

版本信息:

版本

REV2019

时间

01/01/2019

XILINX FPGA-光电通信篇

(基于 MA703-100T 开发板)

电子版自学资料

常州一二三电子科技有限公司

溧阳米联电子科技有限公司

版权所有

米联客学院 04QQ 群: 516869816

米联客学院 03QQ 群: 543731097(半满)

米联客学院 02QQ 群: 86730608(满)

米联客学院 01QQ 群: 34215299(满)

版本	时间	描述
Rev2020	2020-08-16	第一版 V1.0

感谢您使用米联客 ARTIX 系列开发板，以及配套教程。本教程不仅仅适合用于米联客开发板，而且可以用于其他的 ARTIX 开发板。

软件版本：Vitis2019.2

版权声明：

本手册版权归常州一二三电子科技有限公司/溧阳米联电子科技有限公司所有，并保留一切权利，未经我司书面授权，擅自摘录或者修改本手册部分或者全部内容，我司有权追究其法律责任。

版主大神们都等着大家去提问--电子资源论坛 www.uisrc.com

扫描以下二维码注册论坛：



微信公众平台：



目录

XILINX FPGA-光电通信篇.....	1
CH01 利用 IBERT 进行 GTP 信号眼图测试.....	5
1.1 概述.....	5
1.2 测试原理.....	5
1.3.1 千兆 1.25G 速率.....	5
1.3.2 千兆 6.25G 速率.....	8
1.4 使用 example design.....	10
1.5 GTP IBERT 测试.....	12
1.5.1 1.25G 测试.....	12
1.5.2 6.25G 测试.....	17
CH02 aurora_8b10b 光通信.....	21
2.1 概述.....	21
2.2 Setp By Step 搭建 FPGA 工程.....	21
2.2.1 7 Series FPGAs Transceivers Wizard 的配置.....	21
2.2.1 修改 example 工程.....	30
2.3 编译下载测试.....	32
CH03 XILINX 7 系列 GTP 通信之 HDMI 视频传输.....	34
3.1 概述.....	34
3.2 8B10B 编码原理.....	34
3.3 gt IP 核设置.....	37
3.4 修改 example 工程.....	46
3.5 封装自定义 IP.....	52
3.5.1 video_encode.v.....	52
3.5.2 video_decode.v.....	52
3.5.3 data_align.v.....	52
3.5.4 封装 IP.....	53
3.6 环路测试视频传输.....	53
3.6.1 硬件接线图.....	54
3.6.2 FPGA BD 工程.....	54
3.7 板到板间视频传输.....	56
3.7.1 硬件接线图.....	56
3.7.2 发送端 FPGA BD 工程.....	57
3.7.3 接收端 FPGA BD 工程.....	57
3.8 测试结果.....	59

CH01 利用 IBERT 进行 GTP 信号眼图测试

软件版本: VIVADO2019.2

操作系统: WIN10 64bit

硬件平台: 适用 XILINX A7/K7/Z7/ZU/KU 系列 FPGA

登录米联客(MSXBO)FPGA 社区-www.uisrc.com 观看免费视频课程、在线答疑解惑!

1.1 概述

Vivado 中提供了 1 种 IBERT 工具用于对 Xilinx FPGA 芯片的高速串行收发器进行板级硬件测试。通过 IBERT 我们可以获取误码率, 观察眼图, 调节串行收发器的参数, 从而有助于判断可能存在的问题, 便于验证硬件的稳定性和信号完整性。

本教程基于米联 ARTIX-7 系列开发板, 使用 IBERT 工具对与 SFP 连接的 GTP 进行 1.25Gbps 和 6.25Gbps 速率下的测试。从误码率和眼图两个角度来验证开发板 GTP 部分工作的稳定性和可靠性。

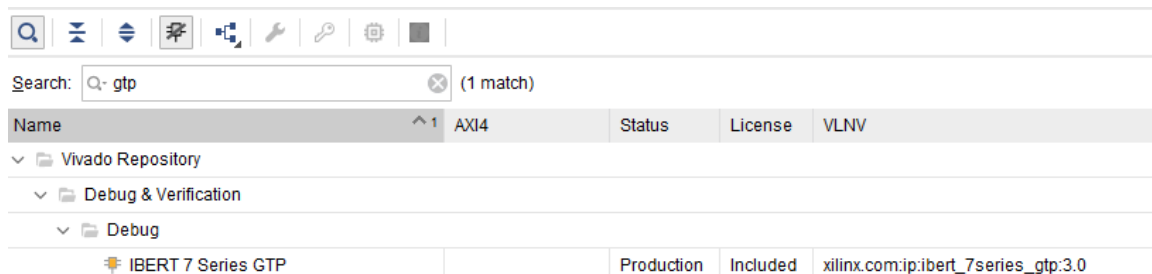
本教程共包含 ibert_1_25g 和 ibert_6_25g 两个例子, 分别对应 1.25Gbps 和 6.25Gbps 速率, 均基于 vivado 2017.4 版本开发。

1.2 测试原理

IBERT 中的 BERT 是 Bit Error Ratio Test 的缩写, 指比特出错概率测试, 简而言之就是误码率测试。Vivado 中 IBERT 工具的测试原理是通过收发器由外部回环进行自收自发而实现。就是将同一组收发器的 TX 和 RX 进行短接, TX 发送端通过发送某种特定序列的数据流, 在 RX 接收端接收后, 通过比对发送和接收的数据, 从而得出接收端误码的统计值。

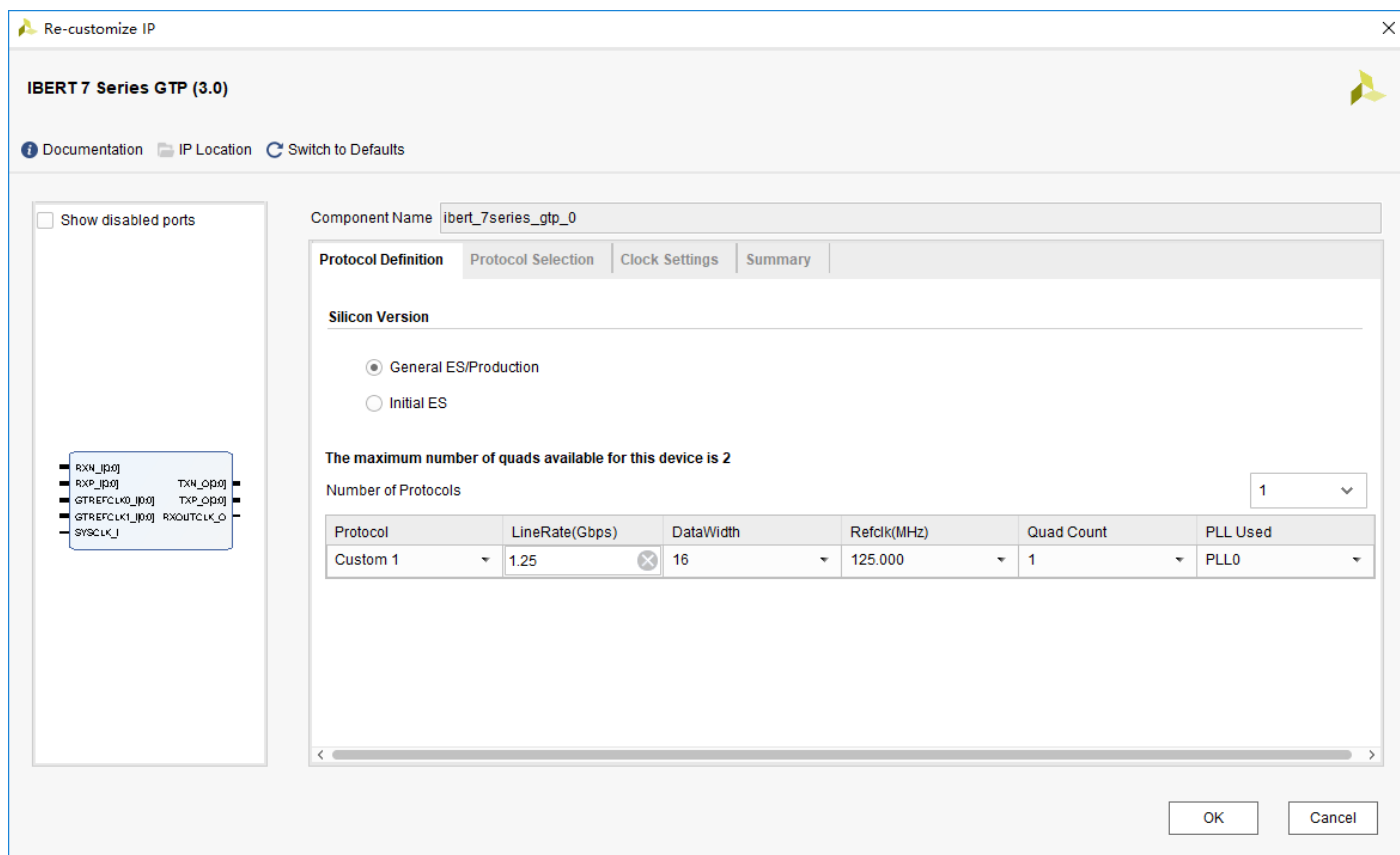
在 ARTIX-7 系列开发板中, 将光模块插入 SFP 屏蔽笼内, 然后通过单根光纤将光模块的 TX 和 RX 短接, 便可以通过 IBERT 工具对 GTP 进行测试。

在 vivado 中找到 IBERT 7 Series GTP 这个 IP 核。

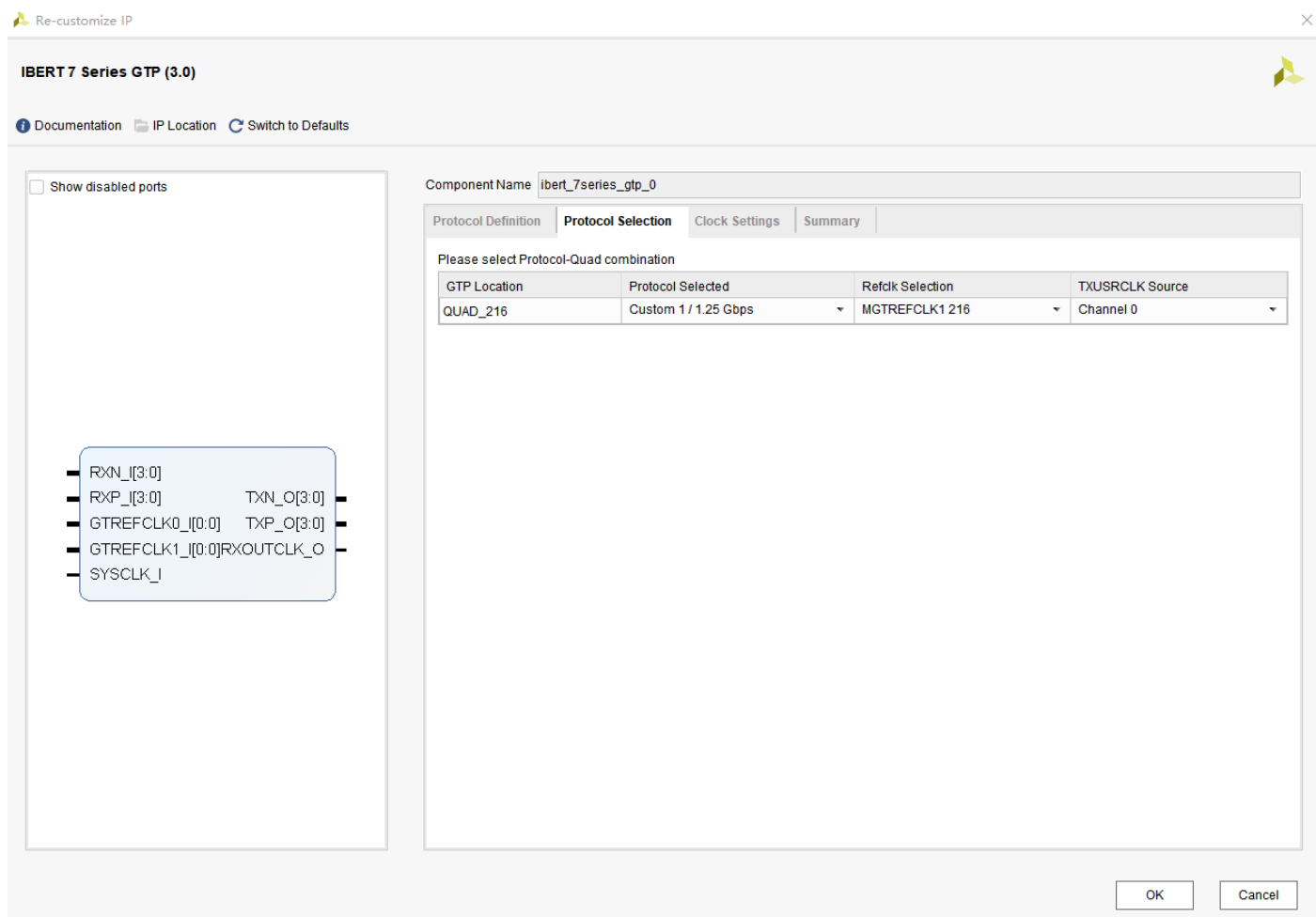


1.3.1 千兆 1.25G 速率

IBERT 测试协议选择 Custom1, GTP 参考时钟选择频率选择 125MHz, 只需要测试与 SFP 连接的 2 组 GTP, 所以 GTP Quad 选择 1。如下图所示。千兆以太网使用了 1000BASEX 标准, 采用了 8b/10b 编码方式, 所以 GTP 的传输速率为 $1000\text{Mbps} \times 10/8 = 1250\text{Mbps} = 1.25\text{Gbps}$ 。如下图所示。采用 Custom1 协议, 我们暂且不研究这个协议内容, 传输速率为 1.25Gbps。

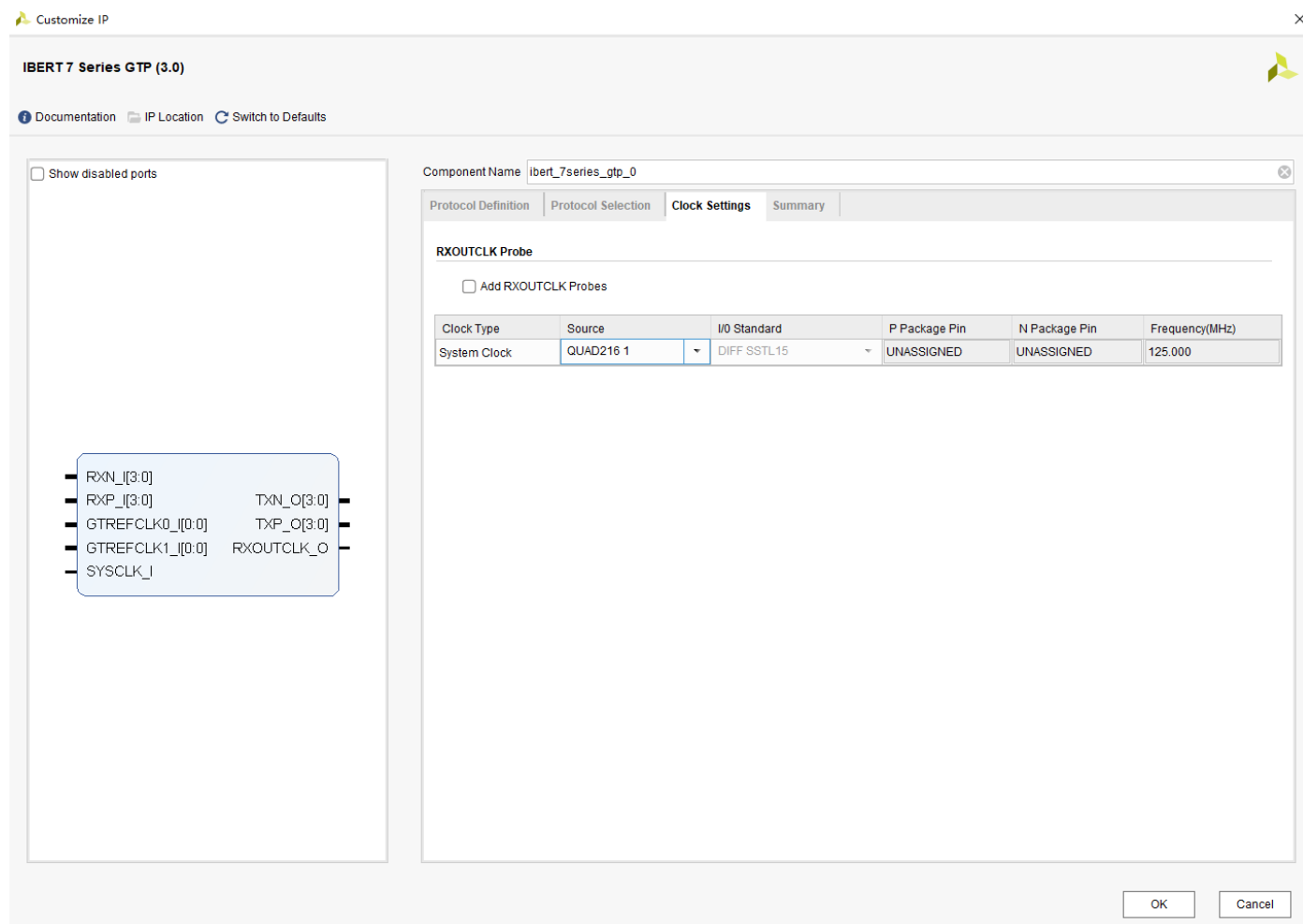


由于 2 组与 SFP 连接的 GTP 都位于 BANK 216，所以在 QUAD_216 中选择使能协议。在 A703 开发板中，底板时钟芯片输出的差分时钟是与 REFCLK1 连接，所以参考时钟要选择为 MGTREFCLK1 216。如下图所示。



不妨将 GTP 输入的 125MHz 的时钟同时作为 IBERT 内部逻辑的工作时钟，这样就可以不使用额外的外部时钟，

如下图所示。当然，用户也可以将该时钟设置为外部时钟，例如通过核心板的 FPGA 时钟晶振提供，大家可以自行尝试。



点击 OK，完成 IBERT IP 核配置。

1.3.2 千兆 6.25G 速率

IBERT 测试协议选择 Custom1，GTP 参考时钟选择频率选择 125MHz，只需要测试与 SFP 连接的 2 组 GTP，所以 GTP Quad 选择 1。如下图所示。千兆以太网使用了 1000BASEX 标准，采用了 8b/10b 编码方式，所以 GTP 的传输速率为 $1000\text{Mbps} \times 10/8 = 1250\text{Mbps} = 1.25\text{Gbps}$ 。如下图所示。采用 Custom1 协议，我们暂且不研究这个协议内容，传输速率为 1.25Gbps。

Re-customize IP

IBERT 7 Series GTP (3.0)

Documentation IP Location Switch to Defaults

Show disabled ports

Component Name: ibert_7series_gtp_0

Protocol Definition Protocol Selection Clock Settings Summary

Silicon Version

☒ General ES/Production
☐ Initial ES

The maximum number of quads available for this device is 1

Number of Protocols: 1

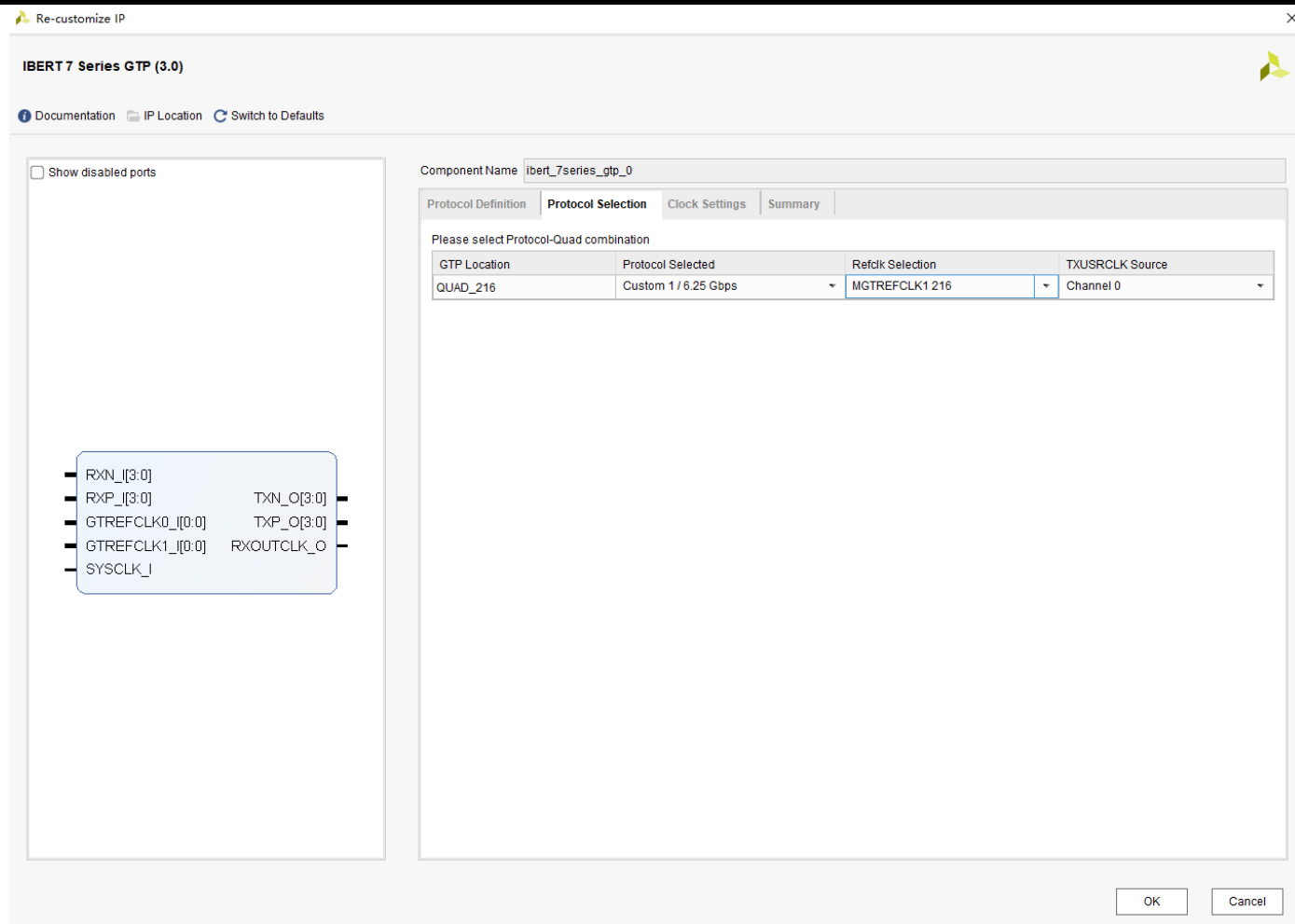
Protocol	LineRate(Gbps)	DataWidth	Refclk(MHz)	Quad Count	PLL Used
Custom 1	6.25	16	125.000	1	PLL0

OK Cancel

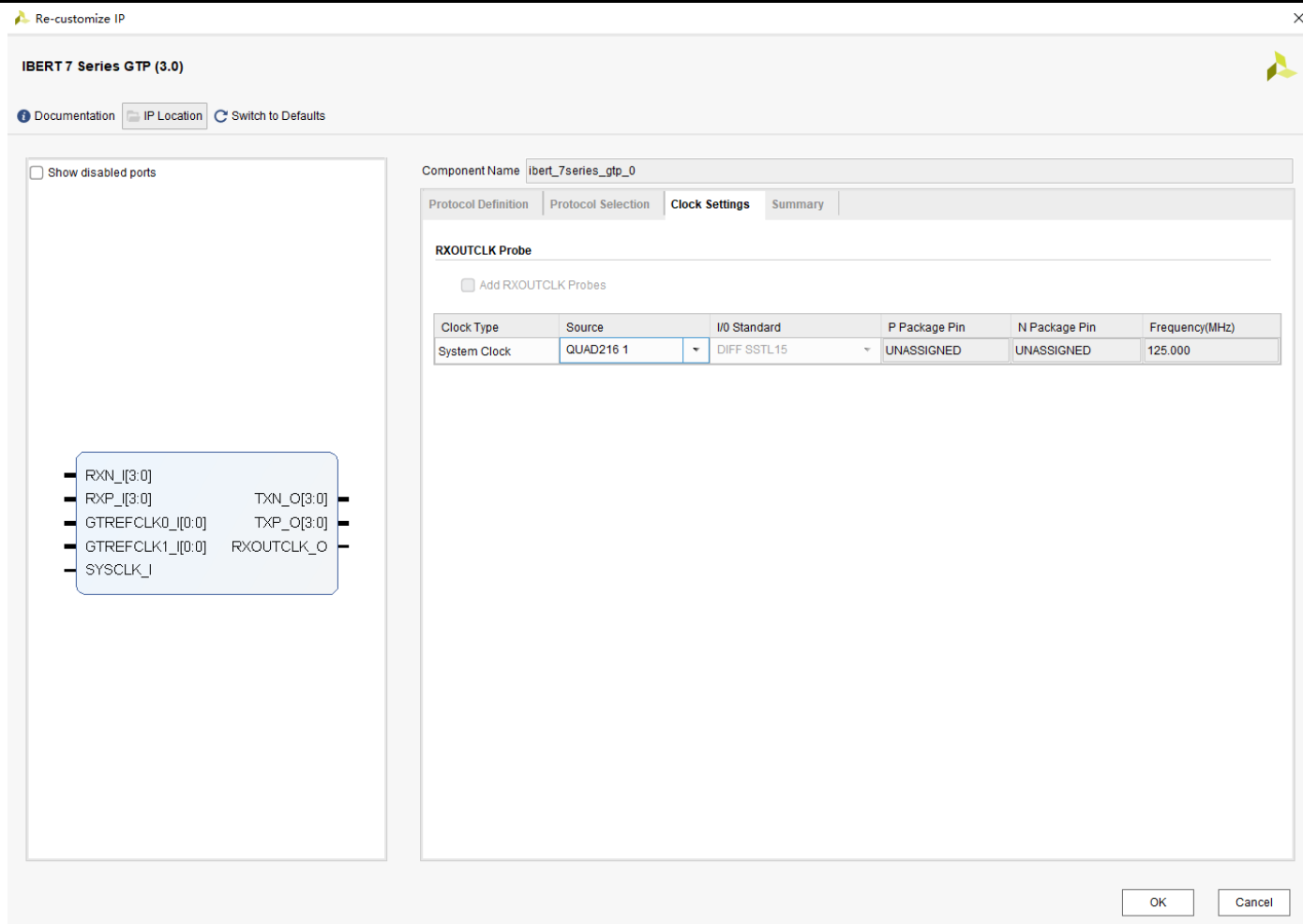
Ports:

- RXN_I[3:0]
- RXP_I[3:0]
- TXN_O[3:0]
- TXP_O[3:0]
- GTREFCLK0_I[0:0]
- GTREFCLK1_I[0:0]
- SYSClk_I
- RXOUTCLK_O

由于 2 组与 SFP 连接的 GTP 都位于 BANK 216，所以在 QUAD_216 中选择使能协议。在 A703 发板中，底板时钟芯片输出的差分时钟是与 REFCLK1 连接，所以参考时钟要选择为 MGTREFCLK1 216。如下图所示。



不妨将 GTP 输入的 125MHz 的时钟同时作为 IBERT 内部逻辑的工作时钟,这样就可以不使用额外的外部时钟,如下图所示。当然,用户也可以将该时钟设置为外部时钟,例如通过核心板的 FPGA 时钟晶振提供,大家可以自行尝试。

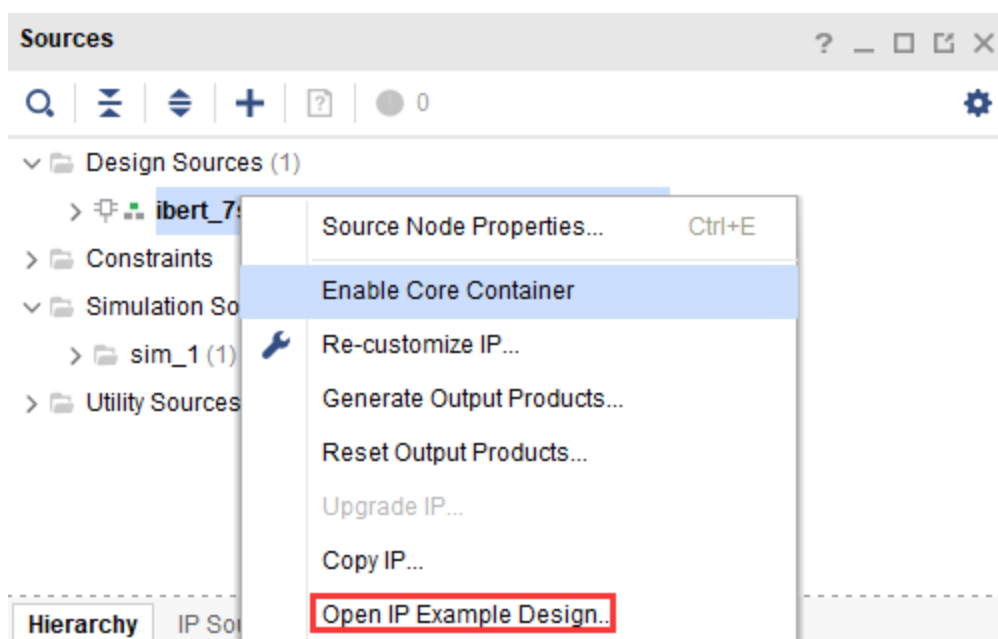


点击 OK，完成 IBERT IP 核配置。

1.4 使用 example design

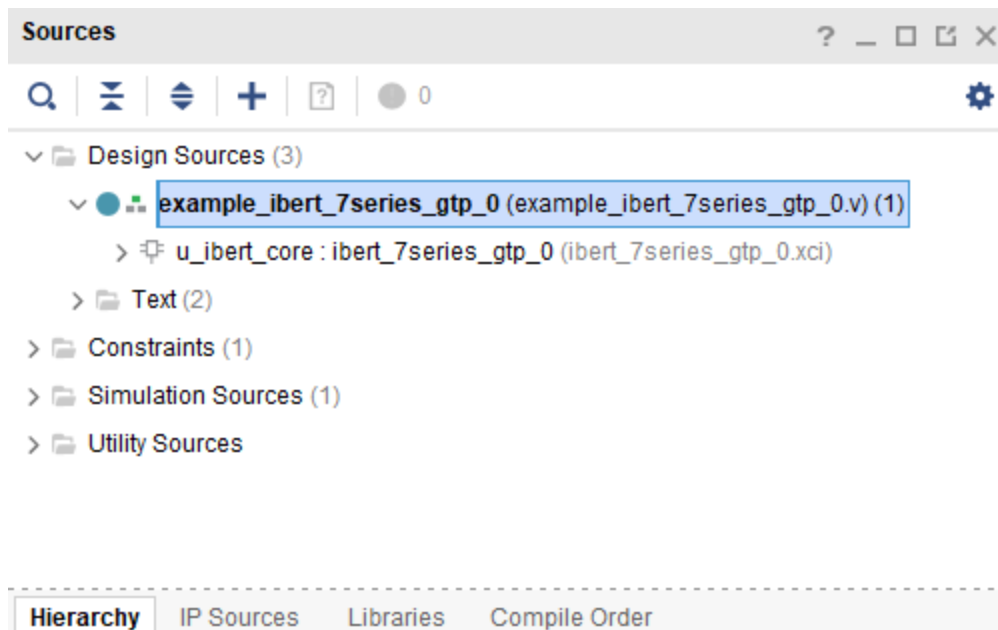
IBERT 的 example design 稍作修改就可以直接使用对 GTP 进行测试。

在 IBERT IP 核配置完成后，右击打开 example design 工程。



由于开发板的 SFP 屏蔽笼的 tx_disable 引脚都默认接了上拉电阻。要使收发回环测试可以正常进行，必须要将 tx_disable 引脚拉低。因此，在 example design 的顶层模块，添加 2 个 sfp_tx_disable 引脚，且均置为 0 即可。如下

图所示。



```

module example_ibert_7series_gtp_0
(
    // GT top level ports
    output [(4*'C_NUM_QUADS')-1:0]    TXN_0,
    output [(4*'C_NUM_QUADS')-1:0]    TXP_0,
    input  [(4*'C_NUM_QUADS')-1:0]    RXN_I,
    input  [(4*'C_NUM_QUADS')-1:0]    RXP_I,
    input  ['C_REFCLKS_USED'-1:0]      GTREFCLKOP_I,
    input  ['C_REFCLKS_USED'-1:0]      GTREFCLKON_I,
    input  ['C_REFCLKS_USED'-1:0]      GTREFCLK1P_I,
    input  ['C_REFCLKS_USED'-1:0]      GTREFCLK1N_I,
    output [1:0]                      sfp_tx_disable
);

//
// Ibert refclk internal signals
//
wire  ['C_NUM_QUADS'-1:0]    gtrefclk0_i;
wire  ['C_NUM_QUADS'-1:0]    gtrefclk1_i;
wire  ['C_REFCLKS_USED'-1:0]    refclk0_i;
wire  ['C_REFCLKS_USED'-1:0]    refclk1_i;

assign sfp_tx_disable = 2'b00;

//
// Refclk IBUFDS instantiations
//

```

然后，在 xdc 文件中添加 sfp_tx_disable 引脚的约束即可，如下图所示。注意：不同的开发板管脚定义不一样

```

set_property PACKAGE_PIN A18 [get_ports {sfp_tx_disable[0]}]
set_property PACKAGE_PIN A20 [get_ports {sfp_tx_disable[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {sfp_tx_disable[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sfp_tx_disable[1]}]

```

1.5 GTP IBERT 测试

1.5.1 1.25G 测试

用户可将千兆光模块插入任意 1 个 SFP 屏蔽笼内，也可以同时使用 1、2 个光模块插入相应的屏蔽笼内。本教程测试时只使用了 1 个光模块，插入丝“SFPA”所对应的屏蔽笼，如下图所示。

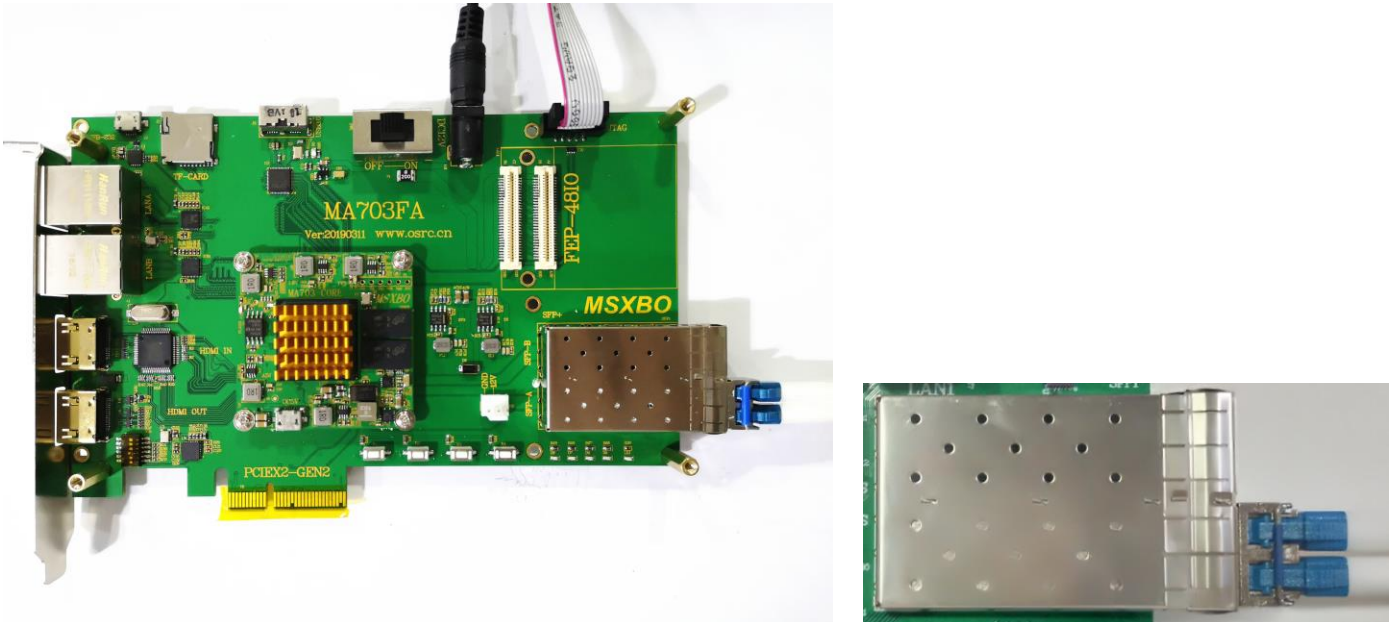


图 1-5-1-1 开发板测试连接图

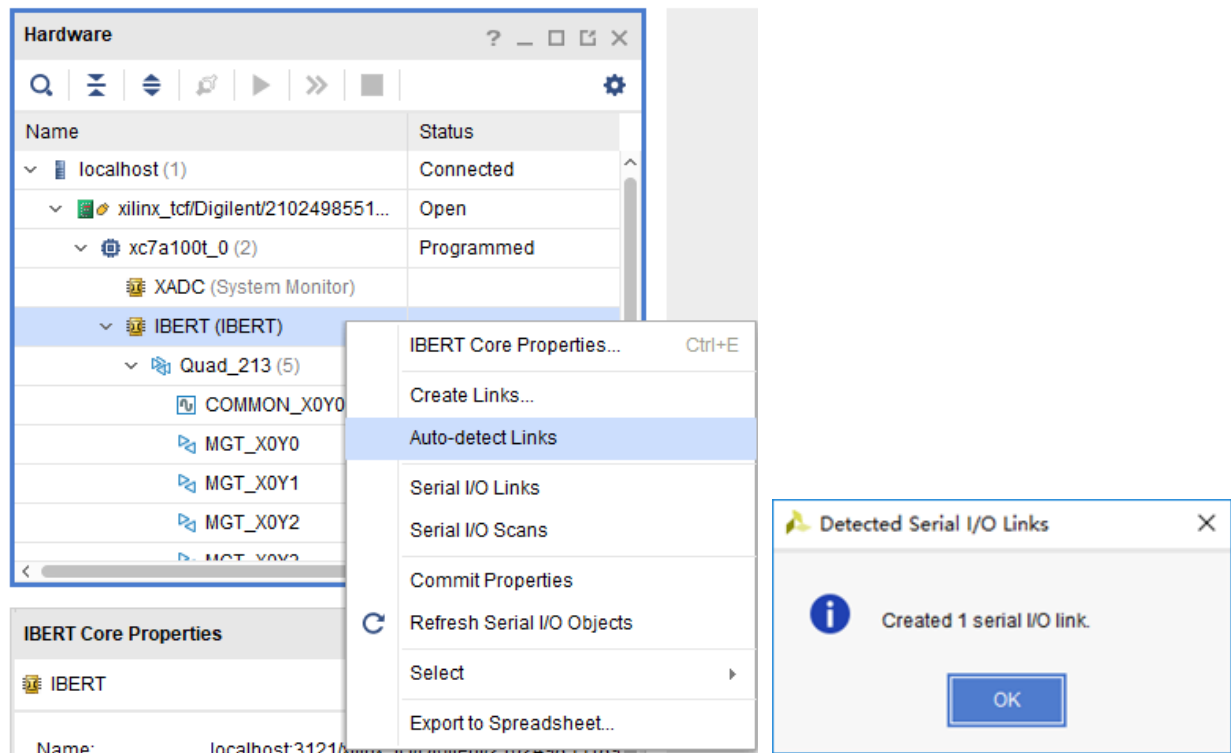
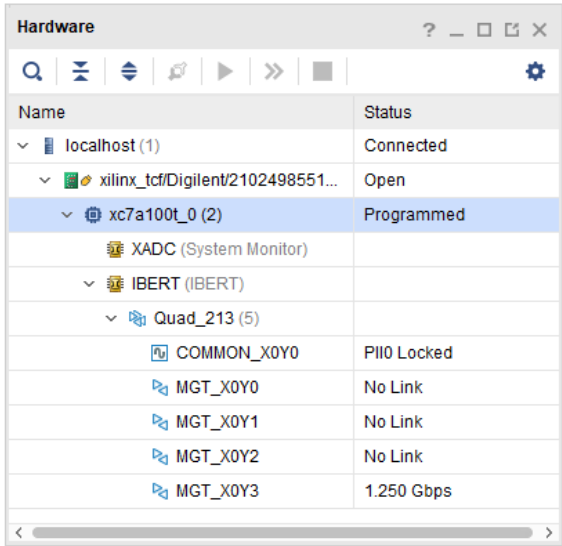
表 1-5-1 可编程时钟模式设置

Prescale Divider	Feedback Divider	PR1/PR0	VCO MHZ	Output Divider	OD2/OD1 /OD0	Output MHZ	Application
4	20	11	2000	4	011	125	GigE



图 1-5-1-2 拨码开关设置

使用单根光纤，将光模块的 TX 和 RX 短接。然后，给开发板上电，在 vivado 中打开 hardware manager，将刚才生成的 bit 文件下载到开发板中。下载完成后，出现如下图所示的界面。



点击 OK。然后出现如下图所示的界面。点击 Reset 按钮，使 IBERT 进行复位，可以看到此时的 Errors 变为 0，代表接收端没有检测到错误。由于测试使只连接了 1 路 GTP，因此这里只显示出了当前所使用的 GTP 链路。其他没有建立收发环路的 GTP 并没有显示。

Tcl ConsoleMessagesSerial I/O LinksSerial I/O Scans

🔍

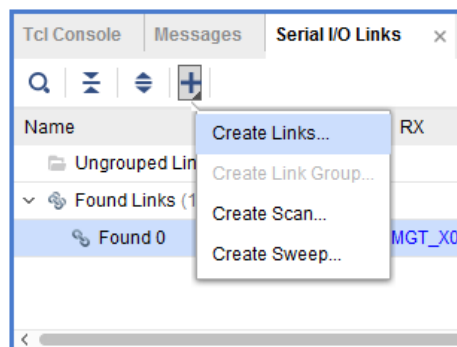
⌵

⌵

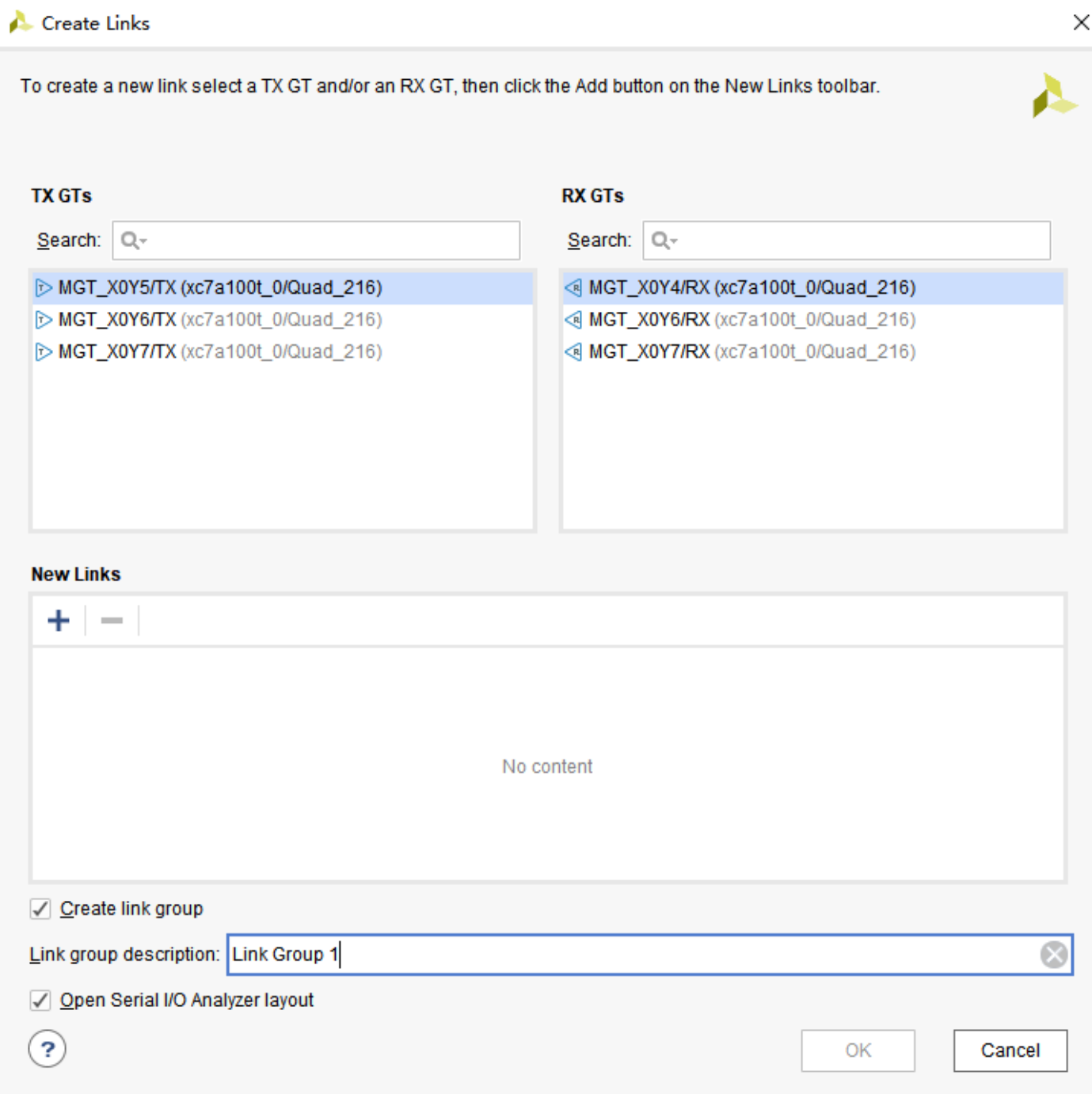
+

Name	TX	RX	Status	Bits	Errors	BER	BERT Reset	TX Pattern	RX Pattern
📁 Ungrouped Links (0)									
▼🔗 Found Links (1)							Reset	PRBS 7-bit	PRBS 7-bit
🔗 Found 0	MGT_X0Y4/TX	MGT_X0Y5/RX	1.250 Gbps	3.648...	0E0	2.741E...	Reset	PRBS 7-bit	PRBS 7-bit

当然，用户也可以手动将其他没有连接的 GTP 加到当前的显示栏中，点击如下图所示按钮。



然后，将同 1 个 GTP 对应的 TX 和 RX 建立 link，如下图所示。



添加完成后如下图所示。点击 OK 即可。

随后显示框中出现了这 3 对没有连接的 GTP，状态显示为“No Link”，如下图所示。

Tcl ConsoleMessagesSerial I/O LinksSerial I/O Scans

Q

≡

⇅

+

Name	TX	RX	Status	Bits	Errors	BER	BERT Reset	TX Pattern	RX Pattern
Ungrouped Links (0)									
Found Links (1)							Reset	PRBS 7-bit	PRBS 7-bit
Found 0	MGT_X0Y4/TX	MGT_X0Y5/RX	1.250 Gbps	1.303E11	0E0	7.677E-12	Reset	PRBS 7-bit	PRBS 7-bit
Link Group 1 (3)							Reset	PRBS 7-bit	PRBS 7-bit
Link 1	MGT_X0Y5/TX	MGT_X0Y4/RX	No Link	1.3E11	3.034E10	2.335E-1	Reset	PRBS 7-bit	PRBS 7-bit
Link 2	MGT_X0Y6/TX	MGT_X0Y6/RX	No Link	1.303E11	5.835E10	4.479E-1	Reset	PRBS 7-bit	PRBS 7-bit
Link 3	MGT_X0Y7/TX	MGT_X0Y7/RX	No Link	1.295E11	6.1E10	4.712E-1	Reset	PRBS 7-bit	PRBS 7-bit

如下图所示, 经过长时间测试, 可以发现 Errors 一直为 0, 这代表, 测试过程中没有出现任何误码, 这说明板级层面的 GTP 硬件工作稳定。

Tcl ConsoleMessagesSerial I/O LinksSerial I/O Scans

Q

≡

⇅

+

Name	TX	RX	Status	Bits	Errors	BER	BERT Reset	TX Pattern	RX Pattern
Ungrouped Links (0)									
Found Links (1)							Reset	PRBS 7-bit	PRBS 7-bit
Found 0	MGT_X0Y4/TX	MGT_X0Y5/RX	1.250 Gbps	1.653E11	0E0	6.051E-12	Reset	PRBS 7-bit	PRBS 7-bit
Link Group 1 (3)							Reset	PRBS 7-bit	PRBS 7-bit
Link 1	MGT_X0Y5/TX	MGT_X0Y4/RX	No Link	1.649E11	3.851E10	2.335E-1	Reset	PRBS 7-bit	PRBS 7-bit
Link 2	MGT_X0Y6/TX	MGT_X0Y6/RX	No Link	1.653E11	7.401E10	4.478E-1	Reset	PRBS 7-bit	PRBS 7-bit
Link 3	MGT_X0Y7/TX	MGT_X0Y7/RX	No Link	1.643E11	7.737E10	4.71E-1	Reset	PRBS 7-bit	PRBS 7-bit

当前测试使用的数据为 7bit 的伪随机序列。

Tcl ConsoleMessagesSerial I/O LinksSerial I/O Scans

Q

≡

⇅

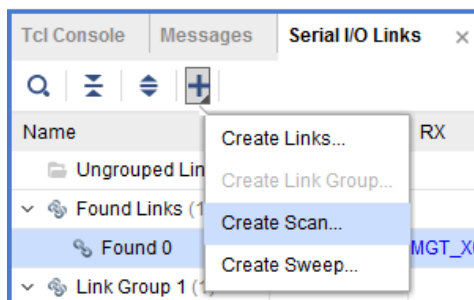
+

Name	TX	RX	Status	Bits	Errors	BER	BERT Reset	TX Pattern	RX Pattern
Ungrouped Links (0)									
Found Links (1)							Reset	PRBS 7-bit	PRBS 7-bit
Found 0	MGT_X0Y4/TX	MGT_X0Y5/RX	1.250 Gbps	2.028E11	0E0	4.932E-12	Reset	PRBS 7-bit	PRBS 7-bit

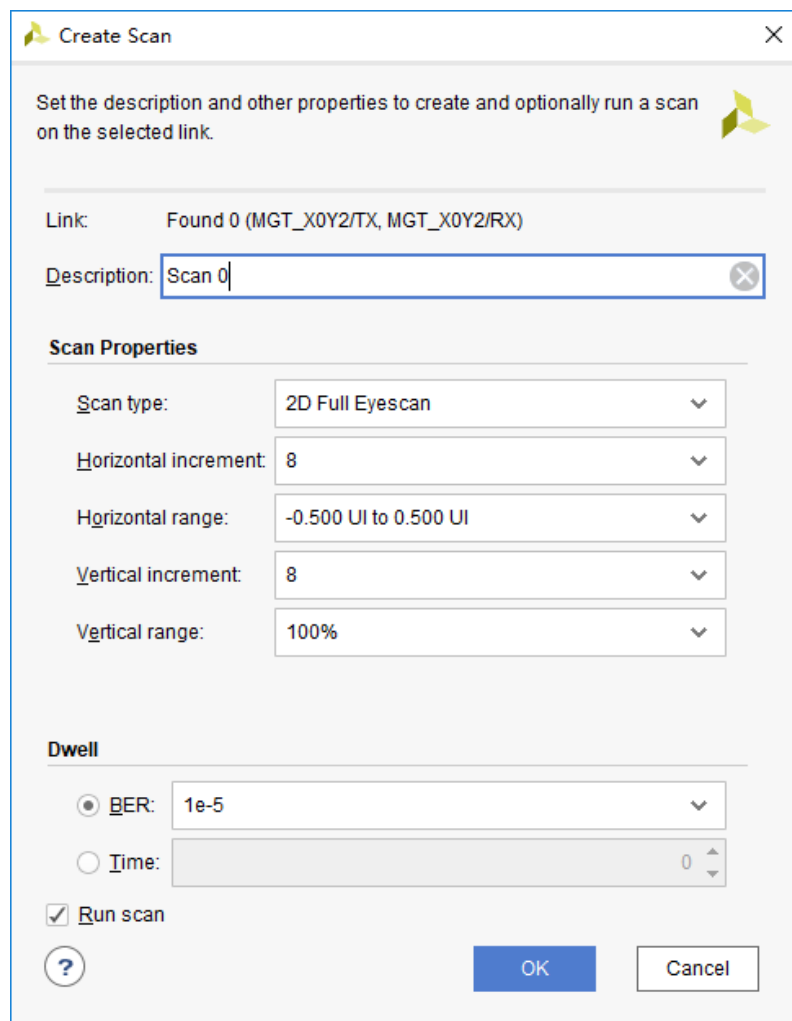
在测试进行时用户也可以修改测试数据类型, 例如 PRBS 23-bit, PRBS 31-bit 等等, 接收与发送所使用的数据类型必须完全一致, 修改完成后, 务必点击 Reset 按钮, 复位 Errors 为 0。如下图所示。

Name	TX	RX	Status	Bits	Errors	BER	BERT Reset	TX Pattern	RX Pattern
Ungrouped Links (0)									
Found Links (1)							Reset	PRBS 23-bit	PRBS 23-bit
Found 0	MGT_X0Y12/TX	MGT_X0Y12/RX	1.250 Gbps	3.779E9	0E0	2.646E...	Reset	PRBS 23-bit	PRBS 23-bit

误码率可以从数学统计的角度判断 GTP 的硬件稳定性, vivado 还提供了一种更直观的方式来观察 GTP 的信号完整性, 那就是眼图。首先, 点击如下图所示按钮。



出现如下界面, 所有设置保持默认即可, 然后点击 OK。



Create Scan

Set the description and other properties to create and optionally run a scan on the selected link.

Link: Found 0 (MGT_X0Y2/TX, MGT_X0Y2/RX)

Description: Scan 0

Scan Properties

Scan type: 2D Full Eyescan

Horizontal increment: 8

Horizontal range: -0.500 UI to 0.500 UI

Vertical increment: 8

Vertical range: 100%

Dwell

☒ BER: 1e-5

☐ Time: 0

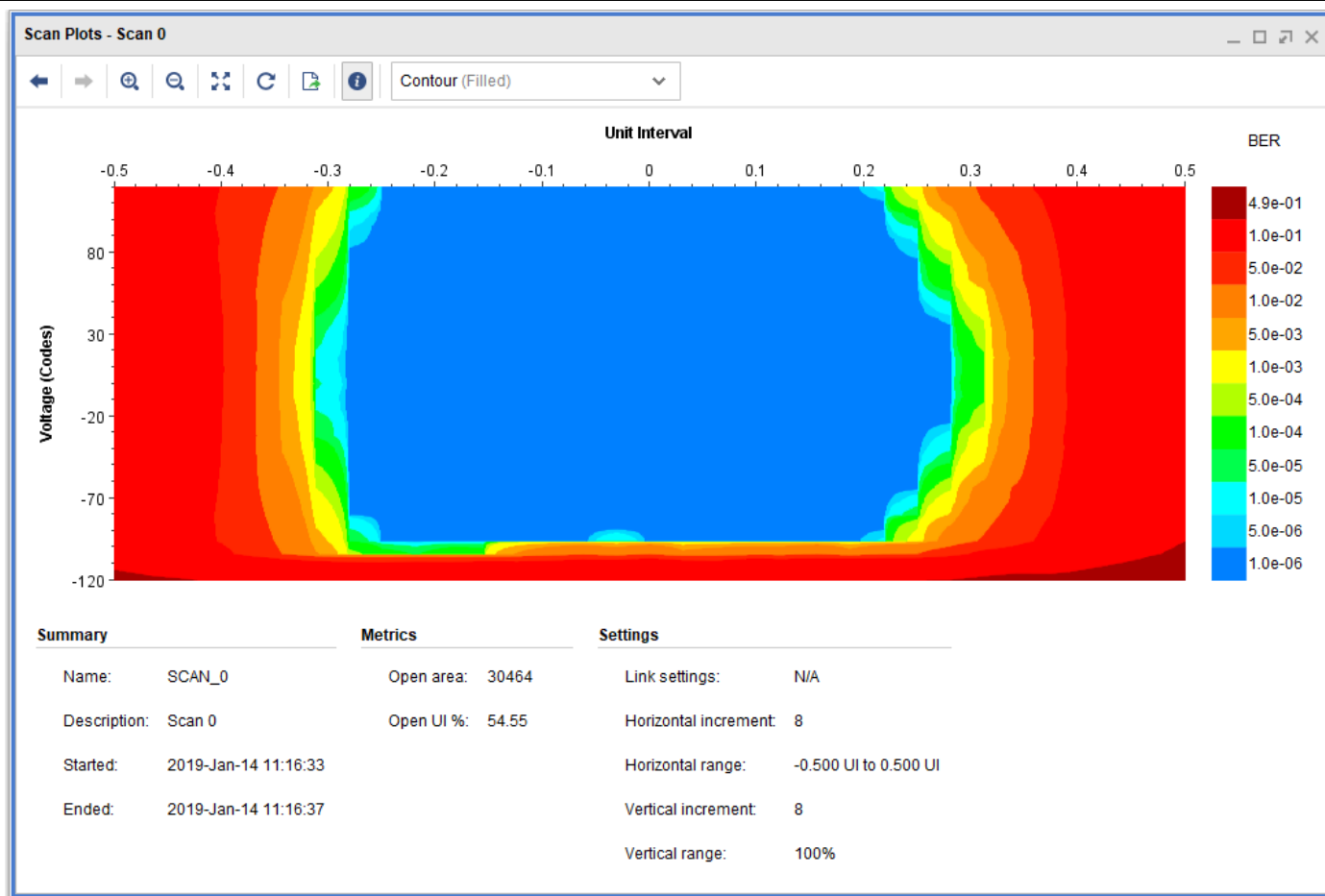
☒ Run scan

OK Cancel

此时 vivado 开始进行眼图扫描和生成，用户需要稍等片刻，等到 Progress 进度为 100%时，眼图扫描过程便结束。

Serial I/O Scans									
Name	Link	Link Settings	Reset RX	Scan Type	Status	Progress	Open Area	Open UI %	Horz Incr
Scans (1)									
Scan 1			<input type="checkbox"/>	2d_full_eye	In Progre...	<div><div></div></div> 67%	62912	0	8

vivado 生成的眼图如下图所示。



从信号完整性的角度来看，眼图中间的蓝色区域越大，GTP 所对应的 PCB 高速电路的信号完整性越好。

1.5.2 6. 25G 测试

6.25G 的测试方法与上述的千兆测试方法完全相同。将光模块插入丝印“SFP-A”所对应的屏蔽笼。

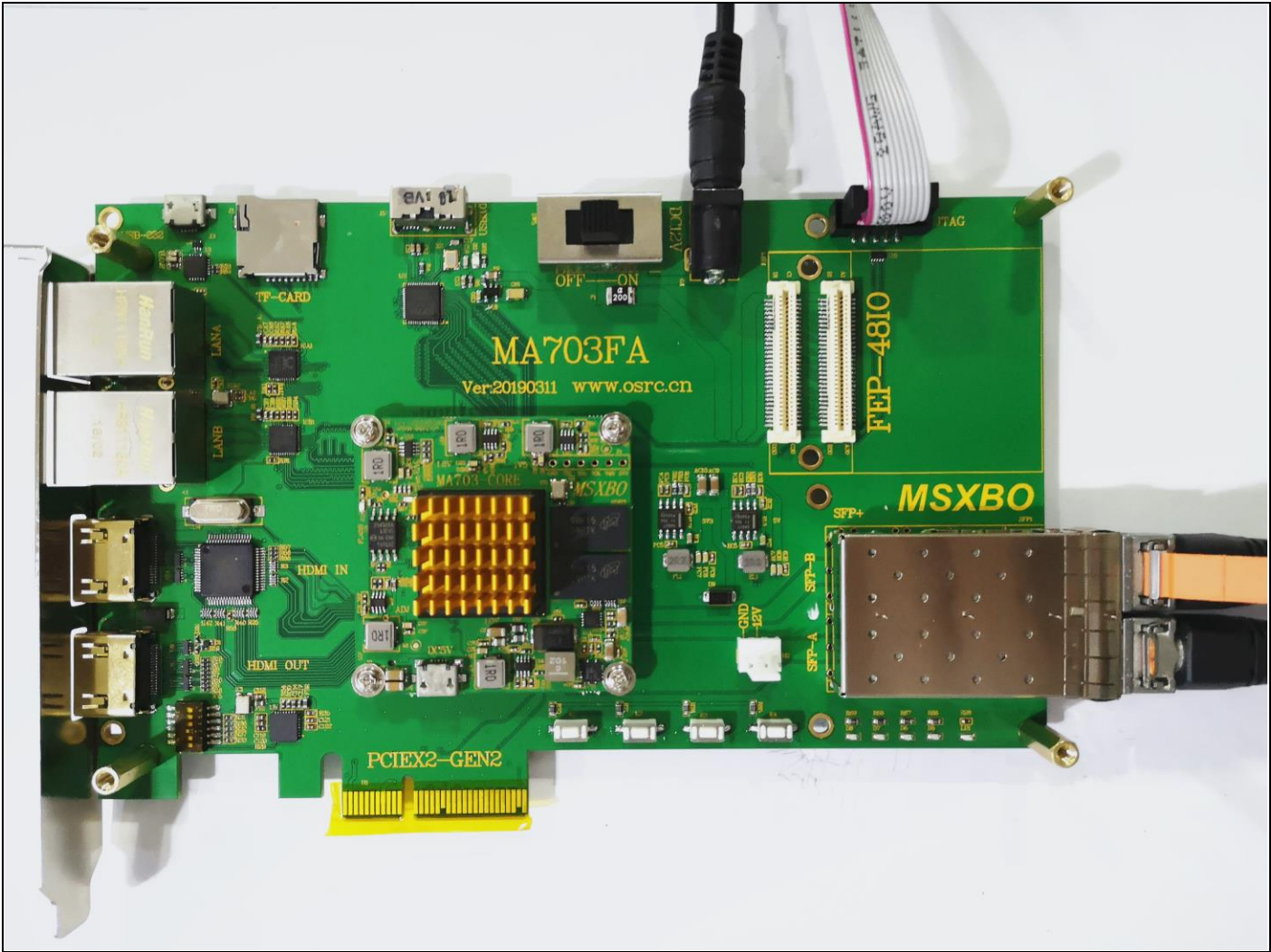


图 1-5-2-1 开发板测试连接图

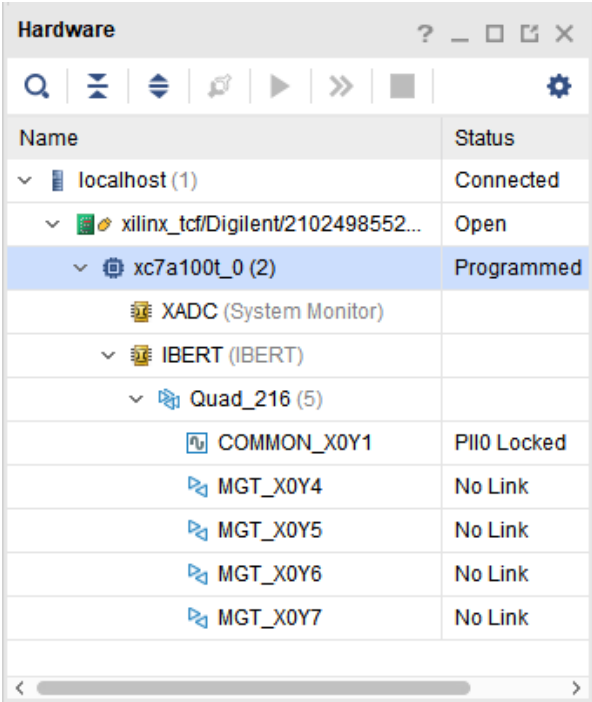
表 1-5-2 可编程时钟模式设置

Prescale Divider	Feedback Divider	PR1/PR0	VCO MHZ	Output Divider	OD2/OD1 /OD0	Output MHZ	Application
4	20	11	2000	4	011	125	GigE



图 1-5-2-2 拨码开关设置

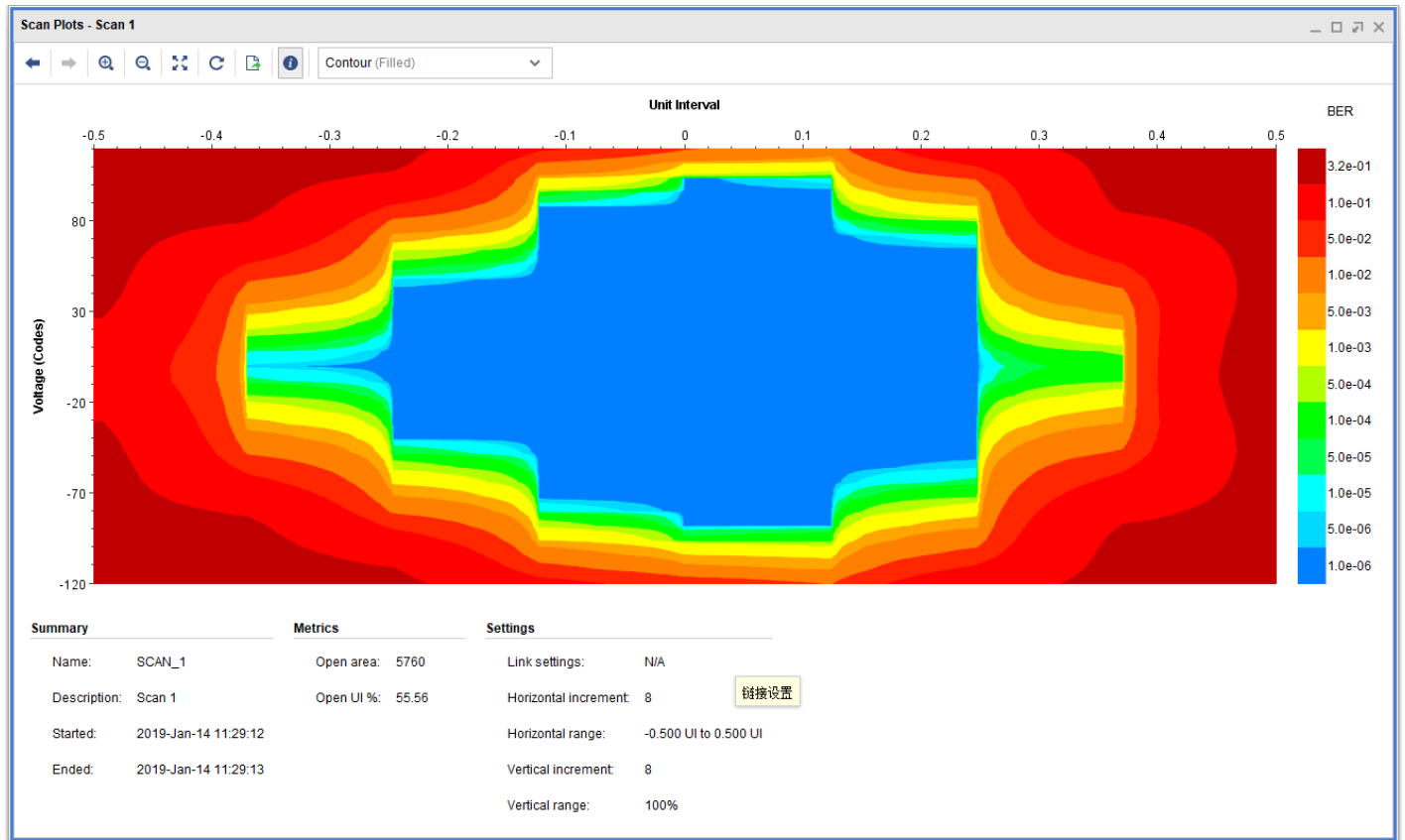
然后上电，下载 example design 生成的 bit 文件。如下图所示。



进行长时间测试后，GTP 在万兆速率下的误码同样为 0。如下图所示。

Tcl Console Messages Serial I/O Links x Serial I/O Scans									
Q Z A +									
Name	TX	RX	Status	Bits	Errors	BER	BERT Reset	TX Pattern	RX Pattern
Ungrouped Links (0)									
Found Links (2)									
Found 0	MGT_X0Y5/TX	MGT_X0Y4/RX	6.250 G...	5.852...	1.219...	2.083E-1	Reset	PRBS 7-bit	PRBS 7-bit
Found 1	MGT_X0Y4/TX	MGT_X0Y5/RX	6.250 G...	5.852...	2.241...	3.829E-1	Reset	PRBS 7-bit	PRBS 7-bit

同理，扫描万兆速率下 GTP 链路的眼图，如下图所示。万兆速率下的眼图蓝色区域明显小于千兆速率时的眼图，而且形状发生了改变，变得有点像椭圆。这与信号的频率有关，信号的频率越高，在传输过程中的损耗越大，上下沿越来越平缓，方波会变得越来越像正弦波，眼图也会“睁得”越来越小。若使用更高的速率，眼图中的蓝色区域可能会变得更小。



CH02 aurora_8b10b 光通信

软件版本: VIVADO2019.2

操作系统: WIN10 64bit

硬件平台: 适用 XILINX A7/K7/Z7/ZU/KU 系列 FPGA

登录米联客(MSXBO)FPGA 社区-www.uisrc.com 观看免费视频课程、在线答疑解惑!

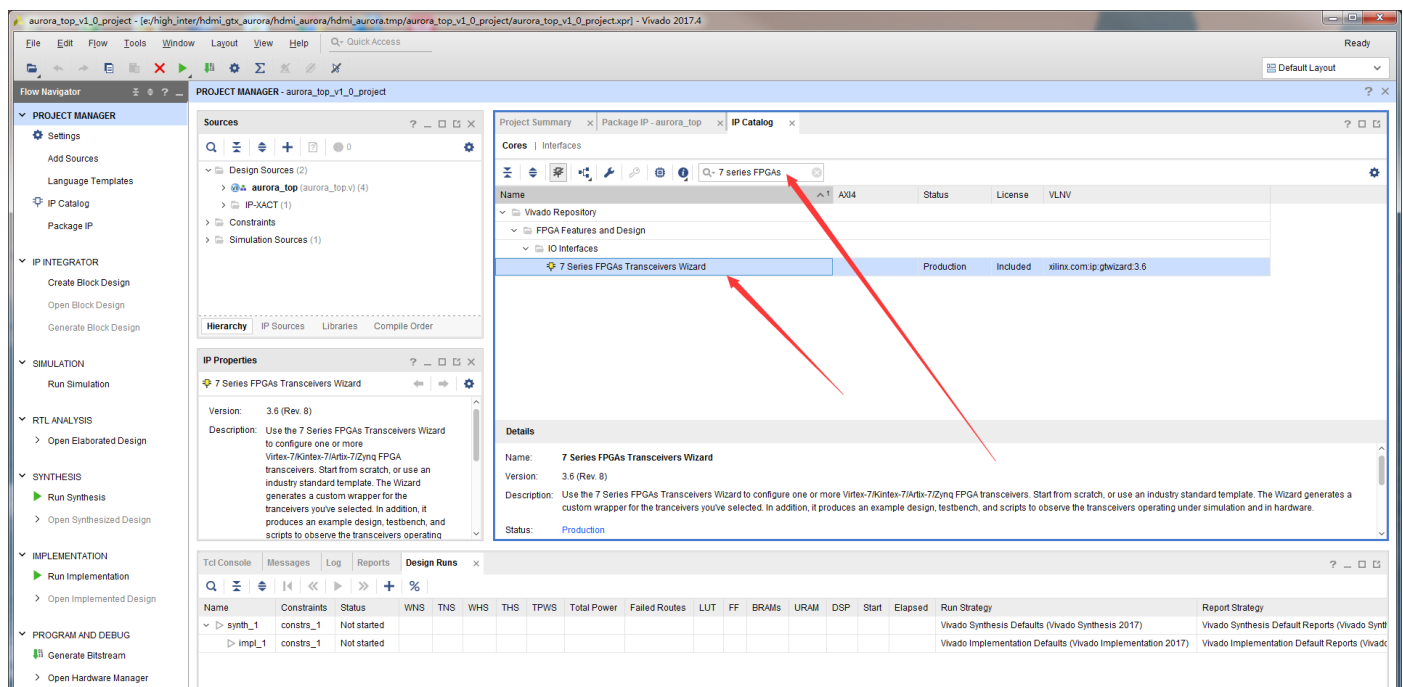
2.1 概述

本课内容讲解 XILINX 官方的 7 系列 FPGA 高速串行收发器的使用。7 Series FPGAs Transceivers Wizard 这个 IP 是一个非常实用的工具。Xilinx 7 系列 FPGA 全系所支持的 GT 一共分为四种: GTP、GTX、GTH 和 GTZ。7 系列中,按支持的最高线速率排序,GTP 是最低的,GTZ 是最高的。GTP 被用于 A7 系列,GTZ 被用于少数 V7 系列。K7 常见的是 GTX。本课是用过外部的光纤线链接的,实际使用中也可以直接用在开发板之间链接,实现板子到板子的通信。本课程内容可以参考 XILINX 官方文档 PG168。

2.2 Setp By Step 搭建 FPGA 工程

2.2.1 7 Series FPGAs Transceivers Wizard 的配置

打开 vivado 在 IP 中搜索 7 series FPGAs, 找到 7 Series FPGAs Transceivers Wizard 如图所示:



双击打开 gt ip 核, 按如下步骤设置:

Step1: 第一页设置: 默认即可

Component Name

GT Selection | Line Rate, RefClk Selection | Encoding and Clocking | Comma Alignment and Equalization | PCIe, SATA, PRBS | CB and CC Sequence | Summary

Artix 7 GTP Silicon revision supported by this IP core version: Production
Please use v2.4 of this wizard to program Initial ES/General ES GTP devices.

GT Type

GT Type

Shared Logic

Select whether the transceiver quad PLL, transceiver differential refclk buffer, clocking and reset logic are included in the core itself or in the example design

☐ Include Shared Logic in core

☒ Include Shared Logic in example design

Shared Logic Overview

Include Shared Logic in example design

- For users who want the Shared Logic outside the core.
- For users who want to edit the Shared Logic or use their own.
- For users who want one core with Shared Logic to drive multiple cores without Shared Logic.

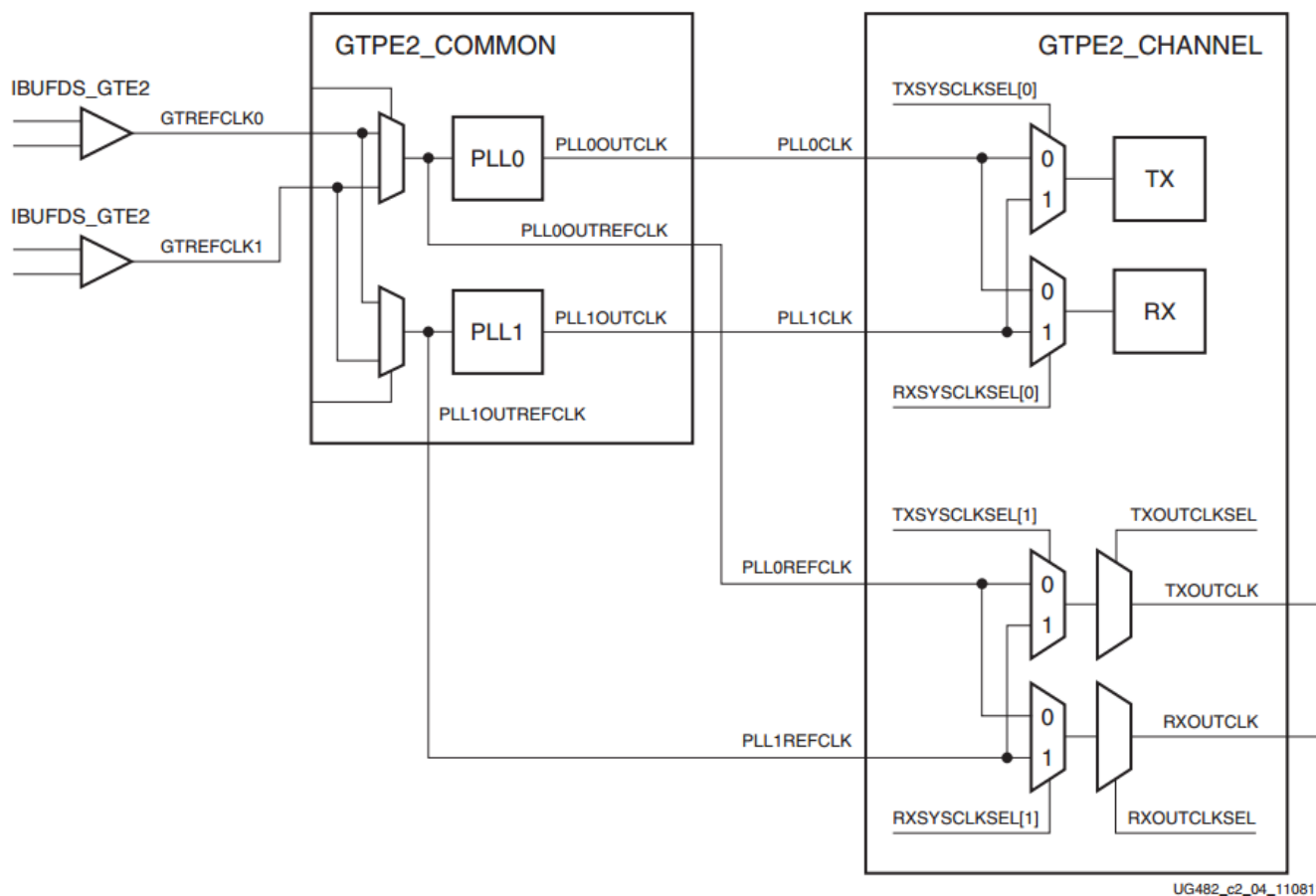
OK Cancel

Step2: 第二页设置: 红色箭头所指, 首先设置 Protocol 为 aurora 8b10b single lane 4 byte, 也就是对外接口为 32 位; 设置 Line Rate 为 5G, 参考时钟为 125, 这个是来源于板子的差分晶振, 左下角黄色的地方就是我们用到的高速收发器。以下内容对于不同的开发板设置不一样, 以 MZ703FA 为例, MZ703FA 有 2 个 SFP 接口分别是位于 GTP_X0Y0 和 GTP_X0Y01。利用光纤环路测试, 可以使用单头的把 RX TX 利用光钎短接直接环路, 也可以用 2 个带光模块的光钎环路。左右的为单头光模块, 右边的为双头光模块。

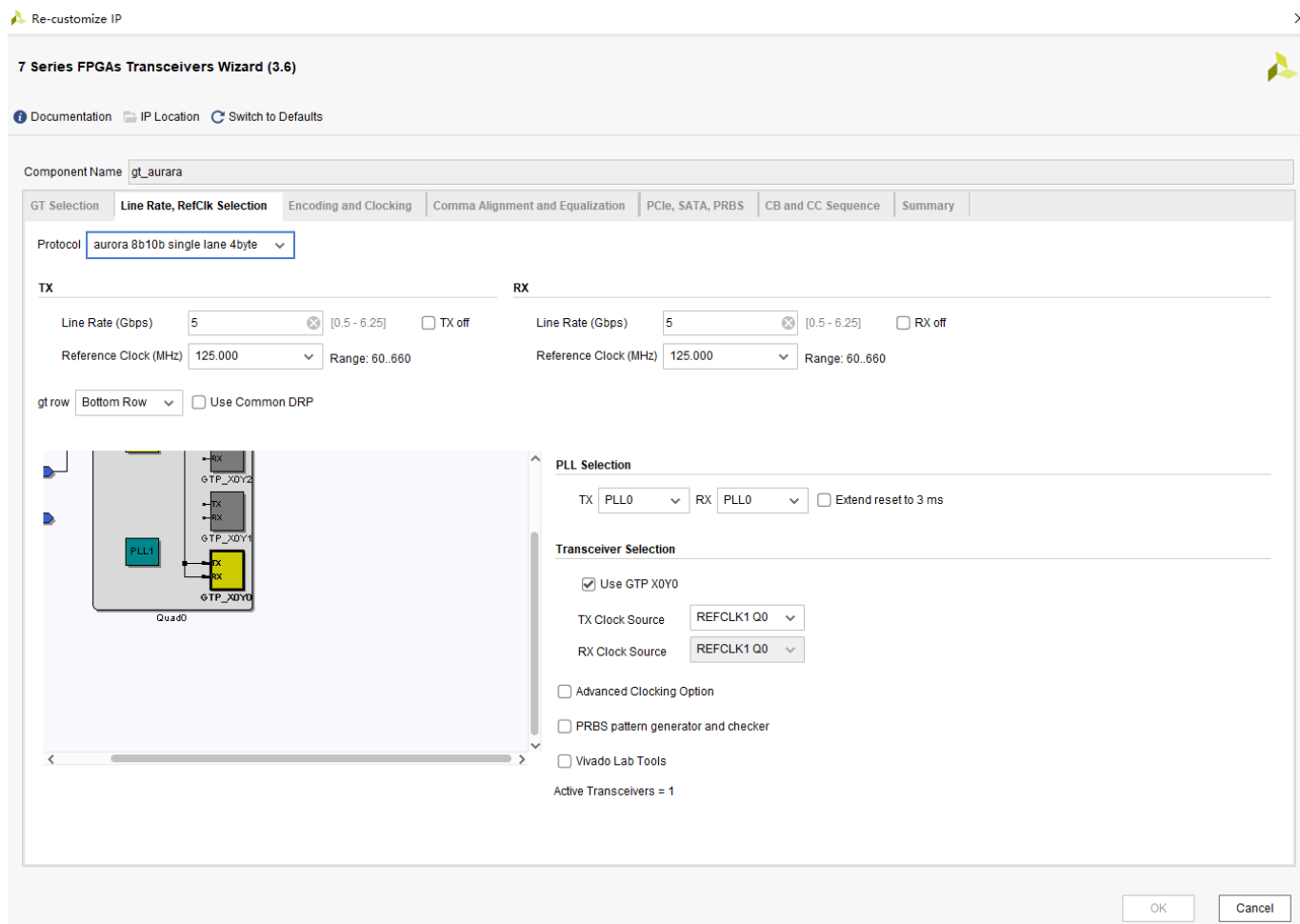


Step3: 速率、参考时钟设置:

Protocol 里面有多种协议可以提供选择, 我们这里选择 Aurora 8b10b single lane 4byte, 收发器是独立的, 可以选择不同的编码和速度, 对于 GTP 收发器, 最大是 6.6Gbps, 我们这里统一选择 5Gbps。这里的参考时钟必须和开发板上的时钟一致, 否则会无法通信。

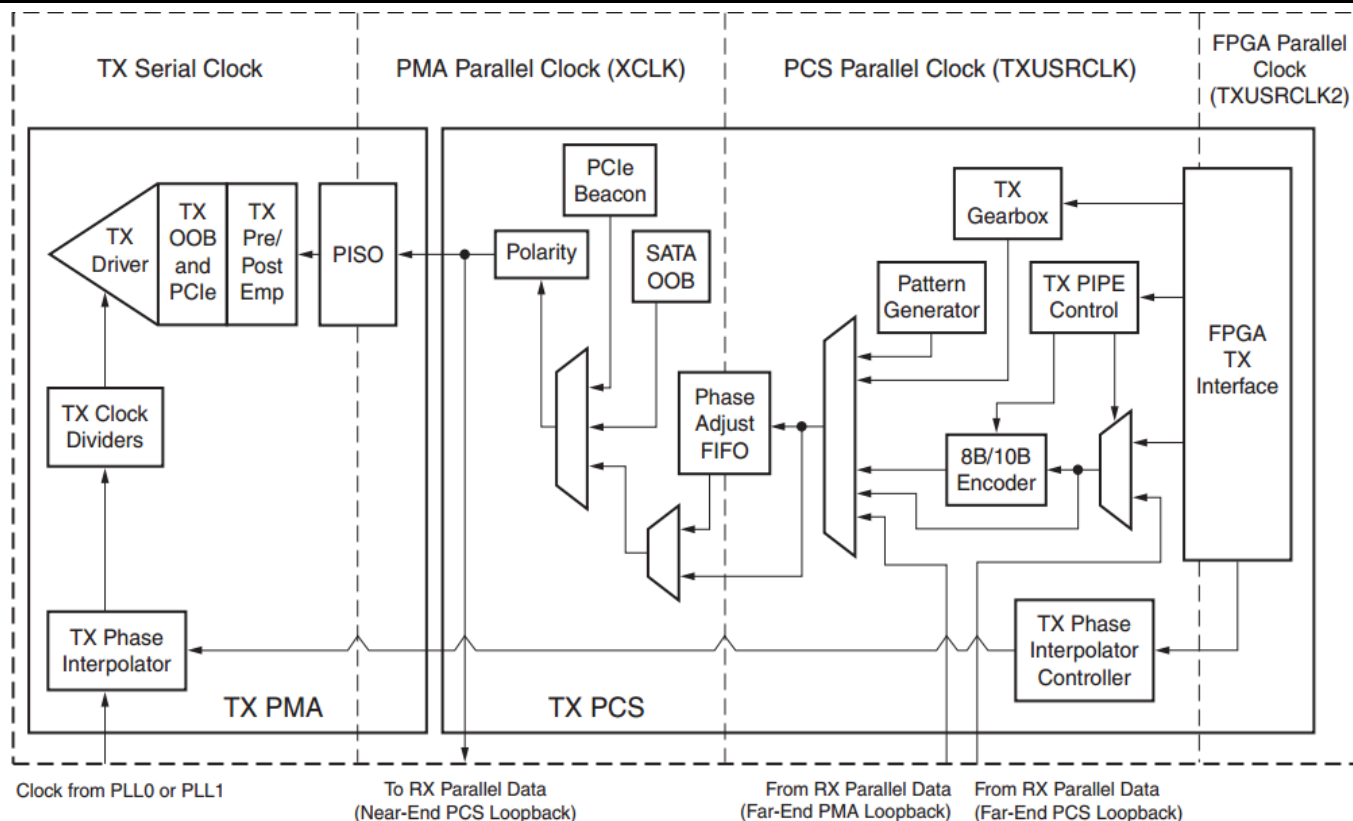


GTP 有两个参考时钟输入端口，经差分-单端转换后通过两个 PLL 产生收发器发送和接收时钟。若 TX 和 RX 线速率一致使用同一个 PLL 产生时钟，否则需要使用两个不同的 PLL。开发板中差分晶振连接 GTPREFCLK1，且收发速率相同，故 PLL Selection TX 和 RX 均选择 PLL0。由于 MA703 开发板上只有 2 个 SFP 接口，分别定义到了 GTP_X0Y0 和 GTP_X0Y1，因此对于使用 1 个光模块的配置，选择 GTP_X0Y0，对于使用 2 个光模块的配置为选择 GTP_X0Y1，GTP_X0Y1。



Step4: 页将内部数据宽度设置为 20(16bit 数据利用 8b10b 需要 20bit 表示), 2 个内部数据拼接为 1 个 32bit 外部数据, 通过计算我们可以得到内部时钟为 $5 \times 10^3 / 20 = 250\text{M}$, 而外部时钟为 $5 \times 10^3 / (20 \times 2) = 125\text{M}$ 。编码方式采样 8B/10B 编码, 这种编码方式最主要的目的是“直流平衡”, 即根据特定的编码表实现数据传输过程中比特“0”和比特“1”的数量基本一致, 且减少连 0 和连 1 的情况。编码后的数据流具有较多的跳变, 有助于接收端时钟数据恢复(CDR)。DRP/System Clock Frequency 是动态重配置或系统工作时钟, 通过 DPR 可以让设计者根据所选线速率和定义的协议实时调整收发器参数, 这个比较复杂我们用不到, 不加说明。DRP 时钟选择 100MHz, 可通过外部 PLL IP 核产生, DRP 时钟主要用来初始化一些 GTP 控制器的信号或者参数。

TXUSERCLK 和 RXUSERCLK 时钟的选择概念必要重要, 一般 TXUSERCLK 可以直接选择 TXOUTCLK 作为用户时钟, 而 RXUSERCLK, 可以选择 TXOUTCLK 也可以选择 RXUSERCLK, 还可以选择 RXPLLREFCLK 作为用户时钟。这里就涉及到一个同步的概念, 如果时钟不一致必须设置 TX 和 RX 的 BUFFER, 默认都是设置的。另外输入 TX 的发送时钟和 RX 的恢复时钟差异交大, 可能会导致 RX BUFFER 的溢出或者读空, 而导致数据出错。比较好的解决办法就是设置 RXUSERCLK 为 RXOUTCLK, 就是利用恢复时钟作为用户时钟, 这样就很好的解决了同步的问题了。



GT Selection	Line Rate, RefClk Selection	Encoding and Clocking	Comma Alignment and Equalization	PCIe, SATA, PRBS	CB and CC Sequence	Summary
<div> <div>TX</div> <div>RX</div> </div>						
<div> <div>External Data Width (Bits) 32</div> <div>External Data Width (Bits) 32</div> </div>						
<div> <div>Encoding 8B/10B</div> <div>Decoding 8B/10B</div> </div>						
<div> <div>Internal Data Width (Bits) 20</div> <div>Internal Data Width (Bits) 20</div> </div>						
<div> <div><input checked="" type="checkbox"/> Use DRP</div> <div>DRP/System Clock Frequency (MHz) 100</div> </div>						
<div>Optional Ports</div> <div> <div><input type="checkbox"/> TXBYPASS8B10B</div> <div><input checked="" type="checkbox"/> TXCHARDISPMODE</div> <div><input checked="" type="checkbox"/> TXCHARDISPVAL</div> <div><input checked="" type="checkbox"/> RXCHARISCOMMA</div> <div><input checked="" type="checkbox"/> RXCHARISK</div> <div><input type="checkbox"/> RXSTARTOFSEQ</div> </div>						
<div>Synchronization and Clocking</div> <div> <div>TX</div> <div>RX</div> </div>						
<div> <div><input checked="" type="checkbox"/> Enable TX Buffer</div> <div><input checked="" type="checkbox"/> Enable RX Buffer</div> </div>						
<div> <div>TXUSRCLK Source TXOUTCLK</div> <div>RX Buffer Bypass Mode Auto</div> </div>						
<div> <div>TXOUTCLK Source <input type="checkbox"/> Use TXPLLREFCLK</div> <div>RXUSRCLK Source RXOUTCLK</div> </div>						
<div> <div>RXOUTCLK Source <input type="checkbox"/> Use RXPLLREFCLK</div> </div>						
<div>Optional Ports</div> <div> <div><input checked="" type="checkbox"/> TXPCRESET</div> <div><input checked="" type="checkbox"/> TXPMARESET</div> <div><input type="checkbox"/> TXSYSCLKSEL</div> <div><input type="checkbox"/> TXRATE</div> <div><input checked="" type="checkbox"/> TXBUFSTATUS</div> <div><input type="checkbox"/> TX8B10BEN</div> <div><input checked="" type="checkbox"/> RXPCRESET</div> <div><input checked="" type="checkbox"/> RXPMARESET</div> <div><input type="checkbox"/> RXSYSCLKSEL</div> <div><input type="checkbox"/> RXRATE</div> <div><input checked="" type="checkbox"/> RXBUFSTATUS</div> </div>						

Step5: 设置 K 码为 K28.5, 就是我们后续编代码所使用的 bc, K 码用来修正数据对齐, 代码部分有详细解释, 其他的默认。(K28.5 解码后就是 BC, 为什么是 BC 可以看前面 8b10b 原理介绍部分)

GT Selection	Line Rate, RefClk Selection	Encoding and Clocking	Comma Alignment and Equalization	PCIe, SATA, PRBS	CB and CC Sequence	Summary
RXCOMMA Alignment						
RX COMMA detection						
<input checked="" type="checkbox"/> Use comma detection		Comma Value	K28.5	Comma Mask	1111111111	
<input type="checkbox"/> Decode valid comma only		Plus Comma	0101111100	Align to	Two Byte Boundaries	
<input type="checkbox"/> Combine plus/minus commas (double-length comma)		Minus Comma	1010000011			
Optional Ports						
<input checked="" type="checkbox"/> ENPCOMMAALIGN (Enables positive Comma Alignment) <input checked="" type="checkbox"/> ENMCOMMAALIGN (Enables negative Comma Alignment)						
<input type="checkbox"/> RXSLIDE <input checked="" type="checkbox"/> RXBYTEISALIGN <input checked="" type="checkbox"/> RXBYTEREALIGN <input checked="" type="checkbox"/> RXCOMMADET						
Termination and Equalization						
Differential Swing and Emphasis Mode Custom						
RX Equalization						
Mode	LPM-Auto	Voltage	Programmable			
Automatic Gain Control	Auto	Trim Value (mV)	800			
RX Termination						
Optional Ports						
<input checked="" type="checkbox"/> TXPOLARITY <input type="checkbox"/> TXINHIBIT <input checked="" type="checkbox"/> TXDIFFCTRL <input checked="" type="checkbox"/> TXPOSTCURSOR <input checked="" type="checkbox"/> TXPRECURSOR <input checked="" type="checkbox"/> TXMAINCURSOR						
<input type="checkbox"/> TXQPISENN <input type="checkbox"/> TXQPISENP <input type="checkbox"/> TXQPIBIASEN <input type="checkbox"/> TXQPIWEAKPUP <input type="checkbox"/> TXQPISTRONGPDOWN						

Step6: 这页都是高级配置，默认不选

GT Selection	Line Rate, RefClk Selection	Encoding and Clocking	Comma Alignment and Equalization	PCle, SATA, PRBS	CB and CC Sequence	Summary
PCle Express and SATA						
<input type="checkbox"/> Enable PCI Express						
SATA COM sequence						
Bursts <input type="text" value="4"/> [0 - 7] Idles <input type="text" value="4"/> [0 - 7]						
PCI Express Parameters						
Transition Time						
To P2 <input type="text" value="100"/> [0 - 255] From P2 <input type="text" value="60"/> [0 - 4095] To/From Non P2 <input type="text" value="25"/> [0 - 255]						
Optional Ports						
<input checked="" type="checkbox"/> LOOPBACK <input type="checkbox"/> RXCOMWAKEDET <input type="checkbox"/> TXDETECTRX <input type="checkbox"/> RXSTATUS						
<input type="checkbox"/> TXCOMINIT <input type="checkbox"/> TXELECIDLE <input type="checkbox"/> RXVALID <input type="checkbox"/> TXCOMSAS						
<input type="checkbox"/> PHYSTATUS <input type="checkbox"/> RXCOMINITDET <input type="checkbox"/> TXCOMWAKE <input type="checkbox"/> RXCOMSASDET						
<input type="checkbox"/> TXCOMFINISH <input checked="" type="checkbox"/> TXPOWERDOWN <input checked="" type="checkbox"/> RXPOWERDOWN						
OOB signalling and PRBS						
<input type="checkbox"/> Use RX OOB Signal Detection						
PRBS						
<input checked="" type="checkbox"/> Use PRBS Detector <input checked="" type="checkbox"/> Use Port TXPRBSSEL <input checked="" type="checkbox"/> Use Port TXPRBSFORCEERR <input type="checkbox"/> RXPRBS_LOOPBACK						

Step7: CB 和 CC 顺序

GT Selection	Line Rate, RefClk Selection	Encoding and Clocking	Comma Alignment and Equalization	PCIe, SATA, PRBS	CB and CC Sequence	Summary
--------------	-----------------------------	-----------------------	----------------------------------	------------------	--------------------	---------

Channel Bonding

☐ Use Channel Bonding ☐ Use Two Channel Bonding Sequences

Sequence Max Skew: Sequence length:

Clock correction

☒ Use Clock Correction PPM Offset +/-: [-1250 - 1250]

☐ Use Two Clock Correction Sequences Periodicity of the CC sequence (bytes):

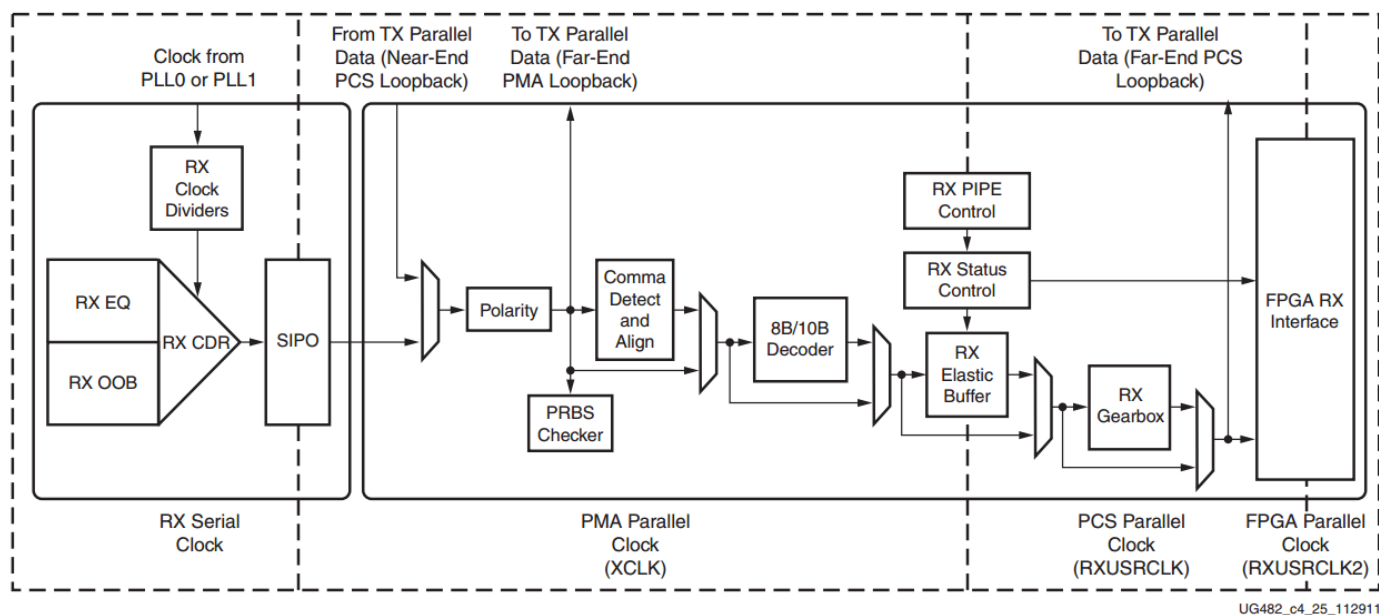
Sequence length:

Sequence Definition

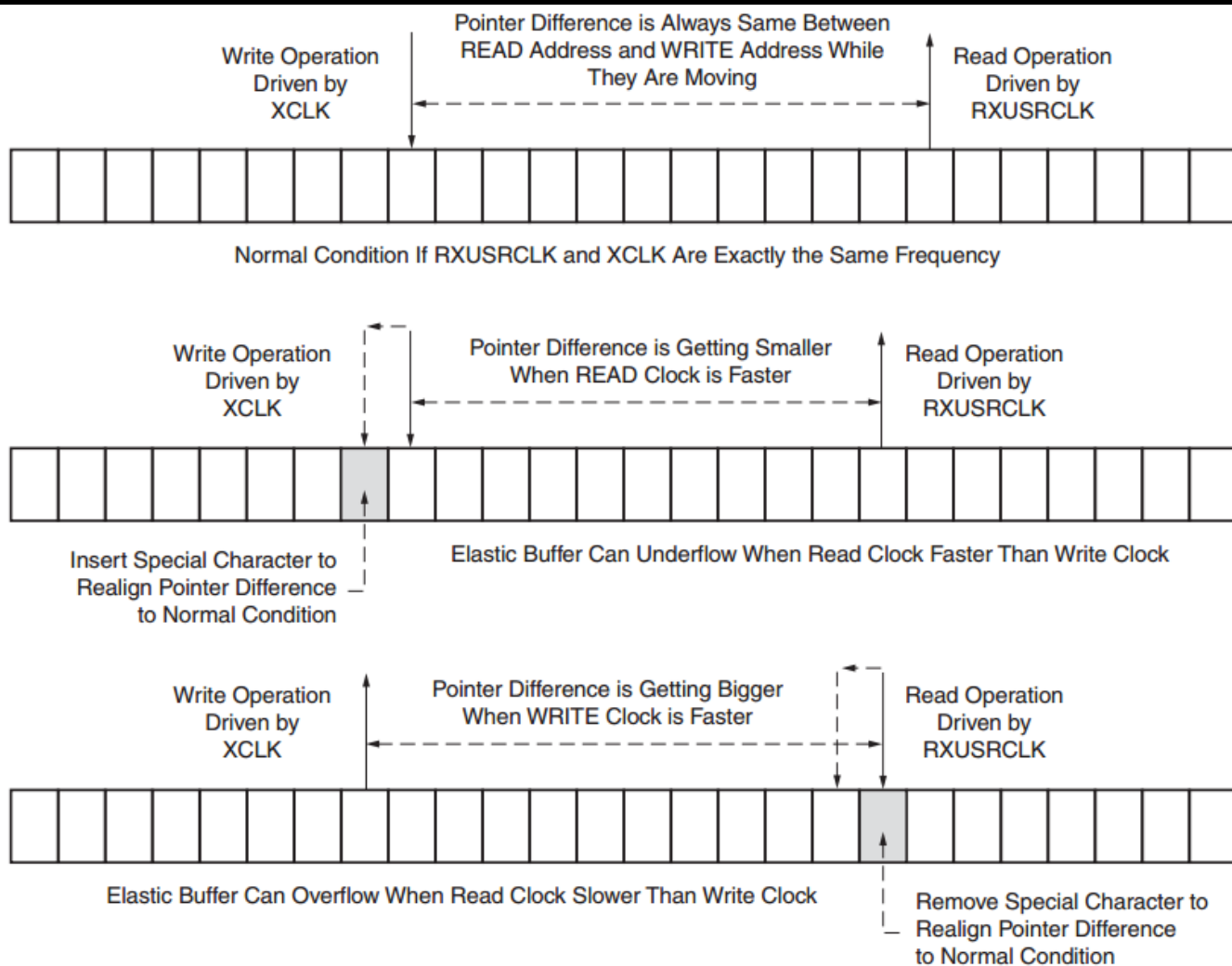
	Sequence	K Character	Inverted Disparity	Don't Care
Sequence1, Byte1	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sequence1, Byte2	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sequence1, Byte3	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sequence1, Byte4	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sequence2, Byte1	00000000	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sequence2, Byte2	00000000	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sequence2, Byte3	00000000	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sequence2, Byte4	00000000	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

选择支持时钟校准，并且设置每 5000 个字节发送一组 CC Seq。因为从数据流中恢复出来的时钟和本地产生的 FIFO 读侧的时钟的频率不可能完全一致，所以才要进行 CC 处理。这个周期是根据发送侧和接收侧的时钟差算出来，然后再根据实际调试结果进行一定的修正。不同的板子不同的环境，这个值都是不同的。

Xilinx 收发器 IP 核支持通道绑定，将多个收发器通道“绑定”成一个速率更高的传输通道，利用 FIFO 消除其间的延时不确定性。Clock correction 是最后一个重要的点。先来看看 RX 通道的结构和弹性缓存概念。



接收通道中同样有两个时钟域：从 CDR 恢复出的 XCLK 和接收通道工作时钟 RXUSRCLK。RX 通道使用 RX Elastic Buffer 来桥接两时钟域，但由于两者细微的差异会使缓存变空或溢出。为此引入时钟矫正，在发送端周期性发送一些特殊字符，接收端在弹性缓存快满时删除这些字符，快空时复制这些字符从而保证缓存内数据维持动态平衡的状态。



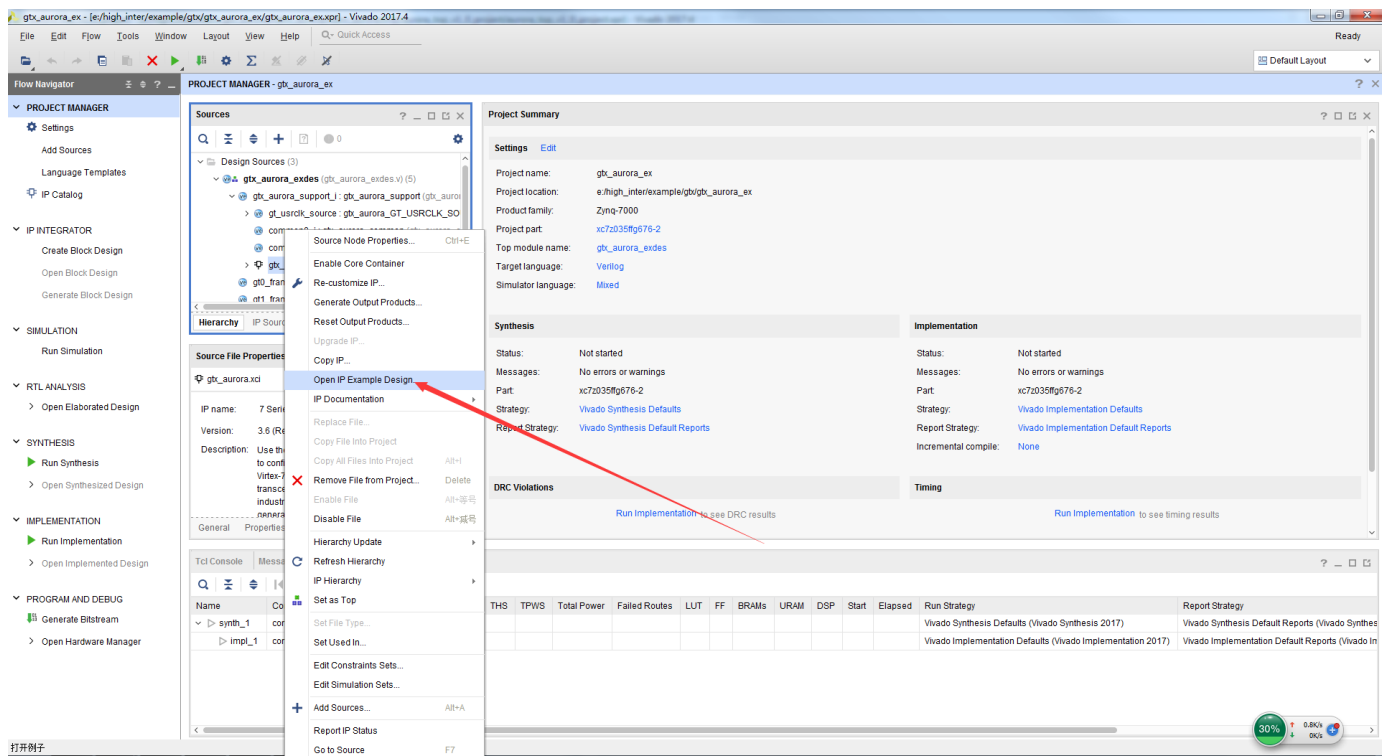
UG482_c4_26_071612

所以，GTP 的发送时钟和 GTP 的接收后的恢复时钟，会存在一定的动态变化，这就是前面我们设置 TXUSRCLK 和 RXUSRCLK 的时候必须要考虑的问题，比如增加 FIFO，解决跨时钟的问题。

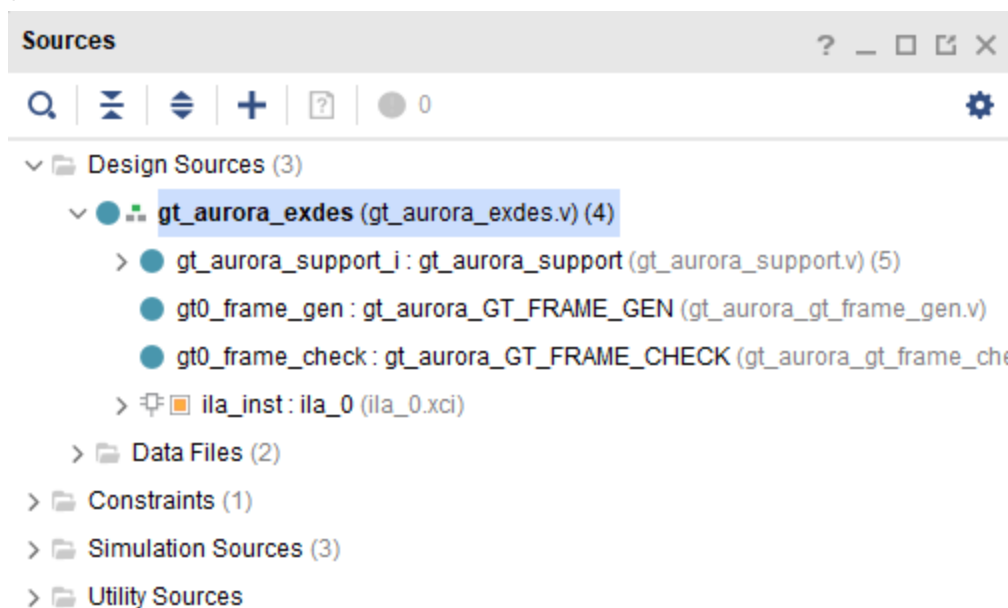
Step8: 可以看到 TXUSRCLK 和 RXUSRCLK 频率为 250M, TXUSRCLK2 和 RXUSRCLK2 为 125M,也就是我们将来写代码要使用的时钟,和前面我们计算的一致。

Component Name gtx_aurora	
GT Selection	Line Rate, RefClk Selection
Encoding and Clocking	Comma Alignment and Equalization
PCle, SATA, PRBS	CB and CC Sequence
Summary	
Summary	
Features	GT
Protocol File	aurora_8b10b_single_lane_4byte
TX Line Rate(Gbps)	5.000
TX reference clock(MHz)	125.000
Encoding	8B/10B
TX Internal Data width	20
TX External Data width	32
TXUSRCLK(MHz)	250.0
TXUSRCLK2(MHz)	125.0
TX Buffer Enabled	true
RX Line Rate(Gbps)	5.000
RX reference clock(MHz)	125.000
Decoding	8B/10B
RX Internal Data width	20
RX External Data Width	32
RXUSRCLK(MHz)	250.0
RXUSRCLK2(MHz)	125.0
RX Buffer Enabled	true

Step9: 点击 OK 生成 IP 核，选中生成的 gtp IP 核，右击，选中 open IP example design。



Step10:打开后工程如下图所示，gt_aurora_GT_FRAME_GEN 模块产生要发送的测试帧数据；gt_aurora_GT_FRAME_CHECK 模块检查回环后收到的数据是否正确；



2.2.1 修改 example 工程

由于开发板的 SFP 屏蔽笼的 tx_disable 引脚都默认接了上拉电阻。要使收发回环测试可以正常进行，必须要将 tx_disable 引脚拉低。因此，在 example design 的顶层模块，添加 2 个 sfp_tx_disable 引脚，且均置为 0 即可。如下图所示。

```

)
(
    input wire Q0_CLK1_GTREFCLK_PAD_N_IN,
    input wire Q0_CLK1_GTREFCLK_PAD_P_IN,
    //    input wire  DRP_CLK_IN_P,
    //    input wire  DRP_CLK_IN_N,
    input wire drp_clk,
    input wire GTTX_RESET_IN,
    input wire GTRX_RESET_IN,
    input wire PLLO_RESET_IN,
    input wire PLL1_RESET_IN,
    //    output wire TRACK_DATA_OUT,
    input wire      RXN_IN,
    input wire      RXP_IN,
    output wire      TXN_OUT,
    output wire      TXP_OUT,
    output [1:0]      sfp_tx_disable
);

assign sfp_tx_disable = 2'b00;

//***** Register Declarations *****
wire      gt_tx fsmresetdone_i;
wire      gt_rx fsmresetdone_i;
(* ASYNC_REG = "TRUE" *)reg      gt_tx fsmresetdone_r;
(* ASYNC_REG = "TRUE" *)reg      gt_tx fsmresetdone_r2;
wire      gt0 tx fsmresetdone_i;

```

3.将 drp_clk 直接连入 sysclk_in 即可，官方的例子这个时钟是引脚进来的加了 bufg，我们用 PLL 产生即可。修改前如下图

```

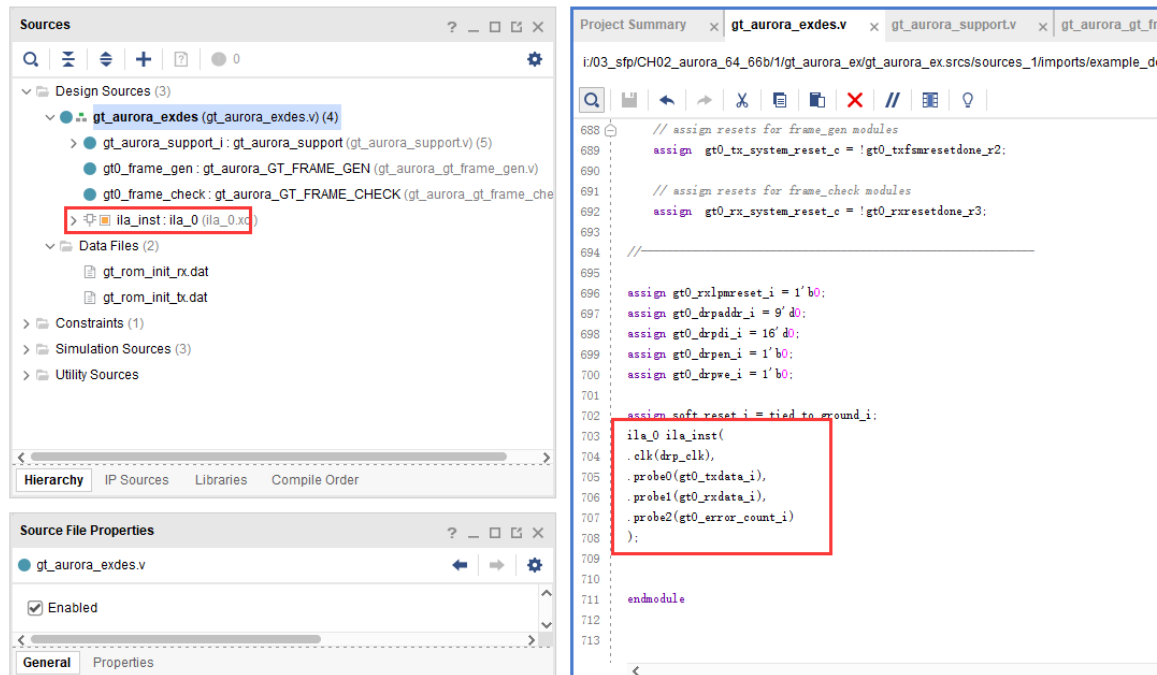
IBUFDS IBUFDS_DRP_CLK
(
    .I (DRP_CLK_IN_P),
    .IB (DRP_CLK_IN_N),
    .O (DRPCLK_IN)
);

BUF8 DRP_CLK_BUF8
(
    .I (DRPCLK_IN),
    .O (drpclk_in_i)
);

```

修改为: assign drpclk_in_i = drp_clk;

Step11: 添加 ila



2.3 编译下载测试

Step1: 单块板子可以直接用 1 根光纤把 RX 和 TX 环路起来，如图所示。

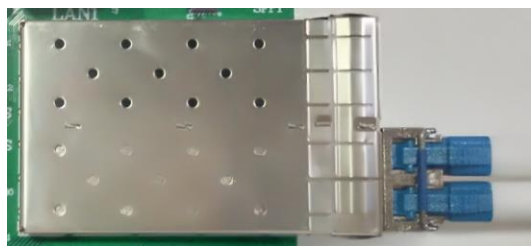


图 1-7-3-1 开发板测试连接图

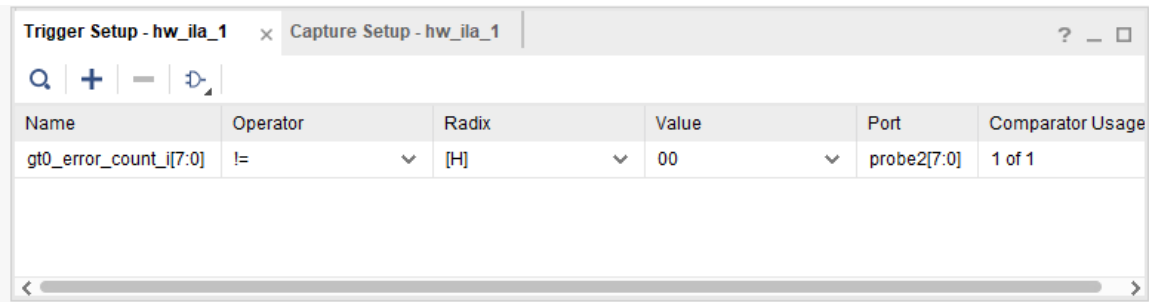
表 1-7-1 可编程时钟模式设置

Prescale Divider	Feedback Divider	PR1/PR0	VCO MHZ	Output Divider	OD2/OD1 /OD0	Output MHZ	Application
4	20	11	2000	4	011	125	GigE



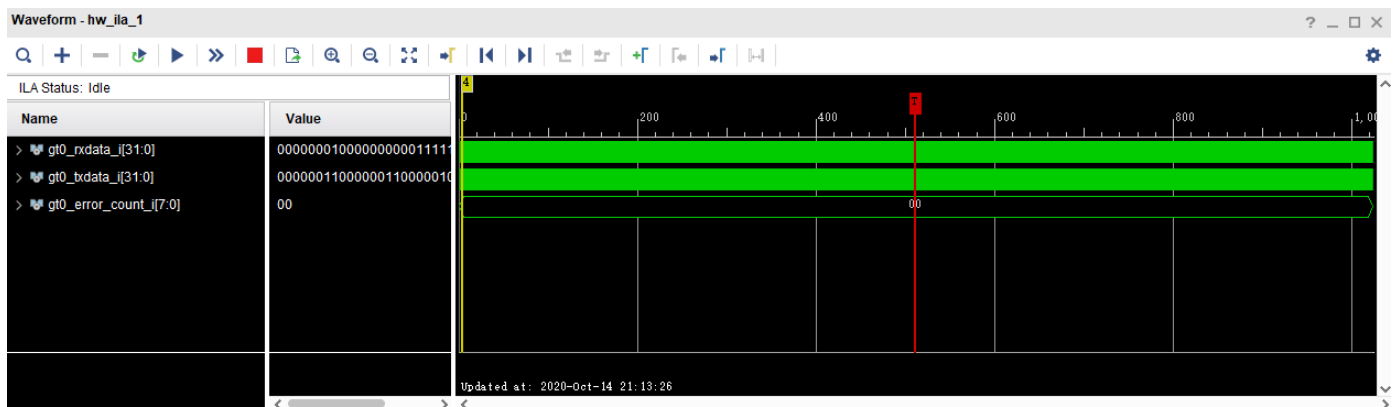
图 1-7-3-2 拨码开关设置

Step2: 设置逻辑分析的触发信号



Step3: 观察发送部分信号和数据

由于数据没有错误，所以需要手动触发以此以此来观察数据



CH03 XILINX 7 系列 GTP 通信之 HDMI 视频传输

软件版本: VIVADO2019.2

操作系统: WIN10 64bit

硬件平台: 适用 XILINX A7/K7/Z7/ZU/KU 系列 FPGA

登录米联客(MSXBO)FPGA 社区-www.uisrc.com 观看免费视频课程、在线答疑解惑!

3.1 概述

8B/10B 编码是 1983 年由 IBM 公司的 Al Widmer 和 PeterFrasaszek 所提出的数据传输编码标准, 目前已经被广泛应用到高速串行总线, 如 IEEE1394b、SATA、PCI-Express、Infini-band、FiberChannel、XAUI、RapidIO、USB 3.0 的美好。8B/10B 编码将待发送的 8 位数据转换成 10 位代码组, 其目的是保证直流平衡, 以及足够密集的电平转换。

本课程讲解“7 Series FPGAs Transceivers Wizard”IP 的使用。测试方案, 利用 HDMI 输入 1080P 视频信号后经光模块回传回来, 再通过 HDMI 输出视频图像, 以此测试光通信功能。

本课程的理论部分摘录自网络文章“8B10B 详解&综述”, 如有侵权请联系本公司。

本课程还给出了开发板到开发板的视频输出传输 demo。

3.2 8B10B 编码原理

以下是理论部分, 学习完成后对于我们后面分析代码还是有所帮助。

在光纤通信中, 线路编码是必要的, 因为电端机输出的数字信号是适合电缆传输的双极性码, 而光源不能发射负脉冲, 只能用光脉冲的“有”和“无”来表示二进制码中的“1”和“0”。该方法虽然简单, 却存在三个问题:

- 1)遇到数字序列中出现长连“0”或长连“1”时, 将给光纤线路上再生中继器和终端光接收机的定时信息提取工作带来困难;
- 2)简单的单极性码中含有直流分量。由于线路上光脉冲中“1”和“0”是随机变化的, 这将导致单极性码的直流成分也作随机性的变化。这种随机性变化的直流成分, 可以通过光接收机的交流耦合电路引起数字信号的基线漂移, 给数字信号的判决和再生带来困难;
- 3)不能实现不中断通信业务下的误码检测;

为解决以上问题, 通常对于由电端机输出的信号码流, 在未对 LED(或 LD)调制以前, 一般要先进行码型变换使调制后的光脉冲码流由简单的单极性码, 转换为适合于数字光纤传输系统传输的线路码。适合于光纤通信的线路码型有多种, 但都要满足以下要求:

- a)能保证比特序列独特性。
- b)能提供足够的定时信息。

由于在光纤数字传输系统的传输中, 只传送信码, 而不传送时钟, 因此在接收端, 必须从收到的码流中提取出定时信息, 以利于上述的定时提取。必须限制线路码流中同符号连续数不能过大, 也就是说, 应避免长连“0”及长连“1”的出现, 提高电平跳变的密度, 使定时提取较为简单。

- c)减少功率密度中的高低频分量。

线路码的功率谱密度中的低频分量是由码流中的“0”、“1”分布状态来决定的, 低频分量小, 说明“0”、“1”分布比较均匀, 直流电平比较恒定, 也就是信号基线浮动小, 有利于接收端判决电路的正常工作。高频分量是由线路码的速率决定的, 这在带宽(色散)限制系统中特别值得注意, 在这种系统中, 中继距离主要由光纤线路的总带宽(总色散)决定, 如果线路码速率提高的太多, 会使中继距离大大缩短。

- d)要有利于减少码流的基线漂移, 即要求码流中的“1”、“0”码分布均匀, 否则不利于接收端的再生判决
- e)码率增加要少, 光功率代价要低。

- f)接收端将线路码还原后, 误码增殖要小。

线路传输中发生的一个误码, 往往使接收端的解码(反变换)发生多个错误, 这就是误码倍增, 也叫做误码扩展或误码增值。由于误码倍增, 使光接收机要达到原要求的误码性能指标, 必须付出光功率代价, 即光接收机灵敏度劣化。因此误码倍增系数越小越好。

g)能提供适当的冗余度。

h)低的对称抖动。

传输的比特序列必须保持低的码型相关的抖动。

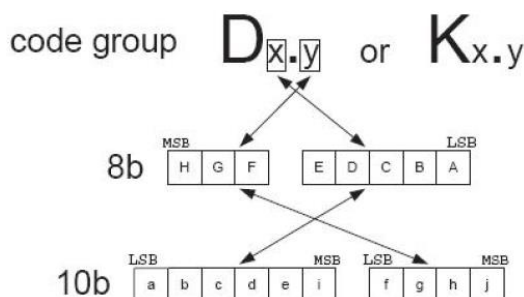
i)易于实现。

数字光纤通信系统中常用的线路码型有:加扰二进制、 插入比特码和 mBnB 码。

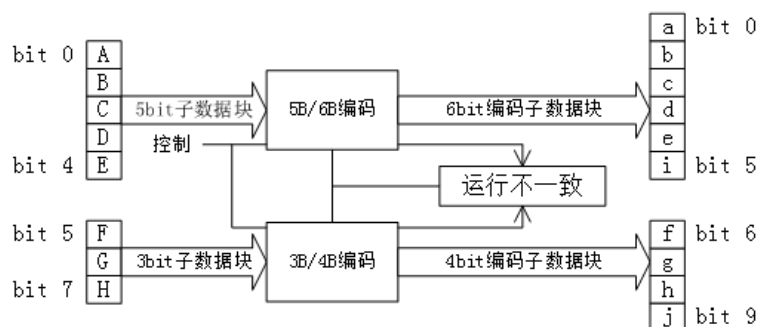
8B/10B 编码最初由 IBM 公司的 Albert X.Widemer 和 Peter A.Franaszek 发明,并应用于 ESCON(200M 互连系统)中。它是 mBnB 编码中的一个特例。

8B/10B 编码方法是把 8bit 代码组合编码成 10bit 代码,代码组合包含 256 个数据字符编码和 12 个控制字符编码,分别记为 $D_{x.y}$ 和 $K_{x.y}$ 。通过仔细选择编码方法可以获得不同的优化特性。这些特性包括满足串行/解串行器功能必须的变换: 确保“0”码元与“1”码元个数的一致, 又称为直流均衡; 确保字节同步易于实现(在一个比特流中找到字节的起始位); 以及对误码率有足够的容忍能力和降低设计复杂度。

8B/10B 编码方案是把 8bit 数据分成 2 个子分组: 3 个最高有效位(y)和 5 个最低有效位(x)。代码字按顺序排列,从最高有效位到最低有效位分别记为 H、G、F 和 E、D、C、B、A。3bit 的子分组编码成 4 bit, 记为 j、h、g、f; 5 bit 的子分组编码成 6bit, 记为 i、e、d、c、b、a, 其映射关系如下图所示, 4bit 和 6bit 的子分组再组合成 10bit 的编码值。



编码时, 低 5bit 原数据 EDCBA 经过 5B/6B 编码成为 6bit 码 abcdei, 高 3bit 原数据 HGF 经 3B/4B 成为 4bit 码 fghj, 最后再将两部分组合起来形成一个 10bit 码 abcdeifghj。10B 码在发送时, 按照先发送低位在发送高位的顺序发送。



将 8bit 数据分成 3bit 和 5bit 两组, 分别对应 10bit 中的 4bit 和 6bit, 直流平衡代码的不平衡度就是通过“0”的个数减去“1”的个数来计算得到的。如果 4bit 和 6bit 的各分组中“0”和“1”的个数相等, 称为完美平衡代码, 或称为完美的直流平衡代码, 无需补偿, 但是这种情况是不可能的。因为在 4bit 的子分组中, 16 种编码中只有 6 种是完美平衡的, 这对于 3bit 的 8 种编码值是不够的。同时, 在 6bit 的子分组中也只有 20 种编码是完美平衡的, 对于 5bit 的 32 种编码值也是不够的。由于 4 bit 和 6bit 的两个子分组都是偶数个位数, 而不平衡度不可能是“+1”或“-1”, 因此, 在 8B/10B 编码方案中还要使用不平衡度为“+2”和“-2”的值。在编码过程中, 用一个极性偏差(running disparity, RD)参数表示不平衡度, 在不平衡时用 2 个 10 bit 字符表示一个 8 位字符, 其中一个称为 RD-, 表示“1”的个数比“0”的个数多 2 个, 另一个称为 RD+, 表示“0”的个数比“1”的个数多 2 个。这种不平衡差值为 2 的数需要采用 2 个 10bit 表示 1 个 8bit, 主要用于 K 码的控制字符。

Rules for Running Disparity		
Previous RD	Disparity of 6 or 4 Bit Code	Next RD
-1	-2	not used (choose +2 encoding instead)
-1	0	-1
-1	+2	+1
+1	-2	-1
+1	0	+1
+1	+2	not used (choose -2 encoding instead)

8B/10B 编码中将 K28.1、K28.5 和 K28.7 作为 K 码的控制字符，称为“comma”。在任意数据组合中，comma 只作为控制字符出现，而在数据负荷部分不会出现，因此可以用 comma 字符指示帧的开始和结束标志，或始终修正和数据流对齐的控制字符。

5B/6B 编码和 3B/4B 编码的映射有标准化的表格，可以通过基于查找表的方式实现。使用“不一致性 (Disparity)”来描述编码中“1”的位数和“0”的位数的差值，它仅允许有“+2”(“0”比“1”多两个)、“0”(“0”与“1”个数相等)以及“-2”(“1”比“0”多两个)这三种状况。由于数据流不停地从发送端向接收端传输，前面所有已发送数据的不一致性累积产生的状态被称为“运行不一致性 (Running Disparity, RD)”。RD 仅会出现+1 与-1 两种状态，分别代表位“1”比位“0”多或位“0”比位“1”多，其初始值是-1。Next RD 值依赖于 Current RD 以及当前 6B 码或者 4B 码的 Disparity。根据 Current RD 的值，决定 5B/4B 和 3B/4B 编码映射方式，如下图所示。

5b/6b code							
Input		RD = -1	RD = +1	Input		RD = -1	RD = +1
	EDCBA	abcdei			EDCBA	abcdei	
D.00	00000	100111	011000	D.16	10000	011011	100100
D.01	00001	011101	100010	D.17	10001	100011	
D.02	00010	101101	010010	D.18	10010	010011	
D.03	00011	110001		D.19	10011	110010	
D.04	00100	110101	001010	D.20	10100	001011	
D.05	00101	101001		D.21	10101	101010	
D.06	00110	011001		D.22	10110	011010	
D.07	00111	111000	000111	D.23 †	10111	111010	000101
D.08	01000	111001	000110	D.24	11000	110011	001100
D.09	01001	100101		D.25	11001	100110	
D.10	01010	010101		D.26	11010	010110	
D.11	01011	110100		D.27 †	11011	110110	001001
D.12	01100	001101		D.28	11100	001110	
D.13	01101	101100		D.29 †	11101	101110	010001
D.14	01110	011100		D.30 †	11110	011110	100001
D.15	01111	010111	101000	D.31	11111	101011	010100
				K.28	11100	001111	110000

3b/4b code

input		RD = -1	RD = +1	input		RD = -1	RD = +1
	HGF	fghj			HGF	fghj	
D.x.0	000	1011	0100	K.x.0	000	1011	0100
D.x.1	001	1001		K.x.1 ‡	001	0110	1001
D.x.2	010	0101		K.x.2 ‡	001	1010	0101
D.x.3	011	1100	0011	K.x.3	011	1100	0011
D.x.4	100	1101	0010	K.x.4	100	1101	0010
D.x.5	101	1010		K.x.5 ‡	001	0101	1010
D.x.6	110	0110		K.x.6 ‡	001	1001	0110
D.x.P7 †	111	1110	0001				
D.x.A7 †	111	0111	1000	K.x.7 † ‡	111	0111	1000

Control symbols

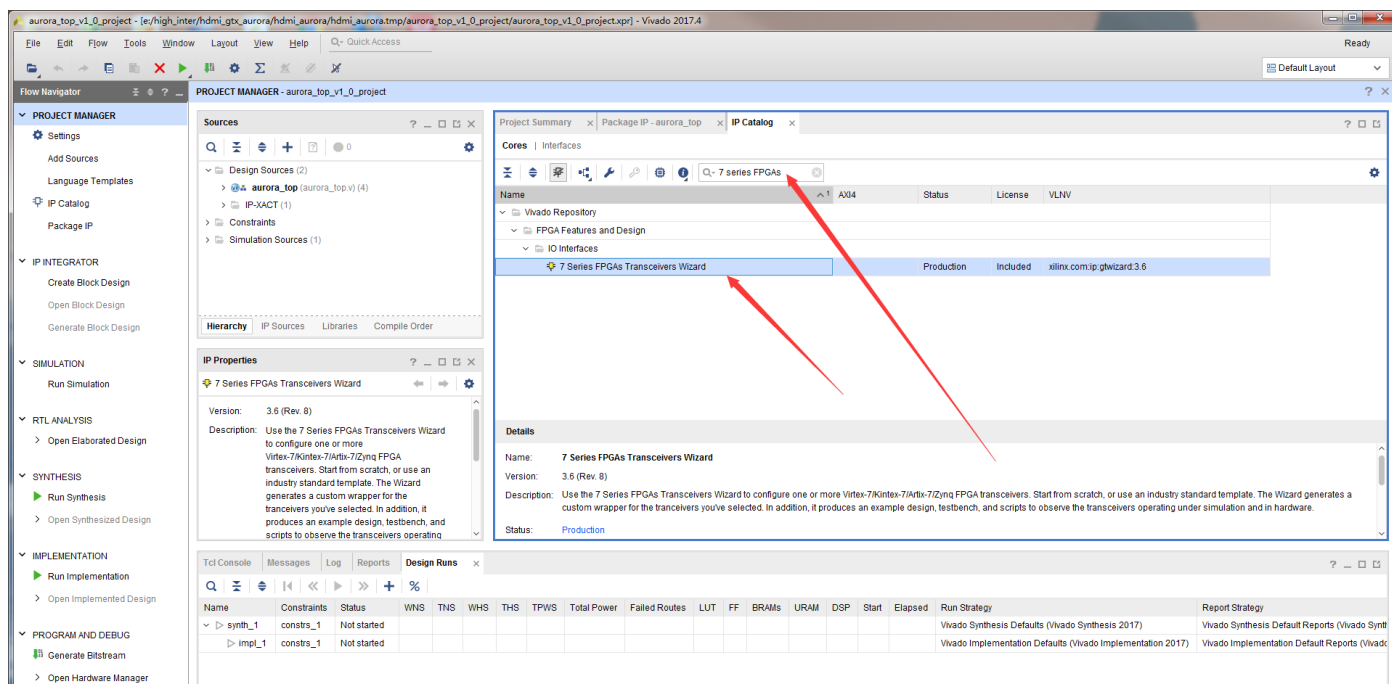
Input				RD = -1	RD = +1
	DEC	HEX	HGF EDCBA	abcdei fghj	abcdei fghj
K.28.0	28	1C	000 11100	001111 0100	110000 1011
K.28.1 †	60	3C	001 11100	001111 1001	110000 0110
K.28.2	92	5C	010 11100	001111 0101	110000 1010
K.28.3	124	7C	011 11100	001111 0011	110000 1100
K.28.4	156	9C	100 11100	001111 0010	110000 1101
K.28.5 †	188	BC	101 11100	001111 1010	110000 0101
K.28.6	220	DC	110 11100	001111 0110	110000 1001
K.28.7 ‡	252	FC	111 11100	001111 1000	110000 0111
K.23.7	247	F7	111 10111	111010 1000	000101 0111
K.27.7	251	FB	111 11011	110110 1000	001001 0111
K.29.7	253	FD	111 11101	101110 1000	010001 0111
K.30.7	254	FE	111 11110	011110 1000	100001 0111

这样，经过 8B/10B 编码以后，连续的“1”和“0”基本上不会超过 5bit，只有在使用 comma 时，才会出现连续的 5 个 0 或 1。接收端的数据解码过程如下图所示：

0101010011011001111000001010110111010010100100111010011001001110100111001					
???	K28.5	D16.2	D31.3	D11.3	D0.0

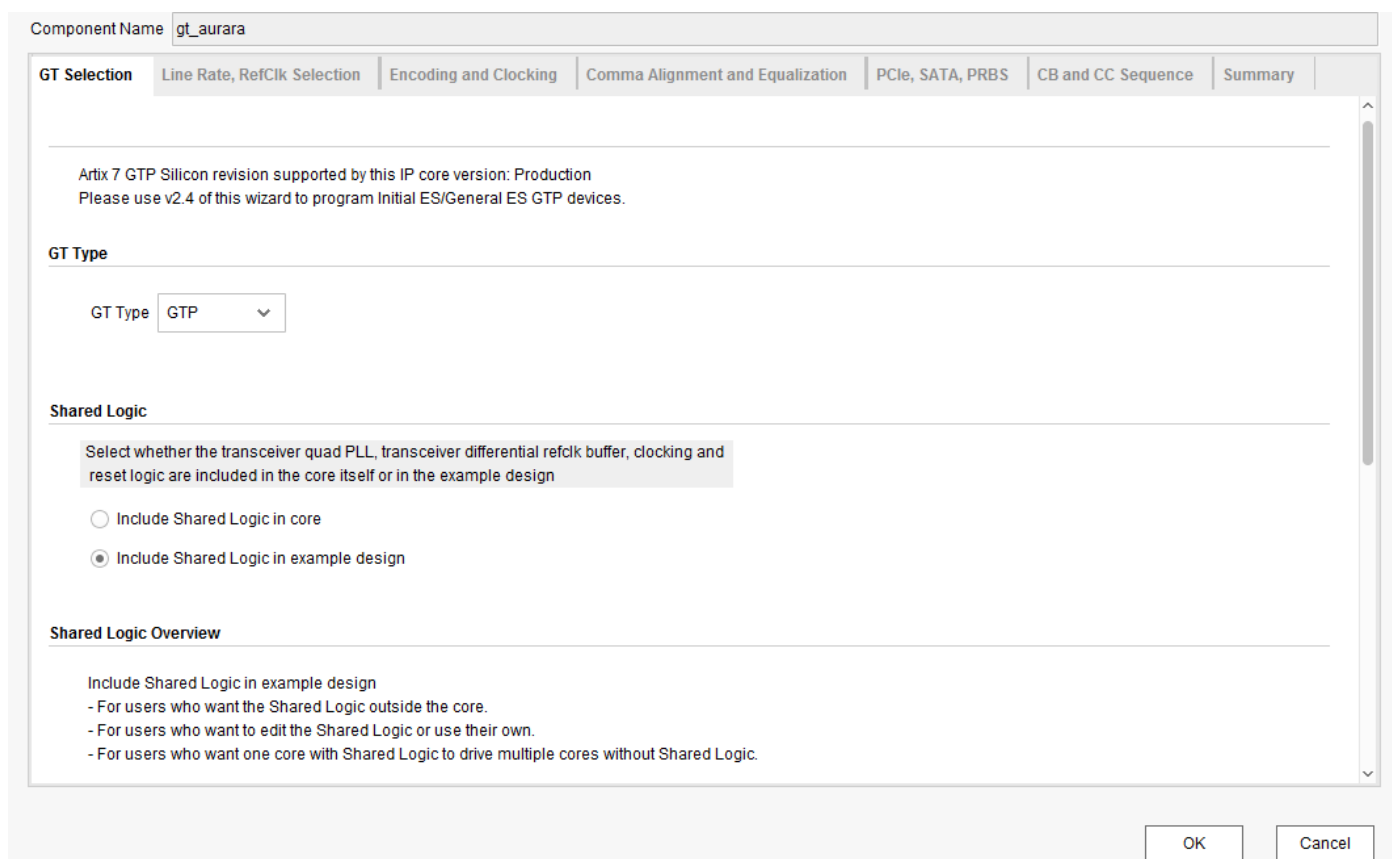
3.3 gt IP 核设置

好了，现在开始我们的课程实验的具体内容部分，完成利用光纤实现视频图像的传输方案。
打开 vivado 在 IP 中搜索 7 series FPGAs，找到 7 Series FPGAs Transceivers Wizard 如图所示：

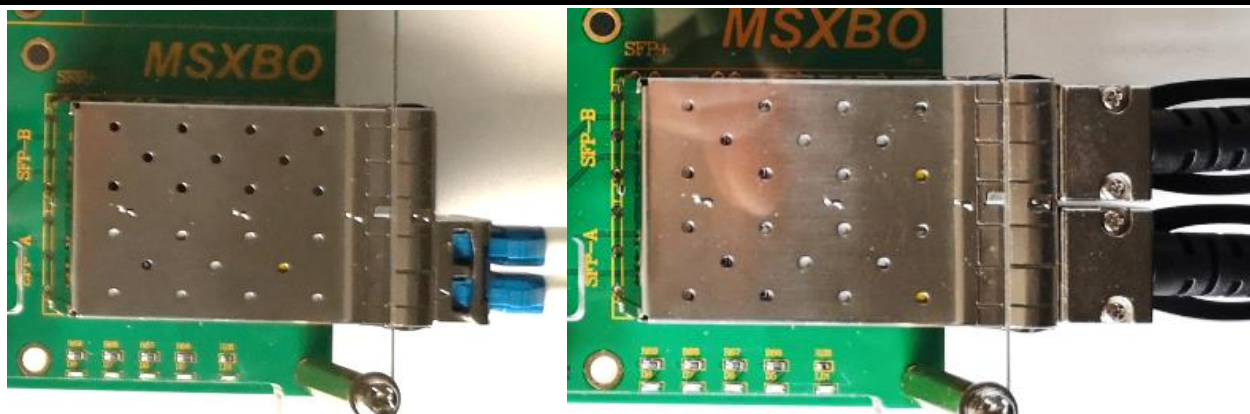


双击打开 gt ip 核，按如下步骤设置：

1：第一页设置：默认即可

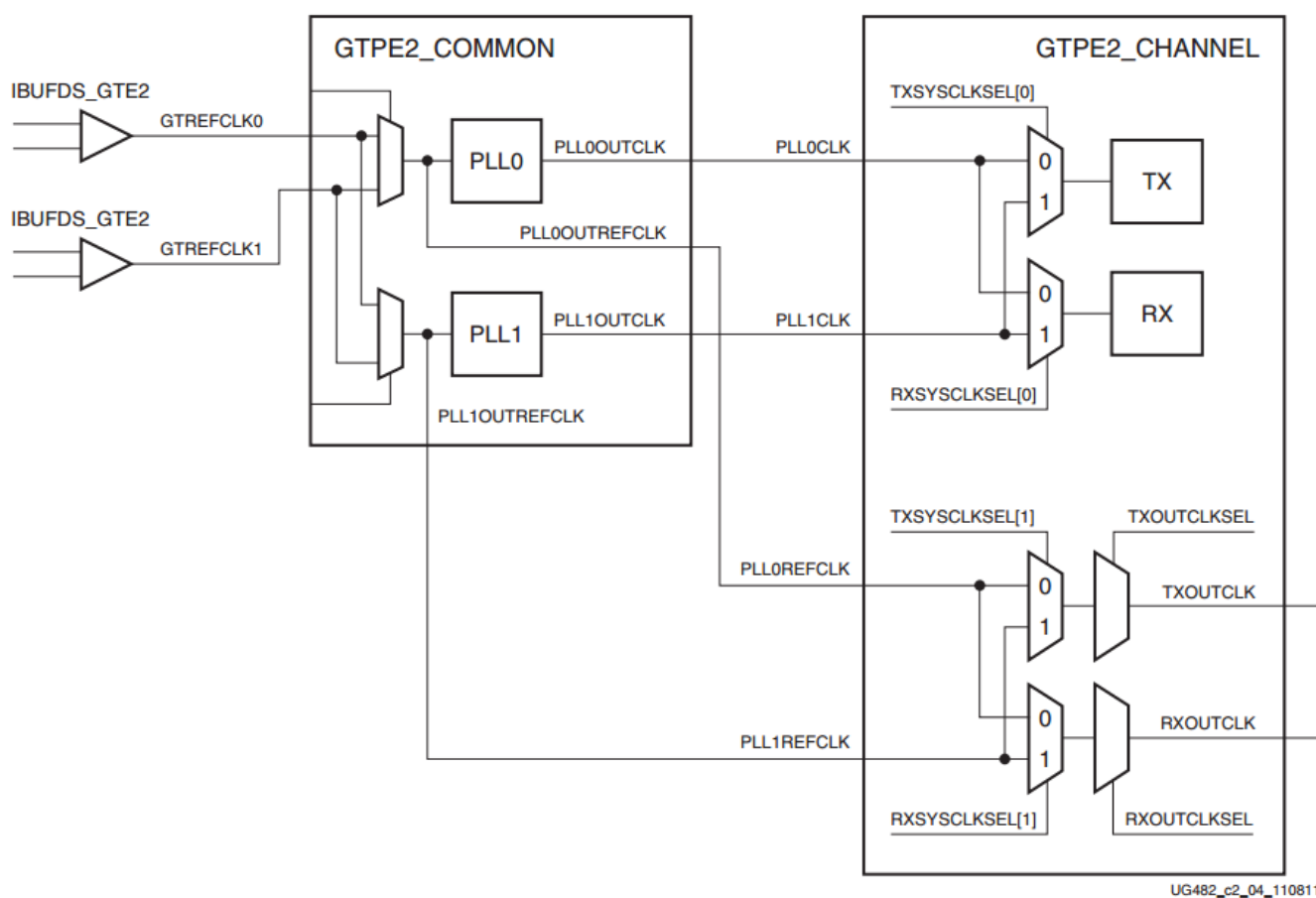


2：第二页设置：红色箭头所指，首先设置 Protocol 为 aurora 8b10b single lane 4 byte ,也就是对外接口为 32 位；设置 Line Rate 为 5G，参考时钟为 125，这个是来源于板子的差分晶振，左下角黄色的地方就是我们用到的高速收发器。以下内容对于不同的开发板设置不一样，以 MZ703FA 为例，MZ703FA 有 2 个 SFP 接口分别是位于 GTP_X0Y0 和 GTP_X0Y1。利用光纤环路测试，可以使用单头的把 RX TX 利用光纤短接直接环路，也可以用 2 个带光模块的光纤环路。左右的为单头光模块，右边的为双头光模块。



3: 速率、参考时钟设置:

Protocol 里面有多种协议可以提供选择, 我们这里选择 Aurora 8b10b single lane 4byte,收发器是独立的, 可以选择不同的编码和速度, 对于 GTP 收发器, 最大是 6.6Gbps,我们这里统一选择 5Gbps。这里的参考时钟必须和开发板上的时钟一致, 否则会无法通信。



GTP 有两个参考时钟输入端口, 经差分-单端转换后通过两个 PLL 产生收发器发送和接收时钟。若 TX 和 RX 线速率一致使用同一个 PLL 产生时钟, 否则需要使用两个不同的 PLL。开发板中差分晶振连接 GTPREFCLK1, 且收发速率相同, 故 PLL Selection TX 和 RX 均选择 PLL0。由于 MA703 开发板上只有 2 个 SFP 接口, 分别定义到了 GTP_X0Y0 和 GTP_X0Y1, 因此对于使用 1 个光模块的配置, 选择 GTP_X0Y0, 对于使用 2 个光模块的配置为选择 GTP_X0Y1, GTP_X0Y1。

为了测试方便, 我们给出了 2 种光模块的配置方法:

单头光模块设置, 只使用 1 个通道进行环路

Re-customize IP

7 Series FPGAs Transceivers Wizard (3.6)

Documentation IP Location Switch to Defaults

Component Name: gt_aurara

GT Selection Line Rate, RefClk Selection Encoding and Clocking Comma Alignment and Equalization PCIe, SATA, PRBS CB and CC Sequence Summary

Protocol: aurora 8b10b single lane 4byte

TX

Line Rate (Gbps): 5 [0.5 - 6.25] ☐ TX off

Reference Clock (MHz): 125.000 Range: 60..660

gt row: Bottom Row ☐ Use Common DRP

RX

Line Rate (Gbps): 5 [0.5 - 6.25] ☐ RX off

Reference Clock (MHz): 125.000 Range: 60..660

PLL Selection

TX: PLL0 RX: PLL0 ☐ Extend reset to 3 ms

Transceiver Selection

☒ Use GTP X0Y0

TX Clock Source: REFCLK1 Q0

RX Clock Source: REFCLK1 Q0

☐ Advanced Clocking Option

☐ PRBS pattern generator and checker

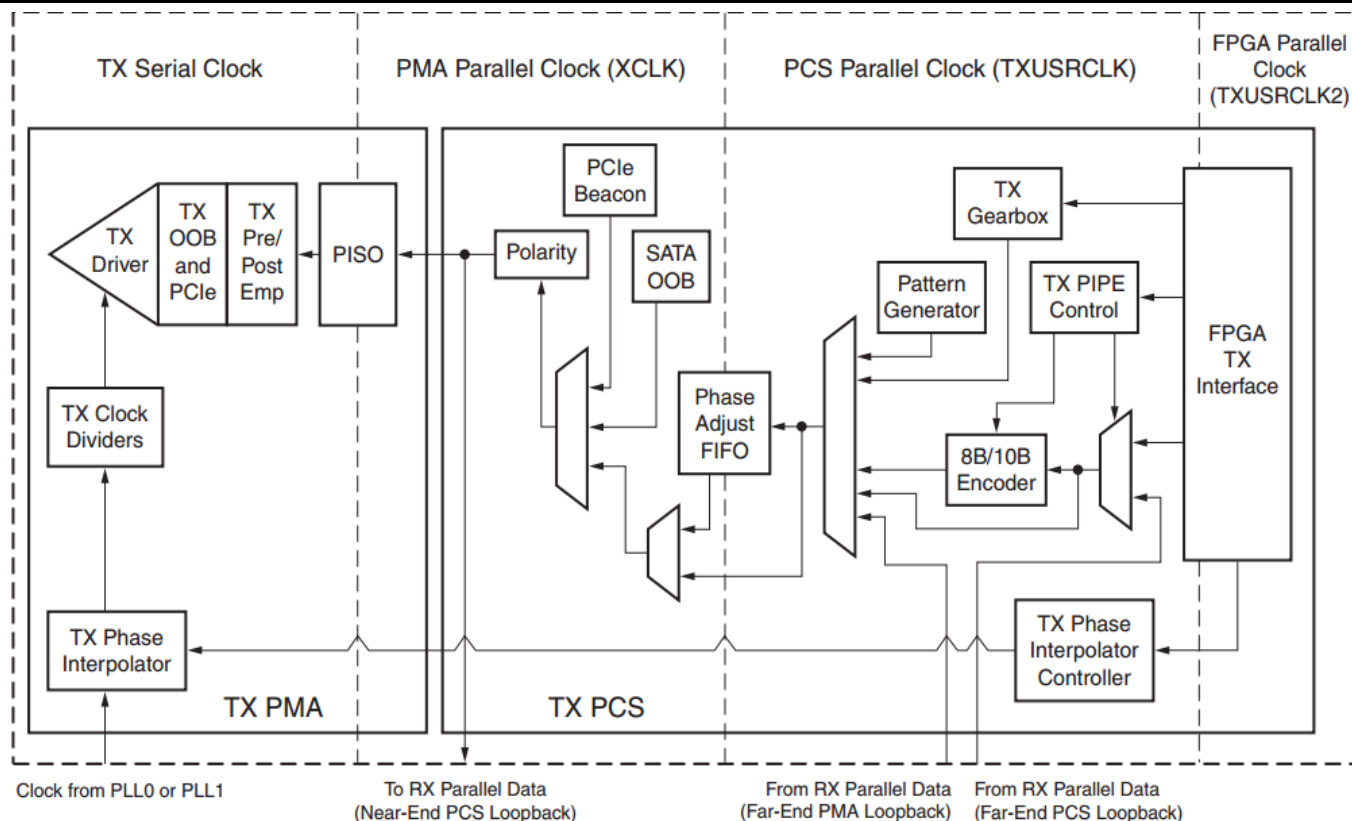
☐ Vivado Lab Tools

Active Transceivers = 1

OK Cancel

3: 页将内部数据宽度设置为 20(16bit 数据利用 8b10b 需要 20bit 表示), 2 个内部数据拼接为 1 个 32bit 外部数据, 通过计算我们可以得到内部时钟为 $5 \times 10^3 / 20 = 250\text{M}$, 而外部时钟为 $5 \times 10^3 / (20 \times 2) = 125\text{M}$ 。编码方式采样 8B/10B 编码, 这种编码方式最主要的目的是“直流平衡”, 即根据特定的编码表实现数据传输过程中比特“0”和比特“1”的数量基本一致, 且减少连 0 和连 1 的情况。编码后的数据流具有较多的跳变, 有助于接收端时钟数据恢复(CDR)。DRP/System Clock Frequency 是动态重配置或系统工作时钟, 通过 DPR 可以让设计者根据所选线速率和定义的协议实时调整收发器参数, 这个比较复杂我们用不到, 不加说明。DRP 时钟选择 100MHz, 可通过外部 PLL IP 核产生, DRP 时钟主要用来初始化一些 GTP 控制器的信号或者参数。

TXUSERCLK 和 RXUSERCLK 时钟的选择概念必要重要, 一般 TXUSERCLK 可以直接选择 TXOUTCLK 作为用户时钟, 而 RXUSERCLK, 可以选择 TXOUTCLK 也可以选择 RXUSERCLK, 还可以选择 RXPLLREFCLK 作为用户时钟。这里就涉及到一个同步的概念, 如果时钟不一致必须设置 TX 和 RX 的 BUFFER, 默认都是设置的。另外输入 TX 的发送时钟和 RX 的恢复时钟差异交大, 可能会导致 RX BUFFER 的溢出或者读空, 而导致数据出错。比较好的解决办法就是设置 RXUSERCLK 为 RXOUTCLK, 就是利用恢复时钟作为用户时钟, 这样就很好的解决了同步的问题了。



GT Selection	Line Rate, RefClk Selection	Encoding and Clocking	Comma Alignment and Equalization	PCIe, SATA, PRBS	CB and CC Sequence	Summary
<div> <div>TX</div> <div>RX</div> </div>						
<div> <div>External Data Width (Bits) 32</div> <div>External Data Width (Bits) 32</div> </div>						
<div> <div>Encoding 8B/10B</div> <div>Decoding 8B/10B</div> </div>						
<div> <div>Internal Data Width (Bits) 20</div> <div>Internal Data Width (Bits) 20</div> </div>						
<div> <div><input checked="" type="checkbox"/> Use DRP</div> <div>DRP/System Clock Frequency (MHz) 100</div> </div>						
<div>Optional Ports</div> <div> <div><input type="checkbox"/> TXBYPASS8B10B</div> <div><input checked="" type="checkbox"/> TXCHARDISPMODE</div> <div><input checked="" type="checkbox"/> TXCHARDISPVAL</div> <div><input checked="" type="checkbox"/> RXCHARISCOMMA</div> <div><input checked="" type="checkbox"/> RXCHARISK</div> <div><input type="checkbox"/> RXSTARTOFSEQ</div> </div>						
<div>Synchronization and Clocking</div> <div> <div>TX</div> <div>RX</div> </div>						
<div> <div><input checked="" type="checkbox"/> Enable TX Buffer</div> <div><input checked="" type="checkbox"/> Enable RX Buffer</div> </div>						
<div> <div>TXUSRCLK Source TXOUTCLK</div> <div>RX Buffer Bypass Mode Auto</div> </div>						
<div> <div>TXOUTCLK Source <input type="checkbox"/> Use TXPLLREFCLK</div> <div>RXUSRCLK Source RXOUTCLK</div> </div>						
<div> <div>RXOUTCLK Source <input type="checkbox"/> Use RXPLLREFCLK</div> </div>						
<div>Optional Ports</div> <div> <div><input checked="" type="checkbox"/> TXPCSRRESET</div> <div><input checked="" type="checkbox"/> TXPMARESET</div> <div><input type="checkbox"/> TXSYSCLKSEL</div> <div><input type="checkbox"/> TXRATE</div> <div><input checked="" type="checkbox"/> TXBUFSTATUS</div> <div><input type="checkbox"/> TX8B10BEN</div> <div><input checked="" type="checkbox"/> RXPCSRRESET</div> <div><input checked="" type="checkbox"/> RXPMARESET</div> <div><input type="checkbox"/> RXSYSCLKSEL</div> <div><input type="checkbox"/> RXRATE</div> <div><input checked="" type="checkbox"/> RXBUFSTATUS</div> </div>						

4: 设置 K 码为 K28.5, 就是我们后续编代码所使用的 bc, K 码用来修正数据对齐, 代码部分有详细解释, 其他的默认。(K28.5 解码后就是 BC, 为什么是 BC 可以看前面 8b10b 原理介绍部分)

GT Selection	Line Rate, RefClk Selection	Encoding and Clocking	Comma Alignment and Equalization	PCIe, SATA, PRBS	CB and CC Sequence	Summary
RXCOMMA Alignment						
RX COMMA detection						
<input checked="" type="checkbox"/> Use comma detection		Comma Value	K28.5	Comma Mask	1111111111	
<input type="checkbox"/> Decode valid comma only		Plus Comma	0101111100	Align to	Two Byte Boundaries	
<input type="checkbox"/> Combine plus/minus commas (double-length comma)		Minus Comma	1010000011			
Optional Ports						
<input checked="" type="checkbox"/> ENPCOMMAALIGN (Enables positive Comma Alignment) <input checked="" type="checkbox"/> ENMCOMMAALIGN (Enables negative Comma Alignment)						
<input type="checkbox"/> RXSLIDE <input checked="" type="checkbox"/> RXBYTEISALIGN <input checked="" type="checkbox"/> RXBYTEREALIGN <input checked="" type="checkbox"/> RXCOMMADET						
Termination and Equalization						
Differential Swing and Emphasis Mode Custom						
RX Equalization						
Mode	LPM-Auto	Voltage	Programmable			
Automatic Gain Control	Auto	Trim Value (mV)	800			
RX Termination						
Optional Ports						
<input checked="" type="checkbox"/> TXPOLARITY <input type="checkbox"/> TXINHIBIT <input checked="" type="checkbox"/> TXDIFFCTRL <input checked="" type="checkbox"/> TXPOSTCURSOR <input checked="" type="checkbox"/> TXPRECURSOR <input checked="" type="checkbox"/> TXMAINCURSOR						
<input type="checkbox"/> TXQPISENN <input type="checkbox"/> TXQPISENP <input type="checkbox"/> TXQPIBIASEN <input type="checkbox"/> TXQPIWEAKPUP <input type="checkbox"/> TXQPISTRONGPDOWN						

5: 这页都是高级配置，默认不选

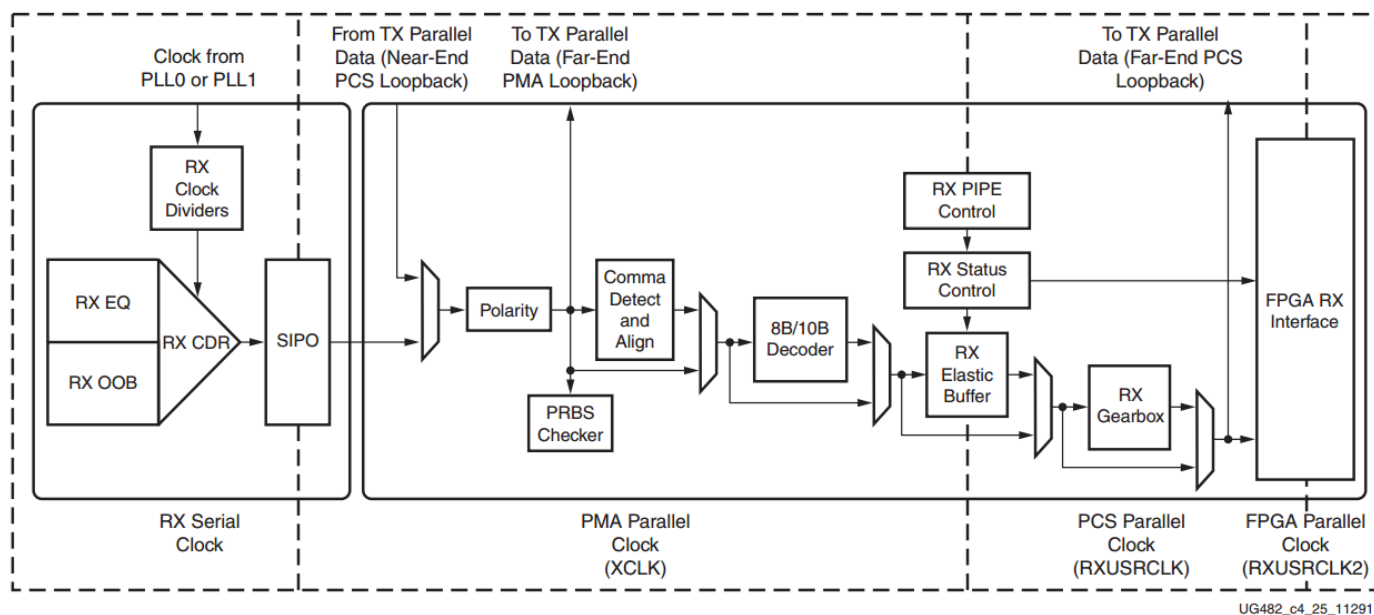
GT Selection	Line Rate, RefClk Selection	Encoding and Clocking	Comma Alignment and Equalization	PCle, SATA, PRBS	CB and CC Sequence	Summary
PCle Express and SATA						
<input type="checkbox"/> Enable PCI Express						
SATA COM sequence						
Bursts <input type="text" value="4"/> [0 - 7] Idles <input type="text" value="4"/> [0 - 7]						
PCI Express Parameters						
Transition Time						
To P2 <input type="text" value="100"/> [0 - 255] From P2 <input type="text" value="60"/> [0 - 4095] To/From Non P2 <input type="text" value="25"/> [0 - 255]						
Optional Ports						
<input checked="" type="checkbox"/> LOOPBACK <input type="checkbox"/> RXCOMWAKEDET <input type="checkbox"/> TXDETECTRX <input type="checkbox"/> RXSTATUS						
<input type="checkbox"/> TXCOMINIT <input type="checkbox"/> TXELECIDLE <input type="checkbox"/> RXVALID <input type="checkbox"/> TXCOMSAS						
<input type="checkbox"/> PHYSTATUS <input type="checkbox"/> RXCOMINITDET <input type="checkbox"/> TXCOMWAKE <input type="checkbox"/> RXCOMSASDET						
<input type="checkbox"/> TXCOMFINISH <input checked="" type="checkbox"/> TXPOWERDOWN <input checked="" type="checkbox"/> RXPOWERDOWN						
OOB signalling and PRBS						
<input type="checkbox"/> Use RX OOB Signal Detection						
PRBS						
<input checked="" type="checkbox"/> Use PRBS Detector <input checked="" type="checkbox"/> Use Port TXPRBSSEL <input checked="" type="checkbox"/> Use Port TXPRBSFORCEERR <input type="checkbox"/> RXPRBS_LOOPBACK						

6: CB 和 CC 顺序

GT Selection	Line Rate, RefClk Selection	Encoding and Clocking	Comma Alignment and Equalization	PCIe, SATA, PRBS	CB and CC Sequence	Summary
Channel Bonding						
<input type="checkbox"/> Use Channel Bonding <input type="checkbox"/> Use Two Channel Bonding Sequences						
Sequence Max Skew		1	Sequence length		1	
Clock correction						
<input checked="" type="checkbox"/> Use Clock Correction <input type="checkbox"/> Use Two Clock Correction Sequences		PPM Offset +/-		100	[-1250 - 1250]	
		Periodicity of the CC sequence (bytes)		5000		
Sequence length		4				
Sequence Definition						
	Sequence	K Character	Inverted Disparity	Don't Care		
Sequence1, Byte1	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Sequence1, Byte2	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Sequence1, Byte3	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Sequence1, Byte4	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Sequence2, Byte1	00000000	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Sequence2, Byte2	00000000	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Sequence2, Byte3	00000000	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Sequence2, Byte4	00000000	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

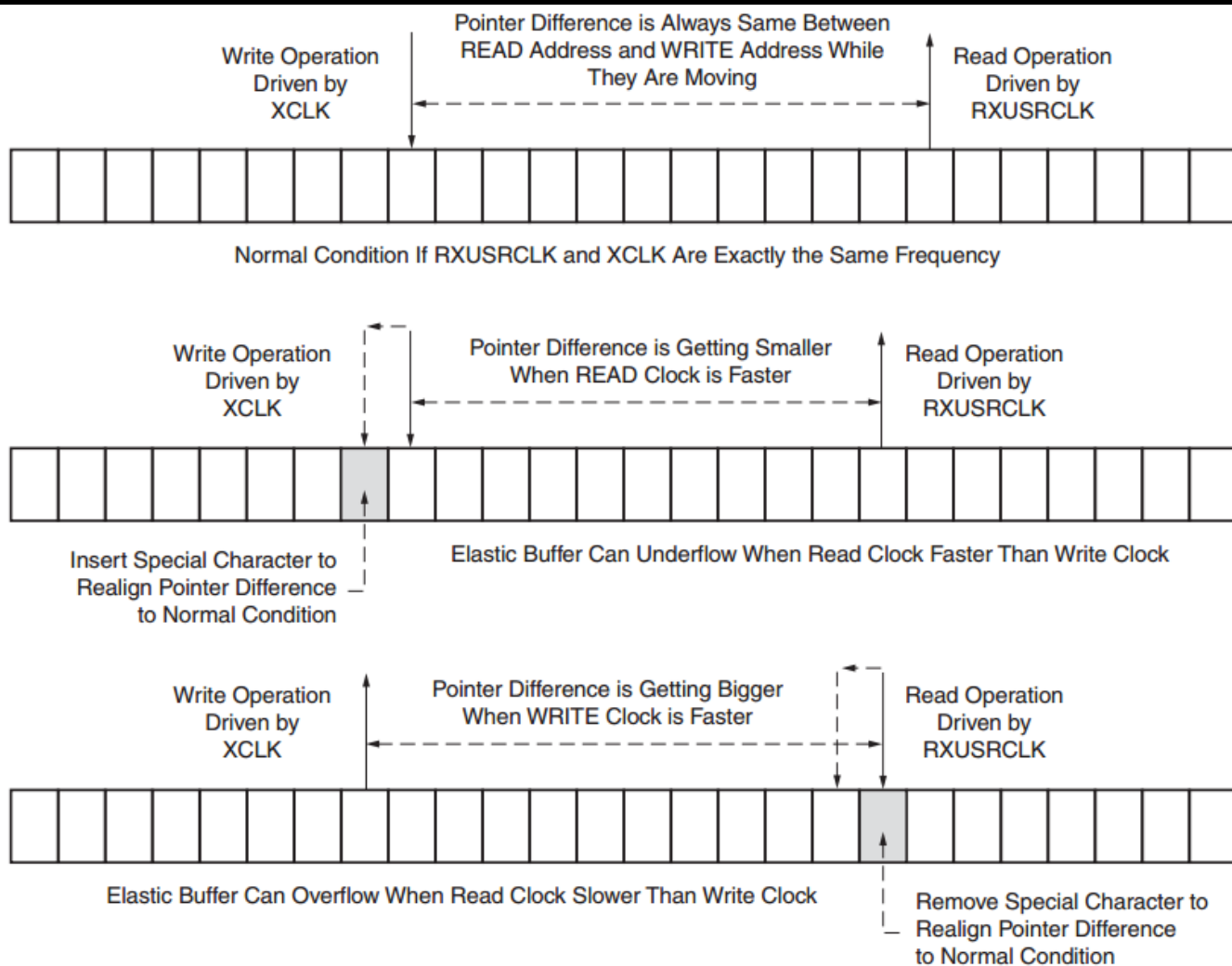
选择支持时钟校准，并且设置每 5000 个字节发送一组 CC Seq。因为从数据流中恢复出来的时钟和本地产生的 FIFO 读侧的时钟的频率不可能完全一致，所以才要进行 CC 处理。这个周期是根据发送侧和接收侧的时钟差算出来，然后再根据实际调试结果进行一定的修正。不同的板子不同的环境，这个值都是不同的。

Xilinx 收发器 IP 核支持通道绑定，将多个收发器通道“绑定”成一个速率更高的传输通道，利用 FIFO 消除其间的延时不确定性。Clock correction 是最后一个重要的点。先来看看 RX 通道的结构和弹性缓存概念。



UG482_c4_25_112911

接收通道中同样有两个时钟域：从 CDR 恢复出的 XCLK 和接收通道工作时钟 RXUSRCLK。RX 通道使用 RX Elastic Buffer 来桥接两时钟域，但由于两者细微的差异会使缓存变空或溢出。为此引入时钟矫正，在发送端周期性发送一些特殊字符，接收端在弹性缓存快满时删除这些字符，快空时复制这些字符从而保证缓存内数据维持动态平衡的状态。

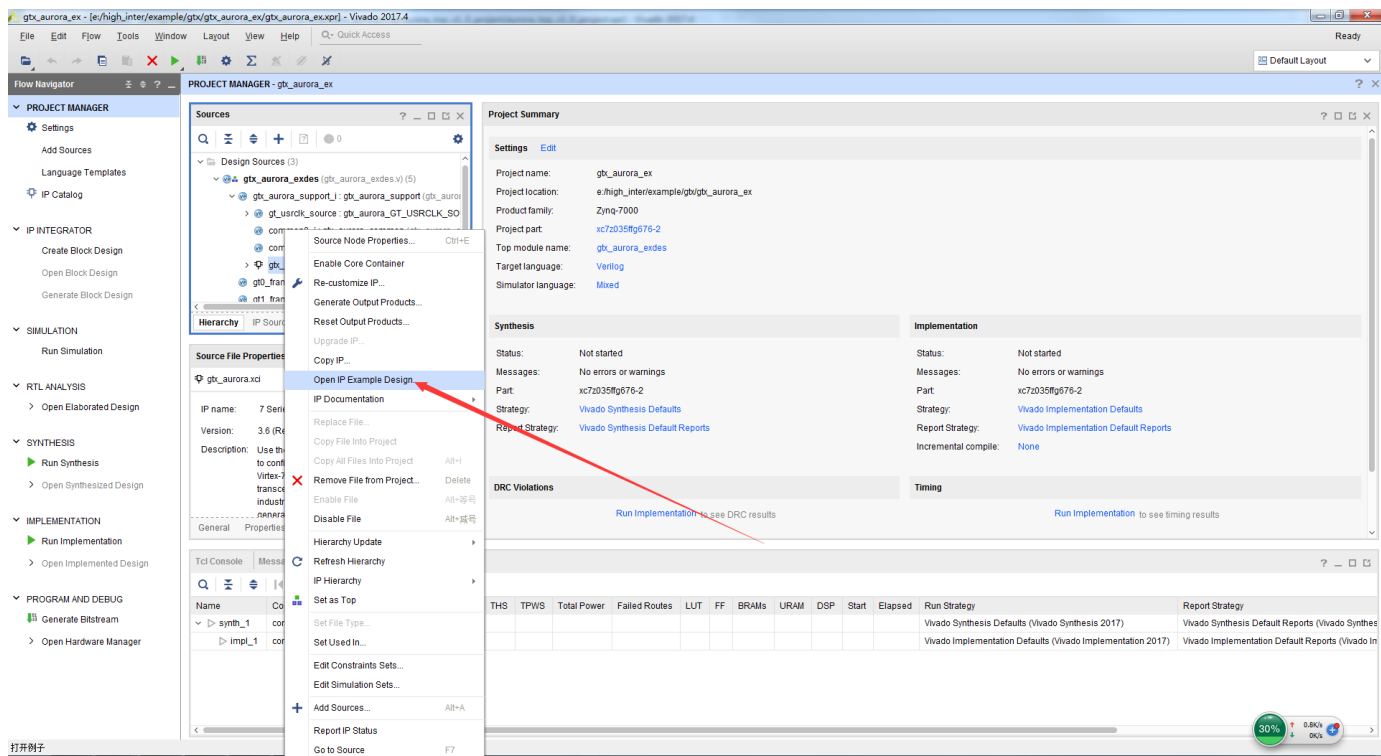


所以，GTP 的发送时钟和 GTP 的接收后的恢复时钟，会存在一定的动态变化，这就是前面我们设置 TXUSRCLK 和 RXUSRCLK 的时候必须要考虑的问题，比如增加 FIFO，解决跨时钟的问题。

7: 可以看到 TXUSRCLK 和 RXUSRCLK 频率为 250M, TXUSRCLK2 和 RXUSRCLK2 为 125M,也就是我们将来写代码要使用的时钟,和前面我们计算的一致。

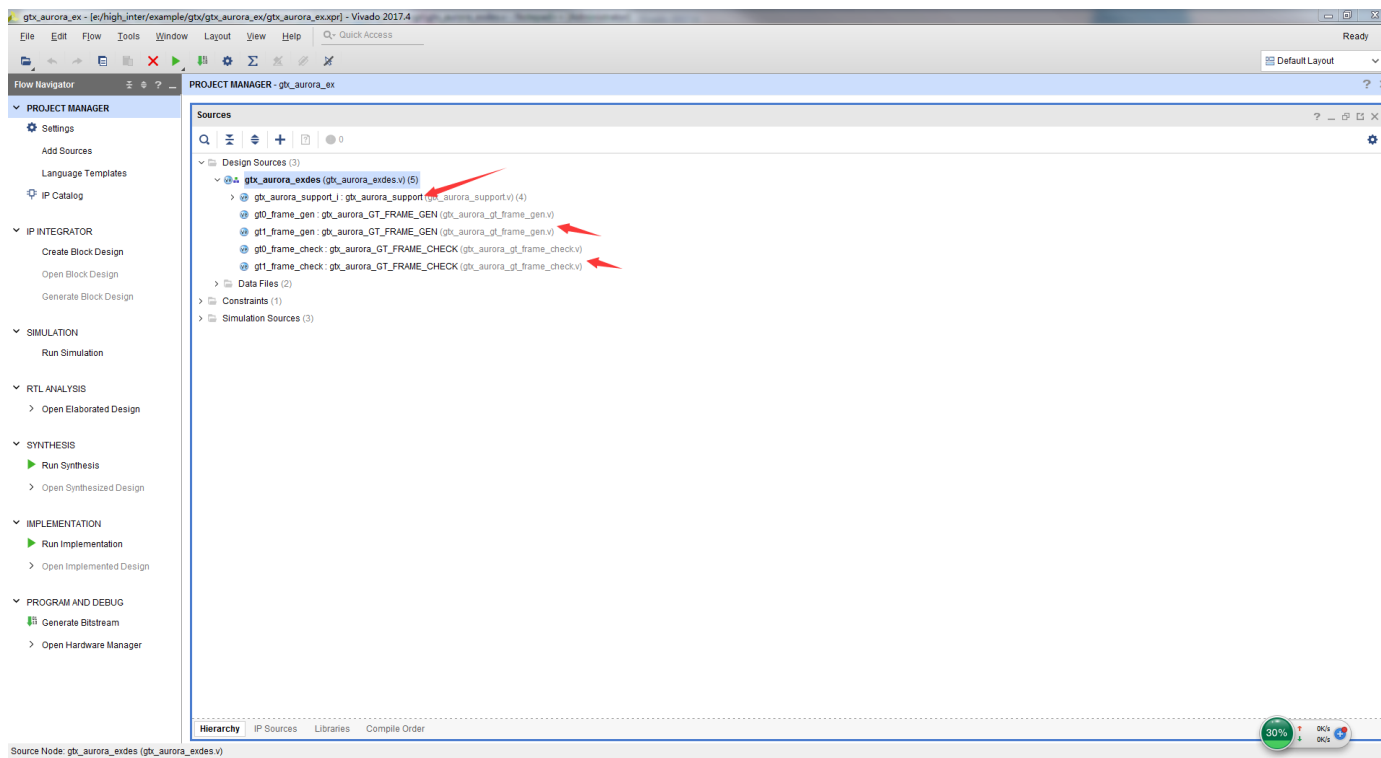
Component Name gtx_aurora	
GT Selection	Line Rate, RefClk Selection
Encoding and Clocking	Comma Alignment and Equalization
PCle, SATA, PRBS	CB and CC Sequence
Summary	
Summary	
Features	GT
Protocol File	aurora_8b10b_single_lane_4byte
TX Line Rate(Gbps)	5.000
TX reference clock(MHz)	125.000
Encoding	8B/10B
TX Internal Data width	20
TX External Data width	32
TXUSRCLK(MHz)	250.0
TXUSRCLK2(MHz)	125.0
TX Buffer Enabled	true
RX Line Rate(Gbps)	5.000
RX reference clock(MHz)	125.000
Decoding	8B/10B
RX Internal Data width	20
RX External Data Width	32
RXUSRCLK(MHz)	250.0
RXUSRCLK2(MHz)	125.0
RX Buffer Enabled	true

8: 点击 OK 生成 IP 核，选中生成的 gtp IP 核，右击，选中 open IP example design。



1.4 修改 example 工程

1:生成的 example 工程结构如下：一个顶层 exdes 模块,主要由三部分构成，support 是 gtp 收发模块，将来我们只用这个模块，GEN 是测试数据生成模块，CHECK 是接收数据后的检查模块，查看接收是否正确，直接将这两个模块移除，加入我们自己的发送数据编码模块和接收数据解码模块。



2: 修改 exdes 模块：由于不使用默认测试代码，所以删除 GEN 模块，删除 CHECK 模块如下图所示：

```

//*****//
//
//----- Frame Generators -----//
//
//*****//
// The example design uses Block RAM based frame generators to provide test
// data to the GTs for transmission. By default the frame generators are
// loaded with an incrementing data sequence that includes commas/alignment
// characters for alignment. If your protocol uses channel bonding, the
// frame generator will also be preloaded with a channel bonding sequence.

// You can modify the data transmitted by changing the INIT values of the frame
// generator in this file. Pay careful attention to bit order and the spacing
// of your control and alignment characters.

gtx_aurora_GT_FRAME_GEN #
(
    .WORDS_IN_BRAM(EXAMPLE_WORDS_IN_BRAM)
)
gt0_frame_gen
(
    // User Interface
    .TX_DATA_OUT          ({gt0_txdata_float_i, gt0_txdata_i, gt0_txdata_float16_i}),
    .TXCTRL_OUT           ({gt0_txcharisk_float_i, gt0_txcharisk_i}),

    // System Interface
    .USER_CLK              (gt0_txusrclk2_i),
    .SYSTEM_RESET          (gt0_tx_system_reset_c)
);
```

```

//*****//
//
//----- Frame Checkers -----//
//
//*****//
// The example design uses Block RAM based frame checkers to verify incoming
// data. By default the frame generators are loaded with a data sequence that
// matches the outgoing sequence of the frame generators for the TX ports.

// You can modify the expected data sequence by changing the INIT values of the frame
// checkers in this file. Pay careful attention to bit order and the spacing
// of your control and alignment characters.

// When the frame checker receives data, it attempts to synchronise to the
// incoming pattern by looking for the first sequence in the pattern. Once it
// finds the first sequence, it increments through the sequence, and indicates an
// error whenever the next value received does not match the expected value.

assign gt0_frame_check_reset_i = (EXAMPLE_CONFIG_INDEPENDENT_LANES==0)?reset_on_data_error_i:gt0_matchn_i;

// gt0_frame_check0 is always connected to the lane with the start of char
// and this lane starts off the data checking on all the other lanes. The INC_IN port is tied off
assign gt0_inc_in_i = 1'b0;

gtx_aurora_GT_FRAME_CHECK #
(
.RX_DATA_WIDTH ( 32 ),
.RXCTRL_WIDTH ( 4 ),
.COMMA_DOUBLE ( 16'h04bc ),
.WORDS_IN_BRAM(EXAMPLE_WORDS_IN_BRAM),
.START_OF_PACKET_CHAR ( 32'h060504bc )
)
gt0_frame_check
(
// GT Interface
.RX_DATA_IN          (gt0_rxddata_i),
.RXCTRL_IN           (gt0_rxcharisk_i),
.RXENMCOMMADET_OUT   (gt0_rxmcommaalignen_i),
.RXENPCOMMADET_OUT   (gt0_rxpcommaalignen_i),
.RX_ENCHAN_SYNC_OUT  ( ),
.RX_CHANBOND_SEQ_IN  (tied_to_ground_i),
// Control Interface
.INC_IN              (gt0_inc_in_i),
.INC_OUT              (gt0_inc_out_i),
.PATTERN_MATCHB_OUT  (gt0_matchn_i),
.RESET_ON_ERROR_IN   (gt0_frame_check_reset_i),

```

3:将 drp_clk 直接连入 sysclk_in 即可，官方的例子这个时钟是引脚进来的加了 bufg，我们用 PLL 产生即可。修改前如下图

```

IBUFDS IBUFDS_DRP_CLK
(
.I (DRP_CLK_IN_P),
.B (DRP_CLK_IN_N),
.O (DRPCLK_IN)
);

BUFG DRP_CLK_BUFG
(
.I (DRPCLK_IN),
.O (drpclkin_i)
);

```


修改为: assign drpclk_in_i = drp_clk;

4:对接口处进行修改,修改后如下

对于使用单光模块进行环路和双头光模块进行环路的设置不一样。

使用 1 个通道环路测试

```
module gt_aurara_exdes
(
    input wire  Q0_CLK1_GTREFCLK_PAD_N_IN,
    input wire  Q0_CLK1_GTREFCLK_PAD_P_IN,
    input wire          drp_clk  ,
    output          tx0_clk  ,
    input    [31:0]    tx0_data  ,
    input    [3:0]     tx0_kchar ,

    output          rx0_clk  ,
    output    [31:0]    rx0_data  ,
    output    [3:0]     rx0_kchar ,
    output          gt0_tx_system_rstn ,
    output          gt0_rx_system_rstn ,
    input wire        RXN_IN,
    input wire        RXP_IN,
    output wire       TXN_OUT,
    output wire       TXP_OUT
);
```

使用 2 个通道环路测试

```
module gt_aurara_exdes
(
    input wire  Q0_CLK1_GTREFCLK_PAD_N_IN,
    input wire  Q0_CLK1_GTREFCLK_PAD_P_IN,
    input wire  drp_clk  ,

    output          tx0_clk  ,
    input    [31:0]    tx0_data  ,
    input    [3:0]     tx0_kchar ,
    output          rx0_clk  ,
    output    [31:0]    rx0_data  ,
    output    [3:0]     rx0_kchar ,
    output          gt0_tx_system_rstn ,
    output          gt0_rx_system_rstn ,

    output          tx1_clk  ,
    input    [31:0]    tx1_data  ,
    input    [3:0]     tx1_kchar ,
    output          rx1_clk  ,
    output    [31:0]    rx1_data  ,
    output    [3:0]     rx1_kchar ,
    output          gt1_tx_system_rstn ,
    output          gt1_rx_system_rstn ,

```

```

input wire [1:0]    RXN_IN,
input wire [1:0]    RXP_IN,
output wire [1:0]    TXN_OUT,
output wire [1:0]    TXP_OUT
);

```

5:添加如下用户代码，这些信号都是此模块的接口信号，直接和 IP 核相连接

对于 1 个通道环路测试的代码如下：

```

//通道 0
assign tx0_clk          = gt0_txusrclk2_i      ;//用户发送时钟
assign rx0_clk          = gt0_rxusrclk2_i      ;//用户接收时钟
assign gt0_txdata_i     = tx0_data             ;//用户发送数据
assign gt0_txcharisk_i  = tx0_kchar            ;//用户发送 K 码
assign rx0_data         = gt0_rxdata_i        ;//用户接收数据
assign rx0_kchar        = gt0_rxcharisk_i      ;//用户接收 K 码
assign gt0_tx_system_rstn = gt0_tx fsmresetdone_i ;//gtp 初始化完成标志
assign gt0_rx_system_rstn = gt0_rx fsmresetdone_i ;//gtp 初始化完成标志

```

对于 2 个通道环路测试的代码如下：

```

//通道 0
assign tx0_clk          = gt0_txusrclk2_i      ;//用户发送时钟
assign rx0_clk          = gt0_rxusrclk2_i      ;//用户接收时钟
assign gt0_txdata_i     = tx0_data             ;//用户发送数据
assign gt0_txcharisk_i  = tx0_kchar            ;//用户发送 K 码
assign rx0_data         = gt0_rxdata_i        ;//用户接收数据
assign rx0_kchar        = gt0_rxcharisk_i      ;//用户接收 K 码
assign gt0_tx_system_rstn = gt0_tx fsmresetdone_i ;//gtp 初始化完成标志
assign gt0_rx_system_rstn = gt0_rx fsmresetdone_i ;//gtp 初始化完成标志

//通道 1
assign tx1_clk          = gt1_txusrclk2_i      ;
assign rx1_clk          = gt1_rxusrclk2_i      ;
assign gt1_txdata_i     = tx1_data             ;
assign gt1_txcharisk_i  = tx1_kchar            ;
assign rx1_data         = gt1_rxdata_i        ;
assign rx1_kchar        = gt1_rxcharisk_i      ;
assign gt1_tx_system_rstn = gt1_tx fsmresetdone_i ;//gtp 初始化完成标志
assign gt1_rx_system_rstn = gt1_rx fsmresetdone_i ;//gtp 初始化完成标志

```

6:此处改为

修改前

```

gtx_aurora_support #
(
    EXAMPLE_SIM_GTRESET_SPEEDUP (EXAMPLE_SIM_GTRESET_SPEEDUP),
    STABLE_CLOCK_PERIOD (STABLE_CLOCK_PERIOD)
)
gtx_aurora_support_i |
(

```

修改后

```
gtx_aurora_support #
(
    .EXAMPLE_SIM_GTRESET_SPEEDUP    ("FALSE"),
    .STABLE_CLOCK_PERIOD             (10)
)
gtx_aurora_support_i
(
```

7:修改箭头所指处为 1'b1,

对于使用 2 个通道进行环路测试的需要把 gt0_data_valid_in 和 gt1_data_valid_in 都设置 1'b1, 对于只使用 1 个通道进行环路测试的只要设置 gt0_data_valid_in 为 1'b1

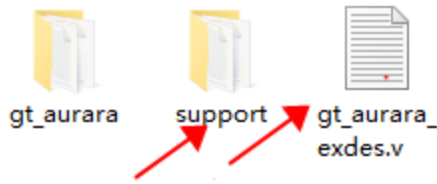
```
gt_aurara_support #
(
    .EXAMPLE_SIM GTRESET_SPEEDUP    ("FALSE"),
    .STABLE_CLOCK_PERIOD            (10)
)
gt_aurara_support_i
(
    .soft_reset_tx_in                (soft_reset_i),
    .soft_reset_rx_in                (soft_reset_i),
    .dont_reset_on_data_error_in     (tied_to_ground_i),
    .q0_clk1_gtrefclk_pad_n_in(Q0_CLK1_GTREFCLK_PAD_N_IN),
    .q0_clk1_gtrefclk_pad_p_in(Q0_CLK1_GTREFCLK_PAD_P_IN),
    .gt0_tx_mmcm_lock_out            (gt0_txmmcm_lock_i),
    .gt0_rx_mmcm_lock_out            (gt0_rxmmcm_lock_i),
    .gt0_tx_fsm_reset_done_out       (gt0_txfsmresetdone_i),
    .gt0_rx_fsm_reset_done_out       (gt0_rxfsmresetdone_i),
    .gt0_data_valid_in               (1'b1),
    .gt0_txusrclk_out(gt0_txusrclk_i),
    .gt0_txusrclk2_out(gt0_txusrclk2_i),
    .gt0_rxusrclk_out(gt0_rxusrclk_i),
    .gt0_rxusrclk2_out(gt0_rxusrclk2_i),
```

设置 gt0_rxmcommaalignen_in 和 gt0_rxmcommaalignen_in 为 1'b1

```
.gt0_rxdisperr_out                (gt0_rxdisperr_i),
.gt0_rxnotintable_out            (gt0_rxnotintable_i),
//----- Receive Ports - RX AFE Ports -----
.gt0_gtprxn_in                   (RXN_IN),
.gt0_gtprxp_in                   (RXP_IN),
//----- Receive Ports - RX Buffer Bypass Ports -----
.gt0_rxbufreset_in               (gt0_rxbufreset_i),
.gt0_rxbufstatus_out             (gt0_rxbufstatus_i),
//----- Receive Ports - RX Byte and Word Alignment Ports -----
.gt0_rxbyteisaligned_out         (gt0_rxbyteisaligned_i),
.gt0_rxbyterealign_out           (gt0_rxbyterealign_i),
.gt0_rxcommadet_out              (gt0_rxcommadet_i),
.gt0_rxmcommaalignen_in          (1'b1),
.gt0_rxpcommaalignen_in          (1'b1),
//----- Receive Ports - RX Decision Feedback Equalizer(DFE) -----
.gt0_dmonitorout_out             (gt0_dmonitorout_i),
//----- Receive Ports - RX Equailizer Ports -----
.gt0_rxlpmhfold_in               (tied_to_ground_i),
.gt0_rxlpmhfovrden_in            (gt0_rxlpmhfovrden_i),
.gt0_rxlpmhfold_in               (tied_to_ground_i),
//----- Receive Ports - RX Fabric Output Control Ports -----
.gt0_rxoutclkfabric_out          (gt0_rxoutclkfabric_i),
//----- Receive Ports - RX Initialization and Reset Ports -----
.gt0_gtrxreset_in                (tied_to_ground_i),
.gt0_rxlpmreset_in               (gt0_rxlpmreset_i),
.gt0_rxpcsreset_in               (tied_to_ground_i),
.gt0_rxpmarereset_in             (gt0_rxpmarereset_i),
```

对于使用 2 个通道进行环路测试的，还要设置 `gt1_rxmcommaalignen_in` 和 `gt1_rxmcommaalignen_in` 为 1'b1

8:最后还有一点需要说明，如果对 `gtp ip` 有任何的修改，重新生成 IP 核之后，需要打开 `example` 工程，然后将 `example support` 目录下的文件以及 `gtp_aurora_exdes.v`，全部复制出来，粘贴到我们的 `gtk_8b10b` 目录下。



至此，`gtp` 配置和修改已经完成，可以随心所欲传输我们想传输的数据了，视频编解码模块在代码中有很详细的解释。

3.5 封装自定义 IP

3.5.1 video_encode.v

本模块对视频流进行编码，将外部 HDMI 进来的视频流用 FIFO 进行缓冲，16 位进 32 位出，以视频 `vsync` 信号上升沿时刻算起，当 `vsync` 上升沿到来时，将上升沿编码为：32'h55_00_00_bc, 32'h55_00_01_bc, 当 FIFO 中的数据量达不到要求时，发送无效数据：32'h55_00_02_bc, 32'h55_00_03_bc 交替发送，当 FIFO 中有一定量的数据后，首先发送 1 个 32 位的：32'h55_00_04_bc 表明一行就要开始传输，然后发送每行数据行号+bc, 紧接着将 FIFO 中的数据依次发出去，发送完一行数据后，发送 32'h55_00_05_bc, 表明一行数据发送完成，紧接着发送上述的无效数据：32'h55_00_02_bc, 32'h55_00_03_bc 交替发送，等待 `vsync` 下降沿到来，将其编码为 32'h55_00_06_bc, 32'h55_00_07_bc 发出，到此一行发送结束，以同样的编码将视频流每一行的数据发送给 `gtp` 即可。

接下来，对上述做一些解释：

1. 图像 16 位进 32 位出，因为在 IP 核设置 `gtp` 对于外部接口为 32 位；
2. 上述中的同步信号以及穿插的无用信号的高 24 位皆为自定义，低 8 位为 bc, bc 是 k28.5 的控制字符，这个在 IP 核中会进行设置；代码中有个 `coded_ctr` 信号，这个信号共 4bit，每个 bit 对应 `coded_data` 的一个字节，表示发送数据里面某个字节为 K 码，也就是我们所说的 bc, 由于 bc 在低 8 位，因此把 4bit 中的 0 位置 1；
3. k 码将来在我们 `gtp` 接收到数据后，可以帮我们查看数据是否错位，如果错位可以将数据对齐；

上述中有一个描述：“当 FIFO 中有一定量的数据后是”什么意思？对于 1080P 的视屏，16 位为一个像素点，他的时钟是 148.5Mhz 左右，`gtp` 在此工程中设置为 5gbps，在 IP 核中可以看到，`user_clk2` 为 125M，这是相对于，32 位数据而言，相对于 16 位数据而言，它相当于 250M，也就是说在此模块 FIFO 的写时钟是 148.5，FIFO 的读时钟是 125，但是是 32 位的读时钟，对于 16 位数据也就相当于 FIFO 读时钟是 250, 148.5 和 250 基本差距有点大就压把 FIFO 缓存设置大一些，不至于被读空，这个 FIFO_VTH 的设置和视频输入分辨率和 `gtp` 的传输速率相关，不同的情况具体分析。

3.5.2 video_decode.v

本模块将 `gtp` 接收并对齐后的数据进行解码，跟我们的编码模块相反

3.5.3 data_align.v

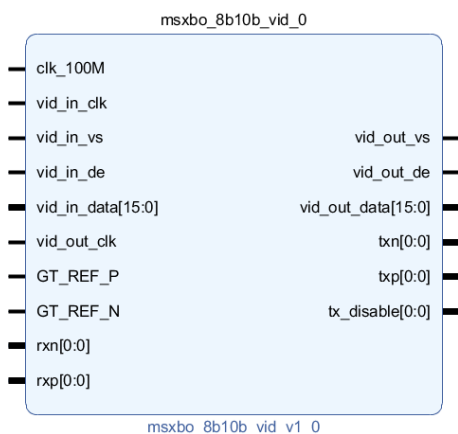
本模块将 `gtp` 接收后的数据利用 k 码进行对齐，通过逻辑分析仪抓取调试过程中，有时候会出现，发送的 32

位数据可能出现 16 位数据移位，也就是上一个 32 位的数据低 16 位可能和下一个 32 位数据的高 16 位拼接在一起，我们如何知道我们发出的数据，通过光纤传输接收后进行了错位，此时就轮到 K 码大显威力了，当我们接收的数据错位时，K 码也会错位，此前我们发送的 0001，可能会变为 0100，然后我们根据这个 0100 来把接收到的数据重新组合，就可以得到正确的结果。

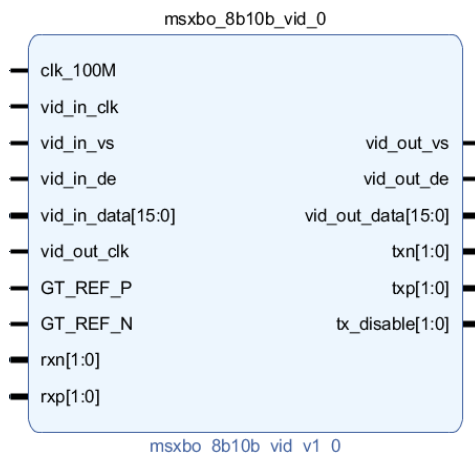
3.5.4 封装 IP

将以上代码和 gtp ip 一起封装成自定义 IP 方便开发测试使用，具体的封装过程不提供，不懂的可以参考我们基础部分课程有关自定义 IP 封装的内容。由于采用 1 个通道的 IP 和两个通道的 IP 不一样，读者可以根据自己实际使用情况而定，当然也许你会使用 4 个通道，或者更多 8 通道。方法都是都是一样的。

对于使用 1 个通道进行通信的 IP



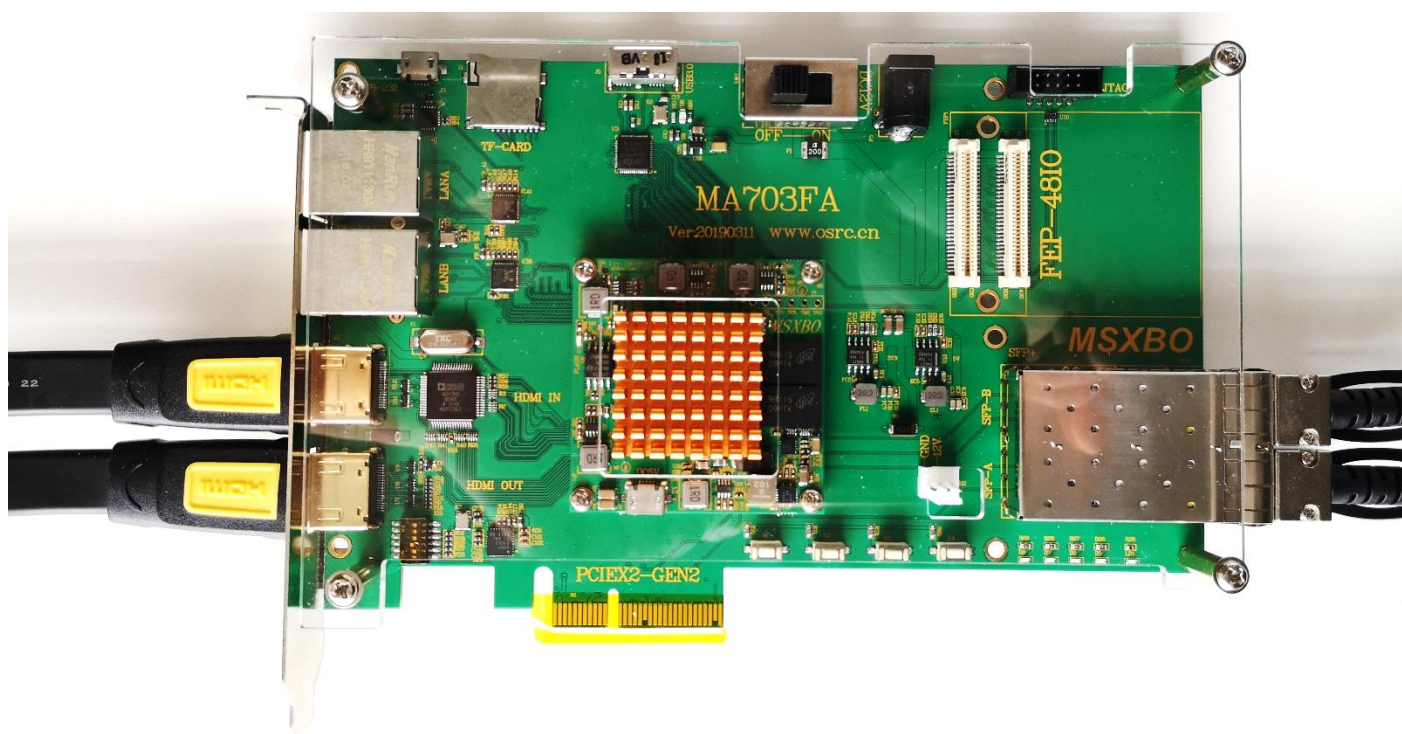
对于使用 2 个通道进行测试的 IP



3.6 环路测试视频传输

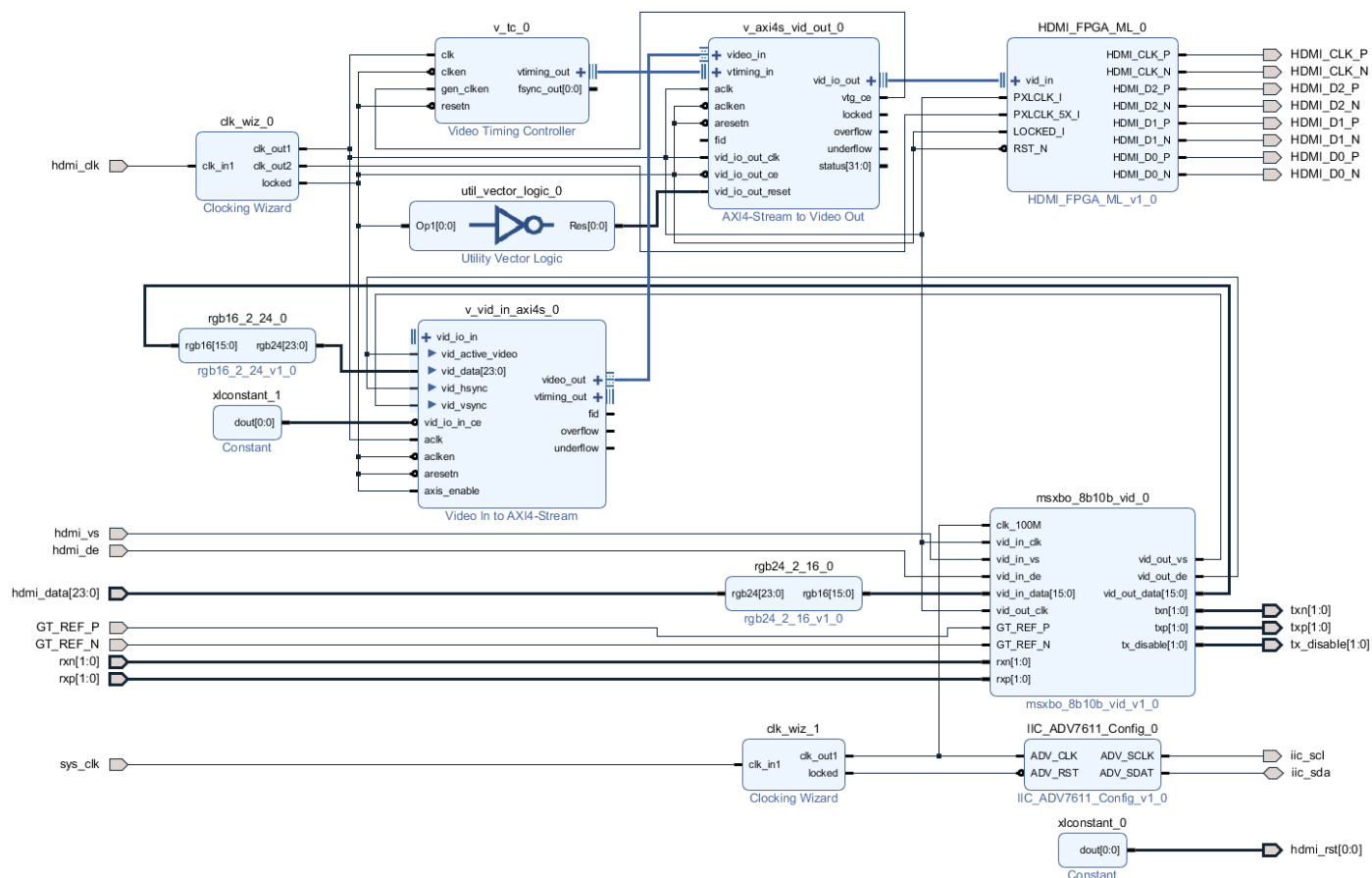
环路测试可以使用单头光模块或者双头光模块，我们这里以双头光模块 2 路 SFP 对接后测试为例，同时代码中给出给出了单头光模块单路 SFP 环路测试的代码

3.6.1 硬件接线图



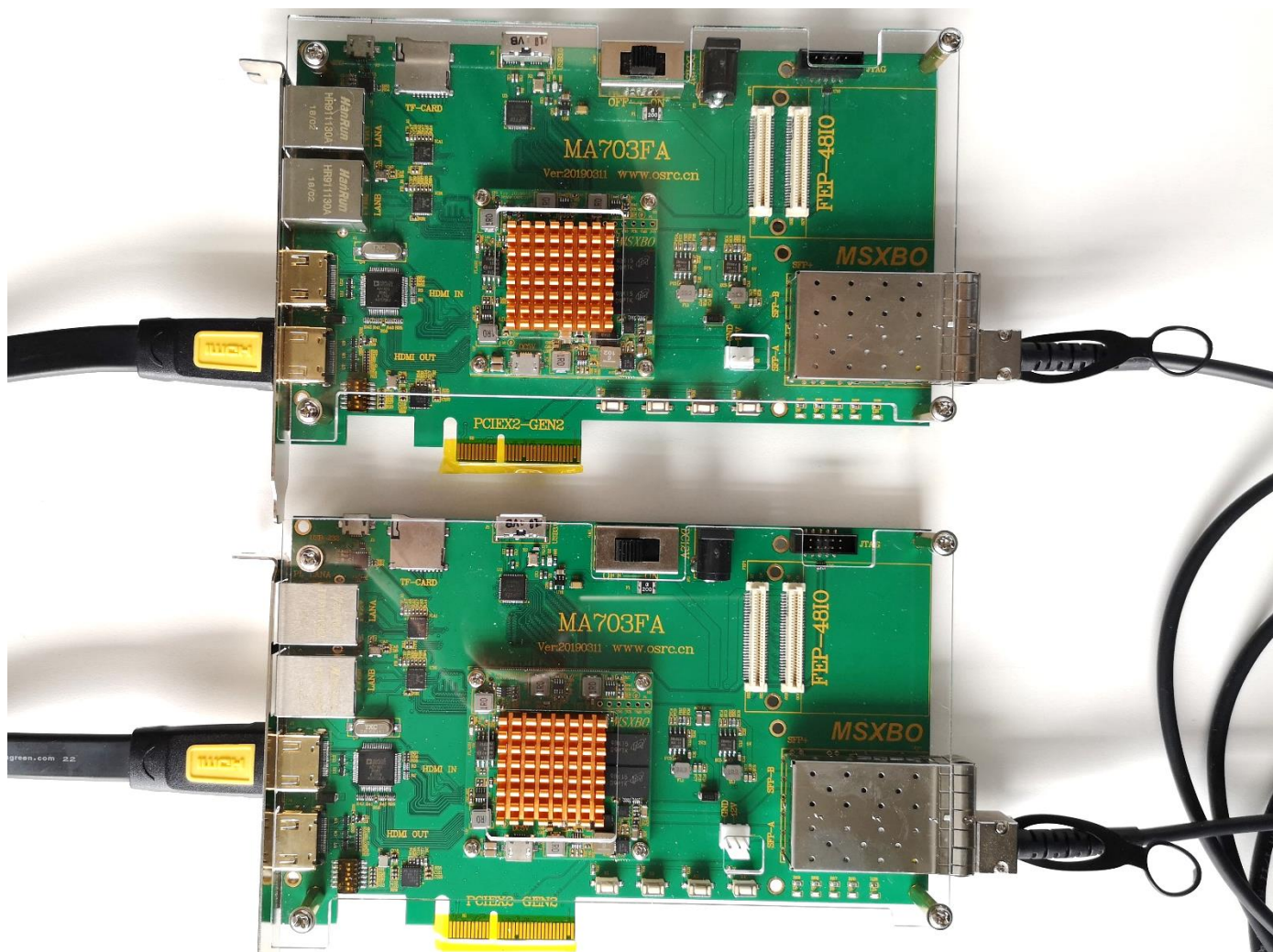
3.6.2 FPGA BD 工程

直通方式测试，HDMI 输入视频采集后进入已经封装好的“msxbo_8b10b_vid”IP 通过 SFP 接口的光纤环路后读取回来，然后利用 HDMI 输出显示，这种方法没有用到 DDR，程序较为简单，硬件成本低。另外需要注意，对于 ADV7611 的 HDMI 输入方案，由于 VS 数据阶段高电平，而我们的 msxbo_8b10b IP 要求数据阶段是低电平所以需要把 hmdi_vs 取反后进入 BD 工程。

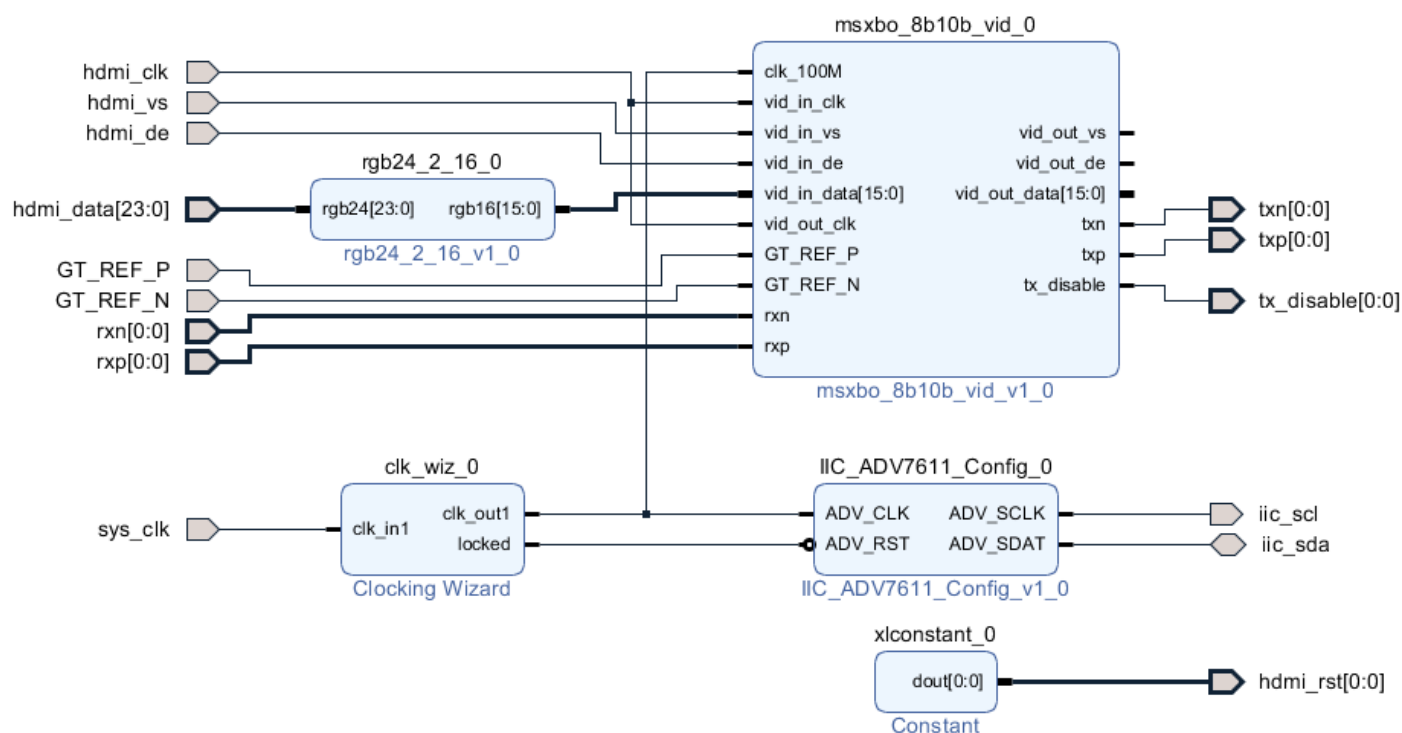


3.7 板到板间视频传输

3.7.1 硬件接线图

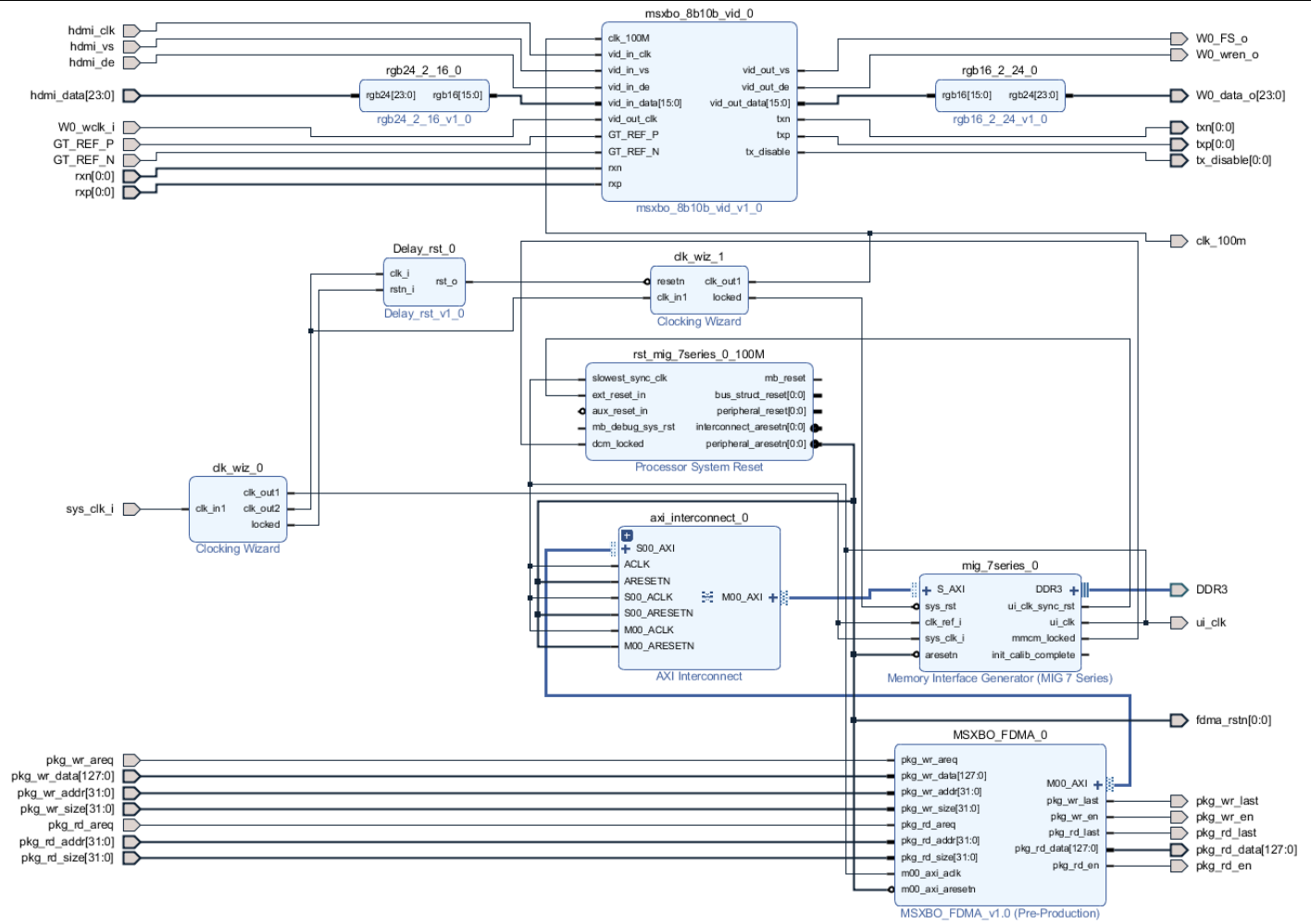


3.7.2 发送端 FPGA BD 工程



3.7.3 接收端 FPGA BD 工程

接收端使用到了 FMDA 进行视频的 3 帧缓存后输出到显示器，如果你对 FDMA IP 使用还不清楚请学习我们 FDMA 方便的课程内容。



3.8 测试结果

