# SALPHOENIX®1A 系列 FPGA
# SERDES 用户手册

上海安路信息科技股份有限公司

UG_909（v1.5）2023 年 08 月

# 目 录

# 1 SERDES 简介

本手册主要介绍 SALPHOENIX®1A（以下简称 PH1A）系列 FPGA SERDES 串行收发器功能，PH1A 系列器件包含 PH1A400、PH1A90、PH1A180、PH1A60 器件。其中 PH1A60 器件不包含 SERDES DUAL。

PH1A400 器件有 SFG900 和 SFG676 两种封装，资源示意图分别如图 1-1、图 1-2 所示，SFG900 封装提供 8 个 SERDES DUAL 共 16 路 SERDES 收发器，位置编号分别是 x133y120z0、x133y159z0、x133y160z0、x133y199z0、x133y200z0、x133y239z0、x133y240z0 和 x133y279z0；SFG676 封装器件提供 4 个 SERDES DUAL 共 8 路 SERDES 收发器，位置编号分别是 x133y120z0、x133y159z0、x133y160z0 和 x133y199z0。

PH1A90 器件有 SBG484、SEG324 和 SEG325 三种封装，SBG484 封装和 SEG325 均提供 2 个 SERDES DUAL 共 4 路 SERDES 收发器，位置编号分别是 x71y120z0、x71y159z0，资源示意图如图 1-3 所示；SEG324 封装提供 4 个 SERDES DUAL 共 8 路 SERDES 收发器，位置编号分别是 x71y80z0、x71y119z0、x71y120z0、x71y159z0，资源示意图如图 1-4 所示。

PH1A180 器件为 SFG676 封装，提供 4 个 SERDES DUAL 共 8 路 SERDES 收发器，具体资源示意如图 1-5 所示，SERDES DUAL 位置编号分别是 x103y120z0、x103y159z0、x103y160z0、x133y199z0。

PH1A400、PH1A90 的 Bank 82 和 Bank 83 与 PH1A180 的 Bank 80 和 Bank 81 对应的 SERDES DUAL 均可以用于 PCIe 硬核的 PHY，当 PCIe 硬核不使用时它们也可以当做普通 SERDES 使用。

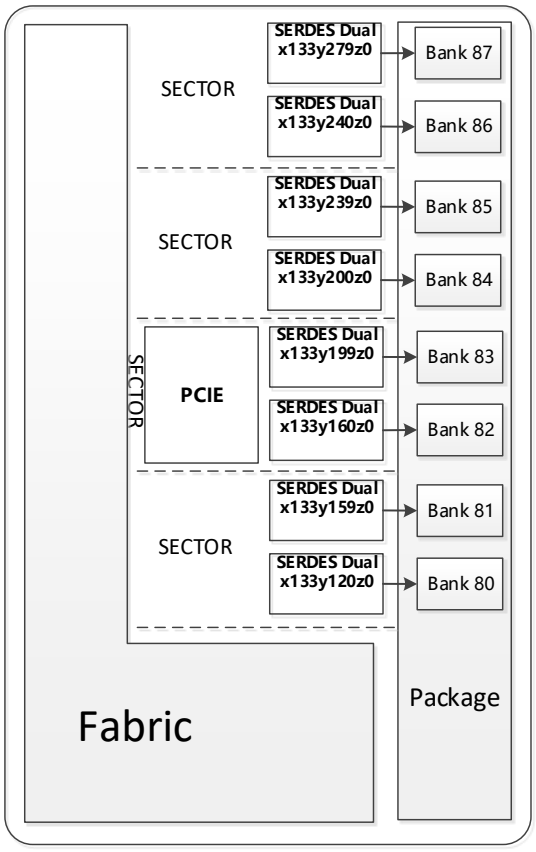


图 1-1 PH1A400SFG900 器件中的 SERDES 资源示意图

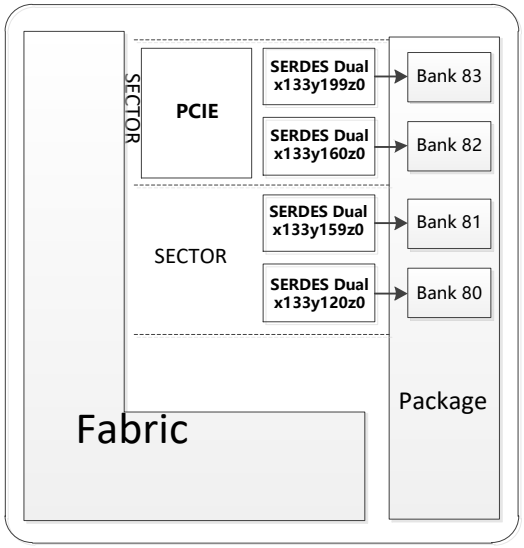


图 1-2 PH1A400SFG676 器件中的 SERDES 资源示意图

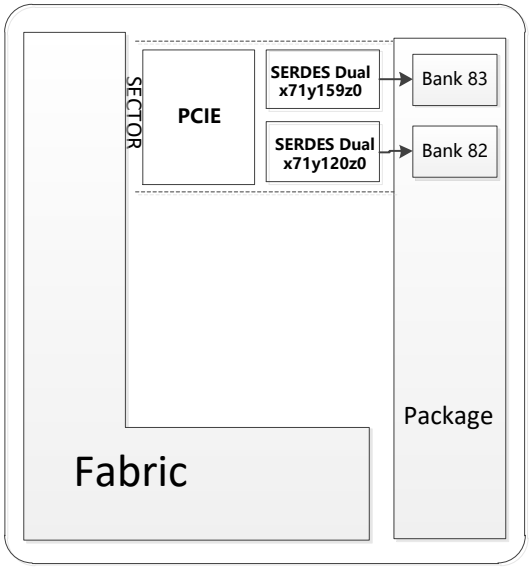


图 1-3 PH1A90SBG484 和 PH1A90SEG325 器件中的 SERDES 资源示意图

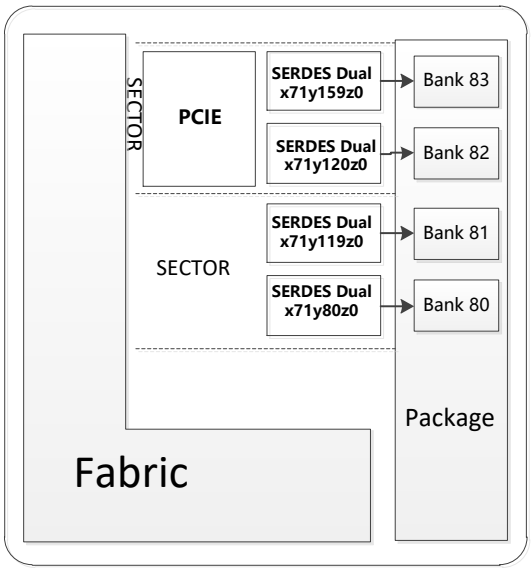


图 1-4 PH1A90SEG324 器件中的 SERDES 资源示意图

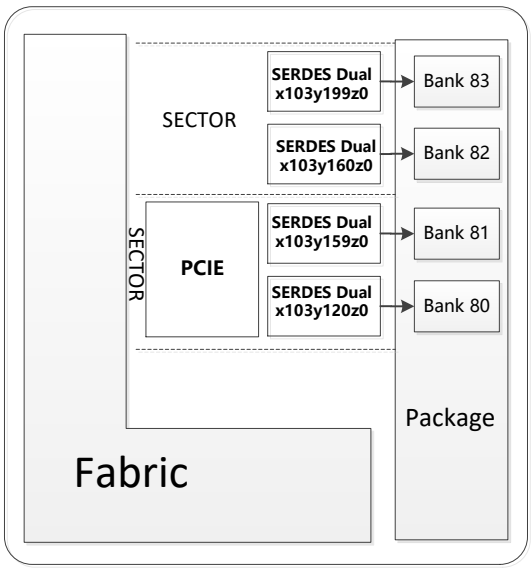


图 1-5 PH1A180SFG676 器件中的 SERDES 资源示意图

## 1.1 SERDES 支持的线速范围

- 1.200-1.600 Gbps
- 1.867-2.133 Gbps
- 2.400-3.200 Gbps
- 3.734-4.266 Gbps
- 4.800-6.400 Gbps
- 7.467-8.533 Gbps

- 9.600-12.500 Gbps

不同器件对应线速率上限不同，具体线速率支持请参考《DS900_PH1A_Datasheet》。

## 1.2 SERDES 支持的协议

SERDES 支持的协议列表如表 1-1 所示。

表 1-1 SERDES 支持的协议

| 高速协议 | 支持 |
|---|---|
| PCI Express | Gen1/Gen2/Gen3, X1/X2/X4 |
| 1000BASE-KX | Y |
| SGMII | Y |
| QSGMII | Y |
| XAUI | Y |
| RXAUI | Y |
| 10GBASE-KX4 | Y |
| 10GBASE-KR | Y |
| CEI | 6/11G |
| CPRI | 2.4576、4.9152、6.144、9.8304 |
| JESD204B | 1.25、2.5、5.0、6.25、12.5 |
| SRIO | 1.25/2.5/3.125/5.0/6.25 |

## 1.3 SERDES 内部结构

### 1.3.1 SERDES DUAL 内部结构简图

PH1A 系列 FPGA 的 SERDES DUAL 包含两条 Lane，每条 Lane 都有发送器和接收器。SERDES DUAL 内部分为 SERDES Common 以及 2 个 SERDES Channel，即 Channel 0 和 Channel 1，如图 1-6 所示。

SERDES Common 是 Channel 0 和 Channel 1 共用的电路，主要包括参考时钟相关电路，PLL，以及端接电阻校准等功能模块。SERDES Channel 0 和 SERDES Channel 1 分别对应 Lane0 和 Lane1 两对收发器。

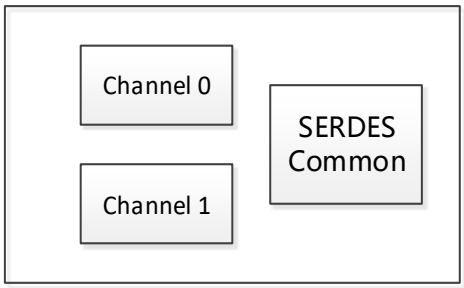


图 1-6 SERDES DUAL 结构简图

注：本文中 Lane 0 表示 Channel 0，Lane 1 表示 Channel 1，使用通配符*表示 SERDES 内部对应的 Lane 编码，可以为 0 或者 1，当*为 0 时表示 Lane0，当*为 1 时为 Lane1。

## 1.3.2 SERDES Channel 结构详图

SERDES Channel 分为 TX PMA、RX PMA、TX PCS 和 RX PCS 四个部分。TX PMA 和 TX PCS 组成 TX Channel；RX PMA 和 RX PCS 组成 RX Channel，如图 1-7 所示。

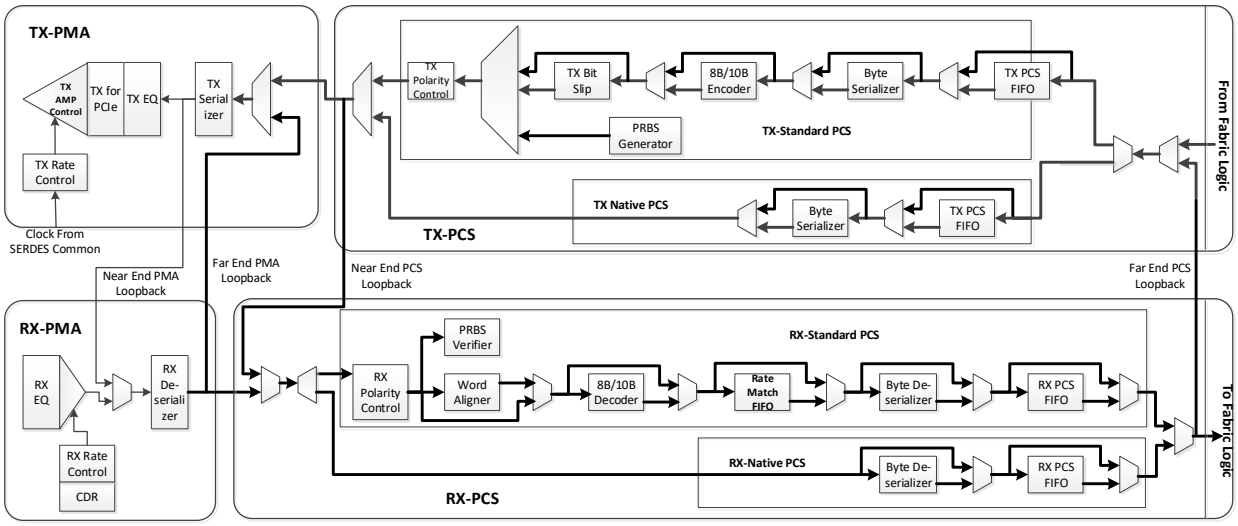


图 1-7 SERDES Channel 结构详图

本文档的以下章节，将分别介绍 SERDES Common、TX Channel 和 RX Channel。

# 2 SERDES Common

本章介绍 TX 和 RX 共用的功能模块（SERDES Common）。它主要包括参考时钟，锁相环（PLL），端接电阻校准，以及 TX 和 RX 都需要的上电初始化流程和复位流程，环回模式，CRI 接口，以及 SERDES 相关的外部管脚介绍。

图 2-1 表示在收发共用部分 SERDES Common 模块中所包含的主要功能模块。

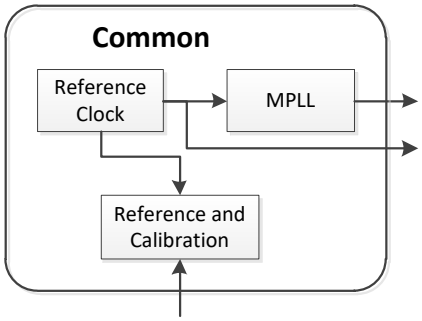


图 2-1 SERDES Common 内部框图

下文分别介绍 SERDES Common 中各个功能模块。

## 2.1 参考时钟

### 2.1.1 功能简介

参考时钟支持外部差分时钟输入，每个 SERDES DUAL 对应一个参考时钟差分对。参考时钟内部结构如图 2-2 所示。

参考时钟内部无端接电阻，P 和 N 两端之间呈现高阻状态，因此需在外部进行端接来匹配阻抗，提高信号完整性，SERDES 参考时钟输入端接方法参见 UG907。

参考时钟进入 PLL 之前经过一个可选的 2 分频电路，该电路由 REF_CLK_DIV2_EN 使能。

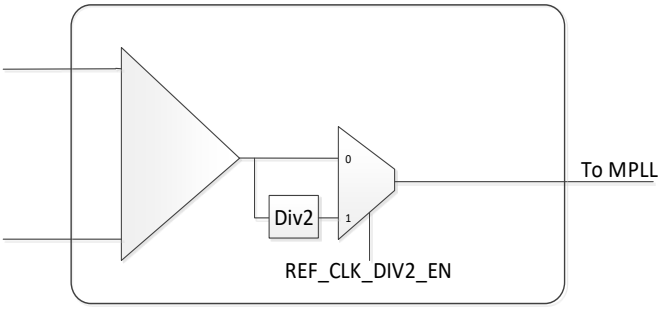


图 2-2 参考时钟内部结构图

### 2.1.2 端口和属性

参考时钟相关的端口如表 2-1 所示。参考时钟相关属性如表 2-2 所示。

表 2-1 参考时钟相关端口

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| ref_clk_en | In | Async | 参考时钟使能信号。高有效。在参考时钟稳定后将该信号拉高。 |
| ref_clk_req | Out | Async | SERDES 请求外部逻辑提供参考时钟的指示信号。保留。当这个信号为高的时候，参考时钟不能停止。 |
| ref_clkdet_result | Out | Async | 参考时钟探测结果。表示参考时钟输入已经输入到 PAD。高有效。 |
| clk0_sel[1:0] | In | Async | clk0_out 时钟源选择信号，保留，接 2’b00。 |
| clk0_out | Out | Clock | clk0_sel 所选时钟输出信号，用于调试，保留。 |
| clk1_sel[1:0] | In | Async | clk1_out 时钟源选择信号，保留，接 2’b00。 |
| clk1_out | Out | Clock | clk1_sel 所选时钟输出信号，用于调试，保留。 |
| clk2_sel[1:0] | In | Async | clk2_out 时钟源选择信号，保留，接 2’b00。 |
| clk2_out | Out | Clock | clk2_sel 所选时钟输出信号，用于调试。当 clk2_sel=2’b00 时，该端口可输出参考时钟的同频时钟。具体用法详见“参考时钟单端信号送给 Fabric 时钟 BUFFER 的用法”这一章节。 |
| phy_ref_dig_clk | Out | Clock | 参考时钟经过 BUFFER 之后的单端形式。该时钟可以来自于 ref_pad_clk_p/ref_pad_clk_m，也可以来自于 ref_alt_clk_p/ref_alt_clk_m 或者 BUFG，取决于 REF_USE_PAD 和其他相关参数的设置，详见“参考时钟共享用法”和“用 BUFG 输出时钟作为 SERDES 参考时钟”这两个章节。该时钟的频率取决于 REF_CLK_DIV2_EN 和 REF_RANGE[2]。当 REF_CLK_DIV2_EN 为 1 时，该端口输出频率为参考时钟的二分频；当 REF_RANGE[2]为 1 时，该时钟频率又被二分频。该时钟的频率不能高于 100MHz。该时钟在 phy_reset 拉高时仍然可以输出，但 ref_clk_en 拉低时，该时钟停止。 |

表 2-2 参考时钟相关属性

| 属性名 | 类型和宽度 | 说明 |
|---|---|---|
| REF_USE_PAD | 1 | 选择参考时钟由外部管脚提供还是内部提供。接 1 使用外部管脚做参考时钟。接 0 使用内部提供的参考时钟。默认值为 0。推荐使用 IP Generator 产生的值。 |

| 属性名 | 类型和宽度 | 说明 |
| --- | --- | --- |
| REF_RANGE | 3 | 设置参考时钟经过可选的 2 分频器（由 REF_CLK_DIV2_EN 控制）之后的频率范围。<br>下面是编码映射关系：<br>• 3'b000：20 – 26 MHz<br>• 3'b001：26.1 – 52 MHz<br>• 3'b010：52.1 – 78 MHz<br>• 3'b011：78.1 – 104 MHz<br>• 3'b100：104.1 – 130 MHz<br>• 3'b101：130.1 – 156 MHz<br>• 3'b110：156.1 – 182 MHz<br>• 3'b111：182.1 – 200 MHz<br>**注释：**<br>这个属性如果发生变化，必须用 phy_reset 重新复位 SERDES。默认值 3'b101，推荐使用 IP Generator 产生的值。 |
| REF_REPEAT_CLK_EN | 1 | 使能参考时钟复制功能，用于参考时钟共享。参考时钟共享功能具体用法，见本文 2.1.3 节。默认值为 0，推荐使用 IP Generator 产生的值。 |
| REF_CLK_UP_DOWN | 1 | 参考时钟共享时参考时钟来源选择，合法设置包括“UP”和“DOWN”，当参考时钟共享自本 SERDES DUAL 位置上方时，设置为“UP”，从本 SERDES DUAL 位置下方来时，设置为“DOWN”，当 REF_USE_PAD 为 0 时该属性有效。默认值设置为“UP”，推荐使用 IP Generator 产生的值。该属性在 PH1A90、PH1A180 器件生效。 |
| REF_CLK_DIV2_EN | 1 | 参考时钟 2 分频使能。当拉高时，参考时钟被 2 分频再送到 PLL。这个属性如果发生变化，必须用 phy_reset 重新复位 SERDES。默认值为 0，推荐使用 IP Generator 产生的值。 |
| REF_CLKDET_EN | 1 | 使能参考时钟探测电路。高有效。默认值为 1，推荐使用 IP Generator 产生的值。 |
| REF_CLK_SEL | String | 合法设置包括“DISABLE”和“ENABLE”，当使用 BUFG 驱动参考时钟时，设置为“ENABLE”；其他情况下设置为“DISABLE”。默认值“DISABLE”，推荐使用 IP Generator 产生的值。 |
| REF_CLK_CS_SEL | 3 | 默认值 3’b011，保留，推荐使用 IP Generator 产生的值。 |
| REF_CLK_MUX_EN | 1 | 默认值 1’b0。当使用 BUFG 驱动参考时钟时，设置为 1’b1，其他情况下设置为 1’b0，推荐使用 IP Generator 产生的值。 |
| RXCLK_BUF_EN | 1 | 默认值 1’b0，恢复时钟差分对专用输出使能。具体用法参见 |

| 属性名 | 类型和宽度 | 说明 |
|---|---|---|
| | | 2.7.3 章节。推荐使用 IP Generator 产生的值。 |
| RXCLK_BUF_SEL | 1 | 默认值 1’b0，选择 Lane0 或者 Lane1 的恢复时钟输出，推荐使用 IP Generator 产生的值。 |
| RXCLK_BUF_SRC | 3 | 默认值 3’b111，保留，推荐使用 IP Generator 产生的值。 |
| RXCLK_BUF_CS | 3 | 默认值 3’b011，保留，推荐使用 IP Generator 产生的值。 |

## 2.1.3 用法

参考时钟可以从专用差分输入管脚输入，也可以从相邻 DUAL 借用，还可以从 BUFG 输出的时钟接入，参考时钟源的选择由时钟 MUX 来实现。下文将介绍参考时钟从相邻 SERDES DUAL 共享的用法，以及 BUFG 提供参考时钟的使用方法。

### 2.1.3.1. PH1A 器件参考时钟共用

对于 PH1A 系列器件来说，当应用需要 4 个 LANE 的协议时（如 XAUI），Bank N（N 代表 Bank 的编号，PH1A400SFG900 器件 N 范围是从 80 到 87 之间，PH1A400SFG676 器件 N 范围是从 80 到 83 之间、PH1A90SBG484、PH1A90SEG325 器件 N 范围是从 82 到 83 之间、PH1A90SEG324 器件 N 范围是从 80 到 83 之间、PH1A180SFG676 器件 N 范围是从 80 到 83 之间）和 Bank N+1 可以绑定在一起，形成一个 X4 的 Link。

PH1A400 仅支持单向参考时钟级联，Bank N 接外部参考时钟时，Bank N+1 无需接参考时钟，它可以借用 Bank N 的参考时钟，参考时钟级联只能从 Bank N 级联给 Bank N+1，即 Bank N+1 可以借用 Bank N 的参考时钟，但 Bank N 不能借用 Bank N+1 的参考时钟。

PH1A90、PH1A180 支持双向参考时钟级联，Bank N 接外部参考时钟时，Bank N+1 无需接参考时钟或者 Bank N+1 接外部参考时钟时，Bank N 无需接参考时钟，即 Bank N+1 可以借用 Bank N 的参考时钟，Bank N 也可以借用 Bank N+1 的参考时钟，参考时钟级联方向由 REF_CLK_UP_DOWN 参数选择。

当 Bank N 和 Bank N+1 分别由各自的参考时钟管脚来提供时钟源时，参考时钟分配如图 2-3 所示。

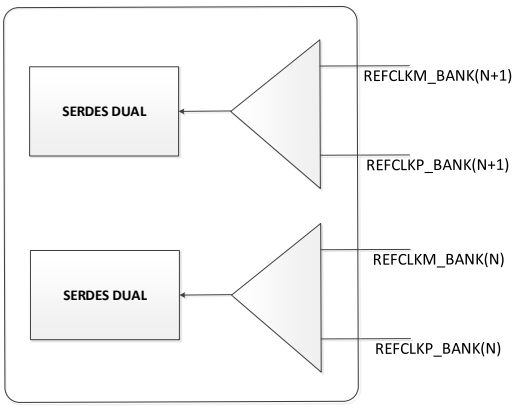


图 2-3 PH1A 参考时钟分别各自提供的示意图

当 Bank N 和 Bank N+1 共用一个参考时钟差分对时，参考时钟连接方式如图 2-4 所示。其中虚线部分在 PH1A90、PH1A180 支持，PH1A400 不支持。

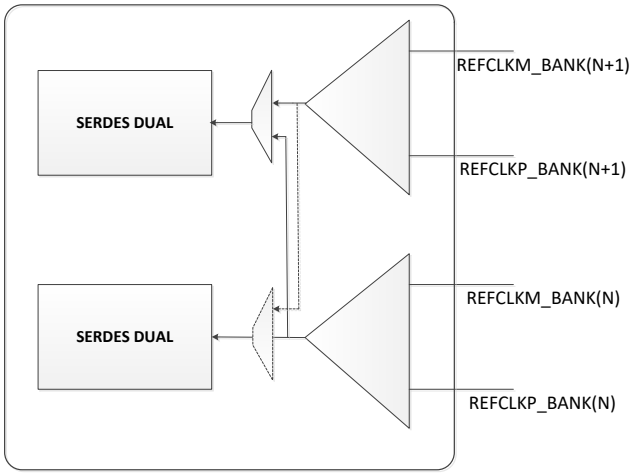


图 2-4　PH1A 参考时钟共享示意图

对于 PH1A 器件，Bank N 的 SERDES DUAL 和 Bank N+1 对应的 SERDES DUAL 可以共用参考时钟，形成一个 X4 的状态。使用专用参考时钟管脚的 SERDES DUAL，通过 ref_repeat_clk_o_p/m 输出时钟给相邻 SERDES DUAL 的 ref_alt_clk_p/m 输入。该走线为“硬”连接，未使用普通 fabric 走线。该走线由软件自动控制，用户无需在 RTL 代码中手工连接；若需要进行仿真，用户需在 test bench 中进行手工连接。

**PH1A 器件 Bank N 使用专用差分时钟输入 Bank N+1 共享 Bank N 的时钟时，分别设置如下：**

**使用专用差分时钟输入的 Bank N 相关参数设置如下：**

将参数 REF_REPEAT_CLK_EN 设置为 1'b1,使能 ref_repeat_clk_o_p/m 输出，输出的时钟频率和专用参考时钟管脚输入的时钟频率相等；

将参数 REF_USE_PAD 设置为 1'b1，选择使用专用差分时钟输入 ref_pad_clk_p/m；

将参数 REF_CLK_SEL 设置为"DISABLE"；

将参数 REF_CLK_CS_SEL 设置为 3'b011；

将参数 REF_CLK_MUX_EN 设置为 1'b0，关闭参考时钟 MUX。

**借用相邻参考时钟的 Bank N+1 相关参数设置如下：**

将参数 REF_REPEAT_CLK_EN 设置为 1'b0；

将参数 REF_USE_PAD 设置为 1'b0，选择不使用 ref_pad_clk_p/m；

将参数 REF_CLK_SEL 设置为"DISABLE" ，选择相邻的 repeat clk 作为参考时钟输入；

将参数 REF_CLK_CS_SEL 设置为 3'b011；

将参数 REF_CLK_MUX_EN 设置为 1'b1，使能参考时钟 MUX。

对于 PH1A90、PH1A180 器件需要额外设置 REF_CLK_UP_DOWN，将参数 REF_CLK_UP_DOWN 设置为

“DOWN”，选择 Bank N+1 下方的 Bank N 共享参考时钟。

**PH1A90、PH1A180 Bank N+1 使用专用差分时钟输入，Bank N 共享 Bank N+1 的时钟时，其分别设置如下：**

**使用专用差分时钟输入的 Bank N+1 相关参数设置如下：**

将参数 REF_REPEAT_CLK_EN 设置为 1'b1,使能 ref_repeat_clk_o_p/m 输出，输出的时钟频率和专用参考时钟管脚输入的时钟频率相等；

将参数 REF_USE_PAD 设置为 1'b1，选择使用专用差分时钟输入 ref_pad_clk_p/m；

将参数 REF_CLK_SEL 设置为"DISABLE"；

将参数 REF_CLK_CS_SEL 设置为 3'b011；

将参数 REF_CLK_MUX_EN 设置为 1'b0，关闭参考时钟 MUX。

**借用相邻参考时钟的 Bank N 相关参数设置如下：**

将参数 REF_REPEAT_CLK_EN 设置为 1'b0；

将参数 REF_USE_PAD 设置为 1'b0，选择不使用 ref_pad_clk_p/m；

将参数 REF_CLK_SEL 设置为"DISABLE" ，选择相邻的 repeat clk 作为参考时钟输入；

将参数 REF_CLK_CS_SEL 设置为 3'b011；

将参数 REF_CLK_MUX_EN 设置为 1'b1，使能参考时钟 MUX。

将参数 REF_CLK_UP_DOWN 设置为“UP”，选择 Bank N 上方的 Bank N+1 共享参考时钟。

#### 2.1.3.2. 用 BUFG 输出时钟作为 SERDES 参考时钟

安路建议用户使用专用参考时钟管脚输入，或者借用相邻 SERDES DUAL 的专用参考时钟输入，作为 SERDES 参考时钟。不推荐使用 BUFG 来驱动 SERDES 参考时钟。这是因为 BUFG 输出的时钟有很大的抖动（Jitter），会影响链路裕量（Link Margin）。BUFG 输出驱动 SERDES 参考时钟仅作为测试或 Debug 来使用。

使用 BUFG 输出时钟驱动 SERDES 参考时钟具体使用方法为：

BUFG 输出信号接 ref_alt_clk_p 输入端口；

ref_alt_clk_m 输入端口悬空；

将参数 REF_CLK_SEL 设置为“ENABLE”；

将参数 REF_CLK_MUX_EN 设置为 1'b1；

将参数 REF_USE_PAD 设置为 1'b0。

#### 2.1.3.3. 参考时钟单端输出信号送给 Fabric 用户逻辑的用法

在一些应用中，需要将参考时钟未经 SERDES 内部 PLL 倍频和分频，直接送入 FPGA Fabric 时钟走线。该时钟的特点是不会受到 SERDES 复位的影响，可以认为是一个自由时钟（free running clock）。安路 SERDES 可以支持参考时钟从差分信号转为单端信号以后，送入 Fabric 时钟走线，用于给用户逻辑提供时钟源。具体用法为，将 clk2_sel[1:0] 设置为 2’b00，clk2_out 输出的单端时钟频率为参考时钟频率。该时钟不会受到 phy_reset 的影响，当 phy_reset 为高时，该时钟仍可正常输出。该时钟会受到 ref_clk_en 信号的控制，当 ref_clk_en 信号为 0 时，该时钟输出常 0。参考时钟稳定后，ref_clk_en 需有从低到高的过程，确保该时钟输出稳定。clk2_out 无法自动通过专用走线上全局时钟资源，用户需将 clk2_out 先送入 LCLK，再送到 BUFG，才能上全局时钟走线，驱动相应的用户逻辑。这样会消耗掉一个 LCLK 资源。如果不使用 LCLK，将会使用普通走线上 BUFG。

#### 2.1.3.4. 专用恢复时钟输出管脚的用法

PH1A SERDES DUAL 提供的专用输出恢复时钟管脚功能仅在 PH1A400SFG900 器件 Bank 80 和 Bank 84 中支持，其他器件和 BANK 均不支持。

专用恢复时钟管脚相比于使用普通 IO 输出恢复时钟的优势在于，它 jitter 相对较小，并且受到用户逻辑翻转的影响也比较小。要使能专用恢复时钟管脚输出的功能，用户需将 RXCLK_BUF_EN 设置为 1’b1，同时用 RXCLK_BUF_SEL 指定 SERDES DUAL 中具体的 Lane。RXCLK_BUF_SEL 为 1’b0 时，输出 Lane0 的恢复时钟；RXCLK_BUF_SEL 为 1’b1 时，输出 Lane1 的恢复时钟。

专用恢复时钟管脚输出的频率为 l*_rx_line_rate/ l*_rx_int_datawidth。其中 l*_rx_line_rate 表示线速率，l*_rx_int_datawidth 表示内部 PMA 数据位宽，该位宽由 l*_rx_pmadata_width 端口控制。专用恢复时钟管脚最高输出频率是 350MHz，其电平标准为 CML，建议使用交流耦合送到时钟接收器件。

这些管脚不能作为普通 IO 例化和使用。使用专用恢复时钟输出管脚时，对于一个 SERDES DUAL 的两条 Lane，只能选择其中一条 Lane 的恢复时钟输出。

## 2.2 锁相环 PLL

### 2.2.1 功能简介

TX PLL 内部包含两个 VCO：MPLLA 和 MPLLB，对应有 2 个频段，分别是 4.800-6.400 GHz 和 7.467-8.533 GHz。MPLLA 和 MPLLB 的线速率频段范围参见《DS900_PH1A_Datasheet》。

PLL 的输出时钟经过分频后得到串行发送时钟，该时钟为双沿有效。用户应根据所需线速率使能相应的 PLL。PLL 的选择信号是 tx*_mpllb_sel。SERDES DUAL 中的两条 Lane 都有 tx*_mpllb_sel 信号，分别为 tx0_mpllb_sel 和 tx1_mpllb_sel，用户可以根据需要对两条 Lane 分别选择不同的 PLL，也可以两条 Lane 都选择同一个 PLL。

### 2.2.2 端口和属性

PLL 相关的端口如表 2-3 所示。PLL 相关属性如表 2-4 所示。

表 2-3 PLL 相关端口

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| mplla_bandwidth[15:0] | In | Async | 控制 MPLLA 的带宽。推荐使用 IP Generator 产生的值。保留。 |
| mplla_div10_clk_en | In | Async | 设置 MPLLA 输出的 ‘word_clk’为 VCO 时钟频率的 10 分频。优先级高于 mplla_div8_clk_en。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mplla_div16p5_clk_en | In | Async | 保留。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mplla_div8_clk_en | In | Async | 设置 MPLLA 输出的 ‘word_clk’为 VCO 时钟频率的 8 分频。优先级低于 mplla_div10_clk_en 。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mplla_div_clk_en | In | Async | 使能 mplla_div_clk 输出。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mplla_div_multiplier[6:0] | In | Async | 设置 mplla_div_clk 倍频系数，保留。默认值为 7’b1111111，推荐使用 IP Generator 产生的值。 |
| mplla_force_ack | Out | Async | mplla_force_en 的应答信号。保留。 |
| mplla_force_en | In | Async | 保留。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mplla_fracn_ctrl[10:0] | In | Async | 保留。默认值为 11’b11111111111，推荐使用 IP Generator 产生的值。 |
| mplla_init_cal_disable | In | Async | 保留。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mplla_multiplier[7:0] | In | Async | MPLLA 反馈路径的分频器。也就是 MPLLA 输出的倍频系数。 mplla_multiplier[6:0]的值和实际分频值（即图 2-5 中的 M）的相互关系如表 2-6。mplla_multiplier[7]在反馈路径上又提供了一个额外的二分频器。当 mplla_multiplier[7]为 1 时，使能该二分频 |

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| | | | 器。推荐使用 IP Generator 产生的值。<br>MPLLA 输入的时钟是，外部输入参考时钟经过以下两个可选分频器之后的时钟。<br>REF_CLK_DIV2_EN 所使能的可选 2 分频电路；<br>REF_CLK_MPLLA_DIV2_EN 所使能的可选 2 分频电路。<br>经过这两个可选的 2 分频电路，在外部参考时钟输入到 PLL 之前，一共有 1，2，4 三种分频选项。<br>这些输入端口只能在 phy_reset 拉高时才能改变。 |
| mplla_recal_bank_sel[1:0] | In | Async | 保留。推荐使用 IP Generator 产生的值。该输入只能在 MPLL 被 power down 的情况下改变。 |
| mplla_ssc_clk_sel | In | Async | 保留。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mplla_ssc_en | In | Async | 使能 SSC，默认值为 0，推荐使用 IP Generator 产生的值。 |
| mplla_ssc_freq_cnt_init[11:0] | In | Async | 保留。默认值为 12’b111111111111，推荐使用 IP Generator 产生的值。这个值只能在 MPLL 处于 power down 的状态下改变。 |
| mplla_ssc_freq_cnt_peak[7:0] | In | Async | 保留。默认值为 8’b11111111，推荐使用 IP Generator 产生的值。这个值只能在 MPLL 处于 power down 的状态下改变。 |
| mplla_ssc_up_spread | In | Async | 保留。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mplla_state | Out | Async | MPLLA 锁定状态指示信号。高电平表示 MPLLA 已被供电且锁定。 |
| mplla_tx_clk_div[1:0] | In | Async | 设置 MPLLA 输出时钟的分频系数，PLL VCO 输出时钟经过这个分频器的时钟提供给发送器 TX。这个值只能在 P1 或 P2 状态下改变。<br>编码映射如下：<br>• 2'b00 ：div1 ；<br>• 2'b01 ：div2 ； |

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| | | | • 2'b10 : div4 ;<br>• 2'b11 : div1 (duplicate state)。<br>默认值为 2’b01，推荐使用 IP Generator 产生的值。 |
| mpllb_bandwidth[15:0] | In | Async | 控制 MPLLB 的带宽。默认值为 16'b0000000001000011，推荐使用 IP Generator 产生的值。 |
| mpllb_div10_clk_en | In | Async | 设置 MPLLB 输出的 ‘word_clk’为 VCO 时钟频率的 10 分频。优先级高于 MPLLB_DIV8_CLK_EN。高有效。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mpllb_div8_clk_en | In | Async | 设置 MPLLB 输出的‘word_clk’为 VCO 时钟频率的 8 分频。优先级低于 MPLLB_DIV10_CLK_EN。高有效。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mpllb_div_clk_en | In | Async | 使能 mpllb_div_clk 输出。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mpllb_div_multiplier[6:0] | In | Async | 设置 mpllb_div_clk 输出的倍频因子。它倍频的是，参考时钟经过可选的 2 分频（ref_clk_div2_en）和 PLL 之前可选的 2 分频(ref_clk_mpllb_div2_en)得到的时钟。这个时钟可能是参考时钟频率的 1，2，4 分频。默认值为 7’b1111111，推荐使用 IP Generator 产生的值。 |
| mpllb_force_ack | Out | Async | mpllb_force_en 的应答信号。保留。 |
| mpllb_force_en | In | Async | 保留。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mpllb_fracn_ctrl[10:0] | In | Async | 保留。默认值为 11’b11111111111，推荐使用 IP Generator 产生的值。 |
| mpllb_init_cal_disable | In | Async | 保留。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mpllb_multiplier[7:0] | In | Async | MPLLB 反馈路径的分频器。也就是 MPLLB 输出的倍频系数。 mpllb_multiplier[6:0]的值和实际分频值（即图 2-5 中的 N）的相互关系如表 2-6。mpllb_multiplier[7]在反馈 |

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| | | | 路径上又提供了一个额外的二分频器。当 mpllb_multiplier[7]为 1 时，使能该二分频器。推荐使用 IP Generator 产生的值。<br>MPLLB 输入的时钟是，外部输入参考时钟经过以下两个可选分频器之后的时钟。<br>REF_CLK_DIV2_EN 所使能的可选 2 分频电路；<br>REF_CLK_MPLLB_DIV2_EN 所使能的可选 2 分频电路。<br>经过这两个可选的 2 分频电路，在外部参考时钟输入到 PLL 之前，一共有 1，2，4 三种分频选项。这些输入端口只能在 phy_reset 拉高时才能改变。 |
| mpllb_recal_bank_sel[1:0] | In | Async | 保留。默认值为 2’b11，推荐使用 IP Generator 产生的值。 |
| mpllb_ssc_clk_sel | In | Async | 保留。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mpllb_ssc_en | In | Async | 保留。默认值为 0，推荐使用 IP Generator 产生的值。 |
| mpllb_ssc_freq_cnt_init[11:0] | In | Async | 保留。推荐使用 IP Generator 产生的值。该属性只能在 MPLL 处于 power down 时改变。 |
| mpllb_ssc_freq_cnt_peak[7:0] | In | Async | 保留。推荐使用 IP Generator 产生的值。该属性只能在 MPLL 处于 power down 时改变。 |
| mpllb_ssc_up_spread | In | Async | 保留。推荐使用 IP Generator 产生的值。该属性只能在 MPLL 处于 power down 时改变。 |
| mpllb_state | Out | Async | MPLLB 锁定状态指示信号。高电平表示 MPLLB 已被供电且锁定。 |
| mpllb_tx_clk_div[1:0] | In | Async | 设置 MPLLB 输出时钟的分频系数，PLL VCO 输出时钟经过这个分频器的时钟提供给发送器 TX。这个属性只能在 P1 和 P2 状态下改变。<br>编码映射关系如下：<br>• 2'b00 ： div1；<br>• 2'b01 ： div2；<br>• 2'b10 ： div4； |

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| | | | • 2'b11 : div1 (duplicate state)。<br>默认值为 2’b01，推荐使用 IP Generator 产生的值。 |
| sup_misc[7:0] | In | Async | 保留，推荐使用 IP Generator 产生的值。 |
| tx*_mpllb_sel | In | Async | 当这个信号为高时，选中 MPLLB 为发送时钟源，否则选中 MPLLA 为发送时钟源。推荐使用 IP Generator 产生的值。 |

表 2-4 PLL 相关属性

| 属性名 | 类型和宽度 | 说明 |
| --- | --- | --- |
| REF_CLK_MPLLA_DIV2_EN | 1 | MPLLA 的参考时钟分频控制，高有效。默认值为 0，推荐使用 IP Generator 产生的值。<br>参考时钟模块在其内部可选的 2 分频之后，再送入到这个可选的 2 分频器，之后才进入 MPLLA。因此，经过这两个可选的 2 分频电路，在外部参考时钟输入到 PLL 之前，一共有 1，2，4 三种分频选项。 |
| REF_CLK_MPLLB_DIV2_EN | 1 | MPLLB 的参考时钟分频控制，高有效。默认值为 0，推荐使用 IP Generator 产生的值。<br>参考时钟模块在其内部可选的 2 分频之后，再送入到这个可选的 2 分频器，之后才进入 MPLLB。因此，经过这两个可选的 2 分频电路，在外部参考时钟输入到 PLL 之前，一共有 1，2，4 三种分频选项。 |

### 2.2.3 用法

PLL 内部框图如图 2-5 所示。PLL 的输入时钟是参考时钟经过 REF_CLK_DIV2_EN 使能的可选分频器，再经过 REF_CLK_MPLL(A,B)_DIV2_EN 所使能的可选分频器，两次分频得到的。VCO 的反馈时钟经过三个分频器，第一个分频器是固定的 2 分频（图 2-5 中反馈路径上的‘div2’），第二个分频器是 Feedback Divider（mplla_multiplier[6:0]或 mpllb_multiplier[6:0]，即图中反馈路径上的 M），第三个分频器是 mplla_multiplier[7]或 mpllb_multiplier[7]提供的一个额外的可选二分频器。

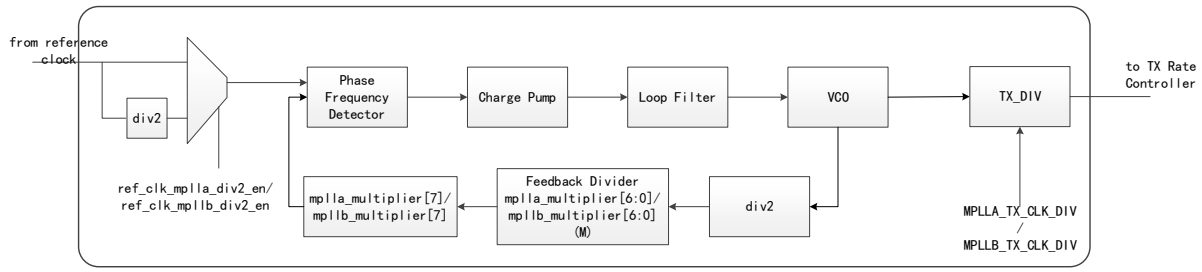


图 2-5 PLL 内部结构框图

MPLL VCO 频率计算公式如下。ref_clk_div 是 REF_CLK_DIV2_EN 使能的 2 分频器。当 REF_CLK_DIV2_EN=0 时，ref_clk_div=1；当 REF_CLK_DIV2_EN=1 时，ref_clk_div=2。ref_clk_mpll_div 是 REF_CLK_MPLL(A,B)_DIV2_EN 所使能的 2 分频器。当 REF_CLK_MPLL(A,B)_DIV2_EN=0 时，ref_clk_mpll_div=1；当 REF_CLK_MPLL(A,B)_DIV2_EN=1 时，ref_clk_mpll_div=2。当 mplla_multiplier[7]或 mpllb_multiplier[7]为 1 时，反馈路径上还有一个额外的二分频，此时 mpll_multiplier[7]_div 为 2；否则，当 mplla_multiplier[7]或 mpllb_multiplier[7]为 0 时，mpll_multiplier[7]_div 为 1。

$$MPLLVCOFREQ = \frac{ref\_clk\_freq \times 2 \times M \times mpll\_mulitplier[7]\_div}{ref\_clk\_div \times ref\_clk\_mpll\_div}$$

MPLL 经过 TX_DIV 分频出来的时钟 mpll_tx_clk_div，其频率按照下面公式来计算。在这个公式中，TX_DIV 是 MPLL(A,B)_TX_CLK_DIV 所控制的分频器。具体分频值见上表 MPLL(A,B)_TX_CLK_DIV 相应介绍。这是提供给各条 TX Channel 的发送时钟。

$$MPLLOUTFREQ = \frac{MPLLVCOFREQ}{mpll\_tx\_clk\_div}$$

TX 线速率（line rate）根据 MPLL 输出时钟被 TX Rate 控制器分频后再乘以 2 得到。因为发送是上升沿和下降沿都可以发出的。下面公式中 tx*_rate_div 是被 tx*_rate 控制的分频器。

$$l*\_tx\_linerate = \frac{MPLLOUTFREQ \times 2}{tx*\_rate\_div}$$

表 2-5 为 mplla_mulitplier[6:0]/ mpllb_mulitplier[6:0]的设置和 M 的对应关系。

表 2-5 mplla/b_mulitplier[6:0]和实际分频值 M 的关系

| mplla/b_mulitplier[6:0] | Feedback Divider M values in Dec |
|---|---|
| 7'h7F – 7’ h08 | mplla_mulitplier[6:0]本身的值（127-8） |
| 7'h07 – 7’ h04 | 非法值，不能使用 |
| 7'h03 – 7’ h00 | 分别对应 131, 130, 129, 128 |

#### 2.2.3.1. PLL 单独复位流程

在安路科技 PH1A 系列 FPGA 中，每个 SERDES DUAL 包含两个 PLL，分别为 MPLLA 和 MPLLB。这两个

PLL 可以分别驱动 SERDES DUAL 中两条 Lane 中的任何一条，任意一个 PLL 也可以同时驱动两条 Lane。在一些应用中，需要对两条 Lane 进行单独复位，包括 PLL 单独复位。通常复位 PLL 有两种方式，一种是使用 phy_reset 信号来触发 PLL 复位，该信号能够复位所有使能了的 MPLL 以及 SERDES DUAL 中其他的电路，是整体复位，通常用于上电加载后的首次复位；另一种是使用 MPLL PowerDown/PowerUp 的方式对 MPLLA 或者 MPLLB 进行单独复位。phy_reset 流程在 2.3 章节中做详细介绍，本节主要介绍 MPLL 单独复位的流程。

MPLL 单独复位的流程相对于 phy_reset 触发的整体复位有两个优点，一是当 MPLLA 和 MPLLB 分别驱动 SERDES DUAL 中的两条 Lane 时，复位其中一个 PLL 不会影响另外一条 Lane；二是单独复位不会像 phy_reset 信号触发的整体复位那样，清空了 CRI 寄存器用户逻辑配置的信息。

单独复位 MPLLA 或者 MPLLB 的流程如图 2-6 所示。单独复位 PLL 必须在 phy_reset 启动的上电复位都完成之后进行，一般是用于需要进行自恢复的场景。

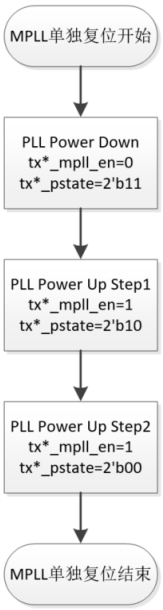


图 2-6 PLL 单独复位流程图

单独复位 MPLL 主要的步骤有，先将 MPLL 设置为 Power Down，即 tx*_mpll_en 设置为 0，并且 tx*_pstate 设置为 P2。之后再将 MPLL Power Up，Power Up 的过程又分为两步：第一步先将 tx*_mpll_en 设置为 1，并且 tx*_pstate 设置为 P1；第二步将 tx*_mpll_en 保持为 1，并且 tx*_pstate 设置为 P0，当 tx*_pstate 进入 P0 状态后，MPLL 单独复位结束。

PLL 单独复位（Power Down、Power Up）相关的具体信号有 tx*_mpll_en, tx*_pstate, tx*_req 和 tx*_ack。其时序图如图 2-7 所示。第一阶段 tx*_pstate 尚处于 P0 状态，此时 MPLL 和 SERDES 在正常工作中；第二阶段 MPLL 单独复位开始，用户逻辑将 tx*_mpll_en 拉低，并将 tx*_pstate 设置为 P2 低功耗模式，拉高 tx*_req 直到 tx*_ack 拉高再拉低 tx*_req，等到 tx*_ack 拉低后 MPLL 处于 Power Down 模式；第三阶段用户逻辑将 tx*_mpll_en 拉高，并将 tx*_pstate 设置为 P1 低功耗模式，拉高 tx*_req 直到 tx*_ack 拉高再拉低 tx*_req，等到 tx*_ack 拉低后 MPLL 处于 Power Up Step1 模式；第四阶段用户逻辑将 tx*_mpll_en 继续拉高，并将 tx*_pstate 设置为 P0 低功耗模式，拉高 tx*_req 直到 tx*_ack

拉高再拉低 tx*_req，等到 tx*_ack 拉低后 MPLL 处于 Power Up Step2 模式，即正常工作模式，此时 MPLL 单独复位过程结束。

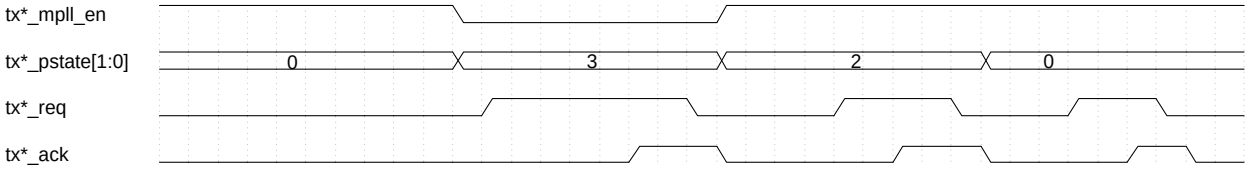


图 2-7 PLL 单独复位时序图

从图 2-7 中可以看出，在 PLL 单独复位过程中 tx*_ack 会拉高拉低多次，另外，tx*_mpllb_sel 信号用于控制 Lane0 和 Lane1 选用 MPLLA 或 MPLLB 来驱动。例如，tx*_mpllb_sel=0，表示 Lane*使用 MPLLA 驱动；tx*_mpllb_sel=1，表示 Lane*使用 MPLLB 驱动。若用户使用了 MPLLA 则 mplla_init_cal_disable 应设置为 0，若使用了 MPLLB 则 mpllb_init_cal_disable 应设置为 0。

## 2.3 初始化和复位

### 2.3.1 功能简介

SERDES 上电后，必须经过初始化才能使用。SERDES 初始化分为上电启动初始化，和启动之后再次复位（warm reset），两种情况。

### 2.3.2 端口和属性

复位相关的管脚如表 2-6 所示。

表 2-6 复位信号列表

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| PMA 相关复位信号 | | | |
| pg_reset | In | Async | Power gated reset. 保留。接低电平。 |
| sram_bypass | In | Async | 当该信号接为高电平时，SRAM 端口被旁路，SERDES 从 FPGA Fabric 预设值开始进行自适应（Adaption）和校准（Calibiration）；当该信号接为低电平时，SRAM 端口被使能，SERDES 先将 FPGA Fabric 预设值装载（Load）到 SRAM 内，用户此时可以修改 SRAM 内容，SERDES 从更新后的值开始进行 Adaption 和 Calibration。修改 SRAM 内容是通过 CRI 接口并由 sram_ext_ld_done 和 sram_init_done 握手来完成。这个功能是专门为 debug 设计的，一般情况下不会使用。sram_bypass 端口 |

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| | | | 的值不能在 phy_reset 为低电平时改变。 |
| sram_sel | In | Async | SRAM 选择信号。当 sram_bypass 为 0 时，SRAM 被使能，此时 sram_sel 必须为 1；当 sram_bypass 为 1 时，SRAM 被旁路，此时 sram_sel 可以接 0 也可以接 1。 |
| phy_reset | In | Async | 整体复位，高有效，电平触发。上电后第一次初始化复位脉冲宽度须大于 20us，初始化之后再次复位时脉冲宽度须大于 20ns。复位 SERDES 内部所有模块。如果参考时钟停止或者失锁后又恢复正常，需要对 phy_reset 进行复位，使 PLL 重新锁定。该信号会将 CRI 接口所能访问的寄存器恢复到加载后的状态。 |
| tx*_reset | In | Asynchronous (must not glitch) | TX 方向 PMA 复位。高有效，电平触发。高电平时间长度大于 20ns。tx0_reset 复位 lane0，tx1_reset 复位 lane1。 |
| rx*_reset | In | Asynchronous (must not glitch) | RX 方向 PMA 复位。高有效，电平触发。高电平时间长度大于 20ns。rx0_reset 复位 lane0，rx1_reset 复位 lane1。 |
| tx*_ack | Out | Async | 在 PMA 的复位信号（ phy_reset /tx*_reset）拉高时，该信号会拉高；在 PMA 复位过程结束时，该信号拉低表示复位过程完成。 |
| rx*_ack | Out | Async | 在 PMA 的复位信号（ phy_reset /rx*_reset）拉高时，该信号会拉高；在 PMA 复位过程结束时，该信号拉低表示复位过程完成。 |
| tx*_data_en | In | Async | TX 数据路径使能信号。高有效。当此信号为 0 时，发送管脚处于空闲状态。 |
| rx*_data_en | In | Async | RX 数据路径使能信号。高有效。当此信号拉高时，CDR 开始跟踪输入数据，这时 CDR 的工作模式为“跟踪模式”。在跟踪模式下，线速率应在 1.1 章节所描述的范围。当此信号拉低时，CDR 工作在“非跟踪模式”，在非跟踪模式下，恢复时钟的频率和本地参考时钟有频偏，并且抖动会增大，因此恢复时钟和发 |

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| | | | 送时钟不是严格同频的。关于 CDR 非跟踪模式的说明详见 4.5.3 节。 |
| rx*_valid | Out | Async | CDR 锁定指示信号。高电平表示 CDR 锁定。 |
| ref_clk_en | In | Async | 参考时钟使能信号。高有效。参考时钟稳定后拉高该信号；如果参考时钟在加载前已经是稳定的，上电后直接给该信号一个从低到高的上升沿即可。有些应用中使用 clk2_out 端口来输出参考时钟（clk2_sel=2’b00）给逻辑用，当 ref_clk_en 拉低时，clk2_out 输出为常零；当 ref_clk_en 拉高时，clk2_out 可以输出参考时钟。其他说明见表 2-1。 |
| sram_init_done | Out | Async | SERDES 内部 SRAM 初始化完成信号，该信号拉高后用户逻辑可以通过 CRI 接口操作内部控制寄存器。 |
| sram_ext_ld_done | In | Async | 用户逻辑访问 CR 接口完成信号。在初始化阶段，如果有必要，用户逻辑可以通过 CRI 接口改变 SERDES 的配置。用户逻辑看到 Sram_init_done 拉高之后，即可访问 CRI 接口，访问操作结束后用户逻辑需拉高 sram_ext_ld_done 表示访问结束，SERDES 才能继续完成后面的复位流程。通常情况下，用户逻辑不需要在初始化阶段去访问 CRI 寄存器，用户逻辑就在 sram_init_done 起来之后直接拉高 sram_ext_ld_done。 |
| mplla_state | Out | Async | MPLLA 锁定状态指示信号。高电平表示 MPLLA 已被供电且锁定。当 tx*_mpllb_sel 信号为低电平时，即 MPLLA 被选中时有效。 |
| mpllb_state | Out | Async | MPLLB 锁定状态指示信号。高电平表示 MPLLB 已被供电且锁定。当 tx*_mpllb_sel 信号为高电平时，即 MPLLB 被选中时有效。 |
| **PCS 相关复位信号** | | | |
| l*_tx_pcs_rst_n | In | Async | 发送端 pcs 复位信号，使用时，需经过去除 glitch 处理，同步到本地时钟域。低有效，电平触发。高电平时间长度大于 20ns。 |

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| l*_tx_pcsfifo_rst_n | In | Async | TX PCS FIFO 复位信号，使用时，需经过去除 glitch 处理，同步到本地时钟域。低有效，电平触发。高电平时间长度大于 20ns。当 TX PCS FIFO 溢出时用户逻辑应发起该复位。 |
| l*_rx_pcs_rst_n | In | Async | 接收端 pcs 复位信号，使用时，需经过去除 glitch 处理，同步到本地时钟域。低有效，电平触发。高电平时间长度大于 20ns。 |
| l*_rx_rmfifo_rst_n | In | Async | 接收端 Rate Match FIFO 复位信号，使用时，需经过去除 glitch 处理,同步到本地时钟域。低有效，电平触发。高电平时间长度大于 20ns。当 RX Rate Match FIFO 溢出时用户逻辑应发起该复位。 |
| l*_rx_pcsfifo_rst_n | In | Async | RX PCS FIFO 复位信号，使用时，需经过去除 glitch 处理，同步到本地时钟域。低有效，电平触发。高电平时间长度大于 20ns。当 RX PCS FIFO 溢出时用户逻辑应发起该复位。 |

### 2.3.3 用法

复位顺序为，上电后做整体复位，等待 TX/RX PMA 复位结束，等待 TX/RX 用户时钟稳定（如果 TX/RX 用户时钟是 Fabric PLL 产生，则需等到 Fabric PLL 锁定，若 TX/RX 用户时钟是 BUFG 直接产生，跳过这一步），然后复位 TX/RX PCS。具体流程如图 2-8 所示。

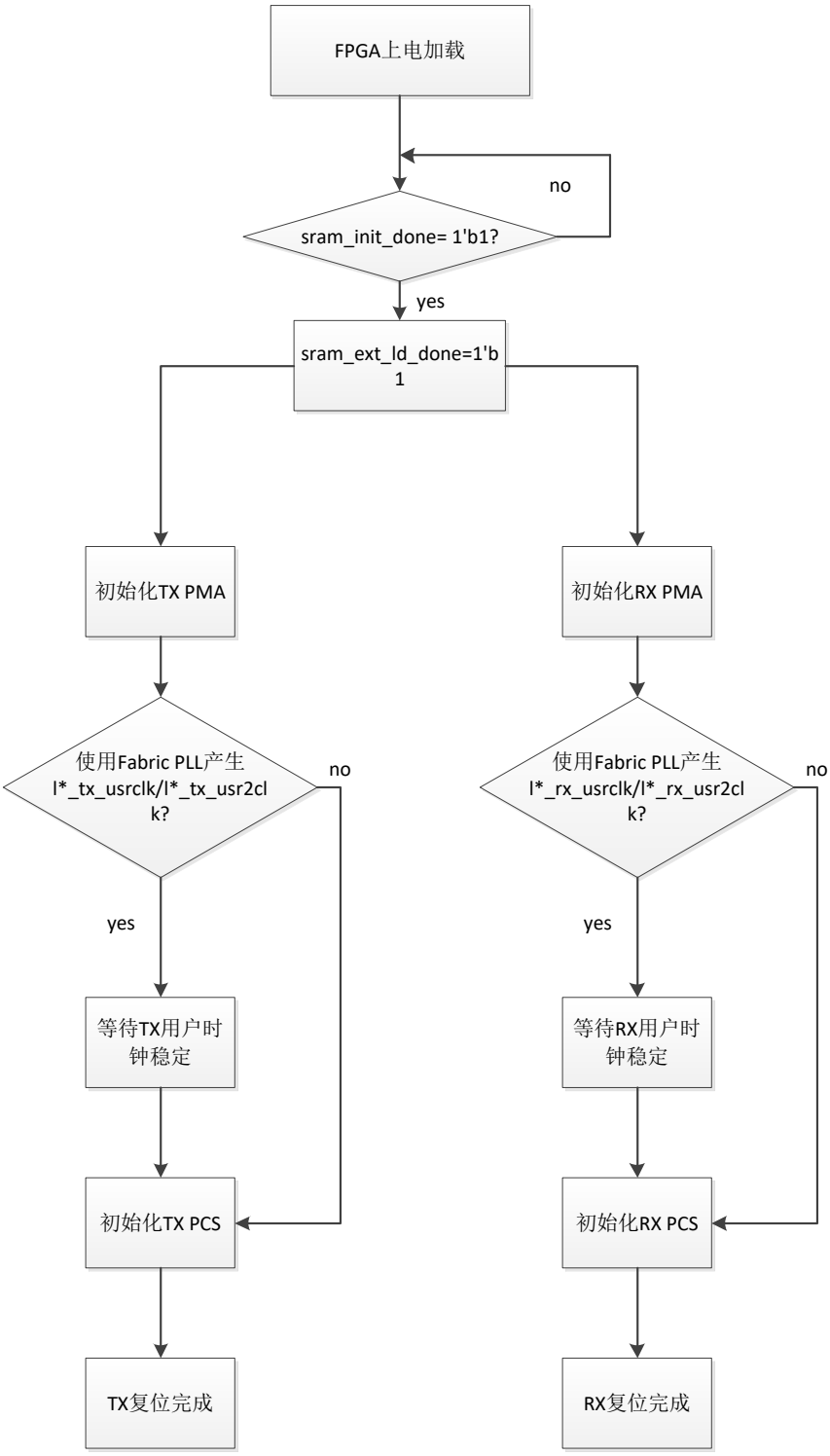


图 2-8 SERDES 初始化流程

#### 2.3.3.1. SERDES 整体复位和初始化流程

1. 上电加载完成后置位 phy_reset，此时每条 lane 对应的应答信号 tx*_ack，rx*_ack 都会随之拉高。此时也可以将各条 Lane 的单独复位信号 tx*_reset 和 rx*_reset 也置为高电平。同时，直接将 l*_tx_pcs_rst_n 和 l*_rx_pcs_rst_n 置为低电平，使两条 Lane 的 TX PCS 和 RX PCS 处于复位状态。

2. 等待 phy_reset 处于高电平足够长时间（上电后第一次拉高需要至少 20us，后续的再次复位仅需要拉高 20ns），将 phy_reset 置为低电平。若上一步骤中 tx*_reset 和 rx*_reset 被置为高电平，此时需将这些信号置为低电平。

3. 当参考时钟稳定时拉高 ref_clk_en。如果参考时钟在加载之前就已经稳定，可以在 phy_reset 拉低时直接将 ref_clk_en 置为高电平。

4. 等待 sram_init_done 置位，此信号置位表示内部 SRAM 初始化完成，此时用户逻辑可以访问 CRI 接口。

5. 当 CRI 接口访问完毕时，用户逻辑将 sram_ext_ld_done 信号置为高电平，告知 SERDES 外部逻辑访问 CRI 接口已经完成。如果用户逻辑不需要在初始化过程中访问 CRI 接口，那么可以在 sram_init_done 上升沿之后直接将 sram_ext_ld_done 置为高电平，无需等待时间。

6. 等待 mplla_state 置位（若 MPLLA 被使能）或 mpllb_state 置位（若 MPLLB 被使能）。此信号置位并保持高电平表示相应的 PLL 锁定。

7. 对于发送方向，等待 tx*_ack 信号拉低，此信号拉低表示 TX PMA 初始化完成。

8. 当 tx*_ack 信号拉低以后，可以无需等待直接将 tx*_data_en 置为高电平，使能发送数据通道。同时将 l*_tx_pcs_rst_n 也置为高电平，释放 TX PCS 的复位。

9. TX PCS 解复位以后需等待至少 20 个 l*_tx_usr2clk 周期后方可发送有效数据，在此之前可以发送空闲字符（IDLE）或者训练序列（Training Squence），此时 TX 方向初始化完全结束。

10. 对于接收方向，等待 rx*_ack 信号拉低。

11. 当 rx*_ack 信号拉低以后，可以无需等待直接将 rx*_data_en 置为高电平使能接收数据通道，此时 CDR 开始跟踪接收数据。

12. 等待 rx*_valid 信号拉高，当此信号拉高并保持高电平表示 CDR 处于锁定状态，此时将 l*_rx_pcs_rst_n 置为高电平释放 RX PCS 的复位，等待至少 20 个 l*_rx_usr2clk 时钟周期后，RX 方向初始化完成。从 rx*_data_en 拉高到 rx*_valid 稳定拉高的时间间隔是 CDR 锁定时间，具体可以参考 DS900。

整体复位时序图如图 2-9 所示。

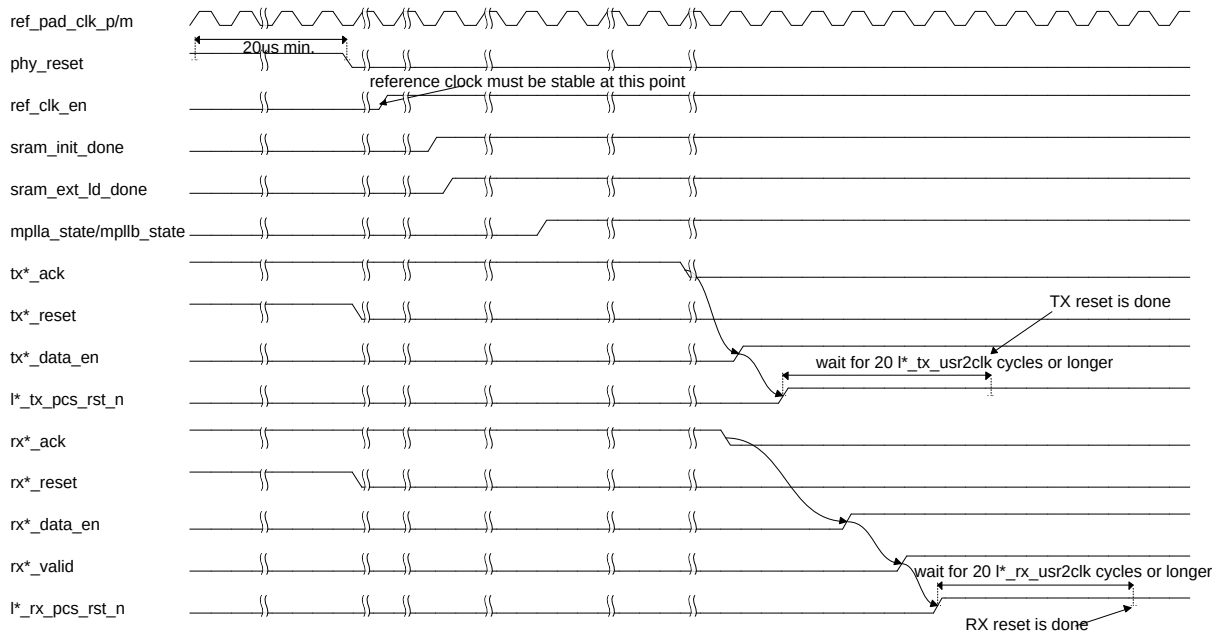


图 2-9 SERDES 上电后整体复位时序图

### 2.3.3.2. 各条 Lane 分别独立复位流程

除了前文所述整体复位之外，SERDES 还支持两条 Lane 分别复位，而不影响另外一条 Lane。各自复位一条 Lane 的时序图如图 2-10 所示。

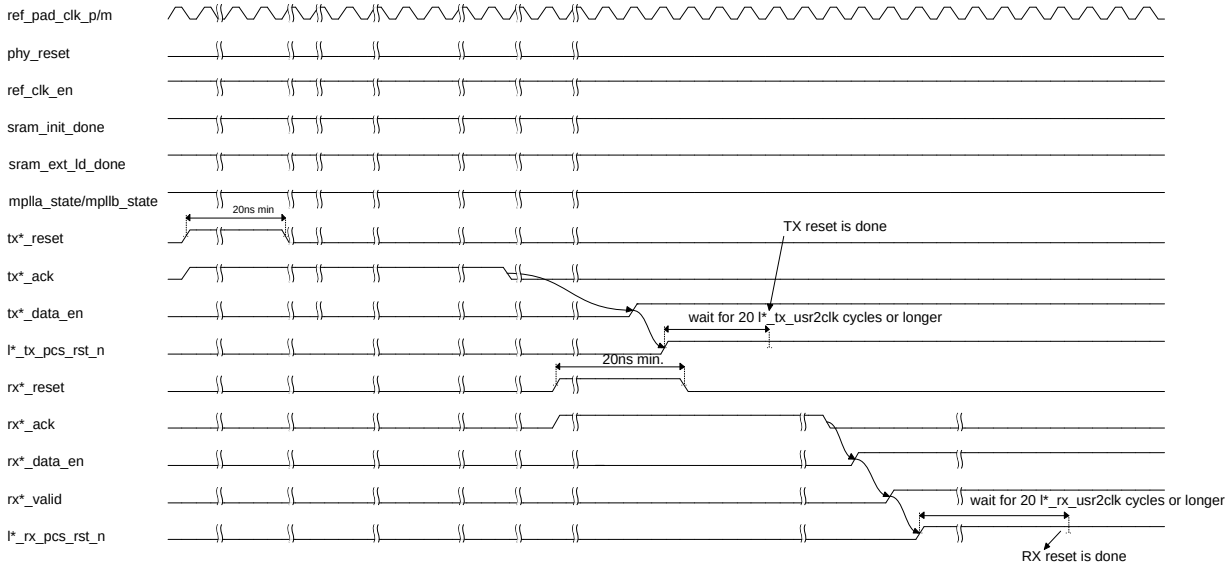


图 2-10 单独复位一条 Lane 的 TX 或 RX 方向时序图

**复位 SERDES 某一条 Lane 的 TX 部分流程**

1. 先将 tx*_reset 置为高电平，星号表示 Lane 编码。此时相应 Lane 的 tx*_ack 信号随之拉高。同时将 l*_tx_pcs_rst_n 置为低电平，使 TX PCS 处于复位状态。

2. 等待 tx*_reset 保持高电平足够长时间（至少 20ns），将 tx*_reset 置为低电平。

3. 等待 tx*_ack 信号拉低，该信号拉低表示 TX PMA 已经复位完成。

4. 当 tx*_ack 信号拉低以后，可以无需等待直接将 tx*_data_en 置为高电平，使能发送数据通道，同时将 l*_tx_pcs_rst_n 也置为高电平释放 TX PCS 的复位。

5. TX PCS 解复位以后需等待至少 20 个 l*_tx_usr2clk 周期后方可发送有效数据，在此之前可以发送空闲字符（IDLE）或者训练序列（Training Sequence），此时 TX 方向初始化结束。

**复位 SERDES 某一条 Lane RX 部分流程**

1. 先将 rx*_reset 置位，星号表示 Lane 的编码。此时相应 Lane 的 rx*_ack 信号随之拉高。同时将 l*_rx_pcs_rst_n 置位低电平，使 RX PCS 处于复位状态。

2. 等待 rx*_reset 保持高电平足够长时间（至少 20ns），将 rx*_reset 置位低电平。

3. 等待 rx*_ack 信号拉低，该信号拉低表示 RX PMA 已经复位完成。

4. 当 rx*_ack 信号拉低以后，可以无需等待直接将 rx*_data_en 置为高电平，使能接收数据通道，此时 CDR 开始跟踪对端发来的数据。

5. 等待 rx*_valid 信号拉高并保持为高电平，表示相应 SERDES Lane 的 RX CDR 锁定，此时将 l*_rx_pcs_rst_n 置为高电平释放 RX PCS 的复位，等待至少 20 个 l*_rx_usr2clk 时钟周期后，RX 方向初始化结束。从 rx*_data_en 拉高到 rx*_valid 稳定拉高的时间间隔是 CDR 锁定时间，具体可以参考 DS900。

#### 2.3.3.3. 针对不同场景推荐使用的复位信号说明

在不同的应用场景下，推荐使用的复位信号如表 2-7 所示。

表 2-7 不同场景推荐使用的复位信号

| 需要复位的场景 | 需要被复位的模块 | 推荐使用的复位信号 |
|---|---|---|
| 上电或加载后 | 整个 SERDES DUAL | phy_reset |
| 参考时钟断开或者失锁,再恢复正常以后 | 整个 SERDES DUAL | phy_reset |
| 参考时钟频率发生变化以后 | 整个 SERDES DUAL | phy_reset |
| TX 方向并行时钟 l*_tx_usrclk、l*_tx_usr2clk 断开或者失锁再恢复锁定以后 | TX PCS | l*_tx_pcs_rst_n |
| TX PCS FIFO 溢出之后 | TX PCS FIFO | l*_tx_pcs_rst_n |
| 进入或者退出远端 PMA 环回以后 | 整个 TX | tx*_reset（参见“复位 SERDES 某一条 Lane TX 部分流程”） |
| 进入或者退出近端 PMA 环回以后 | 整个 RX | rx*_reset（参见“复 |

| 需要复位的场景 | 需要被复位的模块 | 推荐使用的复位信号 |
|---|---|---|
| | | 位 SERDES 某一条 Lane RX 部分流程”，以下同） |
| RX 方向并行时钟 l*_rx_usrclk、l*_rx_usr2clk 断开或者失锁再恢复锁定以后 | RX PCS | l*_rx_pcs_rst_n |
| 对端器件（Link Parnter）上电后 | 整个 RX | rx*_reset |
| RXP/M 从电气空闲恢复到正常数据模式 | 整个 RX | rx*_reset |
| RXP/N 信号被断开又恢复之后 | 整个 RX | rx*_reset |
| 恢复时钟锁定（rx*_valid 拉高）以后 | RX PCS FIFO | l*_rx_pcs_rst_n |
| RX PCS FIFO 溢出之后 | RX PCS FIFO | l*_rx_pcs_rst_n |
| RX PRBS Verifier 探测到错误之后 | RX PRBS Error Counter | l*_rx_prbs_err_clr |

注：推荐使用的复位信号指的是在指定场景下，影响范围最小的复位方式。

#### 2.3.3.4. PH1A 系列 FPGA SERDES 复位注意事项

**上电初始化注意事项**

推荐的整体复位流程是，当 phy_reset 总体复位拉高时，tx*_reset/rx*_reset 这 4 个单独复位也拉高；当 phy_reset 拉高达到足够的时间后拉低，启动整体复位流程，此时 tx*_reset/rx*_reset 这 4 个单独复位也随之拉低。若这 4 个单独复位信号某一个或某几个没有拉低，可能导致 PLL 无法锁定。针对这种情况，用户逻辑的应对的措施是，初始化过程中 4 个单独复位信号都要拉低，不能有 1 个或几个单独复位信号一直保持为高电平。

**单次复位注意事项**

整体复位结束时，可以用 tx*_reset/rx*_reset 进行单独的复位。当其中 2 个或 2 个以上同时拉高时，有可能导致 PLL 失锁，即 mplla_state 或 mpllb_state 拉低；仅有其中任何一个信号拉高时，PLL 不失锁。针对这种情况的应对措施是，当需要进行单独复位时，在同一时段只能触发一个单独复位的过程，当此过程结束后（ack 信号拉低），再进行下一个复位过程。

**PCS 复位注意事项**

当 PMA 复位完成以后，相应的 PCS 也需要进行复位。例如，当 tx*_reset 复位完成以后，即 tx*_ack 信号拉低后，需对 TX PCS 进行复位（l*_tx_pcs_rst_n），PCS 复位解除以后等待 20 个 l*_tx_usr2clk 周期后，TX PCS 复位完成。

**复位完成时间**

上电初始化完成时间，最大值约为 15ms。这个数字可能会随着测试的样本不断增加而发生变化。

# 2.4 环回模式

## 2.4.1 功能简介

环回分为近端环回和远端环回两类。近端环回是 SERDES Channel 发送器将发送的数据直接送到同一个 SERDES Channel 的接收器，把发送的数据再接收回来的环回模式；远端环回是接收器把收到的数据再通过同一个 SERDES Channel 的发送器送回源端的环回模式。

环回模式通常用于调式和定位问题。环回模式可以支持用户数据，也可以支持内部的 PRBS 生成器和检测器。

两类环回模式，分别都可以支持 PCS 和 PMA 两个层次的环回。因此，SERDES 共支持 4 种环回模式。它们分别是，近端 PCS 环回（Near End PCS Loopback），近端 PMA 环回(Near End PMA Loopback)，远端 PMA 环回 (Far End PMA Loopback)，远端 PCS 环回(Far End PCS Loopback)。各种环回数据路径如图 2-11 所示。

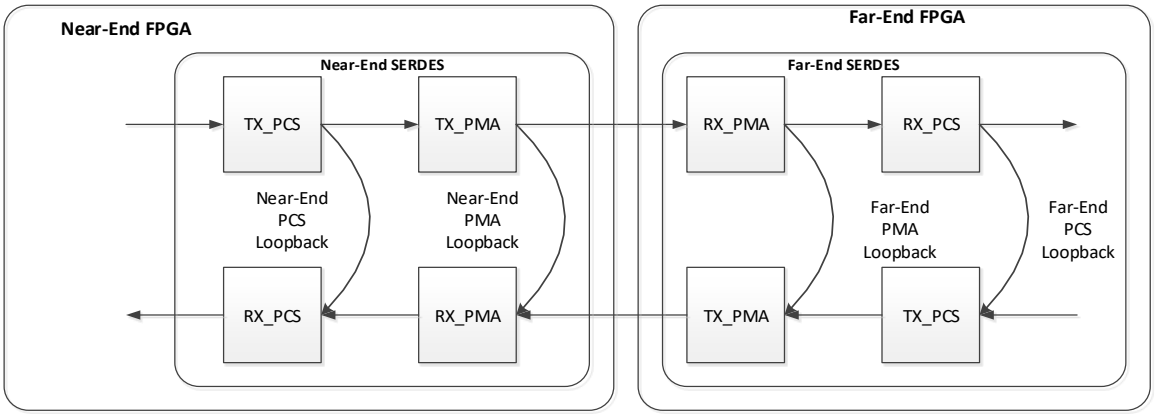


图 2-11 环回路径

## 2.4.2 端口

环回相关端口如表 2-8 所示。

表 2-8 环回相关端口列表

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| l*_loopback_en | In | cr_para_clk | 环回模式使能信号，高有效。 |
| l*_loopback_mode[1:0] | In | cr_para_clk | 环回模式选择信号。编码映射如下：<br>• 2’b00：近端 PCS 环回<br>• 2’b01：近端 PMA 环回<br>• 2’b10：远端 PMA 环回<br>• 2’b11：远端 PCS 环回<br>当 l*_loopback_en 为 1’b1 时，此端口有效。 |

## 2.4.3 用法

进入环回模式之前，需确保下面端口已经按照要求接好，并确保 2.3.3 章节介绍的整体复位流程已经完成，以及参考时钟必须稳定。

test_burnin = 1’ b0;

test_powerdown_i = 1'b0 ;

test_flyover_en = 1'b0 ;

scan_mode = 1'b0 ;

scan_shift = 1'b0 ;

scan_shift_cg = 1'b0 ;

tx*_reset， rx*_reset， tx*_lpd， rx*_lpd 均为低电平状态

tx*_pstate，rx*_pstate 均为 P0 状态

进入环回模式时序如图 2-12 所示。

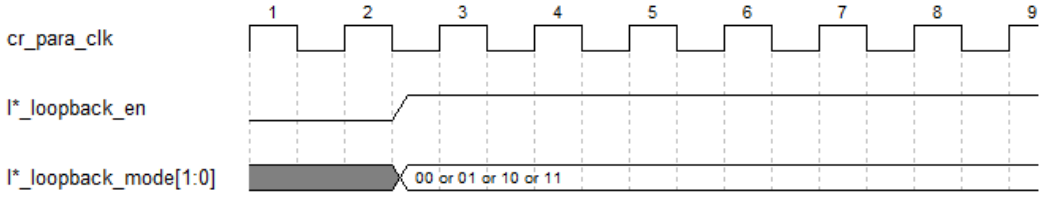


图 2-12 进入环回模式时序图

退出环回模式时序如图 2-13 所示。

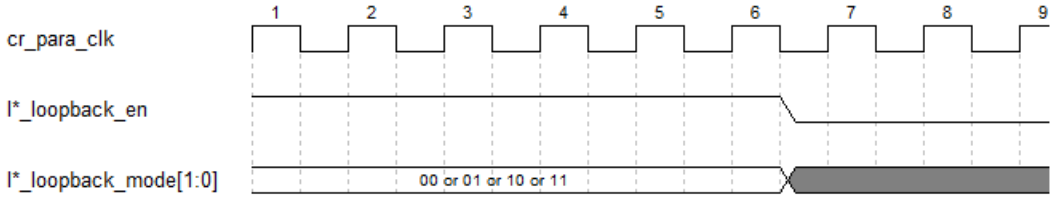


图 2-13 退出环回模式时序图

### 2.4.3.1. 4 种环回模式用法和注意事项

**Near End PCS Loopback 近端 PCS 环回**

RX PCS FIFO必须使能。近端PCS环回数据路径不经过PMA部分，因此RX PMA中的CDR可能会失锁，此时用户逻辑若使用恢复时钟驱动，会导致功能错误。因此如有必要，需在此模式下使用本地时钟来驱动用户逻辑。

**Near End PMA Loopback 近端 PMA 环回**

进入或者退出Near End PMA Loopback以后，必须对rx*_reset进行复位。近端PMA环回包含tx*_invert和rx*_invert电路。Near end PMA Loopback模式仅支持在调试时使用，在该模式下，需要设置rx*_eq_ctle_boost=0、rx*_eq_vga1_gain=7、rx*_eq_vga1_gain=7，具体端口说明见4.2节表4-1。

**Far End PMA Loopback 远端 PMA 环回**

进入或者退出Far End PMA Loopback以后，必须对tx*_reset进行复位。远端PMA环回仅支持同源时钟系统，远端环回器件必须和对端器件由同源时钟驱动。远端PMA环回不包含tx*_invert电路，在这种模式下tx*_invert不起作用。远端PMA环回包含rx*_invert电路。远端PMA环回模式仅支持tx*_rate和rx*_rate相等的情况，当tx*_rate和rx*_rate不相等时，不能使用远端PMA环回模式。

**Far End PCS Loopback 远端 PCS 环回**

如果RX Rate Match FIFO没有使能，那么远端和近端的两个收发器必须被同源参考时钟驱动。如果远端和近端的两个收发器为非同源时钟驱动，那么必须使用Rate Match FIFO来解决两端的PPM频率偏差。

## 2.5 控制寄存器访问接口 CRI

### 2.5.1 功能简介

SERDES 提供一个并行接口，称为 Control Register Interface（CRI），用户可以通过该接口访问 SERDES 内部控制寄存器。

CRI 是一个同步的 16bit 地址总线和数据总线的接口。一般情况下不需要访问内部控制寄存器。在一些特殊情况下，比如需要读取一些 debug 信息或者需要改写某些控制寄存器，那么才需要用到这个接口。

CRI 接口包括下面这些信号：cr_para_addr，cr_para_clk，cr_para_rd_data，cr_para_rd_en，cr_para_wr_data，cr_para_wr_en，和 cr_para_ack。

cr_para_wr_en 和 cr_para_rd_en 只能同时拉高其中一个，并且脉冲宽度只拉高一个 cr_para_clk 周期。拉低后需要保持低电平至少 7 个时钟周期才能再次拉高。在 phy_reset 拉高时，这个 CRI 接口是不能访问的。上电后，phy_reset 拉低之后，CRI 才可以被访问。在参考时钟稳定之后（phy_reset 拉低并且 ref_clk_en 拉高），必须等待至少 50 个参考时钟周期才能使用该接口。

CRI 的读写时序图分别如 2.5.3 章节的图 2-14、2-15 所示。

每次读写要等待 cr_para_ack 信号拉高才能进行下次读写。cr_para_ack 信号拉高一拍。用户可以在 cr_para_ack 刚拉低就直接发起背靠背请求。

### 2.5.2 端口和属性

寄存器控制总线相关的端口如表 2-9 所示。

表 2-9 CRI 接口信号列表

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| cr_para_ack | Out | cr_para_clk | CRI 的结束信号，拉高一个时钟周期，表示读或写操作已经完成。可以进行下一次访问。 |
| cr_para_clk | In | 时钟 | 寄存器控制总线的时钟。最高支持 100MHz。 |
| cr_para_addr[15:0] | In | cr_para_clk | CRI 地址。 |
| cr_para_rd_data[15:0] | Out | cr_para_clk | CRI 读出数据。在 cr_para_ack 拉高的那一拍有效。 |
| cr_para_rd_en | In | cr_para_clk | CRI 读请求信号。拉高 cr_para_clk 一拍发起一次请求。 |
| cr_para_wr_data[15:0] | In | cr_para_clk | CRI 写数据总线。 |
| cr_para_wr_en | In | cr_para_clk | CRI 写请求信号。拉高 cr_para_clk 一拍发起一次请求。 |

## 2.5.3 用法

写操作时序如图 2-14 所示。写请求发出时要确保 cr_para_ack 为低电平。cr_para_wr_en 和 cr_para_addr/cr_para_wr_data 同时有效，cr_para_wr_en 拉高一拍，cr_para_addr/cr_para_wr_data 应保持。等到 cr_para_ack 信号拉高后，本次写操作已完成。等 cr_para_ack 拉低可以进行下一次读或写操作。在 cr_para_wr_en 再次拉高时，更新 cr_para_addr/cr_para_wr_data。

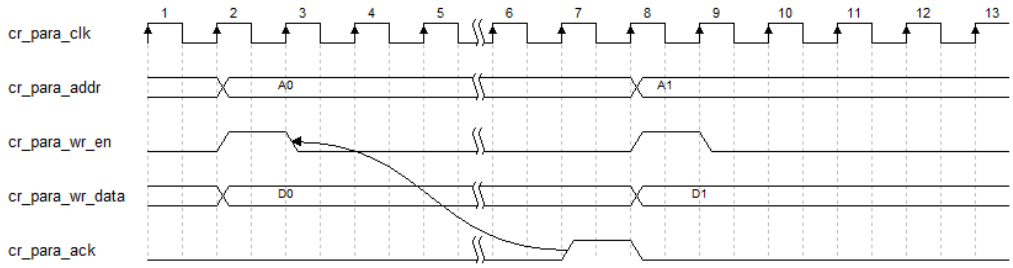


图 2-14 CRI 写时序

读操作时序如图 2-15 所示。读请求发出时要确保 cr_para_ack 为低电平。cr_para_rd_en_i 和 cr_para_addr 同时有效，cr_para_rd_en_i 拉高一拍，cr_para_addr 应保持。等到 cr_para_ack 信号拉高后，表示本次读操作已完成。等到 cr_para_ack 拉低可以进行下一次读或写操作。在 cr_para_rd_en_i 再次拉高时更新 cr_para_addr。

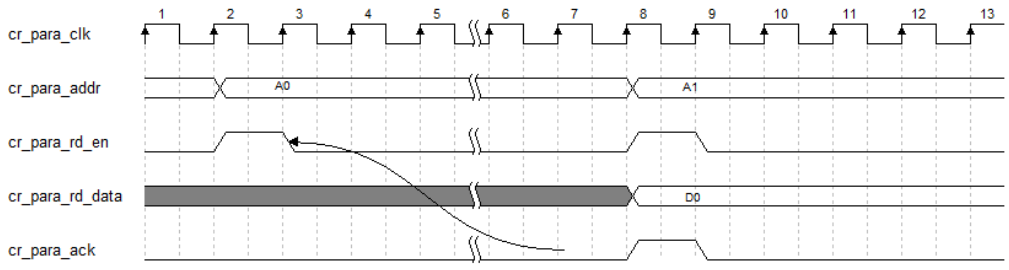


图 2-15 CRI 读时序

## 2.6 端接电阻适配

### 2.6.1 功能简介

发送管脚 TXP，TXM 各有一个 50 欧姆上拉和一个 50 欧姆下拉端接电阻，如图 2-16 所示。这个电阻通过外部的参考电阻进行校准，用于减少通道上的反射，提高信号完整性。

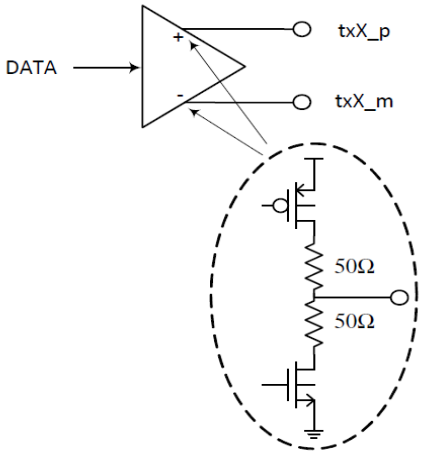


图 2-16 TX 端接电阻

### 2.6.2 端口和属性

端接电阻适配相关列表如表 2-10 所示。

表 2-10 TX 端接相关参数

| 信号名 | 类型和宽度 | 说明 |
|---|---|---|
| TXDN_TERM_OFFSET | 9 | 在外部参考电阻自动校准的基础上，进一步调整 TX 下拉电阻的阻值。这是一个 2 的补码有符号数。保留。推荐使用 IP Generator 产生的值。 |
| TXUP_TERM_OFFSET | 9 | 在外部参考电阻自动校准的基础上，进一步调整 TX 上拉电阻的阻值。这是一个 2 的补码有符号数。保留。推荐使用 |

| 信号名 | 类型和宽度 | 说明 |
| --- | --- | --- |
| | | IP Generator 产生的值。 |

# 2.7 SERDES 相关芯片外部管脚

## 2.7.1 功能简介

本节介绍 SERDES 相关的芯片管脚。包括参考时钟管脚，数据发送和接收管脚，校准电阻管脚，恢复时钟输出管脚，电源管脚等。

## 2.7.2 端口和属性

芯片外部管脚如表 2-11 所示。

表 2-11 管脚列表

| 管脚名 | 方向 | 说明 |
| --- | --- | --- |
| REFCLKM_<Bank#>*<br>REFCLKP_<Bank#> | In | 参考时钟差分输入。每个DUAL对应一个参考时钟差分对输入。如果不用须接地。 |
| RESREF_<Bank#> | In | 端接校准用的外部参考电阻端口。通过200欧姆精度1%的电阻接地。每个DUAL对应一个RESREF管脚。 |
| RXM0_<Bank#>/RXP0_<Bank#><br>RXM1_<Bank#>/RXP1_<Bank#> | In | 差分数据输入RX端口。 |
| TXM0_<Bank#>/TXP0_<Bank#><br>TXM1_<Bank#>/TXP1_<Bank#> | Out | 差分数据输出TX端口。 |
| RXRECCLKM_<Bank#><br>RXRECCLKP_<Bank#> | Out | 恢复时钟输出差分输出，用于恢复时钟输出到外部。每个DUAL有一对差分输出，用户可选哪个Lane的恢复时钟输出到该差分对。仅有SFG900器件Bank 80和84支持这个功能。输出电平标准为CML。 |
| PHYVCCA_<Bank#> | In | SERDES收发器模拟供电。 |
| PHYVCCT_<Bank#> | In | SERDES收发端口模拟供电。 |

*‘Bank#’表示 IO 管脚所在 Bank 的编号，SFG900 器件 8 个 Bank，编号分别为 80~87；SFG676 器件共有 4 个 Bank，编号分别为 80~83。

## 2.7.3 用法

### 2.7.3.1. 专用恢复时钟输出功能使用方法

在同步系统中，例如 CPRI，用户需要用恢复时钟作为下一级收发器的参考时钟源。在这类应用中，送出片外恢复时钟的相位噪声是非常关键的指标。如果用普通 IO 管脚输出恢复时钟，其相位噪声很会

受到 FPGA 逻辑翻转的影响，难以控制在较为理想的状态。而对于低频段的相位噪声，外部 Clean Up PLL 很难滤除。针对这类应用，PH1A400SFG900 Bank 80 和 Bank 84 SERDES 使用专用恢复时钟输出差分管脚 RXRECCLKP/ RXRECCLKM 将恢复时钟送出片外，避免了 FPGA 逻辑翻转的影响，给下一级设备的参考时钟提供了干净的时钟源。

PH1A400SFG900 Bank 80 和 Bank 84 SERDES 专用恢复时钟管脚用法如下：

PH1A400SFG900 Bank 80 和 Bank 84 的 SERDES DUAL 提供一对差分 CML 电平专用恢复时钟输出管脚。设置 RXCLK_BUF_EN 为 1’b1，使能此项功能；用 RXCLK_BUF_SEL 选择要输出恢复时钟的 Lane，RXCLK_BUF_SEL=1’b0 时选择 Lane0 恢复时钟输出到该管脚，RXCLK_BUF_SEL=1’b1 时选择 Lane1 恢复时钟输出到该管脚。

PH1A90、PH1A180、PH1A400SFG676 器件，均不支持此功能。

#### 2.7.3.2. SERDES 不使用时管脚处理方法

在一些应用中不需要使用 SERDES，此情况下按照表 2-12 来处理。

表 2-12 SERDES 不使用的情况下的管脚接法

| 管脚名 | 说明 |
| --- | --- |
| PHYVCCA<br>PHYVCCT | 接地 |
| RESREF | 悬空 |
| RXP/RXM | 接地或者悬空 |
| TXP/TXM | 悬空 |
| REFCLKP/REFCLKM | 接地 |
| RXRECCLKP/ RXRECCLKM | 悬空 |

# 3 TX Channel

## 3.1 TX Channel 总览

### 3.1.1 功能简介

这一节介绍如何配置和使用 TX Channel 里的每一个模块。每个 SERDES Channel 有 1 个 TX Channel，包括 TX PMA 和 TX PCS。图 3-1 显示了 TX Channel 内的所有模块。并行数据从 TX Fabric Interface 进入，通过 TX PCS 和 TX PMA，然后从 TX AMP Control（TX 幅度控制）输出高速串行数据。

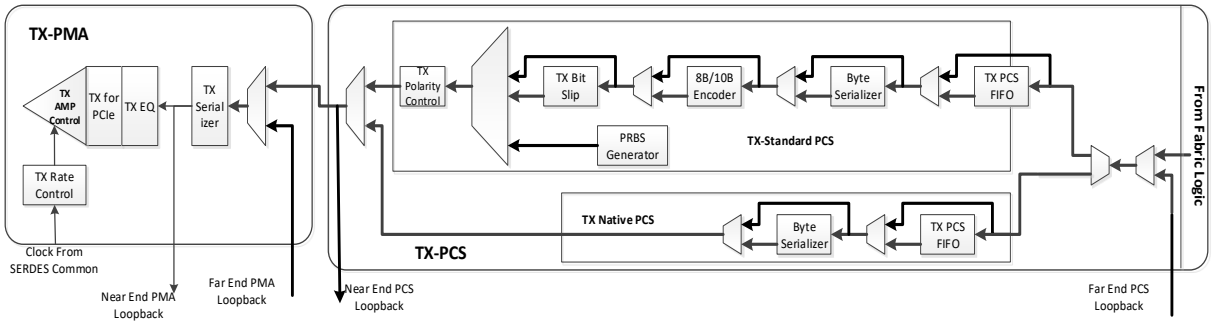


图 3-1 TX Channel 框图

TX PCS 分为 TX Standard PCS 和 TX Native PCS。如果用户需要使用 8B/10B 编码，需使能 TX Standard PCS；如果用户不使用 8B/10B 编码，既可以使能 TX Native PCS，也可以使能 TX Standard PCS。在用户不使用 8B/10B 功能，但使能 TX Standard PCS 的场景，需设置 8B/10B Encoder 为旁路状态。

TX Standard PCS 包含 TX PCS FIFO、Byte Serializer、8B/10B Encoder、TX Bit Slip、PRBS Generator 和 TX Polarity Control 等模块。TX Native PCS 包含 TX PCS FIFO、Byte Serializer 等模块。

TX PMA 包含模块 TX Serializer、TX EQ、TX for PCIe、TX AMP Control 和 TX Rate Control 等模块。

### 3.1.2 TX Channel 的时钟域

TX Channel 的时钟域如图 3-2 所示。

TX 方向主要有 3 个时钟域：

- TX Serial Clock 是发送串行高速时钟
- l*_tx_xclk(TX XCLK)是 PMA/PCS 并行时钟。
- l*_tx_usrclk/l*_tx_usr2clk 是用户时钟域，用于将用户并行数据读入到 SERDES 内部。二者频率不同。l*_tx_usrclk 的频率是线速率除以 PMA/PCS 内部数据位宽；而 l*_tx_usr2clk 的频率是线速率除以用户数据接口位宽。

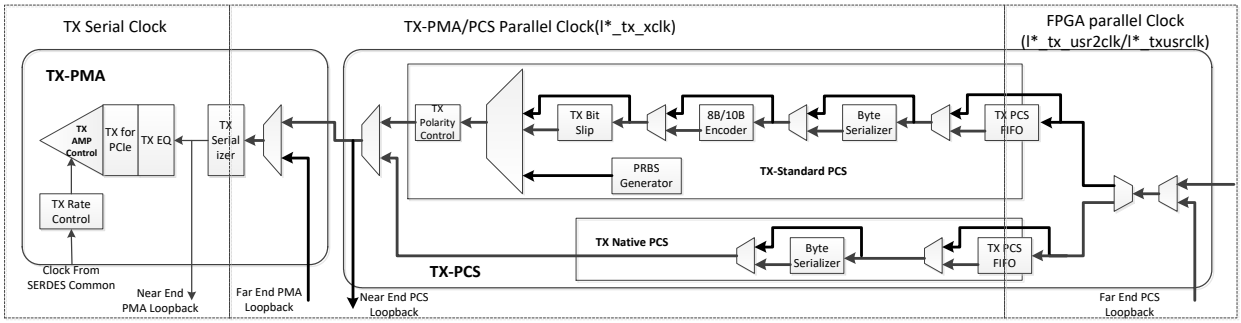


图 3-2 TX Channel 时钟域图

## 3.2 TX FABRIC 时钟

### 3.2.1 功能简介

发送时钟结构如图 3-3 所示。

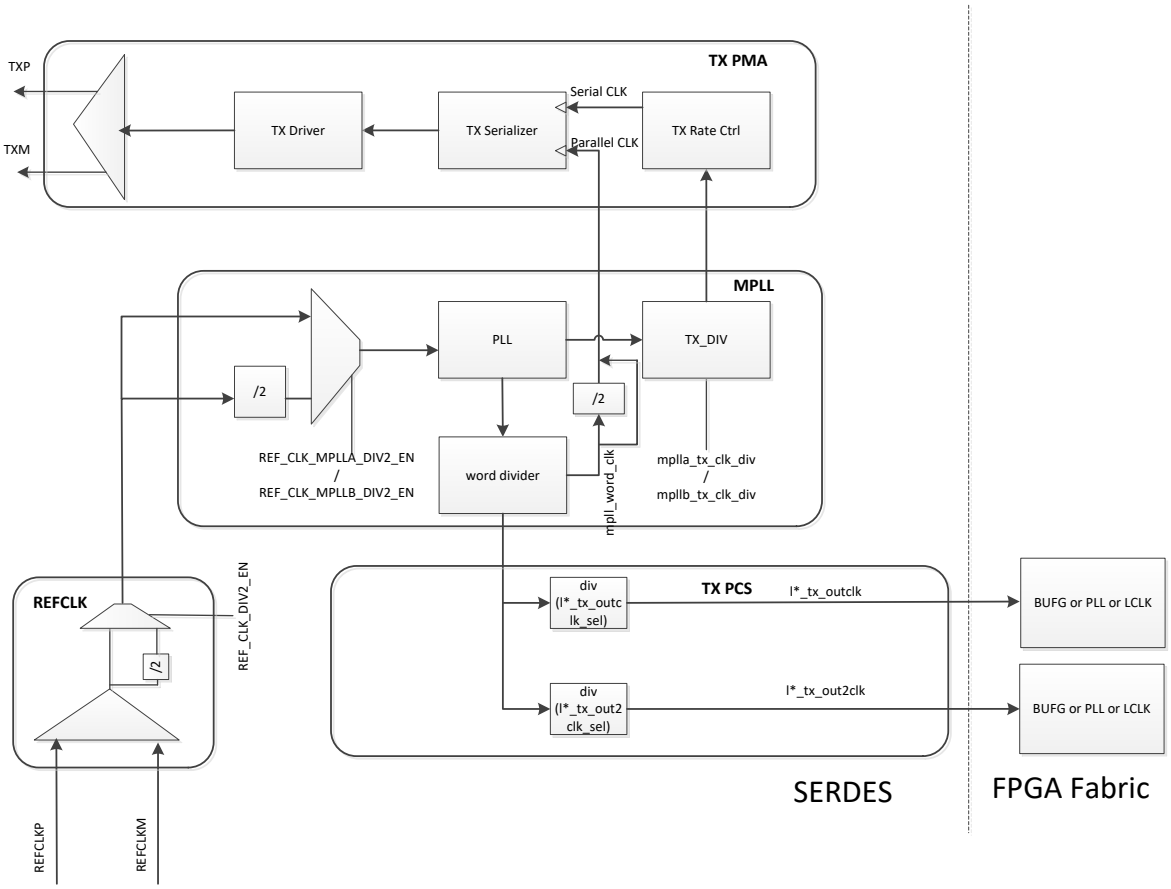


图 3-3 发送时钟结构

在发送方向，SERDES 可以输出 I*_tx_outclk 和 I*_tx_out2clk 这两个时钟，这两个时钟都可以进入全局时钟 BUFFER 或区域时钟 BUFFER 来驱动用户逻辑，这些时钟称为用户时钟。用户时钟送回给

SERDES 的 l*_tx_usrclk 或 l*_tx_usr2clk 时钟输入端口，作为用户逻辑并行数据(l*_tx_usrdata_in，l*_tx_usrdatak_in)的采样时钟，将用户逻辑要发送的数据采样到 SERDES TX 内部。下文描述 l*_tx_outclk 和 l*_tx_out2clk 产生的机理。

参考时钟进入参考时钟（REFCLK）模块，差分信号转为单端信号。经过可选的 2 分频器（由 REF_CLK_DIV2_EN 使能）进入 MPLL 模块。

在 MPLL 模块中，又经过一个可选的 2 分频器（由 REF_CLK_MPLLA/B_DIV2_EN 使能）送入相应的 PLL（MPLLA 或 MPLLB）。这个时钟经过 PLL 倍频后得到的高速时钟经由‘word divider’分频器进行分频得到 mpll_word_clk。mpll_word_clk 频率计算公式如下：

$$WORD\_CLKFREQ = \frac{MPLLVCOFREQ}{8}$$ （MPLLA/B_DIV8_CLK_EN=1’b1）或

$$WORD\_CLKFREQ = \frac{MPLLVCOFREQ}{10}$$ （MPLLA/B_DIV10_CLK_EN=1’b1）

以上公式表明，work_clk 频率是 VCO 的 8 分频或者 10 分频。这取决于内部数据位宽是 8 的整数倍还是 10 的整数倍。当内部位宽为 8bit (pmadata_width=0) 或 16bit (pmadata_width= 2)时，MPLLA/B_DIV8_CLK_EN 为 1’b1，这时 work_clk 为 MPLL VCO 的 8 分频；当内部位宽为 10bit (pmadata_width=1) 或 20bit (pmadata_width= 3)时，MPLLA/B_DIV10_CLK_EN 为 1’b1，这时 word_clk 为 MPLL VCO 的 10 分频。MPLLA/B_DIV8_CLK_EN 和 MPLLA/B_DIV10_CLK_EN 不可同时为 1。

当数据位宽为 1 字节时（8bit 或 10bit），word_clk 直接驱动 TX Serializer 的并行时钟；当数据位宽为 2 字节时（16bit 或 20bit），word_clk 经过 2 分频后送入 TX Serializer 的并行时钟。

PLL 倍频后的时钟经过 TX_DIV 分频器(由 MPLLA_TX_CLK_DIV 和 MPLLB_TX_CLK_DIV)，在经由 TX Rate 控制器分频，得到 TX Serializer 的串行时钟。

经由‘word divider’分频后的时钟 work_clk，经过分频电路，送入 TX PCS 模块，得到 l*_tx_outclk，和 l*_tx_out2clk 两个时钟信号。l*_tx_outclk 的分频系数由 l*_tx_outclk_sel 参数控制；l*_tx_out2clk 的分频系数由 l*_tx_out2clk_sel 来控制。如前文所述，l*_tx_outclk 和 l*_tx_out2clk 这两个时钟都可以经过 BUFG、PLL 或 LCLK 等时钟 BUFFER 送到 FPGA Fabric 中的时钟走线，提供给用户逻辑使用。同时这个时钟还需要送回 SERDES 的 l*_tx_usrclk 或 l*_tx_usr2clk 做为用户数据的采样时钟。

## 3.2.2 端口和属性列表

TX Fabric 时钟相关端口及属性如表 3-1、表 3-2 所示。

表 3-1 TX Fabric 时钟相关端口

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| tx*_mpllb_sel | In | Async | TX MPLLB 选择信号<br>用于选择使用 MPLLA 还是 MPLLB。拉高时选择 |

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| | | | MPLLB，否则选择 MPLLA。 |
| l*_tx_outclk | Out | 时钟 | 该时钟由时钟产生模块产生，它由 PLL 倍频后的时钟再经过分频产生。该时钟能驱动专用或者全局时钟网络，产生用户时钟 l*_tx_usrclk 和 l*_tx_usr2clk。 |
| l*_tx_out2clk | Out | 时钟 | 该时钟由时钟产生模块产生，为 MPLL 产生时钟的分频时钟，频率等于线速率除以 PCS/用户侧位宽。 |
| l*_tx_usrclk | In | 时钟 | 该时钟由用户产生，为 tx 方向用户侧时钟，频率等于线速率除以 PCS/PMA 位宽。l*_tx_outclk 可以用来驱动专用或者全局时钟网络，产生该时钟。 |
| l*_tx_usr2clk | In | 时钟 | 该时钟由用户产生，为 tx 方向用户侧时钟，频率等于线速率除以 PCS/用户侧位宽。l*_tx_outclk 可以用来驱动专用或者全局时钟网络，产生该时钟。 |
| l*_tx_outclk_sel[3:0] | In | Async | work_clk 分频得到 l*_tx_outclk 的分频系数选择信号。按位编码映射如下：<br>• 4’d0：static 0 （输出常 0）；<br>• 4’d1：mplla_word_clk path （MPLLA word_clk 同频）；<br>• 4’d2：mplla_dword_clk path （MPLLA word_clk 的 2 分频）；<br>• 4’d3：mplla_qword_clk path （MPLLA word_clk 的 4 分频）；<br>• 4’b4：mplla_oword_clk path （MPLLA word_clk 的 8 分频）；<br>• 4’d5：保留；<br>• 4’d6：保留；<br>• 4’d7：保留；<br>• 4’d8：mplla_div_clk （由 MPLLA_DIV_CLK_EN 参数使能，频率为 MPLLA_VCO_FREQ / （2 * MPLLA_DIV_MULTIPLIER [6:0]））；<br>• 4’d9：mpllb_word_clk path（MPLLB word_clk 同频）；<br>• 4’d10：mpllb_dword_clk path （MPLLB |

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| | | | word_clk 的 2 分频）<br>• 4’d11: mpllb_qword_clk path （MPLLB word_clk 的 4 分频）;<br>• 4’b12: mpllb_oword_clk path （MPLLB word_clk 的的 8 分频）;<br>• 4’d13: mpllb_div_clk （由 MPLLB_DIV_CLK_EN 参数使能，频率为 MPLLB_VCO_FREQ / （2 * MPLLB_DIV_MULTIPLIER [6:0]））;<br>• 4’d14: 保留;<br>• 4’d15：保留。<br>推荐使用 SERDES IP GUI 生成的默认值。 |
| l*_tx_out2clk_sel[3:0] | In | Async | work_clk 分频得到 l*_tx_out2clk 的分频系数选择信号。按位编码映射如下：<br>• 4’d0: static 0 （输出常 0）;<br>• 4’d1: mplla_word_clk path （MPLLA word_clk 同频）;<br>• 4’d2: mplla_dword_clk path （MPLLA word_clk 的 2 分频）;<br>• 4’d3: mplla_qword_clk path （MPLLA word_clk 的 4 分频）;<br>• 4’b4: mplla_oword_clk path （MPLLA word_clk 的 8 分频）;<br>• 4’d5: 保留;<br>• 4’d6: 保留;<br>• 4’d7: 保留;<br>• 4’d8: mplla_div_clk （由 MPLLA_DIV_CLK_EN 参数使能，频率为 MPLLA_VCO_FREQ / （2 * MPLLA_DIV_MULTIPLIER [6:0]））;<br>• 4’d9: mpllb_word_clk path（MPLLB word_clk 同频）;<br>• 4’d10: mpllb_dword_clk path （MPLLB word_clk 的 2 分频）; |

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| | | | • 4’d11: mpllb_qword_clk path （MPLLB word_clk 的 4 分频）；<br>• 4’b12: mpllb_oword_clk path （MPLLB word_clk 的 8 分频）；<br>• 4’d13: mpllb_div_clk （由 MPLLB_DIV_CLK_EN 参数使能，频率为 MPLLB_VCO_FREQ / （2 * MPLLB_DIV_MULTIPLIER [6:0]））；<br>• 4’d14: 保留；<br>• 4’d15：保留。<br>推荐使用 SERDES IP GUI 生成的默认值。 |
| l*_tx_align_reset | In | Async | 保留，接 0。 |
| l*_tx_align_start | In | Async | 保留，接 0。 |
| l*_tx_align_code_in[8:0] | In | Async | 保留，接 0。 |
| l*_tx_align_code_out[8:0] | Out | Async | / |
| l*_tx_ph_aligned | Out | Async | / |
| l*_tx_align_fail | Out | Async | / |

表 3-2 TX Fabric 时钟相关属性

| 属性名 | 类型和宽度 | 说明 |
| --- | --- | --- |
| L*_TX_ALIGN_EN | 1 | 保留，接 0。 |
| L*_TX_EXT_MODE | 1 | 保留，接 0。 |
| L*_TX_WINDOW_SET | 5 | 保留，接全 0。 |
| L*_TX_DIS_GSR | 1 | 保留，接 0。 |

## 3.3 TX Fabric Interface

### 3.3.1 功能简介

TX Fabric Interface 是 SERDES TX Channel 用户侧发送数据的接口。用户在 l*_tx_usr2clk 的上升沿把需要发送的数据写到 TX Fabric Interface。TX Fabric Interface 的数据位宽用 l*_tx_usrdata_width 端口来配置。用户可以把数据位宽配置为 8/10/16/20/32/40/64/80 bit。l*_tx_usr2clk 的频率，由 TX 线速、TX Fabric Interface 的数据位宽来决定。如果用户使能了 TX Standard PCS 里的 8B/10B Encoder，TX Fabric Interface 数据位宽应该是 8bit 的倍数。

#### 3.3.1.1. 接口位宽配置

PH1A SERDES 中 TX Channel 的内部数据位宽，就是 TX PCS 和 TX PMA 之间并行接口的数据位宽，可以是 8/10/16/20 bit。 当 TX Standard PCS 中的 8B/10B Encoder 被使能时，TX Channel 内部数据位宽只能是 10 bit 的倍数，即 10/20 bit，而 TX Fabric Interface 的数据位宽必须是 8 bit 的倍数，即 8/16/32/64 bit，TX PCS 数据通路位宽配置如表 3-3 所示。

表 3-3 TX PCS 数据通路位宽配置表

| TX PCS 模块控制信号 | TX Fabric Interface 数据位宽控制信号 | TX Channel 用户侧数据 | TX Channel |
|---|---|---|---|
| **l*_pcs_standard_en, l*_pcs_native_en, l*_tx_enc_en** | **l*_tx_usrdata_width [2:0]** | **l*_tx_usrdata_in [79:0]** | **内部数据位宽** |
| l*_pcs_standard_en= 1’b1 且 l*_pcs_native_en = 1’b0 且 l*_tx_enc_en = 1’b1 | 3’b000 | 8 | 10 |
| | 3’b010 | 16 | 10/20 |
| | 3’b100 | 32 | 10/20 |
| | 3’b110 | 64 | 20 |
| l*_pcs_standard_en = 1’b1 且 l*_pcs_native_en = 1’b0 且 l*_tx_enc_en = 1’b0 或 l*_pcs_standard_en = 1’b0 且 l*_pcs_native_en = 1’b1 | 3’b000 | 8 | 8 |
| | 3’b001 | 10 | 10 |
| | 3’b010 | 16 | 8/16 |
| | 3’b011 | 20 | 10/20 |
| | 3’b100 | 32 | 8/16 |
| | 3’b101 | 40 | 10/20 |
| | 3’b110 | 64 | 16 |
| | 3’b111 | 80 | 20 |

由上表可以看出，TX Channel 的用户侧数据位宽跟内部数据位宽，可以是 1:1/2:1/4:1 的比例关系。当 8B/10B Encoder 使能时，用户侧的每 8bit 数据等同于 10bit。相对应的，TX Channel 的用户侧时钟跟 SERDES PCS 内部工作时钟的频率，可以是 1:1/1:2/1:4。

#### 3.3.1.2. l*_tx_usrclk 和 l*_tx_usr2clk 的要求

TX Fabric Interface 有 2 个时钟输入：l*_tx_usrclk 和 l*_tx_usr2clk。这两个时钟都可以作为用户接口的时钟，仅需要接其中一个，另一个接地。SERDES 具体使用哪个时钟作为 TX 用户逻辑数据的采样时钟，取决于 TX PCS FIFO 是否使能，以及 TX PCS FIFO 写入时钟的选择信号（l*_tx_iclk_sel）。用户可以直接在 TD 软件的 IP Generator 中进行配置。IP Generator 产生的 SERDES 应用通常使用 l*_tx_usr2clk，并将 l*_tx_usrclk 接地。在 TX PCS FIFO Bypass 的应用中，可能会使用 l*_tx_usrclk 作为 TX 用户逻辑并行数据的采样时钟。在 TX PCS FIFO 使能的应用中，l*_tx_usr2clk 或 l*_tx_usrclk

用于驱动 TX PCS FIFO 的写时钟。l*_tx_usr2clk（或 l*_tx_usrclk）的频率跟 TX 线速和 TX Channel 用户侧数据位宽（l*_tx_usr_datawidth）有关，其对应关系如下：

$$l*\_tx\_usr2clk\_freq = \frac{l*\_tx\_linerate}{l*\_tx\_usr\_datawidth}$$

## 3.3.2 端口和属性

TX Fabric Interface 端口如表 3-4 所示。

表 3-4 TX Fabric Interface 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| l*_tx_usrdata_in[79:0] | In | l*_tx_usr2clk | TX PCS 发送端用户侧数据，有效位宽根据 l*_tx_usrdata_width 设置。<br>• l*_tx_usrdata_width=0, l*_tx_usrdata_in[7:0]有效；<br>• l*_tx_usrdata_width=1, l*_tx_usrdata_in[9:0]有效；<br>• l*_tx_usrdata_width=2, l*_tx_usrdata_in[15:0]有效；<br>• l*_tx_usrdata_width=3, l*_tx_usrdata_in[19:0]有效；<br>• l*_tx_usrdata_width=4, l*_tx_usrdata_in[31:0]有效；<br>• l*_tx_usrdata_width=5, l*_tx_usrdata_in[39:0]有效；<br>• l*_tx_usrdata_width=6, l*_tx_usrdata_in[63:0]有效；<br>• l*_tx_usrdata_width=7, l*_tx_usrdata_in[79:0]有效。 |
| l*_tx_usrdata_width[2:0] | In | cr_para_clk | TX PCS 用户侧数据有效位宽。<br>• 0 表示 8bit 有效；<br>• 1 表示 10bit 有效；<br>• 2 表示 16bit 有效；<br>• 3 表示 20bit 有效；<br>• 4 表示 32bit 有效；<br>• 5 表示 40bit 有效； |

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| | | | • 6 表示 64bit 有效；<br>• 7 表示 80bit 有效。 |
| l*_tx_usrdatak_in[7:0] | In | l*_tx_usr2clk | K 字符指示，高有效。<br>• l*_tx_usrdatak_in[7] 对应 l*_tx_usrdata_in[63:56]；<br>• l*_tx_usrdatak_in[]6 对应 l*_tx_usrdata_in[55:48]；<br>• l*_tx_usrdatak_in[5] 对应 l*_tx_usrdata_in[47:40]；<br>• l*_tx_usrdatak_in[4] 对应 l*_tx_usrdata_in[39:32]；<br>• l*_tx_usrdatak_in[3] 对应 l*_tx_usrdata_in[31:24]；<br>• l*_tx_usrdatak_in[2] 对应 l*_tx_usrdata_in[23:16]；<br>• l*_tx_usrdatak_in[1] 对应 l*_tx_usrdata_in[15:8]；<br>• l*_tx_usrdatak_in[0] 对应 l*_tx_usrdata_in[7:0]。 |
| l*_tx_forcedisp_in[7:0] | In | l*_tx_usr2clk | 8B/10B 编码的强制设置极性，高有效，指示采用 l*_tx_forcedispval_in 作为极性。<br>• l*_tx_forcedisp_in[0] 对应 l*_tx_forcedispval_in[0]；<br>• l*_tx_forcedisp_in[1] 对应 l*_tx_forcedispval_in[1]；<br>• l*_tx_forcedisp_in[2] 对应 l*_tx_forcedispval_in[2]；<br>• l*_tx_forcedisp_in[3] 对应 l*_tx_forcedispval_in[3]；<br>• l*_tx_forcedisp_in[4] 对应 l*_tx_forcedispval_in[4]；<br>• l*_tx_forcedisp_in[5] 对应 l*_tx_forcedispval_in[5]；<br>• l*_tx_forcedisp_in[6] 对应 |

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| | | | l*_tx_forcedispval_in[6]；<br>• l*_tx_forcedisp_in[7] 对应 l*_tx_forcedispval_in[7]。 |
| l*_tx_forcedispval_in[7:0] | In | l*_tx_usr2clk | 从用户侧得到的 8B/10B 编码的强制极性，高为正，低为负。当 l*_tx_forcedisp_in 对应位为高时，该值有效。<br>• l*_tx_forcedispval_in[7] 对应 l*_tx_usrdata_in[63:56]；<br>• l*_tx_forcedispval_in[6] 对应 l*_tx_usrdata_in[55:48]；<br>• l*_tx_forcedispval_in[5] 对应 l*_tx_usrdata_in[47:40]；<br>• l*_tx_forcedispval_in[4] 对应 l*_tx_usrdata_in[39:32]；<br>• l*_tx_forcedispval_in[3] 对应 l*_tx_usrdata_in[31:24]；<br>• l*_tx_forcedispval_in[2] 对应 l*_tx_usrdata_in[23:16]；<br>• l*_tx_forcedispval_in[1] 对应 l*_tx_usrdata_in[15:8]；<br>• l*_tx_forcedispval_in[0] 对应 l*_tx_usrdata_in[7:0]。 |
| l*_tx_usrclk | In | 时钟信号 | 该时钟和 l*_tx_usr2clk 都可以作为 TX 用户逻辑并行数据的采样时钟，若使用 l*_tx_usr2clk 作为采样时钟，l*_tx_usrclk 可以接地，但不能悬空或者接高电平。反之，若使用 l*_tx_usrclk 作为 TX 用户逻辑数据的采样时钟，l*_tx_usr2clk 可以接地；此时 l*_tx_usrclk 频率见公式 3-1。该时钟可以由 l*_tx_outclk 或者 l*_tx_out2clk 经过全局时钟 BUFFER（BUFG），区域时钟 BUFFER（LCLK）或者 PLL 产生。在 IP Generator 中，大多数应用使用 l*_tx_usr2clk，而将 l*_tx_usrclk 接地。 |
| l*_tx_usr2clk | In | 时钟信号 | 该时钟和 l*_tx_usrclk 都可以作为 TX 用户逻 |

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| | | | 辑并行数据的采样时钟，若使用 I*_tx_usrclk 作为采样时钟，I*_tx_usr2clk 可以接地，但不能悬空或者接高电平。反之，若使用 I*_tx_usr2clk 作为 TX 用户逻辑数据的采样时钟，I*_tx_usrclk 可以接地；此时 I*_tx_usr2clk 频率见公式 3-1。该时钟可以由 I*_tx_outclk 或者 I*_tx_out2clk 经过全局时钟 BUFFER（BUFG），区域时钟 BUFFER（LCLK）或者 PLL 产生。在 IP Generator 中，大多数应用使用 I*_tx_usr2clk，而将 I*_tx_usrclk 接地。 |
| I*_tx_outclk | Out | 时钟信号 | 该时钟由 SERDES Common 的 PLL 模块经过分频产生，和 I*_tx_out2clk 都可以驱动全局时钟，区域时钟或者 PLL，作为 TX 用户逻辑使用的时钟以及 SERDES TX 用户时钟 I*_tx_usrclk 或 I*_tx_usr2clk 。在 IP Generator 中，通常使用 I*_tx_out2clk，而将 I*_tx_outclk 悬空。 |
| I*_tx_out2clk | Out | 时钟信号 | 该时钟由 SERDES Common 的 PLL 模块经过分频产生，和 I*_tx_outclk 都可以驱动全局时钟，区域时钟或者 PLL，作为 TX 用户逻辑使用的时钟以及 SERDES TX 用户时钟 I*_tx_usrclk 或 I*_tx_usr2clk 。在 IP Generator 中，通常使用 I*_tx_out2clk，而将 I*_tx_outclk 悬空。 |

### 3.3.3 I*_tx_outclk/I*_tx_out2clk 驱动 TX Fabric Interface

I*_tx_outclk、I*_tx_out2clk 输出的时钟可以送到全局时钟走线（通过 BUFG 或 PLL），也可以送到区域时钟走线（通过 LCLK），用于驱动相应的用户逻辑，以及 TX Fabric Interface 接口的输入时钟 I*_tx_usrclk 或者 I*_tx_usr2clk。在 TD 软件 IP Generator 中，通常使用 I*_tx_out2clk 这个时钟来产生用户逻辑所需要的时钟，而将 I*_tx_outclk 悬空。以下分别介绍使用 BUFG，LCLK 和 PLL 的接法。

#### 3.3.3.1. 使用 BUFG 产生用户时钟

单个通道使用 BUFG 产生用户时钟接法如图 3-4。在 SERDES 内部 TX PLL VCO 经过一系列分频可以得到 I*_tx_out2clk，该时钟输出送给 BUFG，BUFG 输出时钟驱动 TX 用户逻辑，同时送回给 I*_tx_usr2clk 作为用户逻辑 TX 并行数据的采样时钟，将用户逻辑的数据输入 SERDES 内部进行处理。此时 I*_tx_outclk 可以悬空，I*_tx_usrclk 可以接地，这两个时钟未使用。

若用户逻辑有需要，I*_tx_outclk 也可以输出给时钟 BUFFER，即 SERDES 可以多提供一路输出时钟给用户逻辑使用。若用户逻辑不需要，将其悬空即可。

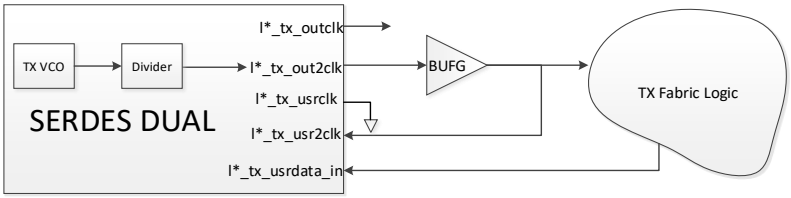


图 3-4 单个 Channel 用 BUFG 驱动 I*_tx_usr2clk 和用户逻辑的时钟方案

若使用多个 Channel 的协议，在各个 Channel 被同源时钟驱动的前提下，可以使用一个 Lane 的 I*_tx_out2clk 驱动多个 Lane 的 I*_tx_usr2clk。N 条 lane 的 Multi channel link 中，使用 BUFG 驱动用户逻辑的时钟接法如图 3-5 所示。

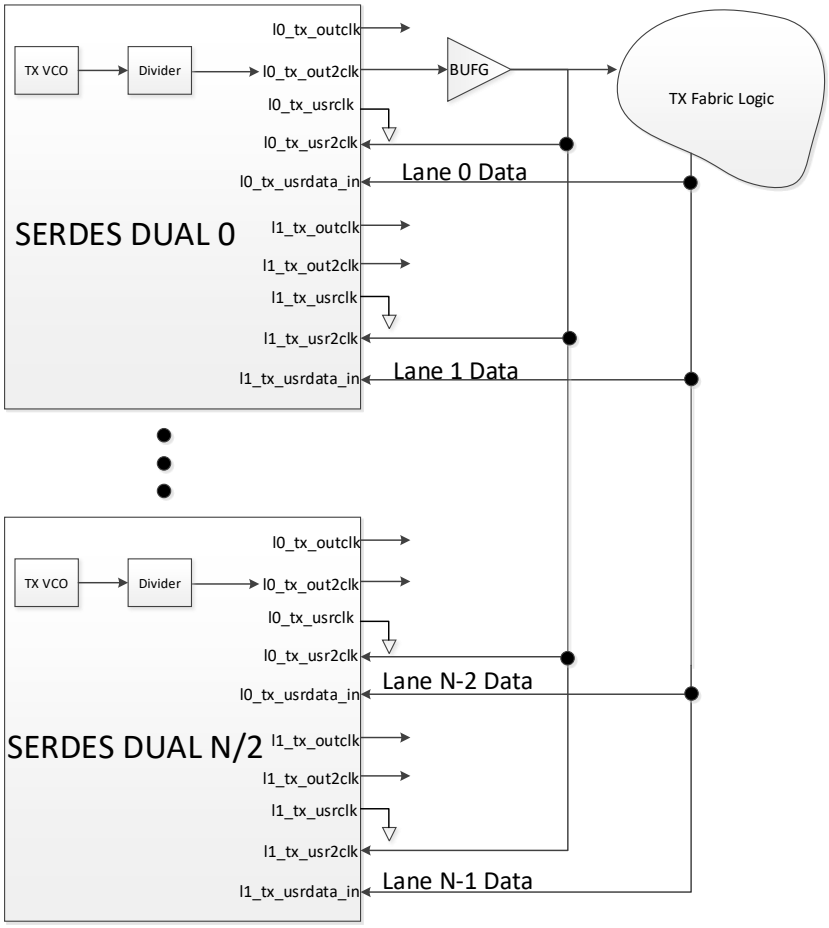


图 3-5 Multi Channel xN 模式用 BUFG 驱动 I*_tx_usr2clk 和用户逻辑的时钟方案

图 3-5 中，I0_tx_out2clk 作为时钟源驱动各个 Lane 的 I0_tx_usr2clk 和 I1_tx_usr2clk，也可以使用 I1_tx_out2clk 作为时钟源驱动 I0_tx_usr2clk 和 I1_tx_usr2clk。

#### 3.3.3.2. 使用 LCLK 产生用户时钟

单个通道使用区域时钟资源 LCLK 产生用户时钟接法如图 3-6 所示。在 SERDES 内部 TX PLL VCO 经

过一系列分频可以得到 l*_tx_out2clk，该时钟输出送给 LCLK 的时钟输入端 clkin。LCLK 输出时钟有两路，一路是 clkout，一路是 clkdivout。clkout 走区域时钟走线，而 clkdivout 通过 BUFG 走全局时钟走线。图中使用 clkout 端口（不经过 BUFG）驱动 TX 用户逻辑，同时送回给 l*_tx_usr2clk 作为用户逻辑 TX 并行数据的采样时钟，将用户逻辑的数据输入 SERDES 内部进行处理。此时 l*_tx_outclk 可以悬空，l*_tx_usrclk 可以接地，这两个时钟未使用。

若用户逻辑有需要，l*_tx_outclk 也可以输出给 LCLK 或其他时钟 BUFFER，即 SERDES 可以多提供一路输出时钟给用户逻辑使用。若用户逻辑不需要，将其悬空即可。

LCLK 和 BUFG 的区别是，LCLK 的 clkout 输出时钟用于驱动本时钟域的用户逻辑。若使用 LCLK 驱动时钟域之外的用户逻辑，会有较大的 clock skew，用户设计应避免使用 clkout 输出的时钟驱动本时钟域之外的用户逻辑。LCLK 和 BUFG 相比的另一个区别是 LCLK 具有分频功能，它可以提供 1-8 等几种分频，而不用依靠 SERDES 内部分频功能。LCLK 的 clkout 和 clkdivout 都具有分频功能，具体用法见《UG910_PH1A Series FPGA Libraries Guide for HDL Designs》。

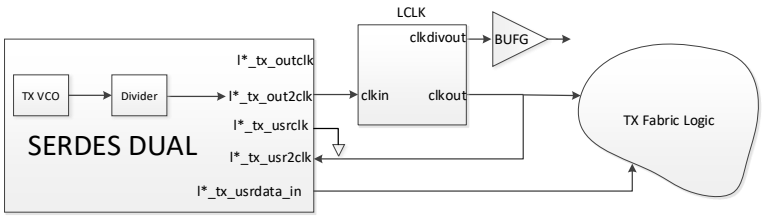


图 3-6 单个 Channel 用 LCLK 驱动 l*_tx_usr2clk 和用户逻辑的时钟方案

若使用 Multi channel 的协议，可以使用一个 Lane 的 l*_tx_out2clk 经过 LCLK 驱动多个 Lane 的 l*_tx_usr2clk。在一个时钟区域内，共有 4 个 LCLK 资源。在 PH1A 系列 FPGA 器件中，2 个 SERDES DUAL 分布在一个时钟区域内，这两个 SERDES DUAL 共用这 4 个 LCLK 资源。因此使用 LCLK 最多仅能驱动 2 个位于同一个时钟区域内的 SERDES DUAL，最多 4 条 Lane，其时钟接法如图 3-7 所示。

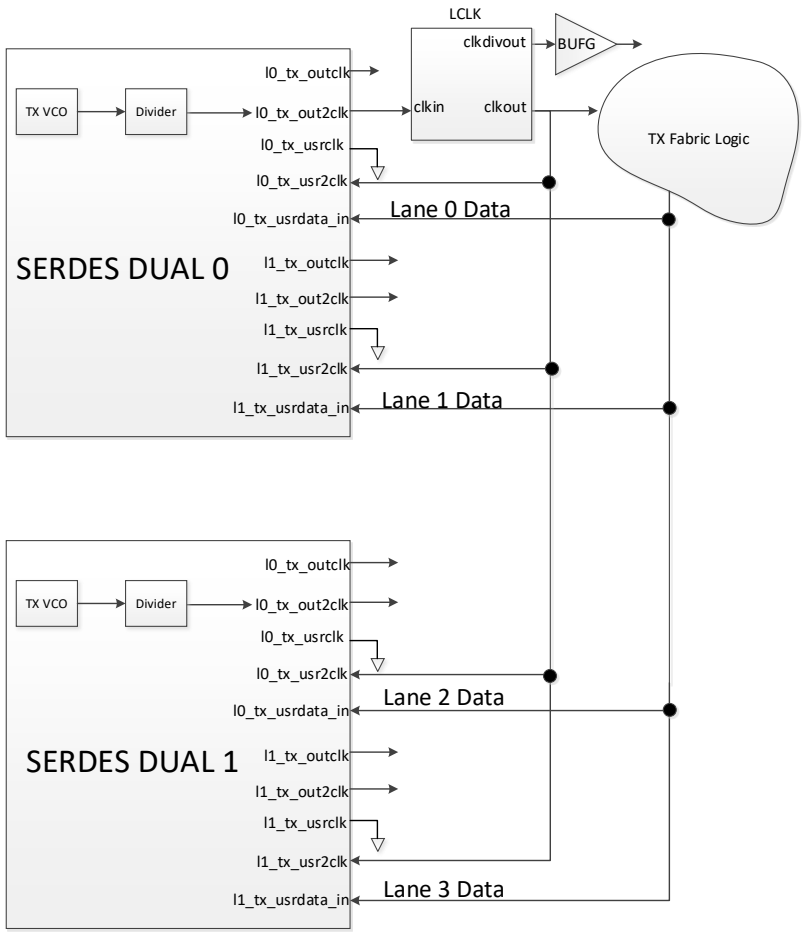


图 3-7 Multi Channel x4 模式用 LCLK 驱动 I*_tx_usr2clk 和用户逻辑的时钟方案

图 3-7 中，I0_tx_out2clk 作为时钟源驱动各个 Lane 的 I0_tx_usr2clk 和 I1_tx_usr2clk，也可以使用 I1_tx_out2clk 作为时钟源驱动 I0_tx_usr2clk 和 I1_tx_usr2clk。

LCLK 主要用在用户逻辑比较少，一个时钟区域足以容纳的情形；另外使用 LCLK 驱动 TX 用户逻辑的方案还可以用于 TX PCS FIFO Bypass 的场景，因为 LCLK 输入和输出时钟走线都很短，可以认为输入和输出时钟相位差很小，因此 SERDES 内部时钟和外部时钟相位很接近。

#### 3.3.3.3. 使用 PLL 产生用户时钟

单个通道使用 PLL 产生用户时钟接法如图 3-8 所示。在 SERDES 内部 TX PLL VCO 经过一系列分频可以得到 I*_tx_outclk 和 I*_tx_out2clk，这两个时钟其中任何一路可以送给 PLL 的 refclk，PLL 输出时钟 clkc 驱动 BUFG 产生用户逻辑所需时钟，同时送回该时钟给 I*_tx_usr2clk 作为用户逻辑 TX 并行数据的采样时钟，将用户逻辑的数据输入 SERDES 内部进行处理。此时 I*_tx_outclk 可以悬空，I*_tx_usrclk 可以接地，这两个时钟未使用。

若用户逻辑有需要，I*_tx_outclk 或 I*_tx_out2clk 也可以输出给时钟 BUFFER，即 SERDES 可以多提供一路输出时钟给用户逻辑使用。若用户逻辑不需要时钟这个时钟，将其悬空即可。

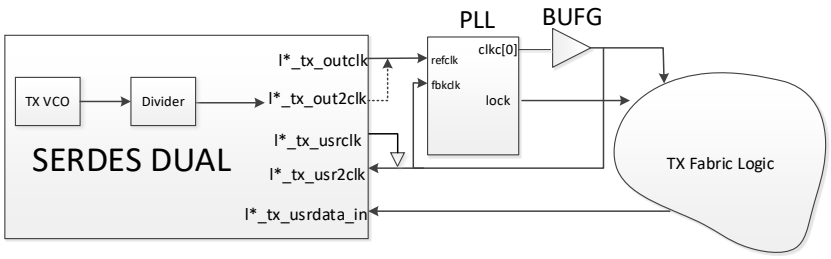


图 3-8 单个 Channel 用 PLL 再驱动 l*_tx_usr2clk 和用户逻辑的时钟方案

图中 l0_tx_out2clk 处虚线表示 l0_tx_outclk 和 l0_tx_out2clk 都可以驱动 PLL 的 refclk 输入端口。

在用户的设计中，多个 SERDES Channel 作为一个高速协议连接来用，如何用其中一个 Channel 的 l*_tx_outclk 或者 l*_tx_out2clk，经由 PLL 来驱动多个 Channel 的 l*_tx_usrclk 或 l*_tx_usr2clk，请见图 3-9。

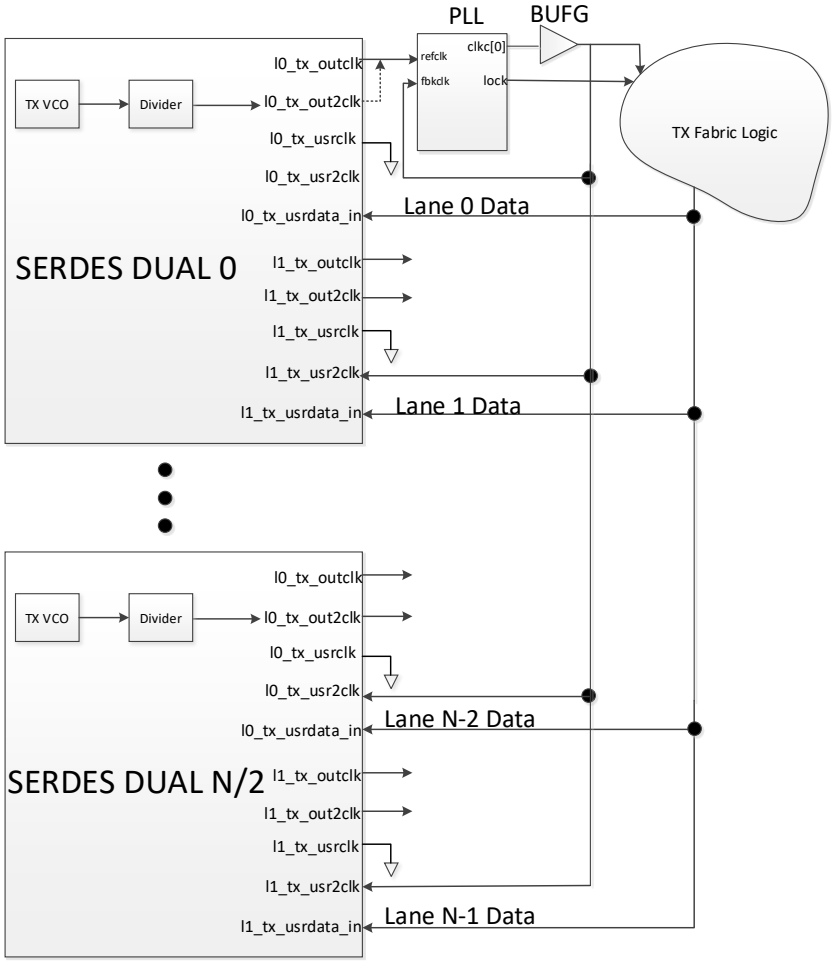


图 3-9 Multi Channel 协议使用 PLL 驱动用户逻辑和 l*_tx_usr2clk 的时钟方案

图 3-9 中，l0_rx_out2clk 作为时钟源驱动各个 Lane 的 l0_rx_usr2clk 和 l1_rx_usr2clk，也可以使用 l1_rx_out2clk 作为时钟源驱动 l0_rx_usr2clk 和 l1_rx_usr2clk。图中 l0_rx_out2clk 处虚线表

示 IO_rx_outclk 和 IO_rx_out2clk 都可以驱动 PLL 的 refclk 输入端口。

## 3.4 TX PCS FIFO

### 3.4.1 功能简介

TX PCS 数据通路有 2 个内部并行时钟域：TX PCS 并行时钟域 I*_tx_xclk 和 I*_tx_usr2clk。为了完整的发送数据，I*_tx_xclk 必须匹配用户时钟速率。这两个时钟域之间的相位差问题必须解决。

TX Standard PCS 和 TX Native PCS 内置 TX PCS FIFO 来解决 I*_tx_xclk 和用户时钟之间的相位问题。

TX PCS FIFO 有单独的复位信号 I*_tx_pcsfifo_rst_n。当 TX PCS FIFO 发生空或满状态时，会发出错误指示信号 I*_tx_pcsfifo_overflow 和 I*_tx_pcsfifo_underflow，分别指示下溢出和上溢出。

当用户逻辑的时钟跟 TX PCS 并行时钟域(I*_tx_xclk)是同一个时钟时，没有相位差，TX PCS FIFO 可以被旁路。本模块有以下特性：

支持对输入数据和输入控制信号进行相位补偿。

支持最大的输入数据有效位宽 80bit。

支持对 TX PCS FIFO 模块进行 bypass 处理，bypass 时数据延时较小。

### 3.4.2 端口和属性

TX PCS FIFO 相关参数及属性分别如表 3-5、表 3-6 所示。

表 3-5 TX PCS FIFO 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| I*_tx_pcsfifo_rst_n | In | cr_para_clk | TX PCS FIFO 复位信号，低有效，低电平脉冲宽度不能少于 20ns。 |
| I*_tx_pcsfifo_en | In | cr_para_clk | TX PCS FIFO 使能信号，高有效。 |
| I*_tx_dff_en | In | cr_para_clk | Tx DFF 使能信号，在 TX PCS FIFO 不使能时，如果该信号为高，则输入数据经过一级 tx_iclk 驱动的 DFF。 |
| I*_tx_pcsfifo_delay[3:0] | Out | I*_tx_xclk | 数据经过 TX PCS FIFO 的延迟指示。 |
| I*_tx_pcsfifo_overflow | Out | I*_tx_usr2clk | TX PCS FIFO 模块出现上溢错误指示信号，高电平有效。 |
| I*_tx_pcsfifo_underflow | Out | I*_tx_usr2clk | TX PCS FIFO 模块出现下溢错误指示信 |

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| | | | 号，高电平有效。 |

表 3-6 TX PCS FIFO 属性列表

| 属性名 | 类型和宽度 | 说明 |
|---|---|---|
| L*_TX_PCSFIFO_RD_INIT | 3 | TX PCS FIFO 模块初始读地址，该信号设置不同，可以使得 FIFO 的延迟不同。 |

## 3.4.3 用法

TX Native PCS 中的 TX PCS FIFO 功能与 TX Standard PCS 中的相同。

不同之处在于，TX Native PCS 的 FIFO 必须在 TX Native PCS 使能时才能使用，而 TX Standard PCS 中的 TX PCS FIFO 必须在 TX Standard PCS 使能时才能使用。

TX Native PCS 的 FIFO 的端口和属性跟 TX Standard PCS 中 TX PCS FIFO 一样。

当 TX PCS FIFO 发生溢出时，包括上溢出和下溢出，都要对 TX PCS FIFO 进行复位。

### 3.4.3.1. TX PCS FIFO Bypass 用法

TX PCS 主要有两个时钟域，一个是 TX XCLK 时钟域，一个是 l*_tx_usrclk/l*_tx_usr2clk，见图 3-2。通常 TX XCLK 时钟可以选择 l*_tx_outclk，l*_tx_usrclk/l*_tx_usr2clk 时钟是由 l*_tx_outclk 或者 l*_tx_out2clk 经过时钟 BUFFER 产生的，因此 l*_tx_usrclk/l*_tx_usr2clk 时钟和 TX XCLK 时钟是有相位差的，TX PCS FIFO 的作用是解决这两个时钟域之间的相位差。当 TX PCS FIFO 被旁路时，用户设计需解决这两个时钟的相位差。

解决这个相位差有几个办法，一个是由 l*_tx_outclk 通过 LCLK 来产生 l*_tx_usrclk/l*_tx_usr2clk 时钟，另一个办法是用 PLL 解决 l*_tx_usrclk/l*_tx_usr2clk 和 l*_tx_outclk 之间的相位差。

**TX PCS FIFO Bypass - LCLK**

使用 LCLK 解决 TX PCS FIFO Bypass 的时钟连接方法见图 3-10。

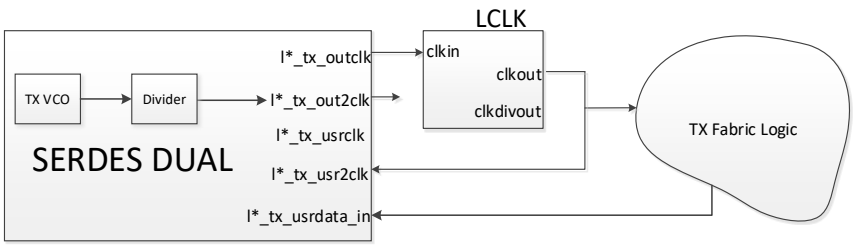


图 3-10 TX PCS FIFO Bypass 模式下用 LCLK 产生用户时钟

首先，将 l*_tx_outclk 送入 LCLK 的 clkin，不能使用 l*_tx_out2clk 作为 LCLK 的输入时钟。然后，l*_tx_outclk 频率需要配置成和 l*_tx_usr2clk 频率相等，因为 LCLK 的分频功能 clkdivout 在这

里不能使用。最后，LCLK 的 clkout 输出端口送回 l*_tx_usr2clk 同时驱动用户逻辑。

**TX PCS FIFO Bypass - PLL**

使用 PLL 解决 TX PCS FIFO Bypass 的时钟连接方法见图 3-11。

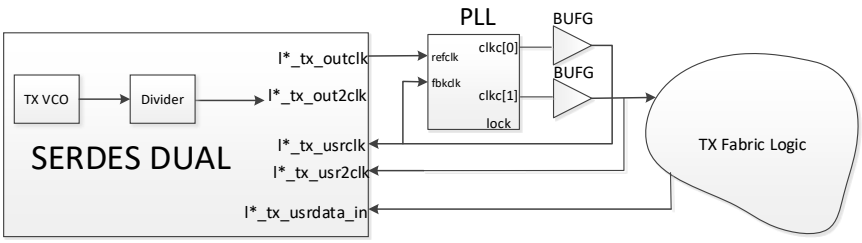


图 3-11 TX PCS FIFO Bypass 模式下用 PLL 产生用户时钟

l*_tx_outclk 送入 PLL 的 refclk 输入端口，PLL 的 2 个时钟输出端口分别输出 l*_tx_outclk 同频时钟，和 1，2 或 4 分频时钟（根据内外位宽比例选择），分频时钟送给 l*_tx_usr2clk 和用户逻辑，同频时钟送给 l*_tx_usrclk 和 PLL 的 fbkclk 输入，设置 PLL 输出时钟和输入时钟相位对齐模式。

## 3.5 TX Byte Serializer

### 3.5.1 功能简介

当 TX PCS 用户侧数据位宽是 TX PCS 内部数据位宽的倍数时，TX PCS 需要实现位宽转换。本 TX Byte Serializer 能支持 2:1 和 4:1 两种数据位宽转换。当位宽比例为 1:1 时，用户必须设置本 TX Byte Serializer 为旁路。TX PCS 内部数据位宽的设置，请见 TX Channel 的 TX Serializer 模块。

本模块的特性如下：

支持对输入数据和输入控制信号进行位宽转换。

支持对输入数据按字节(8bits/10bits/16bits/20bits)为粒度进行位宽转换。

支持对 80 位有效位宽的输入数据按照 20bits 为粒度进行 4:1 的并串位宽转换。

支持对 64 位有效位宽的输入数据按照 16bits 为粒度进行 4:1 的并串位宽转换。

支持对 40 位有效位宽的输入数据按照 20bits、10bits 为粒度进行 2:1，4:1 的并串位宽转换。

支持对 32 位有效位宽的输入数据按照 16bits、8bits 为粒度进行 2:1，4:1 的并串位宽转换。

支持对 20 位有效位宽的输入数据按照 10bits 为粒度进行 2:1 的并串位宽转换。

支持对 16 位有效位宽的输入数据按照 8bits 为粒度进行 2:1 的并串位宽转换。

支持对 20/16/10/8 位有效位宽的输入数据通过配置 byteserial_en 为低，bypass 该模块，使得输出数据等于输入数据。

TX Byte Serializer 模块输出的数据位宽，就是 TX PCS 的内部数据位宽，跟 TX PMA 和 TX PCS 接

口的数据位宽一致。当本模块被使能时，输出的数据格式是 8bit@（TX 线速/10），或 16bit@（TX 线速/20）。TX Byte Serializer 位宽转换比例设置，请见表 3-7。

表 3-7 TX Byte Serializer 位宽转换列表

| TX PCS 模块控制信号 | TX Fabric Interface 数据位宽控制信号 | TX Channel 用户侧数据 | TX Channel | 位宽比例 |
|---|---|---|---|---|
| l*_pcs_standard_en, l*_pcs_native_en, l*_tx_enc_en | l*_tx_usrdata_width [2:0] | l*_tx_usrdata_in [79:0] | 内部数据位宽 | |
| l*_pcs_standard_en=1’b1 且 l*_pcs_native_en = 1’b0 且 l*_tx_enc_en = 1’b1 | 3’b000 | 8 | 10 | 1:1 |
| | 3’b010 | 16 | 10/20 | 2:1/1:1 |
| | 3’b100 | 32 | 10/20 | 4:1/2:1 |
| | 3’b110 | 64 | 20 | 4:1 |
| l*_pcs_standard_en=1’b1 且 l*_pcs_native_en=1’b0 且 l*_tx_enc_en = 1’b0, 或 l*_pcs_standard_en = 1’b0 且 l*_pcs_native_en = 1’b1 | 3’b000 | 8 | 8 | 1:1 |
| | 3’b001 | 10 | 10 | 1:1 |
| | 3’b010 | 16 | 8/16 | 2:1/1:1 |
| | 3’b011 | 20 | 10/20 | 2:1/1:1 |
| | 3’b100 | 32 | 8/16 | 4:1/2:1 |
| | 3’b101 | 40 | 10/20 | 4:1/2:1 |
| | 3’b110 | 64 | 16 | 4:1 |
| | 3’b111 | 80 | 20 | 4:1 |

## 3.5.2 端口和属性

TX Byte Serializer 端口说明请见表 3-8。

表 3-8 TX Byte Serializer 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| l*_tx_byteserial_en | In | cr_para_clk | TX Byte Serializer 使能信号，高有效。<br>当低电平时，Byte Serializer 被旁路。 |
| l*_tx_byteserial_mode | In | cr_para_clk | TX Byte Serializer 模式设置。仅当 l*_tx_byteserial_en 为高时有效。<br>• 0：表示并串比为 2:1;<br>• 1：表示并串比为 4:1。 |

## 3.5.3 特别说明

TX Native PCS 中的 Byte Serializer 功能与 TX Standard PCS 中的相同。

不同之处在于，TX Native PCS 中的 Byte Serializer 必须在 TX Native PCS 使能时才能使用，而 TX Standard PCS 中的 Byte Serializer 必须在 TX Standard PCS 使能时才能使用。

TX Native PCS 中的 Byte Serializer 的端口和属性跟 TX Standard PCS 中 Byte Serializer 一样。

## 3.6 8B10B Encoder

### 3.6.1 功能简介

8B/10B 是业界标准的编码机制。这种编码机制，有 2bit 的开销，目的是保持直流平衡和丰富的数据翻转，以便于时钟恢复。TX Standard PCS 含有 8B/10B Encoder 模块。如果用户需要用 8B/10B 编码，请使能本模块；如果不需要，本模块可以被设置为旁路。

本模块的特性如下：

支持对 8bits/16bits 有效数据进行 8B/10B 编码。

支持对 8B/10B 编码码块根据 encode_8b10b_en 信号，进行 bypass。

支持用户强制的设置当前编码极性（disparity）。

8B/10B Encoder 模块能把 8bit 编码成 10bit。8B/10B Encoder 模块的输入为 8bit 的数据和 1bit 的 K 字符指示信号，输出为编码后的 10bit 数据字符或控制字符（K 字符）。

一个 TX PCS 的 8B/10B Encoder 模块，能同时实现 1 byte 或 2 byte 的 8bit 到 10bit 的编码。8B/10B Encoder 模块可以被使能，也能被旁路。当 8B/10B Encoder 模块被使能时，TX Channel 用户侧的数据位宽必须是 8bit 的倍数，如 8/16/32/64bit。

8B/10B Encoder 模块支持自动调整极性（running disparity）规则，也支持用户强制 disparity 控制。

### 3.6.2 端口和属性

8B/10B Encoder 端口说明见表 3-10。

表 3-9 8B/10B Encoder 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| l*_tx_enc_en | In | cr_para_clk | 8B/10B Encoder 使能，高有效。 |
| l*_tx_usrdata_in [79:0] | In | l*_tx_usr2clk | TX PCS 发送端用户侧数据，有效位宽根据 l*_tx_usrdata_width 设置。<br>• l*_tx_usrdata_width=0, l*_tx_usrdata_i |

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| | | | n[7:0]有效；<br>• l*_tx_usrdata_width=1，l*_tx_usrdata_in[9:0]有效；<br>• l*_tx_usrdata_width=2，l*_tx_usrdata_in[15:0]有效；<br>• l*_tx_usrdata_width=3，l*_tx_usrdata_in[19:0]有效；<br>• l*_tx_usrdata_width=4，l*_tx_usrdata_in[31:0]有效；<br>• l*_tx_usrdata_width=5，l*_tx_usrdata_in[39:0]有效；<br>• l*_tx_usrdata_width=6，l*_tx_usrdata_in[63:0]有效；<br>• l*_tx_usrdata_width=7，l*_tx_usrdata_in[79:0]有效。 |
| l*_tx_usrdata_width[2:0] | In | l*_tx_usr2clk | 发送端用户侧数据有效位宽。<br>• 0 表示 8bit 有效；<br>• 1 表示 10bit 有效；<br>• 2 表示 16bit 有效；<br>• 3 表示 20bit 有效；<br>• 4 表示 32bit 有效；<br>• 5 表示 40bit 有效；<br>• 6 表示 64bit 有效；<br>• 7 表示 80bit 有效。 |
| l*_tx_usrdatak_in[7:0] | In | l*_tx_usr2clk | K 字符指示，高有效。<br>• l*_tx_usrdatak_in[7] 对应 l*_tx_usrdata_in[63:56]；<br>• l*_tx_usrdatak_in[6] 对应 l*_tx_usrdata_in[55:48]；<br>• l*_tx_usrdatak_in[5] 对应 l*_tx_usrdata_in[47:40]；<br>• l*_tx_usrdatak_in[4] 对应 l*_tx_usrdata_in[39:32]；<br>• l*_tx_usrdatak_in[3] 对应 |

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| | | | l*_tx_usrdata_in[31:24]；<br>• l*_tx_usrdatak_in[2] 对应 l*_tx_usrdata_in[23:16]；<br>• l*_tx_usrdatak_in[1] 对应 l*_tx_usrdata_in[15:8]；<br>• l*_tx_usrdatak_in[0] 对应 l*_tx_usrdata_in[7:0]。 |
| l*_tx_forcedisp_in [7:0] | In | l*_tx_usr2clk | 8B/10B 编码的强制校验命令，高有效，指示采用 l*_tx_forcedispval_in 作为校验值。<br>• l*_tx_forcedisp_in[0] 对应 l*_tx_forcedispval_in[0]；<br>• l*_tx_forcedisp_in[1] 对应 l*_tx_forcedispval_in[1]；<br>• l*_tx_forcedisp_in[2] 对应 l*_tx_forcedispval_in[2]；<br>• l*_tx_forcedisp_in[3] 对应 l*_tx_forcedispval_in[3]；<br>• l*_tx_forcedisp_in[4] 对应 l*_tx_forcedispval_in[4]；<br>• l*_tx_forcedisp_in[5] 对应 l*_tx_forcedispval_in[5]；<br>• l*_tx_forcedisp_in[6] 对应 l*_tx_forcedispval_in[6]；<br>• l*_tx_forcedisp_in[7] 对应 l*_tx_forcedispval_in[7]。 |
| l*_tx_forcedispval_in [7:0] | | | 从用户侧得到的 8B/10B 编码的强制校验值，高为正，低为负。当 l*_tx_forcedisp_in 对应位为高时，该值有效。<br>• l*_tx_forcedispval_in[7] 对应 l*_tx_usrdata_in[63:56]；<br>• l*_tx_forcedispval_in[6] 对应 l*_tx_usrdata_in[55:48]；<br>• l*_tx_forcedispval_in[5] 对应 l*_tx_usrdata_in[47:40]； |

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| | | | • l*_tx_forcedispval_in[4] 对应 l*_tx_usrdata_in[39:32]；<br>• l*_tx_forcedispval_in[3] 对应 l*_tx_usrdata_in[31:24]；<br>• l*_tx_forcedispval_in[2] 对应 l*_tx_usrdata_in[23:16]；<br>• l*_tx_forcedispval_in[1] 对应 l*_tx_usrdata_in[15:8]；<br>• l*_tx_forcedispval_in[0] 对应 l*_tx_usrdata_in[7:0]。 |

## 3.6.3 说明

### 3.6.3.1. 输入/输出数据流格式

输入数据：8bit@（TX 线速/10），或 16bit@（TX 线速/20）

输出数据：10bit@（TX 线速/10），或 20bit@（TX 线速/20）

### 3.6.3.2. TX 8B/10B 的 bit 和 byte 排序

8B/10 Encoder 模块输出的 bit 顺序，是符合 8B/10B 码表的顺序。8B/10B 编码规定输出的 a bit 先被发送。如果对端的接收器要求 j bit 先发、a bit 最后发送，可以在 TX Bit Slip 中的 bit 反序来适应对端。

### 3.6.3.3. K 字符

8B/10B 码表在 256 个数据字符码以外，还包括了一些特殊字符码。这些特殊码，我们称之为 K 字符。K 字符经常被用作控制信息。l*_tx_usrdatak_in 端口指示数据端口 l*_tx_usrdata_in 是数据还是 K 字符。当 l*_tx_usrdata_in 有 bit 为高时，8B/10B Encoder 会检查对应的发送数据 l*_tx_usrdatak_in 的对应 byte 是否匹配。

### 3.6.3.4. TX Running Disparity

8B/10B 编码是 DC 平衡的，这就意味着从长期来看，1 的个数和 0 的个数应该各占 50%。为了实现 DC 平衡，8B/10B Encoder 总是计算发送的 1 的个数和 0 个数的差值，并在发完一个字符时，以+1 或-1 来标记 1/0 差异。这个差异就是 Running Disparity。Disparity 翻译成中文，就是不一致性。

一些协议用不一致性来发送控制信息。为了适应这些协议，Running Disparity 不仅能用 8B/10B Encoder 产生，而且也能让用户通过端口 l*_tx_forcedisp_in 和 l*_tx_forcedispval_in 来控制。Disparity 控制示例如表 3-9 所示。

表 3-10 Disparity 控制列表

| l*_tx_forcedisp_in | l*_tx_forcedispval_in | 发出去的 Disparity |
|---|---|---|
| 0 | 0 | 由 8B/10B Encoder 计算。 |
| 0 | 1 | 当对 l*_tx_usrdata_in 进行编码时，取反 Running Disparity 。 |
| 1 | 0 | 当对 l*_tx_usrdata_in 进行编码时，强制 Running Disparity 为-1。 |
| 1 | 1 | 当对 l*_tx_usrdata_in 进行编码时，强制 Running Disparity 为+1。 |

## 3.7 TX Bit Slip

### 3.7.1 功能简介

用户在 TX Channel 里控制发送路径上 bit 级的延迟，TX Standard PCS 的 TX Bit Slip 能实现这个功能。该功能仅在 Standard PCS 模式下才能支持。

本模块特性如下：

支持对输入数据按照有效位宽进行 l*_tx_bitslip_width 位宽的左移。

支持对进行位宽左移后的数据进行 bit 反序。

支持对进行位宽左移后的数据进行 byte 反序。

支持对输入数据根据 l*_tx_bitslip_en 使能 bypass 处理。

TX Bit Slip 允许用户控制并行数据的边界，可应用于对延迟敏感的高速协议，如 CPRI 等。

TX Bit Slip 的最大可移动 TX PCS 并行数据的位宽-1 个 bit。如果并行数据位宽是 20，那么最多可移动 19 位。如果并行数据位宽是 10，那么最多可移动 9 位。Slip 方向是从 LSb 向 MSb，从当前数据项向前一个数据。图 3-12 是移动 2bit 的情况。图中的并行数据是 TX Parallel Data。每个 TX Parallel Data 假设是 10bit。

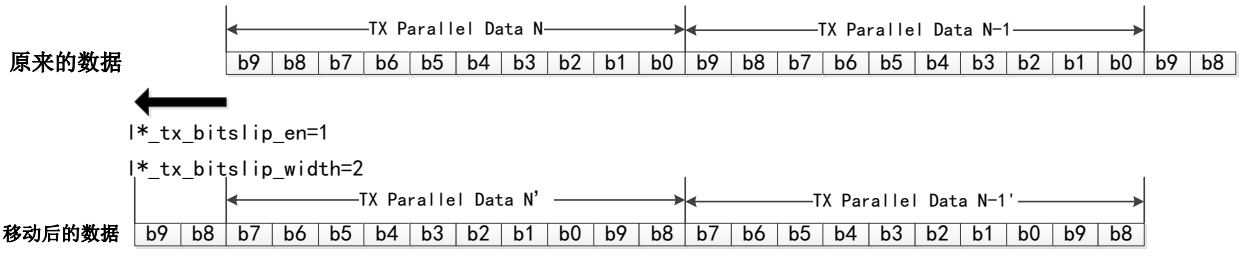


图 3-12 TX Bit Slip 示意图

此模块支持以 bit 为单位的反序。比如输出的 10bit，bit9 和 bit0 对调、bit8 和 bit1 对调、bit7 和 bit2 对调，……，以此类推。用户可用属性 TX_BITREV_EN 来控制是否做 bit 反序。

此模块还支持以 Byte 为单位的反序。当输入为 16bit 时，高 Byte 的编码在低 10bit 输出，而低 Byte 的编码在高 10bit 输出。用户可用属性 TX_BYTEREV_EN 来控制是否做 byte 反序。

### 3.7.2 端口和属性

表 3-11 TX Bit Slip 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| l*_tx_bitslip_en | In | cr_para_clk | TX Bit Slip 模块的使能，高有效。 |
| l*_tx_bitslip_width[4:0] | In | cr_para_clk | 本信号控制 TX Bit Slip 模块向左移的 bit 数。 |

表 3-12 TX Bit Slip 属性列表

| 属性 | 类型 | 描述 |
|---|---|---|
| L*_TX_BITREV_EN | 1 bit | bit 反序使能，设置为 1 时反序，默认值为 0。 |
| L*_TX_BYTEREV_EN | 1 bit | byte 反序使能，设置为 1 时反序，默认值为 0。 |

## 3.8 PRBS Generator

### 3.8.1 功能简介

PRBS(Pseudo-random bit sequences)通常被同于测试高速链路的信号完整性。这些时序几乎是随机的，但有特定的属性来测量链路的质量。仅在 TX Standard PCS 模式下，才能支持 PRBS Generator。

PRBS Generator 模块能产生多种 Pattern，除了 PRBS-7/PRBS-9/ PRBS-15/PRBS-23/PRBS31，还可以产生用户自定义的波形，如图 3-13 所示。

当 PRBS Generator 工作在用户自定义波形模式时，用户自定义波形属性参数 L*_TX_CUSTOMIZED_WAVE 为 20 位。

当 L*_TX_CUSTOMIZED_WAVE 被设置为 20’b00000000001111111111 或 20’b11111111110000000000 时，就可以产生 TX 线速/20 的方波，（前提是 TX PMA 和 TX PCS 之间的并行数据位宽是 20bit 时。）当发送并行内部位宽是 10bit 时，L*_TX_CUSTOMIZED_WAVE 仅有低 10bit 有效。

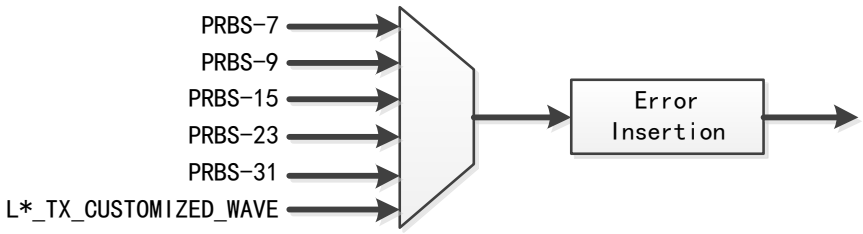


图 3-13 PRBS Generator 框图

本模块的特性如下：

支持 PRBS-7、PRBS-9、PRBS-15、PRBS-23、PRBS-31、用户自定义图案（pattern）产生。

支持对产生的 PRBS 进行插错。

支持根据 I*_tx_prbs_en 信号， 进行该模块的 bypass 处理。

## 3.8.2 端口和属性

PRBS Generator 端口及参数说明分别如表 3-13、表 3-14 所示。

表 3-13 PRBS Generator 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| I*_tx_prbs_en | In | cr_para_clk | PRBS Generator 模块使能信号，高有效 |
| I*_tx_prbs_sel[2:0] | In | cr_para_clk | PRBS Generator 模式选择<br>• 3’b000：PRBS-7；<br>• 3’b001：PRBS-9；<br>• 3’b010：PRBS-15；<br>• 3’b011：PRBS-23；<br>• 3’b100：PRBS-31；<br>• 3’b101：用户自定义的 20bits 波形；<br>• 3’b110：保留；<br>• 3’b 111：保留。 |
| I*_tx_prbs_force_err | In | cr_para_clk | PRBS Generator 插入错误的控制信号，高有效<br>当 tx_psbs_sel 为 3’b000， 3’b001，3’b010，3’b011，3’b100 时，该信号生效。 |

表 3-14 PRBS Generator 属性列表

| 属性 | 类型 | 描述 |
|---|---|---|
| L*_TX_CUSTOMIZED_WAVE | 20 bit | 本属性是用来让 PRBS Generator 产生用户自定义波形。默认值为 20'b00010001000100010001，推荐使用 IP Generator 产生的值。<br>当 I*_tx_prbs_sel 为 3’b101 时，此属性生效。 |

## 3.8.3 用法

PRBS Generator 按照表 3-15 所示的多项式产生工业标准的伪随机序列，用于测试信号连接的质量。

表 3-15 PRBS Generator 支持的 PRBS Pattern

| 名称 | 多项式 | 序列长度 | 描述 |
|---|---|---|---|
| PRBS-7 | $1 + X^6 + X^7$ | $2^7 - 1$ bits | 用于测试使用 8B10B 协议的通道 |
| PRBS-9 | $1 + X^5 + X^9$ | $2^9 - 1$ bits | 见 ITU-T O.150, Section 5.1。PRBS-9 也是 SFP+推荐的测试码型之一。 |
| PRBS-15 | $1 + X^{14} + X^{15}$ | $2^{15} - 1$ bits | 见 ITU-T O.150, Section 5.3。通常用于 Jitter 测量。 |
| PRBS-23 | $1 + X^{18} + X^{23}$ | $2^{23} - 1$ bits | 见 ITU-T O.150, Section 5.6。用于测试非 8B10B 的应用。PRBS-23 也是 SONET 推荐的测试码型之一。 |
| PRBS-31 | $1 + X^{28} + X^{31}$ | $2^{31} - 1$ bits | 见 ITU-T O.150, Section 5.8。用于测试非 8B10B 的应用。PRBS-31 也是 10G 以太网推荐的测试码型之一。见 EEE 802.3ae-2002。 |

如有必要，可以用 tx*_invert 参数来控制码型的极性。进入或退出 TX PRBS Generator，即 l*_tx_prbs_en 发生了高低变化之后，需对 l*_tx_pcs_rst_n 进行复位，也可以复位 tx*_reset 并带动 l*_tx_pcs_rst_n 复位。

## 3.9 TX Polarity Control

### 3.9.1 功能简介

PCB 上 SERDES 输出的差分对，如果硬件设计上必须反转 P 和 M 端口的极性，发送端支持 P 和 M 极性取反。TX Polarity Control 模块实现这一功能。

### 3.9.2 端口和属性

TX Polarity 控制相关端口和属性分别如表 3-16、表 3-17 所示。

表 3-16 TX Polarity Control 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| tx*_invert | In | cr_para_clk | TX PMA 里的 TX 极性反转的控制信号，高有效。 |

表 3-17 TX Polarity Control 属性列表

| 属性 | 类型 | 描述 |
|---|---|---|
| L*_TX_POLARITYINV_EN | 1 bit | 控制 TX PCS 里的 TX Polarity 极性翻转的属性，保留，设 |

| | | |
|---|---|---|
| | | 置为 0。推荐使用 PMA 的 TX 极性反转的控制信号。 |

### 3.9.3 特别说明

用户在 TX 方向，可以设置 Polarity 极性反转， tx*_invert 设置为 1，TX Polarity 反转。

## 3.10 TX 串行器（TX Serializer）

### 3.10.1 功能简介

TX Serializer 将 SERDES 内部数据从并行转成串行数据送出芯片外部。

### 3.10.2 端口和属性列表

TX Serializer 相关端口如表 3-18 所示。

表 3-18 TX Serializer 相关端口

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| l*_tx_pmadata_width[1:0] | In | Async | PMA 并行数据位宽设置，保留。推荐使用 IP Generator 产生的值。用户不能修改。信号值对应的位宽如下。<br>• 2'b00: 8-bit;<br>• 2'b01: 10-bit;<br>• 2'b10: 16-bit;<br>• 2'b11: 20-bit。 |

## 3.11 TX 幅度控制

### 3.11.1 功能简介

TX 输出峰峰值摆幅 Swing 即 Vswing，一般情况下不超过或者接近于 PHYVCCT 即 0.9V （典型值）。

发送摆幅还支持 BOOST 功能，该功能被 TX*_VBOOST_EN 使能，使能 VBOOST 之后可以输出更高的差分摆幅。最高摆幅由 TX_VBOOST_LVL 来控制。当 TX_VBOOST_LVL 值为 3’ b101 时，差分摆幅最大值不超过或接近于 1100mV。

### 3.11.2 端口和属性列表

TX 幅度控制相关属性列表如表 3-19 所示。

表 3-19 幅度控制相关属性

| 属性名 | 类型和宽度 | 说明 |
| --- | --- | --- |
| TX*_IBOOST_LVL | 4 | 设置 iboost 幅度。<br>当 TX*_VBOOST_EN 为高时，和 TX*_IBOOST_LVL[2:0]联合控制每条 lane 的 TX Swing 幅度。<br>默认值为 4’b0000，推荐使用 IP Generator 产生的值。 |
| TX*_VBOOST_EN | 1 | 使能 Voltage Boost，高有效。<br>推荐使用 IP Generator 产生的值。 |
| TX_VBOOST_LVL | 3 | 设置 vboost 幅度。<br>和 TX*_IBOOST_LVL 共同决定实际的摆幅 Vswing。<br>默认值为 3’b111，推荐使用 IP Generator 产生的值。 |

## 3.11.3 用法

通常情况下，当 TX*_VBOOST_EN=0 时，发送摆幅受到 PHYVCCT 限制，不会超过 PHYVCCT。某些情况下，当需要更大的摆幅时，需要设置 TX_VBOOST_EN=1，使能 vboost 功能增加摆幅。此时，TX*_IBOOST_LVL 和 TX_VBOOST_LVL 共同决定实际的摆幅。

当 TX*_IBOOST_LVL [3:0]='1111'最大值时，不同的 TX_VBOOST_LVL 值对应的 Vswing 最大化的典型值如表 3-20 所示。

表 3-20 Typical Maximum TX Swing (Vswing) When TX*_IBOOST_LVL [3:0]='1111'

| TX_VBOOST_LVL [2:0] | Amplitude (mVppd) |
| --- | --- |
| 000 | PHYVCCT |
| 001 | PHYVCCT |
| 010 | 保留 |
| 011 | 900 |
| 100 | 1000 |
| 101 | 1100 |
| 110 | 不推荐 |
| 111 | 不推荐 |

当 TX_VBOOST_LVL[2:0]固定为 5 的时候，不同的 TX*_IBOOST_LVL [3:0]相对应的 Vswing 的平均值如表 3-21 所示。

表 3-21 iboost 对 Vswing 典型值对应关系

| TX*_IBOOST_LVL [3:0] | Amplitude (mVppd) |
| --- | --- |
| 4'b0000 | 854 |

| TX*_IBOOST_LVL [3:0] | Amplitude (mVppd) |
|---|---|
| 4'b0010 | 888 |
| 4'b0100 | 923 |
| 4'b0110 | 958 |
| 4'b1000 | 989 |
| 4'b1010 | 1022 |
| 4'b1100 | 1055 |
| 4'b1110 | 1084 |

注：此表格中的数据是 PVT 测试的平均值，测试条件为：

TX*_VBOOST_EN=1’b0；TX_VBOOST_LVL=3’b101；tx*_eq_main=6’d40； tx*_eq_post =6’d0；tx*_eq_pre =6’d0。

随着测试样本和数据不断增加，表格中的具体数据可能会有变化。

**调节发送摆幅的方法：**

当 TX*_VBOOST_EN=0 时，通过调节 MAIN CURSOR，POST CURSOR 和 PRE CURSOR 来控制摆幅，具体参考 3.12.3 这个章节。

当 TX*_VBOOST_EN=1 时，通过控制 TX*_IBOOST_LVL [3:0]来进行细调，细调的幅度参考表 3-17。

## 3.12 TX Equalizer

### 3.12.1 功能简介

发送驱动器实现了 3-tap 的 FFE（feed-forward equalization），用于改善信号质量，提高信号完整性。

图 3-14 是发送驱动器 FFE 示意图。

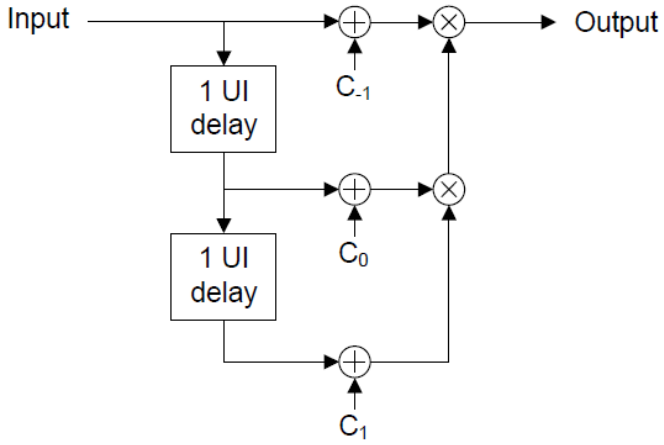


图 3-14 3-tap FFE

图中，C0，C-1，和 C1 是 TX Driver 的 main tap，pre-cursor tap，和 post-cursor tap 系数。其中

C0 = tx*_eq_main[5:0]

|C-1| = tx*_eq_pre[5:0]/4

|C1| = tx*_eq_post[5:0]/4

## 3.12.2 端口和属性列表

TX FFE 端口说明如表 3-22 所示。

表 3-22 TX FFE 端口列表

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| tx*_eq_main[5:0] | In | Async | 发送 main cursor 系数 C0，取值范围整数（0-40）。 |
| tx*_eq_post[5:0] | In | Async | 发送 post cursor 系数 C1，取值范围如下。<br>• tx*_eq_post[5:2]：Integer value (0 to 15) 整数部分；<br>• tx*_eq_post[1:0]：Fraction value (0, 0.25, 0.5, 0.75) 小数部分。<br>注解：<br>最大 post value 是 15。如果 tx*_eq_post[5:2]设置为 15，那么小数部分 tx*_eq_post[1:0]必须是 0；否则， tx*_eq_post[5:2]必须是 14 或者更小。 |
| tx*_eq_pre[5:0] | In | Async | 发送 pre cursor 系数 C-1，取值范围如下。<br>• tx*_eq_pre[5:2]：Integer value (0 to 10) 整 |

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| | | | 数部分；<br>• tx*_eq_pre[1:0]: Fraction value (0, 0.25, 0.5, 0.75)小数部分。<br>注解：<br>最大 post value 是 10。如果 tx*_eq_post[5:2]设置为 10，那么小数部分 tx*_eq_post[1:0]必须是 0；否则， tx*_eq_post[5:2]必须是 9 或者更小。 |

### 3.12.3 用法

发送方向的 pre-amplitude，post-amplitude，和 DC Swing 如 3-15 图所示。

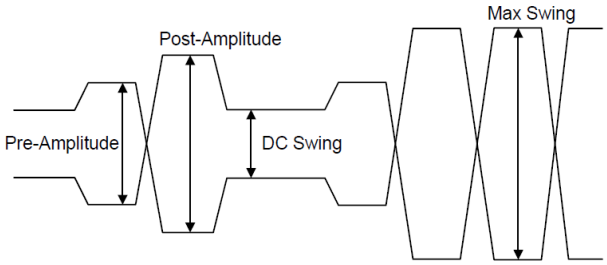


图 3-15 发送均衡效果

计算差分 pre-amplitude, post-amplitude, 和 DC Swing 用下面的公式：

Pre-amplitude = Vswing * (C0 + |C-1| - | C1|)/40

Post_amplitude = Vswing * (C0 - |C-1| + |C1|)/40

DC_swing = Vswing * (C0 - |C-1| - |C1|)/40

Max_Swing = Vswing * (C0 + |C-1| + |C1|)/40

如果要最大的 Swing 输出，设置 C0 + |C-1| + |C+1| = 40，这时 Max_Swing = Vswing。 要减小发送 Swing，把 C0 + |C-1| + |C1|设置到 40 以下即可。比如，在 C0 + |C-1| + |C1| = 20 的情况下，输出 Swing 减半。

发送摆幅和预加重的设置，必须通过 serdes-AMI 仿真进行优化，并上板进行实测验证。

## 3.13 TX PCIe Beacon 和 TXDETECTRX

### 3.13.1 端口和属性列表

Beacon 和 DetectRX 端口说明如表 3-23 所示。

表 3-23 Beacon 和 DetectRX 端口列表

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| tx*_beacon_en | In | Async | 使能发送 Beaconing。高有效。发送脉冲的周期介于 20-50ns 之间。当这个信号拉低后，Beacon osc 会完成当前的周期，然后再停止 Beaconing。 |
| tx*_detrx_req | In | Async | 发送探测接收请求 TX Detect RX<br>请求发送以后，等待 tx*_ack 拉高。在 tx*_ack 拉高时，探测结果出现在 tx*_detrx_result。<br>注释：<br>这个信号一直拉高，直到 tx*_ack 输出拉高，然后 拉低 tx*_detrx_req。RX 探测只能工作在 TX 为 P1 Power state。 |
| tx*_detrx_result | Out | Async | 发送探测接收结果。<br>这个信号在 tx*_ack 拉高后有效。<br>• 1'b0：接收不在位；<br>• 1'b1：接收在位。 |

### 3.13.2 用法

在 tx*_pstate 为 P1 时，拉高 tx*_detrx*_req 输入发起接收器探测请求。请求结果出现在 tx*_detrx_result 这个输出端口上，这个端口在 tx*_ack 拉高时有效。如果 tx*_detrx_result 为高电平，说明接收器在位；反之，接收器不在位。这个握手过程如图 3-16 所示。

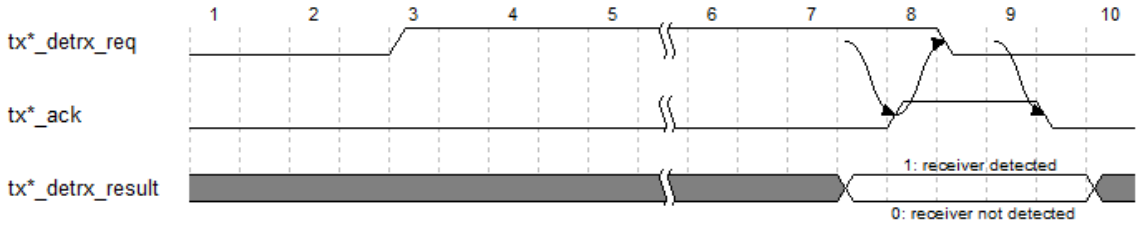


图 3-16 TX Detect RX 时序

## 3.14 TX Rate 控制

### 3.14.1 功能简介

PLL 输出的时钟会被送到 TX Rate 控制器，该控制器可以对 PLL 输出时钟进行分频，作为发送串行时钟。发送串行时钟把 TX Serializer 的串行数据送达到 TXP/M 管脚上。TX 串行时钟的频率为线速率的二分之一。

### 3.14.2 端口和属性列表

TX Rate 控制相关端口说明如表 3-24 所示。

表 3-24 TX Rate 控制相关端口

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| tx*_rate[2:0] | In | Async | 发送数据速率选择。<br>• 3'b000：全线速率（Full Rate）；<br>• 3'b001：二分之一线速率（全线速率/2）；<br>• 3'b010：四分之一线速率（全线速率/4）；<br>• 3'b011：八分之一线速率（全线速率/8）；<br>• 3'b100：不支持；<br>• 3'b101：五分之一线速率（全线速率/5）；<br>• 3'b110：十二分之一线速率（全线速率/12）；<br>• 3'b111：十分之一线速率（全线速率/10）。 |

### 3.14.3 用法

TX Rate 控制器在 SERDES 中所处的位置参考“TX Fabric 时钟”这一章节。TX Rate 更新参考“TX 配置更新”这一章节。

## 3.15 TX 配置更新

### 3.15.1 功能简介

发送侧的一些配置可以支持动态更新。具体包括 MPLL 参数更新，Power State 更新，TX Rate 更新，和其他的一些发送相关参数的更新。

### 3.15.2 端口和属性列表

TX 配置更新相关端口及参数说明分别如表 3-25、表 3-26 所示。

表 3-25 TX 配置更新相关端口

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| tx*_req | In | Async | 发送器配置更新请求。<br>该信号拉高发起配置更新请求。这个信号只能在 tx*_ack 为低的时候拉高，并一直保持为高电平，直到 tx*_ack 拉高。在 tx*_ack 拉高以后必须拉低 tx*_req。<br>下面这些信号会被 tx*_req 抓取： |

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| | | | • tx*_pstate<br>• tx*_lpd<br>• tx*_rate<br>• l*_tx_pmadata_width<br>• tx*_mpllb_sel<br>• tx*_mpll_en<br>• mplla_*<br>这些信号在 tx*_req 拉高之前必须已经稳定并且一直保持稳定，直到 tx*_req 拉低。 |
| tx*_ack | Out | Async | 发送配置更新应答信号。<br>该信号拉高表示配置更新请求已经被接受并处理完成。如果 tx*_req 不拉低，那么这个信号会一直保持高电平。tx*_req 拉低以后，这个信号会随之拉低。 |
| tx*_pstate[1:0] | In | Async | 发送 Power State。<br>设置发送 Power State。PCIe 的设置举例如下。<br>• 2'b00: P0-Transmitter is fully powered up.<br>• 2'b01: P0S-Transmitter common mode is held, TX analog clocks are active, but TX serializer is off.<br>• 2'b10: P1-Transmitter common mode is held, but TX analog clocks and TX serializer are off.<br>• 2'b11: P2-Transmitter is powered down. |
| tx*_lpd | In | Async | 发送 Lane power down<br>将发送器置为 power down 模式，等效于 P1. 如果 tx*_pstate[1:0]设置为 P2，那么 tx*_lpd 会被克服，发送器会置为 P2 状态。 |
| tx*_rate[2:0] | In | Async | 发送数据速率选择。 |
| l*_tx_pmadata_width[1:0] | In | Async | 发送 PMA 数据位宽，见‘TX 串行器’。 |
| tx*_mpllb_sel | In | Async | MPLL 选择信号。<br>• 0：选择 MPLLA；<br>• 1：选择 MPLLB。 |

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| tx*_mpll_en | In | Async | MPLL 使能信号。 |
| mplla_* | In | Async | MPLLA 相关配置信号。 |

表 3-26 TX 配置更新相关属性

| 属性名 | 类型和宽度 | 说明 |
|---|---|---|
| TX*_MISC | 8 | 保留。 |

## 3.15.3 用法

更新过程是通过 tx*_req 和 tx*_ack 握手来实现的。握手过程时序波形如图 3-17 所示。当接收配置设置好新的值以后，拉高 tx*_req，并保持为高。等待 tx*_ack 拉高，此时表示新的配置更新已经完成。然后拉低 tx*_req，等待 tx*_ack 拉低。在 tx*_ack 拉低以后，整个更新过程完成，可以进行下一次更新。

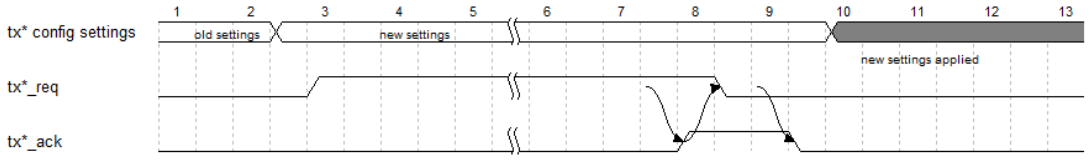


图 3-17 TX 配置更新过程

图中 tx* config settings 包括表 3-25 中所列举的 tx 配置端口。

# 4 RX Channel

## 4.1 RX Channel 总览

### 4.1.1 功能简介

这一节将阐述如何配置和使用 RX Channel 里的每个模块。每个 SERDES Channel 有一个 RX Channel，由 RX PMA 和 RX PCS 组成。图 4-1 中展示了 RX Channel 的每一个模块。高速串行数据从板上的连线进入 RX PMA，再进入 RX PCS，最后进入 FPGA 的 Fabric。

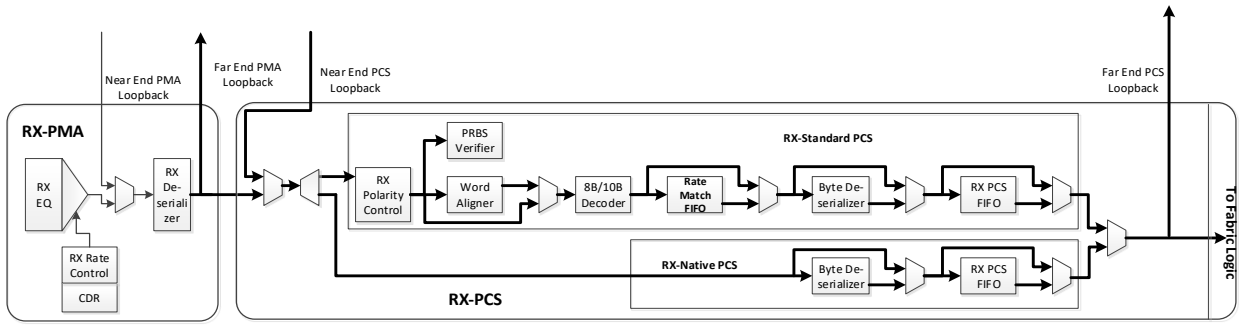


图 4-1 RX Channel 框图

### 4.1.2 RX Channel 的时钟域

RX Channel 的时钟域如图 4-2 所示。

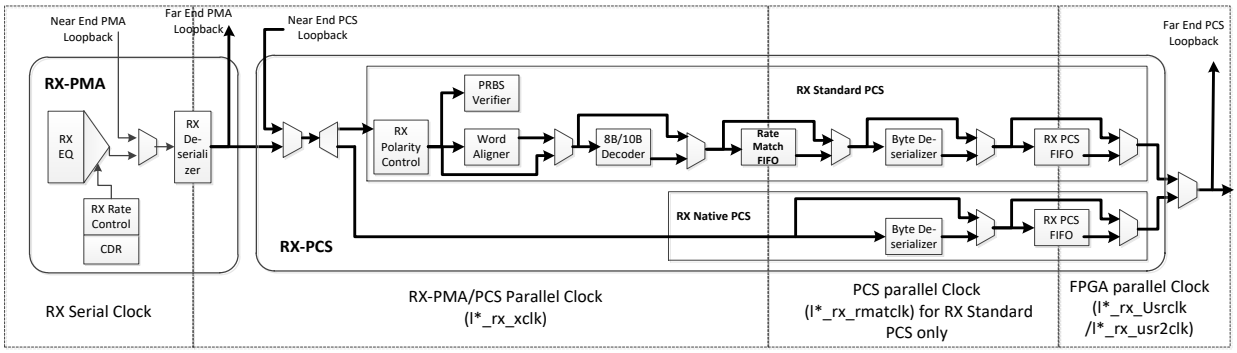


图 4-2 RX Channel 的时钟域

RX 方向主要有 4 个时钟域：

- RX Serial Clock，它是接收端 CDR 产生的高速串行时钟。
- l*_rx_xclk，RX 并行时钟域或 RX XCLK 时钟域。
- l*_rx_rmatclk，Rate Match FIFO 的读时钟，接收数据经过 Rate Match FIFO 之后的 PCS 电路由此时钟来驱动。Native PCS 模式下，无此时钟域。

- l*_rx_usrclk/l*_rx_usr2clk，用户时钟。

## 4.2 接收模拟前端 RX Analog Front End (AFE)和 DFE

### 4.2.1 接收模拟前端（AFE）功能简介

接收模拟前端（AFE）电路接收外部数据并用可编程或自适应的 CTLE 放大高频分量，去补偿通道损耗。

接收模拟前端 AFE 包含 4 个部分。分别是端接（TERM），衰减器（ATT），CTLE 和 VGA。如图 4-3 所示。

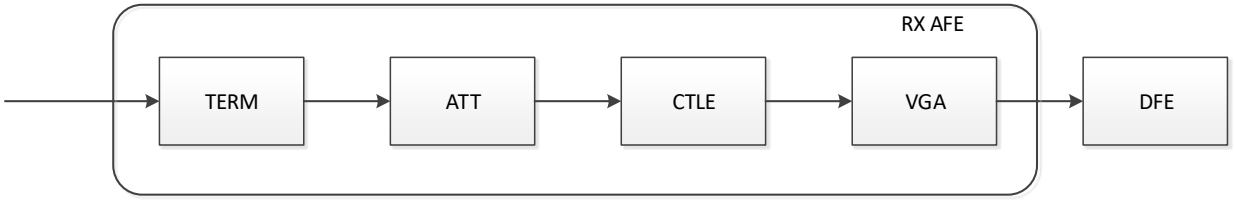


图 4-3 接收模拟前端内部模块图

每个模块分别介绍如下：

**Termination (TERM) 端接**

端接模块是以外部参考电阻为基础进行校准得到的。RX 端接如图 4-4 所示。

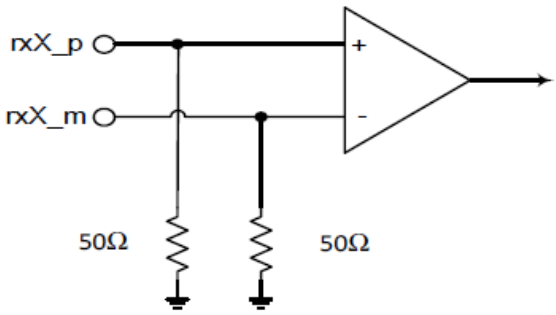


图 4-4 接收端接 RX TERM

**Attenuator (ATT) 衰减器**

衰减器模块可以把输入信号衰减到后续的 CTLE 可接收的电压范围。主要是为了支持 short reach 的通道。衰减是由 RX*_EQ_ATT_LVL 来控制的。

**Continuous time-linear equalizer (CTLE) 线性均衡器**

CTLE 模块可以支持高频放大，放大的范围是 2dB-15dB。CTLE 的 boost level，boost pole frequency 和 boost bandwidth 是可以编程控制的。分别是下面这些端口来控制：rx*_eq_ctle_boost,

rx*_eq_ctle_pole 和 rx*_rate。

**Voltage gain amplifiers (VGA1 and VGA2) 电压增益放大器**

VGA 的主要目的是为了保持信号幅度 Swing 为常量。模拟增益是 rx*_eq_vga1_gain 和 rx*_eq_vga2_gain 来控制的，带宽 BW 控制是 rx*_rate 来控制的。为得到最优的链路性能，接收均衡的参数是自适应到最优值的。

### 4.2.2 Decision Feedback Equalizer （DFE）功能简介

VGA 之后是一个 5-TAP 的 DFE，它是不间断自适应的。DFE 的数据被 sampler 采样，然后进入 RX Deseirializer 转成并行数据，和恢复时钟一起送到 PCS。 DFE 是可以被关闭的，由 rx*_dfe_bypass 端口来控制。

### 4.2.3 端口和属性列表

接收模拟前端和 DFE 相关端口及参数说明分别如表 4-1、表 4-2 所示。

表 4-1 接收模拟前端和 DFE 相关端口

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| rx*_eq_ctle_boost[4:0] | In | Async | 设置 CTLE 的 boost 值。 |
| rx*_adapt_ack | Out | Async | 接收自适应请求的 ACK 信号，高有效。<br>这个信号拉高表示接收自适应序列已经完成，此时 rx*_adapt_fom, rx*_txmain_dir, rx*_txpre_dir, and rx*_txpost_dir 端口上的值有效。<br>注释：<br>rx*_adapt_req 信号如果一直为高，那么 rx*_adapt_ack 也会保持为高。直到 rx*_adapt_req 拉低，那么 rx*_adapt_ack 才会拉低。 |
| rx*_adapt_fom[7:0] | Out | Async | 接收 FOM （Receiver figure of merit）<br>表征接收眼图质量。值越大表示链路均衡质量越好。在 rx*_adapt_ack 拉高时有效。 |
| rx*_adapt_in_prog | In | Async | 保留。 |
| rx*_adapt_mode[1:0] | In | Async | 选择接收自适应模式，保留。 |
| rx*_adapt_req | In | Async | 接收自适应请求。<br>该信号发起接收自适应请求。该信号必须保持为高，直到 rx*_adapt_ack 拉高。此后，rx*_adapt_req 要拉低。 |

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| | | | 注释：<br>rx*_adapt_req 在 rx*_adapt_ack 拉高之前就拉低，或者在 rx*_adapt_ack 拉高之前发生了 rx*_reset 复位，那么接收均衡设置可能不是最优的。 |
| rx*_adapt_sel | In | Async | 保留。 |
| rx*_delta_iq[3:0] | In | Async | IQ Offset 值，保留。 |
| rx*_dfe_bypass | In | Async | RX DFE Bypass，高有效。<br>该信号拉高 DFE 被旁路。可以在不需要 DFE 的情况下使用。当 DFE 被旁路时，DFE 处于断电状态。该信号为低电平时，DFE 被使能。 |
| rx*_eq_dfe_tap[7:0] | In | Async | DFE Tap1。有符号，编码方式为 2 的补码。保留。 |
| rx*_eq_vga1_gain[2:0] | In | Async | 控制 VGA1 的增益， binary encoded。 |
| rx*_eq_vga2_gain[2:0] | In | Async | 控制 VGA2 的增益， binary encoded。 |
| rx*_txmain_dir[1:0] | Out | Async | 给远端发送 TX main cursor 提供修改方向。该信号在 rx*_adapt_ack 拉高时有效。INC/DEC 具体定义见下文。<br>• 2'b00: HOLD ；<br>• 2'b01: INC ；<br>• 2'b10: DEC ；<br>• 2'b11: Reserved。 |
| rx*_txpost_dir[1:0] | Out | Async | 给远端发送 TX post cursor 提供修改方向。该信号在 rx*_adapt_ack 拉高时有效。INC/DEC 具体定义见下文。<br>• 2'b00: HOLD ；<br>• 2'b01: INC ；<br>• 2'b10: DEC ；<br>• 2'b11: Reserved。 |
| rx*_txpre_dir[1:0] | Out | Async | 给远端发送 TX pre cursor 提供修改方向。该信号在 rx*_adapt_ack 拉高时有效。INC/DEC 具体定义见下文。<br>• 2'b00: HOLD ；<br>• 2'b01: INC ；<br>• 2'b10: DEC ； |

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| | | | • 2'b11: Reserved。 |

表 4-2 接收模拟前端和 DFE 相关属性

| 属性名 | 类型和宽度 | 说明 |
|---|---|---|
| RX*_ADAPT_CONT | 1 | 该属性设置为 1，表示接收自适应不间断发生。如果设置为 0，接收自适应会在 rx*_adapt_ack 拉高后停止。 |
| RX*_EQ_ATT_LVL | 3 | 设置衰减器的幅度。从-2dB（000）到-6dB（111），二进制编码。 |
| RX*_EQ_CTLE_POLE | 2 | 设置 CTLE boost pole，二进制编码。 |
| RX*_MARGIN_IQ | 7 | 保留。 |
| RX*_OFFCAN_CONT | 1 | Offset Cancellation 不间断操作。保留。 |

## 4.2.4 用法

接收器支持接收均衡，接收参数自适应和增减对端器件发送侧均衡参数调整命令。对 DFE 和链路自适应必须的协议，接收均衡由 rx*_adapt_req 发起。接收自适应完成后，rx*_adapt_ack 会拉高。

### 4.2.4.1. 一次性自适应均衡用法

RX*_ADAPT_CONT 设置为 0，RX*_OFFCAN_CONT 设置为 0，rx*_adapt_req 拉高触发自适应过程，自适应过程结束后，rx*_adapt_ack 拉高。此后 rx*_adapt_req 拉低，随后 rx*_adapt_ack 信号拉低，表示自适应过程结束。时序图如 4-5 所示。

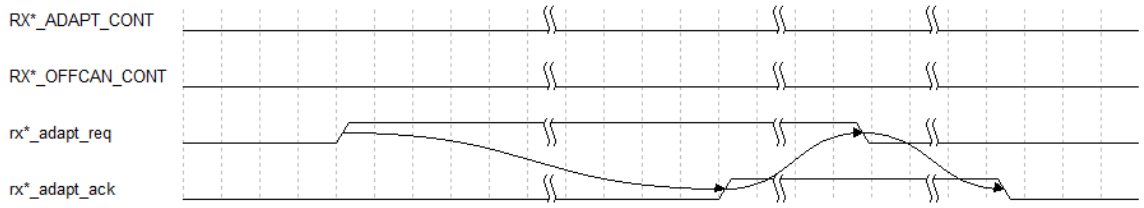


图 4-5 触发一次性自适应均衡的方法

### 4.2.4.2. 连续自适应均衡用法

连续自适应模式下，VGA 和 DFE 可以做不间断的自适应，Offset Cancellation 电路也可以做连续不间断自适应。

RX*_ADAPT_CONT 置为 1，设置 VGA 和 DFE（如果已经使能）不间断自适应模式；RX*_OFFCAN_CONT 置为 1，设置 Offset Cancellation 不间断自适应。RX*_ADAPT_CONT 和 RX*_OFFCAN_CONT 可以同时为 1，设置 VGA，DFE 和 Offset Cancellation 不间断自适应。设置好以上参数以后，再通过 rx*_adapt_req 来触发连续不间断的自适应。时序和触发一次性自适应是类似的，如图 4-6 所示。

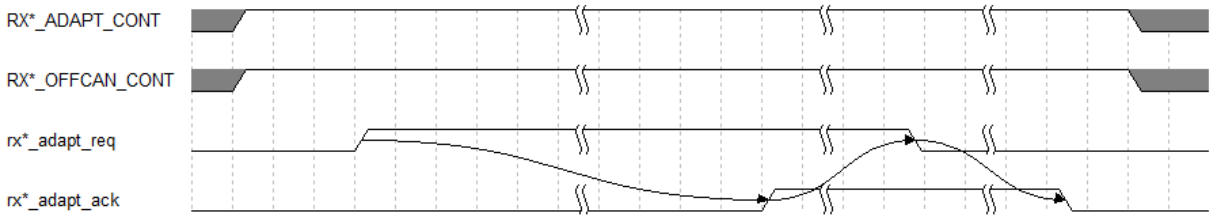


图 4-6 触发连续不间断自适应均衡的方法

触发自适应之后，可以用下面两种机制其中之一去更新对端器件的发送参数。

**Figure of Merit (FOM)**

8 位输出端口 rx*_adapt_fom[7:0]表示接收眼图的质量。值越高意味着眼图质量越好。rx*_adapt_fom[7:0]在 rx*_adapt_ack 拉高时有效。该机制用于 PCIe。

**Direct Change（更改方向）**

rx*_txpre_dir[1:0]，rx*_txmain_dir[1:0]，和 rx*_txpost_dir[1:0]输出端口发出 increment（增）/decrement（减）/hold（保持）命令给对端器件，命令器发送参数根据所提示的方向进行更改和调整。rx*_txpre_dir[1:0]，rx*_txmain_dir[1:0]，和 rx*_txpost_dir[1:0]输出端口在 rx*_adapt_ack 拉高时有效。该方式用于 10GBASE-KR。

接收 AFE 设置，包括 ATT,CTLE,VGA,DFE 等模块的具体配置。需通过 serdes-AMI 仿真进行优化，并进行上板扫参实测验证后确认，扫参方法请参考《TN903_安路科技 PH1A 系列 FPGA SERDES 模拟参数说明》。

## 4.3 RX LOS (Loss Of Signal) DETECTION

### 4.3.1 功能简介

rx*_los 输出信号用于指示 RXP/RXM 输入管脚上是否探测到电器空闲（Electrical Idle）。RX LOS 探测器在 rx*_disable 和 rx*_reset 拉低的时候被打开。RX*_LOS_LFPS_EN 为高时，LOS 探测器被配置成 LFPS Detector，用于 USB3 或类似应用中；RX*_LOS_LFPS_EN 为低时，LOS 探测器工作在普通模式下，通常可用于 PCI Express 或其他需要探测电气空闲的场景中。

### 4.3.2 端口和属性列表

RX LOS 探测相关端口及属性说明分别如表 4-3、表 4-4 所示。

表 4-3 RX LOS 探测相关端口

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| rx*_los | Out | Async | 接收 LOS 指示信号，高有效。<br>拉高表示接收器探测到电气空闲（Electrical Idle），判别电气空闲的阈值由 RX*_LOS_THRESHOLD 设定。这个信号只在 RXP/RXM 为低速信号（2.5Gbps 及以下）下才有效。<br>• rx*_los=1 ：RXP/RXM 探测到电气空闲（Electrical Idle）；<br>• rx*_los=0 ：RXP/RXM 探测到正常数据。<br>在线速率高于 2.5Gbps 时 rx*_los 信号无效，rx*_los 的值不确定。 |

表 4-4 RX LOS 探测电路相关属性

| 属性名 | 类型和宽度 | 说明 |
| --- | --- | --- |
| RX*_LOS_LFPS_EN | 1 | LFPS LOS 探测模式使能。默认设为 0，即普通探测模式。 |
| RX*_LOS_THRESHOLD | 3 | LOS 或电气空闲 Electrical Idle 探测阈值电压。保留，推荐使用 IP Generator 产生的值。 |

### 4.3.3 用法

在 rx*_disable 为高时，rx*_los 信号的值不确定。当 rx*_disable 拉低以后，LOS 探测器开始启动，启动时间需要 10us，在此过程中 rx*_los 输出的值不确定，是无效的。当 rx_term_en 为低电平时，RX termination 被关闭，此时 rx*_los 输出的值也是不确定。

RX*_LOS_THRESHOLD 参数共有 3 bits，合法取值范围是 1-7。0 是保留值，用户不使用。该数值越大，表示判别电气空闲的阈值电压越大。该阈值电压为差分峰峰值。

rx*_los 和 CDR 锁定信号（rx*_valid）没有直接关联。

## 4.4 RX Rate 控制

### 4.4.1 功能简介

接收高速时钟是由 CDR 环路产生的，CDR 环路控制并按照 RX data 来对齐高速时钟的频率和相位。

CDR 所控制的 VCO 提供 4GHz 到 8GHz 频率范围的时钟。这个时钟被 RX Data Rate 控制器（rx*_rate[1:0]）分频后的时钟可以接收 1.2Gbps 到 16.0Gbps 的数据。比如，设置 rx*_rate[1:0]=2’b10，接收时钟频率为 5GHz/4=1.25GHz，此时线速率是 2.5Gbps。

### 4.4.2 端口和属性

RX Rate 控制相关端口说明如表 4-5 所示。

表 4-5 RX Rate 控制相关端口

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| rx*_rate[1:0] | In | Async | 接收速率选择，RX VCO 频率分频参数。<br>• 2'b00：全线速率（Full Rate）；<br>• 2'b01：二分之一线速率（Half Rate）；<br>• 2'b10：四分之一线速率（Quarter Rate）；<br>• 2'b11：八分之一线速率（Octal Rate）。 |

### 4.4.3 用法

rx*_rate 更新方法参见“RX 配置更新”这一章节。

## 4.5 RX Clock and Data Recovery (CDR)

### 4.5.1 功能简介

CDR 电路按照二阶环路来实现。环路包含一个 Integral 路径和一个 Proportional 路径。

如图 4-7 所示。Integral 路径和 Proportional 路径共同作用在 RX VCO，使得 VCO 产生和输入数据频率和相位都对齐的时钟。

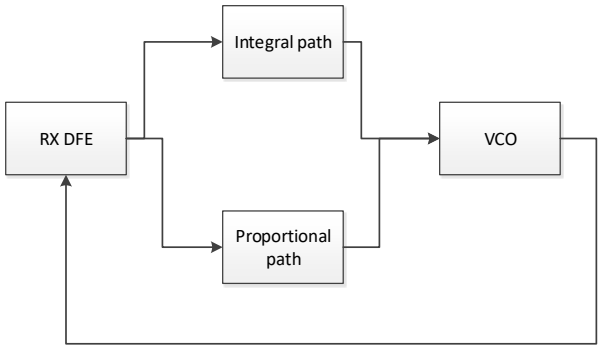


图 4-7 RX CDR 结构

### 4.5.2 端口和属性列表

CDR 相关端口及属性说明分别如表 4-6、表 4-7 所示。

表 4-6 CDR 相关端口

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| rx*_cdr_ppm_max[4:0] | In | Async | CDR 最大允许的 PPM 频率偏置。<br>设置在相对于期望频率的最大 PPM 漂移，使用 IP 模 |

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| | | | 式设置。 |
| rx*_cdr_ssc_en | In | Async | 接收 CDR SSC 模式使能，高有效。<br>如果接收数据由 SSC，这个信号应该被拉高；反之，则拉低。 |
| rx*_cdr_vco_freqband[1:0] | In | Async | 保留。使用 IP 默认设置。 |
| rx*_cdr_vco_step_ctrl | In | Async | 保留。使用 IP 默认设置。 |
| rx*_valid | Out | Async | 接收 rx_data 有效指示信号，高有效。<br>这个信号拉高表示 CDR 已经锁定。当 CDR 漂移到 rx*_cdr_ppm_max[4:0]设定的 PPM 偏置范围之外的时候，这个信号会拉低。 |
| rx*_vco_ld_val[12:0] | In | Async | 保留，使用 IP 默认设置。 |
| rx*_ref_ld_val[6:0] | In | Async | 保留，使用 IP 默认设置。 |
| rx*_ppm_drift_vld | In | Async | rx*_ppm_drift 的有效指示信号。当 rx*_ppm_drift_vld 为高电平时，rx*_ppm_drift 的值有效。 |
| rx*_ppm_drift[5:0] | In | Async | CDR VCO 相对于基准时钟的频偏。为 2 的补码。其值在-32 (6'b100000)到+31(6'b011111)之间。当这个值的绝对值大于 rx*_cdr_ppm_max 的值时，RX CDR 判定为失锁，rx*_valid 拉低。 |

表 4-7 CDR 相关属性

| 属性名 | 类型和宽度 | 说明 |
|---|---|---|
| RX*_CDR_VCO_TEMP_COMP_EN | 1 | RX CDR 温度补偿使能，高有效。推荐使用 IP Generator 产生的值。 |

## 4.5.3 用法

以下为接收线速率（I*_rx_linerate）计算方法。

首先计算 CDR 的 VCO 频率，公式如下。下面公式中 ref_clk_freq 为参考时钟频率。Ref_clk_div 是 REF_CLK_DIV2_EN 所使能的分频器。如果 REF_CLK_DIV2_EN=0，该分频器不分频；如果 REF_CLK_DIV2_EN=1，该分频器为 2 分频。

$$RX*\_CDRVCOFREQ = \frac{ref\_clk\_freq \times rx*\_vco\_ld\_val}{ref\_clk\_div \times rx*\_ref\_ld\_val}$$

RX 接收线速率计算公式如下。其中 rx*_rate_div 是 rx*_rate 所控制的分频器。

$$l*\_rx\_linerate = \frac{RX*\_CDROUTFREQ \times 2}{rx*\_rate\_div}$$

#### 4.5.3.1. CDR 非跟踪模式（non-tracking mode）

CDR 可以工作在跟踪模式（tracking mode,rx*_data_en=1）和非跟踪模式（non-tracking mode,rx*_data_en=0）。

通常情况下，CDR 工作在跟踪模式，它可以实时跟踪输入数据的频率和相位，使得恢复时钟和对端器件（remote parnter）发送时钟同频或者相位锁定。但是，在线速率较低时，例如快速以太网（125Mbps），CDR 可能无法准确跟踪对端器件的数据，这时可以将 SERDES 配置为标称线速率（1.2Gbps 以上），同时将 CDR 设置为非跟踪模式，此时 SERDES 通过本地时钟对接收到的慢速数据进行数倍、数十倍或更高倍数的过采样（Over Sampling），用户逻辑在过采样数据中可以通过特定算法选取采样数据，从而恢复出对端发来的低频数据。

在非跟踪模式下，需设置 rx*_data_en=0，此时 CDR 不再跟踪输入数据，CDR 锁定信号 rx*_valid 会一直处于低电平。在非跟踪模式下，恢复时钟频率和本地发送时钟频率会有微小的偏差，并不是完全锁定在本地参考时钟的频率上。这个状态通常不影响过采样数据恢复，但是在过采样模式下对时钟精度有较高要求的应用中可能会有限制。

在过采样的应用中，RX Equalizer 不开自适应（Self Adaptation），用户需将 DFE 关闭（rx*_dfe_bypass=1）。

CDR 恢复时钟与本地参考时钟无相关性。

## 4.6 RX 解串器（RX Deserializer）

### 4.6.1 功能简介

接收解串器将串行接收数据转化为并行数据发送给 PCS。

接收解串器使用 CDR 产生的高速串行恢复时钟对数据进行采样，然后用低速并行恢复时钟对数据进行解串。解串器把并行数据发送给接收通道的 PCS。解串器支持 16 位或 20 位并行数据接口，在线速率较低时，还支持可选的 8 位或 10 位并行数据接口。线速率和解串器位宽对应关系如表 4-8 所示。

表 4-8 线速率和解串器位宽对应关系

| Line rate 范围（Gbps） | 发送/接收支持的并行数据位宽 |
|---|---|
| 1.2 - 6.4 | 8-bit, 10-bit, 16-bit, 20-bit |
| 6.4 - 12.5 | 16-bit, 20-bit |

### 4.6.2 端口和属性列表

RX 解串器相关端口如表 4-9 所示。

表 4-9 RX 解串器相关端口

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| l*_rx_pmadata_width[1:0] | In | Async | PMA 并行数据位宽设置，保留。使用 IP Generator 产生的值。用户不能修改。信号值对应的位宽如下：<br>• 2'b00: 8-bit;<br>• 2'b01: 10-bit;<br>• 2'b10: 16-bit;<br>• 2'b11: 20-bit。 |

## 4.7 RX 配置更新

### 4.7.1 功能简介

接收侧的一些配置可以支持动态更新。

### 4.7.2 端口和属性列表

RX 配置更新相关端口及参数分别如表 4-10、表 4-11 所示。

表 4-10 RX 配置更新相关端口

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| rx*_req | In | Async | 接收器配置更新请求。<br>该信号拉高发起配置更新请求。这个信号只能在 rx*_ack 为低的时候拉高，并一直保持为高电平，直到 rx*_ack 拉高。在 rx*_ack 拉高以后必须拉低 rx*_req。<br>下面这些端口或属性会被 rx*_req 抓取：<br>• rx*_pstate<br>• rx*_lpd<br>• rx*_rate<br>• l*_rx_pmadata_width<br>• rx*_dfe_bypass<br>• RX*_EQ_ATT_LVL<br>• rx*_eq_vga1_gain<br>• rx*_eq_vga2_gain<br>• RX*_EQ_CTLE_POLE<br>• rx*_eq_ctle_boost |

| 信号名 | 方向 | 时钟域 | 说明 |
| --- | --- | --- | --- |
| | | | • rx*_eq_dfe_tap<br>• rx*_ref_ld_val<br>• rx*_vco_ld_val<br>• rx*_adapt_sel<br>• rx*_cdr_vco_freqband<br>• rx*_cdr_vco_step_ctrl<br>• RX*_CDR_VCO_TEMP_COMP_EN<br>• rx*_delta_iq<br>• RX*_MARGIN_IQ<br>• rx*_misc<br>这些信号必须在 rx*_req 拉高之前稳定下来，并且在 rx*_req 拉低之前一直保持稳定。<br>rx*_ack 拉高表示这一次的请求已经完成。 |
| rx*_ack | Out | Async | 接收配置更新应答信号。<br>该信号拉高表示配置更新请求已经被接受并处理完成。如果 rx*_req 不拉低，那么这个信号会一直保持高电平。rx*_req 拉低以后，这个信号会随之拉低。 |
| rx*_pstate[1:0] | In | Async | 接收 power state<br>设置接收器的 power state。设置不同的值对应的 power state 如下。<br>• 2'b00: P0 - 接收器完全上电，输出时钟活跃。Receiver is fully powered up and output receive clocks are active<br>• 2'b01: P0S - RX VCO 工作在连续 calibration 状态，无输出时钟。RX voltage-controlled oscillator (VCO) is in continuous calibration mode, output receive clocks are not available.<br>• 2'b10: P1- RX AFE 和电压 RX AFE 和电压 regulators 处于上电状态，但 RX VCO 处于复位状态。analog front-end (AFE) and voltage regulators are powered up, but |

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| | | | RX VCO is in reset<br>• 2'b11: P2- RX Los Detector 处于上电状态，接收器其他部分电源关断。RX LOS detector is powered up and the rest of RX is powered down. |
| rx*_lpd | In | Async | 接收 lane power down。<br>将接收端置为 P1 power state。<br>注释：<br>如果 rx_state[1:0]输入设置为 P2，会克服 rx*_lpd 的设置，接收器进入 P2 状态。 |
| rx*_rate[1:0] | In | Async | 见 RX Rate Control。 |
| l*_rx_pmadata_width[1:0] | In | Async | 见 RX 解串器章节。 |
| rx*_dfe_bypass | In | Async | 见接收模拟前端 RX Analog Front End 和 DFE 章节。 |
| rx*_eq_vga1_gain[2:0] | In | Async | 见接收模拟前端 RX Analog Front End 和 DFE 章节。 |
| rx*_eq_vga2_gain[2:0] | In | Async | 见接收模拟前端 RX Analog Front End 和 DFE 章节。 |
| rx*_eq_ctle_boost[4:0] | In | Async | 见接收模拟前端 RX Analog Front End 和 DFE 章节。 |
| rx*_eq_dfe_tap[7:0] | In | Async | 见接收模拟前端 RX Analog Front End 和 DFE 章节。 |
| rx*_ref_ld_val[6:0] | In | Async | 保留。 |
| rx*_vco_ld_val[12:0] | In | Async | 保留。 |
| rx*_adapt_sel | In | Async | 见接收模拟前端 RX Analog Front End 和 DFE 章节。 |
| rx*_cdr_vco_freqband[1:0] | In | Async | 保留。 |
| rx*_cdr_vco_step_ctrl | In | Async | 保留。 |
| rx*_delta_iq[3:0] | In | Async | 保留。 |
| rx*_misc[7:0] | In | Async | 保留。 |

表 4-11 RX 配置更新相关属性

| 属性名 | 宽度 | 说明 |
|---|---|---|
| RX*_CDR_VCO_TEMP_COMP_EN | 1 | 见 RX Clock and Data Recovery (CDR)章节。 |

| | | |
|---|---|---|
| RX*_MARGIN_IQ | 7 | 见接收模拟前端 RX Analog Front End 和 DFE 章节。 |
| RX*_EQ_ATT_LVL | 3 | 见接收模拟前端 RX Analog Front End 和 DFE 章节。 |
| RX*_EQ_CTLE_POLE | 2 | 见接收模拟前端 RX Analog Front End 和 DFE 章节。 |

### 4.7.3 用法

更新过程是通过 rx*_req 和 rx*_ack 握手来实现的。握手过程时序波形如图 4-8 所示。当接收配置设置好新的值以后，拉高 rx*_req，并保持为高。等待 rx*_ack 拉高，此时表示新的配置更新已经完成。然后拉低 rx*_req，等待 rx*_ack 拉低。在 rx*_ack 拉低以后，表示整个更新过程已经完成，可以进行下一次更新。

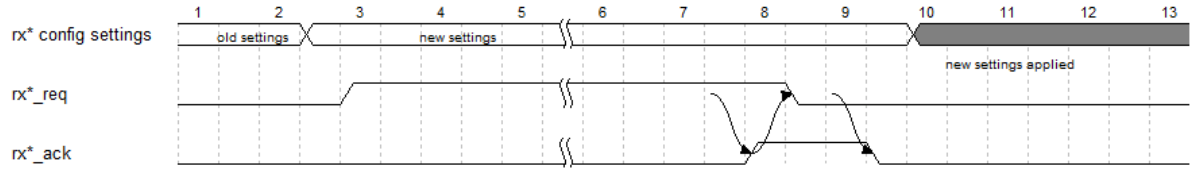


图 4-8 RX 配置更新过程

## 4.8 RX Polarity Control

### 4.8.1 功能简介

当硬件设计者在设计 PCB 时有时不得不把 SERDES 接收差分对的 P 和 M 接反，PH1A 系列 FPGA SERDES 允许用户在 RX PMA 里做极性取反，也可以在 RX Standard PCS 里的 RX Polarity Control 做极性取反。

### 4.8.2 端口和属性

RX Polarity Control 相关端口及参数分别如表 4-12、表 4-13 所示。

表 4-12 RX Polarity Control 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| rx*_invert | In | cr_para_clk | RX PMA 里控制 RX 极性反转的控制信号，高有效。 |

表 4-13 RX Polarity Control 属性列表

| 属性 | 类型 | 描述 |
|---|---|---|
| L*_RX_POLARITYINV_EN | 1 bit | RX PCS 里控制 RX 极性反转的属性，高有效，保留，在 PRBS Verifier 模块使能时禁用，保持赋值为 0。推荐使用 PMA 控制极性。 |

### 4.8.3 用法

用户在 RX 方向，可以设置极性反转，用 rx*_invert 拉高即可反转极性。

## 4.9 PRBS Verifier

### 4.9.1 功能简介

本模块接收来自 RX PMA 的并行数据，其有效数据位宽为 RX PMA 和 RX PCS 接口有效位宽（8/10/16/20）。仅在 RX Standard PCS 模式下，才能支持 PRBS Verifier。

当 l*_rx_prbs_en 为高，则根据 l*_rx_psbs_sel 信号选择 PRBS 检测电路进行检测。PRBS Verifier 框图如图 4-9 所示。

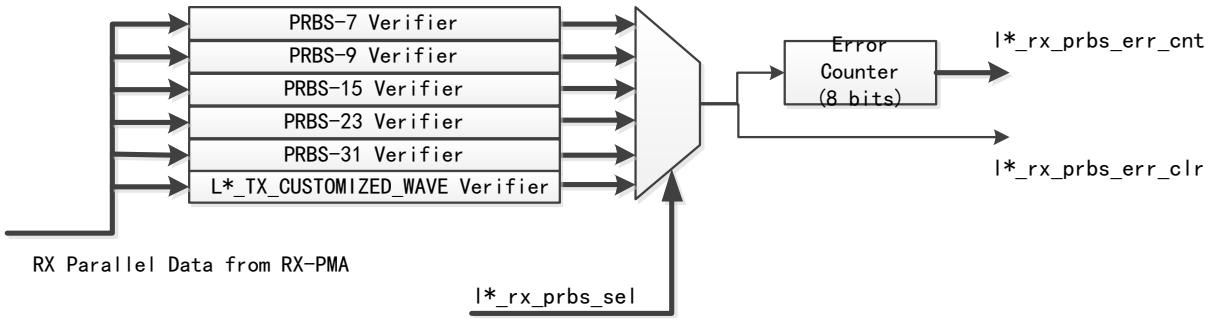


图 4-9 PRBS Verifier 框图

本模块的特性如下：

- 支持对输入的 PRBS 数据进行检测，输出错误指示。
- 支持在 RX CDR 时钟锁定后开始进行 PRBS 检测。

### 4.9.2 端口和属性

PRBS Verifier 端口及属性分别如表 4-14、表 4-15 所示。

表 4-14 PRBS Verifier 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| l*_rx_prbs_en | In | cr_para_clk | PRBS Verifier 模块的使能信号，高有效。<br>当该端口为高时，该模块进行 PRBS 检测。 |
| l*_rx_prbs_sel[2:0] | In | cr_para_clk | PRBS Verifier 检测数据的模式选择。<br>• 3’b000：PRBS-7；<br>• 3’b001：PRBS-9； |

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| | | | • 3’b010：PRBS-15；<br>• 3’b011：PRBS-23；<br>• 3’b100：PRBS-31；<br>• 3’b101*： 用户自定义的 20bits 波形；<br>• 3’b110， 111：保留。 |
| l*_rx_prbs_err_clr | In | cr_para_clk | PRBS Verifier 的清零控制信号，高有效。<br>此信号为高时，l*_rx_prbs_err_cnt 清零。 |
| l*_rx_prbs_err_cnt[7:0] | Out | cr_para_clk | PRBS Verifier 统计的 PRBS 错误计数器。当有一个字符出错时，此错误计数器就累加 1。<br>当 l*_rx_prbs_err_clr 为高时，本计数器被清零。若 l*_rx_prbs_err_cnt[7:0]计数器达到最大值 0xFF，之后即使再有 PRBS Error 被检测到，该计数器的值也不会再改变，而是保持 0xFF，直到被 l*_rx_prbs_err_clr 清零。 |

*当 l*_rx_prbs_sel 为 3’b101 时，用户自定义的波形来自 TX Standard PCS 中 PRBS Generator 的 L*_TX_CUSTOMIZED_WAVE 属性参数。

表 4-15 PRBS Verifier 属性列表

| 属性 | 类型 | 描述 |
| --- | --- | --- |
| L*_TX_CUSTOMIZED_WAVE* | 20 bit | 本属性是来自 TX Standard PCS 中 PRBS Generator 模块，是用来让用户自定义 PRBS Generator 产生的波形。<br>当 l*_tx_prbs_sel 为 3’b101 时，此属性生效。 |

*本模块 l*_rx_prbs_sel 设置的 PRBS 模式，必须跟其连接的发送端的模式一致。如果是 SERDES 自环，l*_rx_prbs_sel 必须跟 SERDES Channel 的 l*_tx_prbs_sel 一致。

### 4.9.3 用法

如有必要，需控制 rx*_invert 端口来翻转 RX PRBS Verifier 的极性。进入或退出 RX PRBS Verifier，即 l*_rx_prbs_en 发生高低变化之后，需对 l*_rx_pcs_rst_n 进行复位，或者对 rx*_reset 进行复位并带动 l*_rx_pcs_rst_n 的复位。

## 4.10 Word Aligner

### 4.10.1 功能简介

接收到的串行数据，在能当做并行数据使用之前，必须对齐到字符编码的边界。为了实现接收侧的边界对齐，对端的发送侧发送可识别的序列，这个序列通常被叫做 COMMA。

Word Aligner 模块对接收到的数据进行边界对齐。从 RX PMA 收到的并行数据边界是不对齐的，要进行解码之前，必须先进行边界对齐。Word Aligner 会在收到的数据中检测 COMMA。当它检测到 COMMA 时，会把并行数据的边界移到找到 COMMA 的位置，这样，对端发送的并行字符可以被接收器恢复出来。

本模块的特性如下：

- 支持对输入数据根据 l*_rx_wordalign_en 信号进行边界对齐或者 bypass 选择。
- 支持对输入数据进行 bit 取反。
- 支持对输入数据进行 byte 取反。
- 支持 COMMA Plus、COMMA Minus 根据用户需求配置。
- 支持对输入 COMMA 根据 l*_rx_aligncomma_mask 进行部分 bit 位匹配。
- 支持对 COMMA Plus、COMMA Minus 选择其中一个、任意一个或者两个一起进行匹配。
- 支持用户根据需求配置重新边界对齐。
- 支持用户手动进行移位对齐操作。
- 支持用户根据需求对检测到的 K 字符进行指示输出。
- 支持边界位置输出指示。

图 4-10 展示了以 10bitCOMMA 字符的对齐。SERDES RX Channel 接收到的未对齐的数据位于图的右边。串行的 COMMA 字符被虚框框住，位于图的中间。已对齐好的并行数据位于图的左边。

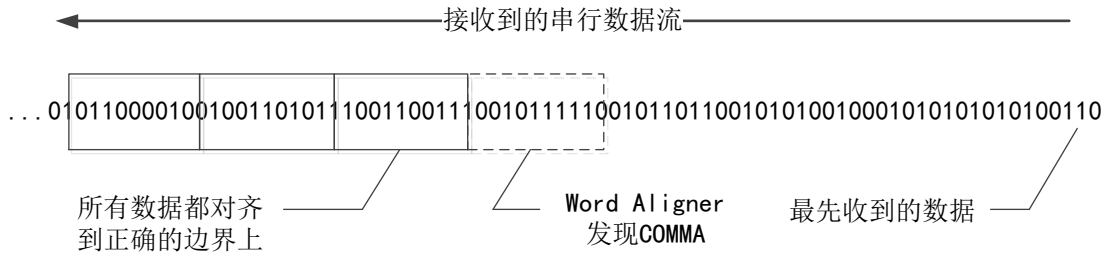


图 4-10 串行数据对齐到 COMMA 的过程图

图 4-11 展示了从并行数据的视角看对齐过程，左边是发送侧的并行数据，右边是接收侧 Word Aligner 输出的并行数据。

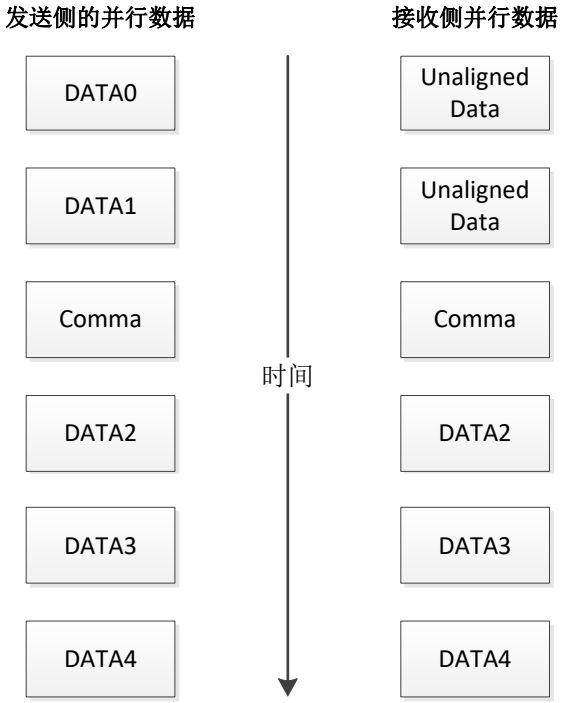


图 4-11 从并行数据的视角看对齐过程

#### 4.10.1.1. 使能 Word Aligner 模块

把端口 l*_rx_wordalign_en 拉高，即可使能 Word Aligner 模块；把此信号拉低，就把此模块设置为旁路。

#### 4.10.1.2. 配置 COMMA 序列

端口 l*_rx_alignmcomma_value/ l*_rx_alignpcomma_value，用作配置 COMMA Minus 和 COMMA Plus 的序列值。端口 l*_rx_aligncomma_mask，用来使能 COMMA 序列值的哪些 bit 进行比对。其逻辑关系图，如图 4-12 所示。

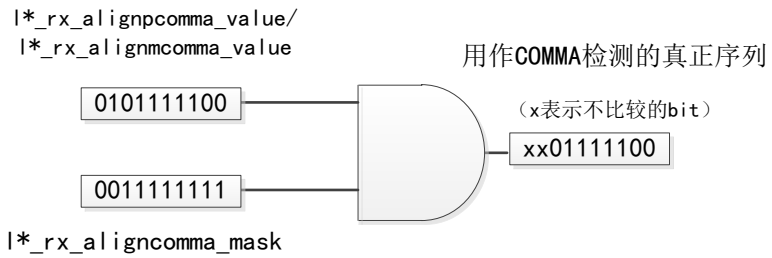


图 4-12 COMMA 序列 bit 使能图

端口 l*_rx_aligncomma_double 为高时，二者均需匹配，且一个 COMMA Plus 后面紧跟一个 COMMA Minus。图 4-13 就是 l*_rx_aligncomma_double 为高时，用作 COMMA 检测的序列值。

| l*_rx_alignmcomma_value | l*_rx_alignpcomma_value |
|---|---|

图 4-13 扩展的 COMMA 序列定义

端口 l*_rx_aligncomma_double 为高，且端口 l*_rx_aligncomma_mask 不是全 1 时，用做 COMMA 检测的真正序列如 4-14 图所示。

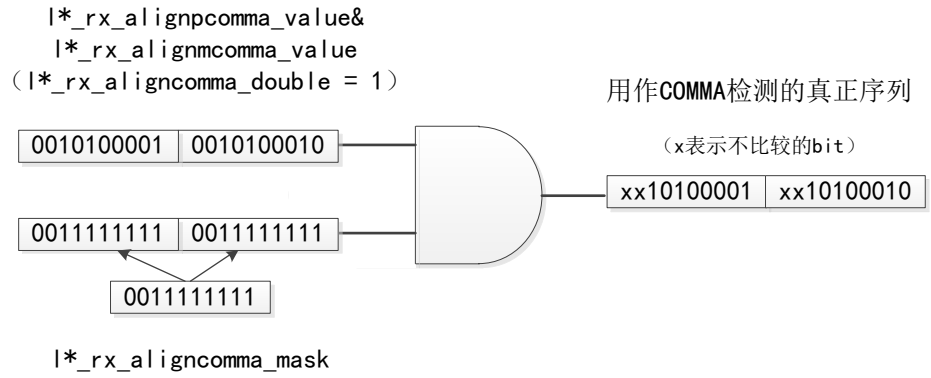


图 4-14 扩展的 COMMA 序列 bit 使能图

#### 4.10.1.3. 激活 COMMA 对齐

当端口 l*_rx_mcommaalign_en 为高时，Word Aligner 就会把并行数据对齐在检测到 COMMA Minus 时的边界。当端口 l*_rx_pcommaalign_en 为高时，Word Aligner 就会把并行数据对齐在检测到 COMMA Plus 时的边界。当 l*_rx_mcommaalign_en 和 l*_rx_pcommaalign_en 都为高时，Word Aligner 就会把并行数据对齐在检测到 Minus COMMA 或 Plus COMMA 时的边界。

当 l*_rx_aligncomma_double 为高时，l*_rx_mcommaalign_en 和 l*_rx_pcommaalign_en 必须都为高，或都为低。

#### 4.10.1.4. 对齐状态信号

当 COMMA Minus 或 COMMA Plus 对齐被激活时，检测到任何跟 COMMA 序列匹配的数据流都会导致一次并行数据边界对齐。成功对齐之后，Word Aligner 端口 l*_rx_wordaligned 会输出并保持高电平。这时，用户可以把 l*_rx_mcommaalign_en 和 l*_rx_pcommaalign_en 拉低，来关掉自对齐，字边界就保持当前的位置。

当 COMMA 对齐没被激活，且端口 l*_rx_wordaligned 已输出高时，Word Aligner 仍然会收到 COMMA。如果这些 COMMA 是跟当前边界位置对齐的，就不会有变化；如果有 COMMA 跟当前边界位置不对齐，端口 l*_rx_wordaligned 就会输出低电平，直到有 COMMA 检测到并跟当前位置边界位置对齐，端口 l*_rx_wordaligned 才会再次输出并保持高电平。

当 COMMA 对齐仍是被激活时，Word Aligner 收到 COMMA，Word Aligner 模块会自动对齐到新的 COMMA 序列被检测到的位置作为新的边界位置，并且会驱动 rx_wordrealign 输出一个高脉冲。

在一些有带外信令的系统中，比如 PCIe，在 Word Aligner 模块已锁定并行数据边界，且已输出

l*_rx_wordaligned 为高电平，即使边界位置没有改变，端口 l*_rx_wordaligned 偶尔地也会被拉低。因此，这种应用中，在 l*_rx_wordaligned 第一次被拉高后，不应该被用作对齐边界的改变的有效指示信号。

#### 4.10.1.5. 对齐边界

在本模块指示已经对齐时，端口 l*_rx_align_sel 指示了对齐的并行数据的边界所在位置。其数值范围是 0~9。

#### 4.10.1.6. 手动对齐

端口 l*_rx_slidemode 是控制自对齐的模式。当 l*_rx_slidemode 设置为 0，Word Aligner 工作在自动模式。当 l*_rx_slidemode 设置为 1 或 2，Word Aligner 工作在手动模式。

用户想要把 Word Aligner 设置在自动模式，l*_rx_wordalign_en 应该被设置为 1，再设置其他控制端口。

用户想要把 Word Aligner 设置在手动模式，l*_rx_wordalign_en 应该被设置为 0，l*_rx_show_realigncomma 应该被设置为 0，l*_rx_pcommaalign_en 和 l*_rx_mcommaalign_en 应该被设置为 0。

端口 l*_rx_slidemode 为 1 时，l*_rx_slide 控制并行数据的边界向左移；l*_rx_slidemode 为 2 时，rx slide 控制并行数据的边界向右移。

手动模式下，用户用端口 l*_rx_slide 控制并行数据对齐边界的移位。l*_rx_slide 高有效。l*_rx_slide 在每次被置高前需要保持低电平至少 32 周期。每被置高一次，对齐边界将被向左或向右移动 1bit。

当 RX PMA 和 RX PCS 直接的并行数据位宽被设置为 10 时，对齐边界位置为 9 时，用户再发送一个 l*_rx_slide 高脉冲，对齐边界位置将会变为 0。当 RX PMA 和 RX PCS 直接的并行数据位宽被设置为 20 时，对齐边界位置为 19 时，用户再发送一个 l*_rx_slide 高脉冲，对齐边界位置将会变为 0。

#### 4.10.1.7. Bit 反序

Word Aligner 支持对接收的数据做 Bit 反序。以 10bit 的接收数据为例，bit[9]和 bit[0]对调，bit[8]和 bit[1]对调……以此类推。用户用属性 L*_RX_BITREV_EN 控制 bit 反序是否使能。

#### 4.10.1.8. Byte 反序

Word Aligner 支持 Byte 反序。以 20 位并行数据为例，原本数据排列[19:0]将被转换成{ [9:0], [19:10]}。用户用属性 L*_RX_BYTEREV_EN 控制 byte 反序是否使能。

### 4.10.2 端口和属性

Word Aligner 端口说明如表 4-16 所示。

表 4-16 Word Aligner 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| l*_rx_wordalign_en | In | cr_para_clk * | Word Aligner 模块的使能信号，高有效。 |
| l*_rx_pcommaalign_en | In | cr_para_clk | 检测 COMMA Plus 的使能信号，高有效。 |
| l*_rx_mcommaalign_en | In | cr_para_clk | 检测 COMMA Minus 的使能信号，高有效。 |
| l*_rx_aligncomma_mask[9:0] | In | cr_para_clk | 用来控制 COMMA 序列值的哪些 bit 进行比对，默认值为 10’b1111111111。 |
| l*_rx_aligncomma_double | In | cr_para_clk | 定义 comma 匹配 COMMA Plus 和 COMMA Minus 二者，或者二者之一。高有效。<br>高电平表示二者均需匹配，且一个 Plus 后面紧跟一个 minus。该信号为高时，l*_rx_alignpcomma_det 与 l*_rx_alignmcomma_det 需配置为相等，l*_rx_alignpcomma_en 与 l*_rx_alignmcomma_en 需配置为相等。<br>低电平表示 comma 匹配 COMMA Plus 和 COMMA Minus 二者之一。 |
| l*_rx_alignmcomma_value[9:0] | In | cr_para_clk | 自定义的 COMMA Minus，默认值为 10’b1100000101(K28.5)。<br>该定义不影响编解码。 |
| l*_rx_alignpcomma_value[9:0] | In | cr_para_clk | 自定义的 comma plus，默认值为 10’b0011111010 (K28.5)。<br>该定义不影响编解码。 |
| l*_rx_show_realigncomma | In | cr_para_clk | 对齐之后能够再次对齐调整边界的使能信号。高有效。<br>当该信号为高时，表示对齐之后如果遇到不在当前边界的 comma 可以再次对齐，即 re-align；<br>为低时，表示对齐之后如果遇到不在当前边界的 comma 不会再次对齐。<br>当 l*_rx_aligncomma_double 为高或者在手动对齐模下，该信号不能为高。 |

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| l*_rx_wordaligned | Out | l*_rx_usr2clk | 输出的字节已对齐的指示信号，高有效 |
| l*_rx_wordrealign | Out | l*_rx_usr2clk | 字节对齐状态变化信号，高有效。<br>高电平说明输入的串行数据已经发生变化，模块已经重新对齐。 |
| l*_rx_align_sel[3:0] | Out | | 保留。 |
| l*_rx_slide | In | cr_para_clk | Word Aligner 工作在手动模式下，进行移位的控制信号，高有效。<br>每置高一次，输入数据移动 1bit。每置高前需要保持低电平至少 32 周期。该模式下，用户需配 l*_rx_pcommaalign_en=0；l*_rx_mcommaalign_en=0；l*_rx_wordalign_en=0; l*_rx_show_realigncomma=false。 |
| l*_rx_slidemode[1:0] | In | cr_para_clk | 定义 Word Aligner 模块的工作模式。<br>0：自动模式。该模式下 rx slide 特性不使用。<br>1：手动左移模式。该模式下，需配置 l*_rx_show_realigncomma 为 0。<br>2：手动右移模式。该模式下，需配置 l*_rx_show_realigncomma 为 0。<br>3：保留。 |
| l*_rx_commadet[7:0] | Out | l*_rx_usr2clk | RX Channel 接收到的 COMMA 指示。在 PH1A 系列器件中，该信号和 l*_rx_usrdata_out 同步。 |

表 4-17 Word Aligner 属性列表

| 属性 | 类型 | 描述 |
|---|---|---|
| L*_RX_BITREV_EN | 1bit | Word Aligner 的 bit 反序使能信号，高有效，推荐使用 IP Generator 产生的值。 |
| L*_RX_BYTEREV_EN | 1bit | Word Aligner 的 byte 反序使能信号，高有效，推荐使用 IP Generator 产生的值。 |

### 4.10.3 用法

RX Bit 反序和 RX Byte 反序用法如下：

首先，使能 RX Standard PCS 模式；

使能 RX Word Aligner，l*_rx_wordalign_en 设置为 1’b1；

将 l*_rx_slidemode[1:0]设置为 2’b00；

L*_BITREV_EN 属性设置为 1 使能 RX Bit 反序功能；

L*_RX_BYTEREV_EN 属性设置为 1 使能 RX Byte 反序功能。

## 4.11 8B10B Decoder

### 4.11.1 功能简介

如果接收到的数据是经过 8B/10B 编码的，就需要 8B/10B 解码。和 TX Channel 有 8B/10B Encoder 一样，RX Channel 有 8B/10B Decoder 模块。8B/10B Decoder 输入的 10bit 或 20bit，必须是已经对齐好的并行数据。

8B/10B Decoder 模块有以下特性：

- 支持对输入的 10bit 编码数据进行 8B/10B 解码
- 能同时支持 10bit 输入或 20bit 输入的解码
- 支持对输入数据进行 bypass 处理
- 支持输入数据进行 disparity 校验，并进行错误校验指示
- 支持对输入数据进行不在 8B/10B 码表中的错误检测和指示
- 用户把端口 l*_rx_dec_en 拉高，就使能了本模块；把 l*_rx_dec_en 拉低，就设置本模块为旁路。

### 4.11.2 端口和属性

8B/10B Decoder 端口如表 4-18 所示。

表 4-18 8B/10B Decoder 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| l*_rx_dec_en | In | cr_para_clk | 8B/10B Decoder 模块的使能信号，高有效。 |
| l*_rx_usrdata_width[2:0] | In | cr_para_clk | RX PCS 用户侧数据有效位宽。<br>当 8B/10B Decoder 被使能时， |

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| | | | l*_rx_usrdata_width 只能是 0/2/4/6。<br>• 0 表示 8bit 有效；<br>• 1 表示 10bit 有效；<br>• 2 表示 16bit 有效；<br>• 3 表示 20bit 有效；<br>• 4 表示 32bit 有效；<br>• 5 表示 40bit 有效；<br>• 6 表示 64bit 有效；<br>• 7 表示 80bit 有效。 |
| l*_rx_usrdata_out[79:0] | Out | l*_rx_usr2clk | RX PCS 用户侧数据，有效位宽根据 l*_rx_usrdata_width 得到。<br>• 当 l*_rx_usrdata_width=0 ，l*_rx_usrdata_out[7:0]有效；<br>• 当 l*_rx_usrdata_width=1 ，l*_rx_usrdata_out[9:0]有效；<br>• 当 l*_rx_usrdata_width=2 ，l*_rx_usrdata_out[15:0]有效；<br>• 当 l*_rx_usrdata_width=3 ，l*_rx_usrdata_out[19:0]有效；<br>• 当 l*_rx_usrdata_width=4 ，l*_rx_usrdata_out[31:0]有效；<br>• 当 l*_rx_usrdata_width=5 ，l*_rx_usrdata_out[39:0]有效；<br>• 当 l*_rx_usrdata_width=6 ，l*_rx_usrdata_out[63:0]有效；<br>• 当 l*_rx_usrdata_width=7 ，l*_rx_usrdata_out[79:0]有效； |
| l*_rx_usrdatak_out[7:0] | Out | l*_rx_usr2clk | RX PCS 用户侧数据 K 字符指示，高有效。<br>• l*_rx_usrdatak_out[7] 对应 l*_rx_usrdata_out[63:56]；<br>• l*_rx_usrdatak_out[6] 对应 l*_rx_usrdata_out[55:48]；<br>• l*_rx_usrdatak_out[5] 对应 l*_rx_usrdata_out[47:40]； |

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| | | | • l*_rx_usrdatak_out[4] 对应 l*_rx_usrdata_out[39:32]；<br>• l*_rx_usrdatak_out[3] 对应 l*_rx_usrdata_out[31:24]；<br>• l*_rx_usrdatak_out[2] 对应 l*_rx_usrdata_out[23:16]；<br>• l*_rx_usrdatak_out[1] 对应 l*_rx_usrdata_out[15:8]；<br>• l*_rx_usrdatak_out[0] 对应 l*_rx_usrdata_out[7:0]。 |
| l*_rx_usrdisperr_out[7:0] | Out | l*_rx_usr2clk | 8B/10B Decoder 输入数据错误指示信号。<br>• l*_rx_usrdisperr_out[7] 对应 l*_rx_usrdata_out[63:56]；<br>• l*_rx_usrdisperr_out[6] 对应 l*_rx_usrdata_out[55:48]；<br>• l*_rx_usrdisperr_out[5] 对应 l*_rx_usrdata_out[47:40]；<br>• l*_rx_usrdisperr_out[4] 对应 l*_rx_usrdata_out[39:32]；<br>• l*_rx_usrdisperr_out[3] 对应 l*_rx_usrdata_out[31:24]；<br>• l*_rx_usrdisperr_out[2] 对应 l*_rx_usrdata_out[23:16]；<br>• l*_rx_usrdisperr_out[1] 对应 l*_rx_usrdata_out[15:8]；<br>• l*_rx_usrdisperr_out[0] 对应 l*_rx_usrdata_out[7:0]。 |
| l*_rx_usrcodeerr_out[7:0] | Out | l*_rx_usr2clk | 8B/10B Decoder 输入数据错误指示信号。<br>• l*_rx_usrcodeerr_out[7] 对应 l*_rx_usrdata_out[63:56]；<br>• l*_rx_usrcodeerr_out[6] 对应 l*_rx_usrdata_out[55:48]；<br>• l*_rx_usrcodeerr_out[5] 对应 l*_rx_usrdata_out[47:40]； |

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| | | | • l*_rx_usrcodeerr_out[4] 对应 l*_rx_usrdata_out[39:32]；<br>• l*_rx_usrcodeerr_out[3] 对应 l*_rx_usrdata_out[31:24]；<br>• l*_rx_usrcodeerr_out[2] 对应 l*_rx_usrdata_out[23:16]；<br>• l*_rx_usrcodeerr_out[1] 对应 l*_rx_usrdata_out[15:8]；<br>• l*_rx_usrcodeerr_out[0] 对应 l*_rx_usrdata_out[7:0]。 |

## 4.11.3 说明

### 4.11.3.1. RX 8B/10B 的 bit 和 byte 排序

进入 8B/10B Decoder 的 bit 顺序，跟 8B/10B 码表中的顺序一致。按照码表，8B/10B 解码的 a bit 先被接收到。如果对端的发送器先发 j bit、最后发 a bit，须将 Word Aligner 里的 bit 反序使能，确保进入 RX 8B/10B Decoder 的数据符合 8B/10B 的码表。

### 4.11.3.2. RX Running Disparity

8B/10B Decoder 会进行 Disparity 检测，当检测到接收到的数据有错误的 disparity 时，会驱动端口 l*_rx_usrdisperr_out 成高电平。除了 Disparity 错误以外，8B/10B Decoder 还能检测输入的 10bit 或 20bit 的编码是否在 8B/10B 码表之外。当输入的 10bit 或 2 个 10bit，不能在 8B/10B 码表中查找到时，本模块会驱动端口 l*_rx_usrcodeerr_out 为高电平，以指示查表解码出错。

图 4-15 是 8B/10B 解码错误举例，波形输入数据有好数据 A、带 Disparity 错误的数据 B、8B/10B 码表外数据 C 和带 Disparity 错误的码表外数据 D。

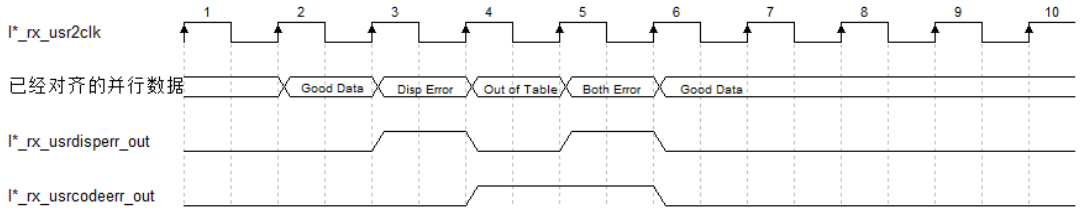


图 4-15 8B/10B 解码错误举例

### 4.11.3.3. 特殊字符

8B/10B 编码中，有一些特殊字符，也叫 K 字符，通常被用作控制信号。当已对齐好的并行数据输入是 K 字符时，本模块会驱动 l*_rx_usrdatak_out 为高。

#### 4.11.3.4. 注意事项

端口 l*_rx_usrdisperr_out 和 l*_rx_usrcodeerr_out 相互影响，当 l*_rx_usrdisperr_out 为高电平时，l*_rx_usrcodeerr_out 也会拉高；当 l*_rx_usrcodeerr_out 为高电平时，l*_rx_usrdisperr_out 也会拉高，无法单一从某一个信号判断是极性错误还是编码错误。

## 4.12 Rate Match FIFO

### 4.12.1 功能简介

SERDES RX Channel 的 RX Standard PCS 内有 3 个并行数据的时钟域：RX PMA/PCS 并行数据时钟域(l*_rx_xclk)、RX PCS 内部并行数据时钟域 l*_rx_rmatclk 和 RX PCS FIFO 的读时钟域，即 RX 用户时钟。为了接收数据，RX PMA/PCS 并行数据时钟域(l*_rx_xclk) 频率必须非常接近 RX PCS 并行数据时钟域 l*_rx_rmatclk 频率。这样，这两个时钟域之间的频率差才能得到解决。

RX Standard PCS 包含了一个缓冲器 Rate Match FIFO，来解决 l*_rx_xclk 和 l*_rx_rmatclk 之间的时钟偏差。

Rate Match FIFO 模块可以接收到的数据从 l*_rx_xclk 同步到 l*_rx_rmatclk 时钟域上。如果 l*_rx_rmatclk 为 RX CDR 恢复时钟，则该 FIFO 可以被旁路。如果 l*_rx_rmatclk 为本地时钟，则该模块完成时钟补偿功能，根据 FIFO 状态在数据中增加或删除 SKIP 序列。

本模块的特性如下：

- 支持对输入数据增删 SKIP 序列进行频率补偿
- 支持速率适配模块可选，bypass 时延时最小
- 支持四种 SKIP 序列图案可配
- 支持 SKIP 序列最小匹配单位可配为 1,2,4 字节
- 支持高低水线用户可配
- 支持删除后保留的 SKIP 序列组数可配
- Rate Match FIFO 模块频率补偿仅支持 8B10B 解码器使能的模式（频率补偿增删 IDLE 可能会破坏编码数据的 disparity，因此必须先解码再进行频率补偿）

### 4.12.2 端口和属性

Rate Match FIFO 端口如表 4-19 所示。

表 4-19 Rate Match FIFO 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| l*_rx_rmfifo_en | In | cr_para_clk | Rate Match FIFO 使能信号，高有效。 |

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| l*_rx_rmfifo_rst_n | In | cr_para_clk | Rate Match FIFO 复位信号，使用时，需经过去除 glitch 处理，同步到本地时钟域。 |
| l*_rx_rmfifo_overflow | Out | l*_rx_rmatclk | Rate Match FIFO 溢出指示，高有效。 |
| l*_rx_rmfifo_underflow | Out | l*_rx_rmatclk | Rate Match FIFO 空指示，高有效。 |
| l*_rx_rmfifo_skip_1byte[9:0] | In | cr_para_clk | 用户自定义的第一个下插 SKIP 序列，默认值为{2’b00,8’h00}。 |
| l*_rx_rmfifo_skip_2byte[9:0] | In | cr_para_clk | 用户自定义的第二个下插 SKIP 序列，默认值为{2’b00,8’h00}。 |
| l*_rx_rmfifo_skip_3byte[9:0] | In | cr_para_clk | 用户自定义的第三个下插 SKIP 序列，默认值为{2’b01,8’hbc}。 |
| l*_rx_rmfifo_skip_4byte[9:0] | In | cr_para_clk | 用户自定义的第四个下插 SKIP 序列，默认值为{2’b00,8’h50}。 |
| l*_rx_rmfifo_skip4byte_en | In | cr_para_clk | 匹配 SKIP 的字节，高有效，表示匹配字节为（1，2，3，4）。 |
| l*_rx_rmfifo_skip2byte_en | In | cr_para_clk | 匹配 SKIP 的字节，高有效，表示匹配字节为（3，4）。 |
| l*_rx_rmfifo_min_ipg_cnt[3:0] | In | cr_para_clk | 删除几组 SKIP 序列后，至少剩余的 SKIP 序列的组数（每组 SKIP 序列为 1 个，2 个还是 4 个由 skip2byte_en 和 skip4byte_en 决定）。取值范围是 0-15，默认值为 3。<br>• 0：删除后至少保留 0 组 SKIP 序列；<br>• 1：删除后至少保留 1 组 SKIP 序列；<br>• 2：删除后至少保留 2 组 SKIP 序列；<br>• 3：删除后至少保留 3 组 SKIP 序列；<br>• 4：删除后至少保留 4 组 SKIP 序列；<br>…<br>• 15：删除后至少保留 15 组 SKIP |

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| | | | 序列。 |
| l*_rx_rmfifo_high_mark[4:0] | In | cr_para_clk | 用户自定义的高水线，高于该水线则删除 SKIP 序列，默认值为 4’b1000。 |
| l*_rx_rmfifo_low_mark[4:0] | In | cr_para_clk | 用户自定义的低水线，低于该水线则增加 SKIP 序列，默认值为 4’b0110。 |
| l*_rx_rmfifo_re_out | Out | l*_rx_rmatclk | 输出的读使能，高有效。为低表示此时的数据为增加的 SKIP 序列。 |
| l*_rx_rmfifo_we_out | Out | l*_rx_rmatclk | 输出的写使能，高有效。为低表示此时正在删除 SKIP 序列。 |
| l*_rx_rmfifo_delay[5:0] | Out | l*_rx_rmatclk | RATE MATCH FIFO 延迟指示。 |

## 4.12.3 用法

### 4.12.3.1. Rate Match FIFO 的使用

下面的设置用来使能 Rate Match FIFO，解决 l*_rx_xclk 和 l*_rx_usrclk 之间的相位差。

- l*_rx_rmfifo_en = 1

Rate Match FIFO 的端口 l*_rx_rmfifo_overflow 和 l*_rx_rmfifo_underflow 指示这 FIFO 的满或空状态。当满或空状态发生时，FIFO 里的数据就不再可用。这种情况下，Rate Match FIFO 应该被复位。可用做复位的信号有：l*_rx_pcs_rst_n 和 l*_rx_rmfifo_rst_n。

Rate Match FIFO 也可用作时钟补偿。当 l*_rx_xclk 和 l*_rx_rmatclk 时钟不匹配时，Rate Match FIFO 可以补偿两个时钟的频差。表 4-20 列举了通常的时钟配置，以及是否需要时钟补偿的情况。

表 4-20 同异步系统是否需要时钟补偿列表

| 时钟类型 | 是否需要时钟补偿 |
|---|---|
| 同步系统，收发两端都把同一个晶振作为参考时钟源。 | NO |
| 异步系统，收发两端用不同的参考时钟，接收端用恢复时钟 | NO |
| 异步系统，收发两端用不同的参考时钟，接收端用本地时钟 | YES |

Rate Match FIFO 被使用时，将会在 RX 的数据通路上引入不确定性延时。本模块不支持通道绑定功能，用户在 Fabric 逻辑里实现该功能。

### 4.12.3.2. 时钟补偿

Rate Match FIFO 是为了桥接 l*_rx_rmatclk 和 l*_rx_xclk。其中，在正常情况下，l*_rx_xclk 是 RX CDR 的恢复时钟。在 l*_rx_xclk 选择本地时钟时，尽管 l*_rx_rmatclk 和 l*_rx_xclk 在名义上

同频，但实际上，它们之间还是会存在一些很小的频差。因为 I*_rx_rmatclk 和 I*_rx_xclk 不是真正的同频，它们之间的频差会被累积，最终会导致 Rate Match FIFO 上溢或下溢，除非有时钟补偿。为了实现时钟补偿，每个 SERDES 发送器会每隔一段时间就发送一个或多个特殊字符即 SKIP 序列，SERDES 接收器可以在必要时在 Rate Match FIFO 内移除或增加这些特殊字符。当 FIFO 超过高水线，也就是 FIFO 半满时，移除特殊字符；当 FIFO 越过低水线，也就是 FIFO 半空时，插入特殊字符，通过移除和插入，来防止 FIFO 发生上溢或下溢。图 4-16 为时钟补偿概念图。

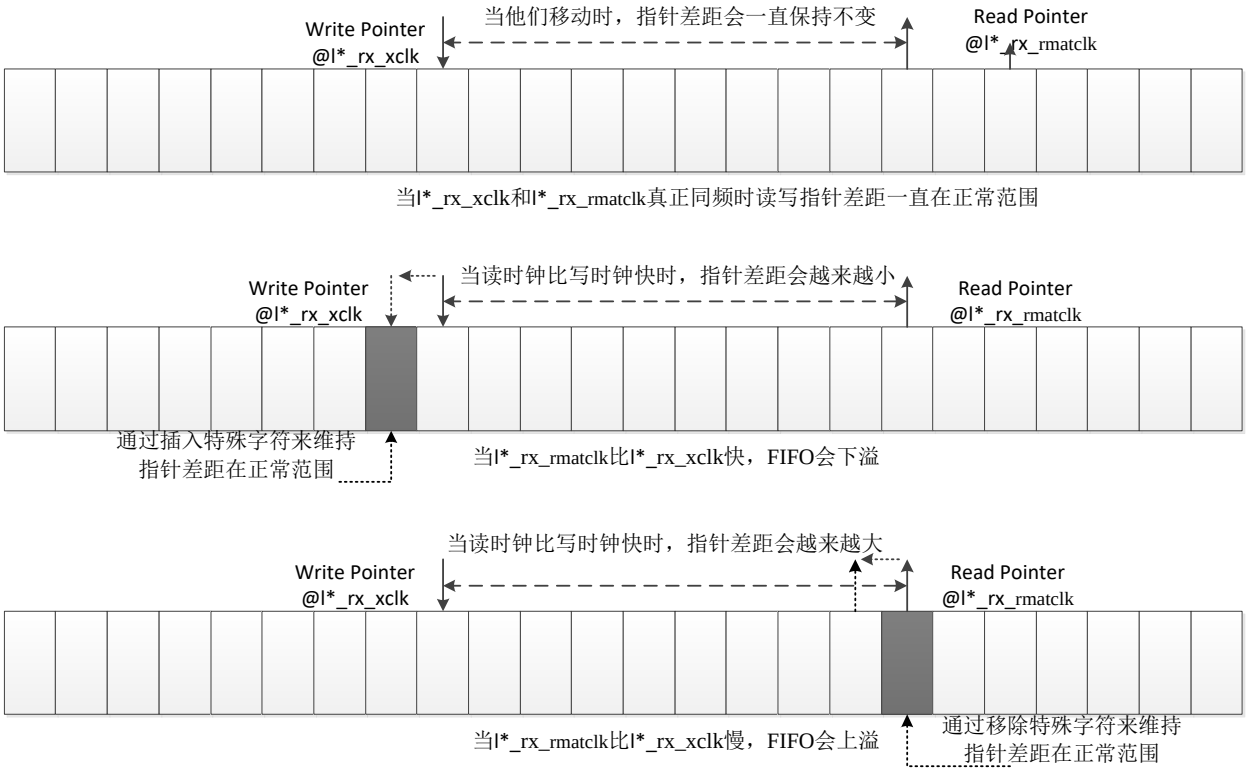


图 4-16 时钟补偿概念图

用户要使用时钟补偿功能，请遵循以下小节描述的步骤：使能时钟补偿、设置 Rate Match FIFO 参数、设置时钟补偿 SKIP 序列。

### 使能时钟补偿

每个 SERDES Channel 都含有时钟补偿电路，它能控制 Rate Match FIFO 的指针来实现时钟补偿功能。为了使用 Rate Match FIFO，将端口 I*_rx_rmfifo_en 设置为 1，使能这个 FIFO。使能了这个 FIFO，时钟补偿控制电路就被使能了。

当 FIFO 的数据太多或太少时，时钟补偿控制电路检测到匹配的序列，时钟补偿功能就被触发。为了使用时钟补偿功能，用户需要配置以下几项：

- Rate Match FIFO 的水线
- 时钟补偿 SKIP 序列

### 设置 Rate Match FIFO 水线

Rate Match FIFO 的水线有 2 个：I*_rx_rmfifo_hiGH_mark 和 I*_rx_rmfifo_low_mark。I*_rx_rmfifo_hiGH_mark 和 I*_rx_rmfifo_low_mark 的默认值，分别是 4’b1000 和 4’b0110。

当 FIFO 里的数据太少，水位低于给 FIFO 设定的低水线时，时钟补偿控制电路在第一个匹配到的时钟补偿 SKIP 序列之后，会写入额外的一些时钟补偿 SKIP 序列，从而防止 FIFO 下溢。同样的，当 FIFO 里的数据太多，水位高于给 FIFO 设定的高水线时，时钟补偿控制电路会从匹配到的时钟补偿 SKIP 序列的第一个 byte 开始，删除一些时钟补偿 SKIP 序列。用户可以根据应用需要设置这 2 个选项。

### 设置时钟补偿 SKIP 序列

设置时钟补偿 SKIP 序列的端口有：I*_rx_rmfifo_skip_1byte 、I*_rx_rmfifo_skip_2byte 、I*_rx_rmfifo_skip_3byte 、I*_rx_rmfifo_skip_4byte 、I*_rx_rmfifo_skip4byte_en 、I*_rx_rmfifo_skip2byte_en、I*_rx_rmfifo_min_ipg_cnt。

用户用端口 I*_rx_rmfifo_skip_1byte 、I*_rx_rmfifo_skip_2byte 、I*_rx_rmfifo_skip_3byte 、I*_rx_rmfifo_skip_4byte 来设置时钟补偿 SKIP 子序列。

当 8B/10 Decoder 被使能时，Rate Match FIFO 的输入连到 Decoder 的输出。这使得电路寻找带 Disparity 的并行 8bit 数据，不管是正 Disparity 还是负 Disparity，并区别 K 字符和正常的数据字符。图 4-17 展示了当 8B/10 Decoder 被使能时，如何设置时钟补偿 SKIP 序列。


图 4-17 8B/10B Decoder 使能时时钟补偿子序列的设置

I*_rx_rmfifo_skip4byte_en、I*_rx_rmfifo_skip2byte_en 用来定义一组 SKIP 序列由几个字节的时钟补偿 SKIP 子序列组成。具体数值，请见表 4-21。

表 4-21 时钟补偿子序列字节数的配置表

| 具体设置 | 时钟补偿子序列 SKIP 的字节数 | 匹配字节 |
|---|---|---|
| I*_rx_rmfifo_skip4byte_en=0<br>且 I*_rx_rmfifo_skip2byte_en=0 | 1 | I*_rx_rmfifo_skip_4byte |
| I*_rx_rmfifo_skip4byte_en=0<br>且 I*_rx_rmfifo_skip2byte_en=1 | 2 | （I*_rx_rmfifo_skip_3byte，<br>I*_rx_rmfifo_skip_4byte） |

| | | |
|---|---|---|
| I*_rx_rmfifo_skip4byte_en=1<br>且 I*_rx_rmfifo_skip2byte_en=1 | 4 | （I*_rx_rmfifo_skip_1byte，<br>I*_rx_rmfifo_skip_2byte，<br>I*_rx_rmfifo_skip_3byte，<br>I*_rx_rmfifo_skip_4byte） |

端口 I*_rx_rmfifo_min_ipg_cnt 用于设置最小剩余的 SKIP 序列的组数，表示在删除 SKIP 序列之后，还剩余的 SKIP 序列组数的最小值。比如 I*_rx_rmfifo_min_ipg_cnt 设置为 1，发生删除之后，RX 用户数据端口至少还能收到 1 组 SKIP 序列。如果需要删除的组数较大，导致剩余 SKIP 的组数少于 I*_rx_rmfifo_min_ipg_cnt 设定的值，则删除动作不会发生。信号只影响删除 SKIP 序列，不会影响增加 SKIP 序列。

**观察时钟补偿**

输出端口 I*_rx_rmfifo_re_out 和 I*_rx_rmfifo_we_out 用于指示此时时钟补偿电路在增加还是删除 SKIP 序列。I*_rx_rmfifo_re_out 和 I*_rx_rmfifo_we_out 平时为高电平，当发生插入或者删除操作，会有低脉冲指示。I*_rx_rmfifo_re_out 低脉冲指示发生了 SKIP 序列增加，I*_rx_rmfifo_we_out 低脉冲指示发生了 SKIP 序列删除。

端口 I*_rx_rmfifo_overflow 和 I*_rx_rmfifo_underflow 如果有高电平输出，不仅能指示 I*_rx_usrclk 比 I*_rx_xclk 快还是慢，而且还说明用户设置的 I*_rx_rmfifo_hiGH_mark 要增大或 I*_rx_rmfifo_low_mark 要减小。

#### 4.12.3.3. RX Rate Match FIFO 使用注意事项

使用 RX Rate Match FIFO 用于时钟频率补偿时，应选择 10bit 内部位宽模式，即 I*_rx_pmadata_width[1:0]设为 2’b01。8B10B Decoder 必须使能，即 I*_rx_dec_en 设为 1’b1。在 10bit 内部位宽模式下，用户数据位宽即外部数据位宽最高为 32bit。

## 4.13 Byte De-serializer

### 4.13.1 功能简介

当 RX PCS 用户侧数据位宽是 RX PCS 内部数据位宽的倍数时，RX PCS 需要实现位宽转换。本 RX Byte Serializer 能支持 1:2 和 1:4 两种数据位宽转换。当位宽比例为 1:1 时，用户须设置本 TX Byte Serializer 为旁路。RX PCS 内部数据位宽的设置，请见 RX Channel 的 De-serializer 模块。

Byte De-serializer 根据端口 I*_rx_bytedeserial_en 和 I*_rx_bytedeserial_mode，对输入并行数据进行 1:2 或者 1:4 位宽转换处理。

本模块的特性如下：

- 支持对输入数据数据和输入控制信号进行位宽转换。
- 支持对输入数据按字节(8bits/10bits)为粒度进行位宽转换。

- 支持对 8 位有效位宽的输入数据按照 8bits 为粒度进行 1:4，1:2 的位宽转换。
- 支持对 10 位有效位宽的输入数据按照 10bits 为粒度进行 1:4，1:2 的位宽转换。
- 支持对 16 位有效位宽的输入数据按照 16bits 为粒度进行 1:4，1:2 的位宽转换。
- 支持对 20 位有效位宽的输入数据按照 20bits 为粒度进行 1:4，1:2 的位宽转换。
- 支持对各种有效位宽的输入数据通过配置 I*_rx_bytedeserial_en 为低，bypass 该模块，使得输出数据位宽等于输入数据位宽。

### 4.13.2 端口和属性

Byte De-serializer 端口如表 4-22 所示。

表 4-22 Byte De-serializer 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| I*_rx_bytedeserial_en | In | cr_para_clk | Byte De-serializer 模块使能信号，高有效。 |
| I*_rx_bytedeserial_mode | In | cr_para_clk | Byte De-serializer 位宽转换比例：<br>• 0：表示位宽转换比为 1:2；<br>• 1：表示位宽转换比为 1:4。 |

### 4.13.3 用法

RX Native PCS 中的 RX Byte De-serializer 功能与 RX Standard PCS 中的相同。

不同之处在于，RX Native PCS 内的 Byte De-serializer 必须在 RX Native PCS 使能时才能使用，而 RX Standard PCS 中的 Byte De-serializer 必须在 RX Standard PCS 使能时才能使用。

RX Native PCS 内的 Byte De-erializer 的端口和属性跟 RX Standard PCS 中 Byte De-erializer 一样。

## 4.14 RX PCS FIFO

### 4.14.1 功能简介

如 RX Channel 的时钟域图所示，RX Standard PCS 内有 3 个并行数据的时钟域：RX PMA/PCS 并行数据时钟域(I*_rx_xclk)、RX PCS 内部并行数据时钟域 I*_rx_rmatclk 和 RX PCS FIFO 的读时钟域用户时钟 I*_rx_usr2clk。对于 RX Standard PCS 内的 RX PCS FIFO，为了完整的接收数据，I*_rx_rmatclk 必须匹配用户时钟速率，但两者之间可能有相位差。RX PCS FIFO 解决这两个时钟域之间的相位差问题。

如 RX Channel 的时钟域图（在‘RX Channel 总览’一节）所示，RX Native PCS 内有 2 个并行数据的时钟域：RX PMA/PCS 并行数据时钟域(I*_rx_xclk)、和 RX PCS FIFO 的读时钟域，即用户时钟。

对于 RX Native PCS 内的 RX PCS FIFO，为了完整的接收数据，l*_rx_xclk 必须匹配用户时钟速率。RX PCS FIFO 解决这两个时钟域之间的相位差问题。

本模块通过 FIFO 实现，完成相位补偿的功能。对 RX Standard PCS 来说，该模块写时钟为 l*_rx_rmatclk，读时钟为用户时钟。对 RX Native PCS 来说，该模块写时钟为 l*_rx_xclk，读时钟为用户。

本模块的特性如下：

- 支持对输入数据和输入控制信号进行跨时钟域处理。
- 支持最大的输入数据有效位宽 20bit。
- 支持对 RX PCS FIFO 模块进行 bypass 处理。Bypass 时数据延时较小，而且确定。

## 4.14.2 端口和属性

RX PCS FIFO 端口及属性分别如表 4-23、表 4-24 所示。

表 4-23 RX PCS FIFO 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| l*_rx_pcsfifo_en | In | cr_para_clk | RX PCS FIFO 模块使能信号，高有效。 |
| l*_rx_pcsfifo_rst_n | In | cr_para_clk | RX PCS FIFO 复位信号，使用时，需经过去除 glitch 处理，同步到本地时钟域。 |
| l*_rx_dff_en | In | cr_para_clk | Rx DFF 使能信号，在 RX PCS FIFO 不使能时，如果该信号为高，则输入数据经过一级 l*_rx_rmatclk/l*_rx_xclk 驱动的 DFF。 |
| l*_rx_pcsfifo_delay[3:0] | Out | l*_rx_usr2clk | 数据经过 RX PCS FIFO 的延迟指示。 |
| l*_rx_pcsfifo_overflow | Out | l*_rx_usr2clk | RX PCS FIFO 模块出现上溢错误指示信号，高电平有效。 |
| l*_rx_pcsfifo_underflow | Out | l*_rx_usr2clk | RX PCS FIFO 模块出现下溢错误指示信号，高电平有效。 |

表 4-24 RX PCS FIFO 相关属性

| 属性名 | 类型和宽度 | 说明 |
|---|---|---|
| L*_RX_PCSFIFO_RD_INIT | 3 | RX PCS FIFO 模块初始读地址，该信号设置不同，可以使得 FIFO 的延迟不同。 |

## 4.14.3 用法

RX Native PCS 中的 RX PCS FIFO 功能与 RX Standard PCS 中的相同。不同之处在于，RX Standard

PCS 内 RX PCS FIFO 写时钟是 l*_rx_rmatclk，而 RX Native PCS 内 RX PCS FIFO 写时钟是 l*_rx_xclk。

使用的不同之处，RX Native PCS 中的 RX PCS FIFO 必须在 RX Native PCS 使能时才能使用，而 RX Standard PCS 中的 RX PCS FIFO 必须在 RX Standard PCS 使能时才能使用。

RX Native PCS 中的 RX PCS FIFO 的端口和属性跟 RX Standard PCS 中 RX PCS FIFO 一样。

#### 4.14.3.1. RX PCS FIFO Bypass 用法

RX Standard PCS 主要有三个时钟域，一个是 RX XCLK 时钟域，一个是 l*_rx_rmatclk 时钟域，还有一个是 l*_rx_usrclk/l*_rx_usr2clk 时钟域，见图 4-2。通常 RX XCLK 时钟可以选择 l*_rx_recclk（恢复时钟），在不使用 rate match 功能的时候 l*_rx_rmatclk 也选择 l*_rx_recclk，l*_rx_usrclk/l*_rx_usr2clk 时钟是由 l*_rx_outclk 或者 l*_rx_out2clk 经过时钟 BUFFER 产生的，因此 l*_rx_usrclk/l*_rx_usr2clk 时钟和 l*_rx_recclk 时钟是有相位差的，RX PCS FIFO 的作用是解决这两个时钟域之间的相位差。当 RX PCS FIFO 被旁路时，用户设计需解决这两个时钟的相位差。

解决这个相位差有几个办法，一个是由 l*_rx_outclk（通常是 l*_rx_recclk）通过 LCLK 来产生 l*_rx_usrclk/l*_rx_usr2clk 时钟，另一个办法是用 PLL 解决 l*_rx_usrclk/l*_rx_usr2clk 和 l*_rx_recclk 之间的相位差。

**RX PCS FIFO Bypass - LCLK**

使用 LCLK 解决 RX PCS FIFO Bypass 的时钟连接方法见图 4-18。

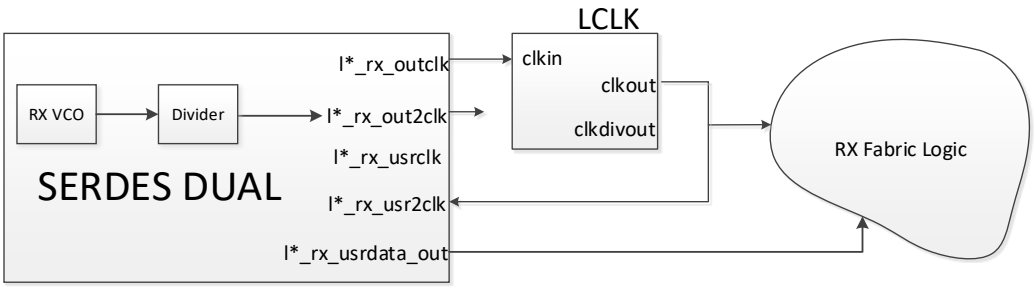


图 4-18 RX PCS FIFO Bypass 模式下用 LCLK 产生用户时钟

首先，将 l*_rx_outclk 送入 LCLK 的 clkin，不能使用 l*_rx_out2clk 作为 LCLK 的输入时钟。然后，l*_rx_outclk 频率需要配置成和 l*_rx_usr2clk 频率相等，因为 LCLK 的分频功能 clkdivout 在这里不能使用。若 l*_rx_outclk 频率无法配置成和 l*_rx_usr2clk 相等的频率，考虑使用下一节的 RX PCS FIFO Bypass - PLL 方案。最后，LCLK 的 clkout 输出端口送回 l*_rx_usr2clk 同时驱动用户逻辑。

**RX PCS FIFO Bypass - PLL**

使用 PLL 解决 RX PCS FIFO Bypass 的时钟连接方法见图 4-19。

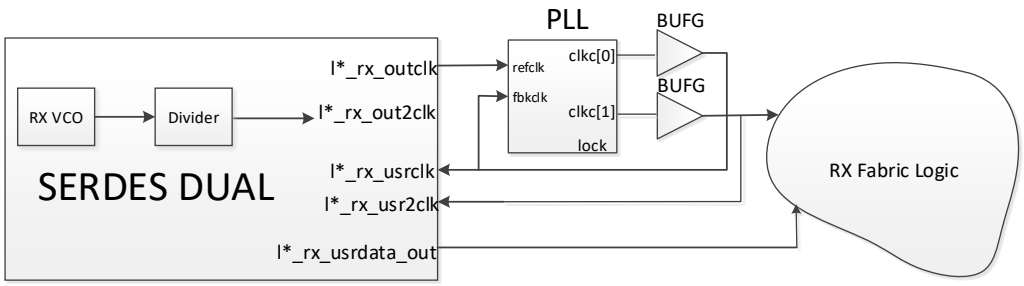


图 4-19 RX PCS FIFO Bypass 模式下用 PLL 产生用户时钟

l*_rx_outclk 送入 PLL 的 refclk 输入端口，PLL 的 2 个时钟输出端口分别输出 l*_rx_outclk 同频时钟，和 1，2 或 4 分频时钟（根据内外位宽比例选择），分频时钟送给 l*_rx_usr2clk 和用户逻辑，同频时钟送给 l*_rx_usrclk 和 PLL 的 fbkclk 输入，设置 PLL 输出时钟和输入时钟相位对齐模式。

## 4.15 RX Fabric Interface

### 4.15.1 功能简介

RX Fabric Interface 是 SERDES RX Channel 的数据通路跟 FPGA Fabric 的接口。用户于 l*_rx_usr2clk 的上升沿，在 RX Fabric Interface 端口上，读取 SERDES RX Channel 接收到的数据。这个端口是数据位宽可以被配置成 8/10/16/20/32/40/64/80 bit。端口 l*_rx_usrdata_width 用于设置数据的有效位宽。RX Fabric Interface 的并行时钟 l*_rx_usr2clk 的频率，由 RX 线速和 RX Fabric Interface 的有效并行数据位宽决定。在某些操作模式下，需提供另一个并行时钟 l*_rx_usrclk 给 TX PCS 的内部并行时钟用。这小节描述了如何驱动并行时钟，解释对于时钟的要求。

#### 4.15.1.1. 接口数据位宽配置

PH1A 系列 FPGA SERDES 中 RX Channel 的内部数据位宽，可以是 8/10/16/20 bit。 当 RX Standard PCS 中的 8B/10B Encoder 被使能时，RX Channel 内部数据位宽只能是 10 bit 的倍数，即 10/20 bit，相对应的，RX Fabric Interface 的数据位宽必须是 8 bit 的倍数，即 8/16/32/64 bit。表 4-25 为 RX PCS 数据通路位宽配置表。

表 4-25 RX PCS 数据通路位宽配置表

| RX PCS 模块控制信号 | RX Fabric Interface 数据位宽控制信号 | RX Channel 用户侧数据 | RX Channel 内部数据位宽 |
|---|---|---|---|
| l*_pcs_standard_en, l*_pcs_native_en, l*_rx_dec_en | l*_rx_usrdata_width [2:0] | l*_rx_usrdata_in [79:0] | |
| l*_pcs_standard_en = 1’b1 且 l*_pcs_native_en = 1’b0 且 l*_rx_dec_en = 1’b1 | 3’b000 | 8 | 10 |
| | 3’b010 | 16 | 10/20 |
| | 3’b100 | 32 | 10/20 |

| RX PCS 模块控制信号 | RX Fabric Interface 数据位宽控制信号 | RX Channel 用户侧数据 | RX Channel 内部数据位宽 |
|---|---|---|---|
| l*_pcs_standard_en, l*_pcs_native_en, l*_rx_dec_en | l*_rx_usrdata_width [2:0] | l*_rx_usrdata_in [79:0] | |
| | 3’b110 | 64 | 20 |
| l*_pcs_standard_en = 1’b1 且 l*_pcs_native_en = 1’b0 且 l*_rx_dec_en = 1’b0 或 l*_pcs_standard_en = 1’b0 且 l*_pcs_native_en = 1’b1 | 3’b000 | 8 | 8 |
| | 3’b001 | 10 | 10 |
| | 3’b010 | 16 | 8/16 |
| | 3’b011 | 20 | 10/20 |
| | 3’b100 | 32 | 8/16 |
| | 3’b101 | 40 | 10/20 |
| | 3’b110 | 64 | 16 |
| | 3’b111 | 80 | 20 |

由上表可以看出，RX Channel 的用户侧数据位宽跟内部数据位宽，可以是 1:1/2:1/4:1。（当 8B/10B Encoder 使能时，用户侧的 8bit 数据等同于内部数据 10bit。）相对应的，RX Channel 的用户侧时钟跟内部时钟的频率，可以是 1:1/1:2/1:4。

#### 4.15.1.2. l*_rx_usrclk 和 l*_rx_usr2clk 的产生

RX Fabric Interface 接口有 2 个并行时钟输入：l*_rx_usrclk 和 l*_rx_usr2clk。这两个时钟都可以作为用户接口的时钟，仅需要接其中一个，另一个接地。该时钟将 SERDES 内部并行数据送出 SERDES，给用户逻辑使用和处理。SERDES 具体使用哪个时钟作为 RX 并行数据的输出时钟，取决于 RX PCS FIFO 是否使能，以及 RX PCS FIFO 读出时钟的选择信号（l*_rx_oclk_sel）。用户可以直接在 TD 软件的 IP Generator 中进行配置。IP Generator 产生的 SERDES 应用通常使用 l*_rx_usr2clk，并将 l*_rx_usrclk 接地。在 RX PCS FIFO Bypass 的应用中，可能会使用 l*_rx_usrclk 作为 RX 并行数据的输出时钟。在 RX PCS FIFO 使能的应用中，l*_rx_usr2clk 或 l*_rx_usrclk 用于驱动 RX PCS FIFO 的读时钟。l*_rx_usr2clk（或 l*_rx_usrclk）的频率跟 RX 线速率和 RX Channel 用户侧数据位宽（rx_usr_datawidth）有关，其对应关系如表 4-25 所示。下面的表达式阐述了如何计算 l*_rx_usrclk 和 l*_rx_usr2clk 的频率。

$$l*\_rx\_usr2clk\_freq = \frac{l*\_rx\_linerate}{l*\_rx\_usr\_datawidth}$$

l*_rx_usr2clk 是 RX Fabric Interface 信号的主要同步时钟。RX Fabric Interface 的数据信号，在 l*_rx_usr2clk 的上升沿打出，进入到用户逻辑。表 4-26 为 RX CLK 和 l*_rx_usr2clk 的频率关系表。

表 4-26 RX CLK 和 l*_rx_usr2clk 的频率关系表

| RX XCLK（SERDES RX 内部工作时钟）和 I*_rx_usr2clk/I*_rx_usr2clk 的频率比 | I*_rx_usrdata_width | I*_rx_pmadata_width |
|---|---|---|
| 1:1 | 0，代表 8bit | 0，代表 8bit |
| | 1，代表 10bit | 1，代表 10bit |
| | 2，代表 16bit | 2，代表 16bit |
| | 3，代表 20bit | 3，代表 20bit |
| 2:1 | 2，代表 16bit | 0，代表 8bit |
| | 3，代表 20bit | 1，代表 10bit |
| | 4，代表 32bit | 2，代表 16bit |
| | 5，代表 40bit | 3，代表 20bit |
| 4:1 | 4，代表 32bit | 0，代表 8bit |
| | 5，代表 40bit | 1，代表 10bit |
| | 6，代表 64bit | 2，代表 16bit |
| | 7，代表 80bit | 3，代表 20bit |

I*_rx_usrclk 和 I*_rx_usr2clk 二者仅需提供一个，具体使用哪个取决于 SERDES 的配置。通常使用的是 I*_rx_usr2clk。用户可以使用 BUFG 或 LCLK 或者 PLL 来驱动 I*_rx_usrclk 或 I*_rx_usr2clk.。

用户可以用 I*_tx_outclk 来驱动 I*_rx_usrclk 或 I*_rx_usr2clk，若本地时钟和恢复时钟有频差，需使能 Rate Match FIFO，作为时钟补偿功能用。

用户还可以用 I*_rx_outclk（用 I*_rx_outclk_sel 选择 RX CDR 的恢复时钟）来驱动 I*_rx_usrclk 和 I*_rx_usr2clk。此时，Rate Match FIFO 可以被设置为旁路。

## 4.15.2 端口和属性

RX Fabric Interface 端口如表 4-27 所示。

表 4-27 RX Fabric Interface 端口列表

| 端口 | 方向 | 时钟域 | 描述 |
|---|---|---|---|
| I*_rx_usrdata_width [2:0] | In | cr_para_clk | RX Channel 用户侧数据有效位宽。<br>• 0 表示 8bit 有效；<br>• 1 表示 10bit 有效；<br>• 2 表示 16bit 有效；<br>• 3 表示 20bit 有效； |

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| | | | • 4 表示 32bit 有效；<br>• 5 表示 40bit 有效；<br>• 6 表示 64bit 有效；<br>• 7 表示 80bit 有效。 |
| l*_rx_usrdata_out[79:0] | In | l*_rx_usr2clk | RX Channel 接收端用户侧输出数据，有效位宽由 l*_rx_usrdata_width 决定。<br>• l*_rx_usrdata_width=0，l*_rx_usrdata_out[7:0]有效；<br>• l*_rx_usrdata_width=1，l*_rx_usrdata_out[9:0]有效；<br>• l*_rx_usrdata_width=2，l*_rx_usrdata_out[15:0]有效；<br>• l*_rx_usrdata_width=3，l*_rx_usrdata_out[15:0]有效；<br>• l*_rx_usrdata_width=4，l*_rx_usrdata_out[31:0]有效；<br>• l*_rx_usrdata_width=2，l*_rx_usrdata_out[39:0]有效；<br>• l*_rx_usrdata_width=6，l*_rx_usrdata_out[63:0]有效；<br>• l*_rx_usrdata_width=7，l*_rx_usrdata_out[79:0]有效。 |
| l*_rx_usrdatak_out[7:0] | Out | l*_rx_usr2clk | RX Channel 接收端用户侧 K 字符指示，高有效。<br>• l*_rx_usrdatak_out [7] 对应 l*_rx_usrdata_out[63:56]；<br>• l*_rx_usrdatak_out [6] 对应 l*_rx_usrdata_out[55:48]；<br>• l*_rx_usrdatak_out [5] 对应 l*_rx_usrdata_out[47:40]；<br>• l*_rx_usrdatak_out [4] 对应 l*_rx_usrdata_out[39:32]；<br>• l*_rx_usrdatak_out [3] 对应 l*_rx_usrdata_out[31:24]；<br>• l*_rx_usrdatak_out [2] 对应 |

| 端口 | 方向 | 时钟域 | 描述 |
| --- | --- | --- | --- |
| | | | l*_rx_usrdata_out[23:16]；<br>• l*_rx_usrdatak_out [1] 对应 l*_rx_usrdata_out[15:8]；<br>• l*_rx_usrdatak_out [0] 对应 l*_rx_usrdata_out[7:0]。 |
| l*_rx_usrdisperr_out[7:0] | Out | l*_rx_usr2clk | RX Channel 接收到的数据错误指示信号，高有效。 |
| l*_rx_usrcodeerr_out[7:0] | Out | l*_rx_usr2clk | RX Channel 接收到的数据错误指示指示信号，高有效。 |
| l*_rx_usrclk | In | 时钟 | 该时钟由用户产生，为 RX Channel 用户侧时钟，频率等于 RX 线速率除以 RX PCS/PMA 位宽。<br>l*_rx_outclk 可以先驱动专用或者全局时钟驱动器，来驱动该时钟。 |
| l*_rx_usr2clk | In | 时钟 | 该时钟由用户产生，为 RX Channel 用户侧时钟，频率等于 RX 线速率除以 RX PCS 用户侧位宽。l*_rx_outclk 或 l*_tx_outclk 可以先驱动专用或者全局时钟驱动器，来驱动该时钟。 |
| l*_rx_outclk | Out | 时钟 | 该时钟由 SERDES Common 的 PLL 模块产生，它由 RX PMA 的输出的时钟驱动。该时钟能能先驱动专用或者全局时钟驱动器，产生用户时钟 l*_rx_usrclk 。 |
| l*_rx_out2clk | Out | 时钟 | 该时钟由 SERDES Common 的 PLL 模块产生，为 RX CDR 恢复时钟再分频的时钟。 |

## 4.15.3 用法

l*_rx_outclk、l*_rx_out2clk 输出的时钟可以送到全局时钟走线（通过 BUFG 或 PLL），也可以送到区域时钟走线（通过 LCLK），用于驱动相应的用户逻辑，以及 RX Fabric Interface 接口的输入时钟 l*_rx_usrclk 或者 l*_rx_usr2clk。l*_rx_usrclk 或者 l*_rx_usr2clk 用于向并行数据输出到 SERDES 外部给 Fabric Logic 进行处理。通常使用 l*_rx_out2clk 这个时钟来产生用户逻辑所需要的时钟。以下分别介绍使用 BUFG，LCLK 和 PLL 的接法。

### 4.15.3.1. 使用 BUFG 产生用户时钟

单个通道使用 BUFG 产生用户时钟接法如图 4-20 所示。在 SERDES 内部 RX CDR VCO 经过分频可以得到 l*_rx_recclk，该时钟可以通过 l*_rx_outclk 输出到 SERDES 外部，也可以经过分频经由 l*_rx_out2clk 输出到 SERDES 外部。通常使用 l*_rx_out2clk 输出送给 BUFG 的时钟输入端口，BUFG 输出时钟驱动 RX 用户逻辑，同时送回给 l*_rx_usr2clk 用于将 SERDES 内部的并行数据输出到用户逻辑

进行处理。此时 l*_rx_outclk 可以悬空，l*_rx_usrclk 可以接地，这两个时钟未使用。

若用户逻辑有需要，l*_rx_outclk 也可以输出给时钟 BUFFER，即 SERDES 可以多提供一路输出时钟给用户逻辑使用。若用户逻辑不需要，将其悬空即可。

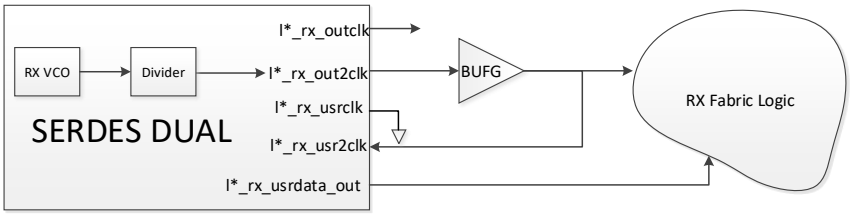


图 4-20 单个 Channel 用 BUFG 驱动 l*_rx_usr2clk 和用户逻辑的时钟方案

若使用 Multi channel 的协议，在 RX 各条 Lane 为同源时钟驱动且对端器件（Link Partner）的 TX 各条 Lane 也为同源时钟驱动的情况下，可以使用一个 RX Lane 的 l*_rx_out2clk 驱动多个 Lane 的 l*_rx_usr2clk。N 条 lane 的 Multi channel link 中，使用 BUFG 驱动用户逻辑的时钟接法如图 4-21 所示。

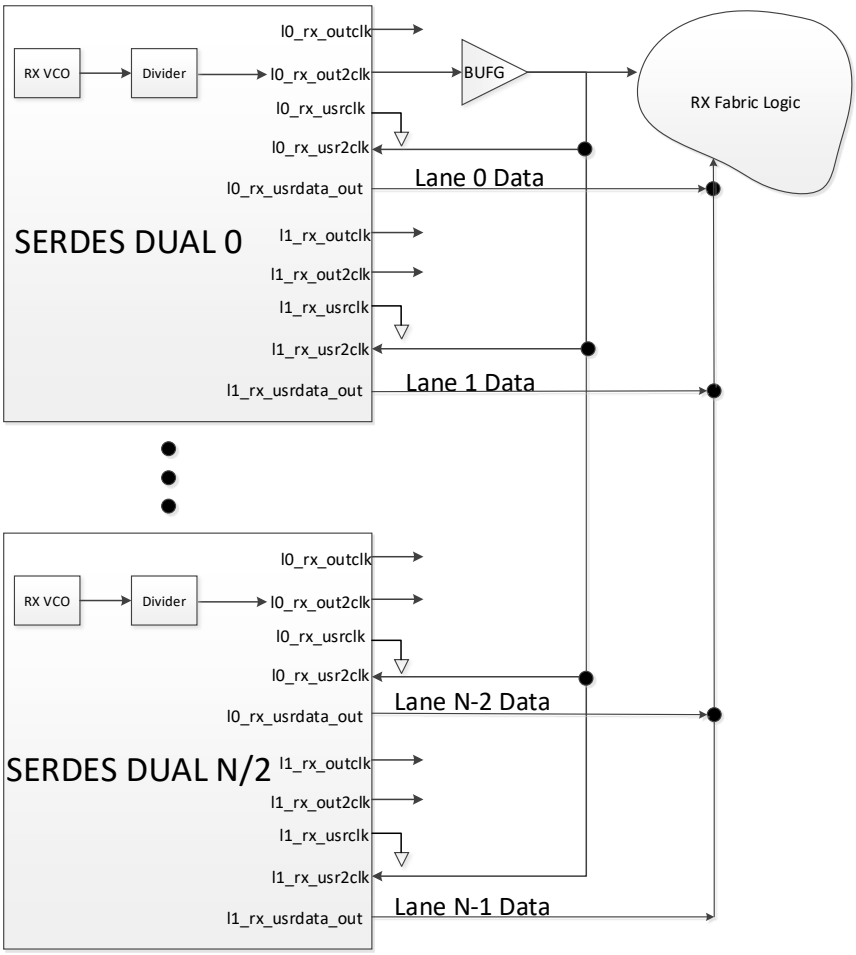


图 4-21 Multi Channel xN 模式用 BUFG 驱动 l*_rx_usr2clk 和用户逻辑的时钟方案

图 4-21 中，l0_rx_out2clk 作为时钟源驱动各个 Lane 的 l0_rx_usr2clk 和 l1_rx_usr2clk，也可以使用 l1_rx_out2clk 作为时钟源驱动 l0_rx_usr2clk 和 l1_rx_usr2clk。

#### 4.15.3.2. 使用 LCLK 产生用户时钟

单个通道使用 LCLK 产生用户时钟接法如图 4-22 所示。在 SERDES 内部 RX CDR VCO 经过分频可以得到 I*_rx_recclk，该时钟可以通过 I*_rx_outclk 输出到 SERDES 外部，也可以再分频后经由 I*_rx_out2clk 输出到 SERDES 外部。通常使用 I*_rx_out2clk 输出送给 LCLK 的时钟输入端口 clkin。LCLK 输出时钟有两路，一路是 clkout，一路是 clkdivout。clkout 走区域时钟走线，而 clkdivout 通过 BUFG 走全局时钟走线。图中使用 clkout 端口（不经过 BUFG）驱动 RX 用户逻辑，同时送回给 I*_rx_usr2clk 用于将 SERDES 内部的并行数据输出到用户逻辑进行处理。此时 I*_rx_outclk 可以悬空，I*_rx_usrclk 可以接地，这两个时钟未使用。

若用户逻辑有需要，I*_rx_outclk 也可以输出给 LCLK 或其他时钟 BUFFER，即 SERDES 可以多提供一路输出时钟给用户逻辑使用。若用户逻辑不需要，将其悬空即可。

LCLK 和 BUFG 的区别是，LCLK 的 clkout 输出时钟用于驱动本时钟域的用户逻辑。若使用 LCLK 驱动时钟域之外的用户逻辑，会有较大的 clock skew，用户设计应避免使用 clkout 输出的时钟驱动本时钟域之外的用户逻辑。LCLK 和 BUFG 相比的另一个区别是 LCLK 具有分频功能，它可以提供 1-8 等几种分频，而不用依靠 SERDES 内部分频功能。LCLK 的 clkout 和 clkdivout 都具有分频功能，具体用法见 UG910。

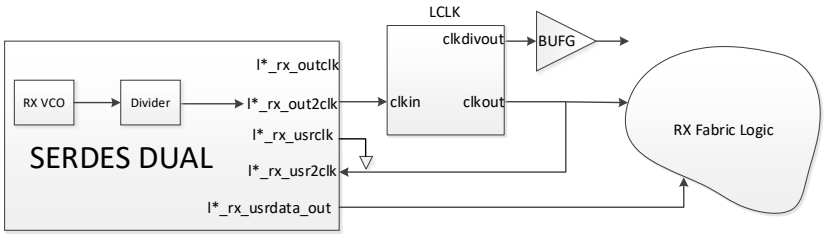


图 4-22 单个 Channel 用 LCLK 驱动 I*_rx_usr2clk 和用户逻辑的时钟方案

若使用 Multi channel 的协议，在 RX 各条 Lane 为同源时钟驱动且对端器件（Link Partner）的 TX 各条 Lane 也为同源时钟驱动的情况下，可以使用一个 Lane 的 I*_rx_out2clk 经过 LCLK 驱动多个 Lane 的 I*_rx_usr2clk。在一个时钟区域内，共有 4 个 LCLK 资源，通常 2 个 SERDES DUAL 分布在一个时钟区域内，这两个 SERDES DUAL 共用这 4 个 LCLK 资源。因此使用 LCLK 最多仅能驱动 2 个位于同一个时钟区域内的 SERDES DUAL，最多 4 条 Lane。使用 LCLK 驱动用户逻辑的时钟接法如图 4-23 所示。

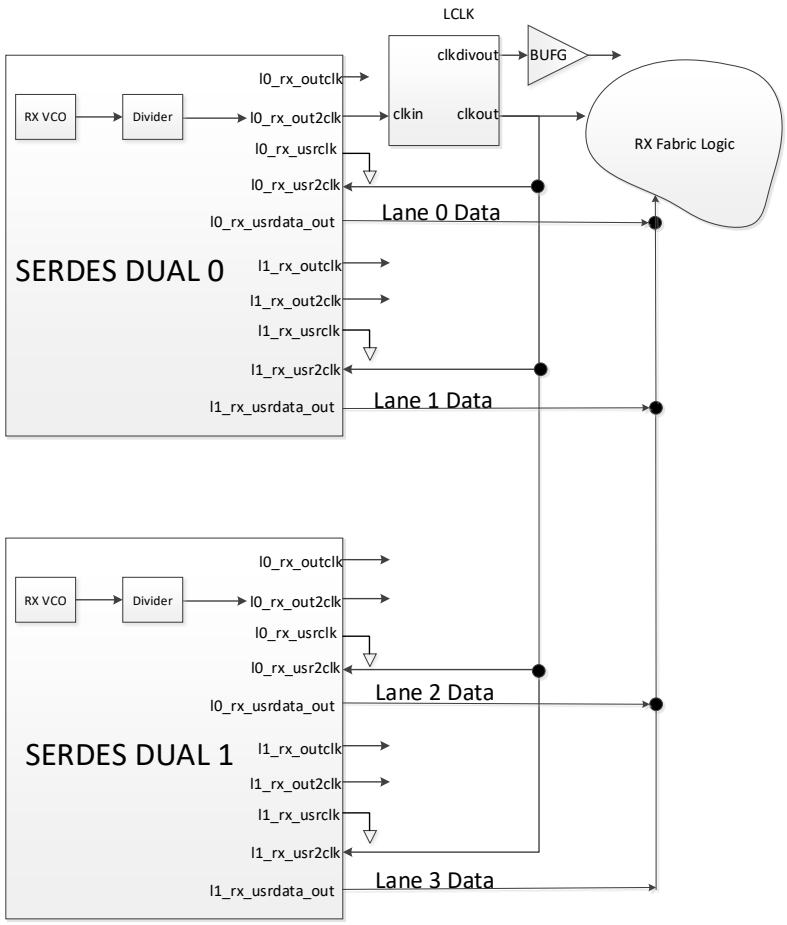


图 4-23 Multi Channel x4 模式用 LCLK 驱动 I*_rx_usr2clk 和用户逻辑的时钟方案

图 4-23 中，I0_rx_out2clk 作为时钟源驱动各个 Lane 的 I0_rx_usr2clk 和 I1_rx_usr2clk，也可以使用 I1_rx_out2clk 作为时钟源驱动 I0_rx_usr2clk 和 I1_rx_usr2clk。

LCLK 主要用在用户逻辑比较少，一个时钟区域足以容纳的情形；另外使用 LCLK 驱动 RX 用户逻辑的方案还可以用于 RX PCS FIFO Bypass 的场景，因为 LCLK 输入和输出时钟走线都很短，可以认为输入和输出时钟相位差很小,因此 SERDES 内部时钟和外部时钟相位很接近。

#### 4.15.3.3. 使用 PLL 产生用户时钟

单个通道使用 PLL 产生用户时钟接法如图 4-24 所示。在 SERDES 内部 RX CDR VCO 经过分频可以得到 I*_rx_recclk，该时钟可以通过 I*_rx_outclk 输出到 SERDES 外部，也可以再分频后经由 I*_rx_out2clk 输出到 SERDES 外部。I*_rx_outclk 和 I*_rx_out2clk 这两个时钟，其中任何一个都可以送给 PLL 的 refclk，PLL 输出时钟 clkc 驱动 BUFG 产生用户逻辑所需时钟，同时送回该时钟给 I*_rx_usr2clk 作为 SERDES RX 并行数据的输出时钟，将并行数据送到用户逻辑进行处理。若使用 I*_rx_out2clk，那么 I*_rx_outclk 可以悬空；若使用 I*_rx_outclk，那么 I*_rx_out2clk 可以悬空。I*_rx_usrclk 可以接地，改时钟未使用。

若用户逻辑有需要，悬空的 I*_rx_outclk 或 I*_rx_out2clk 也可以输出给时钟 BUFFER，即 SERDES 可以多提供一路输出时钟给用户逻辑使用。若用户逻辑不需要时钟这个时钟，将其悬空即可。

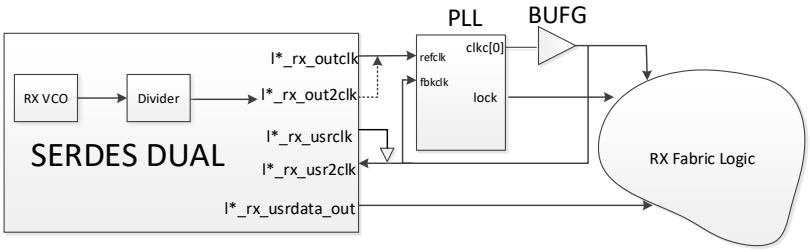


图 4-24 单个 Channel 用 PLL 再驱动 l*_rx_usr2clk 和用户逻辑的时钟方案

图中 l0_rx_out2clk 处虚线表示 l0_rx_outclk 和 l0_rx_out2clk 都可以驱动 PLL 的 refclk 输入端口。

在用户的设计中，多个 SERDES Channel 作为一个高速协议连接来用，如何用其中一个 Channel 的 l*_rx_outclk 或者 l*_rx_out2clk，经由 PLL 来驱动多个 Channel 的 l*_rx_usrclk 或 l*_rx_usr2clk，请见图 4-25 所示。

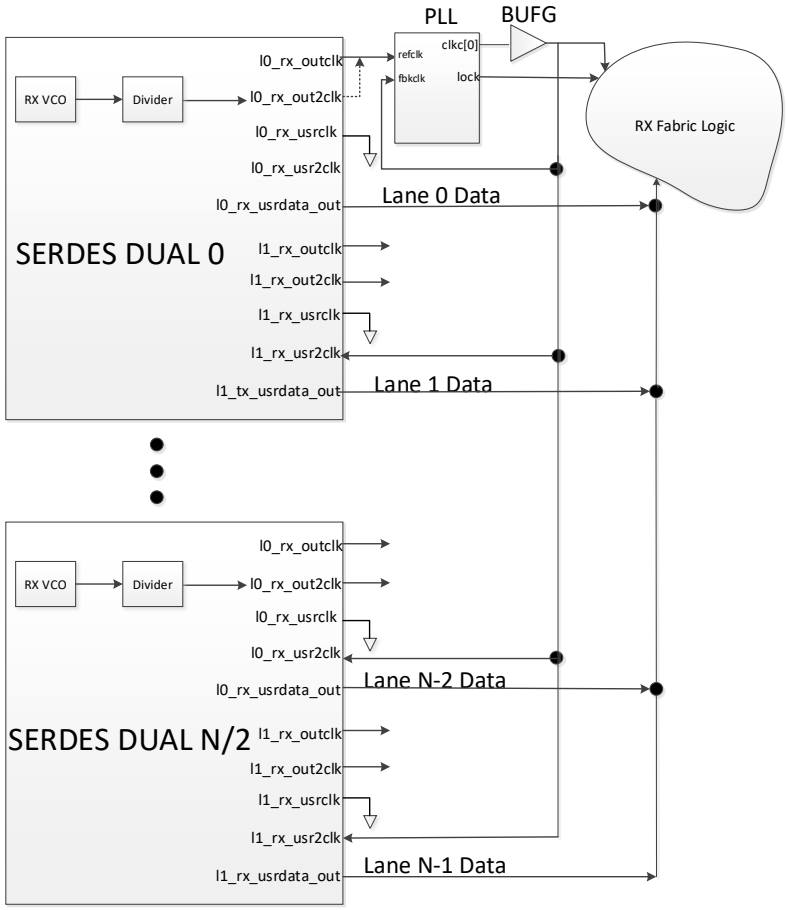


图 4-25 Multi Channel 协议使用 PLL 驱动用户逻辑和 l*_tx_usr2clk 的时钟方案

图 4-25 中，l0_rx_out2clk 作为时钟源驱动各个 Lane 的 l0_rx_usr2clk 和 l1_rx_usr2clk，也可以使用 l1_rx_out2clk 作为时钟源驱动 l0_rx_usr2clk 和 l1_rx_usr2clk。图中 l0_l*_rx_out2clk 处虚线表示 l0_rx_outclk 和 l0_rx_out2clk 都可以驱动 PLL 的 refclk 输入端口。

# 4.16 RX FABRIC 时钟

## 4.16.1 功能简介

接收方向 Fabric 时钟产生过程如图 4-26 所示。

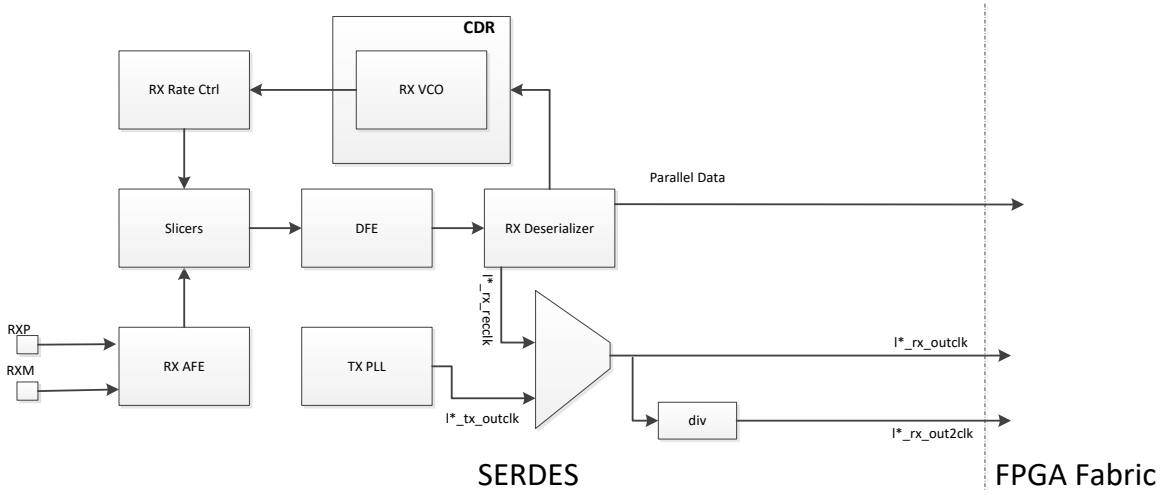


图 4-26 接收时钟结构

如图 4-27 所示，在接收方向，SERDES 可以输出 I*_rx_outclk 和 I*_rx_out2clk 这两个时钟，这两个时钟都可以进入全局时钟 BUFFER 或区域时钟 BUFFER 来驱动用户逻辑。驱动用户逻辑的时钟送回给 SERDES 的 I*_rx_usrclk 或 I*_rx_usr2clk 时钟输入端口，该时钟用于将并行数据(I*_rx_usrdata_in，I*_rx_usrdatak_in)的输出到 FPGA Fabric 用户逻辑里。下文描述 I*_rx_outclk 和 I*_rx_out2clk 产生的机理。

接收差分管脚 RXP/M 接收到的串行数据送入 AFE，Slicers 和 DFE 到达 RX 解串器 Deserializer，最终到达 CDR，CDR 用内部电路恢复出串行高速时钟，该时钟和输入数据的频率和相位都一致。这个时钟经过 RX Rate 控制器分频得到最终的串行时钟，用于 Slicers 对接收数据进行判断。解析出的数据经过 Deserializer 变成并行数据（Parallel data）送入 FPGA Fabric，恢复出的高速串行时钟在 Deserializer 里面变成并行恢复时钟 I*_rx_recclk。

恢复时钟 I*_rx_recclk 的频率计算公式如下：

$$I*\_\ rx\_recclk\_freq = \frac{I*\_rx\_linerate}{I*\_rx\_pma\_datawidth}$$

举例：当线速率为 2.5Gbps，内部数据位宽为 20bit 时，I*_rx_recclk 频率为 2.5Gbps/20=125MHz。

I*_rx_recclk 时钟和发送方向的 I*_tx_outclk 经过时钟 MUX 选择得到 I*_rx_outclk，选择的控制端口是 I*_rx_outclk_sel。I*_rx_outclk 时钟可以送到 FPGA 的时钟 BUFFER（BUFG 或 LCLK）或者 PLL 用于产生用户时钟（I*_rx_usrclk 和 I*_rx_usr2clk）。I*_rx_outclk 频率一般和 I*_rx_recclk 或 I*_tx_outclk 频率相等；I*_rx_recclk 经过 2 分频或者 4 分频得到 I*_rx_out2clk_int，该时钟可直接经由 I*_rx_out2clk 送出 SERDES。I*_rx_out2clk 的分频比由 I*_rx_out2clk_sel 信号决定。

## 4.16.2 端口和属性列表

RX Fabric 时钟相关端口及属性分别如表 4-28、表 4-29 所示。

表 4-28 RX Fabric 时钟相关端口

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| l*_rx_outclk | Out | 时钟 | 该时钟由时钟产生模块产生，它由 PMA 的输出的时钟 l*_rx_recclk 或 l*_tx_outclk 驱动。该时钟能驱动专用或者全局时钟网络，产生用户时钟 l*_rx_usrclk。 |
| l*_rx_out2clk | Out | 时钟 | 该时钟由时钟产生模块产生，为 l*_rx_recclk 分频时钟。 |
| l*_rx_usrclk | In | 时钟 | 该时钟由用户产生，为 rx 方向用户侧时钟，频率等于线速率除以 PCS/PMA 位宽。l*_rx_outclk 可以用来驱动专用或者全局时钟网络，产生该时钟。 |
| l*_rx_usr2clk | In | 时钟 | 该时钟由用户产生，为 rx 方向用户侧时钟，频率等于线速率除以 PCS/用户侧位宽。l*_tx_outclk 或 l*_rx_outclk 可以用来驱动专用或者全局时钟网络，产生该时钟。 |
| l*_rx_outclk_sel[1:0] | In | Async | l*_rx_outclk 的时钟源选择信号。推荐使用 SERDES IP GUI 产生的默认值。按位编码映射关系如下：<br>• 2'b00：static 0；<br>• 2'b01：l*_rx_recclk （cdr 恢复时钟）；<br>• 2’b10：l*_tx_outclk；<br>• 2’b11：static 0。 |
| l*_rx_out2clk_sel[1:0] | In | Async | l*_rx_out2clk 的时钟源选择信号。推荐使用 SERDES IP GUI 产生的默认值。按位编码映射关系如下：<br>• 2’b00：l*_rx_recclk 时钟 2 分频；<br>• 2’b01：l*_rx_reclk 时钟 4 分频；<br>• 2’b10：rx*_div16p5_clk 时钟 2 分频；<br>• 2’b11：l*_rx_recclk 。 |

| 信号名 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| l*_rx_align_reset | In | Async | 保留，接 0。 |
| l*_rx_align_start | In | Async | 保留，接 0。 |
| l*_rx_align_code_in[8:0] | In | Async | 保留，接全 0。 |
| l*_rx_align_code_out[8:0] | Out | Async | / |
| l*_rx_ph_aligned | Out | Async | / |
| l*_rx_align_fail | Out | Async | / |

表 4-29 RX Fabric 时钟相关属性

| 属性名 | 宽度 | 说明 |
|---|---|---|
| L*_RX_ALIGN_EN | 1 | 保留，接 0。 |
| L*_RX_EXT_MODE | 1 | 保留，接 0。 |
| L*_RX_WINDOW_SET | 5 | 保留，接全 0。 |
| L*_RX_DIS_GSR | 1 | 保留，接 0。 |

# 5 附录

## 5.1 CRI 访问寄存器列表（1）

表 5-1 CRI 寄存器列表

| CRI 寄存器地址（Hex） | CRI 数据位 | 参数名称 | 读/写 | 参数说明 |
|---|---|---|---|---|
| 0x12 | 15:10 | reserved | R | 保留。 |
| | 9 | TX_VBOOST_LVL_EN | R/W | Override enable for TX_VBOOST_LVL [2:0]. |
| | 8:6 | TX_VBOOST_LVL | R/W | Override value for TX_VBOOST_LVL [2:0]. |
| | 5 | RX_VREF_CTRL_EN | R/W | Override enable for RX_VREF_CTRL [4:0]. |
| | 4:0 | RX_VREF_CTRL | R/W | Override value for RX_VREF_CTRL [4:0]. |
| 0x1N03(2) | 13 | POST_OVRD_EN | R/W | Override enable for tx*_eq_post input. |
| | 12:7 | TX_POST_CURSOR | R/W | Override value for tx*_eq_post. |
| | 6 | PRE_OVRD_EN | R/W | Override enable for tx*_eq_pre input. |
| | 5:0 | TX_PRE_CURSOR | R/W | Override value for tx*_eq_pre. |
| 0x1N05 | 13 | EN（0x1N05 地址中的控制信号使能） | R/W | Override enable for the following inputs:<br>• rx*_dfe_bypass ;<br>• rx*_div16p5_clk_en ;<br>• l*_rx_pmadata_width ;<br>• rx*_rate ;<br>• rx*_pstate ;<br>• rx*_lpd ;<br>• rx*_req ;<br>• rx*_data_en ;<br>• rx*_invert ;<br>• rx*_reset. |
| | 12 | RX*_DFE_BYPASS | R/W | Override value for rx*_dfe_bypass. |
| | 11 | DIV16P5_CLK_EN | R/W | Override value for rx*_div16p5_clk_en. |
| | 10:9 | WIDTH | R/W | Override value for l*_rx_pmadata_width. |
| | 8:7 | RATE | R/W | Override value for rx*_rate. |

| CRI 寄存器地址（Hex） | CRI 数据位 | 参数名称 | 读/写 | 参数说明 |
|---|---|---|---|---|
| | 6:5 | PSTATE | R/W | Override value for rx*_pstate. |
| | 4 | LPD | R/W | Override value for rx*_lpd. |
| | 3 | REQ | R/W | Override value for rx*_req. |
| | 2 | DATA_EN | R/W | Override value for rx*_data_en. |
| | 1 | INVERT | R/W | Override value for rx*_invert. |
| | 0 | RESET | R/W | Override value for rx*_reset. |
| 0x1N01 | 15 | EN | R/W | Override enable for the following inputs:<br>• tx*_detrx_req ;<br>• tx*_mpllb_sel ;<br>• l*_tx_pmadata_width ;<br>• tx*_rate[2:0] ;<br>• tx*_pstate[1:0] ;<br>• tx*_lpd ;<br>• tx*_req ;<br>• tx*_data_en ;<br>• tx*_invert ;<br>• tx*_reset ;<br>• tx*_clk_rdy. |
| | 14 | DETECT_RX*_REQ | R/W | Override value for tx*_detrx_req. |
| | 13 | MPLLB_SEL | R/W | Override value for tx*_mpllb_sel. |
| | 12:11 | WIDTH | R/W | Override value for l*_tx_pmadata_width. |
| | 10:8 | RATE | R/W | Override value for tx*_rate[2:0]. |
| | 7:6 | PSTATE | R/W | Override value for tx*_pstate[1:0]. |
| | 5 | LPD | R/W | Override value for tx*_lpd. |
| | 4 | REQ | R/W | Override value for tx*_req. |
| | 3 | DATA_EN | R/W | Override value for tx*_data_en. |
| | 2 | INVERT | R/W | Override value for tx*_invert. |
| | 1 | RESET | R/W | Override value for tx*_reset. |
| | 0 | CLK_RDY | R/W | Override value for tx*_clk_rdy. |
| 0x1N02 | 15 | MAIN_OVRD_EN | R/W | Override enable for tx*_eq_main input. |

| CRI 寄存器地址（Hex） | CRI 数据位 | 参数名称 | 读/写 | 参数说明 |
|---|---|---|---|---|
| | 14:9 | TX_MAIN_CURSOR | R/W | Override value for tx*_eq_main. |
| | 8 | EN | R/W | Override enable for the following inputs:<br>• TX*_VBOOST_EN;<br>• TX*_IBOOST_LVL ;<br>• tx*_beacon_en ;<br>• tx*_disable. |
| | 7 | VBOOST_EN | R/W | Override value for TX*_VBOOST_EN. |
| | 6:3 | IBOOST_LVL | R/W | Override value for TX*_IBOOST_LVL . |
| | 2 | BEACON_EN | R/W | Override value for tx*_beacon_en. |
| | 1 | DISABLE | R/W | Override value for tx*_disable. |
| | 0 | NYQUIST_DATA | R/W | 保留，设置为 0。 |
| 0x1N09 | 15:14 | Reserved | R | 保留。 |
| | 13:9 | EQ_CTLE_BOOST | R/W | Override value for rx*_eq_ctle_boost. Its override enable is RX_OVRD_EQ_IN_1.EQ_OVRD_EN |
| | 8:6 | EQ_VGA2_GAIN | R/W | Override value for rx*_eq_vga2_gain. Its override enable is 0x1N0a.EQ_OVRD_EN. |
| | 5:3 | EQ_VGA1_GAIN | R/W | Override value for rx*_eq_vga1_gain. Its override enable is 0x1N0a.EQ_OVRD_EN. |
| | 2:0 | EQ_ATT_LVL | R/W | Override value for RX*_EQ_ATT_LVL. Its override enable is 0x1N0a.EQ_OVRD_EN. |
| 0x1N0a | 15:11 | Reserved | R | 保留。 |
| | 10 | EQ_OVRD_EN | R/W | Override enable for RX EQ inputs Input signals overridden by this enable are:<br>• rx*_eq_dfe_tap;<br>• RX*_EQ_CTLE_POLE;<br>• rx*_eq_ctle_boost;<br>• rx*_eq_vga2_gain;<br>• rx*_eq_vga1_gain;<br>• RX*_EQ_ATT_LVL. |
| | 9:2 | EQ_DFE_TAP1 | R/W | Override value for rx*_eq_dfe_tap. |

| CRI 寄存器地址（Hex） | CRI 数据位 | 参数名称 | 读/写 | 参数说明 |
|---|---|---|---|---|
| | 1:0 | EQ_CTLE_POLE | R/W | Override value for rx*_eq_ctle_pole. |
| 0x1N14 | 15:10 | Reserved | R | 保留。 |
| | 9:2 | EQ_DFE_TAP1 | R | 回读 FPGA 设置的 rx*_eq_dfe_tap 值。 |
| | 1:0 | EQ_CTLE_POLE | R | 回读 FPGA 设置的 rx*_eq_ctle_pole 值。 |
| 0x1NC1 | 1 | l*_loopback_en_reg | R/W | If bit 0 (ovrd_tx_loopback) = 1, this bit enables/disables TX loopback path to RX 1: Enables TX loopback path to RX 0: Disables TX loopback path to RX. |
| | 0 | ovrd_tx_loopback | R/W | If this bit = 1, it allows to turn on/off TX loopback mode using analog register bit 1 (l*_loopback_en_reg) 1: Allows TX loopback mode to be controlled by analog register bit 1 (l*_loopback_en_reg) 0: Disallows TX loopback mode to be controlled by analog register bit 1 (l*_loopback_en_reg). |
| 0x3008 | 8 | CONT_OVRD_EN | R/W | Override enable for the following inputs: RX*_ADAPT_CONT RX*_OFFCAN_CONT. |
| | 7 | OFFCAN_CONT | R/W | Override value for RX*_OFFCAN_CONT. Its override enable is: 0x3008.CONT_OVRD_EN. |
| | 6 | ADAPT_CONT | R/W | Override value for RX*_ADAPT_CONT. Its override enable is: 0x3008.CONT_OVRD_EN. |
| | 5 | ADAPT_REQ_OVRD_EN | R/W | Enable override values for rx0_adapt_req. |
| | 4 | ADAPT_REQ | R/W | Override value for rx0_adapt_req. Its override enable is: 0x3008.ADAPT_REQ_OVRD_EN. |
| | 3 | RX*_LOS_THRSHLD_OVRD_EN | R/W | Enable override for RX_LOS_THRESHOLD |
| | 2:0 | RX*_LOS_THRSHLD_OVRD_VAL | R/W | Override value for RX_LOS_THRESHOLD. Its override enable is: 0x3008.RX*_LOS_THRSHLD_OVRD_EN. |

| CRI 寄存器地址（Hex） | CRI 数据位 | 参数名称 | 读/写 | 参数说明 |
|---|---|---|---|---|
| 0x3019 | 14 | ADAPT_REQ_OVRD_EN | R/W | Enable override values for rx1_adapt_req . |
| | 13 | ADAPT_REQ | R/W | Override value for rx1_adapt_req. Its override enable is: 0x3019.ADAPT_REQ_OVRD_EN. |

注：

1.每个 CRI 寄存器地址对应的数据为 16bit 数据位，对于未描述的数据位，用户不能修改其值，操作 CRI 的时候应先读回，改变需要的 bit 位，保持其他 bit 位的值，再写回 CRI 寄存器。

2.CRI 寄存器地址中 N 代表 lane number，lane0 对应 N 值为 0，lane1 对应 N 值为 1。

- 调整发送摆幅示例 1——VBOOST 使能模式（TX_VBOOST_EN=1）调整 Lane0 的发送摆幅：在该模式下，需调整 TX_IBOOST_LVL 来细调摆幅。仅需要写 0x1002 寄存器。Bit8 设置为 1，激活 TX_VBOOST_EN,TX*_IBOOST_LVL,TX*_BEACON_EN,TX*_DISABLE 4 个参数改写模式;0x1002 Bit7 设置为 1，将 TX*_VBOOST_EN 置成 1，0x1002 Bit 6:3 设置为想要的 IBOOST_LVL 值（F 对应最高摆幅，其他值摆幅减小）。该寄存器其他 bit 均设置为 0。
- 调整发送摆幅过程示例 2——VBOOST 不使能（TX*_VBOOST_EN =0）调整 Lane1 的发送摆幅：控制 0x1102 寄存器。Bit8 设置为 1,TX_VBOOST_EN,TX8_IBOOST_LVL,TX*_BEACON_EN,TX*_DISABLE 4 个参数 Override 模式;0x1102 Bit7 设置为 0,将 TX*_VBOOST_EN 置成 0;通过调整 MAIN CURSOR 来调整发送摆幅。0x1102 Bit15 设置为 1，使能 MAIN CURSOR 参数改写模式；0x1102 Bit14:9 设置为想要的 MAIN CURSOR 值。寄存器 0x1102 其余 Bit 写为 0。还可以配合调整 POST CURSOR 和 PRE CURSOR 参数。这两个参数在寄存器 0x1103 中。先将 Bit13 设为 1，激活 TX_POST_CURSOR 的改写模式，Bit12:7 写入想设置的 TX_POST_CURSOR；将 Bit6 写为 1，激活 TX_PRE_CURSOR 的改写模式，Bit5:0 写入想要的 TX_PRE_CURSOR 的值，该寄存器其他 Bit 写 0。

## 5.2 SERDES 原语例化位置坐标和管脚对应关系

表 5-2、表 5-3、表 5-4、表 5-5、表 5-6 和表 5-7 分别为 PH1A400SFG900、PH1A400SFG676、PH1A90SBG484、PH1A90SEG324、PH1A90SEG325 和 PH1A180SFG676 器件的 SERDES 相关的管脚列表。

表 5-2 PH1A400SFG900 的 SERDES 管脚分配

| Bank 编号 | SERDES DUAL 原语位置坐标 | 管脚号 | 管脚名称 | 说明 |
|---|---|---|---|---|
| BANK80 | x133y120z0 | R8 | REFCLKP_80 | 参考时钟 P 端 |

| Bank 编号 | SERDES DUAL 原语<br>位置坐标 | 管脚号 | 管脚名称 | 说明 |
|---|---|---|---|---|
| | | R7 | REFCLKM_80 | 参考时钟 N 端 |
| | | Y2 | TXP0_80 | Lane0 发送 P 端 |
| | | Y1 | TXM0_80 | Lane0 发送 N 端 |
| | | AA4 | RXP0_80 | Lane0 接收 P 端 |
| | | AA3 | RXM0_80 | Lane0 接收 N 端 |
| | | V2 | TXP1_80 | Lane1 发送 P 端 |
| | | V1 | TXM1_80 | Lane1 发送 N 端 |
| | | Y6 | RXP1_80 | Lane1 接收 P 端 |
| | | Y5 | RXM1_80 | Lane1 接收 N 端 |
| | | V10 | RXRECCLKP_80 | 恢复时钟 P 端 |
| | | V9 | RXRECCLKN_80 | 恢复时钟 N 端 |
| BANK81 | x133y159z0 | U8 | REFCLKP_81 | 参考时钟 P 端 |
| | | U7 | REFCLKM_81 | 参考时钟 N 端 |
| | | U4 | TXP0_81 | Lane0 发送 P 端 |
| | | U3 | TXM0_81 | Lane0 发送 N 端 |
| | | W4 | RXP0_81 | Lane0 接收 P 端 |
| | | W3 | RXM0_81 | Lane0 接收 N 端 |
| | | T2 | TXP1_81 | Lane1 发送 P 端 |
| | | T1 | TXM1_81 | Lane1 发送 N 端 |
| | | V6 | RXP1_81 | Lane1 接收 P 端 |
| | | V5 | RXM1_81 | Lane1 接收 N 端 |
| BANK82 | x133y160z0 | L8 | REFCLKP_82 | 参考时钟 P 端 |
| | | L7 | REFCLKM_82 | 参考时钟 N 端 |
| | | P2 | TXP0_82 | Lane0 发送 P 端 |
| | | P1 | TXM0_82 | Lane0 发送 N 端 |
| | | T6 | RXP0_82 | Lane0 接收 P 端 |
| | | T5 | RXM0_82 | Lane0 接收 N 端 |
| | | N4 | TXP1_82 | Lane1 发送 P 端 |
| | | N3 | TXM1_82 | Lane1 发送 N 端 |
| | | R4 | RXP1_82 | Lane1 接收 P 端 |
| | | R3 | RXM1_82 | Lane1 接收 N 端 |

| Bank 编号 | SERDES DUAL 原语<br>位置坐标 | 管脚号 | 管脚名称 | 说明 |
|---|---|---|---|---|
| BANK83 | x133y199z0 | N8 | REFCLKP_83 | 参考时钟 P 端 |
| | | N7 | REFCLKM_83 | 参考时钟 N 端 |
| | | M2 | TXP0_83 | Lane0 发送 P 端 |
| | | M1 | TXM0_83 | Lane0 发送 N 端 |
| | | P6 | RXP0_83 | Lane0 接收 P 端 |
| | | P5 | RXM0_83 | Lane0 接收 N 端 |
| | | L4 | TXP1_83 | Lane1 发送 P 端 |
| | | L3 | TXM1_83 | Lane1 发送 N 端 |
| | | M6 | RXP1_83 | Lane1 接收 P 端 |
| | | M5 | RXM1_83 | Lane1 接收 N 端 |
| BANK84 | x133y200z0 | G8 | REFCLKP_84 | 参考时钟 P 端 |
| | | G7 | REFCLKM_84 | 参考时钟 N 端 |
| | | K2 | TXP0_84 | Lane0 发送 P 端 |
| | | K1 | TXM0_84 | Lane0 发送 N 端 |
| | | K6 | RXP0_84 | Lane0 接收 P 端 |
| | | K5 | RXM0_84 | Lane0 接收 N 端 |
| | | J4 | TXP1_84 | Lane1 发送 P 端 |
| | | J3 | TXM1_84 | Lane1 发送 N 端 |
| | | H6 | RXP1_84 | Lane1 接收 P 端 |
| | | H5 | RXM1_84 | Lane1 接收 N 端 |
| | | P10 | RXRECCLKP_84 | 恢复时钟 P 端 |
| | | P9 | RXRECCLKM_84 | 恢复时钟 N 端 |
| BANK85 | x133y239z0 | J8 | REFCLKP_85 | 参考时钟 P 端 |
| | | J7 | REFCLKM_85 | 参考时钟 N 端 |
| | | H2 | TXP0_85 | Lane0 发送 P 端 |
| | | H1 | TXM0_85 | Lane0 发送 N 端 |
| | | G4 | RXP0_85 | Lane0 接收 P 端 |
| | | G3 | RXM0_85 | Lane0 接收 N 端 |
| | | F2 | TXP1_85 | Lane1 发送 P 端 |
| | | F1 | TXM1_85 | Lane1 发送 N 端 |
| | | F6 | RXP1_85 | Lane1 接收 P 端 |

| Bank 编号 | SERDES DUAL 原语<br>位置坐标 | 管脚号 | 管脚名称 | 说明 |
|---|---|---|---|---|
| | | F5 | RXM1_85 | Lane1 接收 N 端 |
| BANK86 | x133y240z0 | C8 | REFCLKP_86 | 参考时钟 P 端 |
| | | C7 | REFCLKM_86 | 参考时钟 N 端 |
| | | D2 | TXP0_86 | Lane0 发送 P 端 |
| | | D1 | TXM0_86 | Lane0 发送 N 端 |
| | | E4 | RXP0_86 | Lane0 接收 P 端 |
| | | E3 | RXM0_86 | Lane0 接收 N 端 |
| | | C4 | TXP1_86 | Lane1 发送 P 端 |
| | | C3 | TXM1_86 | Lane1 发送 N 端 |
| | | D6 | RXP1_86 | Lane1 接收 P 端 |
| | | D5 | RXM1_86 | Lane1 接收 N 端 |
| BANK87 | x133y279z0 | E8 | REFCLKP_87 | 参考时钟 P 端 |
| | | E7 | REFCLKM_87 | 参考时钟 N 端 |
| | | B2 | TXP0_87 | Lane0 发送 P 端 |
| | | B1 | TXM0_87 | Lane0 发送 N 端 |
| | | B6 | RXP0_87 | Lane0 接收 P 端 |
| | | B5 | RXM0_87 | Lane0 接收 N 端 |
| | | A4 | TXP1_87 | Lane1 发送 P 端 |
| | | A3 | TXM1_87 | Lane1 发送 N 端 |
| | | A8 | RXP1_87 | Lane1 接收 P 端 |
| | | A7 | RXM1_87 | Lane1 接收 N 端 |

注*： PH1A400SFG900 只有 Bank 80 和 Bank 84 有专用恢复时钟输出管脚，其他 Bank 的恢复时钟输出可以选择普通 IO 输出。

表 5-3 PH1A400SFG676 的 SERDES 管脚分配

| Bank 编号 | SERDES DUAL 原语<br>位置坐标 | 管脚号 | 管脚名称 | 说明 |
|---|---|---|---|---|
| BANK80 | x133y120z0 | H6 | REFCLKP_80 | 参考时钟 P 端 |
| | | H5 | REFCLKM_80 | 参考时钟 N 端 |
| | | P2 | TXP0_80 | Lane0 发送 P 端 |
| | | P1 | TXM0_80 | Lane0 发送 N 端 |

| Bank 编号 | SERDES DUAL 原语位置坐标 | 管脚号 | 管脚名称 | 说明 |
|---|---|---|---|---|
| | | R4 | RXP0_80 | Lane0 接收 P 端 |
| | | R3 | RXM0_80 | Lane0 接收 N 端 |
| | | M2 | TXP1_80 | Lane1 发送 P 端 |
| | | M1 | TXM1_80 | Lane1 发送 N 端 |
| | | N4 | RXP1_80 | Lane1 接收 P 端 |
| | | N3 | RXM1_80 | Lane1 接收 N 端 |
| BANK81 | x133y159z0 | K6 | REFCLKP_81 | 参考时钟 P 端 |
| | | K5 | REFCLKM_81 | 参考时钟 N 端 |
| | | K2 | TXP0_81 | Lane0 发送 P 端 |
| | | K1 | TXM0_81 | Lane0 发送 N 端 |
| | | L4 | RXP0_81 | Lane0 接收 P 端 |
| | | L3 | RXM0_81 | Lane0 接收 N 端 |
| | | H2 | TXP1_81 | Lane1 发送 P 端 |
| | | H1 | TXM1_81 | Lane1 发送 N 端 |
| | | J4 | RXP1_81 | Lane1 接收 P 端 |
| | | J3 | RXM1_81 | Lane1 接收 N 端 |
| BANK82 | x133y160z0 | D6 | REFCLKP_82 | 参考时钟 P 端 |
| | | D5 | REFCLKM_82 | 参考时钟 N 端 |
| | | F2 | TXP0_82 | Lane0 发送 P 端 |
| | | F1 | TXM0_82 | Lane0 发送 N 端 |
| | | G4 | RXP0_82 | Lane0 接收 P 端 |
| | | G3 | RXM0_82 | Lane0 接收 N 端 |
| | | D2 | TXP1_82 | Lane1 发送 P 端 |
| | | D1 | TXM1_82 | Lane1 发送 N 端 |
| | | E4 | RXP1_82 | Lane1 接收 P 端 |
| | | E3 | RXM1_82 | Lane1 接收 N 端 |
| BANK83 | x133y199z0 | F6 | REFCLKP_83 | 参考时钟 P 端 |
| | | F5 | REFCLKM_83 | 参考时钟 N 端 |
| | | B2 | TXP0_83 | Lane0 发送 P 端 |
| | | B1 | TXM0_83 | Lane0 发送 N 端 |
| | | C4 | RXP0_83 | Lane0 接收 P 端 |
| | | C3 | RXM0_83 | Lane0 接收 N 端 |
| | | A4 | TXP1_83 | Lane1 发送 P 端 |
| | | A3 | TXM1_83 | Lane1 发送 N 端 |
| | | B6 | RXP1_83 | Lane1 接收 P 端 |

| Bank 编号 | SERDES DUAL 原语<br>位置坐标 | 管脚号 | 管脚名称 | 说明 |
|---|---|---|---|---|
| | | B5 | RXM1_83 | Lane1 接收 N 端 |

表 5-4 PH1A90SBG484 的 SERDES 管脚分配

| Bank 编号 | SERDES DUAL 原语<br>位置坐标 | 管脚号 | 管脚名称 | 说明 |
|---|---|---|---|---|
| BANK82 | X71y120z0 | F6 | REFCLKP_82 | 参考时钟 P 端 |
| | | F5 | REFCLKM_82 | 参考时钟 N 端 |
| | | F1 | TXP0_82 | Lane0 发送 P 端 |
| | | F2 | TXM0_82 | Lane0 发送 N 端 |
| | | G3 | RXP0_82 | Lane0 接收 P 端 |
| | | G4 | RXM0_82 | Lane0 接收 N 端 |
| | | D1 | TXP1_82 | Lane1 发送 P 端 |
| | | D2 | TXM1_82 | Lane1 发送 N 端 |
| | | E3 | RXP1_82 | Lane1 接收 P 端 |
| | | E4 | RXM1_82 | Lane1 接收 N 端 |
| BANK83 | X71y159z0 | D6 | REFCLKP_83 | 参考时钟 P 端 |
| | | D5 | REFCLKM_83 | 参考时钟 N 端 |
| | | A4 | TXP0_83 | Lane0 发送 P 端 |
| | | A3 | TXM0_83 | Lane0 发送 N 端 |
| | | B6 | RXP0_83 | Lane0 接收 P 端 |
| | | B5 | RXM0_83 | Lane0 接收 N 端 |
| | | B2 | TXP1_83 | Lane1 发送 P 端 |
| | | B1 | TXM1_83 | Lane1 发送 N 端 |
| | | C4 | RXP1_83 | Lane1 接收 P 端 |
| | | C3 | RXM1_83 | Lane1 接收 N 端 |

表 5-5 PH1A90SEG324 的 SERDES 管脚分配

| Bank 编号 | SERDES DUAL 原语<br>位置坐标 | 管脚号 | 管脚名称 | 说明 |
|---|---|---|---|---|
| BANK80 | X71y120z0 | H6 | REFCLKP_80 | 参考时钟 P 端 |
| | | H5 | REFCLKM_80 | 参考时钟 N 端 |
| | | M1 | TXP0_80 | Lane0 发送 P 端 |
| | | M2 | TXM0_80 | Lane0 发送 N 端 |
| | | L4 | RXP0_80 | Lane0 接收 P 端 |
| | | L3 | RXM0_80 | Lane0 接收 N 端 |

| Bank 编号 | SERDES DUAL 原语位置坐标 | 管脚号 | 管脚名称 | 说明 |
|---|---|---|---|---|
| | | K1 | TXP1_80 | Lane1 发送 P 端 |
| | | K2 | TXM1_80 | Lane1 发送 N 端 |
| | | K6 | RXP1_80 | Lane1 接收 P 端 |
| | | K5 | RXM1_80 | Lane1 接收 N 端 |
| BANK81 | X71y119z0 | F6 | REFCLKP_81 | 参考时钟 P 端 |
| | | F5 | REFCLKM_81 | 参考时钟 N 端 |
| | | F1 | TXP0_81 | Lane0 发送 P 端 |
| | | F2 | TXM0_81 | Lane0 发送 N 端 |
| | | G4 | RXP0_81 | Lane0 接收 P 端 |
| | | G3 | RXM0_81 | Lane0 接收 N 端 |
| | | H1 | TXP1_81 | Lane1 发送 P 端 |
| | | H2 | TXM1_81 | Lane1 发送 N 端 |
| | | J4 | RXP1_81 | Lane1 接收 P 端 |
| | | J3 | RXM1_81 | Lane1 接收 N 端 |
| BANK82 | X71y120z0 | E8 | REFCLKP_82 | 参考时钟 P 端 |
| | | E7 | REFCLKM_82 | 参考时钟 N 端 |
| | | D1 | TXP0_82 | Lane0 发送 P 端 |
| | | D2 | TXM0_82 | Lane0 发送 N 端 |
| | | E4 | RXP0_82 | Lane0 接收 P 端 |
| | | E3 | RXM0_82 | Lane0 接收 N 端 |
| | | B1 | TXP1_82 | Lane1 发送 P 端 |
| | | B2 | TXM1_82 | Lane1 发送 N 端 |
| | | C4 | RXP1_82 | Lane1 接收 P 端 |
| | | C3 | RXM1_82 | Lane1 接收 N 端 |
| BANK83 | X71y159z0 | D6 | REFCLKP_83 | 参考时钟 P 端 |
| | | D5 | REFCLKM_83 | 参考时钟 N 端 |
| | | A8 | TXP0_83 | Lane0 发送 P 端 |
| | | A7 | TXM0_83 | Lane0 发送 N 端 |
| | | C8 | RXP0_83 | Lane0 接收 P 端 |
| | | C7 | RXM0_83 | Lane0 接收 N 端 |
| | | A4 | TXP1_83 | Lane1 发送 P 端 |
| | | A3 | TXM1_83 | Lane1 发送 N 端 |
| | | B6 | RXP1_83 | Lane1 接收 P 端 |
| | | B5 | RXM1_83 | Lane1 接收 N 端 |

表 5-6 PH1A90SEG325 的 SERDES 管脚分配

| Bank 编号 | SERDES DUAL 原语位置坐标 | 管脚号 | 管脚名称 | 说明 |
|---|---|---|---|---|
| BANK82 | X71y120z0 | F6 | REFCLKP_82 | 参考时钟 P 端 |
| | | F5 | REFCLKM_82 | 参考时钟 N 端 |
| | | F1 | TXP0_82 | Lane0 发送 P 端 |
| | | F2 | TXM0_82 | Lane0 发送 N 端 |
| | | G4 | RXP0_82 | Lane0 接收 P 端 |
| | | G3 | RXM0_82 | Lane0 接收 N 端 |
| | | D1 | TXP1_82 | Lane1 发送 P 端 |
| | | D2 | TXM1_82 | Lane1 发送 N 端 |
| | | E4 | RXP1_82 | Lane1 接收 P 端 |
| | | E3 | RXM1_82 | Lane1 接收 N 端 |
| BANK83 | X71y159z0 | D6 | REFCLKP_83 | 参考时钟 P 端 |
| | | D5 | REFCLKM_83 | 参考时钟 N 端 |
| | | A3 | TXP0_83 | Lane0 发送 P 端 |
| | | A4 | TXM0_83 | Lane0 发送 N 端 |
| | | B6 | RXP0_83 | Lane0 接收 P 端 |
| | | B5 | RXM0_83 | Lane0 接收 N 端 |
| | | B1 | TXP1_83 | Lane1 发送 P 端 |
| | | B2 | TXM1_83 | Lane1 发送 N 端 |
| | | C4 | RXP1_83 | Lane1 接收 P 端 |
| | | C3 | RXM1_83 | Lane1 接收 N 端 |

表 5-7 PH1A180SFG676 的 SERDES 管脚分配

| Bank 编号 | SERDES DUAL 原语位置坐标 | 管脚号 | 管脚名称 | 说明 |
|---|---|---|---|---|
| BANK80 | x103y120z0 | H6 | REFCLKP_80 | 参考时钟 P 端 |
| | | H5 | REFCLKM_80 | 参考时钟 N 端 |
| | | P2 | TXP0_80 | Lane0 发送 P 端 |
| | | P1 | TXM0_80 | Lane0 发送 N 端 |
| | | R4 | RXP0_80 | Lane0 接收 P 端 |
| | | R3 | RXM0_80 | Lane0 接收 N 端 |
| | | M2 | TXP1_80 | Lane1 发送 P 端 |
| | | M1 | TXM1_80 | Lane1 发送 N 端 |
| | | N4 | RXP1_80 | Lane1 接收 P 端 |
| | | N3 | RXM1_80 | Lane1 接收 N 端 |

| Bank 编号 | SERDES DUAL 原语位置坐标 | 管脚号 | 管脚名称 | 说明 |
|---|---|---|---|---|
| BANK81 | x103y159z0 | K6 | REFCLKP_81 | 参考时钟 P 端 |
| | | K5 | REFCLKM_81 | 参考时钟 N 端 |
| | | K2 | TXP0_81 | Lane0 发送 P 端 |
| | | K1 | TXM0_81 | Lane0 发送 N 端 |
| | | L4 | RXP0_81 | Lane0 接收 P 端 |
| | | L3 | RXM0_81 | Lane0 接收 N 端 |
| | | H2 | TXP1_81 | Lane1 发送 P 端 |
| | | H1 | TXM1_81 | Lane1 发送 N 端 |
| | | J4 | RXP1_81 | Lane1 接收 P 端 |
| | | J3 | RXM1_81 | Lane1 接收 N 端 |
| BANK82 | x103y160z0 | D6 | REFCLKP_82 | 参考时钟 P 端 |
| | | D5 | REFCLKM_82 | 参考时钟 N 端 |
| | | F2 | TXP0_82 | Lane0 发送 P 端 |
| | | F1 | TXM0_82 | Lane0 发送 N 端 |
| | | G4 | RXP0_82 | Lane0 接收 P 端 |
| | | G3 | RXM0_82 | Lane0 接收 N 端 |
| | | D2 | TXP1_82 | Lane1 发送 P 端 |
| | | D1 | TXM1_82 | Lane1 发送 N 端 |
| | | E4 | RXP1_82 | Lane1 接收 P 端 |
| | | E3 | RXM1_82 | Lane1 接收 N 端 |
| BANK83 | x103y199z0 | F6 | REFCLKP_83 | 参考时钟 P 端 |
| | | F5 | REFCLKM_83 | 参考时钟 N 端 |
| | | B2 | TXP0_83 | Lane0 发送 P 端 |
| | | B1 | TXM0_83 | Lane0 发送 N 端 |
| | | C4 | RXP0_83 | Lane0 接收 P 端 |
| | | C3 | RXM0_83 | Lane0 接收 N 端 |
| | | A4 | TXP1_83 | Lane1 发送 P 端 |
| | | A3 | TXM1_83 | Lane1 发送 N 端 |
| | | B6 | RXP1_83 | Lane1 接收 P 端 |
| | | B5 | RXM1_83 | Lane1 接收 N 端 |

注*：PH1A400SFG676、PH1A90SBG484、PH1A90SEG324、PH1A90SEG325 和 PH1A180SFG676 无专用恢复时钟输出管脚，恢复时钟输出可以选择普通 IO 输出。

# 版本信息

| 日期 | 版本 | 修订记录 |
|---|---|---|
| 2021/06/19 | 0.1 | 首次发布中文版 |
| 2021/07/05 | 0.2 | 3.3.3 和 4.15.3 小节中 LCLK 增加 clkout 和 clkdivout 说明，图 3-8，图 3-9，图 4-23 和图 4-24 增加 LCLK 的 clkout 和 clkdivout 端口，之前没有指定具体端口；表 2-2 的 RXCLK_BUF_CS 参数默认值从 3’b010 改为 3’b011； |
| 2021/10/31 | 0.3 | 1. 2.2 节增加 mplla_multiplier[7]和 mpllb_multiplier[7]的描述，更新了表 2-4 中 mplla_multiplier 和 mpllb_multiplier 的描述，增加了表 2-6 mplla_multiplier[6:0]和 mpllb_multiplier[6:0]和实际反馈路径分频数 N 的关系，更新公式 2-1，更新图 2-5 增加 mplla_multiplier[7]和 mpllb_multiplier[7]所对应的分频器；2.2.1 节去掉 mplla<br>2. 和 mpllb 只能同时使能一个的限制，放宽为两条 Lane 可以选择不同的 PLL；增加了“专用恢复时钟输出管脚的用法”一节；表 3-24 增加 tx*_rate 为 3’b101 和 3’b110 的描述；2.2.3 节中增加了“PLL 单独复位流程”说明；4.5.3 增加“CDR 锁定本地参考”使用方法说明；4.14.3 中增加 RX Clock Aligner 不能支持 CDR 锁定本地参考模式的限制。 |
| 2022/01/20 | 1.0 | 首次发布正式版本 |
| 2022/06/28 | 1.1 | 1. 2.3.3 节增加“针对不同场景推荐使用的复位信号说明”；<br>2. 增加 2.4.2 章节，即环回相关的端口列表；<br>3. 2.4.3 节中“Far-End PMA Loopback 远端 PMA 环回”说明中，将进入和退出该环回模式需要对“rx*_reset_i 进行复位”修改为需要对“tx*_reset_i 进行复位”；<br>4. 修改表 4-16 中 l*_rx_show_realigncomma 的描述，修改该信号功能的描述；<br>5. 3.8.4 节增加了进入和退出 TX PRBS 模式需对 l*_tx_pcs_rst_n 进行复位的要求；<br>6. 4.9.3 节增加了进入和退出 RX PRBS 模式需对 l*_rx_pcs_rst_n 进行复位的要求；<br>7. 表 4-14 增加了 l*_rx_prbs_err_cnt[7:0]计数器计满以后的说明；<br>8. “PH1A400 系列”更改为“PH1A 系列”；<br>9. 增加 4.12.3 节，将 4.12.1 节中涉及到具体用法的内容移到 4.12.3 节，并在 4.12.3 节中增加了 RX Rate Match FIFO |

| 日期 | 版本 | 修订记录 |
| --- | --- | --- |
| | | 使用限制；<br>10. 增加 PH1A60 器件不包含 SERDES 的说明；<br>11. 增加图 1-2 PH1A400SFG676 器件 SERDES 资源结构图；<br>12. 增加表 5-3 PH1A400SFG676 的 SERDES 管脚分配 |
| 2022/10/01 | 1.2 | 1. 增加 SFG676 器件不支持专用管脚输出恢复时钟的说明；<br>2. 增加 2.1.3 节“PH1A400 参考时钟共用”SFG676 器件支持的 BANK 范围说明；<br>3. 更新 4.8.2 节，增加 L*_RX_POLARITYINV_EN 赋值为 1 的限制，在 PRBS Verifier 模块使能时禁用，保持赋值为 0；<br>4. 更新 2.3.3 节 SERDES 整体复位和初始化流程第 11 和第 12 条和复位 SERDES 某一条 Lane RX 部分流程第 4 和第 5 条描述，在 rx*_valid 拉高以后才能释放 l*_rx_pcs_rst_n；并同步更新图 2-9，增加图 2-10；<br>5. 增加 PH1A 系列器件说明；<br>6. 更新全文中端口及属性名称，使其与原语端口及属性名称保持一致；将 2.2.3 节关于通配符*说明删除移动至在 1.3.1 节，全文使用通配符表示 Lane 相关信号名称；<br>7. 修改 2.2.3 节 TX Fabric 时钟相关端口中 tx*_mpllb_sel 信号说明，从“tx0_mpllb_sel=1，表示 Lane0 使用 MPLLA 驱动”修改为“tx*_mpllb_sel=0，表示 Lane*使用 MPLLA 驱动”；<br>8. 表 2-7 复位信号列表中增加 phy_reset 使用说明；<br>9. 表 2-7 复位信号列表中增加“CDR 非跟踪模式”的描述；<br>10. 表 2-7 复位信号列表中修改 CDR 锁定信号、Rate Match FIFO 复位信号名称；<br>11. 更新 2.3.4 节复位完成时间说明；<br>12. 修改表 2-11 TX 短接相关参数相关端口为属性；<br>13. 修改 3.2.2 节表 3-1TX Fabric 时钟相关端口中 tx*_mpllb_sel 信号说明，删除“Channel0 或者 Channel1 必须设置相等的值”；<br>14. 全文更新删除 TX/RX Clock Aligner 相关说明，该模块禁用，端口保持赋值为 0；<br>15. 表 3-5 中删除“tx_pcsfifo_err”信号；<br>16. 增加 3.6.3 节，将 3.6.1 节中涉及到具体用法的内容移到 3.6.3 节；<br>17. 图 3-20 图注从“RX 配置更新过程”修改为“TX 配置更新过程”； |

| 日期 | 版本 | 修订记录 |
| --- | --- | --- |
| | | 18. 表 4-1、表 4-10、表 5-1 中信号“rx*_eq_dfe_tap1”修改为“rx*_eq_dfe_tap”；<br>19. 表 4-10 中信号“rx_eq_att_lvl”“rx_eq_ctle_pole”、“rx_cdr_vco_temp_ comp_en”拿出，放入表 4-11 中，并修改名称为“RX*_EQ_ATT_LVL”、“RX*_EQ_CTLE_POLE”、“RX*_CDR_VCO_TEMP_COMP_EN”，表 4-11 中"RX_MISC"拿出放入表 4-10 中，并修改为“rx*_misc”；<br>20. 将表 4-16 Word Aligner 端口列表中 RX 反序相关端口拿出修改为属性，增加表 4-17 Word Aligner 属性列表；<br>21. 增加 4.11.3 节，将 4.11.1 节中涉及到具体用法的内容移到 4.11.3 节；<br>22. 修改 4.11.2 节表 4-18“l*_rx_usrdisperr_out”、“l*_rx_usrcodeerr_out”信号描述，在 4.11.3 节增加使用注意事项；<br>23. 修改 4.13.2 节表 4-22 Byte De-serializer 端口列表 l*_rx_bytedeserial_mode 信号时钟域；<br>24. 表 4-23 中删除“rx_pcsfifo_err”信号；<br>25. 删除 3.2.1 节、4.16.1 节 TX Clock Aligner/RX Clock Aligner 相关说明，更新图 3-3、图 4-26，去除 TX Clock Aligner /RX Clock Aligner 模块；修改表 3-1、表 3-2TX Clock Aligner 及表 4-27、表 4-28 RX Clock Aligner 端口描述，修改为“保留，接 0”；删除 3.2.3 节、4.16.3 节用法说明，删除 3.4.3 节、4.14.3 节 TX PCS FIFO bypass 中 TX Clock Aligner 及 RX PCS FIFO bypass 中 RX Clock Aligner 相关描述；即更新全文，删除 TX Clock Aligner/RX Clock Aligner 模块相关说明，该模块禁用。 |
| 2022/10/31 | 1.3 | 1. 增加支持 PH1A90 器件，增加图 1-3 器件支持资源示意图；<br>2. 修改 1.1 节，增加器件支持的线速率上限说明；<br>3. 更新表 2-2，增加 PH1A90 特有属性 REF_CLK_UP_DOWN 属性说明；<br>4. 更新 2.1.3 节“PH1A 参考时钟共用”，更新图 2-4，增加 PH1A90 器件参考时钟共用说明；<br>5. 删除表 2-3，PLL 支持的线速率范围请参见《DS900_PH1A_Datasheet》；<br>6. 增加表 5-4，PH1A90SBG484 器件 SERDES 原语例化位置坐标和管脚对应关系； |

| 日期 | 版本 | 修订记录 |
| --- | --- | --- |
| | | 7. 更新文档免责声明。 |
| 2023/03/17 | 1.4 | 1. 增加支持 PH1A90SEG324、PH1A180SFG676 器件，全文更新，具体见第 1 节、2.1 节参考时钟、5.2 节 SERDES 原语例化位置坐标和管脚对应关系说明等；<br>2. 修改 1.1 节线速率支持；<br>3. 修改 I*_rx_align_sel 信号范围说明。 |
| 2023/08/7 | 1.5 | 1. 增加支持 PH1A90SEG325 器件，全文更新，具体见第 1 节、2.1 节参考时钟、5.2 节 SERDES 原语例化位置坐标和管脚对应关系说明等；<br>2. 更新 2.4.2 节近端 PMA 环回模式的使用说明。 |



# 免责声明

本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其他方式授予任何知识产权许可；本文档仅为向用户提供使用器件的参考，协助用户正确地使用安路科技产品之用，其著作权归安路科技所有；本文档所展示的任何产品信息均不构成安路科技对所涉产品或服务作出任何明示或默示的声明或保证。

安路科技将不定期地对本文档进行更新、修订。用户如需获取最新版本的文档，可通过安路科技的官方网站（网址为：https://www.anlogic.com）自行查询下载，也可联系安路科技的销售人员咨询获取。