignment***




## Alignment Between Two Devices

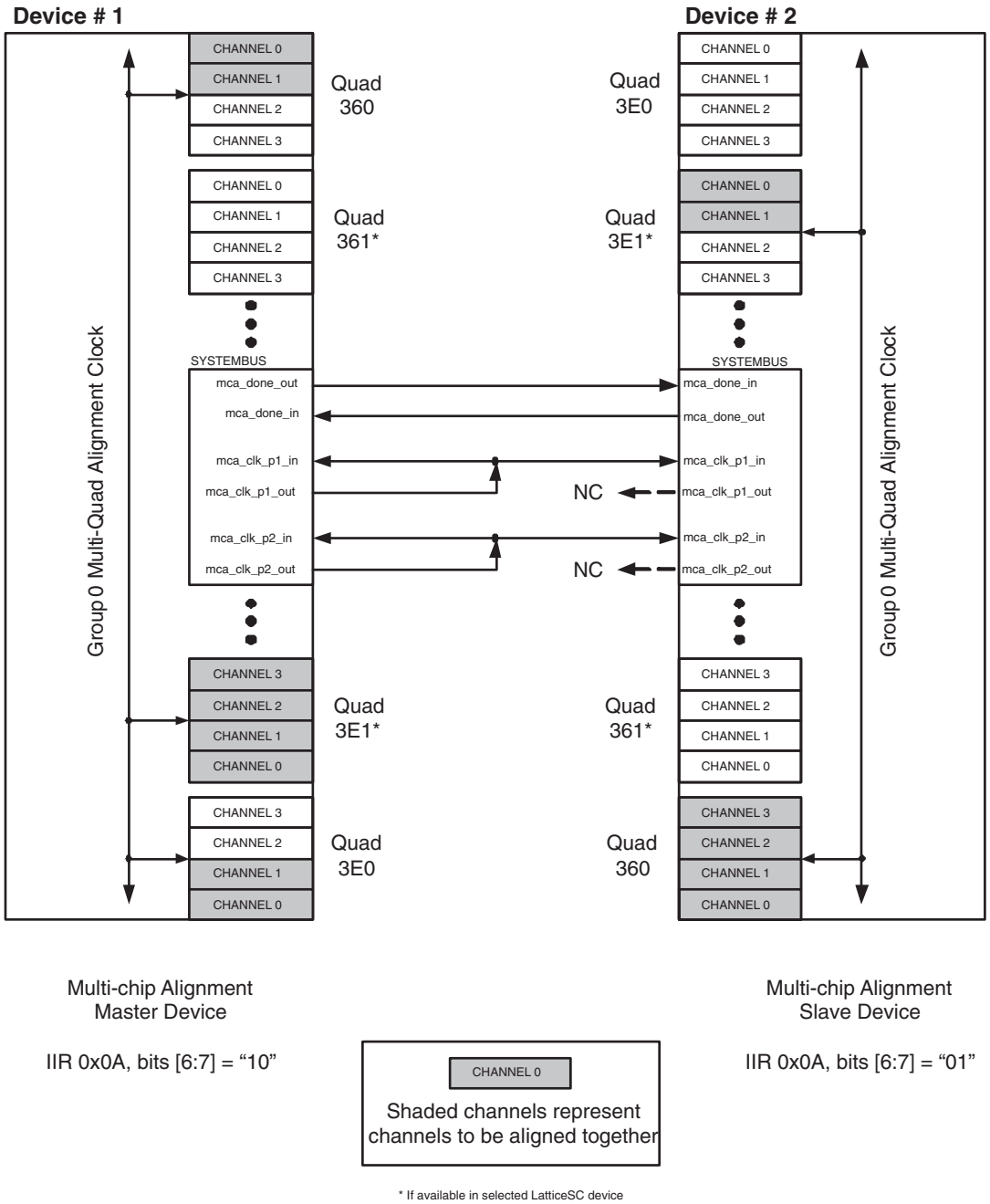
Multi-quad alignment can be extended to include alignment of channels between two LatticeSC devices, allowing up to 64 channel alignment. In multi-chip alignment, the group 0 multi-quad alignment clocks and controls are linked between two devices.This means any channels which are to be aligned across two devices must be assigned to group 0 on both devices.

mca_aligned_01 is not valid when aligning between devices because the signal comes from the multichannel alignment state machine which is disabled in this mode.

For a multi-chip alignment application, one LatticeSC device must be designated as a Master and the other must be designated as the Slave for alignment purposes. This is done by writing to the Inter-quad Interface Register Offset Address 0x0A, bits [6:7] on both devices (bits [6:7] = "10" for Master, bits [6:7] = "01" for Slave).

A total of four signal must be routed between the two devices - two unidirectional DONE signals between the two chips, and two alignment clocks from the Master chip to the Slave chip. These signals are connected to the respective Systembus blocks on both devices as shown in the example in Figure 11-16.

***Figure 11-16. Multi-chip Alignment Example***

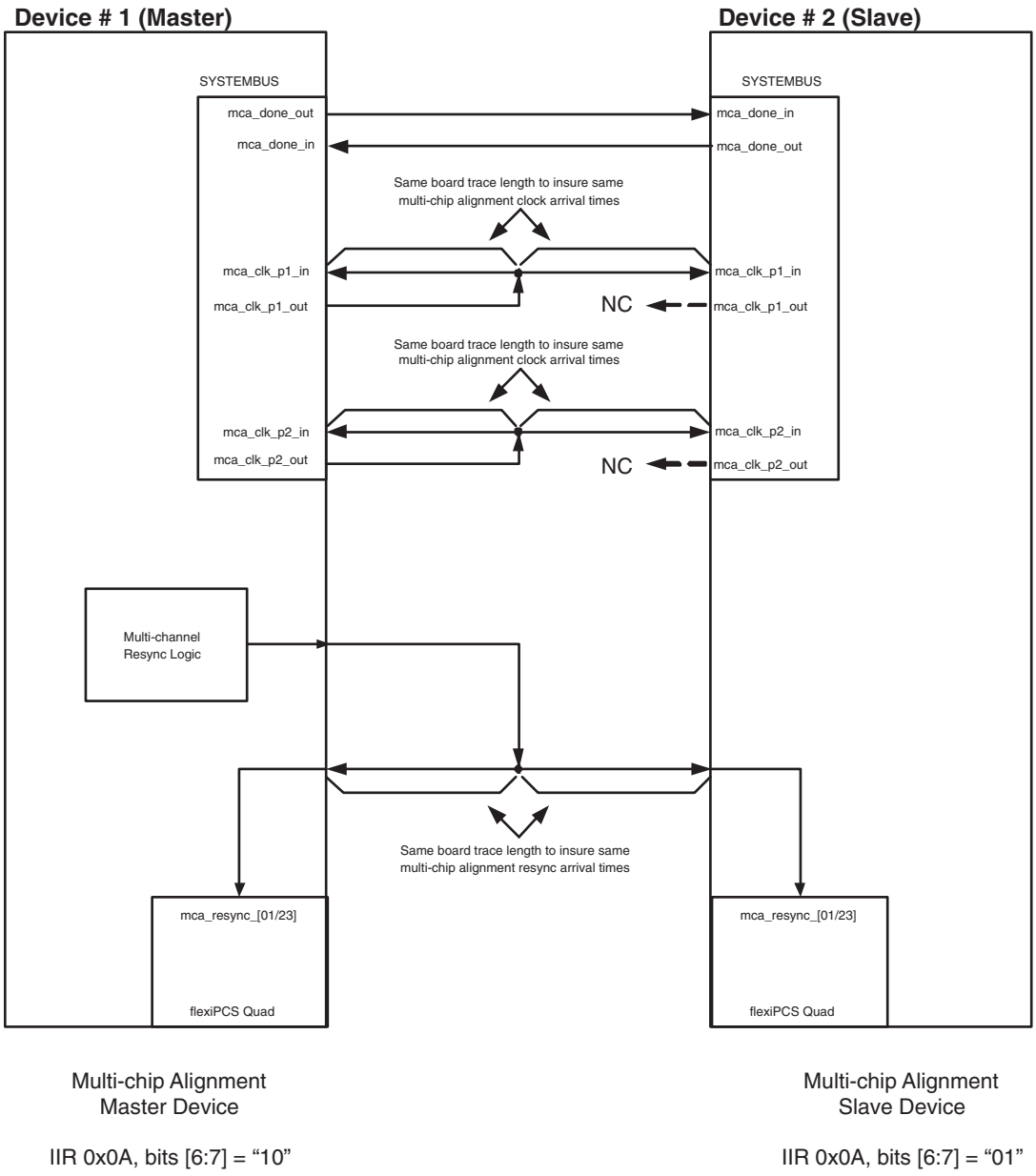



## Synchronization of Aligned Channels Across Two Devices

Care must be taken in the layout of the interconnect signals between two aligned devices to ensure that the quads in both chips receive the clock and resynchronization signals as close to simultaneously as possible. Figure 11-17 shows the proper connection scheme to allow simultaneous reception of the critical alignment inputs. Layout of these signals on the board should be such that the traces to each device are matched as closely as possible.

As long as all of these signals arrive at both devices within one half period of an alignment clock cycle, then data across the two devices will be aligned to within one alignment cycle. A simple resynchronization circuit implemented in FPGA gates in one of the devices is all that is needed to complete alignment.

*Figure 11-17. Multi-chip Alignment Board Layout Example*




Note that the diagram shows the mca_resync_[01/23] signal to the relevant flexiPCS quads. This is an input to quads which make use of the multi-channel aligner (For more information, see specific flexiPCS Mode section in the LatticeSC/M Family Data Sheet). Figure 11-17 shows the correct layout of routing and board traces for this signal to ensure that all aligned quads across chips are synchronized properly.

## Status Interrupt Registers

Some of the status registers associated with multi-channel and multi-quad alignment have an associated status interrupt and status interrupt enable register. Any flexiPCS status interrupt register will generate an interrupt to the Systembus block (if used in the design). For a description of the operation of interrupts with the Systembus, refer to the Systembus section of the LatticeSC/M Family Data Sheet. Operation of the interrupt registers for the flexiPCS is as follows:

User must set the appropriate status interrupt enable bit to a '1'

Whenever the associated status bit goes high, its status interrupt bit will also go high. The status interrupt bit will remain high even if the associated status bit goes low.

The status interrupt bit will remain high until it is read, when it will automatically be cleared. If the associated status bit is still high, then the status interrupt bit will go high again (until read the next time).

Table 11-7 shows a listing of the status registers associated with multi-channel and multi-quad alignment which have a corresponding status interrupt and status interrupt enable bit.

***Table 11-7. Status Interrupt Register Bits***

| Status Register Bit Function | Status Register Bit Address | Status Interrupt Enable Register Bit Address | Status Interrupt Register Bit Address |
|---|---|---|---|
| Multi-channel Alignment State Machine Status (mca_aligned_01 in x1 mode or mca_aligned in x4 mode)<br><br>Channels 0 & 1 (2-channel alignment mode)<br><br>Channels 0,1,2 & 3 (4-channel alignment mode) | QIR 0x82, bit 2 | QIR 0x0C, bit 2 | QIR 0x83, bit 2 |
| Multi-channel Alignment State Machine Status (mca_aligned_23)<br><br>Channels 2 & 3 (2-channel alignment mode)<br><br>Inactive (4-channel alignment mode) | QIR 0x82, bit 3 | QIR 0x0C, bit 3 | QIR 0x83, bit 3 |

# flexiPCS Testing
# LatticeSC/M flexiPCS Family Data Sheet

June 2011 Data Sheet DS1005

## Introduction

The LatticeSC family of devices provides loopback modes for convenient testing of the external SERDES/board interface and the internal PCS/FPGA logic interface. Two near-end loopback modes are provided to loop transmit data back onto the receive data path. The near-end loopback modes are useful for checking the PCS/FPGA interface as well as the embedded flexiPCS logic and any related user FPGA logic. Three far-end loopback modes are also provided to loop received data back onto the transmit data path. The far-end loopback modes are useful for checking the high speed serial SERDES package pin connections as well as the embedded SERDES and/or PCS logic.

In addition, an integrated PRBS generator and PRBS checker for each channel allows easy generation of pseudo-random bit streams for testing. Some loopback modes can be set for each channel independently while others can be set on a per-quad basis only. Loopback modes are controlled by channel or quad specific register bits accessed through the System Bus. PRBS generation and checking is also controlled through the System Bus.

## Near-End Loopback Modes

Two near-end loopback modes allow the user to test the PCS transmit and receive logic as well as user FPGA logic without the need to drive or monitor the device package pins on a LatticeSC device. The user can use these loopback modes to generate their own test patterns and monitor data with user defined logic implemented in the FPGA fabric.

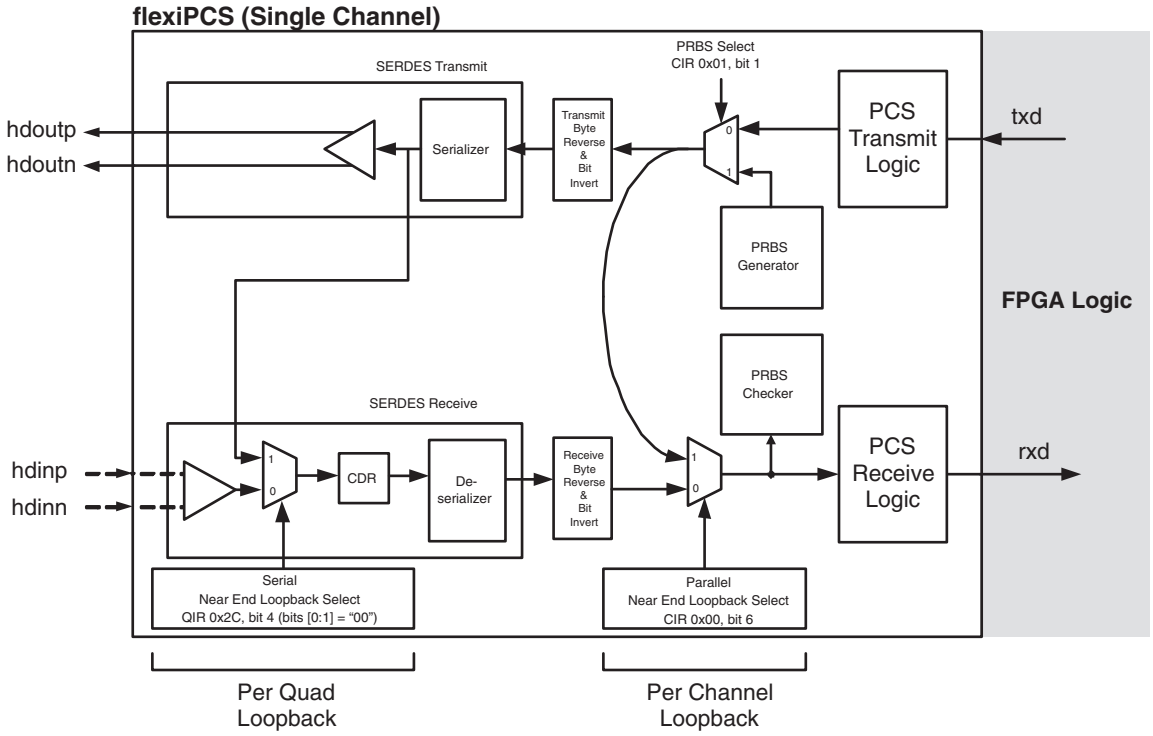
1. **Parallel near-end loopback mode** - Loops parallel transmit data back onto the receive data path without passing through the serializer/deserializer logic in the SERDES. Parallel near-end loopback can be selected for each channel individually by writing the appropriate Channel Interface Register Offset Address 0x00, bit 6 to a '1'. **The PCS transmit and receive paths must be reset after changing this bit.** The PCS logic can be reset by writing a '1' to the Quad Interface Register Offset Address 0x40 (Channel 0 receive = bit 3, channel 0 transmit = bit 7, Channel 1 receive = bit 2, channel 1 transmit = bit 6, Channel 2 receive = bit 1, channel 2 transmit = bit 5, and Channel 3 receive = bit 0, channel 3 transmit = bit 4).

2. **Serial near-end loopback mode** - Loops serial transmit data back onto the receive data path after passing through the transmit buffer but before passing off the chip. Serial near-end loopback is selected on a per-quad basis be writing Quad Interface Register Offset Address 0x2C, bit 4 to '1' while bits [0:1] are set to "00". When serial near-end loopback mode is selected, all channels in the quad are set to that mode. *Note: When using serial near-end loopback, do a SERDES reset.*

Figure 12-1 illustrates the near-end loopback mode for a single channel.

© 2011 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

*Figure 12-1. Near-End Loopback Modes (Single Channel)*




## Far-End Loopback Modes

Four far-end loopback modes allow the user to test connections to the LatticeSC device package pins and various levels of embedded SERDES and PCS logic.

1. **PCS Parallel far-end 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 far-end loopback can be selected for each channel individually by writing the appropriate Channel Interface Register Offset Address 0x00, bit 5 to a '1'.

   When in PCS Parallel far-end loopback mode, care must be taken when the input receive data is not exactly synchronous to the reference clock supplied to the LatticeSC flexiPCS quad. This can be handled in different ways depending on the protocol mode setting of the flexiPCS quad.

   When using a protocol mode that does not include the Clock Tolerance Compensation function (SONET, Generic 8b10b, 8-bit SERDES Only and 10-bit SERDES Only modes), the transmit path is clocked by the rclk_[0-3] clocks. The transmit clocks, tclk_[0-3] must be synchronous (0 ppm difference) to the rclk [0-3] (ideally, they are tied to the same clock). These clocks should be driven from one of the recovered receive clocks: rxa_pclk, rxb_pclk, or rx_[0-3]_sclk.

   When using a protocol mode that uses the Clock Tolerance Compensation function (Gigabit Ethernet, Fibre Channel, XAUI, PCI Express, and Serial RapidIO modes), the receive data must be synchronous (0 ppm) to the reference clock to the LatticeSC flexiPCS quad for random data (data which does not contain Skip words) when in PCS Parallel far-end loopback mode. A tolerance of up to 300 ppm between the receive data and the reference clock can be compensated for by the Clock Tolerance Compensation logic provided that the proper Skip words for the chosen protocol selected are present in the data stream so that CTC insertion and deletion

can be performed. If this cannot be guaranteed and the receive data is not guaranteed to be perfectly synchronous to the reference clock supplied to the flexiPCS quad, then the Generic 8b10b mode should be selected for PCS Parallel far-end loopback testing (see non-CTC protocol mode rules above). When performing PCS Parallel far-end loopback testing in a CTC mode, the rclk_[0-3] and tclk_[0-3] are not used and so these clocks do not have to be toggling for loopback to work.

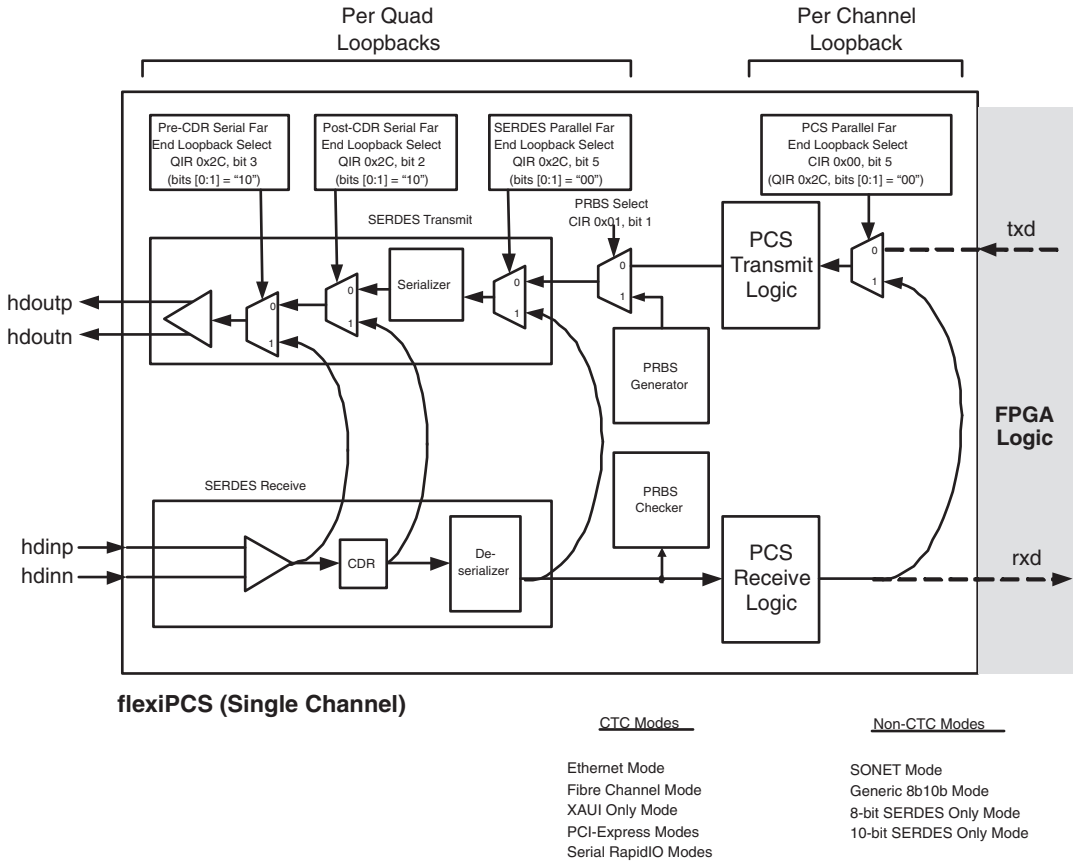
2. **SERDES Parallel far-end loopback mode** - Loops parallel receive data back onto the transmit data path without passing through the PCS logic. Input serial data at the SERDES **hdin** package pins passes through the SERDES receive logic where it is converted to parallel data, and is looped back immediately to the SERDES transmitter where it is reconverted back to serial data and sent out onto the hdout SERDES package pins. SERDES Parallel far-end loopback can be selected by writing the Quad Interface Register Offset Address 0x2C, bit 5 to a '1' while bits [0:1] are set to "00". When SERDES PCS far-end loopback mode is selected, all channels in the quad are set to that mode. In SERDES Parallel far-end loopback mode, the receive data must be synchronous (0ppm) to the reference clock of the LatticeSC flexiPCS quad.

3. **Post-CDR Serial far-end loopback mode** - Loops serial receive data back onto the transmit data path after passing through the receive CDR but before conversion to parallel data. Post-CDR Serial far-end loopback is selected on a per-quad basis by writing Quad Interface Register Offset Address 0x2C, bit 2 to "1" while bits [0:1] are set to "10". When Pre-CDR serial near end loopback mode is selected, all channels in the quad are set to that mode. Note: this mode introduces some noise into the data.

4. **Pre-CDR Serial far-end loopback mode** - Loops serial receive data back onto the transmit data path without passing through the receive CDR. This loopback mode is useful as a basic test of on-board pin connections and the SERDES input and output buffers. Pre-CDR Serial far-end loopback is selected on a per-quad basis by writing Quad Interface Register Offset Address 0x2C, bit 3 to '1' while bits [0:1] are set to "10". When Pre-CDR Serial far-end loopback mode is selected, all channels in the quad are set to that mode.

Figure 12-2 illustrates the far-end loopback mode for a single channel.

***Figure 12-2. Far-End Loopback Modes (Single Channel)***




## PRBS Generator and Checker

The PRBS generator can be used to generate a pseudo-random 8-bit or 10-bit wide (depending on the PCS mode chosen) sequence using one of two user chosen PRBS polynomials. The PRBS generator and checker for a particular flexiPCS channel is enabled by writing the appropriate Channel Interface Register Offset Address 0x01, bit 1 to '1'.

The PRBS polynomial is chosen per quad by writing the appropriate Quad Interface Register Offset Address 0x19, bit 6. When bit 6 is set to '0', (default), the PRBS polynomial is:

$$X^7 + X^6 + 1$$

When bit 6 is set to '1', the PRBS polynomial is:

$$X^{31} + X^{28} + 1$$

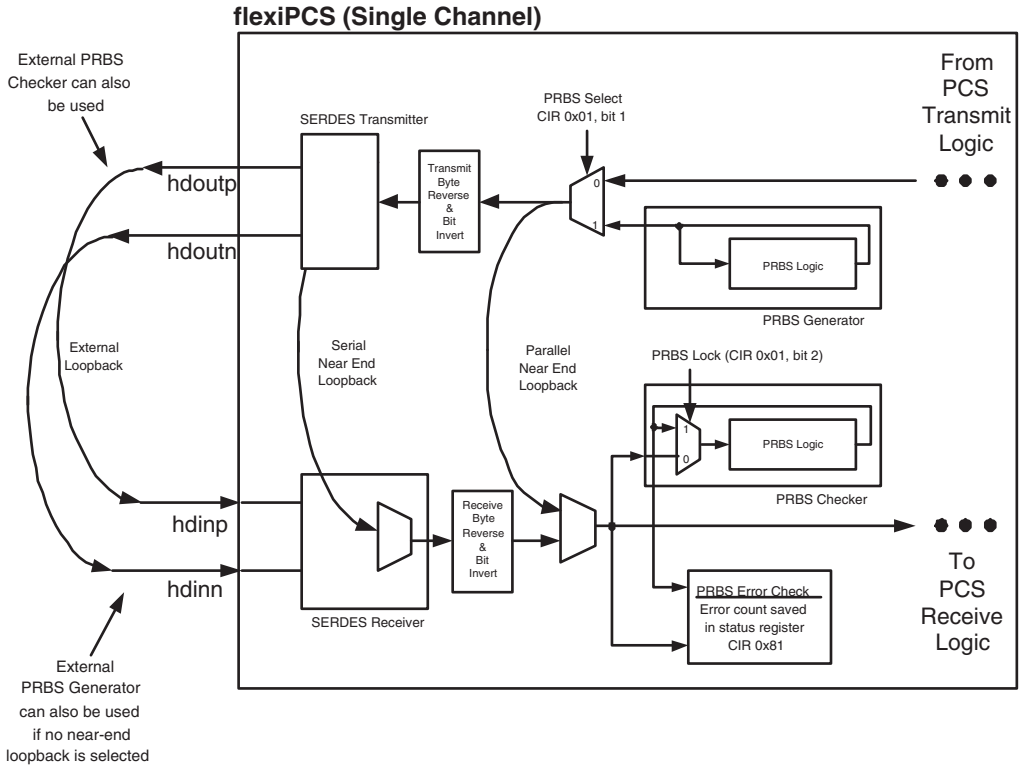
Note: The PRBS polynomial $X^7 + X^6 + 1$ is supported over the full operating range of the PCS. The PRBS polynomial $X^{31} + X^{28} + 1$ is supported up to 2.7 Gbps.

The chosen polynomial for a quad applies for all channels in that quad. The PRBS checker is essentially the same as the generator with a minor difference. The PRBS checker can be set to a lock mode (by writing the appropriate Channel Interface Register Offset Address 0x01, bit 2 to '1') as shown in Figure 12-3. When bit 2 is set to '0' (default), the PRBS checker input is from the incoming data stream. The checker continuously compares each input data word with the expected computed data (computed using the previous cycle's value as a seed) and increments the PRBS error counter when they don't match.

When bit 2 is set to '1', the checker takes its input from its own registered output. If the receive PRBS has been in unlock mode long enough to get in sync with the transmit PRBS, when the lock mode is set, the receive PRBS then functions as a second PRBS generator that remains in sync with the transmit PRBS generator. In this mode, the checker self-synchronizes to the incoming PRBS pattern.

Receive data passes unaltered to the PCS receive logic when the PRBS generator/checker is enabled if near-end loopback is not selected. This means receive data can be processed normally even if the transmit side is sending PRBS data provided no near-end loopback is being performed. When performing near-end loopback, the PRBS generated data is fed to the PCS receive logic. Note that the PRBS data can be monitored on the **hdout** SERDES output pins even when in a near-end loopback mode.

***Figure 12-3. PRBS Generator and Checker (Single Channel)***




A PRBS error counter is provided in a status register for each channel. The value of this counter can be read from Channel Interface Register Offset Address 0x81, bits [0:7].

Note that the transmit side byte reversal (MSB, LSB bit order reversed) and bitwise inversion (all 0's converted to 1's and vice versa) functions are provided after the PRBS Generator and before Parallel Near-end loopback. Likewise, the receive side byte reversal and bitwise inversion functions are provided after the Parallel Near-end loopback and before the PRBS Checker. It is important that the byte reversal functions for both the transmit and receive paths are turned off or serial data delays could cause the PRBS Checker to report errors. The byte reversal function is set by writing the appropriate Channel Interface Register Offset Address 0x00, bit 4 (transmit) and bit 6 (receive). A value of '0' means no byte reversal.

The bitwise inversion function can be either on or off as long as it is the same for both transmit and receive paths. One way to force PRBS errors is to set the bitwise inversion functions to different values for the transmit and receive paths. The bitwise inversion function is set by writing the appropriate Channel Interface Register Offset Address 0x00, bit 5 (transmit) and bit 7 (receive). A value of '0' means no bitwise inversion.

# Memory Map
# LatticeSC/M flexiPCS Family Data Sheet

June 2011 Data Sheet DS1005

## LatticeSC flexiPCS Memory Map

Control and status monitoring of the flexiPCS logic after configuration is done via the embedded System Bus Interface (see the **System Bus** section for more details on the operation of the System Bus). Depending on the specific option, a register may affect operation of multiple quads, one quad or just one channel within a quad. Registers dedicated to multiple quads are listed in Table 13-1, Multi-quad Interface Register Map. Registers dedicated to a single quad are listed in Table 13-2, Quad Interface Register Map. Registers dedicated to a single channel are listed in Table 13-3, Channel Interface Register Map.

Multi-quad Interface, Quad Interface and Channel Interface Registers are listed by Offset Address. Each Multi-quad Interface Register, Quad Interface Register, and Channel Interface Register has a unique 18-bit address determined by the specific quad(s) or channel targeted. The addressing of Multi-quad Interface Registers, Quad Interface Registers, and Channel Interface Registers is shown Figure 13-1. Note that Figure 13-1 provides a list of addresses for a LatticeSC device with the maximum of 8 quads. A particular LatticeSC device may have fewer than 8 quads depending on the size of array and package chosen. For devices with fewer than the maximum of 8 quads, only registers dedicated to available quads or channels can be accessed.

As shown in Figure 13-1, Channel Interface Registers can be written in one of three ways. One way is to write to one specific register corresponding to one specific channel. The other two methods involve writing to multiple channel registers with one write operation. This is referred to as a Broadcast write. A Broadcast write is useful when multiple channel registers are to be written to the same value. The first method of Broadcast writing involves writing to all four channel register in a quad at one time. A second method of Broadcast writing involves writing to all channel registers for all quads on the left or right side of the device at one time. Therefore, a specific Channel Interface Register may be accessed through any one of three channel address, depending on the number of channel registers being written to at one time.

A broadcast write to multiple quads cannot be used to power on SERDES in any device-package combination that does not have all SERDES quads bonded because of excess power on the board and jitter is also affected.

© 2011 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

*Figure 13-1. flexiPCS Multi-Quad, Quad and Channel Interface Register Map Base Addressing*

*Table 13-1. Multi-Quad Alignment Interface Register Map*

| Multi-Quad Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| **Control Registers** | | | |
| 00 | [0:2] | Reserved | |
| | [3] | Cascade enable for quad 360.<br>1 = Enable inter quad or cascade alignment.<br>0 = Disable cascade alignment (default). | 0 |
| | [4:5] | Reserved | |
| | [6:7] | Select one of four group clocks as cascade clock for quad 360.<br>00 = Group 0 clock.<br>01 = Group 1 clock.<br>10 = Group 2 clock.<br>11 = Group 3 clock. | |
| 01 | [0:2] | Reserved | |
| | [3] | Cascade enable for quad 361.<br>1 = Enable inter quad or cascade alignment.<br>0 = Disable cascade alignment (default). | 0 |
| | [4:5] | Reserved | |
| | [6:7] | Select one of four group clocks as cascade clock for quad 361.<br>00 = Group 0 clock.<br>01 = Group 1 clock.<br>10 = Group 2 clock.<br>11 = Group 3 clock. | |
| 02 | [0:2] | Reserved | |
| | [3] | Cascade enable for quad 362.<br>1 = Enable inter quad or cascade alignment.<br>0 = Disable cascade alignment (default). | 0 |
| | [4:5] | Reserved | |
| | [6:7] | Select one of four group clocks as cascade clock for quad 362.<br>00 = Group 0 clock.<br>01 = Group 1 clock.<br>10 = Group 2 clock.<br>11 = Group 3 clock. | |
| 03 | [0:2] | Reserved | |
| | [3] | Cascade enable for quad 363.<br>1 = Enable inter quad or cascade alignment.<br>0 = Disable cascade alignment (default). | 0 |
| | [4:5] | Reserved | |
| | [6:7] | Select one of four group clocks as cascade clock for quad 363.<br>00 = Group 0 clock.<br>01 = Group 1 clock.<br>10 = Group 2 clock.<br>11 = Group 3 clock. | |

*Table 13-1. Multi-Quad Alignment Interface Register Map (Continued)*

| Multi-Quad Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| 04 | [0:2] | Reserved | |
| | [3] | Cascade enable for quad 3E0.<br>1 = Enable inter quad or cascade alignment.<br>0 = Disable cascade alignment (default). | 0 |
| | [4:5] | Reserved | |
| | [6:7] | Select one of four group clocks as cascade clock for quad 3E0.<br>00 = Group 0 clock.<br>01 = Group 1 clock.<br>10 = Group 2 clock.<br>11 = Group 3 clock. | |
| 05 | [0:2] | Reserved | |
| | [3] | Cascade enable for quad 3E1.<br>1 = Enable inter quad or cascade alignment.<br>0 = Disable cascade alignment (default). | 0 |
| | [4:5] | Reserved | |
| | [6:7] | Select one of four group clocks as cascade clock for quad 3E1.<br>00 = Group 0 clock.<br>01 = Group 1 clock.<br>10 = Group 2 clock.<br>11 = Group 3 clock. | |
| 06 | [0:2] | Reserved | |
| | [3] | Cascade enable for quad 3E2.<br>1 = Enable inter quad or cascade alignment.<br>0 = Disable cascade alignment (default). | 0 |
| | [4:5] | Reserved | |
| | [6:7] | Select one of four group clocks as cascade clock for quad 3E2.<br>00 = Group 0 clock.<br>01 = Group 1 clock.<br>10 = Group 2 clock.<br>11 = Group 3 clock. | |
| 07 | [0:2] | Reserved | 0 |
| | [3] | Cascade enable for quad 3E3.<br>1 = Enable inter quad or cascade alignment.<br>0 = Disable cascade alignment (default). | |
| | [4:5] | Reserved | |
| | [6:7] | Select one of four group clocks as cascade clock for quad 3E3.<br>00 = Group 0 clock.<br>01 = Group 1 clock.<br>10 = Group 2 clock.<br>11 = Group 3 clock. | |

*Table 13-1. Multi-Quad Alignment Interface Register Map (Continued)*

| Multi-Quad Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| 08 | [0:1] | Reserved | |
| | [2:4] | Select one of the 8 quad clocks to be used by quads assigned to group 1.<br>000 = Clock from quad 360<br>001 = Clock from quad 361<br>010 = Clock from quad 362<br>011 = Clock from quad 363<br>100 = Clock from quad 3E0<br>101 = Clock from quad 3E1<br>110 = Clock from quad 3E2<br>111 = Clock from quad 3E3 | 0 |
| | [5:7] | Select one of the 8 quad clocks to be used by quads assigned to group 0.<br>000 = Clock from quad 360<br>001 = Clock from quad 361<br>010 = Clock from quad 362<br>011 = Clock from quad 363<br>100 = Clock from quad 3E0<br>101 = Clock from quad 3E1<br>110 = Clock from quad 3E2<br>111 = Clock from quad 3E3 | |
| 09 | [0:1] | Reserved | |
| | [2:4] | Select one of the 8 quad clocks to be used by quads assigned to group 3.<br>000 = Clock from quad 360<br>001 = Clock from quad 361<br>010 = Clock from quad 362<br>011 = Clock from quad 363<br>100 = Clock from quad 3E0<br>101 = Clock from quad 3E1<br>110 = Clock from quad 3E2<br>111 = Clock from quad 3E3 | 0 |
| | [5:7] | Select one of the 8 quad clocks to be used by quads assigned to group 2.<br>000 = Clock from quad 360<br>001 = Clock from quad 361<br>010 = Clock from quad 362<br>011 = Clock from quad 363<br>100 = Clock from quad 3E0<br>101 = Clock from quad 3E1<br>110 = Clock from quad 3E2<br>111 = Clock from quad 3E3 | |
| 0A | [0:5] | Reserved | |
| | [6:7] | set to:<br>2'b10 for 2 chip alignment (master)<br>2'b01 for 2 chip alignment (slave)<br>2'b00 for single chip operation | |
| **Status Registers** | | | |
| 0B | [0:4] | Per quad interrupt for quads 360, 361, 362, and 363. | 0 |
| | [5:7] | Per quad interrupt for quads 3E0, 3E1, 3E2, and 3E3. | |

*Table 13-2. Quad Interface Register Map*

| Quad Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| **Control Registers** | | | |
| ***Per Channel General Registers*** | | | |
| 00 | [0:2] | Reserved | |
| | [3] | 1 = Enable inter quad or cascade alignment.<br>0 = Disable cascade alignment (default). | 0 |
| | [4:5] | Clock select for selecting the quad_clk output to FPGA.<br>00 = Channel 0 recovered clock.<br>01 = Channel 1 recovered clock.<br>10 = Channel 2 recovered clock.<br>11 = Channel 3 recovered clock. | |
| | [6:7] | Select one of four group clocks as cascade clock.<br>00 = Group 0 clock.<br>01 = Group 1 clock.<br>10 = Group 2 clock.<br>11 = Group 3 clock. | |
| 01 | [0:1] | Select clock for channel 3.<br>00 = RXCLK 0.<br>01 = RXCLK 1.<br>10 = RXCLK 2.<br>11 = RXCLK 3. | E4 |
| | [2:3] | Select clock for channel 2.<br>00 = RXCLK 0.<br>01 = RXCLK 1.<br>10 = RXCLK 2.<br>11 = RXCLK 3. | |
| | [4:5 | Select clock for channel 1.<br>00 = RXCLK 0.<br>01 = RXCLK 1.<br>10 = RXCLK 2.<br>11 = RXCLK 3. | |
| | [6:7] | Select clock for channel 0.<br>00 = RXCLK 0.<br>01 = RXCLK 1.<br>10 = RXCLK 2.<br>11 = RXCLK 3. | |

*Table 13-2. Quad Interface Register Map (Continued)*

| Quad Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| 02 | [0] | Reserved | |
| | [1] | 1 = Drive this quad's channel 0 tck clock to the fleximac through the characterization outputs.<br>0 = Drive the previous enabled quad's tck to the fleximac. | 0 |
| | [2:3] | Select one of 4 channels' transmit clocks as primary clock to FPGA.<br>00 = Channel 0 sysclk.<br>01= Channel 1 sysclk.<br>10 = Channel 2 sysclk.<br>11 = Channel 3 sysclk. | |
| | [4:5] | Select one of 4 channels' recovered clocks as primary clock, rxa_pclk, to FPGA.<br>00 = Channel 0 recovered clock.<br>01 = Channel 1 recovered clock.<br>10 = Channel 2 recovered clock.<br>11 = Channel 3 recovered clock. | |
| | [6:7] | Select one of 4 channels' recovered clocks as primary clock, rxb_pclk, to FPGA.<br>00 = Channel 0 recovered clock.<br>01 = Channel 1 recovered clock.<br>10 = Channel 2 recovered clock.<br>11 = Channel 3 recovered clock. | |
| 03 | [0:5] | Reserved | |
| | [6] | 1 = Enable SERDES characterization mode.<br>0 = Disable SERDES characterization mode. | 0 |
| | [7] | Forces interrupt on all interrupt registers.<br>Test for interrupt and clear on read logic. | |
| **Per Quad Align PCS Registers** | | | |
| 04 | [0:1] | Reserved | |
| | [2] | 1 = Disable channel alignment state machine for channels 2 and 3.<br>0 = Enable channel alignment state machine for channels 2 and 3. | 0 |
| | [3] | 1 = Disable channel alignment state machine for channels 0 and 1.<br>0 = Enable channel alignment state machine for channels 0 and 1. | |
| | [4] | 1 = Enable alignment for channel 3.<br>0 = Do not enable alignment for channel 3. | |
| | [5] | 1 = Enable alignment for channel 2.<br>0 = Do not enable alignment for channel 2. | |
| | [6] | 1 = Enable alignment for channel 1.<br>0 = Do not enable alignment for channel 1. | |
| | [7] | 1 = Enable alignment for channel 0.<br>0 = Do not enable alignment for channel 0. | |
| 05 | [0:2] | Reserved | |
| | [3:7] | FIFO latency control. | 1F |
| 06 | [0:1] | Reserved | |
| | [2:7] | Alignment FIFO high water mark (max deskew). | 05 |
| 07 | [0:7] | Lower bits of user defined align pattern mask. | FF |
| 08 | [0:7] | Lower bits of user defined align pattern 'a'. | 7C |
| 09 | [0:7] | Lower bits of user defined align pattern 'b'. | 7C |

*Table 13-2. Quad Interface Register Map (Continued)*

| Quad Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| 0A | [0:1] | Reserved | |
| | [2:3] | Upper bits of user defined align pattern mask.<br>[2] = disparity_error.<br>[3] = K control. | 15 |
| | [4:5] | Upper bits of user defined align pattern 'a'.<br>[4] = disparity_error.<br>[5] = K control. | |
| | [6:7] | Upper bits of user defined align pattern 'b'.<br>[6] = disparity_error.<br>[7] = K control. | |
| 0B | [0:7] | Reserved | |
| 0C | [0] | 1 = Enable interrupt for channel 01 alignment status when it goes low (out of align).<br>0 = Disable interrupt for channel 01 alignment status when it goes low (out of align).<br>Interrupt for channel 01 alignment status when it goes low (out of align) is QIR 0x83[0]. | |
| | [1] | 1 = Enable interrupt for channel 23 alignment status when it goes low (out of align).<br>0 = Disable interrupt for channel 23 alignment status when it goes low (out of align).<br>Interrupt for channel 23 alignment when it goes low is QIR 0x83[1]. | |
| | [2] | 1 = Enable interrupt for channel 01 alignment status.<br>0 = Disable interrupt for channel 01 alignment status.<br>Interrupt for channel 01 alignment is QIR 0x83[2]. | 0 |
| | [3] | 1 = Enable interrupt for channel 23 alignment status.<br>0 = Disable interrupt for channel 23 alignment status.<br>Interrupt for channel 23 alignment status is QIR 0x83[3]. | |
| | [4] | 1 = Enable interrupt for channel 01 failed alignment.<br>0 = Disable interrupt for channel 01 failed alignment.<br>Interrupt for channel 01 failed alignment is QIR 0x83[4]. | |
| | [5] | 1 = Enable interrupt for channel 23 failed alignment.<br>0 = Disable interrupt for channel 23 failed alignment.<br>Interrupt for channel 23 failed alignment is QIR 0x83[5]. | |
| | [6] | 1 = Enable interrupt for channel 01 aligned.<br>0 = Disable interrupt for channel 01 aligned.<br>Interrupt for channel 01 aligned is QIR 0x83[6]. | |
| | [7] | 1 = Enable interrupt for channel 23 aligned.<br>0 = Disable interrupt for channel 23 aligned.<br>Interrupt for channel 23 aligned is QIR 0x83[7]. | |
| **Per Quad Packet PCS Registers** | | | |
| 0D | [0:3] | Clock compensation FIFO high water mark. Mean is 4'b1000. | 97 |
| | [4:7] | Clock compensation FIFO low water mark. Mean is 4'b1000. | |
| 0E | [0:3] | Reserved | |
| | [4] | 1 = enable two character skip matching (using match 4,3). | 0B |
| | [5] | 1 = enable four character skip matching (using match 4, 3, 2, 1). | |
| | [6:7] | CTC Minimum IPG insertion/deletion multiplier factor | |
| 0F | [0:7] | Lower bits of user defined clock compensator skip pattern 1. | 0 |
| 10 | [0:7] | Lower bits of user defined clock compensator skip pattern 2. | 0 |
| 11 | [0:7] | Lower bits of user defined clock compensator skip pattern 3. | BC |
| 12 | [0:7] | Lower bits of user defined clock compensator skip pattern 4. | 50 |

*Table 13-2. Quad Interface Register Map (Continued)*

| Quad Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| 13 | [0:1] | Upper bits of user defined clock compensator skip pattern 1.<br>[0] = disparity_error.<br>[1] = K control. | 05 |
| | [2:3] | Upper bits of user defined clock compensator skip pattern 2.<br>[2] = disparity_error.<br>[3] = K control. | |
| | [4:5] | Upper bits of user defined clock compensator skip pattern 3.<br>[4] = disparity_error.<br>[5] = K control. | |
| | [6:7] | Upper bits of user defined clock compensator skip pattern 4.<br>[6] = disparity_error.<br>[7] = K control. | |
| 14 | [0:7] | udf_comma_mask[7:0]. | 7F |
| 15 | [0:7] | Lower bits of user defined comma character 'a'. | 03 |
| 16 | [0:7] | Lower bits of user defined comma character 'b'. | 7C |
| 17 | [0:1] | Reserved | |
| | [2:3] | Upper bits of user defined comma mask. | 0 |
| | [4:5] | Upper bits of user defined comma character 'a'. | |
| | [6:7] | Upper bits of user defined comma character 'b'. | |
| 18 | [0] | Reserved | |
| | [1:2] | 0 0 = other modes<br>0 1 = SONET mode<br>1 0 = Reserved<br>1 1 = 8-bit SERDES only mode | 0 |
| | [3] | 1 = Selects Generic 8b10b Mode.<br>0 = Other Modes. | |
| | [4] | 1 = Selects Fibre Channel mode.<br>0 = Other Modes. | |
| | [5] | 1 = PCI Express mode.<br>0 = Other modes. | |
| | [6] | 1 = Selects RapidIO mode.<br>0 = Other modes. | |
| | [7] | 1 = Selects XAUI Mode.<br>0 = Other Modes. | |

*Table 13-2. Quad Interface Register Map (Continued)*

| Quad Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| 19 | [0] | 1 = Disable word aligner lock from LSM.<br>0 = Allow word aligner lock from LSM. | 0 |
| | [1] | 1 = Select four channel alignment.<br>0 = Select independent alignment. | |
| | [2] | 1 = Channels 2 and 3 alignment. | |
| | [3] | 1 = Channels 0 and 1 alignment. | |
| | [4] | 1 = Enable 2:1 gearing for transmit path on all channels.<br>0 = Disable 2:1 gearing for transmit path on all channels (no gearing). | |
| | [5] | 1 = Enable 2:1 gearing for receive path on all channels.<br>0 = Disable 2:1 gearing for receive path on all channels (no gearing). | |
| | [6] | 0 = $X^7 + X^6 + 1$.<br>1 = $X^{31} + X^{28} + 1$. | |
| | [7] | Selects the polynomial for the PCI Express scrambler.<br>1 = Selects Draft 1.0 polynomial $X^{16} + X^{15} + X^{13} + X^4 + 1$<br>0 = Selects Draft 1.0a polynomial $X^{16} + X^5 + X^4 + X^3 + 1$ | |
| 1A | [0:7] | Reserved | |
| 1B | [0:7] | Reserved | |
| 1C | [0] | 1 = Enable interrupt for link status channel 0 when it goes low (out of sync).<br>0 = Disable interrupt for link status channel 0 when it goes low (out of sync).<br>Interrupt for link status channel 0 when it goes low is QIR 0x85[0]. | |
| | [1] | 1 = Enable interrupt for link status channel 1 when it goes low (out of sync).<br>0 = Disable interrupt for link status channel 1 when it goes low (out of sync).<br>Interrupt for link status channel 1 when it goes low is QIR 0x85[1]. | |
| | [2] | 1 = Enable interrupt for link status channel 2 when it goes low (out of sync).<br>0 = Disable interrupt for link status channel 2 when it goes low (out of sync).<br>Interrupt for link status channel 2 when it goes low is QIR 0x85[2]. | |
| | [3] | 1 = Enable interrupt for link status channel 3 when it goes low (out of sync).<br>0 = Disable interrupt for link status channel 3 when it goes low (out of sync).<br>Interrupt for link status channel 3 when it goes low is QIR 0x85[3]. | |
| | [4] | 1 = Enable interrupt for link status channel 0.<br>0 = Disable interrupt for link status channel 0.<br>Interrupt status for link status channel 0 is QIR 0x85[4]. | 0 |
| | [5] | 1 = Enable interrupt for link status channel 1.<br>0 = Disable interrupt for link status channel 1.<br>Interrupt status for link status channel 1 is QIR 0x85[5]. | |
| | [6] | 1 = Enable interrupt for link status channel 2.<br>0 = Disable interrupt for link status channel 2.<br>Interrupt status for link status channel 2 is QIR 0x85[6]. | |
| | [7] | 1 = Enable interrupt for link status channel 3.<br>0 = Disable interrupt for link status channel 3.<br>Interrupt status for link status channel 3 is QIR 0x85[7]. | |

*Table 13-2. Quad Interface Register Map (Continued)*

| Quad Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| **Per Quad CRC PCS Registers** | | | |
| 1D | [0] | Reserved | |
| | [1] | 1 = All ones (FFFF_FFFF).<br>0 = All zeros (0000_0000). | 0 |
| | [2:3] | 00 = Bypass CRC.<br>01 = GbE.<br>10 = Fibre Channel.<br>11 = User mode. | |
| | [4] | 1 = Invert the data bytes as they are fed into the CRC logic.<br>0 = Do not invert the data bytes as they are fed into the CRC logic. | |
| | [5] | 1 = Swap over the bit order of the data bytes as they are fed into the CRC logic.<br>0 = Do not swap over the bit order of the data bytes as they are fed into the CRC logic. | |
| | [6] | 1 = Invert the CRC bytes as they are transmitted.<br>0 = Do not invert the CRC bytes as they are transmitted. | |
| | [7] | 1 = Swap over the bit order of the CRC bytes as they are transmitted.<br>0 = Do not swap over the bit order of the CRC bytes as they are transmitted. | |
| 1E | [0:1] | Reserved | |
| | [2] | 1 = Invert the data bytes as they are fed into the CRC logic.<br>0 = Do not invert the data bytes as they are fed into the CRC logic. | 0 |
| | [3] | 1 = Swap over the bit order of the data bytes as they are fed into the CRC logic.<br>0 = Do not swap over the bit order of the data bytes as they are fed into the CRC logic. | |
| | [4] | 1 = Invert the CRC bytes as they are checked.<br>0 = Do not invert the CRC bytes as they are checked. | |
| | [5] | 1 = Swap over the bit order of the CRC bytes as they are checked.<br>0 = Do not swap over the bit order of the CRC bytes as they are checked. | |
| | [6] | 1 = Received CRC byte order is [7:0] first, [15:8] second, [23:16] third, [31:24] fourth.<br>0 = Received CRC byte order is [31:24] first, [23:16] second, [15:8] third, [7:0] fourth. | |
| | [7] | 1 = SOP and EOP ordered sets are 1 character long.<br>0 = SOP and EOP ordered sets are 2 or 4 characters long. | |
| 1F | [0:7] | Bottom 8 bits of user programmable SOP character mask. | 0 |
| 20 | [0:7] | Bottom 8 bits of user programmable SOP character 0. | 0 |
| 21 | [0:7] | Bottom 8 bits of user programmable SOP character 1. | 0 |
| 22 | [0:4] | Reserved | |
| | [5] | Top bit of user programmable SOP character mask (K control bit). | 0 |
| | [6] | Top bit of user programmable SOP character 1 (K control bit). | |
| | [7] | Top bit of user programmable SOP character 0 (K control bit). | |
| 23 | [0:7] | Bottom 8 bits of user programmable EOP character mask. | 0 |
| 24 | [0:7] | Bottom 8 bits of user programmable EOP character 0. | 0 |
| 25 | [0:7] | Bottom 8 bits of user programmable EOP character 1. | 0 |
| 26 | [0:4] | Reserved | |
| | [5] | Top bit of user programmable EOP character mask. | 0 |
| | [6] | Top bit of user programmable EOP character 1. | |
| | [7] | Top bit of user programmable EOP character 0. | |

*Table 13-2. Quad Interface Register Map (Continued)*

| Quad Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| 27 | [0:7] | Reserved | |
| **Per Quad SERDES Registers** | | | |
| 28 | [0] | Reserved | |
| | [1] | 0 = PLL SERDES macro power up. | 0 |
| | [2:3] | 00 = Internal high-speed bit clock is 20x or 16x.<br>01 = Internal high-speed bit clock is 10x or 8x.<br>10 = Internal high-speed bit clock is 5x or 4x. | |
| | [4] | Reserved | |
| | [5:7] | Loss of signal threshold select.<br>000 = lo threshold of 100 mVp-p and hi threshold of 175 mV p-p.<br>001 = lo threshold of 105 mVp-p and hi threshold of 186 mV p-p.<br>010 = lo threshold of 110 mVp-p and hi threshold of 197 mV p-p.<br>011 = lo threshold of 115 mVp-p and hi threshold of 208 mV p-p.<br>100 = lo threshold of 95 mVp-p and hi threshold of 164 mV p-p.<br>101 = lo threshold of 90 mVp-p and hi threshold of 153 mV p-p.<br>110 = lo threshold of 85 mVp-p and hi threshold of 142 mV p-p.<br>111 = lo threshold of 80 mVp-p and hi threshold of 131 mV p-p. | |
| 29 | [0] | Reserved | |
| | [1] | 1 = Reference clock DC coupling enable.<br>0 = Reference clock DC coupling disable (default). | 0 |
| | [2] | Reserved | |
| | [3] | Enable core clock as reference clock for both TX and RX. | |
| | [4:6] | Reserved | |
| | [7] | Termination at reference clock input buffer.<br>1 = high impedance.<br>0 = Not supported (default).<br><br>*Note: A '1' must be written to this register through auto configuration or through the system bus.* | |
| 2A | [0:5] | Reserved | 0 |
| | [7:6] | pll_lol_set (all settings have a tolerance of +/- 80 ppm)<br>lock/unlock<br>00 = +/- 640 ppm<br>01 = +/- 1950 ppm<br>10 = +/- 2200 ppm<br>11 = +/- 5980 ppm | |
| 2B | [0:7] | Reserved | |
| 2C | [0:1] | Far-end Serial Loopback enable:<br>“00” for normal operation<br>“10” for Pre-CDR (bit 3) or Post-CDR (bit 2) Loopback | |
| | 2 | '1' = Post-CDR Serial Far-end Loopback ([0:1]="10") | |
| | 3 | '1' = Pre-CDR Serial Far-end Loopback ([0:1]="10") | |
| | 4 | ‘1’ = Tx to Rx serial near-end loopback. *Note: Perform a reset after using serial near-end loopback.* | |
| | 5 | ‘1’ = Rx to Tx parallel loopback | |
| | [6:7] | Reserved | |
| 2D | [0:7] | Reserved | |

*Table 13-2. Quad Interface Register Map (Continued)*

| Quad Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| 2E | [0:7] | Reserved | |
| 2F | [0:7] | Reserved | |
| 30 | [0:3] | lock2ref[3:0], one bit per channel | 0 |
| | [4] | Reserved | |
| | [5] | Tx Synchronization<br>0 = transmit sync disabled<br>1 = transmit sync enabled | |
| | [7:6] | cdr_lol_set (all settings have a tolerance of +/-80ppm)<br>lock/unlock<br>00 = +/- 560 ppm<br>01 = +/- 1870 ppm<br>10 = +/- 2120 ppm<br>11 = +/- 5900 ppm | 0 |
| 31 | [0:7] | Reserved | |
| 32 | [0:7] | Reserved | |
| 33 | [0:7] | Reserved | |
| 34 | [0:7] | Reserved | |
| 35 | [0:7] | Reserved | |
| 36 | [0:7] | Reserved | |
| 37 | [0:7] | Reserved | |
| 38 | [0:7] | Reserved | |
| 39 | [0:7] | Reserved | |
| 3A | [0:7] | Reserved | |
| 3B | [0:7] | Reserved | |
| 3C | [0:7] | Reserved | |
| 3D | [0:7] | Reserved | |
| 3E | [0:7] | Reserved | |
| 3F | [0:3] | Reserved | 0 |
| | [4] | Interrupt for obtaining of lock on PLL<br>1 = Interrupt enabled for obtaining lock<br>0 = Interrupt not enabled for obtaining lock | |
| | [5:6] | Reserved | |
| | [7] | Interrupt for loss of lock on PLL<br>1 = Interrupt enabled for loss of lock<br>0 = Interrupt not enabled for loss of lock | |
| **Per Quad Reset Registers** | | | |
| 40 | [0] | 1 = Assert reset signal to channel 3 receive logic. | 0 |
| | [1] | 1 = Assert reset signal to channel 2 receive logic. | |
| | [2] | 1 = Assert reset signal to channel 1 receive logic. | |
| | [3] | 1 = Assert reset signal to channel 0 receive logic. | |
| | [4] | 1 = Assert reset signal to channel 3 transmit logic. | |
| | [5] | 1 = Assert reset signal to channel 2 transmit logic. | |
| | [6] | 1 = Assert reset signal to channel 1 transmit logic. | |
| | [7] | 1 = Assert reset signal to channel 0 transmit logic. | |

*Table 13-2. Quad Interface Register Map (Continued)*

| Quad Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| 41 | [0:2] | Reserved | |
| | [3] | Receive side loss of lock reset (reset test feature for all 4 channels). | 04 |
| | [4] | Transmit side loss of lock rest (reset test feature for all 4 channels). | |
| | [5] | 1 = Assert PLL macro reset. | |
| | [6] | 1 = Reset channel pair 2, 3 alignment logic. | |
| | [7] | 1 = Reset channel pair 0, 1 alignment logic. | |
| 42 | [0:5] | Reserved | |
| | [6] | Reserved | 03 |
| | [7] | Enables resets to be sourced from the FPGA interface.<br>1 = enabled (default). | |
| 43 | [0:6] | Reserved | |
| | [7] | quad reset; resets entire quad and all serdes channels. *Note: Perform a reset when using serial near-end loopback.* | 0 |
| **Additional Per Quad General Registers** | | | |
| 44 | [0:7] | Reserved | |
| 45 | [0:7] | Reserved | |
| **Status Registers** | | | |
| **Per Quad General PCS Registers** | | | |
| 80 | [0:2] | Reserved | |
| | [3] | “or” of all channel interrupts of that quad. | 0 |
| | [4] | “or” of all channel 0 register interrupts. | |
| | [5] | “or” of all channel 1 register interrupts. | |
| | [6] | “or” of all channel 2 register interrupts. | |
| | [7] | “or” of all channel 3 register interrupts. | |
| 81 | [0:5] | Reserved | |
| | [6] | Reserved | |
| | [7] | Reserved | |
| **Per Quad Align PCS Registers** | | | |
| 82<br>(These bits only valid when the internal state machine is enabled (CIR 00 bit 7=1)) | [0] | Invert of channel 01 aligned status. | |
| | [1] | Invert of channel 23 aligned status. | 1 |
| | [2] | Channel 01 aligned status. | 1 |
| | [3] | Channel 23 aligned status. | 0 |
| | [4] | Channel 01 outskew status. | 0 |
| | [5] | Channel 23 outskew status. | 0 |
| | [6] | Multi-channel 01 inskew status. | 0 |
| | [7] | Multi-channel 23 inskew status. | 0 |

*Table 13-2. Quad Interface Register Map (Continued)*

| Quad Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| 83 | [0] | Interrupt for channel 01 internal finite state machine alignment status when it goes low (out of align). | |
| | [1] | Interrupt for channel 23 internal finite state machine alignment status when it goes low (out of align). | |
| | [2] | Interrupt for channel 01 internal finite state machine alignment status (in alignment). | 0 |
| | [3] | Interrupt for channel 23 internal finite state machine alignment status (in alignment). | |
| | [4] | Interrupt for channel 01 alignment failed status. | |
| | [5] | Interrupt for channel 23 alignment failed status. | |
| | [6] | Interrupt for multi-channel 01 alignment status. | |
| | [7] | Interrupt for multi-channel 23 alignment status. | |
| **Per Quad Packet PCS Registers** | | | |
| 84 | [0] | Invert of word aligner link status for channel 0. | 1 |
| | [1] | Invert of word aligner link status for channel 1. | 1 |
| | [2] | Invert of word aligner link status for channel 2. | 1 |
| | [3] | Invert of word aligner link status for channel 3. | 1 |
| | [4] | Word aligner link status for channel 0. | 0 |
| | [5] | Word aligner link status for channel 1. | 0 |
| | [6] | Word aligner link status for channel 2. | 0 |
| | [7] | Word aligner link status for channel 3. | 0 |
| 85 | [0] | Interrupt for word aligner link status for channel 0 when it goes low (out of sync). | |
| | [1] | Interrupt for word aligner link status for channel 1 when it goes low (out of sync). | |
| | [2] | Interrupt for word aligner link status for channel 2 when it goes low (out of sync). | |
| | [3] | Interrupt for word aligner link status for channel 3 when it goes low (out of sync). | |
| | [4] | Interrupt for word aligner link status for channel 0 (in sync). | 0 |
| | [5] | Interrupt for word aligner link status for channel 1 (in sync). | |
| | [6] | Interrupt for word aligner link status for channel 2 (in sync). | |
| | [7] | Interrupt for word aligner link status for channel 3 (in sync). | |
| **Per Quad SERDES Registers** | | | |
| 86 | [0:3] | Reserved | |
| | [4] | 1 = PLL lock obtained | 0 |
| | [5:6] | Reserved | |
| | [7] | 1 = PLL loss of lock | |
| 87 | [0:7] | Reserved. | |
| 88 | [0:3] | Reserved | |
| | [4] | Interrupt for PLL lock obtained<br>1 = interrupt generated<br>0 = interrupt not generated on | |
| | [5:6] | Reserved | |
| | [7] | Interrupt for PLL loss of lock<br>1 = Interrupt generated<br>0 = Interrupt not generated | |

*Table 13-3. Channel Interface Register Map*

| Channel Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| **Control Registers** | | | |
| ***Per Channel General Registers*** | | | |
| 00 | [0:2] | Reserved | |
| | [3] | 1 = Disable the CRC logic on Rx when in SOP mode (bit 4)<br>0 = Normal CRC operation | 0 |
| | [4] | 1 = Send the start-of-packet (SOP) indicator in packet modes to the FPGA interface on the code violation port. Code violation can be detected with rx_k = 1 and rxd = EE.<br>0 = Code violation on code violation port | |
| | [5] | 1 = Enable loopback just before FPGA bridge from RX to TX.<br>0 = Normal data operation. | |
| | [6] | 1 = Enable loopback from transmit to receive in SERDES bridge.<br>0 = Normal data operation.<br>**CHANNEL MUST BE RESET AFTER CHANGING THIS REGISTER BIT.** | |
| | [7] | 1 = Link state machine is enabled.<br>0 = Link state machine is not enabled. | |
| 01 | [0] | Reserved | |
| | [1] | 1 = Enable PRBS generator and checker.<br>0 = Normal operational mode. | 0 |
| | [2] | 1 = Lock receive PRBS checker.<br>0 = Unlock receive PRBS checker. | |
| | [3] | Reserved | |
| | [4] | 1 = Reverse bit order of transmit data bus.<br>0 = Don't reverse bit order of transmit data bus. | |
| | [5] | 1 = Invert transmitted data.<br>0 = Don't invert transmitted data. | |
| | [6] | 1 = Reverse bit order of receive data bus.<br>0 = Don't reverse bit order of receive data bus. | |
| | [7] | 1 = Invert received data.<br>0 = Don't invert received data. | |
| 02 | [0:5] | Reserved | |
| | [6] | 1 = Receive outputs can be monitored on the test characterization pins.<br>The test characterization mode (QIR address 0x03, bit 6) should be set to '1'. | 0 |
| | [7] | 1 = Transmit PCS inputs are sourced from test characterization ports. The test characterization mode should be enabled. | |
| 03 | [0:7] | 1 = Enable interrupt on Section B1 bip error.<br>Interrupts for bip error [7:0] are CIR 0x98[0:7]. | 0 |
| 04 | [0:7] | Reserved | |

*Table 13-3. Channel Interface Register Map (Continued)*

| Channel Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| **Per Channel Packet PCS Registers** | | | |
| 05 | [0] | Locks/unlocks word alignment.<br>To lock, first write a '0' to bit '0', then write a '1' to bit '0'.<br>To unlock, write a 0 to bit '0'.<br>See description for word_align_en_[0:3] signal in 8-bit and 10-bit SERDES Only Modes section of this data sheet. | 0 |
| | [1] | 1 = Disable PCI Express scrambler on transmit.<br>0 = Enable PCI Express scrambler on transmit path. | |
| | [2] | 1 = Disable PCI Express descrambler on receive path.<br>0 = Enable PCI Express descrambler on receive path. | |
| | [3] | Reserved | |
| | [4] | 1 = Restart AN process.<br>0 = Normal operation. | |
| | [5] | 1 = Enable AN.<br>0 = Disable AN. | |
| | [6] | 1 = Bypass 8b10b encoder.<br>0 = Normal operation. | |
| | [7] | 1 = Bypass 8b10b decoder.<br>0 = Normal operation. | |
| 06 | [0:7] | Upper bits of AN advertisement register. | 0 |
| 07 | [0:7] | Lower bits of AN advertisement register. | 0 |
| 08 | [0:7] | Lower bits of AN next page transmit register. | 0 |
| 09 | [0:7] | Upper bits of AN next page transmit register. | 0 |
| 0A | [0:1] | Reserved | |
| | [2] | 1 = Enable interrupt for resolve AN priority.<br>0 = Disable interrupt for resolve AN priority.<br>Interrupt for resolve priority is CIR 0x8A[2]. | 0 |
| | [3] | 1 = Enable interrupt for AN process complete.<br>0 = Disable interrupt for AN process complete.<br>Interrupt for AN process complete is CIR 0x8A[3]. | |
| | [4] | 1 = Enable interrupt for AN page received.<br>0 = Disable interrupt for AN page received.<br>Interrupt for AN page received is CIR 0x8A[4]. | |
| | [5] | 1 = Enable interrupt for next page loaded.<br>0 = Disable interrupt for next page loaded.<br>Interrupt for next page loaded is CIR 0x8A[5]. | |
| | [6] | 1 = Enable interrupt for CTC FIFO underrun.<br>0 = Disable interrupt for CTC FIFO underrun.<br>Interrupt for CTC FIFO underrun is CIR 0x8A[6]. | |
| | [7] | 1 = Enable interrupt for CTC FIFO overrun.<br>0 = Disable interrupt for CTC FIFO overrun.<br>Interrupt for CTC FIFO overrun is CIR 0x8A[7]. | |

*Table 13-3. Channel Interface Register Map (Continued)*

| Channel Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| **Per Channel SONET PCS Register** | | | |
| 0B | [0:2] | Reserved | |
| | [3] | 1 = A1 A2 byte generation for the next frame is delayed by 2 clock cycles<br>0 = Normal operation | 0 |
| | [4] | 1 = TOH logic generates B1 BIP byte for first STS-1 position<br>0 = B1 BIP byte for first STS-1 position is provided from the FPGA core logic | |
| | [5] | 1 = A1A2 bytes are inserted by the TOH logic<br>0 = A1A2 bytes are provided from the FPGA core logic | |
| | [6] | Reserved | |
| | [7] | 1 = STS-12 mode<br>0 = STS-48 mode | |
| 0C | [0] | Reserved | |
| | [1] | 1 = Scrambler function is bypassed<br>0 = Scrambler function is active | 0 |
| | [2] | 1 = TFI-5 mode<br>0 = Not TFI-5 mode | |
| | [3] | 1 = Force RDI when AIS-L indication is received from mate RX channel<br>0 = Do not send RDI when AIS-L indication is received | |
| | [4] | 1 = Produce frame pulse on A2<br>0 = Produce frame pulse on A1 | |
| | [5:7] | OOF selector | |
| 0D | [0:1] | Reserved | |
| | [2:3] | 2'b11 = Take output from Pointer Interpreter<br>2'b10 = Take output from Multi-channel Aligner<br>2'b01 = Take output from AIS-L Block output and before the Multi-channel Aligner<br>2'b00 = Take output from Pointer Interpreter | 0 |
| | [4] | 1 = Descrambler function is bypassed<br>0 = Descrambler function is active | |
| | [5] | 1 = AIS-L is inserted on frames during an OOF condition<br>0 = Do not insert AIS-L during OOF condition | |
| | [6] | 1 = AIS-L is inserted to every frame regardless of OOF condition<br>0 = Do not insert AIS-L on every frame | |
| | [7] | 1 = Fast frame mode for framer is enabled<br>0 = Fast frame mode for framer is not enabled | |
| 0E | [0] | Reserved | |
| | [1] | 1 = Enable persistent errors<br>0 = Disable persistent errors | 0 |
| | [2:7] | 6 bit count of K2 error indications to insert | |
| 0F | [0] | Reserved | |
| | [1] | 1 = Enable persistent errors<br>0 = Disable persistent errors | 0 |
| | [2:7] | 6 bit count of B1 error indications to insert | |

*Table 13-3. Channel Interface Register Map (Continued)*

| Channel Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| 10 | [0] | Reserved | |
| | [1] | 1 = Enable persistent errors.<br>0 = Disable persistent errors. | 0 |
| | [2:7] | 6 bit count of A1A2 error indications to insert. | |
| 11 | [0:4] | Reserved | |
| | [5] | 1 = Initiate start of A1A2 insertion errors.<br>0 = Normal operation. | 0 |
| | [6] | 1 = Initiate start of B1 insertion errors.<br>0 = Normal operation. | |
| | [7] | 1 = Initiate start of K2 (RDI) insertion errors.<br>0 = Normal operation. | |
| 12 | [0:4] | Reserved | |
| | [5] | 1 = Enable interrupt to signal that a RDI-L has been detected on the receive channel.<br>Interrupt for RDI-L detect is CIR 0x93[5]. | 0 |
| | [6] | 1 = Enable interrupt to indicate an out of frame condition.<br>Interrupt for out of frame is CIR 0x93[6]. | |
| | [7] | 1 = Enable interrupt to indicate framing has been achieved.<br>Interrupt for good frame is CIR 0x93[7]. | |
| **Per Channel SERDES Registers** | | | |
| 13 | [0:3] | Reserved | |
| | [4] | 0 = Full rate selection for receive.<br>1 = Half rate selection for receive. | 0 |
| | [5] | 0 = Full rate selection for transmit.<br>1 = Half rate selection for transmit. | |
| | [6] | 0 = Power down receive channel.<br>1= Power up receive channel. | |
| | [7] | 0 = Power down transmit channel.<br>1= Power up transmit channel. | |
| 14 | [0] | 1 = TX driver pre-emphasis enable.<br>0 = TX driver pre-emphasis disable. | 0 |
| | [1:3] | TX driver pre-emphasis level setting.<br>000 = 0%<br>001 = 16%<br>010 = 32%<br>011 = 48% | |
| | [4:5] | Resistor termination select for TX.<br>00 = 50 ohm (default)<br>01 = 75 ohm<br>10 = 5K ohm<br>Selection disabled when PCI Express feature is enabled. | |
| | [6:7] | CML driver amplitude setting.<br>00 = default swing amplitude<br>01 = -12.5% default swing<br>10 = -25% default swing<br>11 = +12.5% default swing | |

*Table 13-3. Channel Interface Register Map (Continued)*

| Channel Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| 15 | [0:1] | Reserved | |
| | [2] | 1 = Enables boundary scan input path for routing the high-speed receive inputs to a lower speed SERDES in the FPGA. | 0 |
| | [3] | 1 = Receiver equalization enable.<br>0 = Receiver equalization disable. | |
| | [4] | 1 = Receiver DC coupling enable.<br>0 - AC coupling (default). | |
| | [5] | Level setting for equalization. | |
| | [6:7] | 00 =50 ohm (default).<br>01 = 75 ohm.<br>10 - 2K ohm.<br>11 - 60 ohm. | |
| 16 | [0:7] | Reserved | |
| 17 | [0] | Reserved | |
| | [1] | 1 = Enable interrupt for detection of far-end receiver for PCI Express.<br>Interrupt for far-end receiver detection is CIR 0x97[1]. | 0 |
| | [2] | 1 = Enable interrupt for receiver los when input levels fall below the programmed lo threshold.<br>0 = Disable interrupt for receiver los when input levels fall below the programmed lo threshold.<br>Interrupt for receiver los is CIR 0x97[2]. | |
| | [3] | 1 = Enable interrupt for receiver los when input level meets or is > programmed lo threshold.<br>Interrupt for receiver los > lo threshold is CIR 0x97[3]. | |
| | [4] | 1 = Enable interrupt for receiver los when input levels fall below the programmed hi threshold.<br>0 = Disable interrupt for receiver los when input levels fall below the programmed hi threshold.<br>Interrupt for levels below hi threshold is CIR 0x97[4]. | |
| | [5] | 1 = Enable interrupt for receiver los when input level meets or is > programmed hi threshold.<br>Interrupt for levels above hi threshold is CIR 0x97[5]. | |
| | [6] | 1= Enable interrupt for receiver loss of lock.<br>Interrupt for receiver loss of lock is CIR 0x97[6]. | |
| | [7] | 1= Enable interrupt when receiver recovers from loss of lock.<br>Interrupt for loss of lock recovery is CIR 0x97[7]. | |
| **Status Registers** | | | |
| **Per Channel General Registers** | | | |
| 80 | [0:7] | Reserved | |
| 81 | [0:7] | Count of the number of PRBS errors.<br>Clears to zero on read. Sticks at FF. | 0 |
| 82 | [0:7] | Reserved | |
| 83 | [0:7] | 8-bit BIP error indicator. A '1' on each bit indicates an error in the corresponding bit position within the B1 byte. | 0 |
| 84 | [0:7] | Reserved | |

*Table 13-3. Channel Interface Register Map (Continued)*

| Channel Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| **Per Channel Packet PCS Registers** | | | |
| 85 | [0:1] | Reserved | |
| | [2] | 1 = Request to processor to resolve AN priority. | 0 |
| | [3] | 1 = AN process complete.<br>0 = AN process ongoing. | |
| | [4] | 1 = AN page received.<br>0 = No AN page received. | |
| | [5] | 1 = Next page loaded.<br>0 = Next page not loaded. | |
| | [6] | 1 = Clock compensator FIFO underrun.<br>0 = CC FIFO has not underrun. Latching high, cleared on read. | |
| | [7] | 1 = Clock compensator FIFO overrun.<br>0 = CC FIFO has not overrun. Latching high, cleared on read. | |
| 86 | [0:7] | Lower bits of received (link partner) AN advertisement register. | 0 |
| 87 | [0:7] | Upper bits of received (link partner) AN advertisement register. | 0 |
| 88 | [0:7] | Lower bits of received (link partner) AN next page register. | 0 |
| 89 | [0:7] | Upper bits of received (link partner) AN next page register. | 0 |
| 8A | [0:1] | Reserved | |
| | [2] | 1 = Interrupt generated on resolve AN priority.<br>0 = Interrupt not generated on resolve AN priority.<br>Clear on read.<br>Has associated interrupt control register (CIR address 0x0A[2]) and status register (CIR 0x85[2]). | 0 |
| | [3] | 1 = Interrupt generated on AN process complete.<br>0 = Interrupt not generated on AN process complete.<br>Clear on read.<br>Has associated interrupt control register (CIR address 0x0A[3]) and status register (CIR 0x85[3]). | |
| | [4] | 1 = Interrupt generated on AN page received.<br>0 = Interrupt not generated on AN page received.<br>Clear on read.<br>Has associated interrupt control register (CIR address 0x0A[4]) and status register (CIR 0x85[4]). | |
| | [5] | 1 = Interrupt generated on next page loaded.<br>0 = Interrupt not generated on next page loaded.<br>Clear on read.<br>Has associated interrupt control register (CIR address 0x0A[5]) and status register (CIR 0x85[5]). | |
| | [6] | 1 = Interrupt generated on CTC FIFO underrun.<br>0 = Interrupt not generated on CTC FIFO underrun.<br>Clear on read.<br>Has associated interrupt control register (CIR address 0x0A[6]) and status register (CIR 0x85[6]). | |
| | [7] | 1 = Interrupt generated on CTC FIFO overrun.<br>0 = Interrupt not generated on CTC FIFO overrun.<br>Clear on read.<br>Has associated interrupt control register (CIR address 0x0A[7]) and status register (CIR 0x85[7]). | |

*Table 13-3. Channel Interface Register Map (Continued)*

| Channel Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| **Per Channel SONET PCS Registers** | | | |
| 8B | [0:4] | Reserved | |
| | [5] | 1= Indicates that a RDI-L has been detected. | 0 |
| | [6] | 1= Indicates an out-of-frame. | |
| | [7] | 1 = Indicates that the framer has made transition from an out of frame to a valid frame. | |
| 8C | [0:7] | 8-bit count of BIP ERROR indications. Clears to zero on read. Sticks at hex FF. Clear on read. | 0 |
| 8D | [0:7] | Concatenation status indicator.<br>0 = the STS-1 is the head of a concatenation group.<br>1 = the STS-1 is part of a concatenated group. | 0 |
| 8E | [0:7] | Concatenation status indicator.<br>0 = STS-1 is the head of a concatenation group.<br>1 = STS-1 is part of a concatenated group. | 0 |
| 8F | [0:7] | Concatenation status indicator.<br>0 = STS-1 is the head of a concatenation group.<br>1 = STS-1 is part of a concatenated group. | 0 |
| 90 | [0:7] | Concatenation status indicator.<br>0 = STS-1 is the head of a concatenation group.<br>1 = STS-1 is part of a concatenated group. | 0 |
| 91 | [0:7] | Concatenation status indicator.<br>0 = STS-1 is the head of a concatenation group.<br>1 = STS-1 is part of a concatenated group. | 0 |
| 92 | [0:7] | Concatenation status indicator.<br>0 = STS-1 is the head of a concatenation group.<br>1 = STS-1 is part of a concatenated group. | 0 |
| 93 | [0:4] | Reserved | |
| | [5] | 1= Interrupt generated for RDI-L detect.<br>Has associated interrupt enable register (CIR address 0x12[5]) and status register (CIR 0x8B[5]). | 0 |
| | [6] | 1= Interrupt generated for out-of-frame.<br>Has associated interrupt enable register (CIR address 0x12[6]) and status register (CIR 0x8B[6]). | |
| | [7] | 1= Interrupt generated for good frame.<br>Has associated interrupt enable register (CIR address 0x12[7]) and status register (CIR 0x8B[7]). | |

*Table 13-3. Channel Interface Register Map (Continued)*

| Channel Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| **Per Channel SERDES Registers** | | | |
| 94 | [0] | Reserved | |
| | [1] | 1 = Receiver detection process completed by SERDES transmitter.<br>0 = Receiver detection process not completed by SERDES transmitter. | 0 |
| | [2] | 1= Indicates that the input signal detected by receiver is below the programmed lo threshold. Cleared on read. | |
| | [3] | 1= Indicates that the input signal detected by receiver is ≥ the programmed lo threshold. Cleared on read. | |
| | [4] | 1= Indicates that the input signal detected by receiver is below the programmed hi threshold. Cleared on read. | |
| | [5] | 1= Indicates that the input signal detected by receiver is ≥ the programmed hi threshold. Cleared on read. | |
| | [6] | 1 = Indicates CDR loss of lock to data. CDR is locked to reference clock. | |
| | [7] | 1 = Indicates that CDR has locked to data. | |
| 95 | [0:6] | Reserved | |
| | [7] | 1 = Receiver detected by SERDES transmitter.<br>0 = Receiver not detected by SERDES transmitter. | 0 |
| 96 | [0:7] | Reserved | |
| 97 | [0] | | |
| | [1] | 1= Interrupt generated for pci_det_done.<br>Clear on read.<br>Has associated interrupt enable register (CIR address 0x17[1]) and status register (CIR 0x94[1]). | 0 |
| | [2] | 1 = Interrupt generated for rlos_lo.<br>Clear on read.<br>Has associated interrupt enable register (CIR address 0x17[2]) and status register (CIR 0x94[2]). | |
| | [3] | 1 = Interrupt generated for ~rlos_lo.<br>Clear on read.<br>Has associated interrupt enable register (CIR address 0x17[3]) and status register (CIR 0x94[3]). | |
| | [4] | 1 = Interrupt generated for rlos_hi.<br>Clear on read.<br>Has associated interrupt enable register (CIR address 0x17[4]) and status register (CIR 0x94[4]). | |
| | [5] | 1 = Interrupt generated for ~rlos_hi.<br>Clear on read.<br>Has associated interrupt enable register (CIR address 0x17[5]) and status register (CIR 0x94[5]). | |
| | [6] | 1 = Interrupt generated for rlol.<br>Clear on read.<br>Has associated interrupt enable register (CIR address 0x17[6]) and status register (CIR 0x94[6]). When polling, do not poll faster than the typical value as indicated in Table 2-4. | |
| | [7] | 1 = Interrupt generated for ~rlol.<br>Clear on read.<br>Has associated interrupt enable register (CIR address 0x17[7]) and status register (CIR 0x94[7]). When polling, do not poll faster than the typical value in Table 2-4. | |

*Table 13-3. Channel Interface Register Map (Continued)*

| Channel Interface Register Offset Address 0x (Hex) | Bits | Description | Default Value |
|---|---|---|---|
| **Additional Channel SONET Registers** | | | |
| 98 | [0:7] | 1 = interrupt has been generated for the error in the corresponding bit position of the B1 byte.<br>Clear on read.<br>Has associated interrupt enable register (CIR address 0x03[0:7]) and status register (CIR 0x83[0:7]). | 0 |

# LatticeSC/M Family flexiPCS Data Sheet Revision History



| Date | Version | Section | Change Summary |
|---|---|---|---|
| February 2006 | 01.0 | — | Original version. |
| March 2006 | 01.1 | SERDES Functionality/ Electrical & Timing | Updated SERDES Power Up paragraphs 2, 3 and 4. |
| | | | Updated SERDES Internal Bias Generators diagram for RESP addition. |
| | | | Updated document for removal of rxrefclk. |
| | | | Inserted new section titled Low Speed SERDES Receiver Operation. |
| | | | Updated External CML Reference Clock Specification table for rxrefclk removal. |
| | | | Inserted new section titled “Interfacing LVDS and LVPECL to Reference Clock CML Buffers”. |
| | | | Changed SERDES typical power from 100 to 105 mW. |
| | | | Added 135 mW SERDES max power. |
| | | Memory Map | Changed Quad SERDES Register 29 bit 2 to reserved (was rx ref clock DC coupling). |
| | | | Changed Quad SERDES Register 29 bit 4 to reserved (was enable rx ref clock). |
| | | | Changed Quad SERDES Register 29 bit 7 to “not supported” and added a note. |
| | | Miscellaneous | New diagrams to remove rxrefclk. |
| | | | Changed “secondary clock routing” to “general clock routing” |
| | | | Removed rxrefclk from SERDES port list. |
| | | | Clarified 16/20 Bit Data Bus Mode Transmit Clock Rates section to detail order that bytes are presented to the SERDES. |
| | | | Enhanced quad_rst description. |
| June 2006 | 01.2 | Introduction | Changed total bandwidth per quad from 13.6 Gbps to 15.2 Gbps |
| | | SERDES Functionality/ Electrical & Timing | Added “SERDES Internal Bias Generators (for Packages that Include RESPN_[ULC/URC] Pins” figure. |
| | | | Changed Rx jitter tolerance data pattern from PRBS 2^7-1 to CJPAT |
| | | | Changed “VSSRX” to VSS” in Serial Input Data Input Specifications table. |
| | | | Added “SERDES Power Supply Sequencing Requirements” section. |
| | | | Changed SERDES typical power from 105 mW to 110 mW. |
| | | | Updated SERDES Power Supply Package Requirements table. Changed Description for VDDAX25. Removed VSSTX, VSSRX and VSSP. Added RESP and RESPN. |
| | | PCI Express Mode | Changed “tx_scrm_en input is high” to “tx_scrm_en input is low”. |
| | | | Changed “xx ref_pclk cycles” to “10 ref_pclk cycles”. |
| | | SONET Mode | Removed A1A2, B1, or K2 Error Count Limit Flag row from Status Interrupt Register Bits table. |
| | | Memory Map | Changed CIR 04 bits 6 and 7 to reserved (was Tx and Rx bridge FIFO overrun interrupt enable |
| | | | Changed CIR 0E bit 0 to reserved (was K2 error 6 bit count indication interrupt enable |

© 2011 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

| Date | Version | Section | Change Summary |
|---|---|---|---|
| June 2006 (Cont.) | 01.2 (Cont.) | Memory Map (Cont.) | Changed CIR 0F bit 0 to reserved (was B1 error 6 bit count indication interrupt enable). |
| | | | Changed CIR 10 bit 0 to reserved (was A1A2 error 6 bit count indication interrupt enable). |
| | | | Changed CIR 12 bit 4 to reserved (was group K2, B1, A1A2 error count interrupt enable). |
| | | | Changed CIR 80 bits 6 and 7 to reserved (was Tx and Rx bridge FIFO overrun interrupt). |
| | | | Changed CIR 84 bits 6 and 7 to reserved (was Tx and Rx bridge FIFO overrun interrupt). |
| | | | Changed CIR 8B bit 4 to reserved (was A1A2, B1 and K2 error count has been reached status register). |
| | | | Changed CIR 93 bit 4 to reserved (was A1A2, B1 and K2 error count interrupt). |
| | | Miscellaneous | Changed number of pre-emphasis settings from six to three (16, 32 and 48). |
| | | | Changed all VDDP, VDDTX and VDDRX to VCC12. |
| | | | Changed max SERDES bandwidth from 3.4 Gbps to 3.8 Gbps. |
| | | | Changed half rate bandwidth from 1.7 Gbps to 1.9 Gbps. |
| August 2006 | 01.3 | SERDES Functionality/ Electrical and Timing | Added Transmit Data Skew section |
| | | | Changed rise and fall time values in Table 2-17 from 50 to 80 min, 80 to 125 typ and 110 to 180 max. |
| | | | Added Tx Skew of 200 ps and note 5 to Table 2-17 |
| | | | Added Table 2-19. flexiPCS Transmit Data Latencies |
| | | | Added Table 2-22. flexiPCS Receive Data Latencies |
| | | | Changed typical CDR re-lock time from 1000 bits to 3000 bits. |
| | | 8-bit and 10-bit SERDES Only Modes | Corrected quad 18 40 # Set quad to 10-bit SERDES Only Mode (was quad 18 10...) |
| | | Generic 8b10b Mode | Added “This signal is only valid when CRC is enabled by setting QIR 0x1D bits 2 and 3 = '1'. To prevent this signal from going high on CRC errors, set CIR 0x00 bit 3 ='1'.” to rx_crc_eop_[0-3] and rx_crc_eop_[0-3](1:0) description. |
| | | | Added “This figure shows an example where the SOP and EOP phases are two characters long with the first character being a control character. The length and value of SOP and EOP is programmable via quad interface registers. The generic 8b10b autoconfig example provides details on setting SOP and EOP length and value.” |
| | | | Corrected the timing diagram for signal rx_k_[0-3] |
| | | | Corrected the timing diagram for signal rx_k_[0:3] |
| | | Serial RapidIO Mode | Corrected quad 18 02 # Set quad to Serial RapidIO Mode (was quad 18 20...) |
| | | Memory Map | Added lock2ref[0:3] to address 30 [0:3] |
| November 2006 | 01.4 | SERDES Functionality/ Electrical and Timing | Updated the differential swing typical values. |
| | | | Added differential swing typical values for amplitude boost mode. |
| | | | Updated Tx jitter from 0.29 UI to 0.25 UI. |
| | | | Updated the SERDES power numbers. |
| | | Memory Map | Added Tx pll_lol set to address 2A |
| | | | Added Rx cdr_lol_set to address 30 |

| Date | Version | Section | Change Summary |
|---|---|---|---|
| November 2006 (Cont.) | 01.4 (Cont.) | Memory Map (Cont.) | Added entries to QIR 82 bits [0:1] |
| | | | Added entries to QIR 84 bits [0:3] |
| March 2007 | 01.5 | SERDES Functionality/ Electrical and Timing | Updated first footnote of Serial Output Timing and Levels table. |
| | | 8-bit and 10-bit SERDES Only Modes | Updated 10-Bit SERDES Only with Word Alignment Auto Configuration Example File figure. |
| | | Generic 8b10b Mode | Added lsm_en_[0-3] and lsm_status_[0-3] symbols to Generic 8b10b Mode Pin Description table. |
| August 2007 | 01.6 | General | Changed "ff_refclk" to "refclk". |
| | | | Changed "ff_rxrefclk" to "rxrefclk". |
| | | | Added broadcast write information. |
| | | SERDES Functionality/ Electrical and Timing | Inserted a column indicating type of clock in Table 2-2 SERDES Common Ports. |
| | | | Inserted Channel Output Jitter Data and added a footnote regarding wire bond packages to Table 2-18. |
| | | | Added a column and data for 8bit/10bit SERDES-only to Table 2-19 and Table 2-23. |
| | | | Updated Max. data for Stream of non-transitions in Table 2-20 and Min. data for Jitter Tolerance in Table 2-21. |
| | | | Added new Table for Periodic Jitter Tolerance. |
| | | | Added Figure 2-16 Jitter Transfer |
| | | | Added information to Table 2-25. |
| | | | Modified data in Table 2-26. |
| | | 8-bit and 10-bit SERDES only Modes, Generic 8b10 Mode, Gigabit Ethernet Mode, Fibre Channel Mode, XAUI Mode, PCI-Express Mode, Serial RapidIO Mode, and SONET Mode | Updated information in Auto-Configuration figures. |
| | | | Added Clock information to Pin Description tables. |
| | | Gigabit Ethernet Mode, Fibre Channel Mode, and XAUI Mode, PCI-Express Mode, and Serial RapidIO Mode, and SONET mode | Corrected information regarding underrun and overrun (bit 6 and7). |
| | | SONET Mode | Removed discussion of Auto TOH. |
| | | | Modified Figure 10-11 Frame Pulse Generation on A1 or A2. |
| | | | Modified Table 10-6 Bypass Mode Controls. |
| | | Multi_Channel Alignment | Removed discussion of mca_aligned_01 |
| | | | Modified Figure 11-15. |
| | | flexiPCS Testing | Removed discussion of Post CDR, Serial Far-end loopback mode. |
| | | | Modified Figure 12-2 Far-End Loopmback Modes (Single Channel). |
| | | Memory Map | Modified Table 13-2 Quad Interface Register Map, Offset Address 0x02, 0x0B, 0x0E, 0x18, 0x28 0x2C, 0x3F, 0x86, 0x88 |
| | | | Modified Table 13-3 Channel Interface Register Map, Offset Adddress 0x01, 0x0B, 0x0D, 0x8A, 0x85 |

| Date | Version | Section | Change Summary |
|---|---|---|---|
| January 2008 | 01.7 | SERDES Funtionality/ Electrical & Timing Characteristics | SERDES Port Description, Table 2-2, Updated description for tclk_[0-3] and rclk_[0-3] |
| | | | Changed “FIFOs” to “phase compensation FIFOs” |
| | | | Added paragraph to Receive Clocks from FPGA Logic section. |
| | | | Updated SERDES Performance section |
| | | | Updated Table 2-17 |
| | | | Added footnote to Table 2-24 |
| | | 8-bit and 10-bit SERDES Only Modes | Table 3-2, updated description for tclk_[0-3] and rclk_[0-3] |
| | | Generic 8b10b Mode | Table 4-2, updated description for tclk_[0-3] and rclk_[0-3] |
| | | | 8b10b Decoder section, modified 2nd paragraph |
| | | | Multi-quad and Multi-ship Alignment section and Table 4-15., changed “FIFOs” to “phase compensation FIFOs” |
| | | Gigabit Ethernet Mode | Changed “FIFO” to phase compensation FIFO” |
| | | | Table 5-2, modified description for tclk_[0-3], rclk_[0-3], tx_er_[0-3] |
| | | Fibre Channel Mode | Changed “FIFO” to phase compensation FIFO” |
| | | | Table 6-2, modified description for tclk_[0-3] and rclk_[0-3] |
| | | | Figure 6-9, changed “B=D21.4” to “B-D21.5” |
| | | XAUI Mode | Changed “FIFO” to phase compensation FIFO” |
| | | | Table 7-2, updated description for tclk and rclk |
| | | PCI-Express Mode | Changed “FIFO” to phase compensation FIFO” |
| | | | Table 8-2, modified description for tclk_[0-3] and rclk_[0-3] |
| | | | Table 8-6, updated description for tclk and rclk |
| | | Serial RapidIO Mode | Changed “FIFO” to phase compensation FIFO” |
| | | | Table 9-2, modified description for tclk_[0-3] and rclk_[0-3] |
| | | | 8b10b Encoding section, modified 2nd paragraph |
| | | | Table 9-8, updated description for tclk and rclk |
| | | SONET Mode | Changed “FIFO” to phase compensation FIFO” |
| | | | Table 10-2, modified description for tclk_[0-3] and rclk_[0-3] |
| | | | Modified info for rx_spe_[0-3] |
| | | | Corrected sentence beneath Figure 10-13 |
| | | flexiPCS Testing | Figure 12-1, modified SERDES transmit to SERDES receive |
| | | | Far-End Loopback Modes, inserted #3 |
| | | | Modified Figure 12-2 |
| June 2008 | 01.8 | SERDES Funtionality/ Electrical & Timing Characteristics | Updated Differential LVPECL Driving CML Reference Clock Buffer (DC) figure. |
| December 2008 | 01.9 | SERDES Funtionality/ Electrical & Timing Characteristics | Corrected #7 of flexiPCS Reset Sequence for Initial Loading. |
| | | | Updated Jitter Tolerance Specification tables 2-21 ( Receiver Total Jitter Tolerance Specification) and 2-22 (Periodic Receiver Jitter Tolerance Specification). |
| | | | Updated External CML Reference Clock Table 2-24 (External CML Reference Clock Specification). |
| | | 8-bit and 10-bit SERDES Only Modes | Updates for word alignment control. |

| Date | Version | Section | Change Summary |
|---|---|---|---|
| December 2008 (cont.) | 01.9 (cont.) | Gigabit Ethernet Mode | Added information about IDLE phase. |
| | | PCI Express Mode | Corrections for settings of Quad Interface Register Offset Address 0x0E, bits 4 and 5. |
| | | | Corrections for settings of QIR Offset Address 0x0A. |
| | | Multichannel Alignment | Added information for when the alignment state machine is enabled (below Figure 11-10 - Multi-channel Alignment Control and Monitoring Signals for Multi-channel Alignment (1 Quad)). |
| | | | Added information for QIR 0x82 settings for multichannel alignment within a quad. |
| | | | Added a note for alignment between quads. |
| | | | Updated information for alignment between two devices. |
| | | Memory Map | Corrected Table 13-2 (Quad Interface Register Map):<br>QIR Offset 0A bits [4:5], [6:7]<br>QIR Offset 13 bits [0:1], [2:3], [4:5], [6:7]<br>QIR Offset 30 bits [0:3], [5]<br>QIR Offset 42 bit [6]<br>QIR Offset 81 bits [6], [7] |
| | | | Added note for QIR offset 82 |
| | | | Corrected Table 13-3 (Channel Interface Register Map) Channel Interface Register Offset 05 bits [0],[3] |
| June 2011 | 02.0 | SERDES Functionality/ Electrical & Timing Characteristics | Deleted incorrect text in the SERDES Power Up section. |
| | | | External CML Reference Clock Specification (refclkp/refclkn) table - Deleted Frequency Tolerance, added Data/Referance Clock ppm offset tolerance. |
| | | Gigabit Ethernet Mode | Updated bullet list, exceptions to requirements. |
| | | | Updated Link State Machine paragraph |
| | | XAUI Mode | Updated Link State Machine paragraph |
| | | flexiPCS Testing | Added a note to the PRBS Generator and Checker section. |
| | | Memory Map | Updated ppm data in Quad Interface Register Map table, quad interface register offset address 0x30 |

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Lattice:

LFSCM3GA115EP1-6FC1704C LFSC3GA115E-6FC1704C LFSCM3GA115EP1-5FC1152C LFSC3GA115E-5FF1152C LFSC3GA115E-5FF1152I LFSC3GA115E-5FF1704C LFSC3GA115E-5FF1704I LFSC3GA115E-5FFN1152C LFSC3GA115E-5FFN1152I LFSC3GA115E-5FFN1704C LFSC3GA115E-5FFN1704I LFSC3GA115E-6FF1152C LFSC3GA115E-6FF1152I LFSC3GA115E-6FF1704C LFSC3GA115E-6FF1704I LFSC3GA115E-6FFN1152C LFSC3GA115E-6FFN1152I LFSC3GA115E-6FFN1704C LFSC3GA115E-6FFN1704I LFSC3GA115E-7FCN1152C LFSC3GA115E-7FCN1704C LFSC3GA25E-5FFA1020C LFSC3GA25E-5FFA1020I LFSC3GA25E-5FFAN1020C LFSC3GA25E-5FFAN1020I LFSC3GA25E-6FFA1020C LFSC3GA25E-6FFA1020I LFSC3GA25E-6FFAN1020C LFSC3GA25E-6FFAN1020I LFSC3GA25E-7FFA1020C LFSC3GA25E-7FFAN1020C LFSC3GA40E-5FF1152C LFSC3GA40E-5FF1152I LFSC3GA40E-5FFA1020C LFSC3GA40E-5FFA1020I LFSC3GA40E-5FFAN1020C LFSC3GA40E-5FFAN1020I LFSC3GA40E-5FFN1152C LFSC3GA40E-5FFN1152I LFSC3GA40E-6FF1152C LFSC3GA40E-6FF1152I LFSC3GA40E-6FFA1020C LFSC3GA40E-6FFA1020I LFSC3GA40E-6FFAN1020C LFSC3GA40E-6FFAN1020I LFSC3GA40E-6FFN1152C LFSC3GA40E-6FFN1152I LFSC3GA40E-7FF1152C LFSC3GA40E-7FFA1020C LFSC3GA40E-7FFAN1020C LFSC3GA40E-7FFN1152C LFSC3GA80E-5FF1152C LFSC3GA80E-5FF1152I LFSC3GA80E-5FF1704C LFSC3GA80E-5FF1704I LFSC3GA80E-5FFN1152C LFSC3GA80E-5FFN1152I LFSC3GA80E-5FFN1704C LFSC3GA80E-5FFN1704I LFSC3GA80E-6FF1152C LFSC3GA80E-6FF1152I LFSC3GA80E-6FF1704C LFSC3GA80E-6FF1704I LFSC3GA80E-6FFN1152C LFSC3GA80E-6FFN1152I LFSC3GA80E-6FFN1704C LFSC3GA80E-6FFN1704I LFSC3GA80E-7FF1152C LFSC3GA80E-7FF1704C LFSC3GA80E-7FFN1152C LFSC3GA80E-7FFN1704C LFSCM3GA115EP1-5FF1152C LFSCM3GA115EP1-5FF1152I LFSCM3GA115EP1-5FF1704C LFSCM3GA115EP1-5FF1704I LFSCM3GA115EP1-5FFN1152C LFSCM3GA115EP1-5FFN1152I LFSCM3GA115EP1-5FFN1704C LFSCM3GA115EP1-5FFN1704I LFSCM3GA115EP1-6FF1152C LFSCM3GA115EP1-6FF1152I LFSCM3GA115EP1-6FF1704C LFSCM3GA115EP1-6FF1704I LFSCM3GA115EP1-6FFN1152C LFSCM3GA115EP1-6FFN1152I LFSCM3GA115EP1-6FFN1704C LFSCM3GA115EP1-6FFN1704I LFSCM3GA115EP1-7FCN1152C LFSCM3GA115EP1-7FCN1704C LFSCM3GA25EP1-5FFA1020C LFSCM3GA25EP1-5FFA1020I LFSCM3GA25EP1-5FFAN1020C LFSCM3GA25EP1-5FFAN1020I LFSCM3GA25EP1-6FFA1020C LFSCM3GA25EP1-6FFA1020I LFSCM3GA25EP1-6FFAN1020C LFSCM3GA25EP1-6FFAN1020I LFSCM3GA25EP1-7FFA1020C LFSCM3GA25EP1-7FFAN1020C LFSCM3GA40EP1-5FF1152C