LatticeECP3 SERDES/PCS 使用指南

2012 年 8 月 技术说明 TN1176

概述

LatticeECP3™ FPGA系列结合了高性能FPGA结构、高性能I/O和多达16个通道带有相应的物理编码子层（Physical Coding Sublayer， PCS）逻辑的嵌入式 SERDES。 PCS 逻辑可配置用于支持很多业界标准的、高速串行数据传输协议。

PCS 逻辑的每个通道包含专用的发送和接收 SERDES，可用于高达 3.2 Gbps 的高速、全双工串行数据传输。每个通道的 PCS 逻辑可通过配置来支持一系列常用的数据协议，包括 GbE、XAUI、SONET/SDH、PCI Express、SRIO、CPRI、 OBSAI、 SD-SDI、 HD-SDI 和 3G-SDI。此外，基于协议的逻辑可以在许多配置中被完全或部分绕过，从而使用户在设计自己的高速数据接口时拥有更大的灵活性。

PCS 还提供旁路模式 （bypass mode），允许使用 8 位或 10 位接口将 SERDES 直接连接到 FPGA 逻辑。每个 SERDES 引脚还可以独立实施直流耦合并且允许在同一个 SERDES 引脚上同时支持高速和低速工作，适用于如串行数字视频等应用。

特性

- 多达 16 个通道的高速 SERDES
  - 150 Mbps 至 3.2 Gbps 用于通用 8b10b， 10 位 SERDES 和 8 位 SERDES 模式。参见表 8-1。
  - 所有其他的协议， 230 Mbps 至 3.2 Gbps/ 通道
  - 3.2 Gbps 工作时，功耗低至 110mW/ 通道
  - 适用于小尺寸背板工作的接收均衡和发送预加重
  - 支持 PCI Express、千兆以太网 （1GbE 和 SGMII）、 XAUI 以及多种其他标准
  - 支持用户定义的通用 8b10b 模式
  - 适用于低速输入 （视频应用）的带外 （Out-of-band， OOB）信号接口
- 多种时钟速率支持
  - 每个 PCS quad 独立的参考时钟，便于在一个器件内处理多种协议速率
- 全功能嵌入式物理编码子层 （PCS）逻辑，支持业界标准协议
  - 每个器件支持多达 16 个通道的全双工数据传输
  - 单芯片支持多种协议
  - 支持通用的基于 8b10b 的数据包协议
  - SERDES only 模式允许 8 位或 10 位接口直接连接到 FPGA 逻辑
- 兼容多种协议的时钟容限补偿 （Clock Tolerance Compensation， CTC）逻辑
  - 对参考时钟和接收数据速率之间的频率差异进行补偿
  - 允许用户定义的 1、 2 或 4 字节长度的跳跃模式
- 集成了回环模式 （Loopback mode)，用于系统调试
  - 用于系统调试的三种回环模式

LatticeECP2M™ 所没有的 SERDES/PCS 的新特性

- 在一个SERDES quad中支持多种协议 / 标准。这些标准需支持如表 8-1 中所列标准的全速或半速标称频率。配置灵活性不应成为支持不同混合的协议和标准的障碍。多协议组中支持 PCI Express、千兆以太网、 SGMII 和串行 RapidIO 模式。
- 支持兼容 XAUI 的功能并将 SERDES 的最高性能扩展至 3.2 Gbps。

- 支持 SONET/SDH OC-3/STM-1、 OC-12/STM-4 和 OC-48/STM-16 速率。
- 增加了对于 SD-SDI、 HD-SDI 和 3G-SDI 的每个 RX 和 TX DIV11 的支持。多速 SDI 支持。

本技术说明的使用

莱迪思的 ispLEVER 设计工具支持所有的 PCS 模式。大多数模式都是专为特定行业的标准数据协议应用而设计。其他模式则适用于更通用的目的，让设计人员能对他们自己的应用设置进行定义。ispLEVER 设计工具允许用户在自己的设计中为每个 quad 定义工作模式。本技术说明介绍的 SERDES 和 PCS 的所有模式都得到 ispLEVER 软件的支持。如果您正在使用 Lattice Diamond™ 设计软件，请参见附录 D。

本文档提供了嵌入式 SERDES 和相关 PCS 逻辑的所有功能的全部说明。 LatticeECP3 系列数据手册提供了嵌入式 SERDES 的电气特性和时序特性。本文档 PCS 章节提供了 PCS 逻辑的工作情况。附录给出了可通过 SCI 总线访问的有关 SERDES 和 PCS 逻辑的所有状态和控制寄存器列表。 LatticeECP3 系列数据手册的引脚布局信息章节给出了封装的引脚布局信息。

支持的标准

表 8-1 中列出了支持的标准。

表 8-1. SERDES 支持的标准

标准	数据速率（Mbps）	系统参考时钟（MHz）	FPGA 时钟（MHz）	通用 / 链接宽度的数量	编码类型
PCI Express 1.1	2500	100	250	x1, x2, x4	8b10b
千兆以太网， SGMII	1250	125	125	x1	8b10b
	2500	125	250	x1	8b10b
	3125	156.25	156.25	x1	8b10b
XAUI	3125	156.25	156.25	x4	8b10b
串行 RapidIO Type I, 串行 RapidIO Type II, 串行 RapidIO Type III	1250, 2500, 3125	125, 125, 156.25	125, 250, 156.25	x1, x4	8b10b
OBSAI-1, OBSAI-2, OBSAI-3, OBSAI-4	768, 1536, 2304, 3072	76.8, 76.8, 153.6, 115.2, 153.6	76.8, 153.6, 230.4, 153.6	x1	8b10b
CPRI-1, CPRI-2, CPRI-3, CPRI-4	614.4, 1228.8, 2457.6, 3072.0	61.44, 61.44, 122.88, 122.88, 153.6	61.44, 122.88, 122.88 153.6	x1	8b10b
SD-SDI (259M, 344M)	143[1], 177[1], 270, 360, 540	14.3[1], 17.7[1], 27, 36, 54	143, 177, 27, 36, 54	x1	NRZI/ 扰码
HD-SDI (292M)	1483.5, 1485	74.175, 148.35, 74.25, 148.50	74.175, 148.35, 74.25, 148.5	x1	NRZI/ 扰码
3G-SDI (424M)	2967, 2970	148.35, 148.5	148.35, 148.5	x1	NRZI/ 扰码
SONET STS-3[2] SONET STS-12[2] SONET STS-48[2]	155.52 622.08 2488	15.552 62.208 248.8	15.552 62.208 248.8	x1	N/A
10 位 SERDES	150 - 3125	15 - 312.5	15 - 312.5	x1, x2, x3, x4	N/A

表 8-1. SERDES 支持的标准 （续）

标准	数据速率（Mbps）	系统参考时钟（MHz）	FPGA 时钟（MHz）	通用 / 链接宽度的数量	编码类型
8 位 SERDES	150 - 3125	15 - 312.5	15 - 312.5	x1, x2, x3, x4	N/A
通用 8b10b	150 - 3125	15 - 312.5	15 - 312.5	x1, x2, x3, x4	8b10b

1. 对于较低速率而言，可以绕过 SERDES，将信号可直接传入 FPGA 内核。
2. 8 位 SERDES 模式支持 SONET 协议。请参考本文档的 SONET 章节，了解更多详细信息。

架构概述

quad 中的 SERDES/PCS 模块包含逻辑用于 4 个独立的全双工数据通道。

图 8-1 展示了 LatticeECP3-150 FPGA 中 SERDES/PCS quad 的布局 （其他器件有较少的 quad）。

图 8-1. LatticeECP3-150 框图

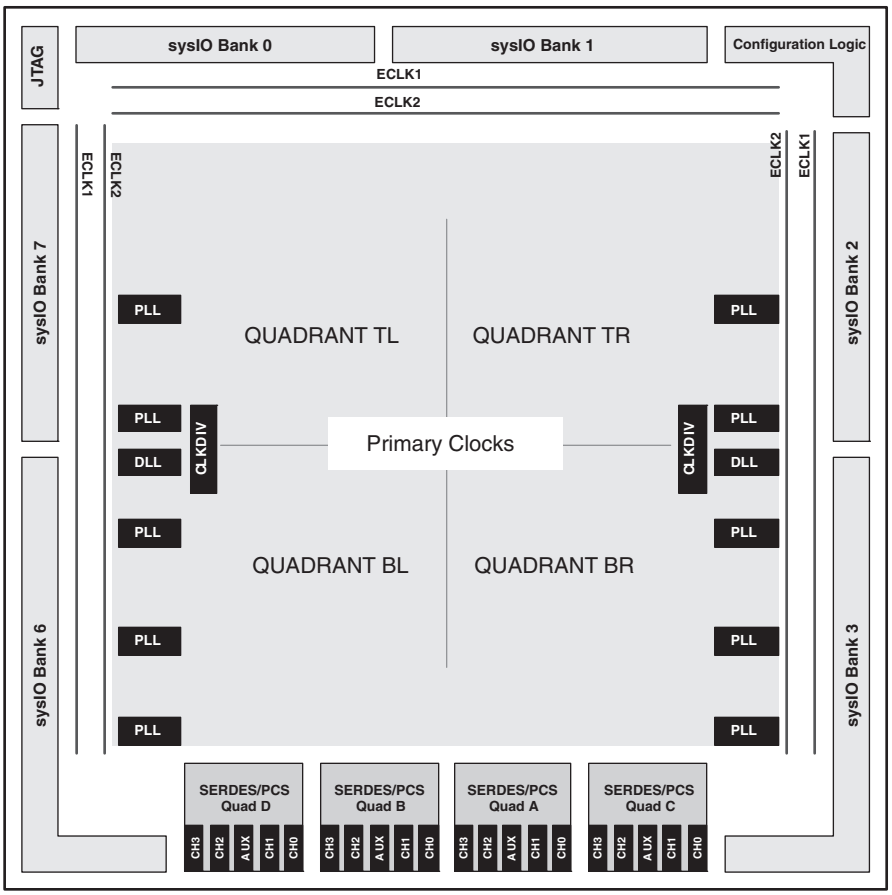


表 8-2 展示了 LatticeECP3 系列中每款器件可用的 SERDES/PCS quad 数量。

表 8-2. 每款 LatticeECP3 器件的 SERDES/PCS Quad 数量

封装	ECP3-17	ECP3-35	ECP3-70	ECP3-95	ECP3-150
256 球型 ftBGA	1	1	—	—	—
328 球型 csBGA	2 通道[1]	—	—	—	—
484 球型 ftBGA	1	1	1	1	
672 球型 ftBGA	—	1	2	2	2
1156 球型 ftBGA	—	—	3	3	4

1. Channel 0 和 3 可用。

每个 quad 可通过编程使用几种基于协议的模式中的一种。每个 quad 需要自己的参考时钟，可通过封装引脚连接到外部时钟源或连接到 FPGA 内部逻辑的时钟源。

每个 quad 可根据所选的支持标称频率的协议进行编程，并可选用每个通道的全速或半速选项。例如，可以在同一个 quad 中使用 PCI Express x1@2.5Gbps 和千兆以太网通道，在千兆以太网通道中使用半速选项。如果一个 quad 与非 PCI Express 通道共享一个 PCI Express x1 通道，请确保这个 quad 的参考时钟兼容这个 quad 内的所有协议。例如：PCI Express 扩频参考时钟与大多数千兆以太网应用不兼容。

因为每个 quad 都有自己的参考时钟，在同一块芯片上不同的 quad 可支持不同标准。这一特性使得 LatticeECP3 系列器件成为桥接不同标准的理想器件。

PCS quad 并不仅限用于业界标准协议。每个 quad （以及一个 quad 内的每个通道）可编程用于许多用户定义的数据操作模式。例如，字对齐和时钟容限补偿可通过编程用于用户自定义操作。

PCS Quad 和通道

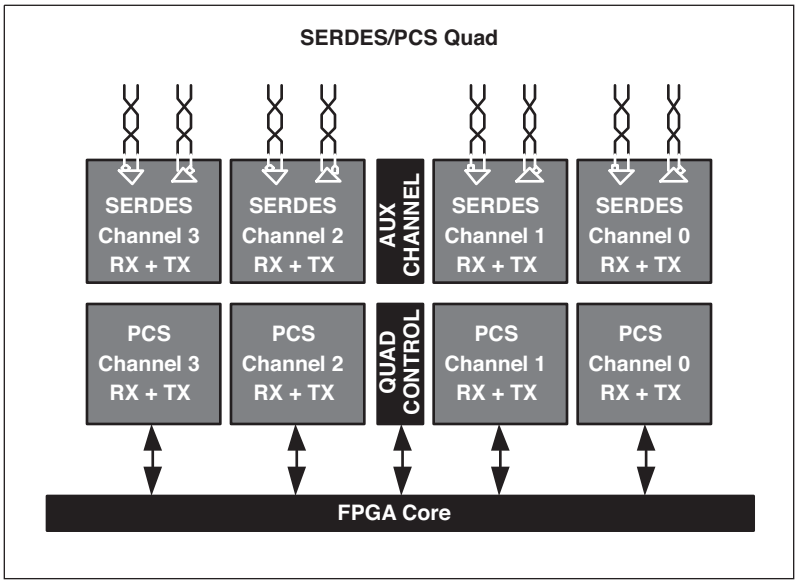
器件上的每个 quad 支持多达 4 个全双工数据通道。根据不同应用，可以使用一个 quad 内的一至四个通道。在一个给定的 quad 中，用户可以为每个通道分别设置许多选项。

图 8-1 展示了带有四个 PCS quad 的器件示例，包含了总共 16 个 PCS 通道。

每个通道的 SERDES/PCS 和 FPGA 接口端口

所有 PCS quad，无论所选的模式，都在封装引脚上有相同的外部高速串行接口。然而，每个 PCS 模式都有其唯一与 FPGA 逻辑作接口的输入 / 输出端口列表，这些端口是与 quad 所选协议相适应的。本文档提供了每个模式下该 quad 的输入 / 输出信号的详细说明。图 8-2 展示了一个简化的 SERDES/PCS quad。

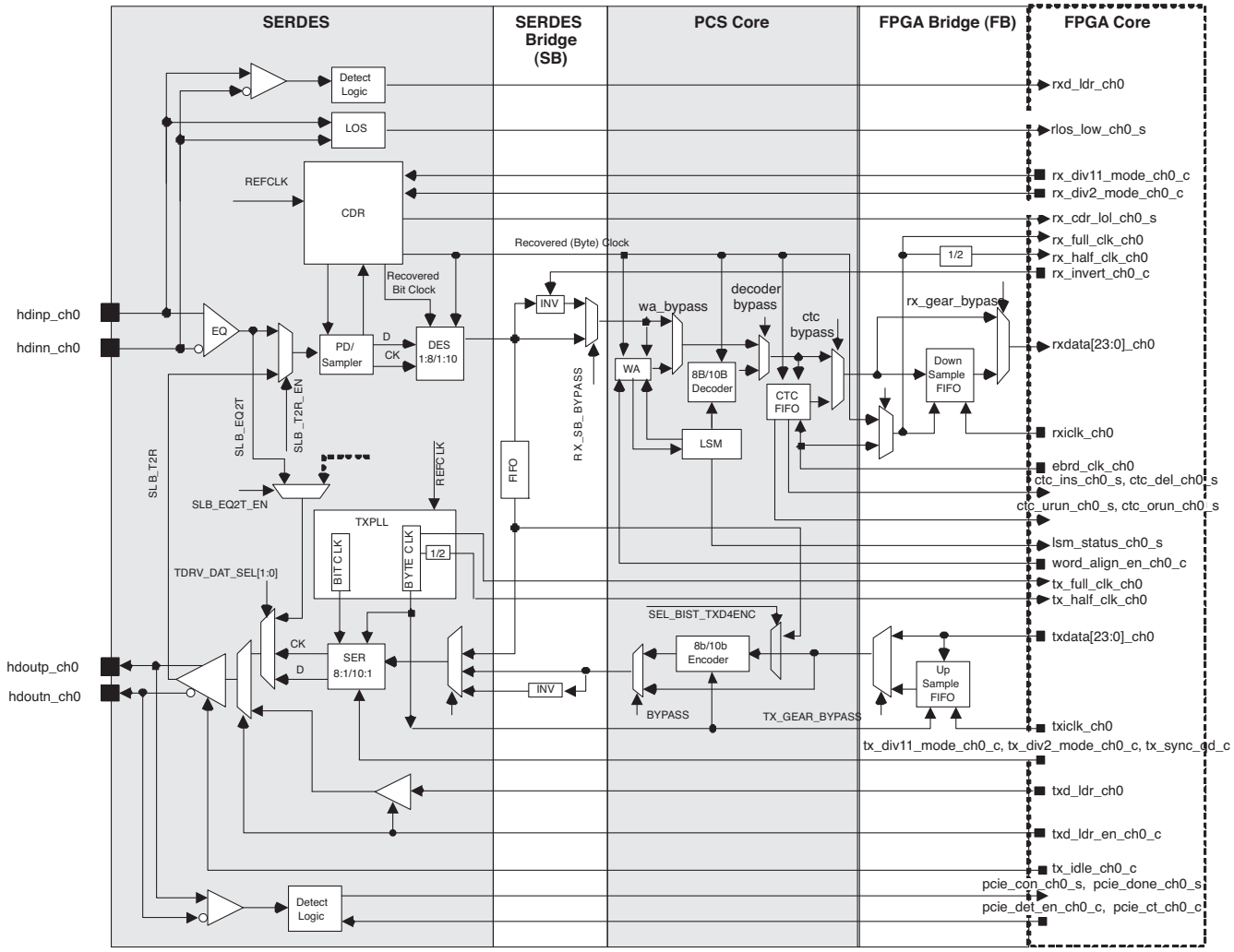
图 8-2. SERDES/PCS Quad 框图



详细的通道框图

图 8-3 是 LatticeECP3 SERDES/PCS 的单通道主要功能的详细框图说明。该图展示了对 FPGA 中的用户逻辑来说可视的所有主要模块和主要控制和状态信号。该图还展示了通道中的主要子模块 ——SERDES、SERDES 桥、PCS 内核以及 FPGA 桥。

图 8-3. LatticeECP3 SERDES/PCS 详细的通道框图



时钟和复位

一个 PCS quad 提供每个通道锁定的参考时钟和每个通道恢复的接收时钟，连到 FPGA 逻辑接口。每个 PCS quad 为主和次 FPGA 时钟走线提供时钟。PCS/FPGA 接口还为每个 quad 的四个通道提供来自 FPGA 逻辑的发送和接收时钟端口。

每个 quad 有复位输入来强制 quad 内的 SERDES 和 PCS 逻辑或仅 SERDES 复位。此外，还为每个通道的发送和接收提供专用于 PCS 逻辑的独立复位。

发送数据总线

在 PCS 块中，发送数据通道的信号是从 FPGA 到 FPGA 桥，数据宽度可调整为 2:1 进入内部 PCS 数据通道，具有 8 位数据宽度（外加控制 / 状态信号）。在 1:1 模式下，PCI Express x1 接口的最高速率是 250 MHz。在 2:1 gearing（如 16 位宽数据通道）模式下，可能的速率为 156.25 MHz（XAUI 4x 通道模式）。SERDES 和 PCS 将支持数据速率高达 3.2 Gbps 的数据，对应接口速率为 160 MHz （2:1 gearing 情况下）。

接收数据总线

接收路径的信号是从 PCS 块的 FPGA 桥到 FPGA。数据路径可调整 （gearing）为 2:1 至数据宽度为 8 位的内部 PCS 数据路径。 FPGA 接口的数据总线宽度为 16 位宽。可通过软件的寄存器位禁用 2:1 gearing 功能，这样的话，总线宽度减半 （8 位宽）。当数据调整为 2:1 ，低位 （rxdata[9:0]）对应先接收的字，高位 （rxdata[19:10]）对应第二个接收的字。如果数据没有调整为 2:1，低位 （rxdata[9:0]）为有效位，并且高位不使用。表 8-3 说明了每个协议模式下数据总线的使用。

表 8-3. 模式使用的数据总线

数据总线 PCS 单元名称[4]	G8B10B	CPRI	OBSAI	PCI Express	SRIO	千兆以太网	XAUI	8 位 SERDES	10 位 SERDES	SDI
FF_TX_D_0_0	txdata_ch0[0]									
FF_TX_D_0_1	txdata_ch0[1]									
FF_TX_D_0_2	txdata_ch0[2]									
FF_TX_D_0_3	txdata_ch0[3]									
FF_TX_D_0_4	txdata_ch0[4]									
FF_TX_D_0_5	txdata_ch0[5]									
FF_TX_D_0_6	txdata_ch0[6]									
FF_TX_D_0_7	txdata_ch0[7]									
FF_TX_D_0_8	tx_k_ch0[0]						txc_ch0[0]	GND	txdata_ch0[8]	
FF_TX_D_0_9	tx_force_disp_ch0[0][1]				GND				txdata_ch0[9]	
FF_TX_D_0_10	tx_disp_sel_ch0[0][1]				GND	xmit_ch0[0][2]	GND			
FF_TX_D_0_11	GND			pci_ei_en_ch0[0]	GND	tx_disp_correct_ch0[0]	GND			
FF_TX_D_0_12	txdata_ch0[8]								txdata_ch0[10]	
FF_TX_D_0_13	txdata_ch0[9]								txdata_ch0[11]	
FF_TX_D_0_14	txdata_ch0[10]								txdata_ch0[12]	
FF_TX_D_0_15	txdata_ch0[11]								txdata_ch0[13]	
FF_TX_D_0_16	txdata_ch0[12]								txdata_ch0[14]	
FF_TX_D_0_17	txdata_ch0[13]								txdata_ch0[15]	
FF_TX_D_0_18	txdata_ch0[14]								txdata_ch0[16]	
FF_TX_D_0_19	txdata_ch0[15]								txdata_ch0[17]	
FF_TX_D_0_20	tx_k_ch0[1]						txc_ch0[1]	GND	txdata_ch0[18]	
FF_TX_D_0_21	tx_force_disp_ch0[1][1]				GND				txdata_ch0[19]	
FF_TX_D_0_22	tx_disp_sel_ch0[1][1]				GND	xmit_ch0[1][2]	GND			
FF_TX_D_0_23	GND			pci_ei_en_ch0[1]	GND	tx_disp_correct_ch0[1]	GND			
FF_RX_D_0_0	rxdata_ch0[0]									
FF_RX_D_0_1	rxdata_ch0[1]									
FF_RX_D_0_2	rxdata_ch0[2]									
FF_RX_D_0_3	rxdata_ch0[3]									
FF_RX_D_0_4	rxdata_ch0[4]									
FF_RX_D_0_5	rxdata_ch0[5]									
FF_RX_D_0_6	rxdata_ch0[6]									
FF_RX_D_0_7	rxdata_ch0[7]									
FF_RX_D_0_8	rx_k_ch0[0]						rxc_ch0[0]	NC	rxdata_ch0[8]	
FF_RX_D_0_9	rx_disp_err_ch0[0]			rxstatus0_ch0[0]	rx_disp_err_ch0[0]			NC	rxdata_ch0[9]	
FF_RX_D_0_10	rx_cv_err_ch0[0][3]			rxstatus0_ch0[1]	rx_cv_err_ch0[0][3]			NC		
FF_RX_D_0_11	NC			rxstatus0_ch0[2]	NC					
FF_RX_D_0_12	rxdata_ch0[8]								rxdata_ch0[10]	
FF_RX_D_0_13	rxdata_ch0[9]								rxdata_ch0[11]	
FF_RX_D_0_14	rxdata_ch0[10]								rxdata_ch0[12]	
FF_RX_D_0_15	rxdata_ch0[11]								rxdata_ch0[13]	
FF_RX_D_0_16	rxdata_ch0[12]								rxdata_ch0[14]	
FF_RX_D_0_17	rxdata_ch0[13]								rxdata_ch0[15]	
FF_RX_D_0_18	rxdata_ch0[14]								rxdata_ch0[16]	
FF_RX_D_0_19	rxdata_ch0[15]								rxdata_ch0[17]	

表 8-3. 模式使用的数据总线 （续）

数据总线 PCS 单元名称[4]	G8B10B	CPRI	OBSAI	PCI Express	SRIO	千兆以太网	XAUI	8 位 SERDES	10 位 SERDES	SDI
FF_RX_D_0_20	rx_k_ch0[1]						rxc_ch0[1]	NC	rxdata_ch0[18]	
FF_RX_D_0_21	rx_disp_err_ch0[1]			rxstatus1_ch0[0]	rx_disp_err_ch0[1]			NC	rxdata_ch0[19]	
FF_RX_D_0_22	rx_cv_err_ch0[1][3]			rxstatus1_ch0[1]	rx_cv_err_ch0[1][3]			NC		
FF_RX_D_0_23	NC			rxstatus1_ch0[2]	NC					

1. force_disp 信号将强制相关数据字的位 [7:0] 在列上的不一致，而 （disparity）值由 tx_disp_sel 信号来选择。如果 disp_sel 是 1， 10 位编码将为 " 当前 RD+" 列（即 positive disparity）。如果 tx_disp_sel 为 0， 10 位编码将为 " 当前 RD-" 列 （即 negative disparity）。
2. 莱迪思千兆以太网 PCS IP 核可提供一个自协商的状态机，生成信号 xmit。它用于与千兆以太网硬逻辑空闲状态机进行交互。
3. 当出现编码违例，数据包 PCS 8b10b 解码器将使用 d=hex EE 和 K=1 （K=1 且 d=EE 不在 8b10b 编码空间内）替代解码器的输出。
4. FF_TX_D_0_0：FPGA 结构发送数据总线 Channel 0 Bit 0。

特定模式控制 / 状态信号说明

表 8-4 说明了特定模式下的控制 / 状态信号。

表 8-4. 控制信号及其功能

信号名称	说明
发送控制信号	
tx_k_ch[3:0]	每个通道，高电平有效控制字符指示。
tx_force_disp_ch[3:0]	每个通道，高电平有效，命令 PCS 接受来自 disp_sel_ch(0-3) FPGA 接口输入的不一致 (disparity) 值。
tx_disp_sel_ch[3:0]	每个通道， FPGA 逻辑提供的不一致 (disparity) 值。当 force_disp_ch(0-3) 为高电平时有效。
tx_correct_disp_ch[3:0]	当有效时，通过调整 8b10b 编码器以取 negative disparity 状态开始来更正 (disparity) 指示符。
接收状态信号	
rx_k_ch[3:0]	每个通道，高有效控制字符指示。
rx_disp_err_ch[3:0]	每个通道， PCS 发出的高有效信号，用以指示相应数据检测到的 (disparity) 错误。
rx_cv_err_ch[3:0]	每个通道，编码违例信号，用以指示相应数据检测到的错误。

控制

每个模式下有其各自的一套控制信号，允许从 FPGA 逻辑直接控制各种 PCS 功能。通常，这些控制输入的每个信号复制写入相应的控制寄存器的一个或多个位中。

{signal}_c 是从 FPGA 内核到 FPGA 桥的控制信号。所有控制信号均在 SERDES/PCS 内异步使用。

状态

每个模式下有其各自的一套状态或报警信号，通过 FPGA 逻辑进行监控。通常，这些状态输出都对应于特定状态寄存的一个或多个位。 Diamond 设计工具给了用户将这些端口输出到 PCS FPGA 接口的选择。

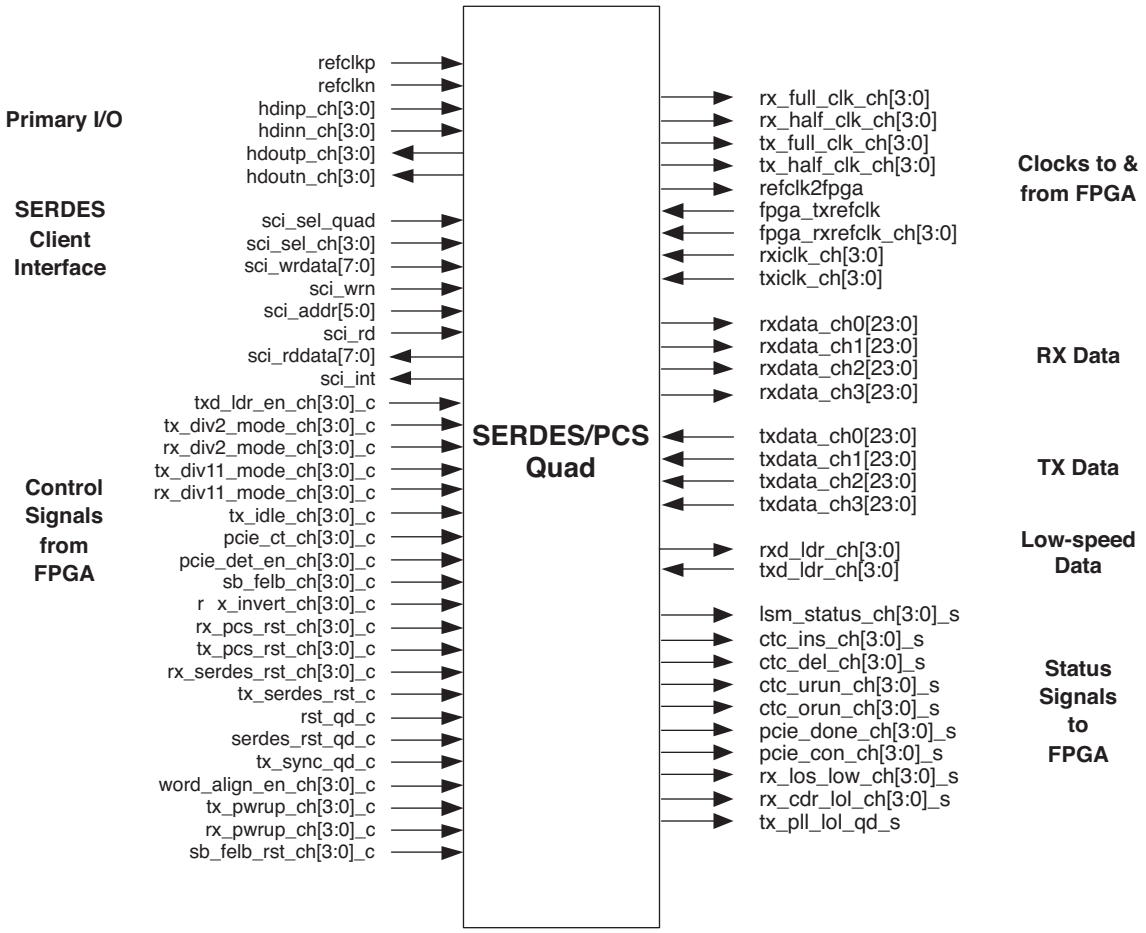
{signal}_s 是从 FPGA 桥到 FPGA 内核的状态信号。来自 SERDES/PCS 的所有状态信号都是异步的。他们都必须在 FPGA 设计使用这些信号之前在时钟域进行同步。

请参考本文档的特定模式控制 / 状态信号章节，了解有关控制和状态信号的详细信息。

SERDES/PCS

quad 包含 4 个通道，每个通道都有 RX 和 TX 电路，以及一个包含 TX PLL 的辅助通道。主差分参考时钟引脚或 FPGA 内核都可以提供 TX PLL 的参考时钟 。quad SERDES/PCS 宏为 4 条数据通道执行串行化或解串功能。此外，SERDES/PCS 块的 TxPLL 为 FPGA 逻辑提供系统时钟。quad 还在每个 TX 和 RX 电路上独立支持全数据速率和半数据速率模式。框图级图如图 8-4 所示。

图 8-4. SERDES/PCS 块信号接口



I/O 说明

表 8-5 列出了所有（至 / 从）PCS quad 的默认和可选择的输入和输出。用户可使用 IPexpress™ GUI 选择可选的端口用于 PCS quad。

表 8-5. SERDES_PCS I/O 说明

信号名称	I/O	类型	说明
主 I/O、 SERDES Quad			
hdinp_ch0	I	通道	高速 CML 输入、正极、 channel 0
hdinn_ch0	I	通道	高速 CML 输入、负极、 channel 0
hdinp_ch1	I	通道	高速 CML 输入、正极、 channel 1
hdinn_ch1	I	通道	高速 CML 输入、负极、 channel 1
hdinp_ch2	I	通道	高速 CML 输入、正极、 channel 2
hdinn_ch2	I	通道	高速 CML 输入、负极、 channel 2
hdinp_ch3	I	通道	高速 CML 输入、正极、 channel 3
hdinn_ch3	I	通道	高速 CML 输入、负极、 channel 3
hdoutp_ch0	O	通道	高速 CML 输入、正极、 channel 0
hdoutn_ch0	O	通道	高速 CML 输出、负极、 channel 0
hdoutp_ch1	O	通道	高速 CML 输入、正极、 channel 1
hdoutn_ch1	O	通道	高速 CML 输出、负极、 channel 1
hdoutp_ch2	O	通道	高速 CML 输入、正极、 channel 2
hdoutn_ch2	O	通道	高速 CML 输出、负极、 channel 2
hdoutp_ch3	O	通道	高速 CML 输入、正极、 channel 3
hdoutn_ch3	O	通道	高速 CML 输出、负极、 channel 3
refclkp	I	Quad	参考时钟输入、正极、专用 CML 输入
refclkn	I	Quad	参考时钟输入、负极、专用 CML 输入
接收 / 发送数据总线 （参见表 8-3 和 8-4，了解有关数据总线的详细使用信息）			
rxdata_ch0[23:0]	O	通道	用于 channel 0 接收路径的数据信号
rxdata_ch1[23:0]	O	通道	用于 channel 1 接收路径的数据信号
rxdata_ch2[23:0]	O	通道	用于 channel 2 接收路径的数据信号
rxdata_ch3[23:0]	O	通道	用于 channel 3 接收路径的数据信号
txdata_ch0[23:0]	I	通道	用于 channel 0 发送路径的数据信号
txdata_ch1[23:0]	I	通道	用于 channel 1 发送路径的数据信号
txdata_ch2[23:0]	I	通道	用于 channel 2 发送路径的数据信号
txdata_ch3[23:0]	I	通道	用于 channel 3 发送路径的数据信号
控制信号			
tx_idle_ch[3:0] _c	I	通道	通过 SERDES 发送器发出控制传输的电气空闲信号。 1 = 命令 SERDES 发送器输出电气空闲信号 0 = 正常操作
pcie_det_en_ch[3:0]_c	I	通道	FPGA 逻辑（用户逻辑）通知 SERDES 块，它将申请用于 PCI Express 接收器检测工作。 1 = 使能 PCI Express 接收器检测， 0 = 正常操作
pcie_ct_ch[3:0]_c	I	通道	1 = 申请发送器进行远端接收器检测 0 = 正常数据操作
rx_invert_ch[3:0]_c	I	通道	控制接收数据求反。 1 = 数据取反， 0 = 无需数据取反

表 8-5. SERDES_PCS I/O 说明 （续）

信号名称	I/O	类型	说明
word_align_en_ch[3:0]_c	I	通道	控制 comma 字符 1 = 使能 comma 对齐功能， 0 = 锁定字对齐位置为当前位置
sb_felb_ch[3:0]_c	I	通道	SERDES 桥接并行回环 1 = 使能回环从 RX 至 TX， 0 = 正常数据工作
sb_felb_rst_ch[3:0]_c	I	通道	SERDES 桥接并行回环 FIFO 清除 1 = 复位回环 FIFO, 0 = 正常回环操作
tx_sync_qd_c	I	Quad	串行器复位 跳变 （Transition） = 复位，不变 （Level） = 正常工作
rx_div2_mode_ch[3:0]_c	I	通道	接收器速率模式选择 （半速 / 全速） 1 = 半速， 0 = 全速
tx_div2_mode_ch[3:0]_c	I	通道	发送器速率模式选择 （半速 / 全速） 1 = 半速， 0 = 全速
rx_div11_mode_ch[3:0]_c	I	通道	接收器速率模式选择 （Div11/ 全速） 1 = Div11， 0 = 全速
tx_div11_mode_ch[3:0]_c	I	通道	发送器速率模式选择 （Div11/ 全速） 1 = Div11， 0 = 全速
txd_ldr_en_ch{3:0]_c	I	通道	使能低数据速率 TX 串行路径 1 = 使能， 0 = 禁用
复位信号			
rx_pcs_rst_ch[3:0]_c	I	通道	高电平有效，异步输入。仅在 PCS 中复位单独的接收通道逻辑。
tx_pcs_rst_ch[3:0]_c	I	通道	高电平有效，异步输入。仅在 PCS 中复位单独的发送通道逻辑。
rx_serdes_rst_ch[3:0]_c	I	通道	高电平有效。在 SERDES 接收通道中复位选择的数字逻辑。
tx_serdes_rst_c	I	Quad	高电平有效。在所有 SERDES 发送通道中复位选择的数字逻辑。
rst_qd_c	I	Quad	高电平有效，异步输入。复位所有 SERDES 通道，包括辅助通道和 PCS。
serdes_rst_qd_c	I	Quad	高电平有效，异步输入至 SERDES quad。复位所有 SERDES 通道，包括 Quad 通道但非 PCS 逻辑。
tx_pwrup_ch[3:0]_c	I	通道	1 = 高电平有效，发送通道上电后。 0 = 发送通道断电。
rx_pwrup_ch[3:0]_c	I	通道	1 = 高电平有效，接收通道上电后。 0 = 接收通道断电。 .
状态信号			
pcie_done_ch[0:3]_s	O	通道	1 = 远端接收器检测完成 0 = 远端接收器检测未完成
pcie_con_ch[3:0]_s	O	通道	远端接收器检测结果。 1 = 检测到远端接收器 0 = 未检测到远端接收器
rx_los_low_ch[3:0]_s	O	通道	每个通道的信号丢失 （LO 阈值范围）检测。
lsm_status_ch[3:0]_s	O	通道	1 = 数据通道已与 comma 字符同步 0 = 数据通道未找到 comma 字符
ctc_urrun_ch[3:0]_s	O	通道	1 = 接收时钟补偿器 FIFO 欠载运行错误 0 = 无 FFIFO 错误
ctc_orun_ch[3:0]_s	O	通道	1 = 接收时钟补偿器 FIFO 过载错误 0 = 无 FIFO 错误
rx_cdr_lol_ch[3:0]_s	O	通道	1 = 接收 CDR 锁定丢失 0 = 保持锁定
tx_pll_lol_qd_s	O	Quad	1 = 发送 PLL 锁定丢失 0 = 保持锁定
ctc_ins_ch[3:0]_s	O	通道	1 = CTC 添加的字符 SKIP
ctc_del_ch[3:0]_s	O	通道	1 = CTC 删除的字符 SKIP
rx_cdr_trained_ch[3:0]_s	O	通道	1 = 说明 CDR_TRAIN_DIV 路径已经训练了 CDR

表 8-5. SERDES_PCS I/O 说明（续）

信号名称	I/O	类型	说明
FPGA 接口时钟			
rx_full_clk_ch[3:0]	O	通道	接收通道恢复时钟。在用户模式下，时钟源总是通道的恢复时钟。对于 10 GbE 等标准，支持时钟补偿，时钟源是相应发送通道的系统时钟。对于 PCS 旁路模式，它也是发送系统时钟，因此要求 raw 模式实际上通过 8b10b 模式下 8b10b 解码器禁止 （10 位或 20 位数据通道）来实现。
rx_half_clk_ch[3:0]	O	通道	接收通道恢复半速时钟。在 2:1 gearing 模式下，输出除以 2。
tx_full_clk_ch[3:0]	O	通道	TX PLL 全速时钟。仅 tx_full_clk_ch0 可直接驱动主时钟网络。所有 tx_full_clk_ch[3:0] 信号可以通过使用 USE SECONDARY 时钟参数驱动次时钟网络。[3]
tx_half_clk_ch[3:0]	O	通道	TX PLL 半速时钟。仅 tx_half_clk_ch0 可直接驱动主时钟网络。所有 tx_half_clk_ch[3:0] 信号可以通过使用 USE SECONDARY 时钟参数驱动次时钟网络。[3]
refclk2fpga	O	Quad	参考时钟至 FPGA 内核。选择该时钟后，只要参考时钟有效就一直保持有效，即使当 quad 为掉电模式。
fpga_rxrefclk_ch[3:0]	I	Quad	来自 FPGA 逻辑的 RX 参考时钟，用于 CDR PLL
fpga_txrefclk	I	Quad	来自 FPGA 逻辑的 TX 参考时钟，用于 TX SERDES PLL
ebrd_clk_ch[3:0][2]	I	通道	来自 FPGA 的接收通道时钟输入，用于 CTC FIFO 读。
rxiclk_ch[3:0]	I	通道	来自 FPGA 的接收通道时钟输入。用于使用与参考和 / 或接收参考时钟同步的时钟来为 RX FPGA Interface FIFO 提供时钟。
txiclk_ch[3:0]	I	通道	来自 FPGA 的发送通道时钟输入。每个通道来自 FPGA 发送的时钟输入。使用与参考时钟同步的时钟来为 TX FPGA Interface FIFO 提供时钟。在使用 CTC 时，还使用与参考时钟同步的时钟来为 RX FPGA Interface FIFO 提供时钟。
低速接收 / 发送数据和 SERDES 客户端接口信号			
rxd_ldr_ch[3:0]	O	通道	单端串行低数据速率输出 （RX）至 FPGA 内核。
txd_ldr_ch[3:0]	I	通道	来自 FPGA 内核的单端串行低数据速率输入 （TX）。
sci_wrdata[7:0]	I	—	写数据输入。
sci_wrn	I	—	写输入选通信号。
sci_sel_quad	I	—	选择 quad 寄存器。
sci_sel_ch[3:0]	I	—	选择通道寄存器。
sci_addr[5:0]	I	—	地址总线输入。
sci_rd	I	—	读数据选择。
sci_rddata[7:0]	O	—	读数据输出。
sci_int	O	—	中断输出。

1. 配置时， hdoutp 和 hdoutn 都拉高至 VCCOB。
2. wrapper 模块端口列表不提供该时钟。根据 CTC 模式，软件自动分配时钟。参见 FPGA 接口时钟使用章节，了解更多详细信息。
3. 通用布线用于访问次时钟网络。用户可能得到一个 PAR 警告，但是在大多数应用中这个延迟小到可以被忽略。使用时序参数并查看追踪报告，确保没有时序冲突。

SERDES/PCS 功能说明

LatticeECP3 器件有 1 到 4 个嵌入式 SERDES/PCS 逻辑 quad。每个 quad 依次支持 4 个独立的全双工数据通道。单通道可支持一个数据链路，每个 quad 最多可支持四个这样的通道。

嵌入式 SERDES CDR PLL 和 TX PLL 支持的数据速率覆盖各种行业标准协议。

参见图 8-3 了解下面所列的每项内容。

- **SERDES**
  - 均衡器
  - CDR （时钟和数据恢复）
  - 解串器
  - 预加重
  - 串行器
  - 两种串行回环模式， TX 至 RX 或 RX 至 TX
- **SERDES 桥 （SB）**
  - 反相器 —— 根据 PCI Express 的要求对接收数据取反。
  - SERDES 桥并行回环
- **PCS 核**
  - 字对齐
  - 8b10b 解码器
  - 8b10b 编码器
  - 链路状态机
  - 时钟容限补偿
- **FPGA 桥 （FB）**
  - 下采样 FIFO
  - 上采样 FIFO

SERDES

均衡器

随着数字传输的数据速率超过 Gbps 数量级，与频率相关的衰减导致接收信号受到严重的码间干扰，因而必须在数据收发器中使用一个均衡器来正确恢复数据。提供 6 个档位的频率范围：Mid_Low，Mid_Med，Mid_High，Long_Low，Long_Med， Long_High。

预加重

预加重是指一个系统过程，旨在相对于其它频率分量，增强某些频率的信号分量。目的就是尽量减少衰减差异这一现象所造成的不利影响，从而改善整体信号的信噪比。用户可以选择高达 80% 预加重。

参考时钟的使用

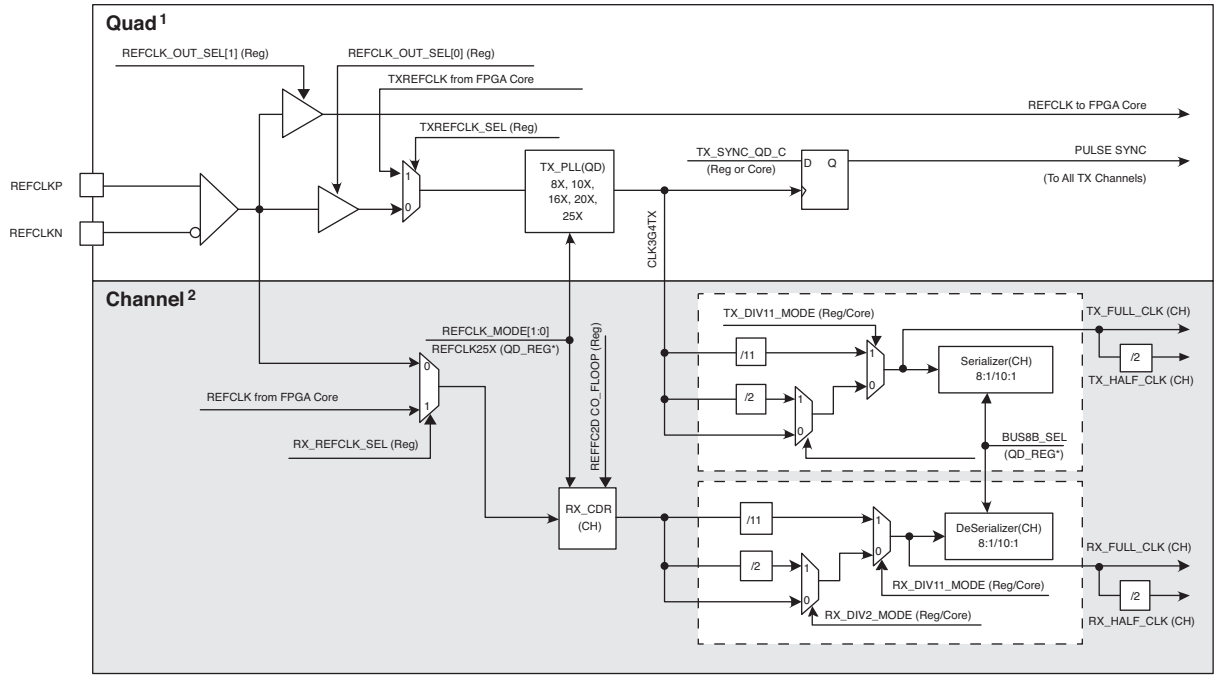
SERDES quad 包含 4 个带有 RX 和 TX 电路的通道，还有一个包含 TX PLL 的辅助通道。 TX PLL 的参考时钟可由主差分参考时钟引脚或相邻 quad 的参考时钟或 FPGA 内核提供。此外，SERDES 块中的 PLL 提供了输出时钟，可用于作为系统时钟来驱动 FPGA。

RX 的参考时钟可由 TX PLL 的参考时钟或 FPGA 内核提供。FPGA 内核为 TX PLL 和 RX 提供的参考时钟可能来自于不同的时钟源。

SERDES 时钟架构

图 8-5 显示了整个 SERDES 时钟架构。这个图分成 2 部分， Quad 和通道。为简洁起见，每条通道仅显示一条。此外，不同的模块的各个控制位如下图所示。这些可能是基于 quad 的控制寄存器位或基于通道的控制寄存器位。在某些情况下，它可以是基于通道控制端口的。有些是寄存器和控制端口两者的结合。使用这两种模式可实现某些特定功能特性的动态控制。

图 8-5. SERDES 时钟架构



1. All control bits are quad based.
2. All control bits are channel based, except as indicated (*).
3. These clocks are user-transparent.

时钟架构的主要部件包括：

- 每个 RX 和每个 TX 分频器 （DIV）模式 ——DIV2， DIV11
- 多个 quad REFCLK 连接
- 使用 FPGA 的 tx_sync_qd_c 信号的多通道发送同步
- OOB 低数据速率应用支持

速率模式

每个通道的 TX 可独立编程以下面某种速率运行：

- FULL_RATE
- HALF_RATE (DIV2)
- DIV11

RX 也可以每通道使用一个独立的参考时钟，使发送器和接收器在完全不同的速率下运行。

这里有一点很重要，请注意 PLL VCO 是未受影响的，它支持该协议的最高速率。该协议所需的所有分频速率都可以

通过对分频器多路开关选择进行适当的编程得到。由于 PLL 无需重新编程，从而实现了非常快速的数据速率转换。这对于许多应用来说是很有价值的。

注：LatticeECP3 PCS 不能在运行时进行速率改变。简单地改变 refclk 速率不会使 SERDES 在新的速率范围下工作。你需要重新对 SERDES 进行编程，并且这仅可能通过使用一个新的位流来实现。

TX PLL 和 4 个 CDR PLL 通常以相同频率工作，该频率为参考时钟频率的若干倍。表 8-6 显示了各种可能的时钟速率模式。列出的位时钟为参考时钟频率的倍数。

表 8-6. TXPLL 和 RX CDRPLL 支持的模式

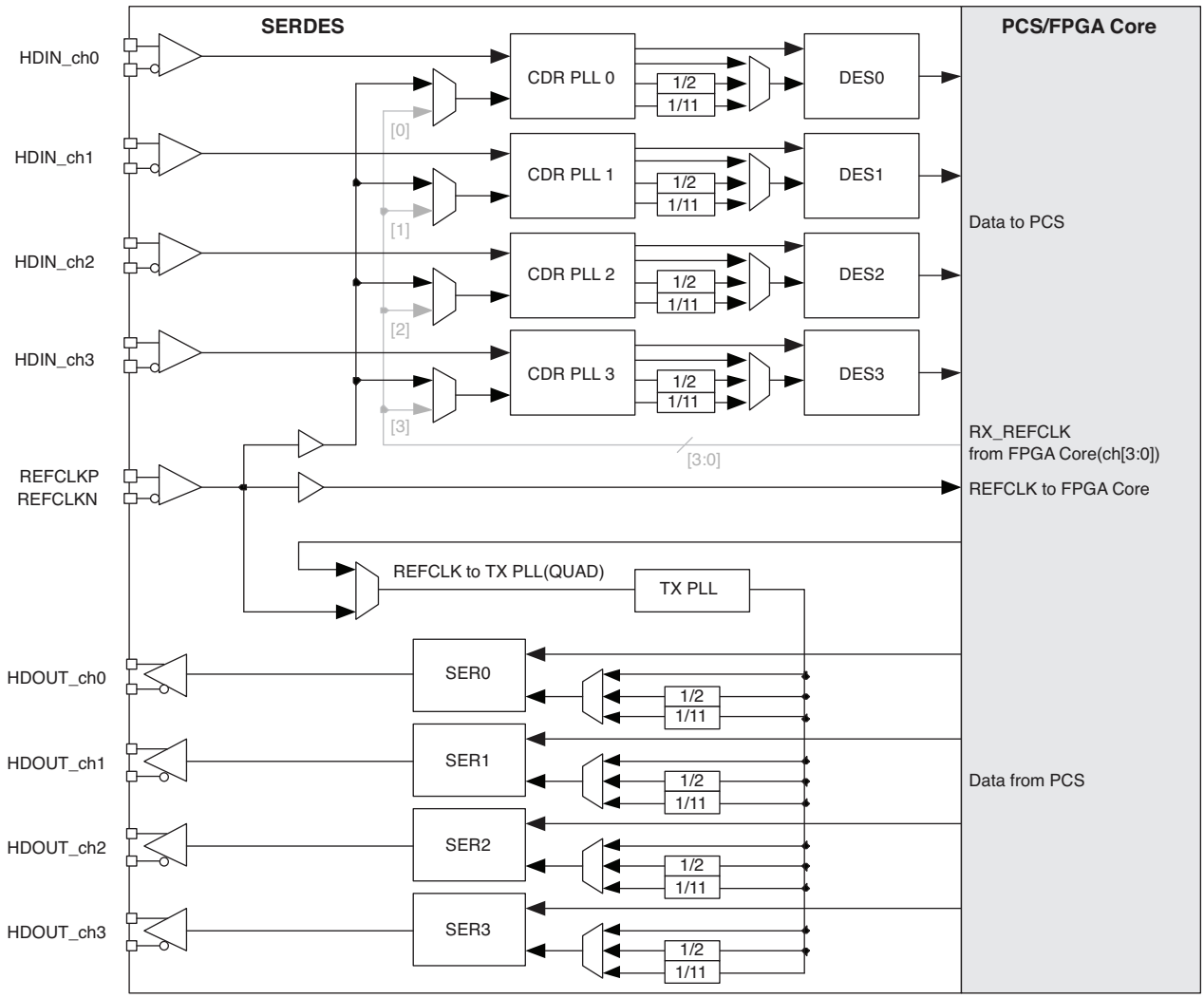
参考时钟模式	refclkPmode （Quad）	Bus_width	位时钟 （全速）	位时钟 （div2， div11）
20x	0	10	Refclk x 20	Refclk x 10
16x	0	8	Refclk x 16	Refclk x 8
10x	1	10	Refclk x 10	Refclk x 5
8x	1	8	Refclk x 8	Refclk x 4
25x	—	8	Refclk x 25	Refclk x 12.5
25x	—	10	Refclk x 25	Refclk x 12.5
20x	0	10	Refclk x 20	Refclk x 20/11[1]

1. DIV11 模式。

来自 FPGA 内核的参考时钟

如图 8-5 中所述，Tx 参考时钟可由 FPGA 内核提供。在这种情况下，FPGA 资源将时钟信号传给 SERDES 所引起的额外的抖动将传递给发送数据，就具体情况而言，可能会违反Tx抖动特性规范。当使用一个来自FPGA的SERDES Tx 参考时钟时需要加以注意。

图 8-6. 参考时钟的使用框图

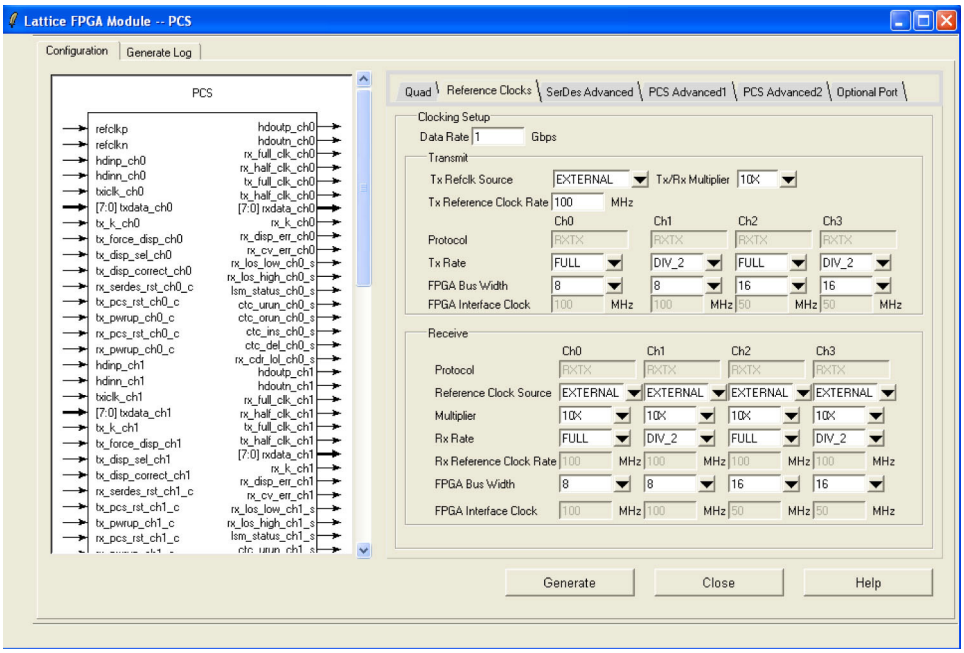


全速、 Div 2 和 Div 11 数据速率

每个 TX 串行器和 RX 解串器根据协议可分为全数据速率和 div2 速率或 div11 速率，在每个通道的每个方向上允许不同的数据速率。请参见图 8-6，了解更多信息。

如图 8-7 所示，四个通道可以进行不同的配置。

图 8-7. IPexpress GUI 的全速和半速示例



本示例中的实际数据速率和 FPGA 接口时钟速率如表 8-7 所示。 IPexpress GUI 将在本文档的后续章节中讨论。

表 8-7. 时钟速率示例

通道	数据速率	参考时钟倍频器	数据速率模式	计算的参考时钟速率	FPGA 接口数据总线宽度	FPGA 接口时钟速率	tx_full_clk	tx_half_clk
Channel 0	1 Gbps	10 x	FULL	100 MHz	8 (10)[3]	100 MHz	100 MHz	50 MHz
Channel 1	500 Mbps	10 x	DIV2	100 MHz	8 (10)	50 MHz	50 MHz	25 MHz
Channel 2	1 Gbps	10 x	FULL	100 MHz	16 (20)	50 MHz	100 MHz	50 MHz
Channel 3	500 Mbps	10 x	DIV2[2]	100 MHz	16 (20)	25 MHz	50 MHz	25 MHz

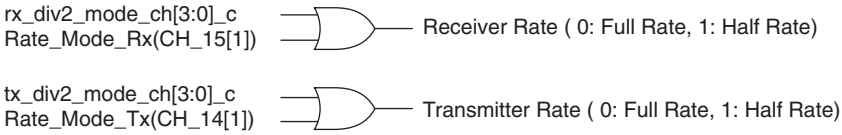
1. 阴影单元格中的时钟用作每个模式下的 FPGA 接口时钟。
2. 在 DIV2 模式下， tx_full_clk 调整为半速。 tx_half_clk 仅用于 16 位总线接口。
3. 10 位 SERDES only 模式或 SDI 模式。

全速和半速 （DIV2）之间的动态开关

本章节说明了如何在全速和半速 （DIV2）之间进行动态的开关控制。

两种速率模式控制信号通过或门，如图 8-8 所示。

图 8-8. 速率模式控制信号



tx_div2_mode_chx_c 是 FPGA 结构发送给 TX 的输入控制信号。

rx_div2_mode_chx_c 是 FPGA 结构发送给 RX 的输入控制信号。

Rate_Mode_Tx(CH_14[1]) 是针对 TX 路径的控制寄存器位。

Rate_Mode_Rx(CH_15[1]) 是针对 RX 路径的控制寄存器位。

在 rx 线上， pcs_rst 需在开关切换后发出。

在 tx 线上，换为新速率时无需复位。

参考时钟源

refclkp， refclkn

专用 CML 输入。除非使用其他的时钟源，这是 rx 和 tx 首选的时钟源。时钟信号可能是 CML、 LVDS 或 LVPECL。请参见 TN1114C，莱迪思 SERDES 的电气建议，例如接口电路。

fpga_txrefclk， fpga_rxrefclk

来自 FPGA 逻辑的参考时钟。主时钟引脚（PCLK）应作为 FPGA 的时钟输入引脚。时钟信号可能是 CML、LVDS、LVPECL 或单端。

FPGA PLL

当 FPGA PLL 用作参考时钟，连到 PLL 的参考时钟应分配给一个专用的 PLL 输入引脚。在更高的数据速率情况下，FPGA PLL 输出抖动可能会不能满足系统特性。不推荐在抖动敏感的应用中使用一个 FPGA PLL。

扩频时钟 （SSC）支持

链路两端的端口必须一直以两者之间低于 600pm 的速率差异传输数据。这个规定允许比特率时钟源有 +/- 300ppm 容限。必须遵守最小的时钟周期。首选的方法是调整扩展技术，不允许超过调制的额定频率。数据速率可在 0%至 -0.5%的额定频率范围内进行调制。调制率在不超过 30KHz 到 33KHz 的范围内。根据 +/- 300ppm 的容限，这里要求，当数据使用 SSC 调制时，两个端口需要相同的比特率时钟。

在 PCI Express 应用中，根复合体 （root complex）是负责扩频参考时钟，然后端点 （endpoint）基本上使用相同的时钟，通过 TX 返回频谱。因此，不需要单独的 RXREFCLK。主要应用是在插件卡上。插件卡无须使用来自连接器的 REFCLK，但必须接收和发送与 PCI Express 连接器 REFCLK 相同的 SSC。

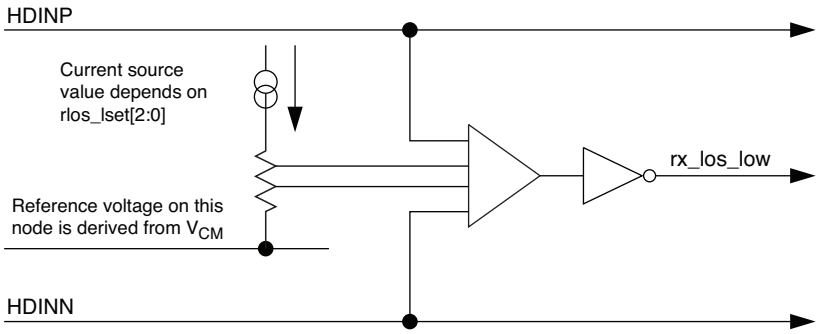
虽然 LatticeECP3 架构允许在一个 quad 中混合 PCI Express 通道和千兆以太网、串行 RapidIO 或 SGMII 通道，使用 PCI Express SSC 作为发送参考时钟将会导致违反千兆以太网、串行 RapidIO 和 SGMII 发送抖动参数的规定。

信号丢失

每个通道包含一个可编程的信号丢失检测器，如图 8-9 所示。

信号丢失阈值由可编程电流源的值决定。电流源值使用rlos_lset[2:0] 控制位来选择。阈值检测的结果通过rx_los_low 状态信号标示给 FPGA。

图 8-9. 信号丢失检测器



Note: rx_los_low shows that a signal has been detected for data rates above 1 Gbps with a maximum CID (Consecutive Identical Digits) of 7 bits (i.e., a minimum input signal transition density as is sent by 8b10b).

rx_los_low is supported with a default setting of rlos_lset[2:0] = 2, except in PCI Express mode and SDI mode. In PCI Express mode, 2 and 3 are supported.

In SDI mode, it is recommended to use the carrier detect output signal (/CD) from the external SDI cable equalizer.

表 8-8. 信号丢失检测器的响应时间

说明	最小值	典型值	最大值	单位
检测到信号丢失所需的时间 （rx_los_low 从 0 变为 1）	—	8	10	ns
检测到信号出现所需的时间 （rx_los_low 从 1 变为 0）	—	8	10	ns

失锁

发送 PLL 和单独通道 CDR 都有数字的基于计数器的失锁检测器。如果发送 PLL 失锁，发出 PLL 失锁信号，并且将一直发出该信号，直到 PLL 重新获得锁定。如果 CDR 失锁，该通道将发出失锁信号并锁定参考时钟，使 CDR 中的 VCO 重新开始训练。完成后，通道的失锁信号释放， CDR 转回锁定传进来的信号。 CDR 将继续锁定数据，或返回再次失锁状态，这样的话将重复再训练周期。有关 CDR 失锁的详细信息，请参阅本文档中的 SERDES/PCS 复位章节。

表 8-9. 锁丢失检测器的响应时间

说明	最小值	典型值	最大值	单位
检测到回路失锁所需的时间 （tx_pll_lol, rx_cdr_lol， 0 变为 1）	—	200	500	us
检测到回路锁定所需的时间 （tx_pll_lol， rx_cdr_lol， 1 变为 0）	—	200	500	us

TX 通道间偏移

控制信号 tx_sync_qd_c 复位所有有效的 TX 通道，从 bit 0 开始串行化。大多数多通道协议标准要求确保 TX 通道间偏移在一个确定的规定值范围内。

TX 串行器的复位通常是由于 tx_sync_qd_c 信号跳变或 PLL 锁定丢失时刻而产生的。

SERDES PCS 配置设置

LatticeECP3 PCS 可配置用于各种应用。使用 IPexpress 模块生成器工具选择设置，允许用户选择 PCS 的模式和特性选项。选项选择保存在自动配置文件中，之后位流生成器可以用它来将用户选择写入位流。更改 PCS 选项选择时，推荐用户重新运行 IPexpress，产生一个新的 PCS 模块，并创建一个新的自动配置文件。在运行位流生成器之前，一些选项可通过手动编辑自动配置文件来进行更改。配置之后， PCS 选项可通过可选的 SERDES 客户端接口总线写入 PCS 寄存器来进行动态修改。SERDES 客户接口允许 SERDES/PCS quad 通过寄存器来进行控制，以不同于原来的配置存储器单元。可通过 SCI 访问的控制和状态寄存器表请参见附录 A。

自动配置文件

每个 PCS 模式的初始寄存器设置可通过使用 IPexpress 中的自动配置功能实现。模块生成器提供了一个自动配置文件，包含了所选模式下 quad 和通道寄存器的设置。可参考这个文件来进行前端仿真，还可以集成到位流中。当一个自动配置文件集成到位流中，那么在配置过程中，所有的 quad 和通道寄存器将根据自动配置文件定义的值进行设置。如果在工作时，用户需要更改控制寄存器或监控状态寄存器，那么设计中必须包含 SCI。

发送数据

PCS quad 发送数据路径的每通道包含一个 8b10b 编码器和串行器。

8b10b 编码器

该模块实现了一个 8b10b 编码器，符合 IEEE 802.3ae-2002 1000BASE-X 规范标准。编码器根据规范实现 8 位到 10 位的代码转换，并且同时遵守所规定的运行 （running disparity）规则。 8b10b 编码器可通过在每个通道上将 CHx_8B10B 属性设置为 “BYPASS” 来将其旁路，其中 x 是通道编号。

串行器

8b10b 编码数据进行了并到串转换并且通过嵌入式 SERDES 进行片外传送。

接收数据

PCS quad 接收数据路径的每个通道包含以下子模块：解串行器、字对齐、8b10b 解码器、可选的链路状态机以及可选的接收时钟容限补偿 （CTC） FIFO。

解串行器

数据是在传至片上到嵌入式 SERDES 的过程中从串行变为并行的。

字对齐 （字节边界检测）

这个模块执行 comma 这一字符检测和对齐操作。接收逻辑使用逗号 （comma）字符来对传入数据流进行 10 位字符的字对齐。逗号 （comma）的描述可以查阅 802.3.2002 1000BASE-X 规范的 36.2.4.9 章节，以及 10GBASE-X 规范 48.2.6.3 章节的图 48-7。

字对齐模块支持大量可编程选项：

- 字对齐控制由嵌入式链路状态机（LSM）或 FPGA 控制实现。除了 8b10b 数据包模式外，还支持 8 位 SERDES Only、 10 位 SERDES Only 以及 SDI 模式。
- 可以设置两种可编程字对齐字符（通常一个用于 positive disparity，一个用于 negative disparity）以及一个可编程的每位掩码寄存器用于对齐比较。对齐字符和掩码寄存器可按每个 quad 来设置。对于许多协议来说，字对齐字符可设为 “XX00000011” （jhgfiedcba 位用于 positive disparity comma 字符对应代码组 K28.1、 K28.5 和 K28.7）以及 “XX01111100” （jhgfiedcba 位用于 negative disparity comma 字符对应代码组 K28.1、 K28.5 和 K28.7）。但是，用户也可定义任意 10 位模式。
- 第一个对齐的字符由分配给属性 COMMA_A 的 10 位数值所定义。该值适用于 PCS quad 中的所有通道。
- 第二个对齐的字符由分配给属性 COMMA_B 的 10 位数值所定义。该值适用于 PCS quad 中的所有通道。

- 掩码寄存器定义了要比较哪个字对齐位（掩码寄存器中值为 '1' 的位表示需要检查字对齐字符寄存器中的相应位）。掩码寄存器由 COMMA_M 属性的 10 位值决定。此值适用于 PCS quad 中的所有通道。当属性 CHx_RXWA （字对齐）设置为 "ENABLED"，并且 CHx_ILSM （内部链路状态机）设置为 "ENABLED"，某个基于协议的链路状态机的工作将控制字对齐。若欲了解更多有关基于协议的链路状态机的工作的信息，请参见下面特定协议的链路状态机章节。

8b10b 解码器

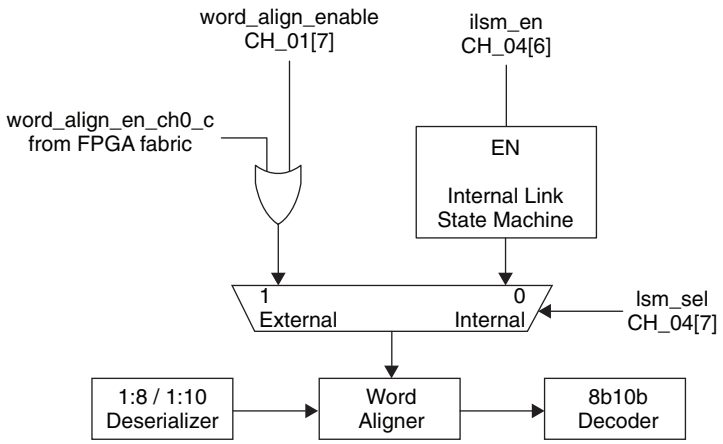
8b10b 解码器实现了一个 8b10b 解码器操作，符合 IEEE 802.3-2002 参数规范标准。解码器根据规范实现 10 位到 8 位的代码转换，并且同时遵守所规定的运行差异规则。当检测到代码违例时，接收数据 rxdata 设为 0xEE，rx_k_chn 设为 '1'。

外部链路状态机选项

当属性 CHx_ILSM （内部链路状态机）设为 "DISABLED"，并且 CHx_RXWA （字对齐）设为 "ENABLED"，控制信号 word_align_en_ch(0-3) 用于使能字对齐。这个信号应当由 FPGA 结构中实现的 FPGA 外部链接状态机产生。当 word_align_en_ch(0-3)_c 为高电平时，字对齐将锁定对齐位并保持锁定状态。它将停止输入数据与用户定义的字对齐字符的比较，并保持当前第一次比较成功的 COMMA_A 或 COMMA_B 的对齐状态。如果需要重新对齐，触发 word_align_en_ch(0-3)_c 信号由低电平变为高电平。字对齐将重新锁定下一次符合用户定义的字对齐字符的输入。如果需要，word_align_en_ch(0-3)_c 可通过 PCS quad 外部实现的链路状态机控制，仅在特定条件下允许对字对齐进行更改。

图 8-10 说明了链路状态机选项。

图 8-10. PCS 字对齐和链路状态机选项

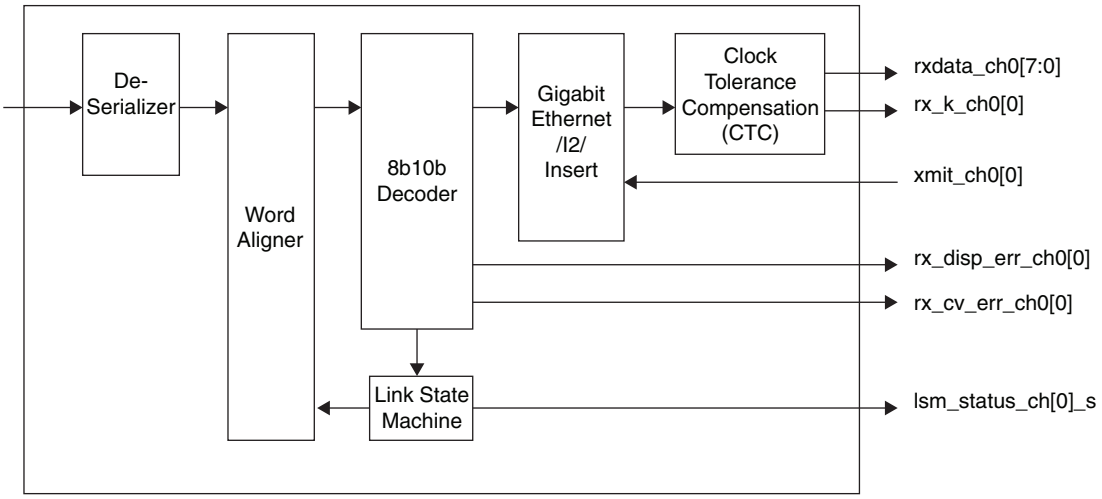


当选择了一个链路状态机并在某一通道使能后，一旦链路同步成功，该通道的 lsm_status_ch(0-3)_s 状态信号将变为高电平。

千兆以太网模式下的 Idle 信号插入

PCS 设为千兆以太网模式，将 /I2/ 信号插入接收数据流用以自动协商。千兆以太网自动协商以软逻辑方式执行。该功能以每 2048 个时钟周期插入连续的 8 个 /I2/ 有序集。/I2/ 插入由输入到 PCS 的 xmit_ch(0-3) 信号来控制，xmit_ch(0-3) 信号由自动协商软逻辑来驱动。图 8-11 说明了当 PCS 设为千兆以太网模式时，接收逻辑的一个通道（此示例中的 channel 0），并说明了这些控制 / 状态信号。

图 8-11. 千兆以太网模式下的 PCS 接收路径 （Channel 0 示例）



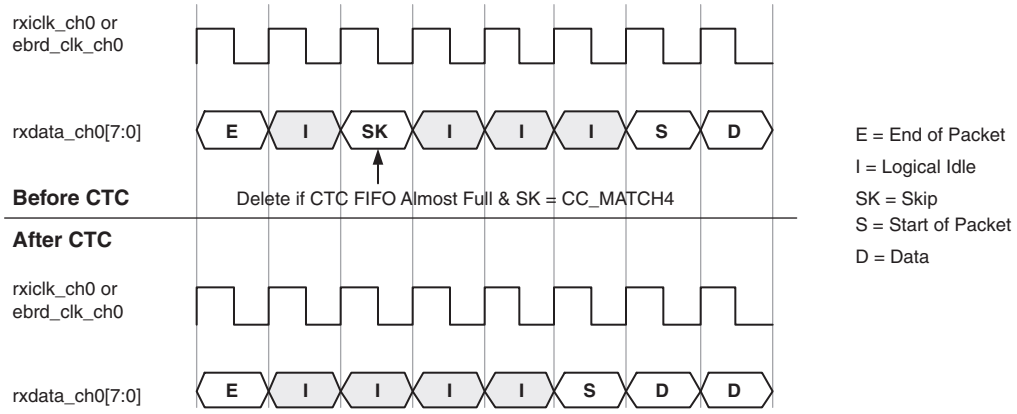
时钟容限补偿

时钟容限补偿 （CTC）模块执行恢复的接收时钟和锁定的参考时钟之间的时钟速率调整。时钟补偿是通过在预先定义的位置插入或删除数据字节来实现的，并且同时保证不会造成数据包丢失。16 字节的 CTC FIFO 用于在两个时钟域之间传输数据，并且将在 LatticeECP3 SERDES 规定的最大 ppm 容限范围内允许时钟差异。（请参见 LatticeECP3 系列数据手册中的直流和开关特性章节）。

在通道属性 CHx_CTC 设为 “ENABLED” 时，该通道就使能了时钟容限补偿模块。当通道属性 CHx_CTC 设为 “DISABLED” 时， CTC 就被旁路。

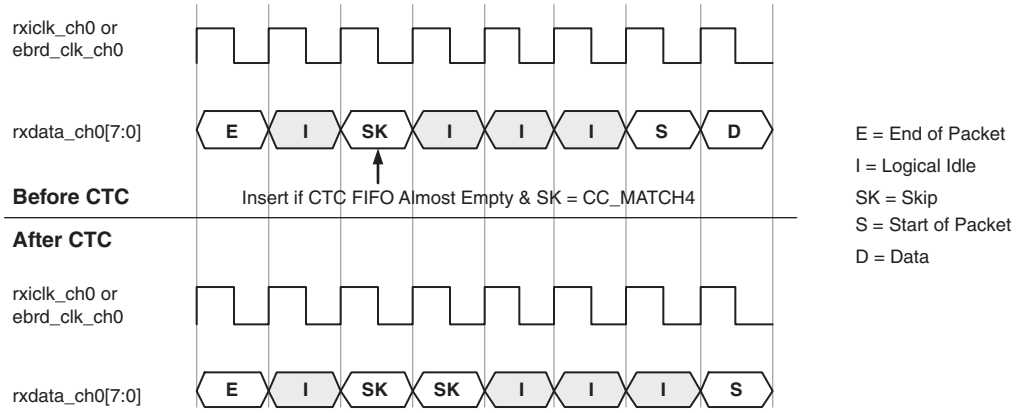
1 字节删除的图示说明如图 8-12 所示。

图 8-12. 时钟容限补偿 1 字节删除示例



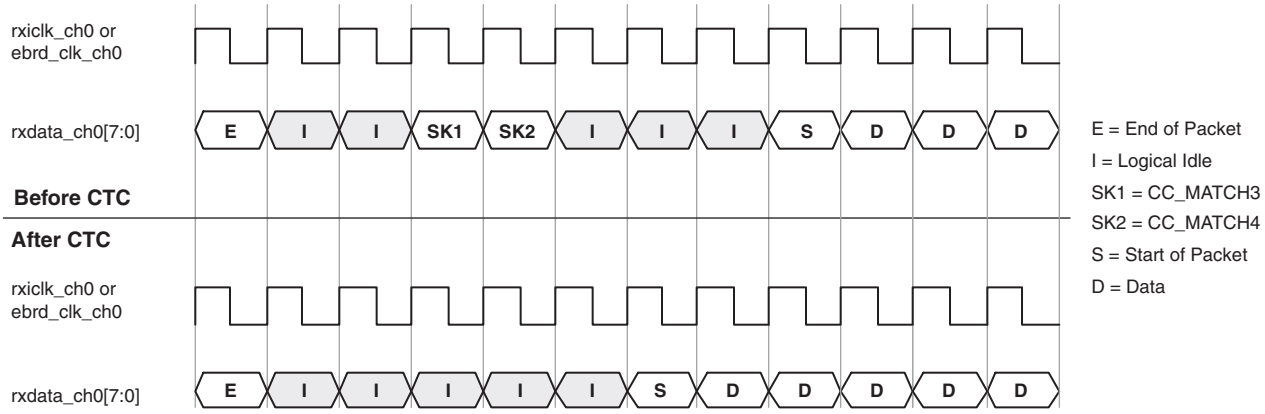
1 字节插入的图示说明如图 8-13 所示。

图 8-13. 时钟容限补偿 1 字节插入示例



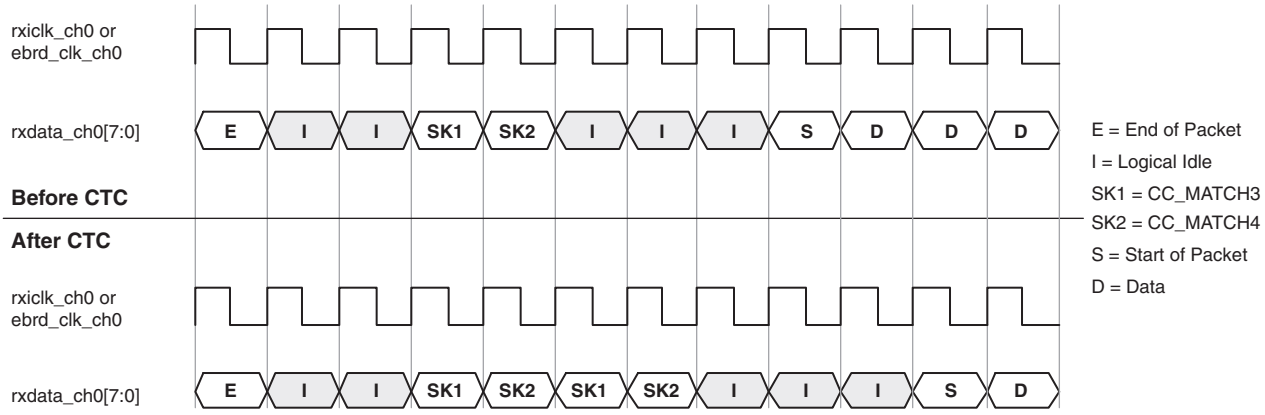
2 字节删除的图示说明如图 8-14 所示。

图 8-14. 时钟容限补偿 2 字节删除示例



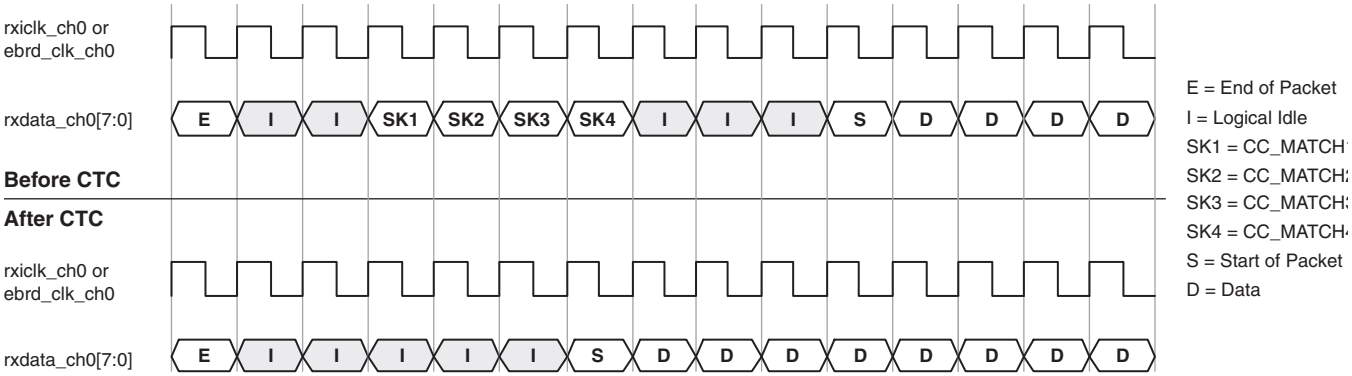
2 字节插入的图示说明如图 8-15 所示。

图 8-15. 时钟容限补偿 2 字节插入示例



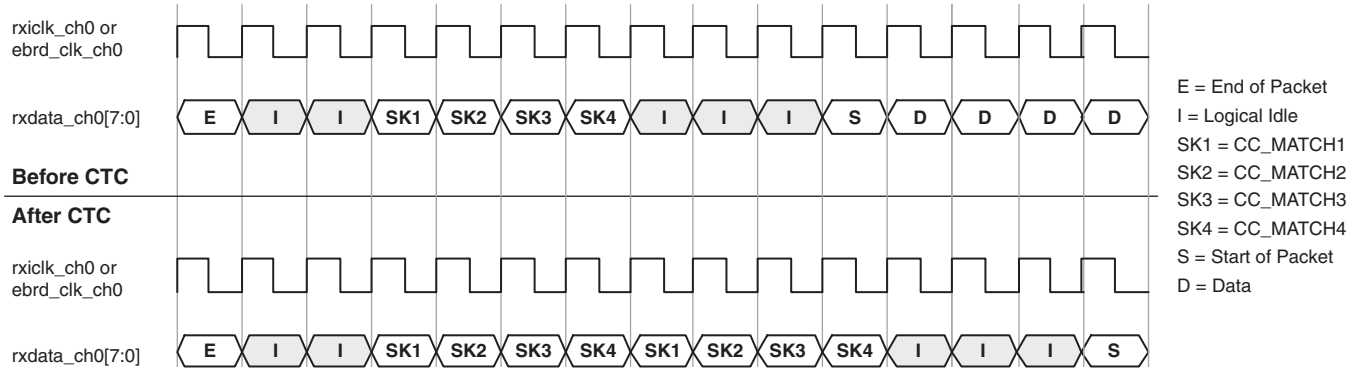
4 字节删除的图示说明如图 8-16 所示。

图 8-16. 时钟容限补偿 4 字节删除示例



4 字节插入的图示说明如图 8-17 所示。

图 8-17. 时钟容限补偿 4 字节插入示例



当使用 CTC 时，针对不同的应用，对时钟补偿必须做以下适当的设置：

- **使用 CC_MATCH_MODE 属性设置插入 / 删除模式长度。**该属性设置了在执行插入和删除之前， CTC 所比较的跳过字节 （skip byte）的数目。 CC_MATCH_MODE 的值为 “1” （1 字节插入 / 删除）、 “2” （2 字节插入 / 删除）以及 “4” （4 字节插入 / 删除）。最小的数据包间间隙也必须根据目标应用适当地进行设置。数据包间间隙通过属性 CC_MIN_IPG 来指定。 CC_MIN_IPG 的允许值为 “0”、 “1”、 “2” 和 “3”。跳过字符删除后允许的最小数据包间间隙按照表 8-10 中这些属性的设置来执行。
- **跳过字节或定序集必须按照所选的 CC_MATCH_MODE 进行设置。**对于 4 字节插入 / 删除（CC_MATCH_MODE = “4”），第一个字节必须分配给属性 CC_MATCH1，第二个字节必须分配给属性 CC_MATCH2，第三个字节必须分配给属性 CC_MATCH3，并且第四个字节必须分配给属性 CC_MATCH4。所有的都为 10 位二进制值。

  例如：

  如果 4 字节跳过定序集设置为 /K28.5/D21.4/D21.5/D21.5，那么 “CC_MATCH1” 应设为 “0110111100”， “CC_MATCH2” = “0010010101”， “CC_MATCH3” = “0010110101” 并且 “CC_MATCH4” = “0010110101”。

  对于 2 字节插入 / 删除 （CC_MATCH_MODE = “2”），第一个字节必须分配给属性 CC_MATCH3，并且第二个字节必须分配给属性 CC_MATCH4。

对于 1 字节插入 / 删除 （CC_MATCH_MODE = “1”），跳过字节必须分配给属性 CC_MATCH4。

- **时钟补偿 FIFO 的高位水印和低位水印必须根据所使用的协议进行适当的设置。**设置值的范围可从 0 至 15，并且高位水印必须设置比低位水印更高的值 （两者不可设为相同的值）。高位水印值通过给属性 CCHMARK 一个值来设置。 CCHMARK 的允许值以十六进制表示，从 “0” 到 “F”。低位水印值通过给属性 CCLMARK 一个值来设置。 CCLMARK 的允许值以十六进制表示，从 “0” 到 “F”。
- 当通过 ispLEVER 模块生成器生成 PCS 模块时，如果选择了 “Error Status Ports”，根据 PCS/FPGA 接口的 cc_overrun_ch(0-3)，可以对每个通道上的时钟补偿 FIFO 过载进行监测。
- 当通过 ispLEVER 模块生成器生成 PCS 模块时，如果选择了 “Error Status Ports”，根据 PCS/FPGA 接口的 cc_underrun_ch(0-3)，可以对每个通道上的时钟补偿 FIFO 欠载进行监测。

计算最小的数据包间间隙

表 8-10 显示了数据包间间隙的用户自定义值（由 CC_MIN_IPG 属性定义）和确保的最小数据包间字节数之间的关系，这个最小数据包间字节数是指，从 PCS 跳过字符删除后的最小数据包间字节数。表中将数据包间间隙显示为一个乘数。数据包间最小字节数等于每次插入 / 删除的字节数乘以表中所列的乘数。例如，如果每次插入 / 删除的字节数为 4（CC_MATCH_MODE 设为 “4”），并且最小的数据包间间隙属性 CC_MIN_IPG 设为 “2”，那么最小的数据包间间隙等于 4（CC_MATCH_MODE = “4”）乘以 3（表 8-10 中 CC_MIN_IPG = “2”）或 12 字节。PCS 不会执行跳过字符删除，直到最小的数据包间字节数通过 CTC。

表 8-10. 最小的数据包间间隙乘数

CC_MIN_IPG	插入 / 删除 乘数
0	1x
1	2x
2	3x
3	4x

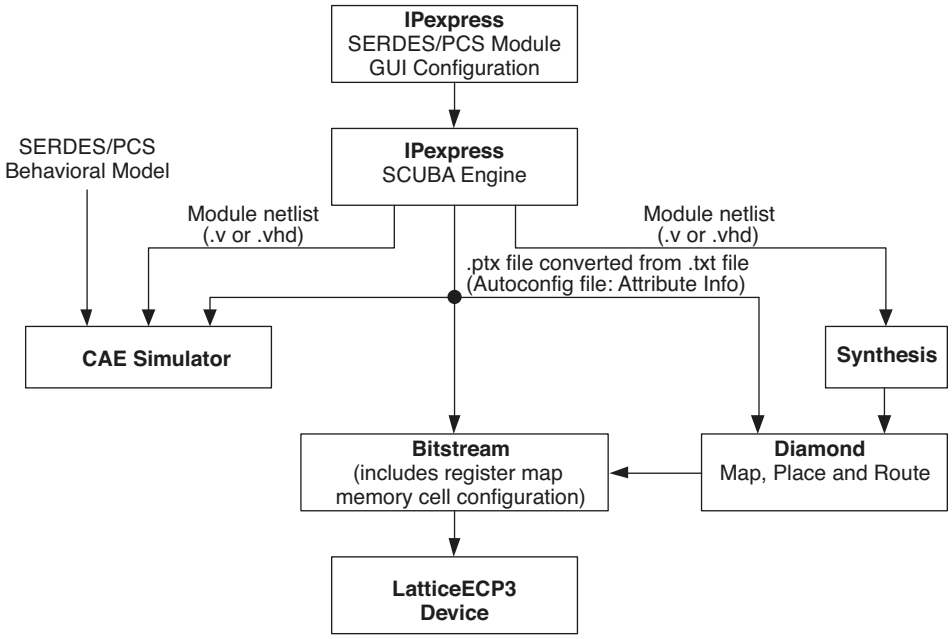
请注意带有 TW 后缀的 LatticeECP3-150EA 器件系列的 CTC 支持：*对于初始版本的带有 TW 后缀的 LatticeECP3-150EA 器件是不支持 PCS 中 CTC 的。CTC 功能可以被旁路，并在软 IP 中实现。目前莱迪思提供的许多 IP 核是以软逻辑形式实现 CTC 逻辑。*

使用 Diamond 中的 IPexpress

IPexpress 是用以创建和配置 SERDES 和 PCS 块的。设计师们使用图形用户界面来为特定 quad 或通道选择 SERDES 协议标准。 IPexpress 从这个图形用户界面获得输入并产生配置文件 （.txt 文件）和 HDL 网表。 HDL 模型用于仿真和综合流程。配置文件包含属性层映射信息。这个文件是仿真和 ispLEVER bitgen 程序的输入。强烈推荐设计师们在 IPexpress 中进行更改和更新，然后重新生成配置文件。在一些例外情况下，用户可以修改配置文件。

图 8-18 显示了当使用 IPexpress 为 SERDES 协议标准生成 SERDES/PCS 块时的工具流程。

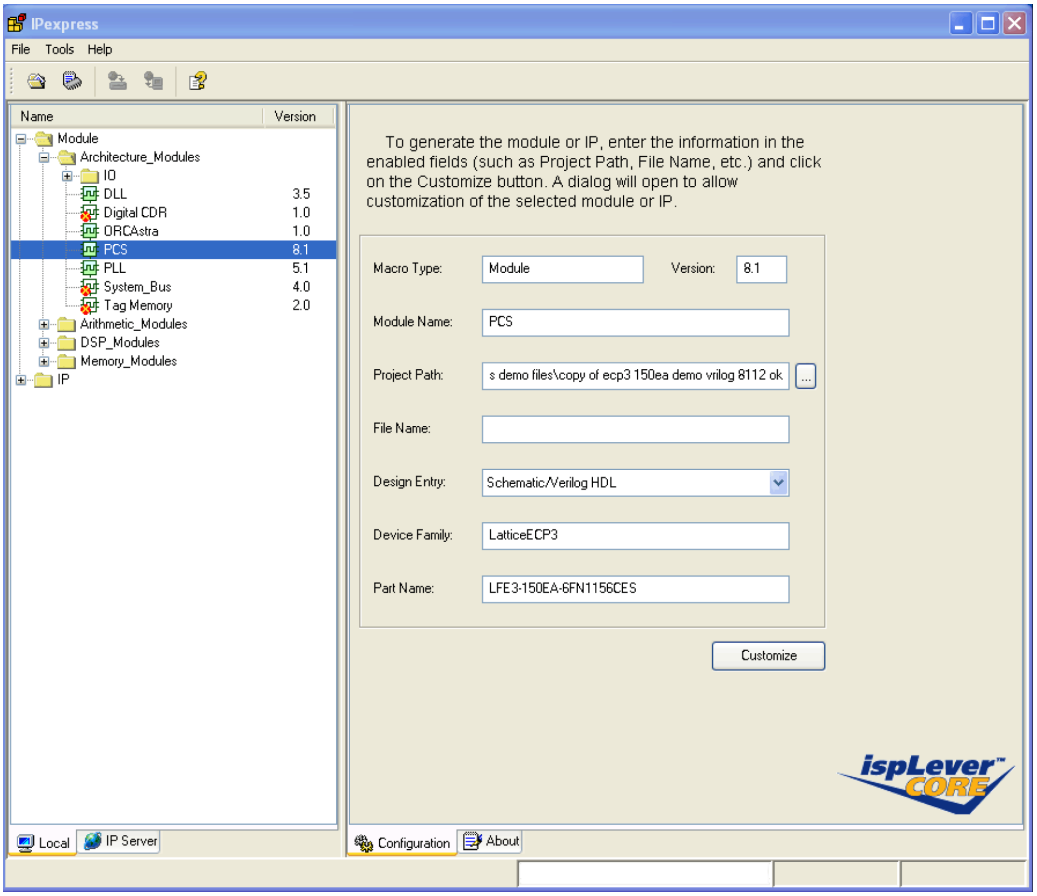
图 8-18. SERDES_PCS Diamond 用户流程



IPexpress 中的 PCS 模块生成器

图 8-19 显示了在 IPexpress 图形用户界面中选择 PCS 时的主窗口。

图 8-19. IPexpress PCS 主窗口



Quad 设置选项卡

图 8-20 显示了当输入了文件名并且在主窗口中勾选了 **Customize** 按钮后，Quad 设置选项卡窗口。在这个窗口中首先必须要求输入的是要为每个通道选择一种协议模式。每个通道可以配置为 ‘RX and TX’、‘RX Only’、‘TX Only’、‘Disabled’ 或 ‘Low Speed Data Port’。

图 8-20. 配置图形用户界面——Quad 设置选项卡

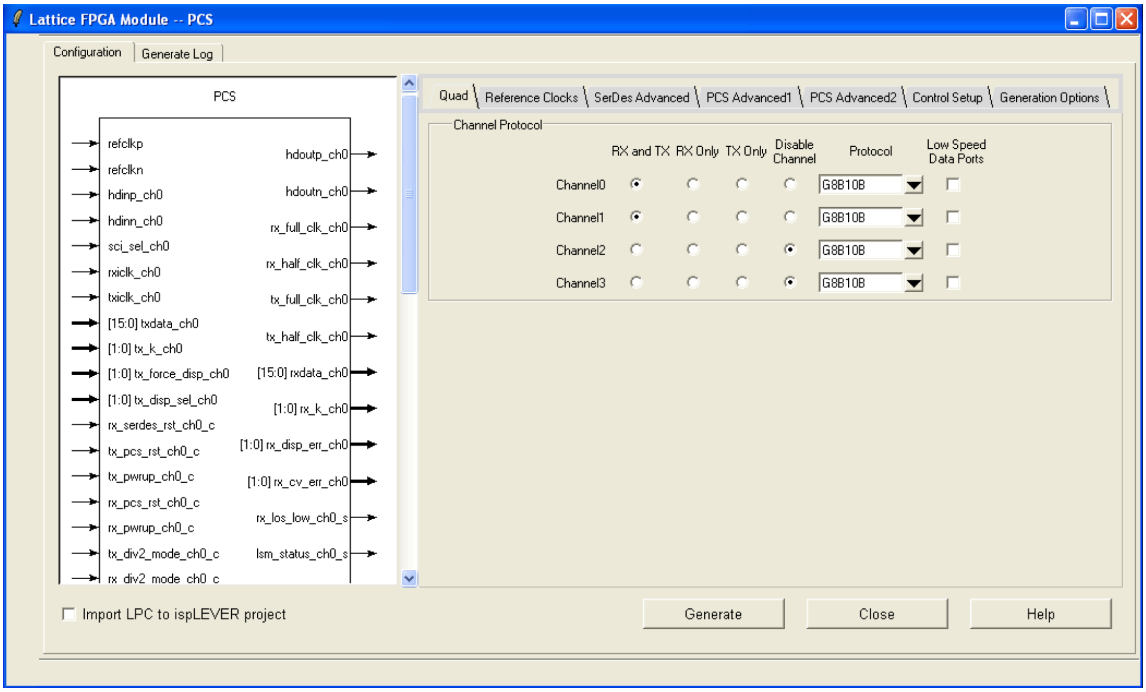


表 8-11. SERDES_PCS 图形用户界面属性——Quad 选项卡设置

图形用户界面内容	属性名称	属性范围	默认值
Channel Protocol	CHx_MODE	RX and TX, RX Only, TX Only	DISABLED
Disable Channel[1]	CHx_MODE	ENABLE, DISABLE	DISABLED
Protocol	CHx_PROTOCOL	GIGE, SGMII, XAUI, SRIO, PCIE, SDI, G8B10B, 10BSER, 8BSER, CPRI, OBSAI	G8B10B
Low Speed Data Port	CHx_LDR	RX and TX, RX Only, TX Only	DISABLED

1. 对于 328-ball csBGA 封装的 LatticeECP3-17EA 器件，仅 channel 0 和 3 可用。

参考时钟设置选项卡

在该选项卡中，选择 TX 和 RX 参考时钟源的属性。用户可以选择 EXTERNAL 或 INTERNAL 参考时钟。此外，还有一个工具来提供所需要的时钟速率以及用于特定数据速率的乘法器设置。除此之外，对于给定的数据总线宽度，工具提供通道到内核接口所需的时钟速率。

图 8-21. 配置图形用户界面 —— 参考时钟设置选项卡

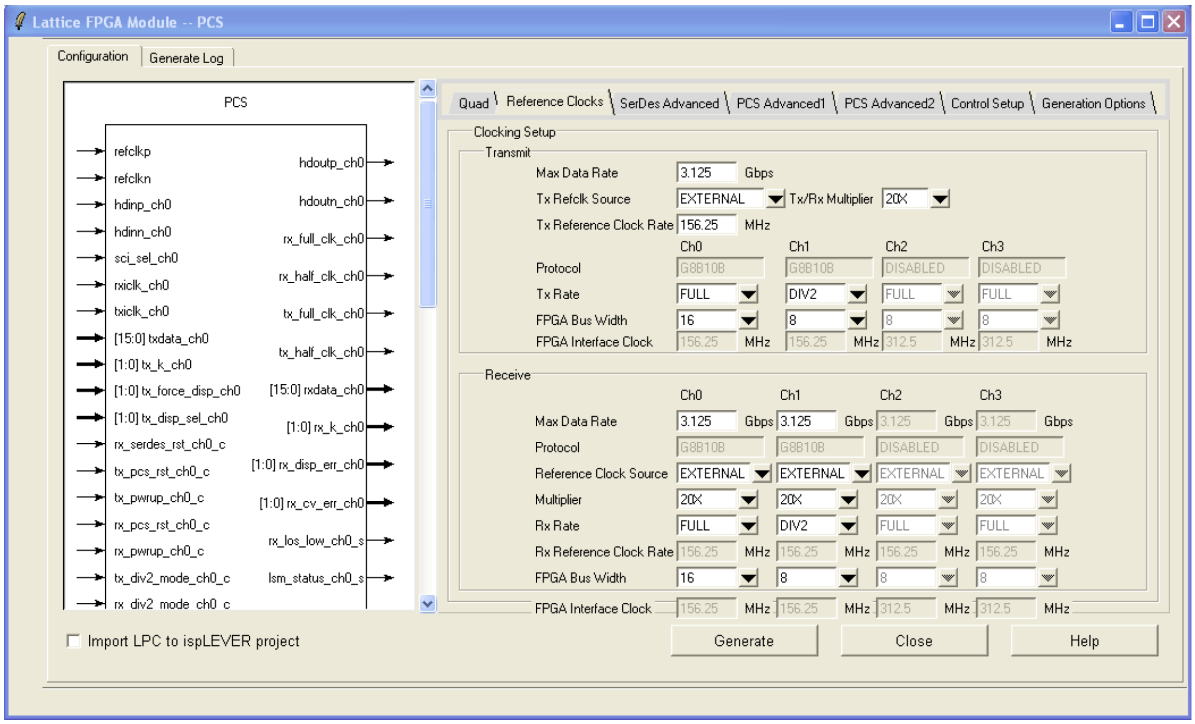


表 8-12. SERDES_PCS 图形用户界面属性 —— 参考时钟设置选项卡

图形用户界面内容	属性名称	范围	默认值（图形用户界面）	默认值（属性）
Transmit				
Max. Data Rate[1]	N/A	0.23 至 3.2 Gbps	2.5 Gbps	N/A
TX Refclk Source	PLL_SRC	INTERNAL, EXTERNAL	INTERNAL	REFCLK_INT
TX/RX Multiplier	REFCK_MULT	8X, 10X, 16X, 20X, 25X	根据协议	
TX Reference clock Rate	#REFCLK_RATE[2]		根据协议	
Protocol	用户无法访问。仅供参考。			
TX Rate	CHx_TX_DATA_RATE	FULL, DIV2, DIV11	FULL	FULL
FPGA Bus Width	CHs_TX_DATA_WIDTH	8, 10, 16, 20	根据协议	
FPGA Interface Clock	#CH0_TX_FICLK_RATE			
Receive				
Max. Data Rate[1]	N/A	0.23 至 3.2 Gbps	2.5 Gbps	N/A
Protocol	用户无法访问。仅供参考。			
Refclk Source	CHx_CDR_SRC	INTERNAL, EXTERNAL	INTERNAL	REFCLK_INT
Multiplier	用户无法访问。仅供参考。			
RX Rate	CHx_RX_DATA_RATE	FULL, DIV2, DIV11	FULL	FULL
RX Reference Clock Rate	#CH0_RXREFCLK_RATE			
FPGA Bus Width	CHx_RX_DATA_WIDTH	8, 10, 16, 20	根据协议	
FPGA Interface Clock	#CH0_RX_FICLK_RATE			

1. 速率并不反映在自动配置文件中。相反， DATARATE RANGE 是为特定数据速率所指定的，如 :150 Mbps ≤ LOWLOW ≤ 230 Mbps， 230 Mbps < LOW ≤ 450 Mbps，450 Mbps < MEDLOW ≤ 0.9 Gbps，0.9 Gbps < MED ≤ 1.8 Gbps，1.8 Gbps < MEDHIGH ≤ 2.55 Gbps，2.55 Gbps < HIGH ≤ 3.2Gbps。
2. '#' 后面的属性表示仅供参考。这些属性也包括在自动配置文件中，以便参考。

SERDES 高级设置

该选项卡是用于访问发送和接收 SERDES 的所有 4 个通道的高级属性的。选择发送属性，如：PreEmphasis、Termination、 Differential Output Voltage。选择接收属性，如：Equalization、 Termination、 I/O Coupling。还可以选择发送 SERDES 时钟和 PLL 的属性。

图 8-22. 配置图形用户界面——SERDES 高级设置选项卡

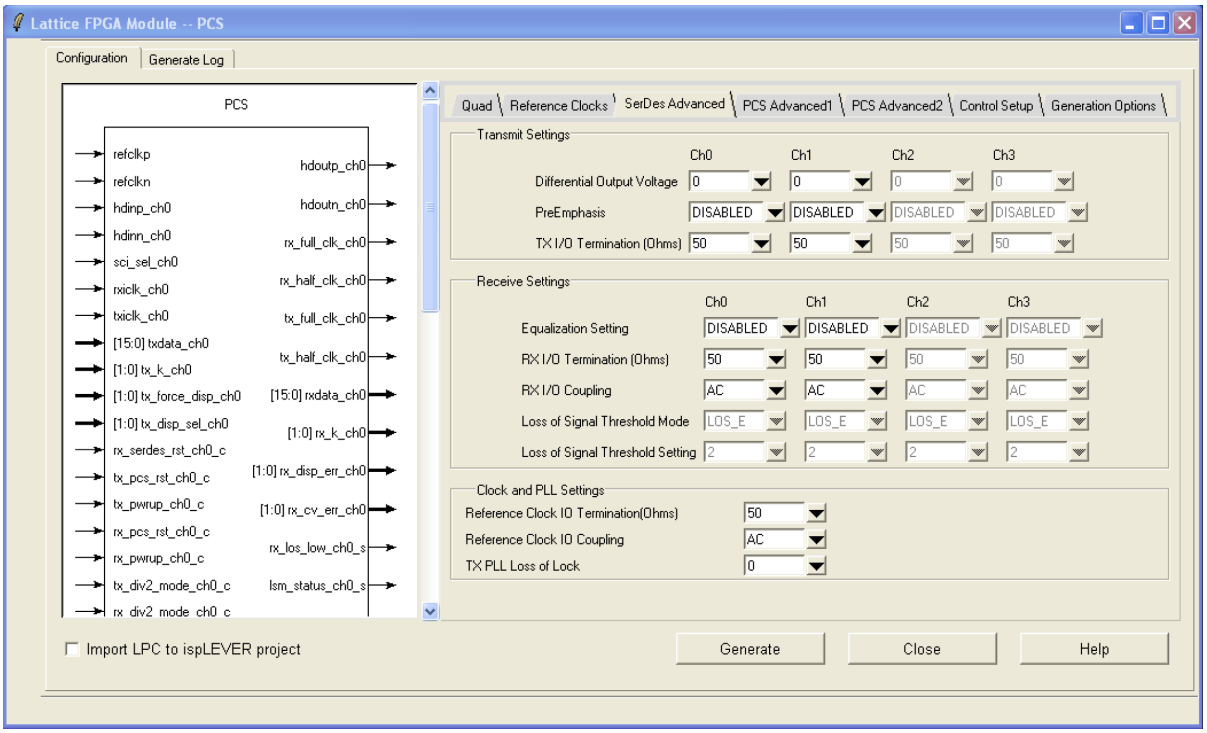


表 8-13. SERDES_PCS 图形用户界面属性——SERDES 高级设置选项卡

图形用户界面内容	属性名称	值	默认值
Differential Output Voltage	CHx_TDRV[8]	-4 (640mV)[5], -3 (780mV), -2 (870mV), -1 (920mV), 0 (1040mV:default), 1 (1130mV)[6], 2 (1260mV)[7], 3 (1350mV)[7], 4 (1440mV)[7]	0
PreEmphasis	CHx_TX_PRE	Disabled, 0 (0%), 1 (5%), 2 (12%), 3 (18%), 4 (25%), 5 (33%), 6 (40%), 7 (48%)	DISABLED
TX I/O Termination (Ohms)[3]	CHx_RTERM_TX	50, 75, 5K	50
Equalization[1]	CHx_RX_EQ	Disabled, Mid_Low, Mid_Med, Mid_High, Long_Low, Long_Med, Long_High	DISABLED
RX I/O Termination (Ohms)[3]	CHx_RTERM_RX	50, 60, 75, High	50
RX I/O Coupling	CHx_RX_DCC	AC, DC	AC[2]
Loss of Signal Threshold	CHx_LOS_THRESHOLD_LO	2 (+15%),3 (+25%)	2[4]
TX PLL Reference Clock I/O Termination (Ohms)[3]	PLL_TERM	50, 2K	50
TX PLL Reference Clock I/O Coupling	PLL_DCC	AC, DC	AC[10]
PLL Loss of Lock	PLL_LOL_SET	0: +/- 1350ppm x2[9] 1: +/- 2400ppm x2 2: +/- 6800ppm 3: +/- 400ppm	0

1. 参见表 8-106 了解详细信息。
2. 内部片上 AC 耦合电容的典型值是 5 pF。
3. 端接电阻及它们的使用：
   RX I/O 端接：
   – 50：除了 SMTPE，目前所有的协议都使用一个 50 欧姆的端接电阻。
   – 60：以供灵活选择。
   – 75：SMPTE 使用一个 75 欧姆的端接电阻。
   – HIGH：不使用 Rx 时的默认值。
   TX I/O 端接：
   – 50：除了 SMTPE，目前所有的协议都使用一个 50 欧姆的端接电阻。
   – 75：SMPTE 使用一个 75 欧姆的端接电阻。
   – 5K：如 PCI Express 电气空闲和 PCI Express RX 检测。用户不需要为 RX 检测设置该端接值。请参见 PCI Express 接收器检测章节。
   TX PLL 端接：
   – 50：如果印刷电路板上没有 50 欧姆的端接电阻。
   – 2K：如果印刷电路板上有 50 欧姆的端接电阻。
4. PCS 配置，对于除 PCI Express 外的所有协议，图形用户界面仅支持值 2。对于 PCI Express，值 2 和值 3 都支持。
5. TDRV_AMP_BOOST(CH_13[3]) 设为 1，以实现该信号幅度。
6. 该设置是 PCI Express 的默认设置。使用 PCI Express 协议时，推荐使用该默认设置。因而 IPexpress 图形用户界面中 TDRV 下拉窗口是不可更改的，显示为灰色。其他设置可仍然通过编辑 autoconfig 文件 （.txt 文件）中的 CHn_TDRV 属性来使用。
7. 这些设置中的 VCCOB 必须为 1.5V。
8. 这些值为典型值。对于整个频率范围内大约有 +/-20% 的裕度。请参见表 8-105 中 CHn_TDRV 行，了解详细信息。
9. ‘x2’ 为内部 LOL 计数器中成功的 ppm 双倍计数。
10. 推荐在大多数应用中使用交流耦合。直流耦合应当仅与外部交流耦合电容结合使用。

PCS 位置的指定

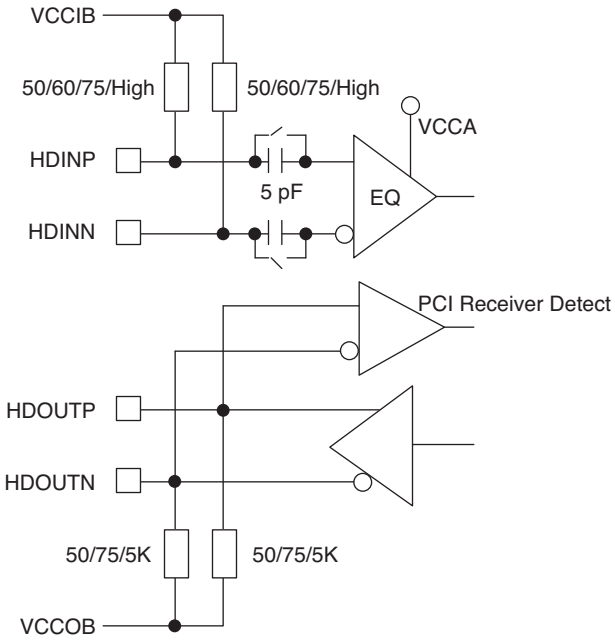
用户可以通过约束文件 （.lpf）来指定其希望的 PCS quad 位置。使用约束 “locate”。语法示例如下所示。

```
LOCATE COMP "pcs_inst_name" SITE "PCSB" ;
```

Quad 名称	位置名称
Quad A	PCSA
Quad B	PCSB
Quad C	PCSC
Quad D	PCSD

高速 I/O 端接拓扑结构如图 8-23 所示。

图 8-23. 高速 I/O 端接



PCS Advanced1 设置

通过此选项卡可访问所有四个通道的发送和接收PCS的高级属性。每个TX和RX通道的极性和工作模式（如8b10b）可以独立进行选择。此外，字对齐值也可以选择，如 comma 值、 comma 掩码和 comma 对齐。

图 8-24. 配置图形用户界面——PCS Advanced1 设置选项卡

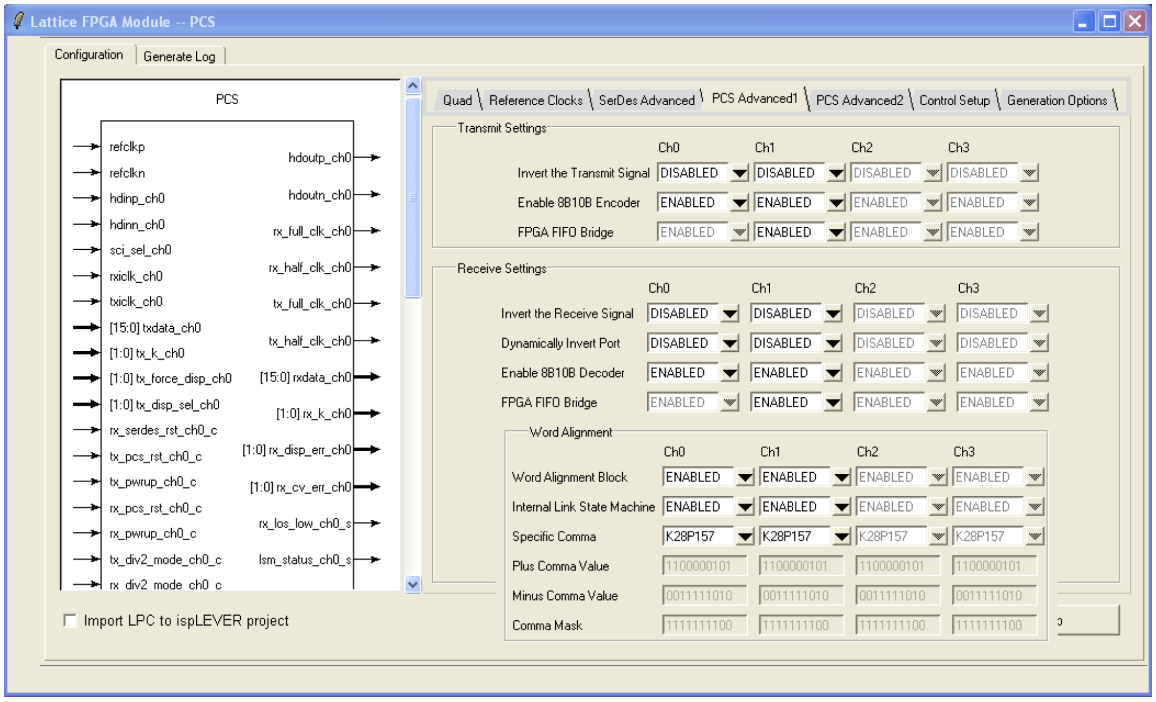


表 8-14. SERDES/PCS 图形用户界面——PCS Advanced1 设置选项卡

图形用户界面内容		属性名称	默认值
Transmitter	Invert the Transmit Signal	CHx_TX_SB	DISABLED
	Enable 8b10b Encoder	CHx_TX_8B10B	根据协议
	FPGA FIFO Bridge	CHx_TX_FIFO	根据协议
Receiver	Invert the Receive Signal	CHx_RX_SB	DISABLED
	Dynamically Invert Port	N/A	DISABLED
	Enable 8b10b Decoder	CHx_RX_8B10B	根据协议
	FPGA FIFO Bridge	CHx_RX_FIFO	根据协议
Word Alignment	Word Alignment Block	CHx_RXWA	根据协议
	Internal Link	CHx_ILSM	根据协议
	Specific Comma	#CHx_SCOMMA	根据协议
	Plus Comma Value	CHx_COMMA_A[1]	1100000101
	Minus Comma Value	CHx_COMMA_B	0011111010
	Comma Mask	CHx_COMMA_M	根据协议[2]

1. 根据定义， COMMA_A 和 COMM_B 是带有 positive 和 negative running disparity 的一组 8b10b 编码控制字符。用户必须提供符合协议的适当的 IDLE 序列来获得链接状态机同步。例如， 1 GbE 协议需要 K28.5+D5.6 或 D16.2 用作 IDLE （字对齐和同步状态机）。默认值使用小端字节序格式。
2. 在大多数应用中， K28.5 用作逗号字符。默认的掩码值是 1111111111。在 G8B10B 模式可以使用任意逗号字符，因此掩码是 1111111100，以检测所有的三种逗号字符， K28.1、 28.5、 28.7。

PCS Advanced2 设置

该选项卡用于为时钟容限补偿模块设置值。

图 8-25. 配置图形用户界面——PCS Advanced2 设置选项卡

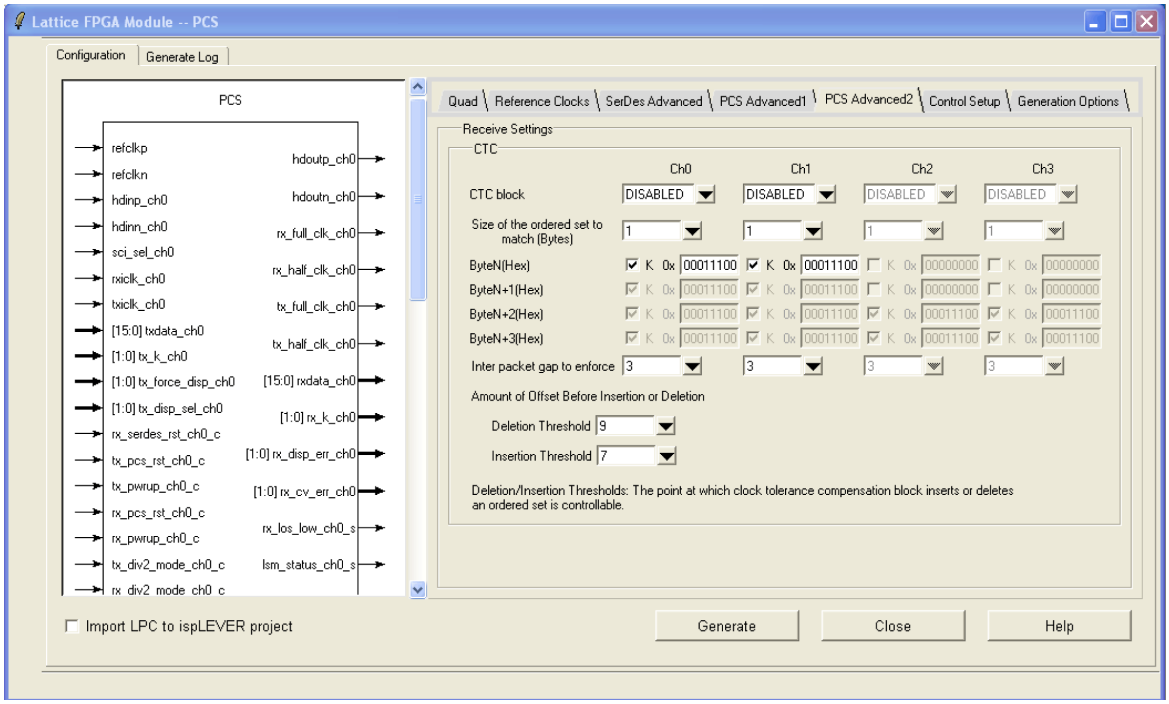


表 8-15. SERDES/PCS 图形用户界面——PCS Advanced2 设置选项卡

图形用户界面内容	属性名称	默认值
CTC block	CHx_CTC	根据协议 [1]
Size of ordered set	CHx_CC_MATCH_MODE	根据协议
Byte N	CHx_CC_MATCH1	根据协议
Byte N+1	CHx_CC_MATCH2	根据协议
Byte N+2	CHx_CC_MATCH3	根据协议
Byte N+3	CHx_CC_MATCH4	根据协议
Inter-packet gap	CHx_CC_MIN_IPG	根据协议
Deletion threshold	CCHMARK	9
Insertion threshold	CCLMARK	7

1. 始终禁止：XAUI、 SDI、 CPRI、 OBSAI、 10 位 SERDES 以及 8 位 SERDES。
始终使能：串行 RapidIO。
所有其他模式：默认为禁止。在 IP 中提供大多数 CTC 功能。

控制设置

该选项卡是用于选择 SCI 接口和其他调试和控制选项的。此外，用户可以使能 SCI、错误报告、PLL quarter 时钟以及回环功能。

图 8-26. 配置图形用户界面 —— 控制设置选项卡

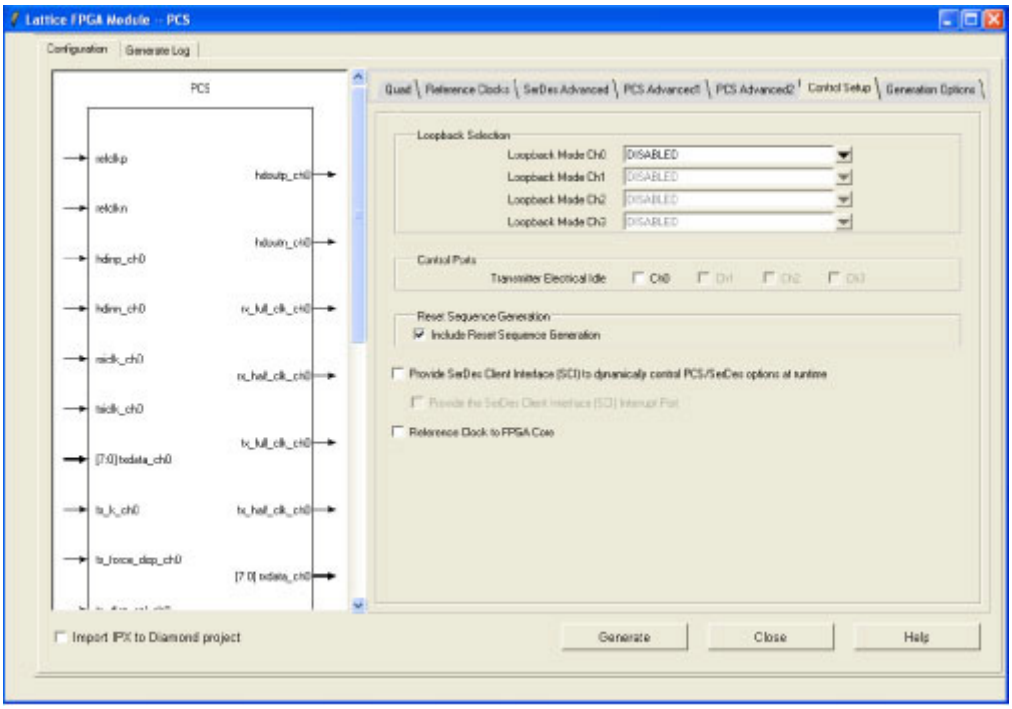


表 8-16. 选项卡 5， SERDES_PCS 图形用户界面属性 —— 控制设置选项卡

图形用户界面内容	属性名称	默认值
Loopback Mode (Ch0, Ch1, Ch2, Ch3)	DISABLED Loopback serial data after equalizer Loopback serial data after transmit driver Loopback parallel data after de_serializer	DISABLED[1]
Transmitter Electrical Idle	signal tx_idle_ch0_c is provided	DISABLED
Include Reset Sequence Generation[2]	Include the TX and RX Reset Sequence	ENABLED
Provide SERDES Client Interface	N/A	
Provide the SERDES Client Interface Interrupt Port	INT_ALL	DISABLED
Reference Clock to FPGA core	QD_REFCK2CORE	DISABLED

注：ispLEVER 8.0 中不支持复位序列发生器。

1. 当回环模式在默认状态 （禁用）时，用户可以在 HDL 模块中使用两个 SERDES 桥并行回环控制信号 （sb_felb_ch[3:0]_c 和 sb_felb_rst_ch[3:0]_c），动态打开和关闭回环模式。如果不使用回环模式，这些信号应连接到地。
2. 本文档的 SERDES/PCS 复位章节说明了复位序列产生。

FPGA 核和复位序列的参考时钟

复位序列在复位状态机中使用参考时钟。

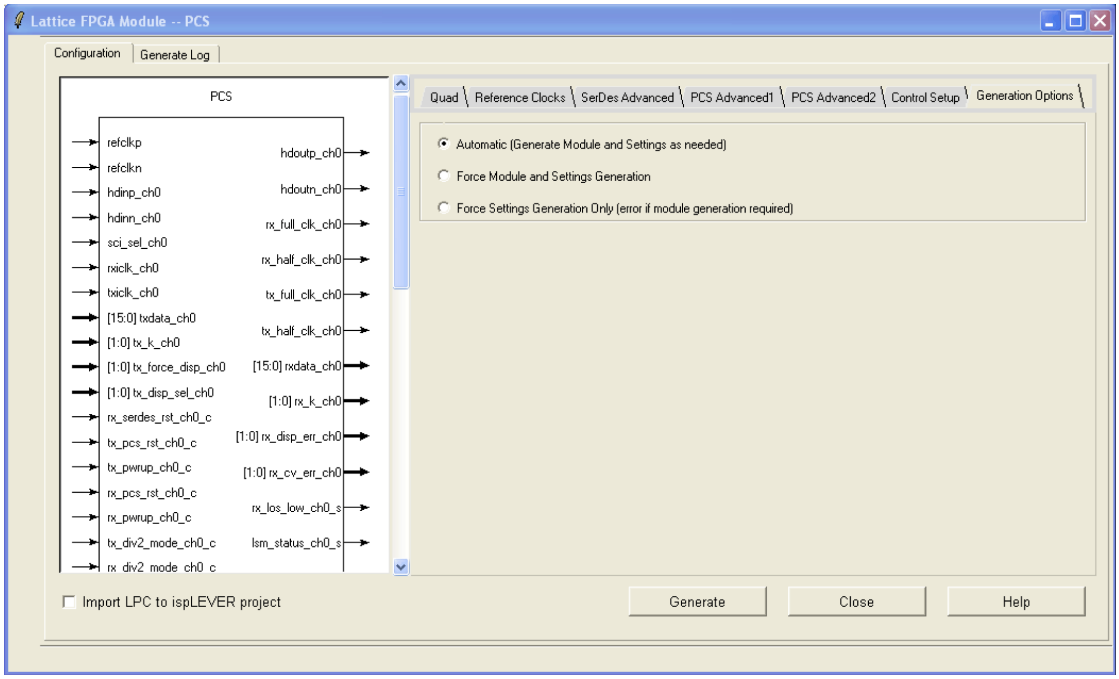
如果 Tx Refclk 时钟源选择 “Internal”，复位状态机使用内部参考时钟。

如果 Tx Refclk 时钟源选择 “External”，复位状态机使用 Reference Clock to FPGA Core 信号，对用户而言该信号可见 （控制寄存器位 QD_0A[1] 置 1）。仅当选择了 “Reference Clock to FPGA Core” 时， REFCLK2FPGA 信号将包括在 wrapper 模块中。

生成选项

该选项卡为用户提供了选择 PCS 模块生成输出文件的选项。

图 8-27. 配置图形用户界面 —— 生成选项选项卡



当第一次将之前在旧的 Diamond 版本中创建的项目移植到最新的版本中时，强烈推荐在最新的版本中重新生成 PCS 模块，即使不需要做任何配置上的更改。这将确保使用最新的原始库文件。如果不这样做的话，设计可能在后续流程中失败，或者通过流程但是会在板上显示无法预计的错误。

当在现有的项目中重新生成 PCS 模块时，通常 HDL 文件保持不变，并且仅配置文件需要重新生成。在这种情况下，用户可以运行 “Generate Bitstream Data” 和 “Force One Level” 选项以节省编译时间。

- **Automatic**—— 选择该方式时，IPexpress 将仅生成必须的文件。这可以包括 HDL 和 TXT 文件或仅 TXT 文件。这是默认设置。
- **Force Module and Settings Generation**—— 选择该方式时，将生成 HDL 和 TXT 文件。这将使项目导航器的处理过程强制复位到综合。
- **Force Settings Generation Only**—— 选择该方式时，将仅生成 TXT 文件。如果 HDL 生成是必需的，那么将会提供错误信息。
- **流程定义** —— 生成选项在两种模块处理流程的工作是不同的。
  - **HDL 源代码处理流程：项目导航器中的 HDL 文件**
    现有的 LPC 文件可以从 IPexpress 中打开，用以重新生成。在这种情况下，通过图形用户界面设置的复位点将成为新的开始点。因而，当用户双击一个处理过程或者运行 “Force One Level” 选项时，它将从复位点开始运行。
  - **LPC 源代码处理流程：项目导航器中的 LPC 文件**
    打开 LPC 文件并重新生成 PCS 模块将重新复位整个处理过程，无论 HDL 模块是否重新生成。

    在这两种情况下，处理过程窗口中的勾选记号将保持不变，但是一旦用户开始处理过程就会立即进行更新该过程的勾选记号状态。

配置文件说明

IPexpress 生成了这个文件，包含属性级映射信息。仿真模型以及位流生成的过程都是用这个文件来自动将 PCSD quad 初始化为 IPexpress 中所选的模式。

配置文件使用 “txt” 作为文件类型扩展名。

下面是一个配置文件示例。

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

DEVICE_NAME “LFE3-95E”
CH0_MODE                “RXTX”
CH1_MODE                “DISABLED”
CH2_MODE                “DISABLED”
CH3_MODE                “DISABLED”
TX_DATARATE_RANGE       “HIGH”
PLL_SRC                 “REFCLK_EXT”
REFCK_MULT              “10X”
#REFCLK_RATE            250.0

CH0_PROTOCOL            “G8B10B”
CH0_LDR                 “RXTX”
CH0_RX_DATARATE_RANGE   “HIGH”
CH0_TX_DATA_RATE        “FULL”
CH0_TX_DATA_WIDTH       “8”
CH0_TX_FIFO             “DISABLED”
CH0_CDR_SRC             “REFCLK_EXT”
#CH0_TX_FICLK_RATE      250.0
CH0_RX_DATA_RATE        “FULL”
CH0_RX_DATA_WIDTH        “8”
CH0_RX_FIFO             “DISABLED”
#CH0_RX_FICLK_RATE      250.0
CH0_TDRV                “0”
CH0_TX_PRE              “DISABLED”
CH0_RTERM_TX            “50”
CH0_RX_EQ               “DISABLED”
CH0_RTERM_RX            “50”
CH0_RX_DCC              “AC”
CH0_LOS_THRESHOLD_LO    “2”
CH0_TX_SB               “DISABLED”
CH0_TX_8B10B            “ENABLED”
CH0_RX_SB               “DISABLED”
CH0_RX_8B10B            “ENABLED”
CH0_RXWA                “ENABLED”
CH0_ILSM                “ENABLED”
#CH0_SCOMMA             “1111111111”
CH0_COMMA_A             “1100000101”
CH0_COMMA_B             “0011111010”
CH0_COMMA_M             “1111111100”
CH0_CTC                 “ENABLED”
CH0_CC_MATCH_MODE       “2”
CH0_CC_MATCH1           “0000000000”
```

```
CH0_CC_MATCH2           "0000000000"
CH0_CC_MIN_IPG          "3"

CH0_SSLB                "DISABLED"
CH0_SPLBPORTS           "DISABLED"
CH0_PCSLBPORTS          "DISABLED"
PLL_TERM                "50"
PLL_DCC                 "AC"
PLL_LOL_SET             "0"
CCHMARK                 "9"
CCLMARK                 "7"
INT_ALL                 "DISABLED"
QD_REFCK2CORE           "ENABLED"
```

8 位和 10 位 SERDES-Only 模式

本章节说明了 SERDES/PCS 模块的两种工作模式， 8 位 SERDES-Only 和 10 位 SERDES-Only。这些模式都是专为需要访问高速 I/O 接口，并且无需 LatticeECP3 PCS 逻辑提供的基于协议的操作的应用而设计的。

发送路径

- 串行器：8 位或 10 位并行数据转换为串行数据。

接收路径

- 解串行器：串行数据转换为 8 位或 10 位并行数据。
- 用户定义的对齐模式下的可选字对齐。

通用 8b10b 模式

SERDES/PCS 块的通用 8b10b 模式是专用于需要 8b10b 编码 / 解码应用，并且无需额外特定协议的数据操作而设计的。 LatticeECP3 SERDES/PCS 块可支持通用 8b10b 应用，高达 3.2 Gbps/ 通道。在通用 8b10b 模式下，字对齐器可以由嵌入式 PCS 链路状态机 （LSM）控制。

当选择并使能了嵌入式链路状态机时，一旦链接同步成功， lsm_status_ch[3:0]_s 状态信号将变为高电平。

为了该模式下的链接同步，8b10b模式下SERDES通道接收器输入(hdinp_ch[0-3]/hdinn_ch[0-3])需要满足以下条件:

- 串行数据需要周期性的使用 8b10b 编码的逗号字符。需要周期性的使用是因为一旦没有同步， LSM 可以重新与逗号字符同步。逗号字符应该对应 “ 特定逗号 ” 值，如图 8-24 所示。例如，当特定逗号值设为 K28P157，串行链路上的逗号值应该为 K28.1 (k=1, Data=0x3C)、K28.5 (k=1, Data=0xBC) 或 K28.1 (k=1, Data=0xFC) 中任意一个的 8b10b 编码值。注意：通常 K28.5 是最常用的。
- 逗号字符后要紧跟一个数据字符
- 后一个逗号字符要在前一个逗号字符后的偶数个周期上出现

其他信息：

- 需要大约四个有效的逗号 / 数据对， LSM 才能达到链路同步
- 四个连续错误 （非法的 8b10b 编码字符、编码违例、不一致 （disparity）错误、两个逗号间非偶数个时钟周期）可引发 LSM 未锁定

- CDR 失锁条件将导致 LSM 未锁定，大量的代码违例和不一致错误也将导致 LSM 未锁定
- 当使用内部复位序列状态机， CDR 失锁 (rx_cdr_lol_ch[3:0]_s) 或信号丢失 (rx_los_low_ch[3:0]_s) 条件，将导致 RX 复位序列状态机复位 SERDES 以及 LSM 未锁定

下面的两个例子说明了两个**逗号**之间有效和无效的偶数时钟周期边界的差别 （注：*C = 逗号， D = 数据*）。

有效 （偶数）逗号边界：

字时钟周期	0	1	2	3	4	5	6	7	8	9	10	11
字符	C	D	C	D	C	D	D	D	D	D	C	D

无效 （奇数）逗号边界：

字时钟周期	0	1	2	3	4	5	6	7	8	9	10	11	12
字符	C	D	C	D	C	D	D	D	D	C	D	C	D

在上面无效 （奇数）逗号边界的例子中，逗号字符出现在周期 9 和 11，因而是无效的，因为它们不是在前一个逗号字符后的偶数个周期上出现。

此外，LSM 可以被禁用，字对齐器可由 FPGA 结构的 word_align_en_ch[3:0]_c 输入引脚控制。参见第 21 页 “ 外部链路状态机选项 ” 和第 33 页 “PCS Advanced1 设置 ”，了解更多信息。

发送路径

- 串行器
- 8b10b 编码器

接收路径

- 解串行器
- 根据用户定义的字对齐字符或来自嵌入式 GbE 链路状态机的字符进行字对齐
- 8b10b 解码
- 时钟容限补偿 （可选）

千兆位以太网和 SGMII 模式下的 LatticeECP3 PCS

LatticeECP3 SERDES/PCS 块的千兆位以太网模式支持全面兼容，从串行 I/O 到 IEEE 802.3-2002 1000 BASE-X 千兆位以太网标准的 GMII/SGMII 接口。

发送路径

- 串行器
- 8b10b 编码器

接收路径

- 解串行器
- 基于 IEEE 802.3-2002 1000 BASE-X 定义的对齐字符的字对齐。
- 8b10b 解码

- 千兆以太网链路状态机符合图 36-9 中 （同步状态机， 1000BASE-X）的 IEEE 802.3-2002 标准，仅一点例外。图 36-9 要求接收 4 个连续正确的代码组，以使 LSM 从 SYNC_ACQUIRED_{N} (N=2,3,4) 变为 SYNC_ACQUIRED_{N-1}。而现在实际的 LSM 实现要求 5 个连续正确的代码组来触发转换。
- 千兆以太网载波检测：IEEE 802.3-2002 （1000BASE-X）的 36.2.5.1.4 章节定义了 carrier_detect 功能。在千兆以太网模式下，此功能不包含在 PCS 中，并且 carrier_detect 信号不会提供给 FPGA 结构。
- 时钟容限补偿逻辑能处理时钟域的差异。

千兆以太网 （1000BASE-X） Idle 插入

这对于时钟补偿和自动协商来说是必需的。自动协商是由 FPGA 逻辑实现的。莱迪思千兆以太网 PCS IP 核提供自动协商，下面将详细讨论。

Idle 模式插入对于时钟补偿和自动协商来说是必需的。自动协商是由 FPGA 逻辑实现的。该模块在自动协商过程中，自动在接收数据流中插入 /I2/ 符号。在自动协商时，链路的另一端将不断发送 /C1/ 和 /C2/ 序列集。时钟补偿器不会删除这些序列集，因为它被配置为仅插入 / 删除 /I2/ 序列集。为了防止时钟补偿器的过载 / 欠载，必须周期性地插入 /I2/ 序列集，为时钟补偿器提供插入 / 删除机会。

在执行自动协商时，该模块将每隔 2048 个时钟周期插入连续的 8 个 /I2/ 的序列集（每个占两个字节）。由于此模块在 8b10b 解码器后面，此操作将不会引入任何运行不一致 （running disparity）的错误。这些 /I2/ 序列集不会传入 FPGA 接收接口，因为在自动协商过程中，GMII 接口由 RX 状态机驱动为 IDLE。一旦自动协商完成后，禁止 /I2/ 插入，以防止破坏任何接收到的数据。

请注意，该状态机只在自动协商时有效。自动协商状态机和 GbE 接收状态机在软逻辑中实现。该状态机由自动协商状态机的信号 xmit_ch 决定。这个信号是在 TX 数据总线上提供。虽然这个信号是相对静态的 （尤其是在自动协商后），它包含在 TX 数据总线中。

表 8-17. GbE IDLE 状态机控制和状态信号

模块信号	方向	说明
xmit_ch[3:0]	In	来自 FPGA 逻辑自动协商状态机

千兆以太网 Idle 插入和 correct_disp_ch[3:0] 信号的使用

correct_disp_ch[3:0] 信号用于 PCS 的发送端，以确保数据包间间隙在 negative disparity 状态下开始。请注意，在以太网帧最后，发送器的当前 disparity 状态可以是 positive 或者 negative，根据以太网帧的大小和数据内容决定。

然而，从 PCS 的 FPGA 软逻辑端，PCS 发送器目前的 disparity 状态是未知的。这就是 correct_disp_ch[3:0] 信号的作用。如果一旦进入一个数据包间间隙，发出 correct_disp_ch[3:0] 信号一个时钟周期，当目前是 positive disparity 时，将强制 PCS 发送器在发送数据流中插入一个 IDLE1 序列集。但是，如果目前是 negative disparity，则不对发送数据流做任何改变。

从 PCS 的 FPGA 软逻辑端，数据包间间隙为不断发送的 IDLE2 序列集如下：tx_k_ch=1，txdata= 0xBC tx_k_ch=0，txdata=0x50。

请注意，在 PCS 通道中， IDLE2 意味着目前的 disparity 状态将被保留。 IDLE1 意味着目前的 disparity 状态应当翻转。那么，就有可能可以确保数据包间间隙以 negative disparity 状态开始。如果在数据包间间隙前的 disparity 状态为 negative，那么数据包间间隙将不断发送 IDLE2。如果在数据包间间隙段前的 disparity 状态为 positive，那么将发送一个 IDLE 后面跟连续的 IDLE2。

在 PCS 的 FPGA 软逻辑端，数据包间间隙总是将 IDLE2 输入 PCS。当第一次出现数据包间间隙时，将发出一个时钟周期的 correct_disp_ch[3:0] 信号，k_cntrl=0，data=0x50。如果需要的话，PCS 将会把这个 IDLE2 转换为 IDLE1。在余下的数据包间间隙， IDLE2 应输入 PCS 并且 correct_disparity_chx 信号应保持无效。

例如，如果发送连续的 512 字节以太网帧和 512 字节 /I/，可以观察到：

- 在第一个数据包间间隙，可以看到所有的 negative disparity /I2/ （K28.5(-) D16.2(+)）
- 在下一个数据包间间隙，周期以 positive disparity /I1/（K28.5 (+), D5.6 (+/- 相同)）开始，然后所有剩下的序列都是 negative disparity /I2/
- 在下一个数据包间间隙，可以看到所有的 negative disparity /I2/
- 在下一个数据包间间隙，周期以 positive disparity /I1/（K28.5 (+), D5.6 (+/- 相同)）开始，然后所有剩下的序列都是 negative disparity /I2/

编码器模块支持许多可编程选项。它们是：

- 按每个字强制 negative 或 positive disparity
- 直接从 FIFO 桥输入数据 —— 外部的多路复用器
- 根据运行差异 （running disparity）（100BASE-X 和 FC）改变编码字
- 软件寄存器控制的旁路模式

XAUI 模式

使用莱迪思 XAUI IP 核， SERDES/PCS 块的 XAUI 模式支持全面兼容，从串行 I/O 到 IEEE 802.3-2002 XAUI 标准的 XGMII 接口。 XAUI 模式支持 10 千兆以太网。

发送路径

- 串行器
- 发送状态机根据 IEEE 802.3ae-2002 规范，将 XGMII idle 转化为相应的 ||A||、 ||K||、 ||R|| 字符。
- 8b10b 编码

接收路径

- 解串行器
- 基于 IEEE 802.3-2002 定义的对齐字符的字对齐 。
- 8b10b 解码
- XAUI 链路状态机符合图 48-7——IEEE 802.3ae-2002 标准的 PCS 同步状态图，仅有一点例外。图 48-7 要求接收 4 个连续正确的代码组，以使 LSM 从 SYNC_ACQUIRED_{N} (N=2,3,4) 变为 SYNC_ACQUIRED_{N-1}。而现在实际的 LSM 实现要求 5 个连续正确的代码组来触发转换。
- PCS 中的时钟容限补偿逻辑在 XAUI 模式下是禁止的。MCA（Multi-Channel Alignment, 多通道对齐）及 CTC 是在 XAUI IP 核中实现的。
- x4 多通道对齐应当在 FPGA 内核逻辑中实现。

PCI Express 版本 1.1 （2.5Gpbs）模式下的 LatticeECP3 PCS

SERDES/PCS 块的 PCI Express 模式支持 x1、 x2 和 x4 PCI Express 应用。

发送路径

- 串行器
- 8b10b 编码
- 接收器检测
- 电气空闲状态

接收路径

- 解串行器
- 基于 Sync 编码的字对齐
- 8b10b 解码
- 链路同步状态机功能包括了在 IEEE 802.3ae-2002 10GBASE-X 规范下的 PCS 同步状态机（图 48-7）中定义的操作。
- x2 或 x4 PCI Express 操作，可用一个 PCS quad 设为 PCI Express 模式。
- 时钟容限补偿逻辑能处理时钟域的差异。
- x2 或 x4 多通道对齐应当在 FPGA 内核逻辑中实现。

表 8-18 说明了 PCI Express 模式下的特定端口。

表 8-18. PCI Express 模式特定端口

信号	方向	类别	说明
pcie_done_ch[3:0]_s	Out	通道	1 = 远端接收器检测完毕 0 = 远端接收器正在检测
pcie_con_ch[3:0]_s	Out	通道	远端接收器检测的结果。 1 = 检测到远端接收器 0 = 未检测到远端接收器
pcie_det_en_ch[3:0]_c	In	通道	FPGA 逻辑（用户逻辑）通知 SERDES 块，它要申请一次 PCI Express 接收器检测操作。 1 = 使能 PCI Express 接收器检测 0 = 正常工作
pcie_ct_ch[3:0]_c	In	通道	1 = 申请发送器进行远端接收器检测 0 = 正常工作
rxstatus[2:0]	Out	通道	每个通道 PCI Express 接收状态端口。 RxStatus# 是接收数据路径的一种编码状态。如果在 16 位数据总线模式下是 2 位宽。

状态信号 rxstatus 是接收数据路径的编码状态。编码如下所示。

表 8-19. rxstatus 编码

rxstatus[2:0]			说明	优先级
0	0	0	接收到的数据正常	8
0	0	1	CTC 插入 1 个字节	7
0	1	0	CTC 删除 1 个字节	6
0	1	1	检测到接收器 （pcie_done， pcie_con）	1
1	0	0	8b10b 解码错误 （编码违例 ——rx_cv_err）	2
1	0	1	CTC FIFO 上溢 （ctc_orun）	3
1	1	0	CTC FIFO 下溢 （ctc_urun）	4
1	1	1	接收不一致 （disparity）错误 （rx_disp_err）	5

PCI Express 端接

在电气层， PCI Express 采用了两根单向的低压差分信号对，每个通道速率为 2.5Gbps。发送和接收是独立的差分对，每个通道总共有 4 根数据线。通过带有可编程均衡器的输入接收器和带有可编程预加重的输出发送器优化了链路。 PCI Express 规范要求差分线路必须在接收端使用共模端接。每条链路要求在远端 （接收器）使用一个端接电阻。使用的标称电阻值为 100 欧姆。这是通过使用如图 8-28 所示的 CML 输入的嵌入式端接特性实现的。该规范要求在链路的发送端使用交流耦合电容（CTX）。这消除了发送和接收器件之间潜在的共模偏置的不匹配。莱迪思 CML 输出必须外加该电容。

PCI Express L2 状态

对于 PCI Express L2 状态， rx_pwrup_c 信号不应该变为无效，从而使 rx 通道掉电。这将强制 rx 端接为高阻状态，并不允许远端检测到接收器。而应该使用 rx_pcs_rst_c 信号将通道保持在复位状态来节省功耗。

图 8-28. PCI Express 接口图

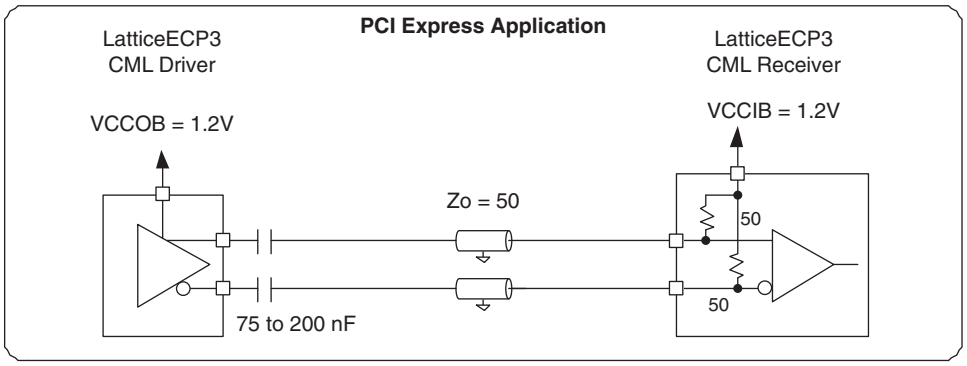


表 8-20. 差分 PCI Express 规范

信号	参数	最小值	正常值	最大值	单位	说明	位置
ZTX-DIFF-DC	直流差分 TX 阻抗	80	100	120	Ohm	TX 直流差分模式低阻抗。 ZTX-DIFF-DC 是发送器的小信号阻抗，在直流工作点上测得，等于当 TX 在驱动静态逻辑信号 1 或者 0 时，在 D+ 和 D- 之间连接一个 100 欧姆电阻。	内部
ZRX-DIFF-DC	直流差分输入阻抗	80	100	120	Ohm	所有 LTSSM 状态下的 RX 直流差分模式阻抗。当发送从 Fundamental Reset 变为 Detect 时，（LTSSM 的初始状态），在满足一个端口上所有未配置的线路上的接收器端接值之前，需要 5 ms 转换时间。	内部
CTX	交流耦合电容	75		200	nF	所有的发送器都是交流耦合的。传输介质或者发送元件本身都需要交流耦合。	外部

PCI Express Electrical Idle 发送

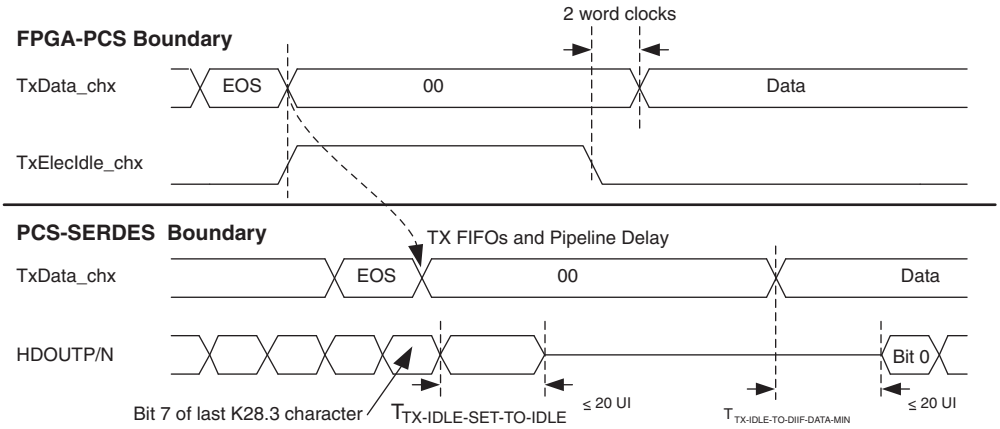
Electrical Idle 是一个稳定状态，发送器 P 和 N 电压都保持在相同的恒定值（即，Electrical Idle 差分峰值输出电压，VTX-IDLE-DIFFp，在 0 和 20mV 之间）。 Electrical Idle 主要用于节省功耗和非激活状态。

根据 PCI Express 基本规范，在发送器进入 Electrical Idle 之前，通常必须发送 Electrical Idle 序列 （EIOS），一个 K28.5（COM）之后有三个 K28.3（IDL）。在发送完 Electrical Idle 序列的最后一个符号后，发送器必须在一个 TTX-IDLE-SET-TO-IDLE 规定的小于 20UI 时间内进入有效的 Electrical Idle 状态。

要做到这一点， FPGA 内核逻辑至 PCS 的 Electrical Idle Enable （tx_idle_chx_c）随每个发送数据一起发送。该信号以类似于流水线的方式发送到 PCS-SERDES 边界。对于所有有效数据来说，该信号为 LOW。为初始化 Electrical Idle，在它发出最后一个 K28.5 （IDL）字符后的这个时钟， FPGA 逻辑将该信号变为 HIGH。因为当信号以流水线方式发送到 PCS-SERDES 边界，发送数据和该信号之间的关系和在 FPGA-PCS 边界上的完全相同。

在 PCS-SERDES 边界，Electrical Idle Enable 信号处于上升边沿的 14UI 后，发送最后一个字符 K28.3（IDL）的最后一位 （bit7）。在 16UI （<20UI）后，发送差分缓冲器实现了 Electrical Idle 状态。

图 8-29. 发送 Electrical Idle



只要 FPGA 内核逻辑认为发送器需要保持 Electrical Idle 状态，它需要用时钟打入数据（最好是全零）以及 Electrical Idle Enable 信号 （tx_idle_chx_c）为有效 （高电平）。发送器必须保持在 Electrical Idle 状态，最少 50UI （20ns）（TTX-IDLE-MIN）。

PCI Express Electrical Idle 检测

quad 中的每条通道都有一个信号丢失检测器。一旦 Electrical Idle 序列 （EOS）的三个 K28.3 （IDL）中的两个接收到后，即为检测到 Electrical Idle。在 Electrical Idle 序列收到后，接收器应当至少等待 50ns （TTX-IDLE-MIN），然后使能 Electrical Idle Exit 检测器。

这些信号（每个通道一个，每个 quad 四个）应当从 PCS 输出，并且应当可以用于 FPGA 内核。从而，支持 electrical idle 所需的状态机可以在 FPGA 内核实现。

PCI Express 接收器检测

图 8-30 显示了一个接收器检测先后次序。接收器检测测试可以在一个 quad 的每条通道独立进行。在开始接收器检测测试之前，必须通过将 tx_idle_ch#_c 输入设为高电平来使发送器进入 electrical idle。接收器检测测试可以通过将相应的 pci_det_en_ch#_c 驱动为高电平，在 tx_elec_idle 变为高电平的 120 ns 后开始。这通过将驱动器端接电阻设为高阻抗，并通过高阻抗的驱动器端接电阻将两个差分输出拉高到 VCCOB，使相应的 SERDES 发送缓冲器进入接收器检测模式。

将 SERDES 发送缓冲器设为接收器检测状态需要高达 120 ns。 120 ns 后，接收器检测测试可以通过将通道的 pcie_ct_ch#_ 输入驱动为高电平四个字节（字）的时钟周期来启动。然后，相应通道的 pcie_done_ch#_s 被异步清零。经过足够的时间来完成接收器检测测试后 （由发送端的时间常数决定）， pcie_done_ch#_s 接收器检测状态端

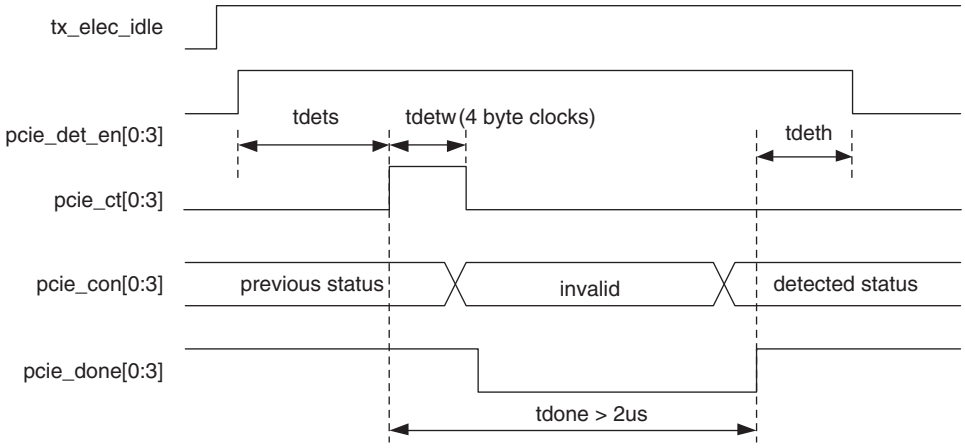
口将变为高电平，并且接收器检测状态可以通过 pcie_con_ch#_s 端口来监测。如果同时 pcie_con_ch#_s 端口为高电平，那么该通道的接收器就被检测到了。但是，如果 pcie_con_ch#_s 端口为低电平，那么该通道接收器没有被检测到。一旦接收器检测测试结束， tx_idle_ch#_c 信号可以变为无效。

接收器检测处理如下：

1. 用户将 pcie_det_en 驱动为高电平，使相应的 TX 驱动器进入接收器检测模式。这将驱动器端接电阻设为高阻抗 （5K 欧姆）并通过高阻抗驱动器终端电阻将两个差分驱动器的输出变为共模状态。 TX 驱动器需要一些时间来进入这个状态，因而 pcie_det_en 必须驱动为高电平至少 120ns，然后 pcie_ ct 信号有效。

2. 用户将 pcie_ct 驱动为高电平四个字节时钟。
3. SERDES 将相应的 pcie_done 驱动为低电平。
4. SERDES 按其所需时间 （基于时间常数）驱动内部信号 （对应 pcie_ct）来检测接收器。
5. SERDES 驱动相应的 pcie_con 连接状态。
6. SERDES 将相应的 pcie_done 驱动为高电平。
7. 用户可以根据 pcie_done 信号的有效状态来采样 pcie_con 状态，以决定接收器检测是否成功。

图 8-30. PCI Express 模式接收器检测序列



PCI Express 掉电模式的使用

应当使用 rx_serdes_rst_ch[3:0] 复位信号，而非 rx_pwrup_ch[3:0] 信号。这使得 RX 终端保持在 50 欧姆，以使远端发送器可检测到接收器已连接。

PCI Express Beacon 支持

本章节重点说明 LatticeECP3 PCS 是如何支持 Beacon 检测和发送的。用于 Beacon 检测的 PCI Express 要求已经与 Beacon 发送和 Beacon 检测所需的 PCS 支持一起说明。

Beacon 检测要求

- 对于从 L2 （P2）状态的退出来说， Beacon 是必须的。
- Beacon 是周期随机数据的一个直流平衡信号，需要包含一些脉冲宽度 ≥ 2ns （500 MHz）且 < 16us （30 Khz）。
- 脉冲间的最大时间应 < 16 us。
- 直流平衡必须在 < 32 us 内恢复。
- 对于脉冲宽度 > 500 ns，输出 beacon 电压必须比 VTX-DIFFp-p （800 mV 至 1200 mV）低 6 db。
- 对于脉冲宽度 < 500 ns，输出 beacon 电压必须 ≤ VTX-DIFFp-p 且比 VTX-DIFFp-p 低 ≥ 3.5 db。

PCS Beacon 检测支持

- 信号丢失阈值检测电路检测在接收器缓冲器端规定的电压是否达到。
- 这由 rlos_lo_ch(0-3) 信号标识。

- 该设置可用于 PCI Express Electrical Idle 检测以及 PCI Express beacon 检测 （在电源状态 P2 下）。
- 远端发送器件的 beacon 输出电压可以比 VTX-DIFFpp 低 6 db（即 201 mV）。如果检测到该信号，那么就检测到 beacon。

PCS Beacon 发送支持

发送 K28.5 字符 （IDLE）（5 个 1 后面跟 5 个 0），每隔 2 ns 提供 2 ns 的周期脉冲宽度 （1.0 UI = 400 ps，乘以 5 = 2 ns）。这满足了较低的要求。输出 beacon 电压可为 VTX-DIFFp-p。这是有效的 beacon 发送。

SDI （SMPTE）模式

LatticeECP3 SERDES/PCS 块的 SDI 模式支持所有三种 SDI 模式， SD-SDI、 HD-SDI 和 3G-SDI。

发送路径

- 串行器

接收路径

- 解串器
- 根据用户定义的对齐模式的可选的字对齐。

下面是广播视频行业中最常用的数据速率。

- SD-SDI （SMPTE259M）：270Mbps
- HD-SDI （SMPTE292M）：1.485Gbps， 1.485Gbps/1.001 = 1.4835Gbps
- 3G-SDI （SMPTE424M）：2.97Gbps， 2.97Gbps/1.001 = 2.967Gbps

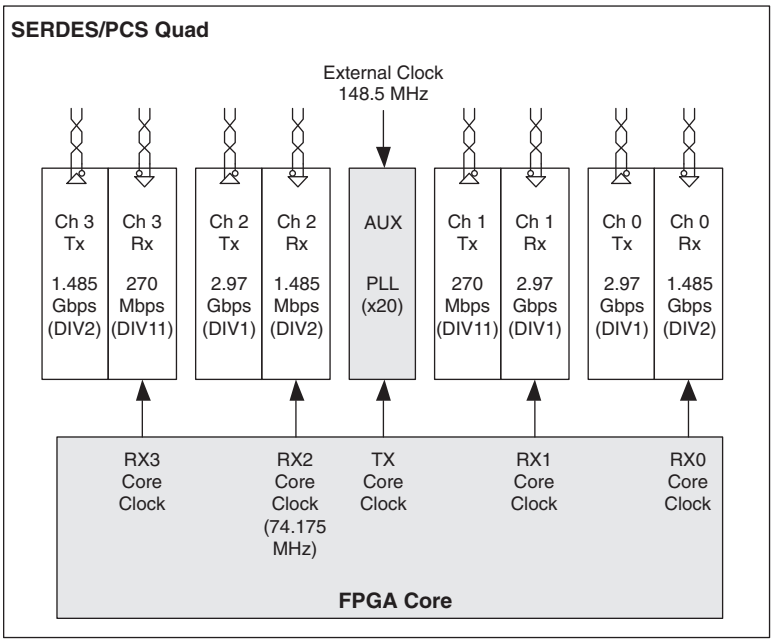
大多数设计人员提出，他们希望能够支持所有这些速率。因为这样的话，在广播台或者卫星前端或电缆前端，他们就无需事先知道 RX 的数据速率。

不同速率之间的切换时间越短越好。重新锁定 CDR 的时间是不可避免的。在 LatticeECP3 SERDES 中， PLL 不需要重新锁定。这是因为 LatticeECP3 的每个 RX 和 TX 都有分频器。视频链路通常有单向性 （即：不同的通道可以以不同的速率运行，更重要的是，同一个通道中的 RX 和 TX 可以以不同的速率运行）。

并且，根据设备部署的地区，要么使用全 HD/3G-SDI 速率 （欧洲 / 亚洲）来传输视频，要么使用分数速率 （北美——NTSC）。这使我们能够开发两种可用的解决方案样例，提供高 quad 利用率的多速率 SMPTE 支持。

请注意，同一个 SERDES Quad 中不能同时支持 3G/HD 全速和部分 TX 速率。一般来说，根据上述讨论，不同地区使用不同的速率，因而这一限制是可以接受的。

图 8-31. 示例 A：3G/HD/SD 全 RX/TX 速率支持以及 3G/HD 分数 TX 速率支持



为了支持大多数的应用需求，提供针对每个 RX 和 TX 的可选的 DIV 支持。在 LatticeECP3 中，已经添加 DIV11。一个可用的多速率配置是从主引脚到 TX PLL，提供一个 148.5MHz REFCLK。 TX PLL 将为 x20 模式。由此产生的输出时钟将为 2.97GHz。然后，通过在 1.485Gbps 使用 DIV2 并且在 270Mbps 使用 DIV11 实现快速切换，并且无需重新训练和锁定 PLL。

串行 RapidIO （SRIO）模式

本章节说明了 SERDES / PCS 块的串行 RapidIO 模式的工作。LatticeECP3 支持使用一个 PCS quad 的 1x 和 4x 串行 RapidIO 应用。SRIO1.0 突出了其具有多频支持，包括 3.125Gbps、 2.5Gbps 和 1.25Gbps。这些速率的比例是 2.5:2:1。通过整数分频器在同一个 quad 中支持所有这些速率是不可能的，但 2.5 Gbps 和 1.25 Gbps 的比例为 2:1（全速率：半速）。

发送路径

- 串行器
- 8b10b 编码

接收路径

- 解串器
- 基于在 RapidIO 物理层 1x/4x LP—— 串行规范中定义的同步编码组的字对齐。
- 8b10b 解码
- 时钟容限补偿逻辑能处理时钟域的差异。

串行数字视频和带外低速 SERDES 工作

LatticeECP3 SERDES/PCS 支持任意低于 SERDES TX PLL 和 RX CDR 本身支持的数据速率 （<250Mbps：带外信号， OOB），通过旁路接收器 CDR 和相关的 SERDES/PCS 逻辑 （如：100Mbps 快速以太网， 143Mbps 或 177Mbps 的 SD-SDI）。尽管这些带外路径大多使用低数据速率，更高速率可以用于其他功能。请参见本文档的多速率 SMPTE 支持章节，了解更多信息。

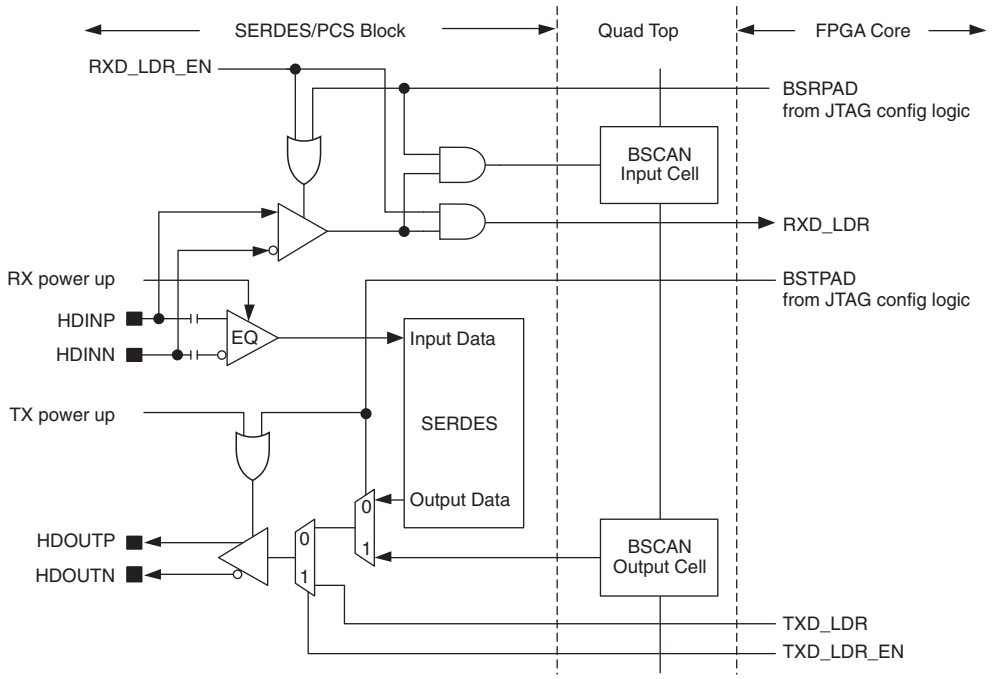
此外，对于 SD-SDI，这些速率有时必须与 HD-SDI 速率共用同一个差分 RX 对 （即：SD-SDI 速率可能有效，然后数据速率可能切换为 HD-SDI 速率）。因为没有办法预测这两种速率哪一种是有效的，因此可以并行发送输入数据流到两个 SERDES，一个高速 SERDES（已经在 quad 中）以及一个低速 SERDES（在 quad 外实现）。一种可能的实现如图 8-32 所示。

每个通道有一个 RXD_LDR 输入，即低数据速率单端输入从 RX 缓冲器到 FPGA 内核。在内核中，可使用软逻辑构建一个低速时钟数据恢复（CDR）模块或一个数据恢复单元（DRU）。一个通道寄存器位 RXD_LDR_EN 可以使能该数据路径。如果通过其他寄存器位使能，来自 FPGA 的信号也可以在 LatticeECP3 中使能它。

在发送方向，也可以使用一个在 FPGA 内核中通过软逻辑构建的串行器并使用 TXD_LDR 引脚来发送数据到 SERDES。它将在预加重逻辑之前进行多路复用输入，靠近与边界扫描路径多路复用的常用高速 SERDES 路径。概念图如图 8-32 所示。低数据率路径可通过设置一个通道寄存器位 TX_LDR_EN 来选择。

另外，在输出端，高速 SERDES 用于发送高速数据或使用抽取方式来发送低速数据 （SERDES 持续以高速运行，但是输出数据仅可以在第 n 个时钟周期时进行变化，其中 n 是一个抽取因子）。

图 8-32. 可能的串行数字视频支持的实现



开放式基站架构计划 （OBSAI）

OBSAI 是一个开放的论坛，旨在形成蜂窝基站的一个开放的市场。

LatticeECP3 SERDES/PCS 支持大多数 OBSAI 特性，除了 3.84Gbps 速率。

发送路径

- 串行器
- 发送状态机设为千兆以太网模式
- 8b10b 编码

接收路径

- 解串行器
- 基于 IEEE 802.3-2002 1000 BASE-X 定义的对齐字符的字对齐
- 8b10b 解码

一个基站收发机系统 （Basestation Transceiver System， BTS）有四个组件 / 模块，它们之间有三个主要接口或者参考点 （Reference Points， RP）。

- **RP3** – RF 模块从便携式设备 （终端）接收信号并向下转换为数字数据。
- **RP2** – 基带模块获取编码的信号并进行处理，然后发送到传输模块，它将通过地面网络进行发送。
- **RP1** – 一个包含这三个功能之间的协调的控制模块。

目前，业界主要关注的是提供较低功耗的射频模块和功率放大器，因此 OBSAI 的主要工作是定义参考点 3（RP3）。事实上，现在关注的规范是 RP3-01，即射频拉远单元。

OBSAI RP3 电气规范是基于 XAUI 电气规范并根据基站收发机系统的需求进行定制化。 XAUI 电气接口符合 IEEE 802.3ae-2002. RP3 3.1 版的 Clause 47 规定，规定了支持以下速率 3.84Gbps、3.072Gbps、2.304Gbps、1.536Gbps 和 0.736Gbps 中的后四个。

RP3 电气特性定义了接收器符合的掩码，并提供了一个示例发送器输出掩码。BER 应优于 1 x 10-15，它比 XAUI 要求的 1 x 10-12 更为严格。RP3 电气规范在 UI 上的规定也不同于 XAUI 规范。XAUI 允许 +/- 100ppm 的差异。该差异不适用于 OBSAI 系统，由于 BTS （Base Transceiever Station）是完全同步的系统。

由于 BTS 是一个同步系统，它必须通过测量并校准经过任何总线的延迟。OBSAI 已经非常仔细地考虑到这点，并且提出一个在 RP3 链路上进行主帧同步的方法 （将在之后的章节进行更多有关帧的讨论）。延迟校准考虑到所有因素，包括处理和收发模块的缓冲器延迟以及链路上的延迟。

数据链路层的另一个主要考虑是发送器和接收器之间的同步。同步确保了链路上的实际数据可以成功解码。错误的频率以及同步的状态一直受到监控。

RP3-01 已经进一步规范了线路速率，应为 768Mbps 的整数倍，最高可达 3.84Gbps，并且也考虑了 OBSAI 兼容的线路速率。由于可用线路速率数量有限，远程射频单元和本地单元之间的自协商已经定义。这个规范扩展的内容，包括两个 RP3-01 节点间的以太网传输、将 RP1 信息映射到 RP3 链路 （由于 RRH 没有实际的 RP1 链路）、延迟测量、 RP3-01 单元间的同步以及数据在 RP3-01 链路上的多路复用。

LatticeECP3 SERDES/PCS 中每个功能模块的延迟在 CPRI 章节中进行说明。

通用公共无线接口 （CPRI）

CPRI 的目的是允许基站制造商和元件供应商能共享公共协议并更方便地适用于不同客户的平台。

发送路径

- 串行器
- 发送状态机设为千兆以太网模式
- 8b10b 编码

接收路径

- 解串行器
- 基于 IEEE 802.3-2002 1000 BASE-X 定义的对齐字符的字对齐
- 8b10b 解码

与 OBSAI 不同的是， CPRI 不指定机械或者电气接口规范。从范围上看， CPRI 比 OBSAI 的带宽窄得多。 CPRI 单纯只看 RRH 和基带模块之间的链路。在 CPRI 中，这些模块分别被称为无线电设备 （RE）和无线电设备控制（REC）。换言之， CPRI 规定了与 OBSAI RP3 规范相同的接口。CPRI 的主要内容包括接口的物理和数据链路层。它还规定了如何传输用户层数据、控制和管理 （C&M）层数据以及同步层的数据。

两大原因成为了 CPRI 更大的推动力 —— 很多公司的支持以及仅专注于一种接口连接 （在射频模块和基带模块之间），甚至主要针对的是物理层和数据链路层。

CPRI 允许 4 种线路位速率选择：614.4Mbps、 1.2288Gbps、 2.4576Gbps 和 3.072Gbps ；至少支持这些速率中的一个。更高的线路速率通常会与稍低的线路速率相比。

CPRI 没有物理层协议的强制性要求，但使用的协议必须达到 1 × 10-12 的 BER 要求，这比 OBSAI 规范稍宽松一些。它还规定了时钟稳定性和相位噪声的要求。

CPRI 还推荐了两个电气变量：高电压 （high voltage， HV）和低电压 （low voltage， LV）。 HV 根据 IEEE 802.3-2002 中 1000Base-CX clause 39 在 100 欧姆阻抗时来指定。LV 根据 XAUI 来指定。LV 推荐用于所有速率，并且是本器件的关注焦点。

在处理 CPRI 和 OBSAI 规范时，理解两个链路层的需求是很重要的：

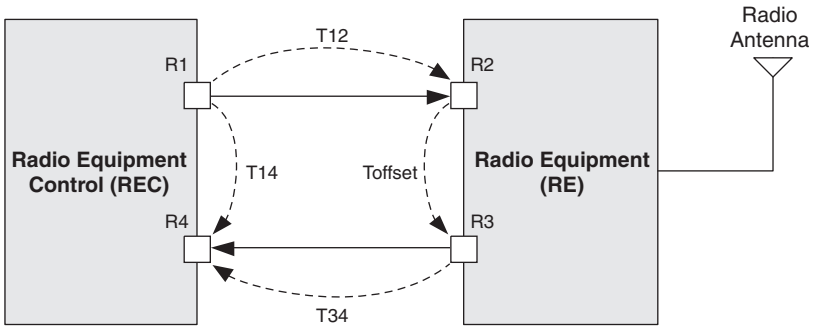
- 链接延迟准确性和电缆延迟校正
- 启动同步

链路延迟准确性和电缆延迟校准

虽然下面主要是从 CPRI 的要求来讨论，同样的要求也适用于 OBSAI 实现。

RE 或 RRH 的频率锁定至 REC 或 BTS。因此，在这个同步系统中，为了满足空中接口时序要求，校准 RRH 和 BTS 之间的所有延迟是十分必要的。该接口需要基本机制的支持，以校准链路上的电缆延迟和单跳和多跳连接的往返延迟。具体而言，定义了延迟校准的参考点以及 RE （无线电设备）输入输出信号之间的时序关系。在单跳情况下，REC 主端口和 RE 从端口之间的链路的所有定义和需求如图 8-33 所示。

图 8-33. REC 主端口和 RE 从端口之间的链路 （单跳情况下）



参考点 R1-4 对应 REC 的输出点 （R1）和 REC 的输入点 （R4），并且 RE 的输入点 （R2）和输出点 （R3）终止了一种特定的逻辑连接。天线如 Ra 所示，以供参考。

- T12 是从 REC （R1）输出点到 RE （R2）输入点的下行链路信号的延迟，实际上是下行链路电缆的延迟。
- T34 是从 RE （R3）输出点到 REC （R4）输入点的上行链路信号的延迟，实际上是上行链路电缆的延迟。
- Toffset 是 R2 的输入信号和 R3 的输出信号之间的帧偏置。
- T14 是 R1 的输出信号和 R4 的输入信号之间的帧时序差异 （即往返延迟 ——RTT）。

延迟测量是使用帧时序来实现的。 CPRI 有一个 10ms 帧基于 UMTS 射频帧号或 B 节点帧编号，也称为 BFN。每个 UMTS 无线帧有 150 个超帧 （hyperframe）（即每个超帧是 66.67us），带有相应超帧号 （HFN = 0<=Z<=149）。每个超帧有 256 个 （0<=W<=255）基本帧 （即每个基本帧是 260.42ns = Tchip 或 Tc）。

RE 决定了其输出信号 （上行链路）的帧时序相对于其输入信号 （下行链路）的帧时序是固定偏置 （Toffset）的。Toffset 是一个任意值，其大于等于 0 且小于 256 Tc（它不能超出一个超帧大小）。不同的 RE 可能使用不同的 Toffset 值。 REC 事先知道每个 RE 的 Toffset 值 （预先定义的值或 RE 通过更高层消息通知 REC）。

为了确定 T14，从 REC 到 RE 的下行链路 BFN 和 HFN 将返回给从 RE 到 REC 的上行链路。如果上行链路发出错误条件信号， REC 认为上行链路 BFN 和 HFN 是无效的。因此， T14 = T12 + Toffset + T34。

如前所述，系统是同步的。此外，假设超帧是固定长度的并且 RRH-BTS 互连（电缆长度）在两个方向上都相同（即 T12 = T34，两根光纤是在同一光纤束中），互连延迟传输后变为 (T14 - Toffset)/2。确定 T14 的方法之前已经讨论过。因而，影响延迟校准的主要因素是 Toffset。因而，互连延迟仅是在链路的每侧测得的超帧到达和离开的时间差。

延迟校准要求根据 3GPP 和 UTRAN 规范，特别是 CPRI 规范中的要求 R-21（CPRI v3.0 第 20 页），其中规定，一条链路的电缆延迟的往返延迟测量值精度为 +/- Tc/16。此外，R-20 规定了接口的往返延迟时间的绝对精度，不包括在传输介质上的往返延迟 （不考虑电缆长度）应当满足类似的要求 （T14 为 +/- Tc/16） 。考虑到前面的讨论，在 REC 主端口和 RE 从端口之间的下行链路上的绝对链路延迟精度，不考虑电缆长度，应为上述规定的一半（+/- Tc/32 或大约为 8ns （8.138ns））。因此， T14 和 Toffset 都需要绝对精度小于 +/- 8ns。

下一步重要的是要确定对于不同的速率，多少不确定的位数是可以接受的。从本质上讲，各 CPRI 和 OBSAI 比特率可以乘以 8.138ns 来确定可接受的非确定值 / 变量的位数。当讨论了 SERDES 的串行 / 并行数据路径之后，其影响结果将非常清晰。

大多数 SERDES 有一定程度的不确定性，会引入到串行和解串的过程中。因此，由于每个字的位数翻倍了， 16 位总线架构的 SERDES 的延迟不确定性可能是 8 位总线架构的 SERDES 的两倍。

TX 和 RX 延迟分别在表 8-23 中列出。表中还列出了延迟之间的可变性。该可变性直接造成了前面讨论中所要求的绝对延迟精度。可变性来自三个方面：TX FPGA 桥 FIFO、RX FPGA 桥 FIFO 和 RX 时钟容限补偿 FIFO。由于 CPRI 系统是一个同步系统， RX CTC FIFO 被旁路并且使用 RX 恢复时钟。

造成延迟可变性的其他因素还有 FPGA 桥 FIFO。这个 FIFO 可以被旁路，如果接口到 FPGA 的是 8 位总线模式。在 16 位接口模式下， FPGA 桥 FIFO 不能被旁路，由于 2:1 gearing 是通过 FIFO 实现的。

SONET/SDH

同步光纤网络 （Synchronous Optical Networking， SONET）和同步数字体系 （Synchronous Digital Hierarchy，SDH）是标准化的复用协议，通过光纤或电气接口传输数据。 SONET 通用标准在 Telcordia Technologies Generic Requirements 文件 GR-253-CORE 中进行了详细说明。适用于 SONET 和其他传输系统（如异步光纤系统或数字无线系统）的通用标准可以在 Telcordia GR-499-CORE 中找到。SONET 和 SDH 最初设计用于传送来自于各种不同源的电路模式通信 （如 T1， T3）。在 SONET 之前，实现电路间通信的最大难点是，这些不同电路的同步源是不同的。这意味着每个电路都是以一个略有不同的速度和相位工作。SONET 允许在一个帧协议下，同步传输来自许多不同源的不同电路。

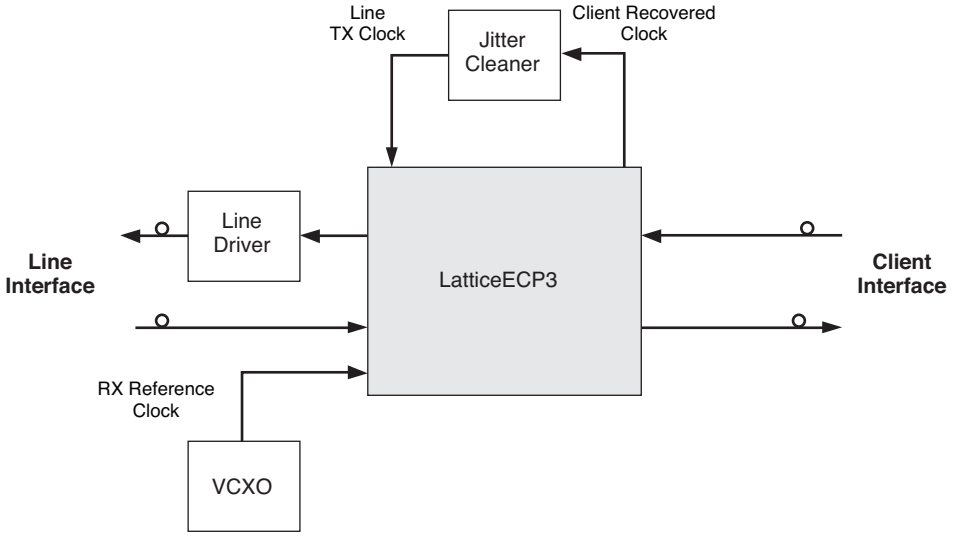
LatticeECP3 SERDES/PCS 为收发器提供了三种 SONET/SDH 数据速率支持：STS-3/STM-1 （155.52 Mbps）、STS-12/STM-4（622.08 Mbps）以及 STS-48/STM-16（2.488 Gbps）。SONET/SDH 应用使用 8 位 SERDES 模式。

为了实现 SONET/SDH 线路兼容，LatticeECP3 还需要外部元件。SERDES 的输出端需要一个外部线路驱动器。为了过滤来自输入数据流的高频抖动，需要在将恢复时钟用作发送参考时钟之前，使用一个抖动清除器。

对于片到片或者背板应用，不需要外部线路驱动器和时钟抖动清除器。

图 8-34 显示了使用外部元件的线路端解决方案。

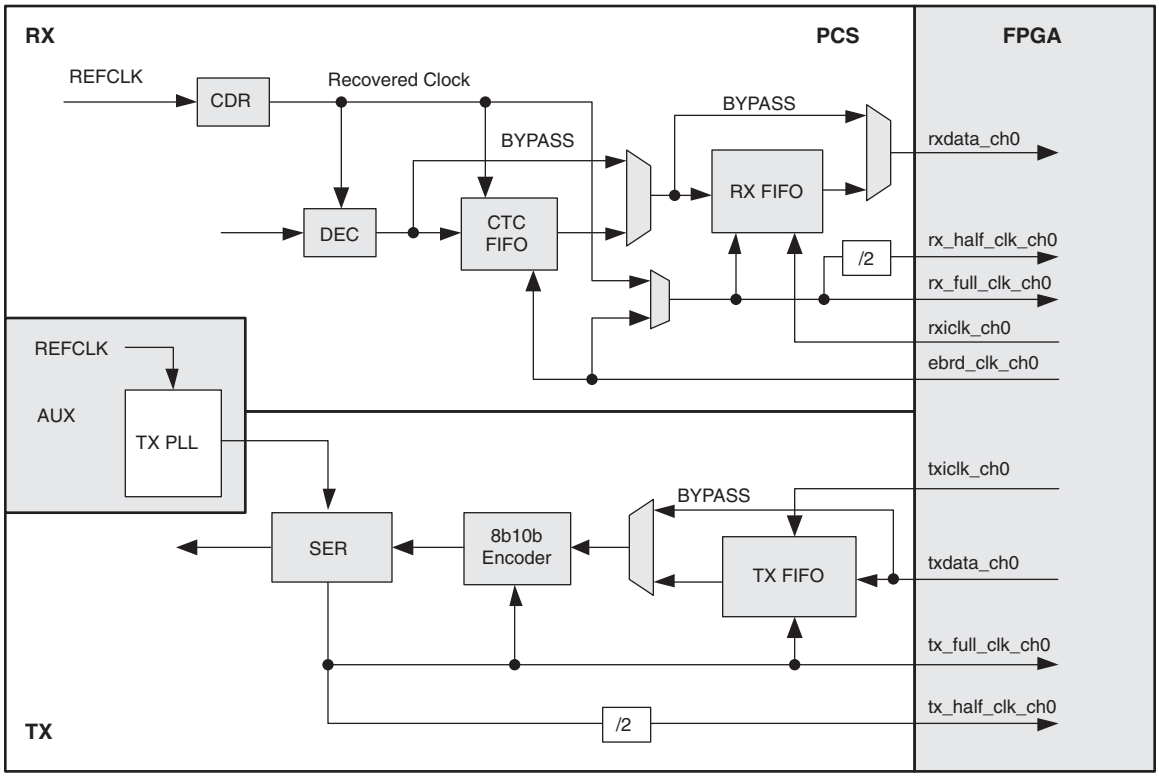
图 8-34. SONET/SDH 线路接口



FPGA 接口时钟的使用

图 8-35 说明了在 PCS 核和 FPGA 桥后段的概念图，以及 PCS 和 FPGA 边界间的主要时钟。

图 8-35. PCS/FPGA 时钟接口概念图



在上图以及本章节随后的时钟图中，请注意后缀的 “i” 表示 [3:0]，即每个表示一个通道。

要求如果通过写入到寄存器位来改变时钟多路复用的任何一个选择器， FPGA 软逻辑应复位使用该多路复用时钟的逻辑电路。

PCS 输出 16 个时钟。有两个发送时钟 （每通道）以及两个接收时钟 （每通道）。两个发送时钟提供全速和半速时钟，都来自于 TX PLL。每个接收通道还有两个时钟 （全速和半速时钟）。所有 16 个时钟可以按需求用作 FPGA 逻辑的本地 （次）或全局 （主）时钟。当 gearing 是 2:1 模式时使用 tx_half_clks。如表 8-5 所示，仅 tx_full_clk_ch0 和 tx_half_clk_ch0 可以直接驱动主时钟布线。其他通道时钟也可以驱动主时钟网络，但是使用通用布线。所有 tx_full_clk_ch[3:0] 和 tx_half_clk_ch[3:0] 信号可以通过使用 USE SECONDARY 时钟参数，驱动次时钟网络。通用布线也用于驱动次时钟网络。

发送时钟用于 TX FIFO 的写端口 （或相移 FIFO，根据不同情况）。两个接收时钟中的一个连接到 RX FIFO 的读时钟。另一个时钟用于 CTC FIFO 的读端口，并且可能可以用于 RX FIFO 的写端口 （根据不同情况）。根据 CTC 和 TX FIFO 是否旁路以及 PCS 是用 8 位 /10 位接口模式还是 16 位 /20 位接口模式，可能有四种使用实例。有效的路径通过加粗线突出显示。它也表明需要多少和什么样的时钟树。有一些更常见的用户首选的模式。

本章节说明了 6 种支持的实例的工作情况。这些实例在表 8-21 中列出。

表 8-21. SERDES/PCS Quad 和 FPGA 内核之间的 6 个接口实例

接口	数据宽度	RX CTC FIFO	RX 相移 / 向下采样 FIFO	TX 相移 / 向上采样 FIFO
Case I-a[2]	8/10 位	是	是	是
Case I-b[2]	8/10 位	旁路	是	是
Case I-c[2]	8/10 位	是	旁路	旁路
Case I-d[2]	8/10 位	旁路	旁路	旁路
Case II-a[1, 2]	16/20 位	是	是	是
Case II-b[1, 2]	16/20 位	旁路	是	是

1. 当使用 16/20 位数据宽度时，通常使用 TX 相移（向上采样）FIFO 和 RX 相移（向下采样）FIFO。它们不能被旁路。不要求 RX 和 TX 同时拥有相同的 FPGA 接口数据路径宽度。可以独立进行控制。为了简洁起见，它们一起在同一个使用实例中进行了说明。
2. TX 相移（向上采样） FIFO 和 RX 相移（向下采样） FIFO 不需要一起被旁路。它们可以独立进行控制。同样为了简洁起见，它们一起在同一个使用实例中进行了说明。

2:1 Gearing

为了确保 FPGA 全局时钟树的性能，推荐当 SERDES 线速率大于 2.5Gbps 时，使用一个 16/20 位宽的接口。在这个接口， FPGA 接口时钟以一半的字节时钟频率运行。

尽管 16/20 位宽的接口可以一半的字节时钟频率在所有的 SERDES 线路上运行，但是当 SREDES 线路速率足够低并满足要求时 （2.5Gbps 及更低）， 8 / 10 位宽接口仍是首选，因为这可在 FPGA 内核中有最高效的 IP 实现。

6 个接口实例的决策矩阵如表 8-22 所示。

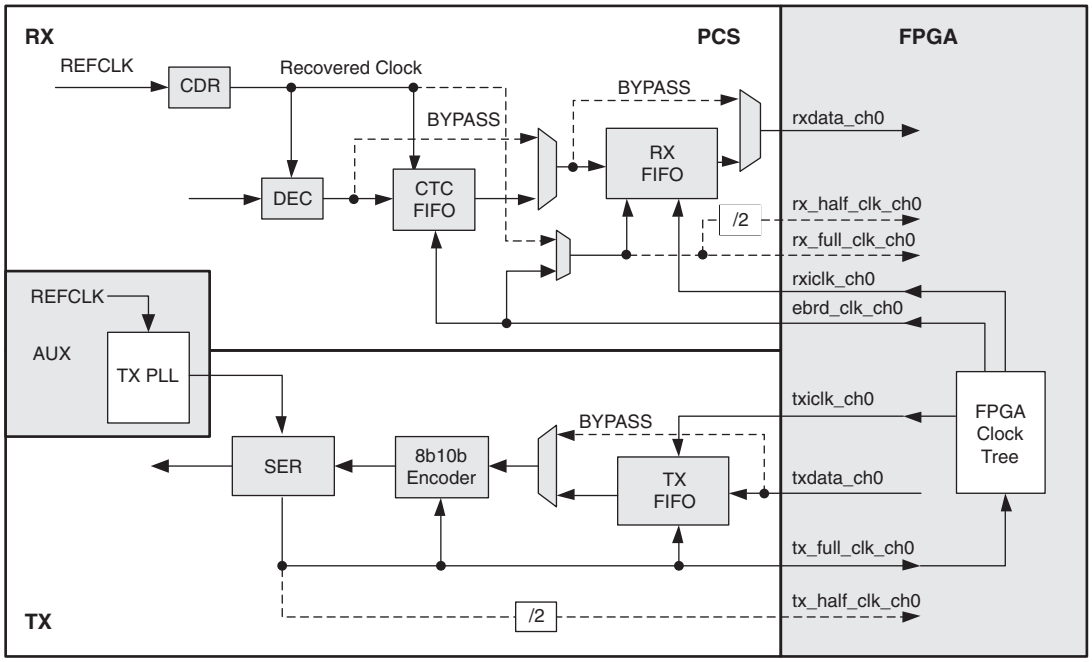
表 8-22. 6 个接口实例的决策矩阵

SERDES线路速率	数据路径宽度	多通道必须对齐？	CTC 是必须的？	RX FIFO 是必须的？	接口实例
小于等于 2.5 Gbps	8/10 位（1:1 gearing）	否，单通道链路	是	是	Case I_a[1]
				否	Case I_c[1]
			否	是	Case I_b[2]
				否	Case I_d[2]
		是，多通道链路	必须旁路，不可用	是	Case I_b[3]
				否	Case I_d[3]
小于等于 3.2 Gbps	16/20 位（2:1 gearing）	否，单通道链路	是	是	Case II_a[4]
			否	是	Case II_b[5]
		是，多通道链路	必须旁路，不可用	是	Case II_b[6]

1. 本实例专用于单通道链接，线路速率为 2.5 Gbps 及更低 （8/10 位宽接口）， quad 中要求时钟容限补偿。当链路两端分别连到各自的参考时钟源时，且两个参考时钟源相差不超过 +/- 300ppm，要求使用 CTC。如果内核中的 IP 要求 RX 相移 FIFO 就使用 Case I_a。如果 IP 不要求这个 FIFO，就使用 Case I_b。
2. 本实例专用于单通道链接，线路速率为 2.5 Gbps 及更低 （8/10 位宽接口）， quad 中**不**要求时钟容限补偿。当链路两端连接到相同的参考时钟源时，不要求使用 CTC。这通常用于同一块电路板上的芯片到芯片的连接。参考时钟间为 0ppm 偏差，因此不要求使用 CTC， CTC 可以被旁路。当 CTC 功能由内核中的 IP 实现时， quad 中也不需要使用 CTC。
3. 本实例专用于多通道链接，线路速率为 2.5 Gbps 及更低 （8/10 位宽接口）。多通道对齐**必须**在 FPGA 设计中实现。由于多通道对齐必须在 CTC 之前实现，当需要多通道对齐时，quad 中的 CTC FIFO 必须被旁路，并且多通道对齐和 CTC（如果需要的话）都由 FPGA 设计实现。
4. 本实例专用于单通道链接，线路速率为 3.2Gbps 及更低，quad 和 FPGA 内核（16/20 位宽接口）之间要求使用一个 2:1 gearbox。quad 中包含时钟容限补偿。当链路两端分别连到各自的参考时钟源时，且两个参考时钟源相差不超过 +/- 300ppm，要求使用 CTC。
5. 本实例专用于单通道链接，线路速率为 3.2Gbps 及更低，quad 和 FPGA 内核（16/20 位宽接口）之间要求使用一个 2:1 gearbox。quad 中**不**包含时钟容限补偿。当链路两端连接到相同的参考时钟源时，不要求使用 CTC。这通常用于同一块电路板上的芯片到芯片的连接。参考时钟间 0ppm 偏差，因此不要求使用 CTC， CTC 可以被旁路。当 CTC 功能由 FPGA 设计实现时， quad 中也不需要使用 CTC。
6. 本实例专用于多通道链接，线路速率为 3.2Gbps 及更低，quad 和 FPGA 内核（16/20 位宽接口）之间要求使用一个 2:1 gearbox。多通道对齐**必须**在 FPGA 设计中实现。由于多通道对齐必须在 CTC 之前实现，当需要多通道对齐时， quad 中的 CTC FIFO 必须被旁路，并且多通道对齐和 CTC （如果需要的话）都由 FPGA 设计实现。

Case I_a：8/10 位， CTC FIFO 和 RX/TX FIFO 未旁路

图 8-36. 8/10 位， CTC FIFO 和 RX/TX FIFO 未旁路



1. TX FIFO 在本实例中仅作为相移 FIFO。
2. RX FIFO 在本实例中仅作为相移 FIFO。
3. 来自 TX PLL（tx_full_clk）的 quad 级全速时钟可以直接到达 FPGA 中央时钟多路复用器。这是相对更高性能的通路。源于中央时钟多路复用器的全局时钟树为 FPGA 中的用户接口逻辑提供时钟。时钟树的叶子节点连接到 FPGA 发送输入时钟 （txiclk），每个通道的 CTC FIFO 读时钟 （ebrd_clk）以及 FPGA 接收输入时钟 （rxiclk）。该实例可能是单通道使用的最常见实例。

FPGA 逻辑中的时钟和数据信号接口实例 （Case I_a）

下面是一部分 SERDES / PCS 在顶层模块的实例化模块，说明了在 Verilog 中时钟和数据端口是如何映射的。

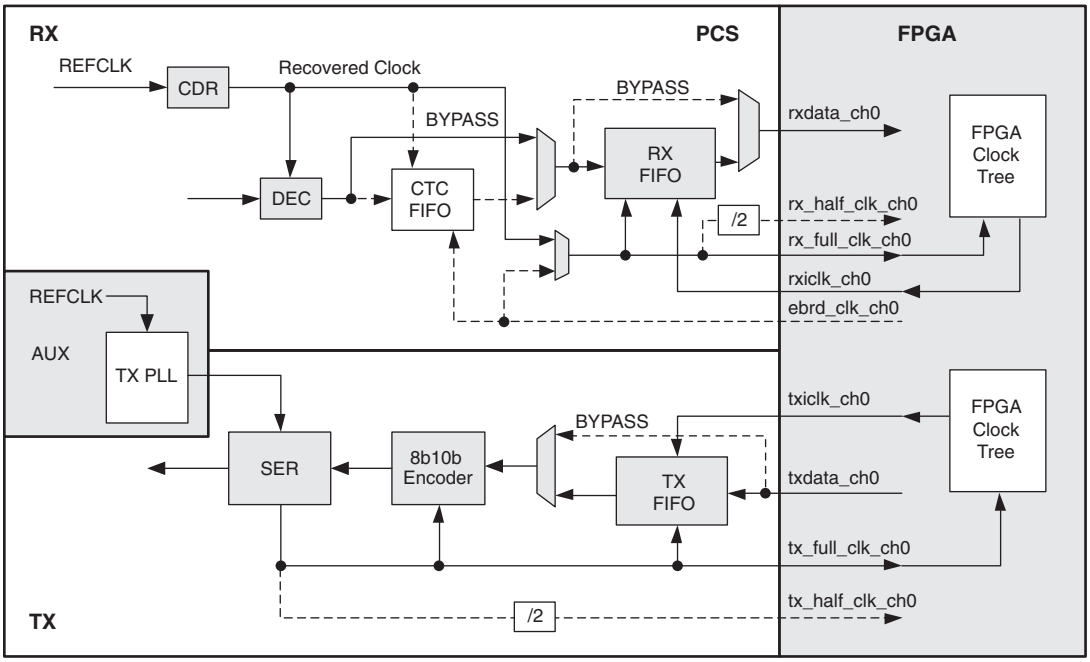
```
.txiclk_ch0(txclk),
.rxiclk_ch0(txclk),
.rx_full_clk_ch0(),
.tx_full_clk_ch0(txclk),
.tx_half_clk_ch0(),
.txdata_ch0(txdata_2_pcs),
.rxdata_ch0(rxdata_from_pcs),
.tx_k_ch0(txkcntl_2_pcs),
.rx_k_ch0(rxkcntl_from_pcs),
```

根据不同情况， ebrd_clk_ch0 由软件自动布线。

请注意 tx_full_clk_ch0 使用线名为 'txclk'，并且同时输出给 txi_clk_ch0 和 rxi_clk_ch0，如图 8-36 所示。

Case I_b：8/10 位， CTC FIFO 被旁路

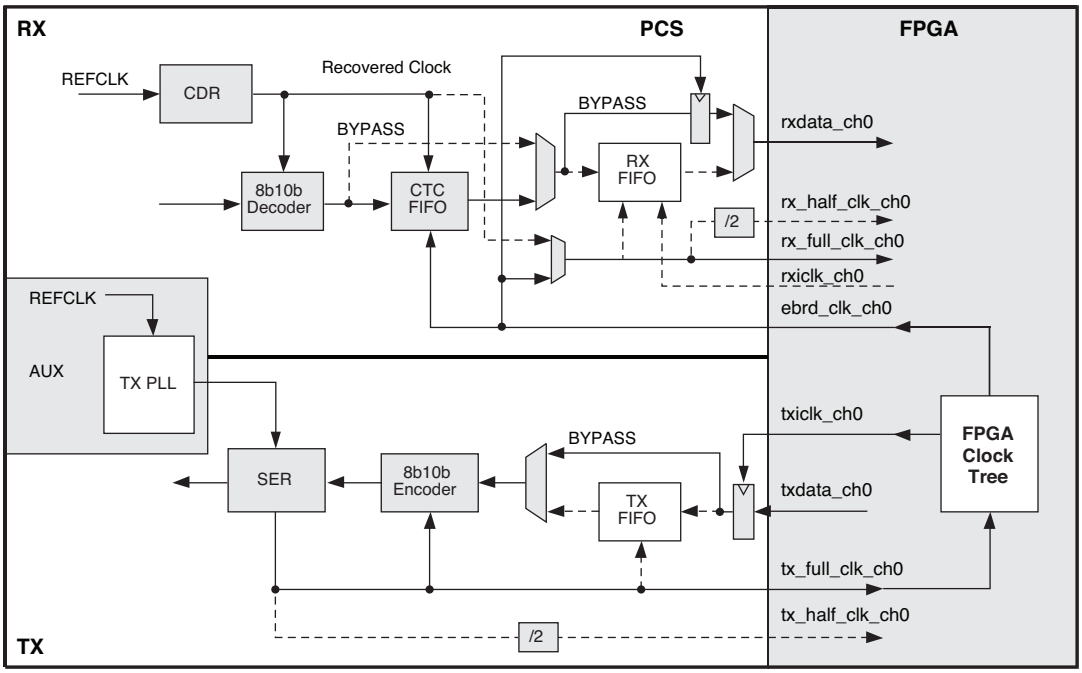
图 8-37. 8b/10 位， CTC FIFO 被旁路



1. TX FIFO 在本实例中仅作为相移 FIFO。
2. RX FIFO 在本实例中仅作为相移 FIFO。
3. TX FPGA 通道输入时钟与之前的实例类似，使用一个时钟树来提供时钟源，时钟树由全速率发送 FPGA 输出时钟直接连接到 FPGA 中央时钟多路开关来驱动。一旦 CTC FIFO 被旁路，恢复时钟需要控制 RX FIFO 的写端口。每个通道的恢复时钟可能需要驱动一个单独的本地或全局时钟树 （即，每个 quad 多达 4 个本地或全局时钟树）。然后时钟树将驱动 FPGA 接收时钟输入来控制 RX FIFO 的读端口。在本实例中将 CTC FIFO 旁路的最大的原因是为了在 FPGA 内核中实现多通道对齐。这也意味着使用弹性缓冲器的 CTC 可以在 FPGA 内核中实现。CTC FIFO 可以通过恢复时钟或主恢复时钟写入。CTC FIFO 的读操作可以通过 TX 时钟树的 TX 时钟实现。

Case I_c：8/10 位， RX/TX FIFO 被旁路

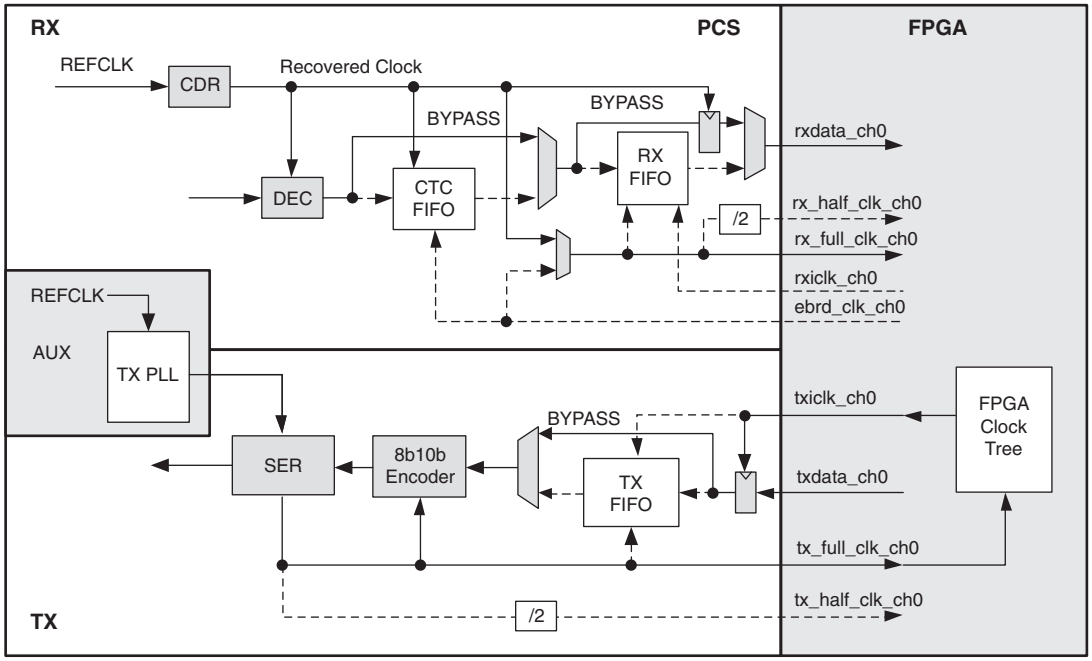
图 8-38. 8/10 位， RX/TX FIFO 被旁路



1. TX 通道时钟类似于之前的两个实例。在 RX 通道，FPGA 的输入时钟现在是 ebrd_clki。FPGA TX 时钟树驱动该时钟。在该实例中， ebrd_clki 由软件自动布线。

Case I_d：8/10 位， CTC FIFO 和 RX/TX FIFO 被旁路

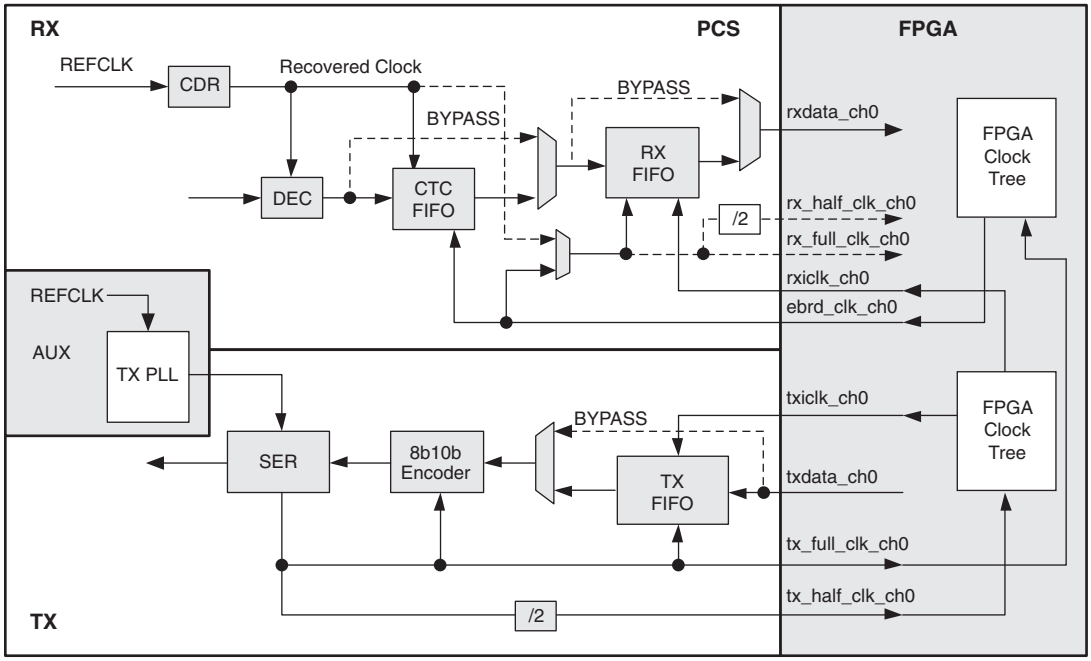
图 8-39. 8b/10 位， CTC FIFO 和 RX/TX FIFO 被旁路



1. 在本实例中， FPGA 的时钟树可以互换看作是时钟域。 TX 通道时钟类似于前面的三个实例。在 RX 通道，恢复通道 RX 时钟发送到 FPGA。本实例对于支持视频应用是十分有帮助的。

Case II_a：16/20 位， CTC FIFO 和 RX/TX FIFO 未旁路

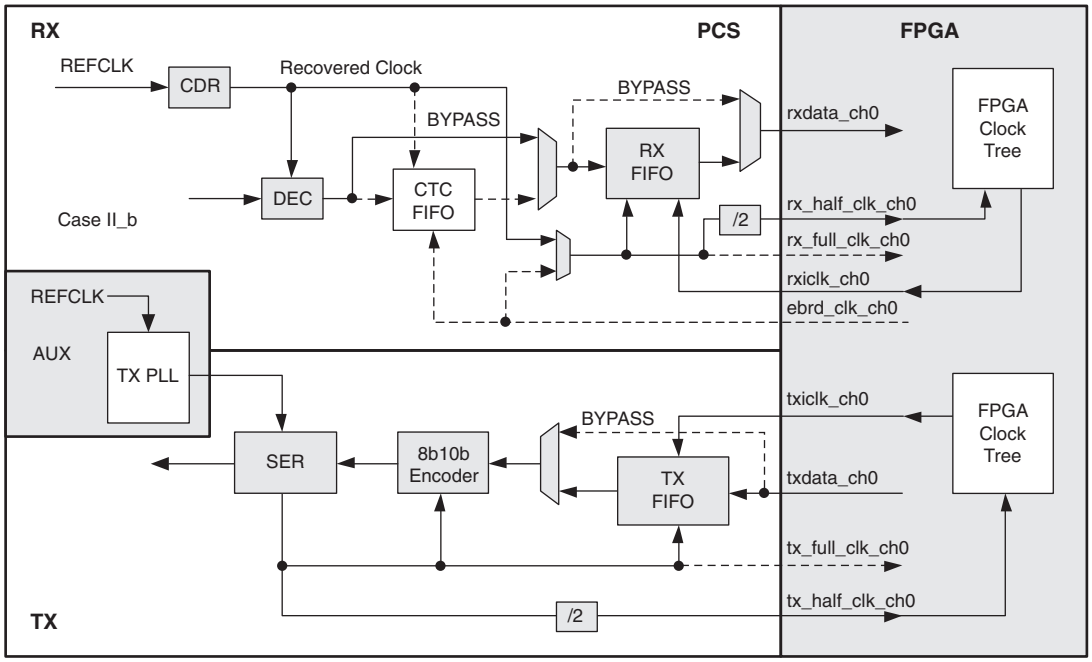
图 8-40. 16/20 位， CTC FIFO 和 RX/TX FIFO 未旁路



1. TX FIFO 在本实例中作为相移 FIFO 和向上采样 FIFO。
2. RX FIFO 在本实例中作为相移 FIFO 和向下采样 FIFO。
3. 这是当 FPGA 无法保持全字节频率时的一种非常常见的单通道使用实例。需要两个时钟树。这些时钟树由发送全速时钟和发送半速时钟直接连到 FPGA 时钟中央多路复用器来驱动。全速时钟树驱动 CTC FIFO 读端口和 RX FIFO 写端口。半速时钟树驱动 RX FIFO 和 FPGA 逻辑。

Case II_b：16/20 位， CTC FIFO 被旁路

图 8-41. 16/20 位， CTC FIFO 被旁路



1. TX FIFO 在本实例中作为相移 FIFO 和向上采样 FIFO。
2. RX FIFO 在本实例中作为相移 FIFO 和向下采样 FIFO。
3. 这是当 FPGA 无法保持全字节频率时的一种非常常见的多通道对齐使用实例。接收时钟树 （多达四个）可以是本地或者全局的。它们以半速时钟工作。发送时钟树由发送半速时钟直接连接到 FPGA 时钟中央多路复用来驱动。

SERDES/PCS 块延迟

表 8-23 说明了发送器和接收器中每个功能块的延迟。延迟以并行时钟周期给出。图 8-42 说明了每个块的位置。

表 8-23. SERDES/PCS 延迟分析

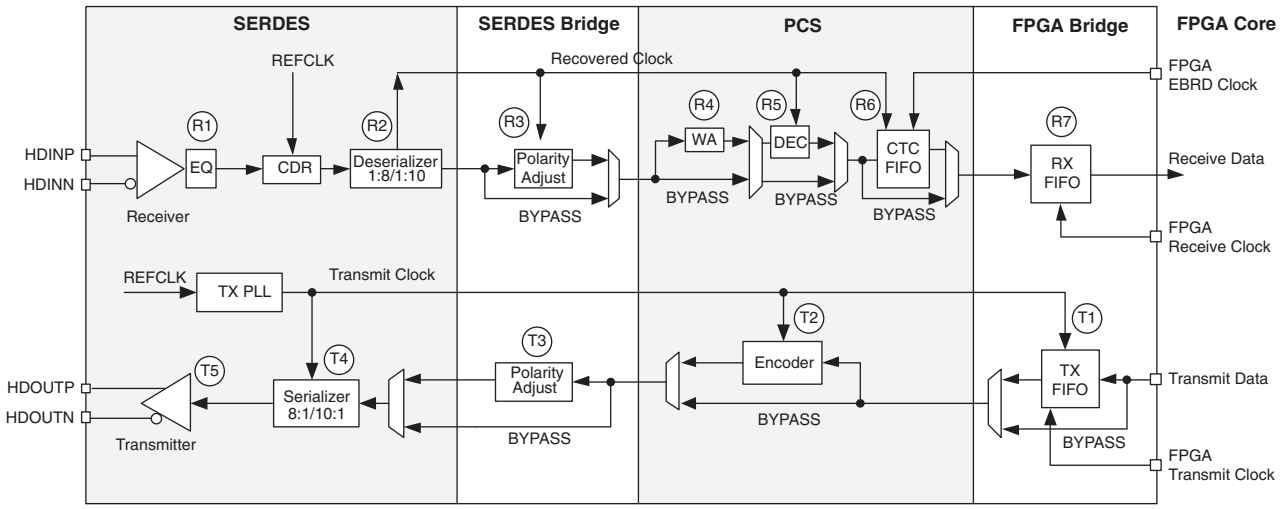
项目	说明	最小值	平均值	最大值	固定值	旁路	单位
发送数据延迟[1]							
T1	FPGA 桥 —— 不同时钟 1:1 Gearing	1	3	5	—	1	word clk
	FPGA 桥 —— 相同时钟 1:1 Gearing	—	—	—	3	1	word clk
	FPGA 桥 ——2:1 Gearing	1	3	5	—	—	word clk
T2	8b10b 编码器	—	—	—	2	1	word clk
T3	SERDES 桥发送	—	—	—	2	1	word clk
T4	串行器：8 位模式	—	—	—	15 + Δ1	—	UI + ps
	串行器：10 位模式	—	—	—	18 + Δ1	—	UI + ps
T5	预加重 ON	—	—	—	1 + Δ2	—	UI + ps
	预加重 OFF	—	—	—	0 + Δ3	—	UI + ps
接收数据延迟[2]							
R1	均衡 ON	—	—	—	Δ1	—	UI + ps
	均衡 OFF	—	—	—	Δ2	—	UI + ps
R2	解串行器：8 位模式	—	—	—	10 + Δ3	—	UI + ps
	解串行器：10 位模式	—	—	—	12 + Δ3	—	UI + ps
R3	SERDES 桥接收	—	—	—	2	1	word clk
R4	字对齐[3]	3.1	—	4	—	—	word clk
R5	8b10b 解码器	—	—	—	1	1	word clk
R6	时钟容限补偿	7	15	23	1	1	word clk
R7	FPGA 桥 —— 不同时钟 1:1 Gearing	1	3	5	—	1	word clk
	FPGA 桥 —— 相同时钟 1:1 Gearing	—	—	—	3	1	word clk
	FPGA 桥 ——2:1 Gearing	1	3	5	—	—	word clk

1. Δ1 = -245ps， Δ2 = +88ps， Δ3 = +112ps。
2. Δ1 = +118ps， Δ2 = +132ps， Δ3 = +700ps。
3. 表 8-24 显示了不同字对齐偏移情况下的字对齐器延迟。确切的偏移值可以从通道状态寄存器 CH_22， bit [3:0] 获得。

表 8-24. 字对齐器延迟 vs. 偏移

wa_offset[3:0] （CH_22[3:0]）	延迟 （字时钟）
0	4.0
1	3.9
2	3.8
3	3.7
4	3.6
5	3.5
6	3.4
7	3.3
8	3.2
9	3.1

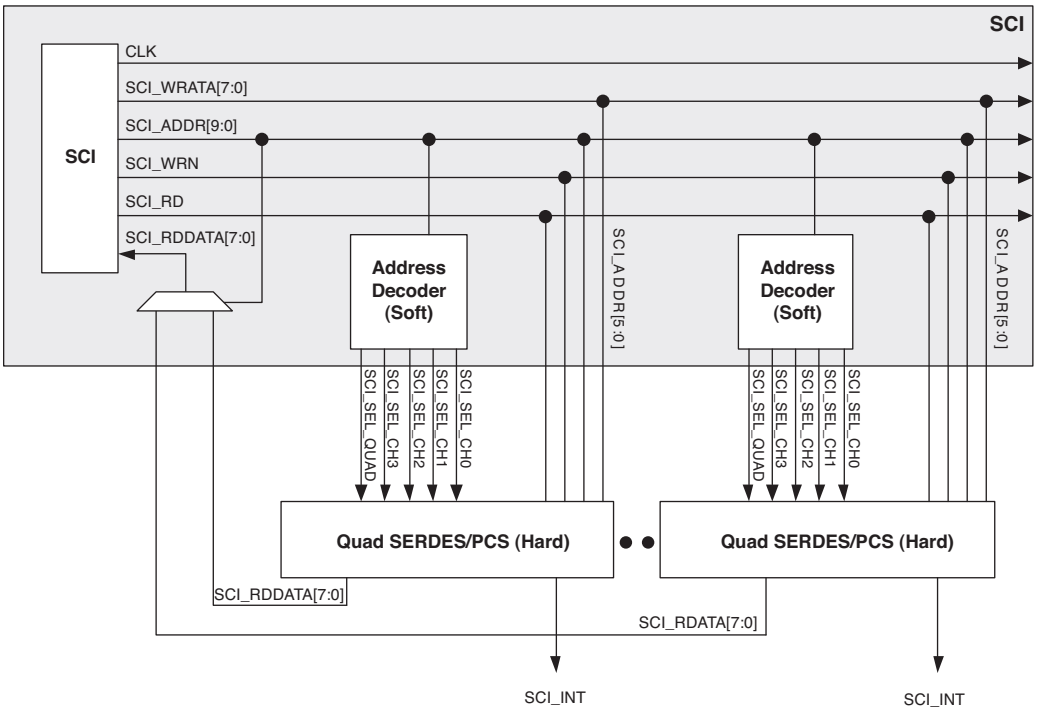
图 8-42. 发送器和接收器延迟框图



SERDES 客户端接口

SCI 允许 SERDES/PCS quad 通过寄存器来控制而不是配置存储器单元。这是一个简单的寄存器配置接口。 FPGA 内核中的 SCI 框图如图 8-43 所示。

图 8-43. SCI 接口框图



FPGA 内核的接口逻辑应该由用户根据其接口机制来开发。请联系莱迪思的技术支持获取示例代码。

该块中的 SCI_ADDR 为 6 位总线宽度。块边界的总线宽度为 11 位。高 5 位用于 quad 块选择和通道选择。表 8-25 显示了 SERDES quad 的 SCI 地址映射。

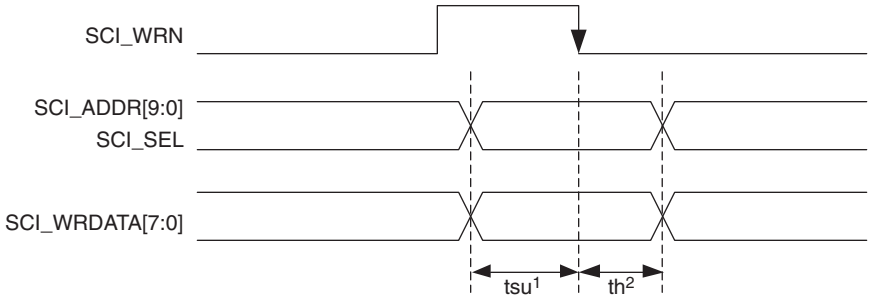
请参见附录 A 和附录 B，了解 SERDES/PCS 寄存器地址和位说明。

表 8-25. 多达 4 个 SERDES/PCS Quad 的 SCI 地址映射

地址位	说明
SCI_ADDR[5:0]	寄存器地址位 000000 = 选择 register 0 000001 = 选择 register 1 ... 111110 = 选择 register 62 111111 = 选择 register 63
SCI_ADDR[8:6]	通道地址位 000 = 选择 channel 0 001 = 选择 channel 1 010 = 选择 channel 2 011 = 选择 channel 3 100 = 选择 Quad 101 = 未用 110 = 未用 111 = 未用
SCI_ADDR[10:9]	Quad 地址位 00 = 选择 Quad A 01 = 选择 Quad B 10 = 选择 Quad C 11 = 选择 Quad D

通过这个接口的读写操作是异步的。在 WRITE 周期内，写数据和写地址必须在 SCI_WR 下降沿建立和保持。在 READ 周期内，时序与 SCI_RD 脉冲有关。图 8-44 和 8-45 分别说明了 WRITE 和 READ 周期。

图 8-44. SCI WRITE 周期，重要时序



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

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

图 8-45. SCI READ 周期，重要时序

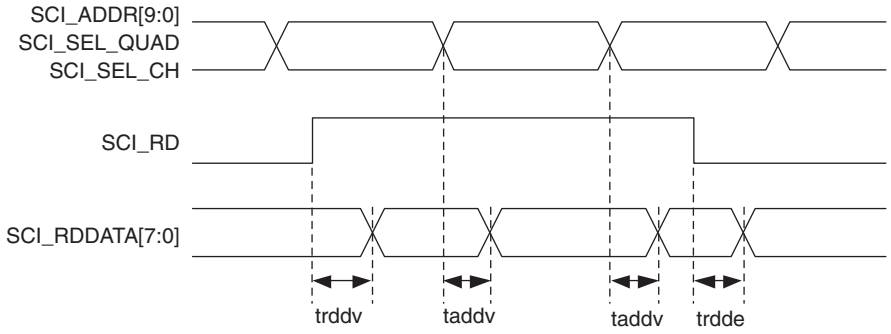


表 8-26. 时序参数

参数	典型值	单位
tsu, trddv, taddv	1.127	ns
th, trdde	0.805	ns

SCI 接口和存储器读 / 写一样简单。下面是伪代码的示例：

Write:

- Cycle 1: Set sci_addr[5:0], sciw_data[7:0], sci_sel = 1’b1
- Cycle 2: Set sci_wrn from 0 ≥ 1
- Cycle 3: Set sci_wrn from 1 ≥ 0, sci_sel = 1’b0

Read:

- Cycle 1: Set sci_addr[5:0], sci_sel = 1’b1
- Cycle 2: Set sci_rd from 0 ≥ 1
- Cycle 3: Obtain reading data from sci_rddata[7:0]
- Cycle 4: Set sci_rd from 1 ≥ 0

中断和状态

状态位可通过 SCI 读取，该位为一个字节宽，因而可以一次读取 8 个中断状态信号的状态。 SCI_INT 信号变为高电平，表示有中断事件发生。然后，用户需要读取 QIF 状态寄存器，判断中断是来自 quad 还是某个通道。这个寄存器**不是**读后清除的。它会在所有 quad 或通道的中断源清除后被清除。一旦中断源总数确定，用户可以读相关的 quad 或通道中的寄存器来确定实际的中断源。表 8-27 和 8-28 列出了所有的中断源。

表 8-27. Quad 中断源

Quad SCI_INT 源	说明	寄存器名称
int_qd_out	Quad 中断。如果在 quad 中有任何中断事件，该寄存器位将置为有效。该寄存器位会在所有中断事件清除后被清除。	PCS Quad 状态寄存器 QD_20
int_ch_out[0:3]	Channel 中断。如果在各相应通道中有任何中断事件，该寄存器位将置为有效。这些寄存器位会在各个相应通道中的所有中断事件清除后被清除。	PCS Quad 状态寄存器 QD_20
ls_sync_statusn_[0:3]_int ls_sync_status_[0:3]_int	Link Status Low （未同步）通道中断 Link Status High （同步）通道中断	PCS Quad 中断状态寄存器 QD_22
~PLOL, PLOL	~PLOL 和 PLOL 上产生中断 ——PLL 失锁	SERDES Quad 状态寄存器 QD_25

表 8-28. 通道中断源

通道 SCI_INT 源	说明	寄存器名称
fb_tx_fifo_error_int fb_rx_fifo_error_int cc_overrun_int cc_underrun_int	FPGA 桥 TX FIFO 错误中断 FPGA 桥 RX FIFO 错误中断 CTC FIFO 过载和欠载中断	PCS 通道通用中断状态寄存器 CH_23
pci_det_done_int rlos_lo_int ~rlos_lo_int rlol_int ~rlol_int	pci_det_done 产生中断 rlos_lo 产生中断 ~rlos_lo 产生中断 rlol 产生中断 ~rlol 产生中断	SERDES 通道中断状态寄存器 CH_2A

SERDES 客户端接口应用示例

莱迪思半导体的 ORCAstra FPGA 配置软件是一款基于 PC 的图形用户界面，允许用户通过对 FPGA 寄存器的控制位编程，来配置莱迪思 FPGA 的运行模式。

SERDES/PCS 状态信息在屏幕上实时显示，并且任何配置可以保存到控制寄存器中，以用于其他的测试。图形用户界面的使用不会影响 FPGA 内核部分的编程。更多有关 ORCAstra 的信息和可下载文件可从莱迪思半导体网站 www.latticesemi.com/products/designsoftware/orcastra.cfm 获取。

用户可以从 IPexpress 获取完整的 LatticeECP3 ORCAstra 接口设计。除了包含 ORCAstra 接口的 HDL 文件，还提供一个包含 ORCAstra 接口的文件，文件名为 "chip.v" （用于 Verilog）。

SERDES/PCS Quad 的动态配置

SERDES/PCS quad 可通过寄存器控制，寄存器可由可选的 SERDES 客户端接口进行访问。

当通过配置存储器单元控制时，配置完成后，要求 SERDES/PCS quad 必须达到工作状态，不需要用户进一步干预。这意味着任何初始化 SERDES/PCS quad 所需的特定复位序列必须通过硬件自动处理。换言之，SCI 的使用是可选的，SERDES/PCS quad 不假定在 FPGA 内核中使用软 IP。

SERDES 调试功能

PCS 回环模式

LatticeECP3 系列提供三种回环模式，通过 PCS/FPGA 接口的控制信号控制，便于测试外部 SERDES/ 板上接口和内部 PCS/FPGA 逻辑接口。提供了两种回环模式，将接收到的数据传回发送数据路径。回环模式对于检测高速串行 SERDES 封装引脚连接以及嵌入式 SERDES 和 / 或 PCS 逻辑是十分有用的。

RX 至 TX 串行回环模式

将接收到的串行数据传回发送缓冲器，无需经过 CDR 或解串行器。在 IPexpress 图形用户界面中选择 RX-to-TX Serial Loopback 选项会使得 LB_CTL[1:0] 设为 '10' 并将 TDRV_DAT_SEL[1:0] 寄存器设为 '11'（请参见表 8-81 和 8-84）。

TX 至 RX 串行回环模式

此模式将发送的串行数据传回接收器 CDR 块。在 IPexpress 图形用户界面中选择 TX-to-RX Serial Loopback 选项会将 LB_CTL[1:0] 设为 '01'（参见表 8-81）。

SERDES 并行回环模式

将接收到的并行数据传回发送数据路径，无需经过 PCS 逻辑。在 IPexpress 图形用户界面中禁止时，并行回环模式可以通过 FPGA 内核控制信号 sb_felb_ch[3:0]_c 和 sb_felb_rst_ch[3:0]_c 进行动态控制。 SERDES 并行回环可以通过分别将每个通道的相应 sb_felb_ch[3:0]_c 位设为 “1” 来进行选择。

如果没有使用该回环模式的动态功能，两个控制信号应连接到地。

当在 IPexpress 图形用户界面中使能回环模式时，控制寄存器位 sb_pfifo_lp(CH_03[5]) 置 1，并且回环模式置 1。FPGA 内核中的控制信号 sb_felb_ch[3:0]_c 和 sb_felb_rst_ch[3:0]_c 在 PCS 模块中不可用。

上述讨论请参见图 8-46。

图 8-46. 三种回环模式

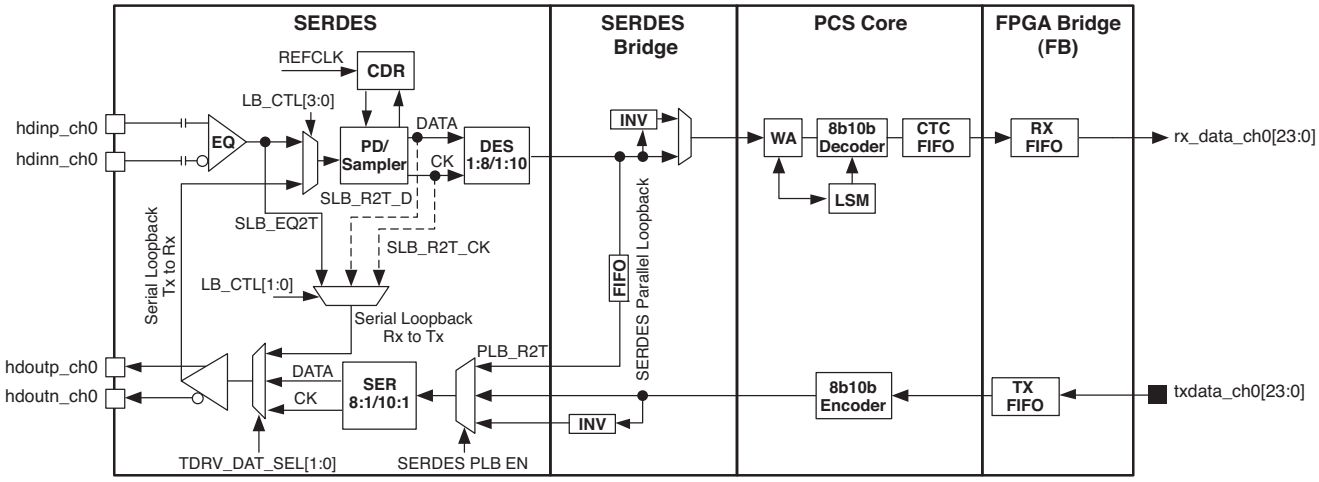
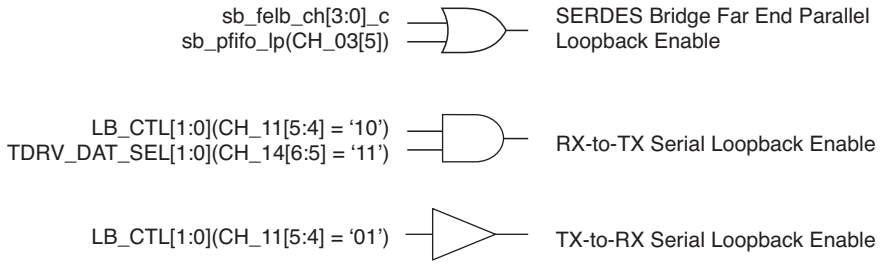


图 8-47. 回环使能信号



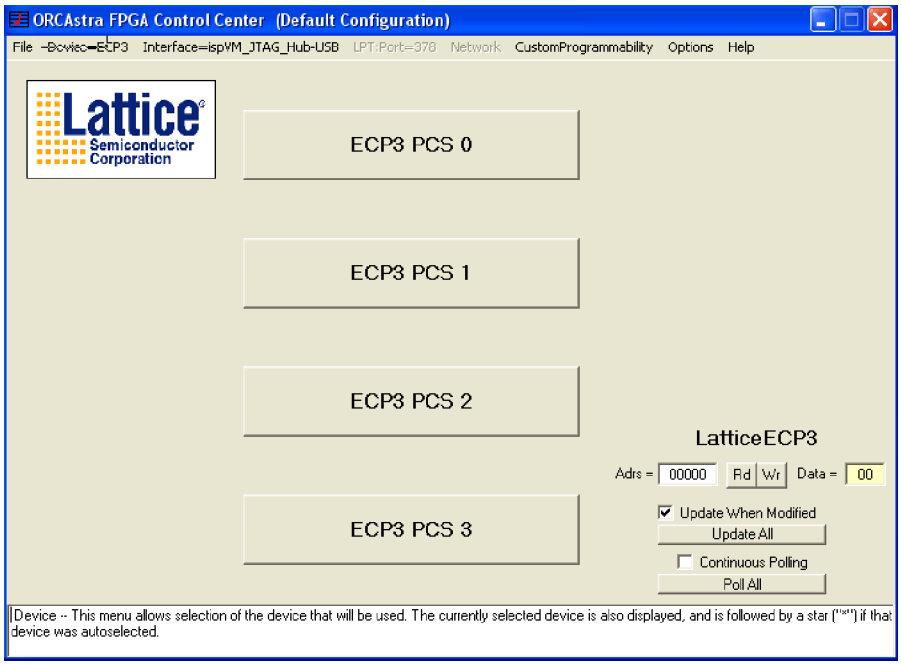
ORCAstra

莱迪思 ORCAstra 软件能帮助您快速探索配置选项，而无需冗长的重新编译过程或者对您的电路板做任何更改。图形用户界面中创建的配置可以保存到存储器中并在以后使用时重新载入。为了使用 ORCAstra，ORCASTRA 模块必须在 IPexpress 中创建，并在 FPGA 设计中使用。

还可以使用宏功能来支持基于脚本的配置和测试。图形用户界面还可以用于实时显示系统状态信息。使用ORCAstra软件不会干扰 FPGA 的编程。

图 8-48 显示了 ORCAstra 图形用户界面的顶层窗口。用户可以在此窗口中读写地址单元的数据而无需进入每个 PCS 通道的子窗口。当调用时， ORCAstra 将自动识别器件类型。或者，可以通过器件下拉菜单选择器件类型。

图 8-48. ORCAstra 顶层屏幕快照



默认情况下，图 8-48 中的数据盒使用的是大端字节序（即，最重要的位在左边）。用户可以通过 **Options** 选项卡下的 **Display Data Reversed in Data Box** 更改为小端字节序。

点击选项卡 **Interface=None** 并在下拉列表中选择 **1 ispVM JTAG Hub USB Interface**。

然后从 **Select Target JTAG Device** 窗口中选择 **C2 0A 80 80**。

图 8-49. JTAG 器件选择

然后点击 ORCAstra Hub I/O 窗口中的 **OK**。

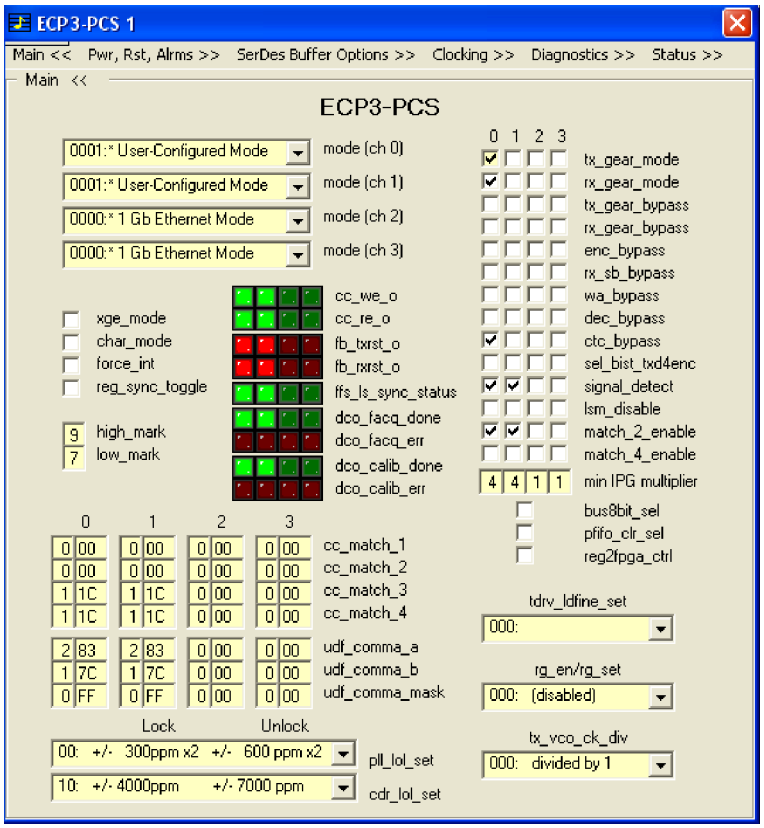
图 8-50. Hub ID 选择

图中显示了在 channel 0 和 channel 1 中有数据通信。在此示例中，我们假设 PCS SCI 地址映射到设计中的 Quad 0。

双击 **PCS0** （Quad 0）按钮将打开主窗口，如图 8-51 所示。

这些标准的 Windows 菜单控制了器件和接口的选择。它们还支持各种配置选项，包括使用储存的文件设置和保存配置。

图 8-51. ORCAstra 主窗口

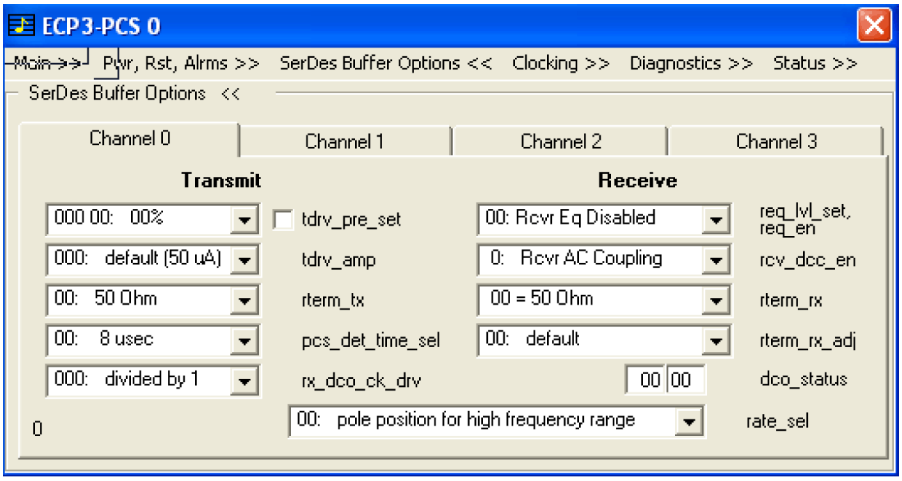


控制盒和按钮、状态盒和文本窗口

将鼠标光标移动到一个控制盒并点击鼠标左键来设置控制位。所选控制盒的位的位置和功能都显示在文本窗口中，并与 LatticeECP3 系列数据手册中的寄存器映射表对应。当光标移动到位的名称上时，仅显示其功能。状态盒与控制盒类似，不过拥有 LED 显示和彩色背景。

图 8-52 显示了 SERDES 缓冲器选项窗口。可以从下拉菜单中选择配置选项。

图 8-52. SERDES 缓冲器选项窗口



更多有关 ORCAstra 的信息和可下载文件可以从下面莱迪思半导体网站获得: www.latticesemi.com/products/design-software/orcastra.cfm.

其他设计考虑

SERDES/PCS 的仿真

表 8-29. 仿真模型位置

仿真器	模型地址
Active-HDL	ispTOOLS\cae_library\simulation\blackbox\pcsc-aldec.zip
ModelSim	ispTOOLS\cae_library\simulation\blackbox\pcsd-mti_6.0-V1-1.zip
NC-Verilog	ispTOOLS\cae_library\simulation\blackbox\pcsd-ncv.zip
VCS	ispTOOLS\cae_library\simulation\blackbox\PCSD_sim.vp.zip

16/20 位字对齐

PCS 接收器不能识别 16 位字边界。当使能字对齐器时， PCS 仅能实现 BYTE 对齐。 16 位字对齐应该在 FPGA 结构中实现并且这是一种非常直接的方法。仿真模型也使用相同的工作方式。如果用户使用下面所述的对齐机制，其功能可以得到增强。

例如，如果 FPGA 接口的发送数据为：

```
YZABCDEFGHIJKLM... （每个字母一个字节， 8 位或 10 位）
```

然后 8b10b 解码器后以及 rx_gearbox 前 PCS 中的输入数据为：

```
YZABCDEFGHIJKLM...
```

在 rx_gearbox 后，数据变为：

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

或

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

明显可以看出，序列 2 没有对齐。有一个字节的偏移，但是需要 16/20 位对齐。在这里我们假定特定字符 'A' 应该一直放在低字节。

翻转一个 20 位数据结合当前的 16/20 位数据形成如下所示的 32/40 位数据：

1. {DCBA}{HGFE}{LKJI}...
   ^
   | **Found the A in lower 10-bit, set the offset to '0', send out aligned data 'BA'

下一个时钟周期：

{FEDC}{JIHG}{NMLK}...
^
| ** 发出对齐的数据 'DC'

等。

16/20 位对齐后，输出数据为：

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

2. {CBAZ}{GFED}{KJIH}....
^
| ** 在前 10 位找到 A，设置偏移为 ‘10’，发出对齐的数据 ‘BA’

下一个时钟周期：

{EDCB}{IHGF}{MLKJ}...
^
| ** 发出对齐的数据 ‘DC’

等。

20 位对齐后，输出数据为：

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

注：通常先发送和接收一个 8/10 位字节或一个 16/20 位字的 LSB。

欲获取 16/20 位字对齐代码示例，请将您的申请发送到 techsupport@latticesemi.com。

未使用的 Quad/ 通道和电源

在未使用的 quad 和通道上，VCCA 应当上电。VCCIB、VCCOB、HDINP/N、HDOUTP/N 和 REFCLKP/N 应保持悬空。未使用的通道输出应为三态，使用大约 10 KOhm 内部电阻连接差分输出对。在配置过程中， HDOUTP/N 拉高至 VCCOB。

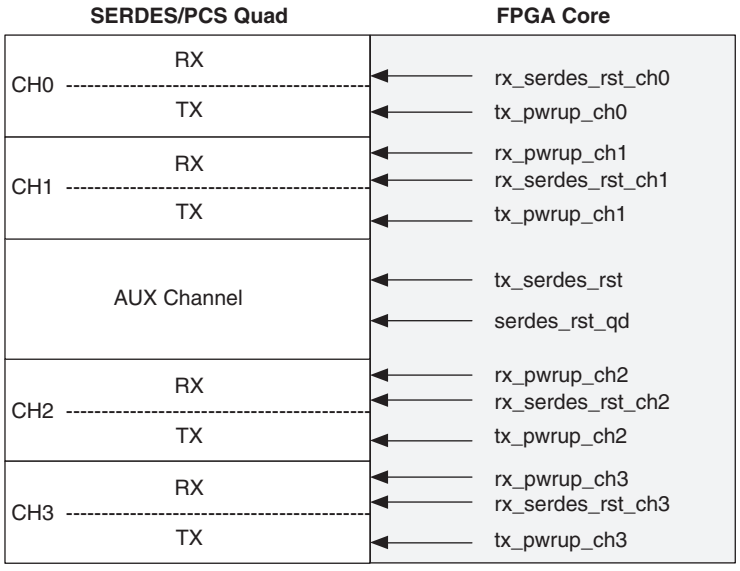
即使当通道用作 Rx Only 模式或 Tx Only 模式，通道的 VCCOB 和 VCCIB 都必须上电。未使用的 SERDES 默认配置为掉电模式。

复位和掉电控制

SERDES quad 对所有通道的每个发送器和接收器都有复位和掉电控制，如图 8-53 所示。复位信号高电平有效，通过将 pwrup 信号驱动为低电平可实现掉电。各种复位和掉电控制操作将在下面章节说明。

注：当器件上电并且芯片级上电复位有效， SERDES 控制位 （在 PCS 中）将清零 （或将变为其默认值）。这将使 SERDES quad 进入掉电状态。

图 8-53. SERDES/PCS Quad 复位和掉电控制



通常，所有复位是通过上电复位和各种 FPGA 结构复位。复位逻辑如图 8-54 和表 8-30 所示。

图 8-54. SERDES/PCS 复位图

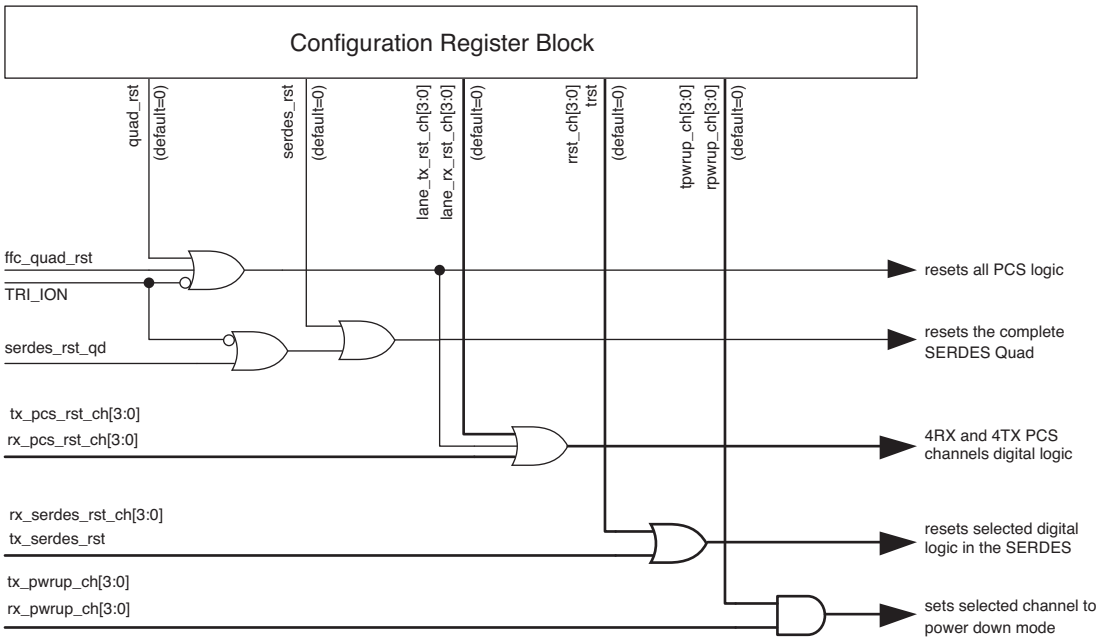


表 8-30. SERDES/PCS 复位表

复位信号		PCS[1] TX	PCS[1] RX	SERDES TX	SERDES RX	PCS CTRL 寄存器	TX PLL	CDR PLL
FPGA	控制寄存器							
tx_pcs_rst_ch[3:0]_c	lane_tx_rst[3:0]	X						
rx_pcs_rst_ch[3:0]_c	lane_rx_rst[3:0]		X					
rst_qd_c	quad_rst	X	X	X	X		X	X
serdes_rst_qd_c	serdes_rst			X	X		X	X
rx_serdes_rst_ch[3:0]_c	rrst[3:0][2]				X			X
tx_serdes_rst_c	trst						X[3]	
TRI_ION （配置）		X	X	X	X	X		

1. 包括 SB （SERDES 桥）、 PCS 内核以及 FB （FPGA 桥）子模块。
2. 仅内部使用。该复位通常应连到 '0'，除非需要复位 CDR PLL。
3. tx_serdes_rst_c 复位不会复位 TX PLL。该信号仅会强制 tx_pll_lol_qd_s 变为高电平。

表 8-31. 复位控制说明[1, 2, 3]

复位信号		说明
FPGA	控制寄存器	
rst_qd_c	quad_rst	高电平有效，异步输入。复位所有的 SERDES 通道，包括辅助通道和 PCS。该复位包括 serdes_rst、 txpll、 cdr、 lane_tx_rst 和 lane_rx_rst。
serdes_rst_qd_c	serdes_rst	高电平有效，异步输入到 SERDES quad。将软件寄存器位用作门控信号。该复位仅用于 SERDES 块，包括 TXPLL 和 CDRPLL。
tx/rx_pcs_rst_ch[3:0]_c	lane_tx/rx_rst[0:3]	高电平有效，异步输入。复位 SB、 PCS 内核和 FB 块中的单个 TX/RX 通道。
rx_serdes_rst_ch[3:0]_c	rrst[0:3]	复位失锁 （rlol），信号丢失和校准电路。
tx_serdes_rst_c	trst	复位 AUX PLL （plol）失锁。

1. 对于 quad 中所有以全数据速率模式运行的通道，并行端时钟确保同相。
2. 对于 quad 中所有以半数据速率模式运行的通道，每个通道都有一个独立的除 2 电路。因为在 “serdes_rst” 信号释放后，quad 中没有机制来确保这些除 2 电路同相， PCS 设计应当假设除法器 （及并行端时钟）**不是**同相的。
3. 在半数据速率模式下，因为无法确保并行端时钟同相，这可能会增加多通道链路的发送和接收端的通道到通道的偏移。

表 8-32. 复位脉冲规范

参数	说明	最小值	典型值	最大值	单位
$t_{SERDES\_RST\_QD}$	Quad SERDES 复位高电平时间	1			us
$t_{RX\_PCS\_RST}$	通道 RX PCS 复位高电平时间	3			ns
$t_{TX\_PCS\_RST}$	通道 TX PCS 复位高电平时间	3			ns
$t_{RX\_SERDES\_RST}$	通道 RX SERDES 复位高电平时间	3			ns
$t_{TX\_SERDES\_RST}$	Quad TX SERDES 复位高电平时间	3			ns

掉电控制说明

每个 RX 和 TX 通道可以单独通过软件寄存器位或来自 FPGA 的控制信号进行掉电控制。每个通道的掉电控制位仅对 SERDES 宏单元中选定的块和高速 I/O 缓冲器进行掉电控制。

表 8-33. 掉电控制说明

信号		说明
FPGA	寄存器	
serdes_pd		低电平有效，异步输入到 SERDES quad，作用在所有通道，包括辅助通道。当驱动为低电平时，它使整个宏断电，包括发送 PLL。所有的时钟都停止工作，因此整个宏的功率耗散降至最低。该信号释放后，需要紧跟 TX 和 RX 复位序列。
tx_pwrup_ch[0:3]_c	tpwrup[0:3]	高电平有效，发送通道上电 —— 串行器和输出驱动器上电。该信号释放后，需要紧跟 TX 复位序列。该信号释放后，需要紧跟 TX 复位序列。
rx_pwrup_ch[0:3]_c	rpwrup[0:3]	高电平有效，接收通道上电 ——CDR、输入缓冲器 （均衡器和放大器）以及信号丢失检测器上电。该信号释放后，需要紧跟 RX 复位序列。

表 8-34. 掉电 / 上电时序规范

参数	说明	最小值	典型值	最大值	单位
t_{PWRDN}	serdes_pd 后的掉电时间	20			ns
t_{PWRUP}	serdes_pd 后的上电时间	20			ns

SERDES/PCS 复位

复位序列和复位状态图

上电和配置后，进行所有 SERDES 复位和 FPGA 复位。

复位序列产生

IPexpress 图形用户界面中包括复位序列 （Diamond 1.1 及以上版本提供）。

我们推荐在 IPexpress 中选择复位序列产生选项，如 Control Setup 选项卡章节所述。为 SERDES/PCS 生成的 HDL 文件将包括 Tx 复位状态机和 Rx 复位状态机。

锁状态信号定义

tx_pll_lol_qd_s:	: 1 = TX PLL 失锁 : 0 = TX PLL 锁定 需要 1,400,000 UI 来申明 TX PLL 锁定
rx_cdr_lol_ch[3:0]_s	: 1 = CDR 失锁 : 0 = 保持锁定 需要 400,000 个参考时钟周期 （最差情况下）来申明 CDR PLL 锁定
rx_los_low_ch[3:0]_s	: 1 = 每个通道的信号丢失检测 : 0 = 检测到信号

rx_cdr_lol_ch[3:0]_s 状态信号是 CDR 锁定状态指示符，如上定义。然而，在 CDR 锁定过程中，当没有输入数据出现时， CDR PLL 将锁定到一个参考时钟。这避免了在其恢复时，输入数据被忽略。

为了确保在 CDR 锁定状态检测时有输入数据出现，推荐结合使用 rx_los_low_ch[3:0]_s 信号和 rx_cdr_lol_ch[3:0]_s 信号。

TX 复位序列

1. QUAD_RESET: 上电时，使 rst_qd_c 和 tx_pcs_rst_ch[3:0]_c 有效。
2. WAIT_FOR_TIMER1: 启动 TIMER1。等待至少 20 ns。
3. CHECK_PLOL: 释放 rst_qd_c。
4. WAIT_FOR_TIMER2: 启动 TIMER2。如果 TIMER2 计时满并且 TX PLL 没有被锁定，转到第 1 步。
5. NORMAL: 释放 tx_pcs_rst_ch#_c。正常工作情况下，如果 tx_pll_lol_qd_s 变为高电平，转到第 1 步。

RX 复位序列

1. WAIT_FOR_PLOL: 等待直到 TX PLL 锁定并且接收数据出现。rx_serdes_rst_ch[3:0]_c 设为 0，因为当它有效时， rx_los_low[3:0]_s 变为高电平。
2. RX_SERDES_RESET: 使 rx_serdes_rst_ch[3:0]_c 和 rx_pcs_rst_ch[3:0]_c 有效。
3. WAIT_FOR_TIMER1: 等待至少 3 ns。
4. CHECK_LOL_LOS: 释放 rx_serdes_rst_ch_c。复位 TIMER2。

5. WAIT_FOR_TIMER2: 等待 cdr_lol_ch[3:0]_s 和 rx_los_low_ch[3:0] 变为低电平。如果 rx_lol_los (rx_cdr_lol_ch_s || rx_los_low_ch_s)信号跳变，转到第4步。如果TIMER2计时满并且 rx_lol_los = 1，转到第 1 步。

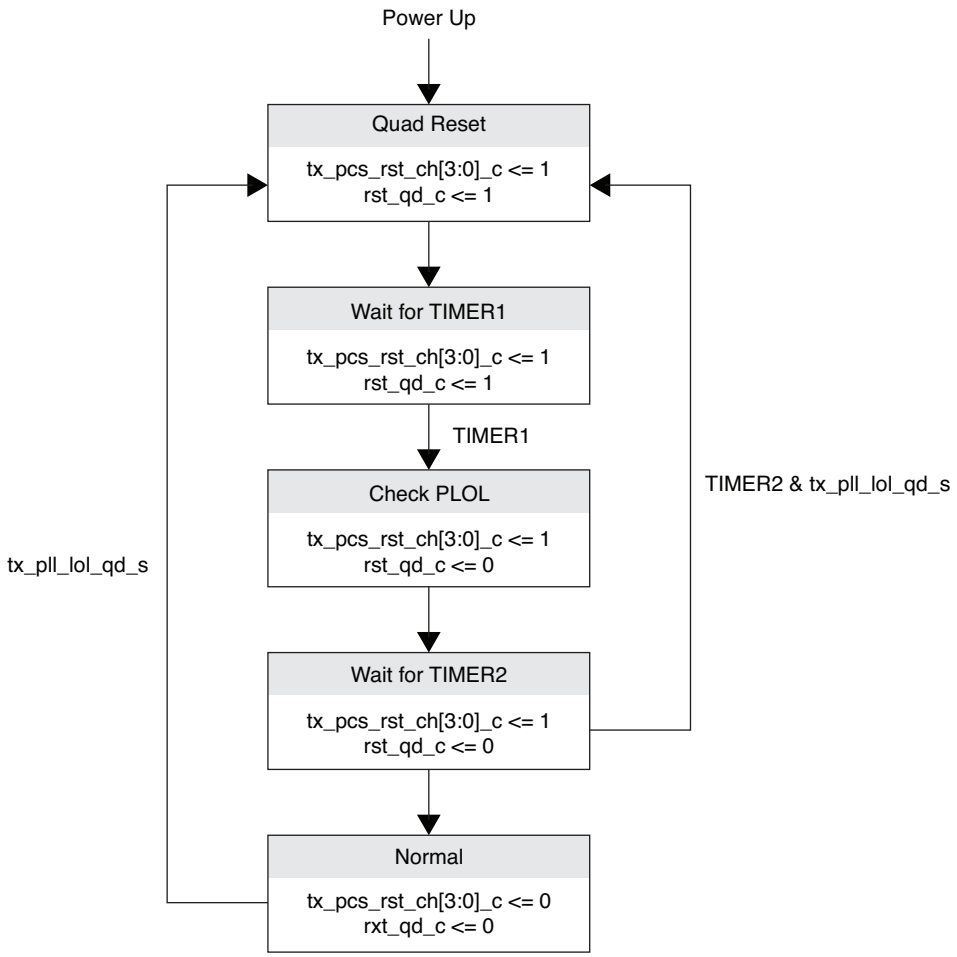
6. NORMAL: 释放 rx_pcs_rst_ch_c。如果 rx_lol_los 变为高电平，转到第 1 步。

注: 当输入数据源在正常工作状态下中断，RX复位序列提供了CDR重锁定特性。RX复位可按每个通道来设置。

复位序列状态图如图 8-55 和 8-56 所示。

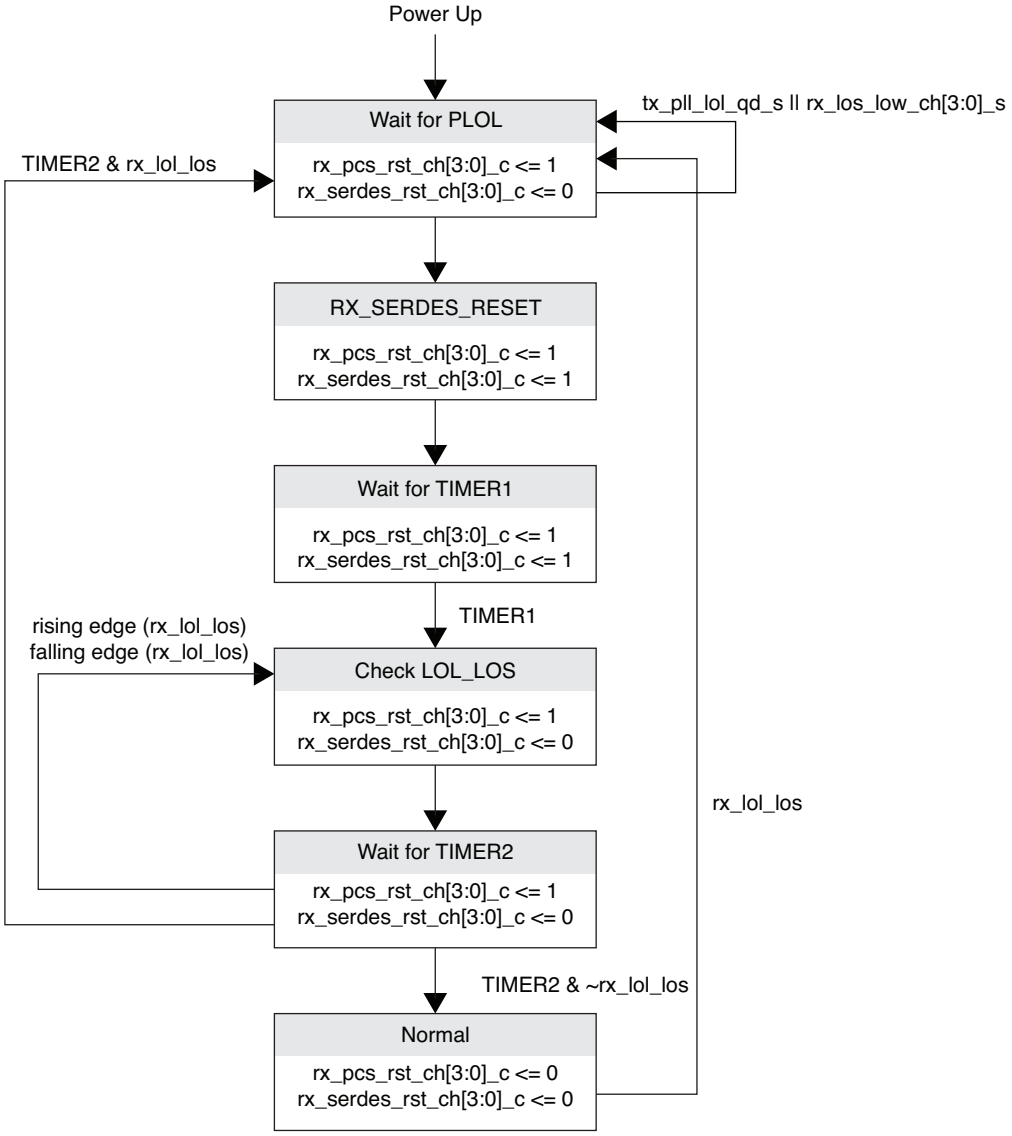
可从莱迪思网站查阅一组复位序列代码示例。

图 8-55. TX 复位状态图



Notes:
TIMER 1: rst_qd_c asserted for a minimum of 20 ns.
TIMER 2: Time to declare TX PLL lock: 1,400,000 UI.

图 8-56. RX 复位状态图



Notes:
TIMER 1: rx_serdes_rst_ch[3:0]_c asserted for minimum 3 ns.

TIMER 2: Time for rx_lol_los signal to stay low (400,000 reference clock cycles). Any FPGA clock can be used to satisfy the timer requirement.

In the diagram above, rx_lol_los is defined as rx_cdr_lol_ch[3:0]_s || rx_los_low_ch[3:0]_s.

The tx_pll_lol_qd_s input to the state diagram RTL code should be tied low in Rx Only mode or when the recovered clock is used as the Tx PLL reference clock, as in SDI applications.

When multiple receiver channels rx_serdes_rst_ch[3:0]_c are to be asserted, it is recommended to activate the reset signals one channel at a time. Simultaneous resetting of multiple receiver SERDES channels may cause a current surge in the SERDES/PCS quad.

The rx_los_low output from SERDES may be triggered for some input streams with continuous zeros, like the SDI pathological pattern. For such applications, the rx_los_low input to the reset state machine must be connected to the carrier detect output (must be inverted) of the cable equalizer if available. In general, CD=1 means carrier is present. So this signal must be inverted to replace rx_los_low. If a cable equalizer is not available, users may tie it to zero but in this case, the CDR can lock to a local reference clock when there is no input data present.

电源供电顺序要求

当使用 SERDES 1.5V VCCIB 或 VCCOB 时， 1.5V 电源供电但 1.2V 电源未供电时， SERDES 不应置于稳定的状态。 1.2V 和 1.5V 电源应同时加到 SERDES 上。对于在电源和稳压器的 ramp_up 时间的正常变化，不需要考虑。

参考文档

- TN1033, 高速印刷电路板的设计考虑
- TN1114, 莱迪思 SERDES 的电气建议
- HB1009, LatticeECP3 系列手册
- DS1021, LatticeECP3 系列数据手册

技术支持

热线电话：+86-21-52989090
电子邮件：techsupport-asia@latticesemi.com
网站： www.latticesemi.com.cn

版本历史

日期	版本	更改摘要
2009 年 2 月	01.0	初始版本。
2009 年 3 月	01.1	针对每种模式表，更新了数据总线的使用。
		更新了参考时钟使用框图。
2009 年 6 月	01.2	tdrv_amp 属性设置改为 tdrv 属性。
		添加了外部链接状态机选项章节。
		在千兆以太网模式章节添加了 Idle 插入。
		更新了图形用户界面截屏。
		更新了字对齐延迟。
2009 年 11 月	01.3	分别给出了 tx 和 rx 端更新的复位序列状态图。
		更新了 los 设置。
		添加了 SONET 支持。
		删除了远端并行回环特性。
		重新定义了数据速率范围。
		最小数据速率降低为 150 Mbps。
2010 年 2 月	01.4	更新了复位序列状态图。
2010 年 3 月	01.5	在 IPexpress 图形用户界面中添加了生成选项卡。
2010 年 6 月	01.6	更新了 Lattice Diamond 设计软件支持。
		时钟图中去除了 refclk_to/from_nq 信号。
2010 年 12 月	01.7	在 IPexpress 图形用户界面中添加了复位序列产生选项。
		添加了有关逗号掩码的详细信息。
2011 年 7 月	01.8	SERDES 支持的标准表 —— 更新了千兆以太网、 SGMII、 10 位 SERDES、 8 位 SERDES 和通用 8b10b 的信息。
		更新了发送数据总线章节。
		更新了接收数据总线章节。
		更新了信号丢失检测器图的注脚。
		SERDES_PCS 图形用户界面属性 ——Quad 选项卡设置 —— 协议增加了 SGMII。
		SERDES/PCS 延迟分析表 —— 更新了 gearing 信息。

日期	版本	更改摘要
		仿真模型位置表 —— 添加了 VCS 行。
		更新了 SERDES/PCS 复位图。
		更新了 Rx 复位状态图的注脚。
2011 年 7 月 （续）	01.8 （续）	更新了 SERDES 控制寄存器 QD_0A 表中 bit 5 的信息。
		更新了 SERDES 控制寄存器 CH_13 表中 bits 3 和 2:0 的信息。
		添加了 PCS 状态寄存器 CH_21 表的注脚。
		参考表中的属性表 —— 更新了 TX_DATARATE_RANGE 和 CHn_RX_DATARATE_RANGE 的属性值。更新了第一个注脚。
2011 年 9 月	01.9	SERDES 控制寄存器 CH_14，更正了寄存器位 tpwrup 的默认值。
		删除了参考表中的属性表中 int_all 的所有保留位和 los hi 位。
		PCS 控制寄存器 QD_02，更新了默认值。
2011 年 11 月	02.0	SERDES_PCS I/O 说明表 —— 更新了 tx_full_clk_ch[3:0] 和 tx_half_clk_ch[3:0] 的说明。
		更新了 Tx 通道间偏移章节。
		更新了字对齐 （字节边界检测）章节的字对齐控制小节。
		更新了外部链路状态机选项章节。
		掉电控制说明表 —— 更新了 tx_pwrup_ch[0:3]_c 信号说明。
		附录 A， SERDES 控制寄存器 QD_0B 表 —— 更新了 Bit 7 的说明。
		附录 A， PCS 控制寄存器 CH_01 表 —— 更新了 Bit 7 的说明。
		附录 C，参考表中的属性表 —— 更新了 {CHn_RXWA, CHn_ILSM} 的属性值。
2012 年 2 月	02.1	更新文档，加入了新的公司标志。
2012 年 4 月	02.2	更新了每款 LatticeECP3 器件的 SERDES/PCS Quad 数量表中的 328 球型 csBGA 封装的相关数据。
		SERDES_PCS I/O 说明表 —— 更新了 refclk2fpga 的说明。
		SERDES_PCS 图形用户界面属性—— SERDES 高级设置选项卡表—— 更新了脚注 3。
		添加了 FPGA 核和复位序列的参考时钟章节。
		更新了通用 8b10b 模式章节。
		更新了 FPGA 接口时钟章节。
		附录 A，通道接口寄存器对应表 —— 更新了 CH_16、 CH_17 和 CH_23 的信息。
		附录 A， SERDES 控制寄存器 CH_11 表 —— 将 Bit 3:2 改为 “ 保留 ” 。
2012 年 5 月	02.3	阐明 8b10b 模式下 LSM 同步的数据模式要求。
2012 年 8 月	02.4	添加了 LatticeECP3-17EA 328 csBGA 器件有限的通道使用限制信息。
		SERDES_PCS I/O 说明表 —— 添加了脚注 3。
		更新了 RX 复位状态图脚注。

附录 A. 配置寄存器

Quad 寄存器概述

表 8-35. Quad 接口寄存器表

BA	寄存器名称	D7	D6	D5	D4	D3	D2	D1	D0
每个 Quad PCS 控制寄存器									
00	QD_00	reg_sync_toggle	force_int	char_mode	xge_mode				
01	QD_01	仅内部使用							
02	QD_02	high_mark[3]	high_mark[2]	high_mark[1]	high_mark[0]	low_mark[3]	low_mark[2]	low_mark[1]	low_mark[0]
03	QD_03						pfifo_clr_sel	仅内部使用	仅内部使用
04	QD_04	仅内部使用							
05	QD_05	仅内部使用							
06	QD_06	仅内部使用							
07	QD_07	仅内部使用							
08	QD_08	仅内部使用							
09	QD_09	ls_sync_status_3_int_ctl	ls_sync_status_2_int_ctl	ls_sync_status_1_int_ctl	ls_sync_status_0_int_ctl	ls_sync_statusn_3_int_ctl	ls_sync_statusn_2_int_ctl	ls_sync_statusn_1_int_ctl	ls_sync_statusn_0_int_ctl
每个 Quad SERDES 控制寄存器									
0A	QD_0A	仅内部使用	保留	tx_refck_sel	refck_dcc_en	refck_rterm		refck_out_sel[1]	refck_out_sel[0]
0B	QD_0B	refck25x	bus8bit_sel	保留	保留	保留	保留	refck_mode[1]	refck_mode[0]
0C	QD_0C	保留	保留	保留	保留	保留	保留	cdr_lol_sel[1]	cdr_lol_sel[0]
0D	QD_0D	仅内部使用	仅内部使用	仅内部使用	pll_lol_sel[1]	pll_lol_sel[0]	tx_vco_ck_div[2]	tx_vco_ck_div[1]	tx_vco_ck_div[0]
0E	QD_0E	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用
0F	QD_0F	plol_int_ctl	-plol_int_ctl	保留	保留	保留	保留	保留	保留
每个 Quad 时钟复位寄存器									
10	QD_10	保留	保留	保留	保留	serdes_pd	serdes_rst	quad_rst	trst
11	QD_11	保留	保留	保留	保留	保留	保留	保留	保留
每个 Quad PCS 状态寄存器									
20	QD_20				int_qd_out	int_ch[3]	int_ch[2]	int_ch[1]	int_ch[0]
21	QD_21	ls_sync_status_3	ls_sync_status_2	ls_sync_status_1	ls_sync_status_0	ls_sync_statusn_3	ls_sync_statusn_2	ls_sync_statusn_1	ls_sync_statusn_0
22	QD_22	ls_sync_status_3_int	ls_sync_status_2_int	ls_sync_status_1_int	ls_sync_status_0_int	ls_sync_statusn_3_int	ls_sync_statuns_2_int	ls_sync_statusn_1_int	ls_sync_statusn_0_int
23	QD_23	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用
24	QD_24	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用
每个 Quad SERDES 状态寄存器									
25	QD_25	plol	-plol	保留	保留	保留	保留	保留	保留
26	QD_26	plol_int	-plol_int	保留	保留	保留	保留	保留	保留
27	QD_27	保留	保留	保留	保留	保留	保留	保留	保留
28	QD_28	保留	保留	保留	保留	保留	保留	保留	保留

每个 Quad PCS 控制寄存器详细信息

表 8-36. PCS 控制寄存器 QD_00

位	名称	说明	类型	默认值
7	reg_sync_toggle	Transition = 复位四个 TX 串行器，最小化 TX 线路间偏移 Level = TX 串行器正常工作	RW	0
6	force_int	1 = 强制产生中断信号 0 = 正常工作	RW	0
5	char_mode	1 = 使能 SERDES 特性模式 0 = 禁止 SERDES 特性模式	RW	0
4	xge_mode	1 = 选择 10Gb 以太网 0 = 根据通道模式选择	RW	0
3:0	保留			

表 8-37. PCS 控制寄存器 QD_01

位	名称	说明	类型	默认值
7:0	仅内部使用			

表 8-38. PCS 控制寄存器 QD_02

位	名称	说明	类型	默认值
7:4	high_mark[3:0]	时钟补偿 FIFO 高位水印。中间值是 4'b1000。	RW	4'b1001
3:0	low_mark[3:0]	时钟补偿 FIFO 低位水印。中间值是 4'b1000。	RW	4'b0111

表 8-39. PCS 控制寄存器 QD_03

位	名称	说明	类型	默认值
7:3	保留			
2	pfifo_clr_sel	1 = pfifo_clr 信号或通道寄存器位清零 FIFO 0 = pfifo_error 内部信号自动清零 FIFO	RW	0
1	仅内部使用			
0	仅内部使用			

表 8-40. PCS 控制寄存器 QD_04

位	名称	说明	类型	默认值
7:0	仅内部使用			

表 8-41. PCS 控制寄存器 QD_05

位	名称	说明	类型	默认值
7:0	仅内部使用			

表 8-42. PCS 控制寄存器 QD_06

位	名称	说明	类型	默认值
7:0	仅内部使用			

表 8-43. PCS 控制寄存器 QD_07

位	名称	说明	类型	默认值
7:0	仅内部使用			

表 8-44. PCS 控制寄存器 QD_08

位	名称	说明	类型	默认值
7:0	仅内部使用			

表 8-45. PCS 控制寄存器 QD_09

位	名称	说明	类型	默认值
7	ls_sync_status_3_int_ctl	1 = 使能 ls_sync_status_3 中断 （同步） 0 = 禁止 ls_sync_status_3 中断 （同步）	RW	0
6	ls_sync_status_2_int_ctl	1 = 使能 ls_sync_status_2 中断 （同步） 0 = 禁止 ls_sync_status_2 中断 （同步）	RW	0
5	ls_sync_status_1_int_ctl	1 = 使能 ls_sync_status_1 中断 （同步） 0 = 禁止 ls_sync_status_1 中断 （同步）	RW	0
4	ls_sync_status_0_int_ctl	1 = 使能 ls_sync_status_0 中断 （同步） 0 = 禁止 ls_sync_status_0 中断 （同步）	RW	0
3	ls_sync_statusn_3_int_ctl	1 = 当 ls_sync_status_3 变为低电平时使能 ls_sync_status_3 中断 （未同步） 0 = 当 ls_sync_status_3 变为低电平时禁止 ls_sync_status_3 中断 （未同步）	RW	0
2	ls_sync_statusn_2_int_ctl	1 = 当 ls_sync_status_2 变为低电平时使能 ls_sync_status_2 中断 （未同步） 0 = 当 ls_sync_status_2 变为低电平时禁止 ls_sync_status_2 中断 （未同步）	RW	0
1	ls_sync_statusn_1_int_ctl	1 = 当 ls_sync_status_1 变为低电平时使能 ls_sync_status_1 中断 （未同步） 0 = 当 ls_sync_status_1 变为低电平时禁止 ls_sync_status_1 中断 （未同步）	RW	0
0	ls_sync_statusn_0_int_ctl	1 = 当 ls_sync_status_0 变为低电平时使能 ls_sync_status_0 中断 （未同步） 0 = 当 ls_sync_status_0 变为低电平时禁止 ls_sync_status_0 中断 （未同步）	RW	0

每个 Quad PCS 控制寄存器详细信息

表 8-46. SERDES 控制寄存器 QD_0A

位	名称	说明	类型	默认值
7	仅内部使用			
6	保留		RW	0
5	TX_REFCK_SEL	TxPLL 参考时钟选择 0 = REFCLKP/N 1 = FPGA 内核	RW	0
4	保留			
3	REFCK_RTERM	参考时钟输入缓冲器的端接电阻 0 = 高阻 1 = 50 欧姆	RW	1
2	保留		RW	0
1	REFCLK_OUT_SEL[1]	0 = refclk2fpga 输出禁止 1 = refclk2fpga 输出使能	RW	0
0	REFCLK_OUT_SEL[0]	0 = tx_refck_local 输出使能 1 = tx_refck_local 输出禁止	RW	0

注：请参见图 8-9，了解参考时钟选择控制信号。

表 8-47. SERDES 控制寄存器 QD_0B

位	名称	说明	类型	默认值
7	REFCK25X	1 = 内部高速位时钟是 25x 0 = 参见 REFCK_MODE	RW	0
6	BUS8BIT_SEL	1 = 选择 8 位总线宽度 0 = 选择 10 位总线宽度	RW	0
5	保留		RW	0
4	保留	1 = 从相邻 quad 选择 RX REFCLK 0 = 从当前 quad 选择 RX REFCLK	RW	0
3	保留	1 = 从相邻 quad 选择 TX REFCLK 0 = 从当前 quad 选择 TX REFCLK	RW	0
2	保留	1 = 对相邻 quad 使能 REFCLK 0 = 对相邻 quad 禁用 REFCLK	RW	0
1:0	REFCK_MODE[1:0]	如果 REFCK25X = 0，则： 00 = 内部高速位时钟为 20x 01 = 内部高速位时钟为 10x 10 = 内部高速位时钟为 16x 11 = 内部高速位时钟为 8x 如果 REFCLK25X = 1，则： xx = 内部高速位时钟为 25x	RW	00

表 8-48. PCS 控制寄存器 QD_0C

位	名称	说明	类型	默认值
7:2	保留			
1:0	CDR_LOL_SET[1:0]	CDR 失锁设置 Lock / Unlock 00 = +/-1000ppm x2 / +/-1500ppm x2 01 = +/-2000ppm x2 / +/-2500ppm x2 10 = +/-4000ppm / +/-7000ppm 11 = +/-300ppm / +/-450ppm	RW	00

表 8-49. PCS 控制寄存器 QD_0D

位	名称	说明	类型	默认值
7:6	仅内部使用			
5	仅内部使用			
4:3	PLL_LOL_SET[1:0]	00 = +/- 1350ppm x2 01 = +/- 2400ppm x2 10 = +/- 6800ppm 11 = +/- 400ppm	RW	0
2:0	TX_VCO_CK_DIV[2:0]	VCO 输出频率选择 00x = 除 1 / 01x = 除 2 100 = 除 4 / 101 = 除 8 110 = 除 16 / 111 = 除 32	RW	0

表 8-50. PCS 控制寄存器 QD_0E

位	名称	说明	类型	默认值
7:0	仅内部使用			

表 8-51. PCS 控制寄存器 QD_0F

位	名称	说明	类型	默认值
7	PLOL_INT_CTL	1 = PLOL 失锁中断使能 0 = PLOL 失锁中断禁止	RO CR	0
6	-PLOL_INT_CTL	1 = PLOL 获得锁定中断使能 0 = PLOL 获得锁定中断禁止	RO CR	0
5:0	保留			

每个 Quad 复位和时钟控制寄存器详细信息

表 8-52. PCS 控制寄存器 QD_10

位	名称	说明	类型	默认值
7:4	保留			
3	serdes_pd	0 = 触发断电	RW	1
2	serdes_rst	1 = 触发 serdes 复位	RW	0
1	quad_rst	1 = 触发 quad 复位	RW	0
0	trst	1 = TX 复位	RW	0

表 8-53. PCS 控制寄存器 QD_11

位	名称	说明	类型	默认值
7:0	保留			

每个 Quad PCS 状态寄存器详细信息

表 8-54. PCS 状态寄存器 QD_20

位	名称	说明	类型	Int?
7:6	保留			
5	ion_delay	0 = 来自 tri_ion 的延迟全局复位	RO	否
4	int_qd_out	1 = 每个 quad 中断状态	RO	否
3:0	int_ch_out[3:0]	1 = 每个通道中断状态	RO	否

表 8-55. PCS 状态寄存器 QD_21

位	名称	说明	类型	Int?
7	ls_sync_status_3	1 = sync_status_3 产生报警 0 = sync_status_3 未产生报警	RO	是
6	ls_sync_status_2	1 = sync_status_2 产生报警 0 = sync_status_2 未产生报警	RO	是
5	ls_sync_status_1	1 = sync_status_1 产生报警 0 = sync_status_1 未产生报警	RO	是
4	ls_sync_status_0	1 = sync_status_0 产生报警 0 = sync_status_0 未产生报警	RO	是
3	ls_sync_statusn_3	1 = 当 sync_status_3 变为低电平时， sync_status_3 产生报警 （未同步） 0 = 当 sync_status_3 变为低电平时， sync_status_3 未产生报警 （未同步）	RO	是
2	ls_sync_statusn_2	1 = 当 sync_status_2 变为低电平时， sync_status_2 产生报警 （未同步） 0 = 当 sync_status_2 变为低电平时， sync_status_2 未产生报警 （未同步）	RO	是
1	ls_sync_statusn_1	1 = 当 sync_status_1 变为低电平时， sync_status_1 产生报警 （未同步） 0 = 当 sync_status_1 变为低电平时， sync_status_1 未产生报警 （未同步）	RO	是
0	ls_sync_statusn_0	1 = 当 sync_status_0 变为低电平时， sync_status_0 产生报警 （未同步） 0 = 当 sync_status_0 变为低电平时， sync_status_0 未产生报警 （未同步）	RO	是

表 8-56. PCS 中断状态寄存器 QD_22

位	名称	说明	类型	Int?
7	ls_sync_status_3_int	1 = sync_status_3 产生报警 0 = sync_status_3 未产生报警	RO CR	是
6	ls_sync_status_2_int	1 = sync_status_2 产生报警 0 = sync_status_2 未产生报警	RO CR	是
5	ls_sync_status_1_int	1 = sync_status_1 产生报警 0 = sync_status_1 未产生报警	RO CR	是
4	ls_sync_status_0_int	1 = sync_status_0 产生报警 0 = sync_status_0 未产生报警	RO CR	是
3	ls_sync_statusn_3_int	1 = 当 sync_status_3 变为低电平时， sync_status_3 产生中断 （未同步） 0 = 当 sync_status_3 变为低电平时， sync_status_3 未产生中断 （未同步）	RO CR	是
2	ls_sync_statusn_2_int	1 = 当 sync_status_2 变为低电平时， sync_status_2 产生中断 （未同步） 0 = 当 sync_status_2 变为低电平时， sync_status_2 未产生中断 （未同步）	RO CR	是
1	ls_sync_statusn_1_int	1 = 当 sync_status_1 变为低电平时， sync_status_1 产生中断 （未同步） 0 = 当 sync_status_1 变为低电平时， sync_status_1 未产生中断 （未同步）	RO CR	是
0	ls_sync_statusn_0_int	1 = 当 sync_status_0 变为低电平时， sync_status_0 产生中断 （未同步） 0 = 当 sync_status_0 变为低电平时， sync_status_0 未产生中断 （未同步）	RO CR	是

表 8-57. PCS 状态寄存器 QD_23

位	名称	说明	类型	默认值
7:0	仅内部使用			

表 8-58. PCS 状态寄存器 QD_24

位	名称	说明	类型	默认值
7:0	仅内部使用			

每个 Quad SERDES 状态寄存器详细信息

表 8-59. SERDES 状态寄存器 QD_25

位	名称	说明	类型	Int?
7	PLOL	1 = PLL 失锁	RO	是
6	-PLOL	1 = PLL 获得锁定	RO	是
5:0	保留			

表 8-60. SERDES 中断状态寄存器 QD_26

位	名称	说明	类型	Int?
7	PLOL_INT	1 = PLOL 触发中断 0 = PLOL 未触发中断	RO CR	是
6	-PLOL_INT	1 = -PLOL 触发中断 0 = -PLOL 未触发中断	RO CR	是
5:0	保留			

表 8-61. SERDES 状态寄存器 QD_27

位	名称	说明	类型	默认值
7:0	保留			

表 8-62. SERDES 状态寄存器 QD_28

位	名称	说明	类型	默认值
7:0	保留			

通道寄存器概述

表 8-63. 通道接口寄存器表

BA	寄存器名称	D7	D6	D5	D4	D3	D2	D1	D0
每个通道通用控制寄存器									
00	CH_00					rio_mode	pcie_mode	fc_mode	uc_mode
01	CH_01	word_align_enable	仅内部使用	仅内部使用	ge_an_enable		仅内部使用	invert_tx	invert_rx
02	CH_02	pfifo_clr	pcie_ei_en	pcs_det_time_sel[1]	pcs_det_time_sel[0]	rx_gear_mode	tx_gear_mode	rx_ch	tx_ch
03	CH_03		sb_bypass	sb_pfifo_lp	internal use only	enc_bypass	仅内部使用	tx_gear_bypass	fb_loopback
04	CH_04	lsm_sel	ilsm_en	rx_gear_bypass	ctc_bypass	dec_bypass	wa_bypass	rx_sb_bypass	sb_loopback
05	CH_05	min_ipg_cnt[1]	min_ipg_cnt[0]	match_4_enable	match_2_enable				
06	CH_06	cc_match_1[7]	cc_match_1[6]	cc_match_1[5]	cc_match_1[4]	cc_match_1[3]	cc_match_1[2]	cc_match_1[1]	cc_match_1[0]
07	CH_07	cc_match_2[7]	cc_match_2[6]	cc_match_2[5]	cc_match_2[4]	cc_match_2[3]	cc_match_2[2]	cc_match_2[1]	cc_match_2[0]
08	CH_08	cc_match_3[7]	cc_match_3[6]	cc_match_3[5]	cc_match_3[4]	cc_match_3[3]	cc_match_3[2]	cc_match_3[1]	cc_match_3[0]
09	CH_09	cc_match_4[7]	cc_match_4[6]	cc_match_4[5]	cc_match_4[4]	cc_match_4[3]	cc_match_4[2]	cc_match_4[1]	cc_match_4[0]
0A	CH_0A	cc_match_4[9]	cc_match_4[8]	cc_match_3[9]	cc_match_3[8]	cc_match_2[9]	cc_match_2[8]	cc_match_1[9]	cc_match_1[8]
0B	CH_0B	udf_comma_mask[7]	udf_comma_mask[6]	udf_comma_mask[5]	udf_comma_mask[4]	udf_comma_mask[3]	udf_comma_mask[2]	udf_comma_mask[1]	udf_comma_mask[0]
0C	CH_0C	udf_comma_a[7]	udf_comma_a[6]	udf_comma_a[5]	udf_comma_a[4]	udf_comma_a[3]	udf_comma_a[2]	udf_comma_a[1]	udf_comma_a[0]
0D	CH_0D	udf_comma_b[7]	udf_comma_b[6]	udf_comma_b[5]	udf_comma_b[4]	udf_comma_b[3]	udf_comma_b[2]	udf_comma_b[1]	udf_comma_b[0]
0E	CH_0E	udf_comma_a[9]	udf_comma_a[8]	udf_comma_b[9]	udf_comma_b[8]	udf_comma_mask[9]	udf_comma_mask[8]		
0F	CH_0F					cc_underrun_int_ctl	cc_overrun_int_ctl	fb_rx_fifo_error_int_ctl	fb_tx_fifo_error_int_ctl
每个通道 SERDES 控制寄存器									
10	CH_10	req_en	req_lvl_set	rcv_dcc_en	rate_sel[1]	rate_sel[0]	rx_dco_ck_div[2]	rx_dco_ck_div[1]	rx_dco_ck_div[0]
11	CH_11	仅内部使用	仅内部使用	lb_ctl[1]	lb_ctl[0]	仅内部使用	仅内部使用	rterm_rx[1]	rterm_rx[0]
12	CH_12	tdrv_amp[2]	tdrv_amp[1]	tdrv_amp[0]	tdrv_pre_set[4]	tdrv_pre_set[3]	tdrv_pre_set[2]	tdrv_pre_set[1]	tdrv_pre_set[0]
13	CH_13	ldr_core2tx_sel				仅内部使用	仅内部使用	仅内部使用	仅内部使用
14	CH_14	tx_div11_sel	tdrv_dat_sel[1]	tdrv_dat_sel[0]	tdrv_ppre_en	rterm_tx[1]	rterm_tx[0]	rate_mode_tx	tpwrup
15	CH_15	仅内部使用	仅内部使用	仅内部使用	仅内部使用	ldr_rx2core_en	rx_refck_sel	rate_mode_rx	rpwrup
16	CH_16	rx_div11_sel	rlos_sel				rlos_lset[2]	rlos_lset[1]	rlos_lset[0]
17	CH_17		pci_det_done_int_ctl	rlos_lo_int_ctl	-rlos_lo_int_ctl			rlol_int_ctl	~rlol_int_ctl
每个通道时钟复位寄存器									
18	CH_18	仅内部使用	仅内部使用				rrst	lane_rx_rst	lane_tx_rst
19	CH_19				tx_f_clk_dis	tx_h_clk_en	rx_f_clk_dis	rx_h_clk_en	sel_sd_rx_clk
每个通道通用状态寄存器									
20	CH_20					cc_underrun	cc_overrun	fb_rx_fifo_error	fb_tx_fifo_error
21	CH_21	prbs_error_cnt[7]	prbs_error_cnt[6]	prbs_error_cnt[5]	prbs_error_cnt[4]	prbs_error_cnt[3]	prbs_error_cnt[2]	prbs_error_cnt[1]	prbs_error_cnt[0]
22	CH_22					wa_offset[3]	wa_offset[2]	wa_offset[1]	wa_offset[0]
23	CH_23					cc_underrun_int	cc_overrun_int	fb_rx_fifo_error_int	fb_tx_fifo_error_int
24	CH_24		ffs_ls_sync_status	fb_rxrst_o	fb_txrst_o			cc_re_o	cc_we_o
25	CH_25								
每个通道 SERDES 状态寄存器									
26	CH_26		pcie_det_done	rlos_lo	-rlos_lo	rlos_hi	-rlos_hi	rlol	-rlol
27	CH_27	仅内部使用	仅内部使用	仅内部使用	仅内部使用			cdr_traine_done	pci_connect
28	CH_28	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用
29	CH_29	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用	仅内部使用
2A	CH_2A		pci_det_done_int	rlos_lo_int	-rlos_lo_int	rlos_hi_int	-rlos_hi_int	rlol_int	-rlol_int
2B	CH_2B								
2C	CH_2C								

每个通道 PCS 控制寄存器详细信息

表 8-64. PCS 控制寄存器 CH_00

位	名称	说明	类型	默认值
7:4	保留			
3	rio_mode	1 = 选择 RapidIO 模式 0 = 选择其他模式 （10GbE， 1GbE）	RW	0
2	pcie_mode	1 = 选择 PCI Express 模式 0 = 选择其他模式 （RapidIO， 10GbE， 1GbE）	RW	0
1	fc_mode	1 = 选择 Fibre Channel 模式 0 = 选择其他模式 （PCI Express， RapidIO， 10GbE， 1GbE）	RW	0
0	uc_mode	1 = 选择 User Configured （G8B10B，仅 8BSER，仅 10BSER）模式 0 = 选择其他模式 （Fibre Channel， PCI Express， RapidIO， 10GbE， 1GbE）	RW	0

表 8-65. PCS 控制寄存器 CH_01

位	名称	说明	类型	默认值
7	word_align_enable	1 = 使能连续逗号对齐 0 = 禁止连续逗号对齐	RW	0
6	仅内部使用			
5	仅内部使用			
4	ge_an_enable	1 = 使能 GbE 自动协商 0 = 禁止 GbE 自动协商	RW	0
3	保留			
2	仅内部使用			
1	invert_tx	1 = 发送数据取反 0 = 发送数据不取反	RW	0
0	invert_rx	1 = 接收数据取反 0 = 接收数据不取反	RW	0

表 8-66. PCS 控制寄存器 CH_02

位	名称	说明	类型	默认值
7	pfifo_clr	1 = 清零 PFIFO，如果 quad 寄存器位 pfifo_clr_sel 设为 1。该信号和接口信号 pfifo_clr 是逻辑或。 0 = 正常工作	RW	0
6	pcie_ei_en	1 = PCI Express 电气空闲使能 0 = 正常工作	RW	0
5:4	pcs_det_time_sel[1:0]	PCS 连接检测时间 11 = 16us 10 = 4us 01 = 2us 00 = 8us	RW	0
3	rx_gear_mode	1 = 在所选通道的接收路径使能 2:1 gearing 0 = 在所选通道的接收路径禁止 2:1 gearing		
2	tx_gear_mode	1 = 在所选通道的发送路径使能 2:1 gearing 0 = 在所选通道的发送路径禁止 2:1 gearing	RW	0
1	rx_ch	1 = 接收的输出可在测试特性引脚上进行监控。测试特性模式 （PCS 控制寄存器 QD_03 第 6 位）应当设为 ‘1’。	RW	0
0	tx_ch	1 = 发送 PCS 输入，来自测试特性端口。测试特性模式应当使能。	RW	0

表 8-67. PCS 控制寄存器 CH_03

位	名称	说明	类型	默认值
7	保留			
6	sb_bypass	1 = 旁路 TX SERDES 桥 0 = 正常工作	RW	0
5	sb_pfifo_lp	1 = 从 RX 到 TX 通过并行 FIFO 使能并行回环 0 = 正常工作	RW	0
4	仅内部使用			
3	enc_bypass	1 = 旁路 8b10b 编码器 0 = 正常工作		
2	仅内部使用			
1	tx_gear_bypass	1 = 旁路 PCS TX gear 0 = 正常工作	RW	0
0	仅内部使用			

表 8-68. PCS 控制寄存器 CH_04

位	名称	说明	类型	默认值
7	lsm_sel	1 = 选择外部 RX 链路状态机 0 = 选择内部链路状态机	RW	0
6	ilsm_en	1 = 强制使能 RX 链路状态机 0 = 强制禁止 RX 链路状态机	RW	0
5	rx_gear_bypass	1 = 旁路 PCS RX gear 0 = 正常工作	RW	0
4	ctc_bypass	1 = 旁路时钟容限补偿 0 = 选择正常数据	RW	0
3	dec_bypass	1 = 旁路 8b10b 解码器 0 = 正常工作		
2	wa_bypass	1 = 旁路字对齐 0 = 正常工作	RW	0
1	rx_sb_bypass	1 = 旁路 RX SERDES 桥 0 = 正常工作	RW	0
0	sb_loopback	1 = 使能 PCS 回环， SERDES 桥从 TX 到 RX 0 = 正常工作	RW	0

表 8-69. PCS 控制寄存器 CH_05

位	名称	说明	类型	默认值
7:6	min_ipg_cnt[1:0]	执行最小 IPG	RW	11
5	match_4_enable	1 = 使能 4 字符跳过匹配 （使用匹配 4， 3， 2， 1）	RW	0
4	match_2_enable	1 = 使能 2 字符跳过匹配 （使用匹配 4， 3）	RW	1
3:0	保留			

表 8-70. PCS 控制寄存器 CH_06

位	名称	说明	类型	默认值
7:0	cc_match_1[7:0]	用户定义的时钟补偿器跳过模式 1 的低 8 位	RW	8’h00

表 8-71. PCS 控制寄存器 CH_07

位	名称	说明	类型	默认值
7:0	cc_match_2[7:0]	用户定义的时钟补偿器跳过模式 2 的低 8 位	RW	8’h00

表 8-72. PCS 控制寄存器 CH_08

位	名称	说明	类型	默认值
7:0	cc_match_3[7:0]	用户定义的时钟补偿器跳过模式 3 的低 8 位	RW	8’hBC

表 8-73. PCS 控制寄存器 CH_09

位	名称	说明	类型	默认值
7:0	cc_match_4[7:0]	用户定义的时钟补偿器跳过模式 4 的低 8 位	RW	8’h50

表 8-74. PCS 控制寄存器 CH_0A

位	名称	说明	类型	默认值
7:6	cc_match_4[9:8]	用户定义的时钟补偿器跳过模式 4 的高 2 位 [9] = 差异错误 [8] = K 控制	RW	2'b01
5:4	cc_match_3[9:8]	用户定义的时钟补偿器跳过模式 3 的高 2 位 [9] = 差异错误 [8] = K 控制	RW	2'b01
3:2	cc_match_2[9:8]	用户定义的时钟补偿器跳过模式 2 的高 2 位 [9] = 差异错误 [8] = K 控制	RW	2'b00
1:0	cc_match_1[9:8]	用户定义的时钟补偿器跳过模式 1 的高 2 位 [9] = 差异错误 [8] = K 控制	RW	2'b00

表 8-75. PCS 控制寄存器 CH_0B

位	名称	说明	类型	默认值
7:0	udf_comma_mask[7:0]	用户定义的逗号掩码的低 8 位	RW	8'hFF

表 8-76. PCS 控制寄存器 CH_0C

位	名称	说明	类型	默认值
7:0	udf_comma_a[7:0]	用户定义的逗号字符 'a' 的低 8 位	RW	8'h83

表 8-77. PCS 控制寄存器 CH_0D

位	名称	说明	类型	默认值
7:0	udf_comma_b[7:0]	用户定义的逗号字符 'b' 的低 8 位	RW	8'h7C

表 8-78. PCS 控制寄存器 CH_0E

位	名称	说明	类型	默认值
7:6	udf_comma_a[9:8]	用户定义的逗号字符 'a' 的高 2 位	RW	2'b10
5:4	udf_comma_b[9:8]	用户定义的逗号字符 'b' 的高 2 位	RW	2'b01
3:2	udf_comma_mask[9:8]	用户定义的逗号掩码的高 2 位	RW	2'b11[1]
1:0	保留			

1. 在大多数应用中，K28.5 用作逗号字符。默认的掩码值是 1111111111。在 G8B10B 模式可以使用任意逗号字符，因此掩码是 1111111100，以检测所有的三种逗号字符， K28.1、 28.5、 28.7。

表 8-79. PCS 中断控制寄存器 CH_0F

位	名称	说明	类型	默认值
7:4	保留			
3	cc__underrun_int_ctl	1 = 使能 cc_underrun 中断 0 = 禁止 cc_underrun 中断	RW	0
2	cc_overrun_int_ctl	1 = 使能 cc_overrun 中断 0 = 禁止 cc_overrun 中断	RW	0
1	fb_rx_fifo_error_int_ctl	1 = 使能接收 FPGA 桥 FIFO 空 / 满条件下的中断	RW	0
0	fb_tx_fifo_error_int_ctl	1 = 使能发送 FPGA 桥 FIFO 空 / 满条件下的中断	RW	0

每个通道 SERDES 控制寄存器详细信息

除非另有说明，在任何 SERDES 控制寄存器被写入后，所有通道必须进行复位。

表 8-80. SERDES 控制寄存器 CH_10

位	名称	说明	类型	默认值
7	REQ_EN	1= 使能接收器均衡 0= 禁止接收器均衡	RW	0
6	REQ_LVL_SET	均衡器级别设置 1 = 远距离均衡 0 = 中距离均衡	RW	0
5	RCV_DCC_EN	1 = 使能接收器 DC 耦合 0 = 使能接收器 AC 耦合	RW	0
4:3	RATE_SEL[1:0]	均衡器频率范围选择： 00 = 高频率范围 01 = 中频率范围 10 = 低频率范围 11 = 保留	RW	00
2:0	RX_DCO_CK_DIV[2:0]	VCO 输出频率选择： 00x = 除 1 01x = 除 2 100 = 除 4 101 = 除 8 110 = 除 16 111 = 除 32	RW	000

表 8-81. SERDES 控制寄存器 CH_11

位	名称	说明	类型	默认值
7:4	LB_CTL[3:0]	回环控制： [3] = 仅内部使用 [2] = 仅内部使用 [1] = slb_eq2t_en，串行 LB 从均衡器到驱动器使能 [0] = slb_t2r_en，串行 tx 到 rx LB 使能	R/W	4'h0
3:2	保留			
1:0	RTERM_RX[1:0]	00 = HiZ， 01 = 50 Ohm， 10 = 60 Ohm， 11 = 75 Ohm	R/W	2'b01

表 8-82. SERDES 控制寄存器 CH_12

位	名称	说明	类型	默认值
7:5	TDRV_AMP[2:0]	CML 驱动放大器幅度设置: 000 = 0% 001 = +8% 010 = +11% 011 = +20% 100 = -17% 101 = -12% 110 = -10% 111 = -6%	RW	000
4:0	TDRV_PRE_SET[4:0]	TX 驱动预加重级别设置 [2:0] 000 = 0% 001 = 5% 010 = 12% 011 = 18% 100 = 25% 101 = 33% 110 = 40% 111 = 48% [4:3] 精调 00 = 0% 01 = 2% 10 = 3% 11 = 5%	RW	5'b00000

表 8-83. SERDES 控制寄存器 CH_13

位	名称	说明	类型	默认值
7	ldr_core2tx_sel	1 = 从 FPGA 内核选择低速串行数据	RW	0
6	pden_sel	保留	RW	0
5:4	保留			
3	TDRV_AMP_BOOST	TX 驱动幅度提升 0 = 0% 1 = -25%	RW	0
2:0	TDRV_DRVCUR_SET[2:0]	000 = 48% 001 = 30% 010 = 60% 011 = 50% 100 = 0% 101 = -7% 110 = 19% 111 = 8%	RW	100

表 8-84. SERDES 控制寄存器 CH_14

位	名称	说明	类型	默认值
7	TX_DIV11_SEL	0 = 发送的全速选择 （高清的 SMPTE） 1 = 除 11 的发送选择 （标清的 SMPTE）	RW	0
6:5	TDRV_DAT_SEL [1:0]	驱动器输出选择： 00 = 来自串行器的数据多路复用到驱动器 （正常工作） 11 = 如果 slb_eq2t_en='1'，串行 LB 从均衡器到驱动器	RW	00
4	TDRV_PRE_EN	1 = TX 驱动器预加重使能 0 = TX 驱动器预加重禁止	RW	0
3:2	RTERM_TX[1:0]	TX 端接电阻选择。当选择 PCI Express 时禁止。 0x = 5K 欧姆 10 = 50 欧姆 11 = 75 欧姆	RW	10
1	RATE_MODE_TX	0 = 发送全速选择 1 = 发送半速选择	RW	0
0	tpwrup	0 = 断电发送通道 1 = 上电发送通道	RW	0

表 8-85. SERDES 控制寄存器 CH_15

位	名称	说明	类型	默认值
7	仅内部使用			
6	仅内部使用			
5	仅内部使用			
4	仅内部使用			
3	ldr_rx2core_en	1 = 使能边界扫描输入路径，用于连接高速接收输入到 FPGA 中的较低速的 SERDES （针对带外应用）	RW	0
2	rx_refck_sel	RX CDR 参考时钟选择 0 = REFCLKP/N 1 = FPGA 内核	RW	0
1	RATE_MODE_RX	0 = 接收的全速选择 1 = 接收的半速选择	RW	0
0	rpwrup	0 = 断电接收通道 1 = 上电接收通道	RW	0

表 8-86. SERDES 控制寄存器 CH_16

位	名称	说明	类型	默认值
7	RX_DIV11_SEL	0 = 接收的全速选择 （高清的 SMPTE） 1 = 除 11 的接收选择 （标清的 SMPTE）	RW	0
6	rlos_sel	1 = 选择 rlos_hi 0 = 选择 rlos_lo	RW	0
5:3	保留		RW	000
2:0	RLOS_LSET[2:0]	针对更小的摆幅， LOS 检测器参考电流调整 000 = 默认 010 = +15% 011 = +25%	RW	010

表 8-87. SERDES 中断控制寄存器 CH_17

位	名称	说明	类型	默认值
7	保留			
6	pci_det_done_int_ctl	1 = 使能 PCI Express 的远端接收器的中断检测	RW	0
5	rlos_lo_int_ctl	1 = 当输入电平低于编程的 LOW 阈值时 （使用 rlos_set），使能 RX 信号丢失中断	RW	0
4	-rlos_lo_int_ctl	1 = 当输入电平满足或高于编程的 LOW 阈值时，使能 RX 信号丢失中断	RW	0
3	保留			
2	保留			
1	rlol_int_ctl	1 = 由于接收器失锁使能中断	RW	0
0	-rlol_int_ctl	1 = 由于接收器从失锁状态恢复，使能中断	RW	0

每个通道的复位和时钟控制寄存器详细信息

表 8-88. 复位和时钟控制寄存器 CH_18

位	名称	说明	类型	默认值
7	仅内部使用			
6	仅内部使用			
5:3	保留			
2	rrst	1 = RX 复位	RW	0
1	lane_rx_rst	1 = 复位接收逻辑	RW	0
0	lane_tx_rst	1 = 复位发送逻辑	RW	0

表 8-89. 复位和时钟控制寄存器 CH_19

位	名称	说明	类型	默认值
7:5	保留			
4	tx_f_clk_dis	1 = 禁止 tx_f_clk	RW	0
3	tx_h_clk_en	1 = 使能 tx_h_clk	RW	0
2	rx_f_clk_dis	1 = 禁止 rx_f_clk	RW	0
1	rx_h_clk_en	1 = 使能 rx_h_clk	RW	0
0	sel_sd_rx_clk	1 = 选择 sd_rx_clk	RW	0

每个通道 PCS 状态寄存器详细信息

表 8-90. PCS 状态寄存器 CH_20

位	名称	说明	类型	Int?
7:5	保留			
4	pfifo_error	1 = 并行 FIFO 错误 0 = 无并行 FIFO 错误	RO	是
3	cc_underrun	1 = CTC FIFO 欠载 0 = CTC FIFO 未欠载	RO	是
2	cc_overrun	1 = CTC FIFO 过载 0 = CTC FIFO 未过载	RO	是
1	fb_rx_fifo_error	1 = FPGA 桥 (FB) RX FIFO 过载 0 = FB RX FIFO 未过载	RO	是
0	fb_tx_fifo_error	1 = FPGA 桥 (FB) TX FIFO 过载 0 = FB TX FIFO 未过载	RO	是

表 8-91. PCS 状态寄存器 CH_21

位	名称	说明	类型	Int?
7:0	prbs_errors[1]	PRBS 错误计数。读时清零。停止翻转。	RO CR	否

1. 内置的 PRBS 发生器和校验器仅供内部使用。

表 8-92. PCS 状态寄存器 CH_22

位	名称	说明	类型	Int?
7:4	保留			
3:0	wa_offset[3:0]	字对齐偏移	RO	否

表 8-93. PCS 中断状态寄存器 CH_23

位	名称	说明	类型	Int?
7:4	保留			
3	cc_underrun_int	1 = cc_underrun 产生中断 0 = cc_underrun 未产生中断	RO CR	是
2	cc_overrun_int	1 = cc_overrun 产生中断 0 = cc_overrun 未产生中断	RO CR	是
1	fb_rx_fifo_error_int	1 = fb_rx_fifo_error 产生中断 0 = fb_rx_fifo_error 未产生中断	RO CR	是
0	fb_tx_fifo_error_int	1 = fb_tx_fifo_error 产生中断 0 = fb_tx_fifo_error 未产生中断	RO CR	是

表 8-94. PCS 状态寄存器 CH_24

位	名称	说明	类型	Int?
7	保留			
6	ffs_ls_sync_status	1 = 链路状态机同步 0 = 链路状态机未同步	RO	否
5	fb_rxrst_o	1 = FPGA 桥 RX 正常工作 0 = FPGA 桥 RX 复位	RO	否
4	fb_txrst_o	1 = FPGA 桥 TX 正常工作 0 = FPGA 桥 TX 复位	RO	否
3	保留			
2	保留			
1	cc_re_o	1 = CTC FIFO 读使能 0 = CTC FIFO 读禁止	RO	否
0	cc_we_o	1 = CTC FIFO 写使能 0 = CTC FIFO 写禁止	RO	否

表 8-95. PCS 状态寄存器 CH_25

位	名称	说明	类型	Int?
7	保留			

每个通道 SERDES 状态寄存器详细信息

表 8-96. SERDES 状态寄存器 CH_26

位	名称	说明	类型	Int?
7	保留			
6	pci_det_done	1 = SERDES 发送器未完成接收器检测流程 0 = SERDES 发送器完成了接收器检测流程	RO CR	是
5	rlos_lo	1 = 指示接收器检测到输入信号低于已编程 LOW 阈值	RO CR	是
4	-rlos_lo	1 = 指示接收器检测到输入信号大于或等于已编程 LOW 阈值	RO CR	是
3	保留		RO CR	是
2	保留		RO CR	是
1	rlol	1 = 指示 CDR 丢失数据锁定。 CDR 锁定为参考时钟	RO	是
0	-rlol	1 = 指示 CDR 已经锁定数据	RO	是

表 8-97. SERDES 状态寄存器 CH_27

位	名称	说明	类型	Int?
7	仅内部使用			
6	仅内部使用			
5	仅内部使用			
4	仅内部使用			
3:2	保留			
1	cdr_trained	1 = 指示 CDR 训练完成	RO	否
0	pci_connect	1 = SERDES 发送器检测到接收器 （在发送器件端） 0 = SERDES 发送器未检测到接收器 （在发送器件端）	RO	否

表 8-98. SERDES 状态寄存器 CH_28

位	名称	说明	类型	Int?
7:0	仅内部使用			

表 8-99. SERDES 状态寄存器 CH_29

位	名称	说明	类型	Int?
7:0	仅内部使用			

表 8-100. SERDES 中断状态寄存器 CH_2A

位	名称	说明	类型	Int?
7	保留			
6	pci_det_done_int	1 = 产生 pci_det_done 中断	RO CR	是
5	rlos_lo_int	1 = 产生 rlos_lo 中断	RO CR	是
4	-rlos_lo_int	1 = 产生 -rlos_lo 中断	RO CR	是
3	保留		RO CR	是
2	保留		RO CR	是
1	rlol_int	1 = 产生 rlol 中断	RO CR	是
0	-rlol_int	1 = 产生 -rlol 中断	RO CR	是

表 8-101. PCS 状态寄存器 CH_2B

位	名称	说明	类型	Int?
7	保留			

表 8-102. PCS 状态寄存器 CH_2C

位	名称	说明	类型	Int?
7	保留			

附录 B. 各种标准下的寄存器设置

各种标准下的每个通道寄存器的设置

表 8-103. 各种标准下的每个通道寄存器的设置

字符	1GbE	10GbE	1GFC	PCI-Ex	RapidIO
K23.7 (F7)	Carrier extend			PAD	
K27.7 (FB)	SOP	ST		Start TLP	A （对齐）
K28.0 (1C)		SKIP R		SKIP	SC
K28.1 (3C)				FTS	
K28.2 (5C)		SoS		Start DLP	
K28.3 (7C)		ALIGN A		IDLE	PD
K28.4 (9C)		SEQ			
K28.5 (BC)	+D5.6 or D16.2 = IDLE	SYNC K	+D21.4+D21.5 +D21.5 = IDLE	COMMA （用于对齐）	K
K28.6 (DC)					
K28.7 (FC)					R （跳过）
K29.7 (FD)	EOP	T		END	
K30.7 (FE)	ERR	ERR		END BAD	

各种标准下每个 Quad 的寄存器设置

表 8-104. 各种标准下每个 Quad 的寄存器设置

寄存器	1GbE	10GbE	1G, 2G FC	PCI-Ex 1x	PCI-Ex 4x	RapidIO 1x	RapidIO 4x
comma_a_lo	hex 03	hex 03	hex 03	hex 03	hex 03	hex 03	hex 03
comma_b_lo	hex FC	hex FC	hex FC	hex FC	hex FC	hex FC	hex FC
comma_mask_lo	hex 7F	hex 7F	hex 7F	hex 7F	hex 7F	hex 7F	hex 7F

附录 C. 参考表中的属性

表 8-105. 参考表中的属性

独立的属性名称	非独立的属性名称	属性值	寄存器映射
QUAD_MODE	{Qn_REFCK_NQ_EN}[8]	SINGLE :{0} MASTER :{1} SLAVE :{1} SLAVE_END :{0}	{QD_0b[2]}
CHn_PROTOCOL	{10G_MODE, CHn_PROT_MODE, CHn_RX_DET, CHn_GE_AN_EN}	GIGE :{0, 0000,00,1} FC :{0,0010,00,0} XAUI :{1,0000,00,0} SRIO :{0,1000,00,0} PCIE :{0,0100,00,0} SDI :{0,0001,00,0} G8B10B :{0,0001,00,0} 10BSER :{0,0001,00,0} 8BSER :{0,0001,00,0} CPRI :{0,0001,00,0} OBSAI :{0,0001,00,0}	{QD_00[4], CH_00[3:0], CH_02[5:4], CH_01[4]}
CHn_MODE	{CHn_TXPWDNB, CHn_RXPWDNB}	RXTX :{11} RXONLY :{01} TXONLY :{10} DISABLED :{00}	{CH_14[0], CH_15[0]}
TX_DATARATE_RANGE	{PLL_DIV}	LOWLOW :{110} LOW :{101} MEDLOW :{100} MED :{010} MEDHIGH :{010} HIGH :{000}	{QD_0D[2:0]}
CHn_RX_DATARATE_ RANGE	{CHn_CDR_DIV}	LOWLOW :{110} LOW :{101} MEDLOW :{100} MED :{010} MEDHIGH :{000} HIGH :{000}	{CH_10[2:0]}
REFCK_MULT	{REFCK25X, REFCK_MODE}	8X :{0,11} 10X:{0,01} 16X:{0,10} 20X:{0,00} 25X:{1,00}	{QD_0b[7], QD_0b[1:0]}
CHn_RX_DATA_RATE	{CHn_RX_RATE_MODE, CHn_RX_DIV11}	FULL :{00} DIV2 :{10} DIV11:{01}	{CH_15[1], CH_16[7]}
CHn_TX_DATA_RATE	{CHn_TX_RATE_MODE, CHn_TX_DIV11}	FULL :{00} DIV2 :{10} DIV11:{01}	{CH_14[1], CH_14[7]}

表 8-105. 参考表中的属性 （续）

独立的属性名称	非独立的属性名称	属性值	寄存器映射
CHn_TX_DATA_WIDTH	{CHn_TXCLKF, CHn_TXCLKH, CHn_TX_GEAR}	8 :{0,1,0} 10:{0,1,0} 16:{0,1,1} 20:{0,1,1}	{CH_19[4], CH_19[3], CH_02[2]}
CHn_RX_DATA_WIDTH	{CHn_RXCLKF, CHn_RXCLKH, CHn_RX_GEAR}	8 :{0,0,0} 10:{0,0,0} 16:{1,1,1} 20:{1,1,1}	{CH_19[2], CH_19[1], CH_02[3]}
CHn_TX_FIFO		DISABLED:{1} ENABLED :{0}	{CH_03[1]}
CHn_RX_FIFO		DISABLED:{1} ENABLED :{0}	{CH_04[5]}
PLL_SRC	{TXREFCK_NQ_SEL, TXREFCK_SEL}	REFCLK_EXT :{0,0} REFCLK_CORE:{0,1} REFCLK_NQ :{1,0}[8]	{QD_0B[3], QD_0A[5]}
CHn_CDR_SRC	{RXREFCK_NQ_SEL, CHn_RXREFCK_SEL, CHn_TRAIN_EN, CHn_TRAIN_DIV }	REFCLK_EXT :{0,0,0,0} REFCLK_CORE:{0,1,0,0} REFCLK_NQ :{1,0,0,0}[8] TRAIN_DIV4 :{0,0,1,0}[8] TRAIN_DIV8 :{0,0,1,1}[8]	{QD_0B[4], CH_15[2], CH_15[7], CH_15[6]}
CHn_TDRV[7]	{CHn_TDRV_AMP, CHn_TDRV_DRVCUR_SET, CHn_TDRV_AMP_BOOST}	-4: {110,100,1} -3: {100,101,0} -2: {100,100,0} -1: {101,100,0} 0: {000,100,0} 1: {001,100,0} 2: {011,100,0} 3: {100,000,0} 4: {000,000,0}	{CH_12[7:5], CH_13[2:0], CH_13[3]}
CHn_TDRV （仅适用于 PCI Express 协议）	{CHn_TDRV_AMP, CHn_TDRV_DRVCUR_SET, CHn_TDRV_AMP_BOOST, CHn_TDRV_PRE_EN, CHn_TDRV_PRE_SET}	2: {100,000,0,1,00101}	{CH_12[7:5], CH_13[2:0], CH_13[3], CH_14[4], CH_12[4:0]}
CHn_TX_PRE	{CHn_TDRV_PRE_EN, CHn_TDRV_PRE_SET}	DISABLED:{0,00000} 0:{1,00000} 1:{1,00001} 2:{1,00010} 3:{1,00011} 4:{1,00100} 5:{1,00101} 6:{1,00110} 7:{1,00111}	{CH_14[4], CH_12[4:0]}
CHn_RTERM_TX		50:{10} 75:{11} 5K:{0X}	{CH_14[3:2]}
CHn_RX_EQ	{CHn_REQ_EN, CHn_REQ_LVL_SET, CHn_RATE_SEL}	DISABLED :{0,0,00} MID_LOW :{1,0,10} MID_MED :{1,0,01} MID_HIGH :{1,0,00} LONG_LOW :{1,1,10} LONG_MED :{1,1,01} LONG_HIGH:{1,1,00}	{CH_10[7], CH_10[6], CH_10[4:3]}
CHn_RTERM_RX	{CHn_RX_RTERM}	50 :{01} 60 :{10} 75 :{11} HIGH:{00}	{CH_11[1:0]}

表 8-105. 参考表中的属性 （续）

独立的属性名称	非独立的属性名称	属性值	寄存器映射
CHn_RX_DCC		AC:{0} DC:{1}	{CH_10[5]}
CHn_LOS_THRESHOLD_LO[1]	{CHn_RLOS_E}	0:{0000} 1:{0001} 2:{0010} 3:{0011} 4:{0100} 5:{0101} 6:{0110} 7:{0111}	{mc1_ser_ctl_chN[75], CH_16[2:0]}
PLL_TERM		50:{1} 2K:{0}	{QD_0A[3]}
PLL_DCC		AC:{0} DC:{1}	{QD_0A[4]}
PLL_LOL_SET		0:{00} 1:{01} 2:{10} 3:{11}	{QD_0D[4:3]}
CHn_TX_SB	{CHn_TXPOL, CHn_TXSBBYP}	DISABLED:{0,0} ENABLED :{1,0}	{CH_01[1], CH_03[6]}
CHn_RX_SB	{CHn_RXPOL, CHn_RXSBBYP}	DISABLED:{0,0} ENABLED :{1,0}	{CH_01[0], CH_04[1]}
CHn_TX_8B10B		ENABLED :{0} DISABLED:{1}	{CH_03[3]}
CHn_RX_8B10B		ENABLED :{0} DISABLED:{1}	{CH_04[3]}
CHn_COMMA_A		注 2	{QD_0C[0:7], QD_0E[6:7]}
CHn_COMMA_B		注 2	{QD_0D[0:7], QD_0E[4:5]}
CHn_COMMA_M		注 2	{QD_0B[0:7], QD_0E[2:3]}
CHn_RXWA		DISABLED:{1} ENABLED :{0}	{CH_04[2]}
CHn_ILSM		DISABLED:{1} ENABLED :{0}	{CH_04[7]}
CHn_CTC	{CHn_RXRECCLK}	ENABLED :{0,0} DISABLED{1,1}	{CH_19[0] , CH_04[4]}
CHn_CC_MATCH1		注 2	{QD_0A[1:0], CH_06[7:0]}
CHn_CC_MATCH2		注 2	{QD_0A[3:2], CH_07[7:0]}
CHn_CC_MATCH3		注 2	{QD_0A[5:4], CH_08[7:0]}
CHn_CC_MATCH4		注 2	{QD_0A[7:6], QD_09[7:0]}
CHn_CC_MATCH_MODE	{CHn_MATCH_2_EN, CHn_MATCH_4_EN}	1:{0,0} 2:{1,0} 4:{0,1}	{CH_05[4], CH_05[5]}
CHn_CC_MIN_IPG		0:{00} 1:{01} 2:{10} 3:{11}	{CH_05[7:6]}

表 8-105. 参考表中的属性 （续）

独立的属性名称	非独立的属性名称	属性值	寄存器映射
CCHMARK		0 :{0000} 1 :{0001} 2 :{0010} 3 :{0011} 4 :{0100} 5 :{0101} 6 :{0110} 7 :{0111} 8 :{1000} 9 :{1001} 10:{1010} 11:{1011} 12:{1100} 13:{1101} 14:{1110} 15:{1111}	{QD_02[7:4]}
CCLMARK		0 :{0000} 1 :{0001} 2 :{0010} 3 :{0011} 4 :{0100} 5 :{0101} 6 :{0110} 7 :{0111} 8 :{1000} 9 :{1001} 10:{1010} 11:{1011} 12:{1100} 13:{1101} 14:{1110} 15:{1111}	{QD_02[3:0]}
CHn_SSLB		DISABLED :{0000, 00} ENABLED_EQ2T:{0010, 11} ENABLED_T2R :{0001, 00}	{CH_11[7:4], CH_14[6:5]}
CHn_SPLBPORTS[4]	{PFIFO_CLR_SEL, CHn_SB_PFIFO_LP}	DISABLED:{0,0} ENABLED :{1,1}	{QD_03[2], CH_03[5]}
QD_REFCK2CORE		DISABLED:{0} ENABLED :{1}	{QD_0a[1]}

INT_ALL	{PLOLINT,
	PLOLNINT,
	CHn_PCIDETINT,
	CHn_RLOSLINT,
	CHn_RLOSLNINT,
	CHn_RLOLINT,
	CHn_RLOLNINT,
	CHn_LSSYNCINT,
	CHn_LSSYNCNINT,
	CHn_TXFIFOINT,
	CHn_RXFIFOINT,
	CHn_CCORUNINT,
	CHn_CCURUNINT}
		DISABLED:{
		0,0,0,0,0,0,0,0,0,0000
		,0000,0,0,0,0}
		ENABLED:{
		1,1,1,1,1,1,1,1111,111
		1,1,1,1,1}
			{QD_0F[7],
			QD_0F[6],
			CH_17[6],
			CH_17[5],
			CH_17[4],
			CH_17[1],
			CH_17[0],
			QD_09[7:4],
			QD_09[3:0],
			CH_0F[0],
			CH_0F[1],
			CH_0F[2],
			CH_0F[3]}

CHn_LDR	{CHn_LDR_RX_EN, CHn_LDR_TX_SEL}	DISABLED:{0,0} RXTX :{1,1} RXONLY :{1,0} TXONLY :{0,1}	{CH_15[3], CH_13[7]}

表 8-105. 参考表中的属性 （续）

独立的属性名称	非独立的属性名称	属性值	寄存器映射
{PLL_SRC, CHn_CDR_SRC}	{REFCKLOCAL}	注 5	{QD_0A[0]}
{CHn_TX_8B10B, CHn_RX_8B10B}	{BUS8BIT_SEL}	注 6	{QD_0B[6]}
{CHn_RXWA, CHn_ILSM}	{CHn_SIG_DET}	{DISABLED, X}[3]{0} {ENABLED, DISABLED}{0} {ENABLED, ENABLED} {1}	{CH_04[6]}
{CHn_RXWA, CHn_ILSM}	{CHn_C_ALIGN}	{DISABLED, X}2[3] {0} {ENABLED, DISABLED}{0} {ENABLED, ENABLED}{0}	{CH_01[7]}

1. rx_los_low 仅显示检测到数据速率高于 1 Gbps 的信号，最大 CID（Consecutive Identical Digits，连续相同的数字）为 7 位（即，使用 8b10b 发送的最小输入信号的转换密度）。除 PCI Express 和 SDI 以外的所有协议， rx_los_low 仅支持 rlos_lset[2:0] = 2 的默认设置。对于 PCI Express，支持 2 和 3。在 SDI 模式下，推荐使用来自外部的 SDI 电缆均衡器的载波检测输出信号 （/CD）。 rlos_hset 不支持。
2. 用户在配置图形用户界面的“PCS Advanced Setup”中设置的或默认的 10 位符号代码。因为 GUI 中的 10 位符号表示为 LSB 到 MSB，符号表示在表中进行了适当的替换以适用于软件。
3. X = 不关心。
4. 如果使能了任何通道， quad 位将设为 1。
5. 如果 PLL_SRC 和所有使能的 CHn_CDR_SRC 都设为 REFCLK_CORE，那么该位应该为 1，否则该位应设为 0。
6. 如果使能的通道 CHn_TX_8B10B 或 CHn_RX_8B10B 设为 DISABLED，那么该位应该为 1 否则该位应该设为 0。
7. TDRV_AMP 属性已经被 TDRV 代替。当打开 .lpc 文件，软件将自动用 TDRV 代替 TDRV_AMP 属性。如果用户试图在不重新生成 PCS 模块的情况下进行重新编译，软件将在 automake.log 文件中记录错误。用户应该编辑 .txt 文件或使用 IPexpress 重新生成 PCS 模块。
8. refclk_to/from_nq 信号仅供内部使用。

表 8-106. 特定协议 SERDES 的设置选项

协议	数据率	数据率范围	REFCK 乘法器	数据宽度	RX 均衡器[1]
GbE	1.25	MED	10x, 20x, 25x	8, 16	DISABLE, MID_MED, LONG_MED
SGMII	1.25	MED	10x, 20x, 25x	8	
PCI Express	2.5	HIGH	25x, 20x	8, 16	MID_LOW, DISABLE, MID_HIGH LONG_HIGH
XAUI	3.125	HIGH	20x, 10X	16	
G8B8B	ANY_VALUE	LOWLOW, LOW MEDLOW, MED, MEDHIGH, HIGH	10x, 20x, 25x	8, 16	DISABLE, MID_LOW, MID_MED, MID_HIGH, LONG_LOW, LONG_MED, LONG_HIGH
8 位 SERDES only			8x, 16x	8, 16	
10 位 SERDES only			10x, 20x, 25x	10, 20	
USER_DEF			8x, 10x, 16x, 20x, 25x	8, 10, 16, 20	

1.MID：长度大约 20"
LONG：长度大约 40"
LOW：小于 1.2 Gbps
MED：1.2 Gbps 到 2 Gbps 之间
HIGH：大于 2 Gbps

附录 D. Lattice Diamond 使用概述

本附录讨论了在包括 LatticeECP2M SERDES/PCS 模块的项目中使用 Lattice Diamond 设计软件。

有关 Lattice Diamond 使用的一般信息，请参阅 Lattice Diamond 教程。

如果你一直在使用 ispLEVER 软件进行 FPGA 项目设计， Lattice Diamond 可能看起来变化很大。但如果你仔细观察，你会发现有许多相似之处，因为 Lattice Diamond 与 ispLEVER 是基于相同的工具集和工作流程基础上的。所作的改变是为了提供一个更简单、更集成、增强的用户界面。

将 ispLEVER 项目转换到 Lattice Diamond 中

ispLEVER 中创建的设计项目可以很方便地导入到 Lattice Diamond。这个过程是自动的，除了 ispLEVER 进程属性，这与 Diamond 策略设置相同，以及 PCS 模块。导入项目后，你需要为其设置一个策略，并重新生成所有 PCS 模块。

导入 ispLEVER 设计项目

保存一个 ispLEVER 项目副本备份，或新复制一个作为 Diamond 项目。

1. 在 Diamond 中，选择 **File > Open > Import ispLEVER Project**。
2. 在 ispLEVER Project 对话框中，找到项目的 .syn 文件并打开它。
3. 如果需要，为 Diamond 项目更改基本文件名或位置。如果你改变了文件位置，新的 Diamond 文件将放到新的位置，但是原来的源文件不会被删除或者拷贝。 Diamond 项目将参考原始位置的源文件。

项目文件按默认的策略设置转换为 Diamond 格式。

调整 PCS 模块

IPexpress 创建的 PCS 使用一种不常用的文件结构，从 ispLEVER 导入项目时，需要进行额外的调整。可以使用两种方法进行调整。首选的方法是在 Diamond 中重新生成模块。然而，这可能将模块升级到更新的版本。升级通常是可取的，但如果由于某种原因，你不希望升级 PCS 模块，可以将 .txt 文件复制到实现文件夹来进行手动调整。如果你使用此方法，请务必记住将 .txt 文件复制到所有以后使用的实现文件夹。

重新生成 PCS 模块

1. 在 File List 视图的 Input Files 文件夹中找到 PCS 模块。该模块文件扩展名为 .lpc、 .v 或 .vhd 文件。
2. 如果 File List 视图显示模块的 Verilog 或 VHDL 文件，并且你要重新生成 模块，导入模块的 .lpc 文件。
   a. 在 File List 视图，右击实现文件夹 () 并选择 **Add > Existing File**。
   b. 查看模块的 .lpc 文件， ***<module_name>*.lpc**，并且选择它。
   c. 点击 **Add**。 .lpc 文件添加到 File List 视图。
   d. 右击模块的 Verilog 或 VHDL 文件并选择 **Remove**。
3. 在 File List 中，双击模块的 .lpc 文件。该模块的 IPexpress 对话框打开。
4. 在对话框底部，点击 **Generate**。显示了 Generate Log 选项卡。检查错误并关闭。

在 File List 中，使用 .ipx 文件替代 .lpc 文件。IPexpress manifest (.ipx) 文件新用于 Diamond。.ipx 文件保持追踪复杂模块所需的文件。

使用 Lattice Diamond 中的 IPexpress

使用 Lattice Diamond 中的 IPexpress 与使用 ispLEVER 基本相同。

配置图形用户界面选项卡都是一样的，除了 Generation Options 选项卡。图 8-57 显示了 Generation Options 选项卡窗口。

图 8-57. Generation Options 选项卡

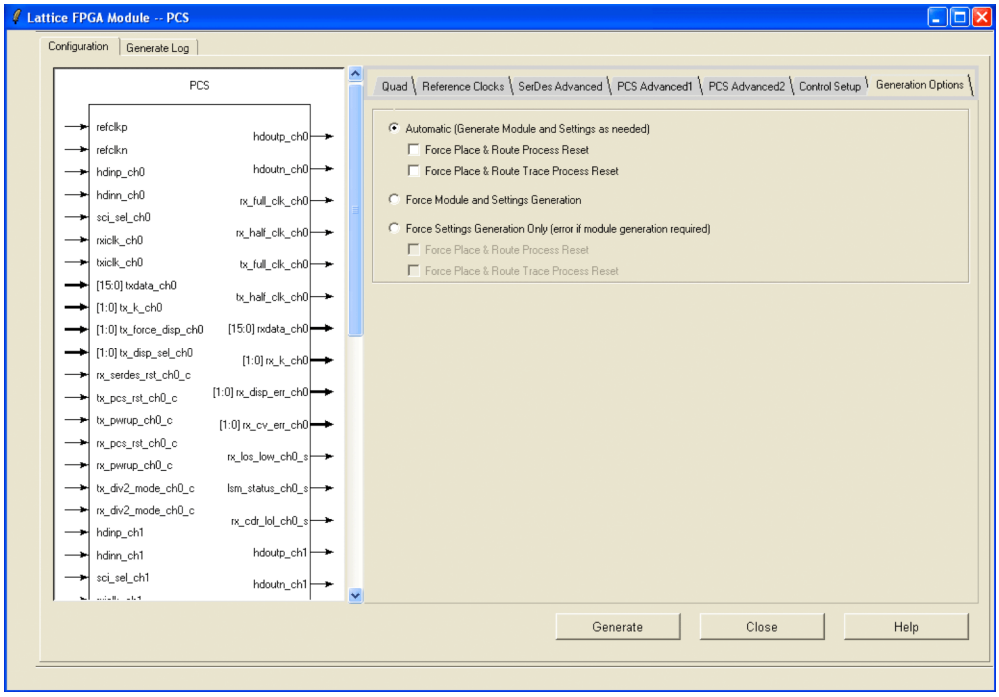


表 8-107. SERDES_PCS 图形用户界面参数——Generation Options 选项卡

图形用户界面显示	说明
Automatic	自动生成所需的 HDL 和配置 （.txt）文件。有些改变不需要重新生成这两个文件。
Force Module and Settings Generation	生成 HDL 和配置文件。
Force Settings Generation Only	仅生成属性文件。如果还需要生成 HDL 文件，你会得到一个错误信息。
Force Place & Route Process Reset	复位 Place & Route Design 进程，迫使它使用新生成的 PCS 模块再运行一遍。
Force Place & Route Trace Process Reset	复位 Place & Route Trace 进程，迫使它使用新生成的 PCS 模块再运行一遍。

注意：
默认选项设置为 Automatic。如果选择 Automatic 或 Force Settings Generation Only，并且不选择子选项 （Process Reset Options），则不会生成 HDL 模块，复位指针设置为自动生成位流。
生成结束后，在这个流程窗口中的复位标志将被相应复位。

使用仿真向导创建一个新的仿真项目

本节介绍如何使用仿真向导来创建一个仿真项目 （.spf）文件，因此你可以将它导入到一个独立的仿真器。

1. 在 Project Navigator 中，点击 **Tools > Simulation Wizard**。打开仿真向导。
2. 在仿真器界面准备页面，点击 **Next**。
3. 在 Simulator Project Name 页面中，在 Project Name 文本框中输入你的项目名称并浏览文件的路径位置，利用 Project Location 文本框和 Browse 按钮找到你希望放置仿真项目的位置。

   当你在此向导页中指定了一个项目名称，将会在你选择的文件路径创建相应的文件夹。弹出的对话框询问你是否希望创建一个新的文件夹，点击 **Yes**。
4. 点击 Active-HDL® 或 ModelSim® simulator 仿真器复选框并点击 **Next**。
5. 在 Process Stage 页，选择您要创建哪种类型的仿真项目的 Process Stage。有效的类型有 RTL、 Post-Synthesis Gate-Level、Post-Map Gate-Level 和 Post-Route Gate-level+Timing。只有这些进程阶段可被激活。

   请注意，如果你的项目中定义超过不止一个策略，你可以重新选择一个作为当前的策略。

   软件支持每个项目实现有多个的策略，让您体验到使用一组共同的源文件来获得优化的设计选择。由于每个策略可能已被处理用于不同阶段，此对话框允许您指定您要载入哪一个阶段。
6. 在 Add Source 页上，在 Source Files 列表框中选择源文件，或使用右边的查看按钮，选择其他所需的源文件。请注意，如果你想将源文件保留在您刚创建的本地仿真项目目录中，请勾选 **Copy Source to Simulation Directory** 选项。
7. 单击 **Next**，出现一个 Summary 页面，提供了项目选择的信息，包括仿真库。默认情况下，勾选了 Run Simulator 复选框，将打开你之前在 Simulator Project Name 页中选择的仿真工具向导。
8. 点击 **Finish**。

运行向导后产生 Simulation Wizard Project (.spf) 文件和一个仿真脚本 DO 文件。如果需要的话，你可以导入 DO 文件到当前项目。如果你使用 Active- HDL，向导将生成一个 .ado 文件，并且如果你正在使用 ModelSim，它还会创建 .mdo 文件。

注：PCS 配置文件 （.txt）必须在第 6 步添加。