GTH 光纤通信测试例程

1 实验简介

Xilinx 的 Kintex-UltraScale 系列 FPGA 集成了 GTH串行高速收发器，可以实现高速串行数据通信。在AXKU040 开发板上，FPGA 的 GTH 其中的 20 个收发器通道中 4路连接到 SFP+光模块接口，用户只需要另外购买 SPF 的光模块就可以实现光纤的数据传输。本实验将介绍通过光纤连接实现光模块之间的数据收发和眼图的测试。

2 实验原理

2.1 GTH 介绍

AXKU040 的 FPGA 芯片(XCKU040_ffva1156)自带 20路高速 GTH 串行高速收发器通道， 每通道的收发速度为高达 16.3 Gb/s。GTH 收发器支持不同的串行传输接口或协议，比如 PCIE 1.1/2.0/3.0 接口、万兆网 XUAI 接口、OC-48、串行 RapidIO 接口、 SATA(Serial ATA) 接口、数字分量串行接口(SDI)等等。

Xilinx 以 Quad 来对串行高速收发器进行分组，四个串行高速收发器和一个 COMMOM（QPLL）组成一个 Quad，每一个串行高速收发器称为一个 Channel(通道)

GTH 的具体内部逻辑框图如下所示，它由 4 个如下图的四个收发器通道 GTHE3/4_CHANNEL 和一个 GTHE3/4_COMMON 组成。每路 GTHE2_CHANNEL 包含发送电路 TX 和接收电路 RX。

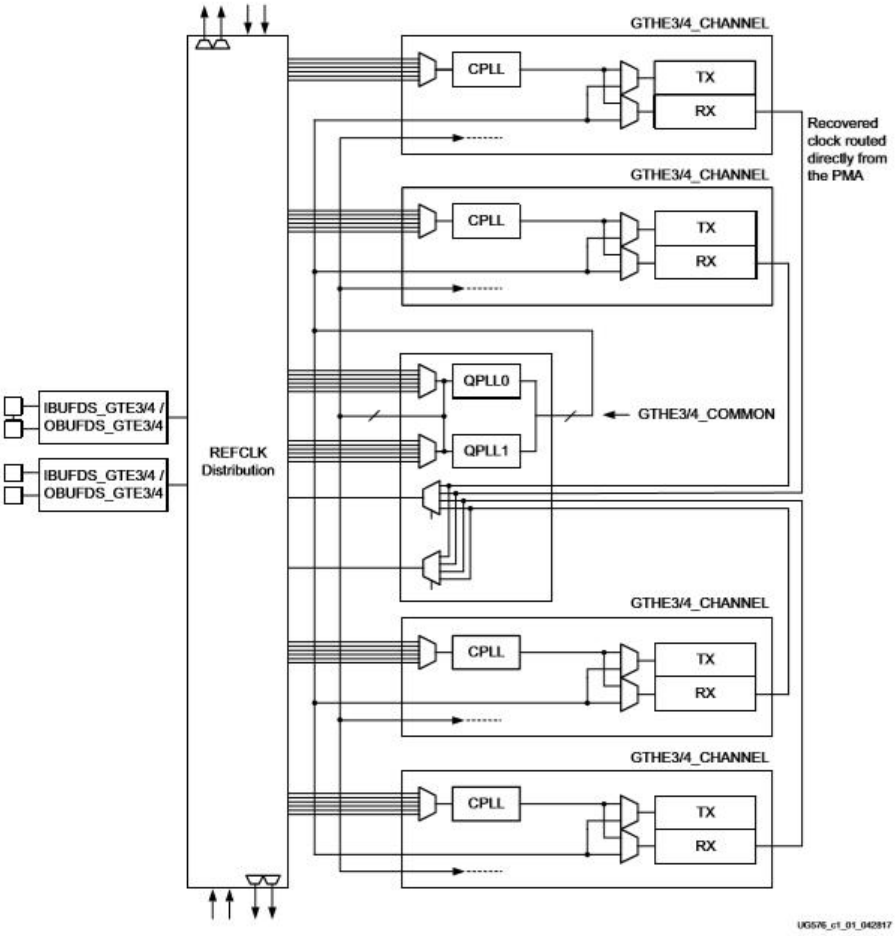


Figure 1-1: **GTH Transceiver Quad Configuration**

每个 GTHE3/4_CHANNEL 的逻辑电路如下图所示：

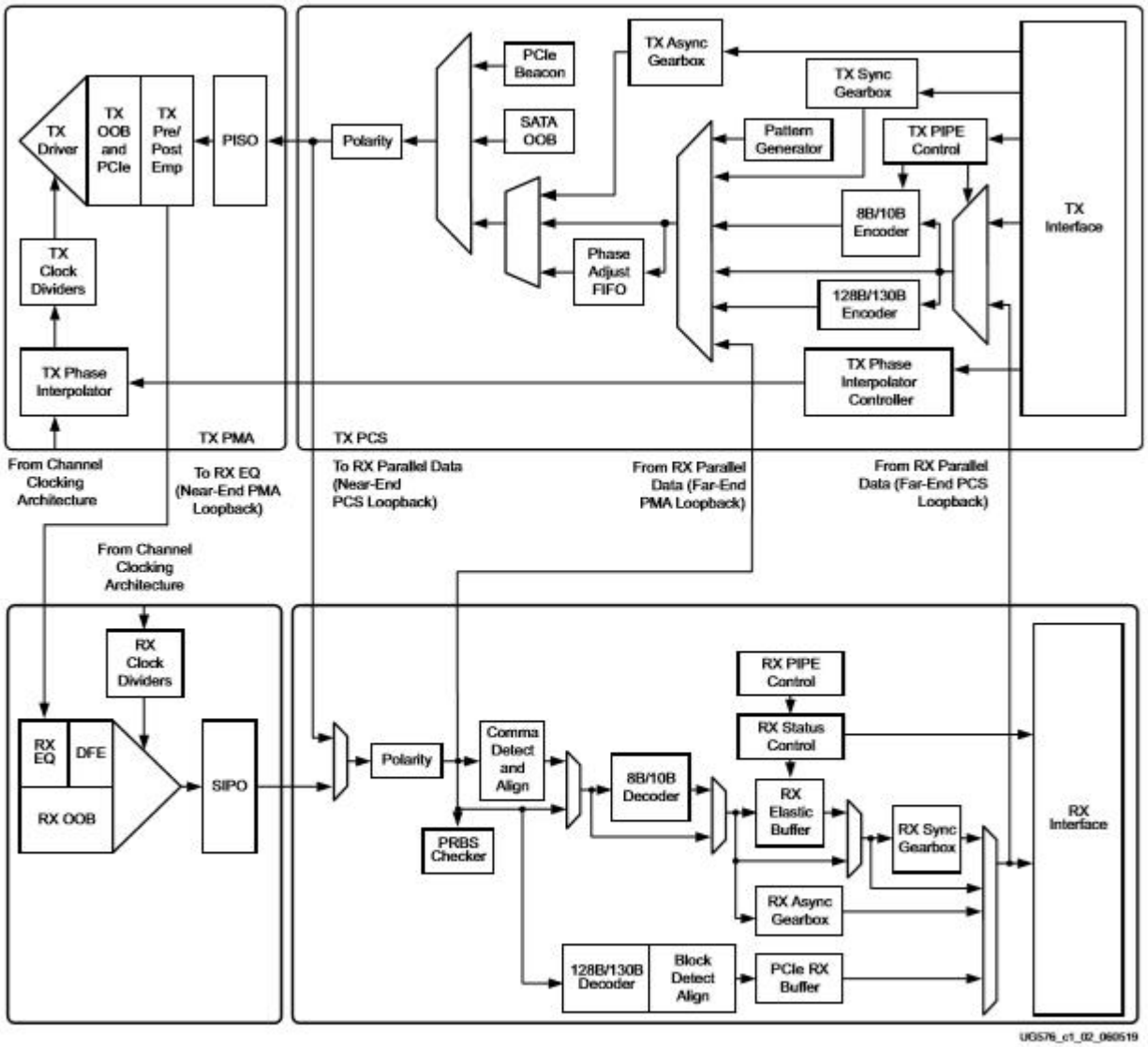


Figure 1-2: **GTHE3/4_CHANNEL Primitive Topology**

GTHE3/4_CHANNEL 的发送端和接收端功能是独立的，均由 PMA(Physical Media Attachment，物理媒介适配层)和 PCS(Physical Coding Sublayer，物理编码子层)两个子层组成。其中 PMA 子层包含高速串并转换(Serdes)、预/后加重、接收均衡、时钟发生器及时钟恢复等电路。PCS 子层包含 8B/10B 编解码、缓冲区、通道绑定和时钟修正等电路。

GTH 发送和接收处理流程：

首先用户逻辑数据经过 8B/10B 编码后，进入一个发送缓存区（Phase Adjust FIFO），该缓冲区主要是 PMA 子层和 PCS 子层两个时钟域的时钟隔离，解决两者时钟速率匹配和相位差异的问题，最后经过高速 Serdes 进行并串转换(PISO)，有必要的话，可以进行预加重(TX Pre-emphasis)、后加重。值得一提的是，如果在 PCB 设计时不慎将 TXP 和 TXN 差分引脚交叉连接，则可以通过极性控 制(Polarity)来弥补这个设计错误。接收端和发送端过程相反，相似点较多，这里就不赘述了，需 要注意的是 RX 接收端的弹性缓冲区，其具有时钟纠正和通道绑定功能。

GTH 的参考时钟

GTH 模块有四组参考时钟， 每组有两个差分参考时钟输入管脚(MGTREFCLK0P/N 和 MGTREFCLK1P/N），作为 GTH 模块的参考时钟源，用户可以自行选择。AXKU040 的核心板上，有一路 125Mhz 的 GTH 参考时钟连接到 MGTREFCLK0P/N 上，作为 GTH 的参考时钟。差分参考时钟通过 IBUFDS 模块转换成单端时钟信号进入到 GTHE3/4_COMMOM 的 QPLL 和 GTHE3/4_CHANNEL 的 CPLL 中，产生 TX 和 RX 电路中所需的时钟频率。TX 和 RX 收发器速度相同的话，TX 电路和 RX 电路可以使用同一个 PLL 产生的时钟，如果 TX 和 RX 收发器速度不相同的话，需要使用不同的 PLL 时钟产生的时钟。

GTH 的 FPGA TX 接口信号

TX 接口信号是 FPGA 的用户数据发往 GTH 的接口信号，该接口信号的名称和说明如下表所示：

Port	Dir	Clock Domain	Description
TXCHARDISPMODE[7:0]	In	TXUSRCLK2	When 8B/10B encoding is disabled, TXCHARDISPMODE is used to extend the data bus for 20-, 40- and 80-bit TX interfaces.
TXCHARDISPVAL[7:0]	In	TXUSRCLK2	When 8B/10B encoding is disabled, TXCHARDISPVAL is used to extend the data bus for 20-, 40- and 80-bit TX interfaces.

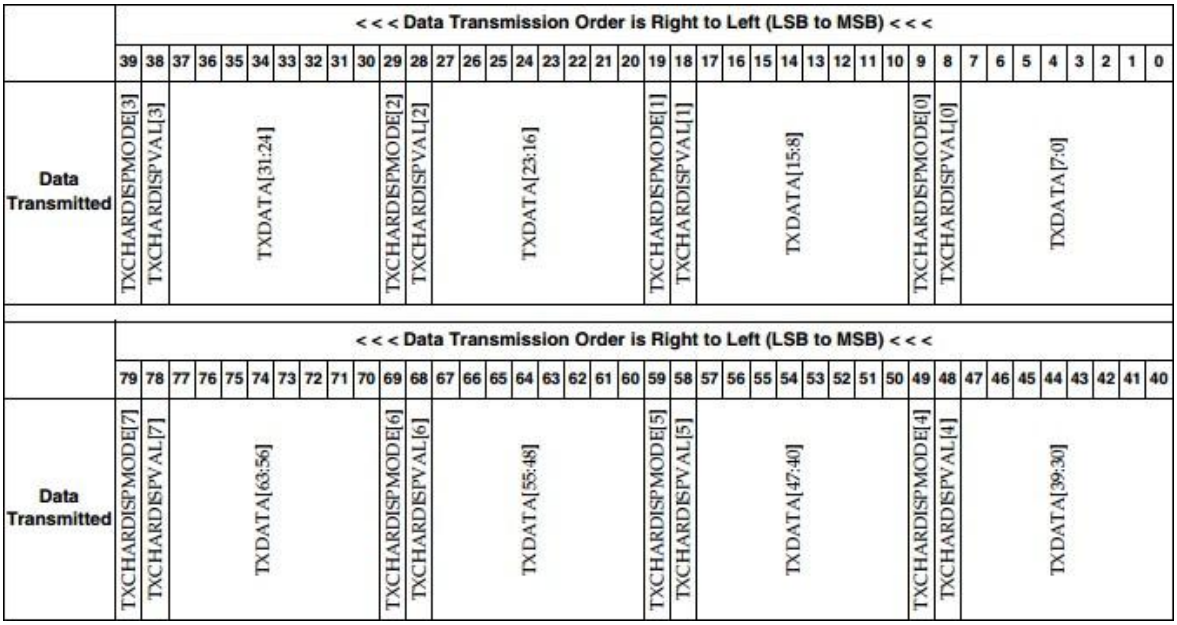
Port	Dir	Clock Domain	Description
TXDATA[63:0]	In	TXUSRCLK2	The bus for transmitting data. The width of this port depends on TX_DATA_WIDTH: TX_DATA_WIDTH = 16, 20: TXDATA[15:0] = 16 bits wide TX_DATA_WIDTH = 32, 40: TXDATA[31:0] = 32 bits wide TX_DATA_WIDTH = 64, 80: TXDATA[63:0] = 64 bits wide When a 20-bit, 40-bit, or 80-bit bus is required, the TXCHARDISPVAL and TXCHARDISPMODE ports from the 8B/10B encoder is concatenated with the TXDATA port. See Table 3-2, page 109.
TXUSRCLK	In	Clock	This port is used to provide a clock for the internal TX PCS datapath.
TXUSRCLK2	In	Clock	This port is used to synchronize the FPGA logic with the TX interface. This clock must be positive-edge aligned to TXUSRCLK when TXUSRCLK is provided by the user.

其中 TXCHARDISPMODE[7:0]和 TXCHARFDISPVAL[7:0]的功能由 TX8B10BEN 是否使能来决定，当 8B/10B 使能，这些信号作为 TX 的数据信号。

TXDATA[63:0]发送数据接口信号，数据宽度由 TX_DATA_WIDTH 参数决定，当 TX_DATA_WIDTH 为 16，或者 20 时，TXDATA 数据宽度为 16；当 TX_DATA_WIDTH 为 32，或者 40 时，TXDATA 数据宽度为 32，当 TX_DATA_WIDTH 为 64，或者 80 时，TXDATA 数据宽度为 64。通过 TX8B10BEN、TX_DATA_WIDTH 参数设置可以配置成不同的 FPGA 接口数据位宽和 GTH 内部数据宽度，如下表所示：

TX8B10BEN	TX_DATA_WIDTH	TX_INT_DATAWIDTH	FPGA Interface Width	Internal Data Width
0	16	0	16	16
	20	0	20	20
	32	0	32	16
	32	1	32	32
	40	0	40	20
	40	1	40	40
	64	1	64	32
	80	1	80	40

当 TX8B10BEN 不使能时，由 TXCHARDISPMODE[7:0]、TXCHARFDISPVAL[7:0]信号和 TXDATA 组合成 20 位、40 位和 80 位的数据信号。



所有的 FPGA 接口信号的采样时钟是 TXUSRCLK2，在 TXUSRCLK2 的上升沿对 TXDATA 进行采样。TXUSRCLK 是提供给 GTH 模块的 PCS logic 和数据发送，TXUSRCLK 的时钟频率由 GTH 的串行发送速度和内部的数据宽度决定的。计算的公式如下：

$$TXUSRCLK\ Rate = \frac{Line\ Rate}{Internal\ Datapath\ Width}$$

TXUSRCLK2 的频率和 TXUSRCLK 有相关性，它是由 TXUSRCLK 的频率和 TX_DATA_WIDTH 的值决定的。TXUSRCLK2 的时钟频率计算如下表所示：

Table 3-3: **TXUSRCLK2 Frequency Relationship to TXUSRCLK**

FPGA Interface Width	TX_DATA_WIDTH	TX_INT_DATAWIDTH	TXUSRCLK2 Frequency
2-Byte	16, 20	0	$F_{TXUSRCLK2} = F_{TXUSRCLK}$
4-Byte	32, 40	0	$F_{TXUSRCLK2} = F_{TXUSRCLK}/2$
4-Byte	32, 40	1	$F_{TXUSRCLK2} = F_{TXUSRCLK}$
8-Byte	64, 80	1	$F_{TXUSRCLK2} = F_{TXUSRCLK}/2$

TXUSRCLK2 和 TXUSRCLK 两个时钟时应该遵循下面两个准则：

1. TXUSRCLK 和 TXUSRCLK2 必须是上升沿对齐的，偏差越小越好，因此应该使用 BUFGs 或者 BUFRs 来驱动这两个时钟。

2. 即使 TXUSRCLK、TXUSRCLK2 和 GTH 的参考时钟运行在不同的时钟频率，必须保证三者必须使用同源时钟。所以 TXUSRCLK、TXUSRCLK2 时钟频率必须由 GTH 的参考时钟倍频或者分频得到。

GTH 的 FPGA RX 接口信号

RX 接口信号是 GTH 模块发往 FPGA 用户数据的接口信号，该接口信号的名称和说明如下表所示：

Port	Dir	Clock Domain	Description
RXDATA[127:0]	Out	RXUSRCLK2	The bus for receiving data. The width of this port depends on RX_DATA_WIDTH: RX_DATA_WIDTH = 16, 20: RXDATA[15:0] = 16 bits wide RX_DATA_WIDTH = 32, 40: RXDATA[31:0] = 32 bits wide RX_DATA_WIDTH = 64, 80: RXDATA[63:0] = 64 bits wide When a 20-bit, 40-bit, or 80-bit bus is required, the RXCTRL0 and RXCTRL1 ports from the 8B/10B encoder are concatenated with the RXDATA port. Bits [127:64] are unused. See Table 4-47, page 316.
RXUSRCLK	In	Clock	This port is used to provide a clock for the internal RX PCS datapath.
RXUSRCLK2	In	Clock	This port is used to synchronize the interconnect logic with the RX interface. This clock must be positive-edge aligned to RXUSRCLK when RXUSRCLK is provided by the user.
RXDATAEXTENDRSVD	Out	RXUSRCLK2	Reserved.

Port	Dir	Clock Domain	Description
RXCTRL1[15:0]	Out	RXUSRCLK2	When 8B/10B decoding is disabled, RXCTRL1 is used to extend the data bus for 20-bit, 40-bit and 80-bit RX interfaces.
RXCTRL0[15:0]	Out	RXUSRCLK2	When 8B/10B decoding is disabled, RXCTRL0 is used to extend the data bus for 20-bit, 40-bit and 80-bit RX interfaces.

RX 接口信号基本和 TX 接口信号类似, 该用户端口根据不同的 RX8B10BEN 和 RX_DATA_WIDTH 两个参考可以设置成 FPGA 的数据接口宽度和 GTH 内部并行数据宽度如下表所示:

Table 4-51: **FPGA RX Interface Datapath Configuration**

RX8B10BEN	RX_DATA_WIDTH	RX_INT_DATAWIDTH	FPGA Interface Width	Internal Data Width
1	20	0	16	20
	40	0	32	20
	40	1	32	40
	80	1	64	40
0	16	0	16	16
	20	0	20	20
	32	0	32	16
	32	1	32	32
	40	0	40	20
	40	1	40	40
	64	1	64	32
	80	1	80	40

同样 RXUSRCLK 的频率和 RXUSERCLK2 的频率关系也跟 RX_DATA_WIDTH 有关。

Table 4-53: **RXUSRCLK2 Frequency Relationship to RXUSRCLK**

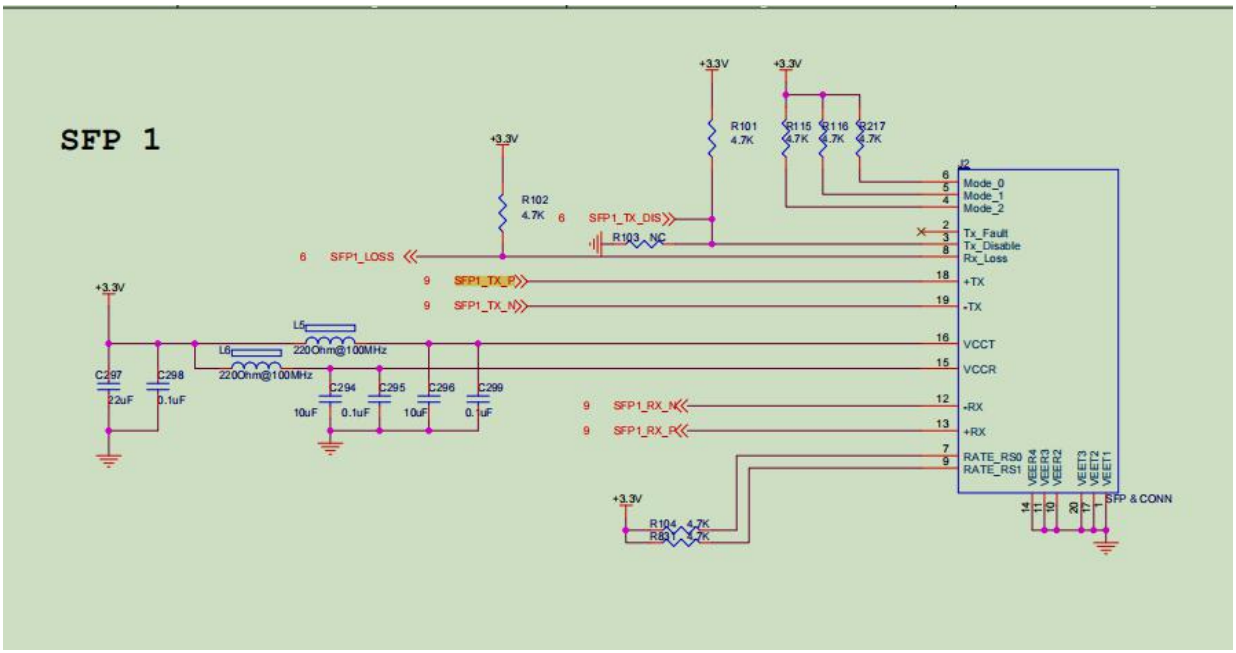
FPGA Interface Width	RX_DATA_WIDTH	RX_INT_DATAWIDTH	RXUSRCLK2 Frequency
2-Byte	16, 20	0	$F_{RXUSRCLK2} = F_{RXUSRCLK}$
4-Byte	32, 40	0	$F_{RXUSRCLK2} = F_{RXUSRCLK} / 2$
4-Byte	32, 40	1	$F_{RXUSRCLK2} = F_{RXUSRCLK}$
8-Byte	64, 80	1	$F_{RXUSRCLK2} = F_{RXUSRCLK} / 2$

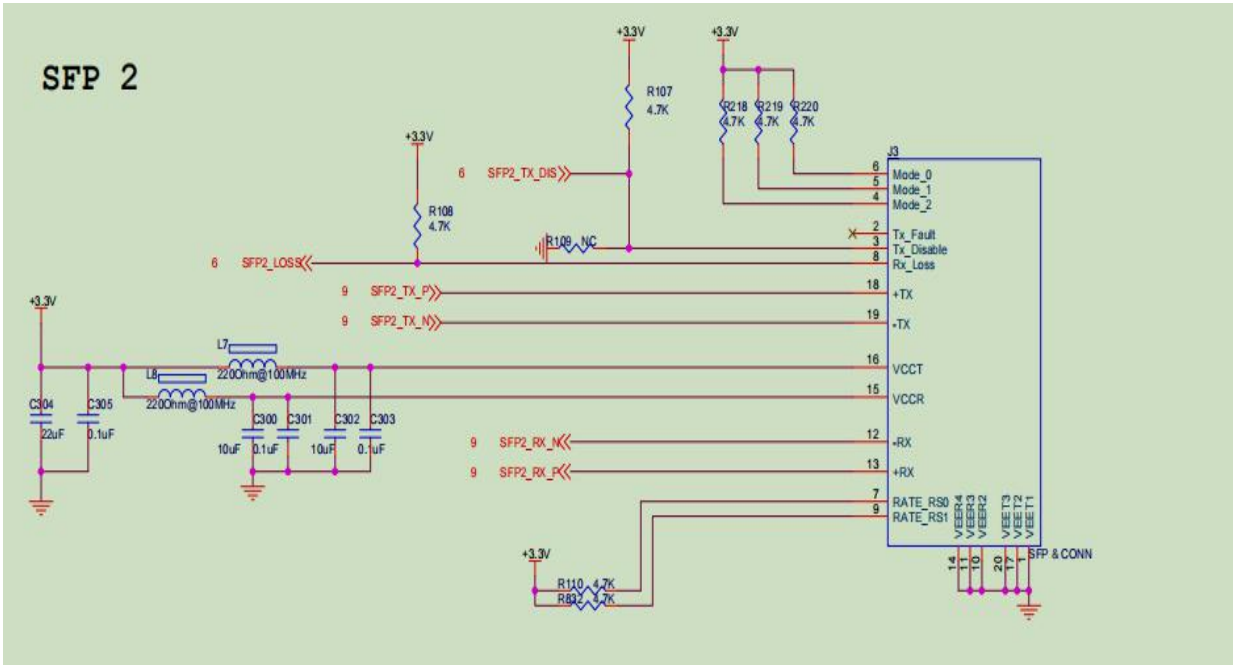
关于 GTH 的知识就讲到这里，大家想了解更多的 GTH 部分硬件的资料和应用，请参考 Xilinx 提供的文档"UG576"。

2.2 硬件介绍

在 AXKU040 开发板上，有 4路光纤接口 连接到 FPGA 芯片的 GTH 的通道上。

光模块的 LOSS 信号和 TX_Disable 信号连接到 FPGA 的普通 IO 上。LOSS 信号用来检测光模块的光接收是否丢失，如果没有插入光纤或者 Link 上，LOSS 信号为高，否则为低。TX_Disable 信号用来使能或者不使能光模块的光发射，如果 TX_Disable 信号为高，光发射关闭，否则光发送使能，正常使用的时候需要拉低此信号。硬件原理图如下：

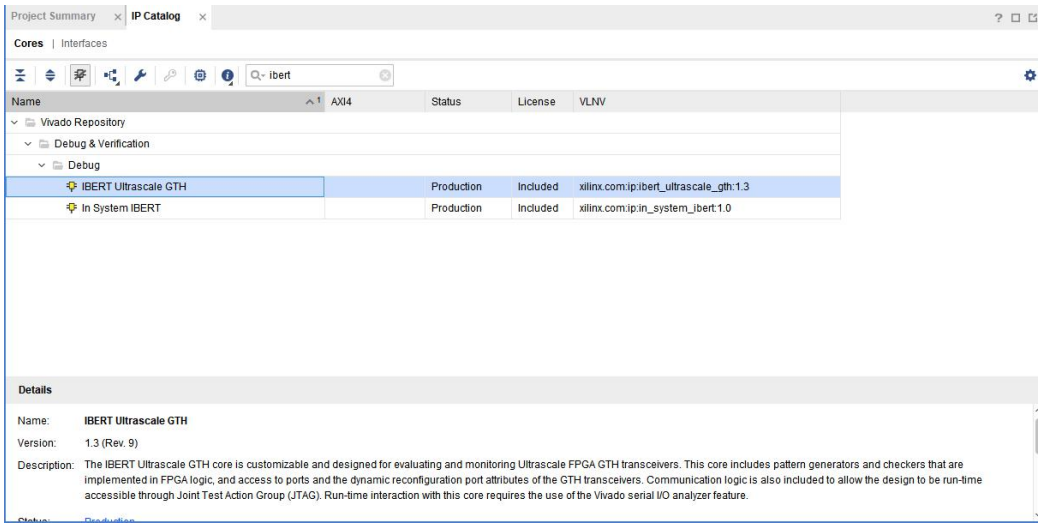

SFP 1
+3.3V
R101 4.7K
R115 4.7K
R116 4.7K
R217 4.7K
R102 4.7K
SFP1_TX_DIS
R103 NC
SFP1_LOSS
SFP1_TX_P
SFP1_TX_N
L5
L6
220Ohm@100MHz
C297 22uF
C298 0.1uF
C294 10uF
C295 0.1uF
C296 10uF
C299 0.1uF
SFP1_RX_N
SFP1_RX_P
R104 4.7K
R83 4.7K
J2
Mode_0
Mode_1
Mode_2
Tx_Fault
Tx_Disable
Rx_Loss
+TX
-TX
VCCT
VCCR
+RX
-RX
RATE_RS0
RATE_RS1
VEER4
VEER3
VEER2
VEET3
VEET2
VEET1
SFP & CONN


SFP 2
+3.3V
R107 4.7K
R218 4.7K
R219 4.7K
R220 4.7K
R108 4.7K
SFP2_TX_DIS
R109 NC
SFP2_LOSS
SFP2_TX_P
SFP2_TX_N
L7
L8
220Ohm@100MHz
C304 22uF
C305 0.1uF
C300 10uF
C301 0.1uF
C302 10uF
C303 0.1uF
SFP2_RX_N
SFP2_RX_P
R110 4.7K
R82 4.7K
J3
Mode_0
Mode_1
Mode_2
Tx_Fault
Tx_Disable
Rx_Loss
+TX
-TX
VCCT
VCCR
+RX
-RX
RATE_RS0
RATE_RS1
VEER4
VEER3
VEER2
VEET3
VEET2
VEET1
SFP & CONN

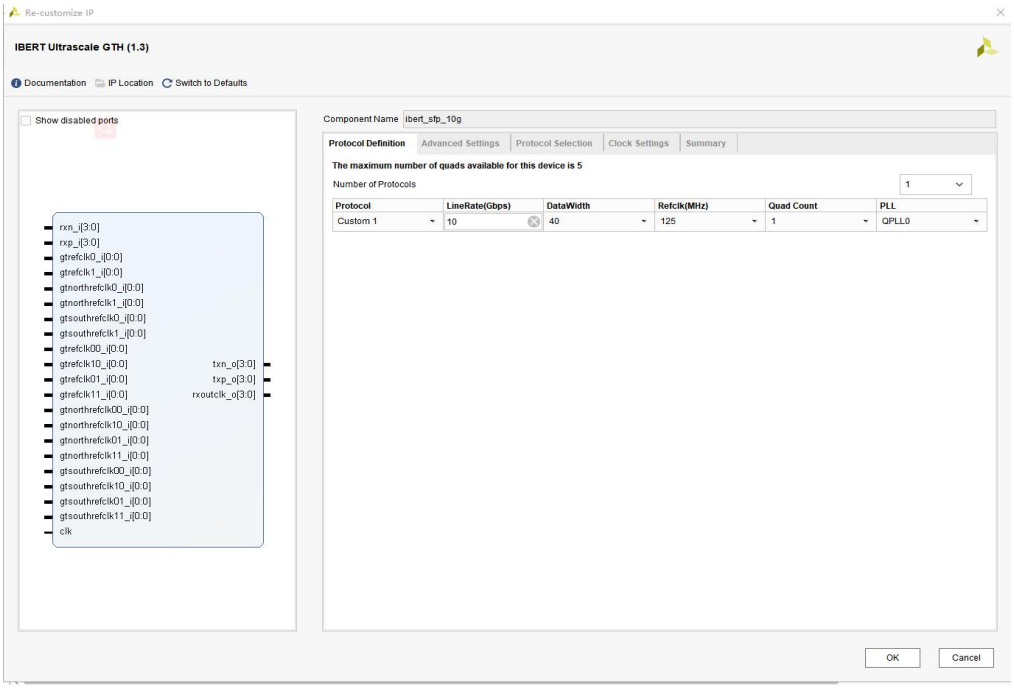
3 程序设计

在使用 GTH 之前，我们先来测试一下开发板上的 GTH 模块工作是否正常。以下为 GTH 数据通信的具体测试步骤：

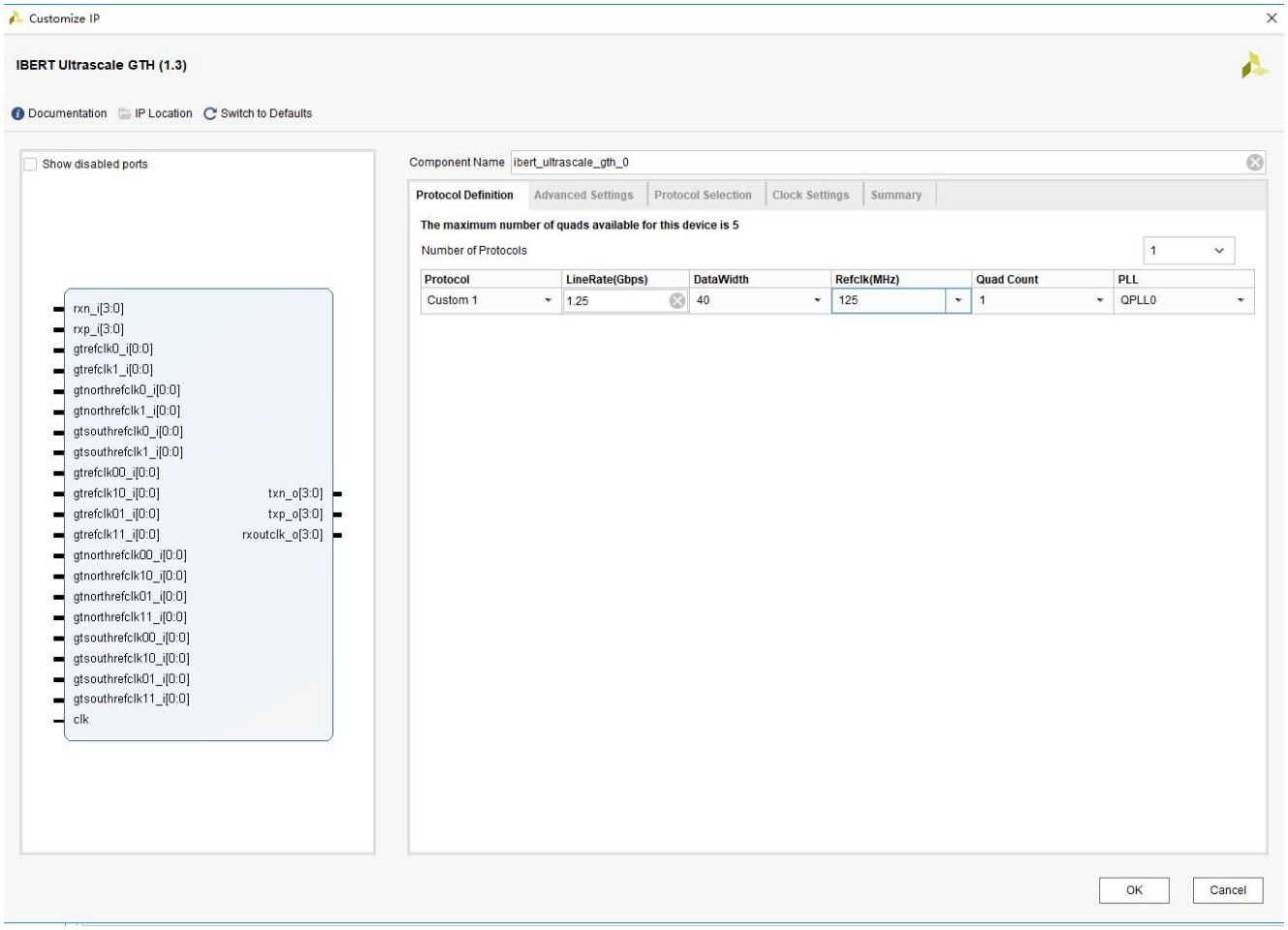
1. 新建一个工程，在 IP Catalog 搜索 ibert



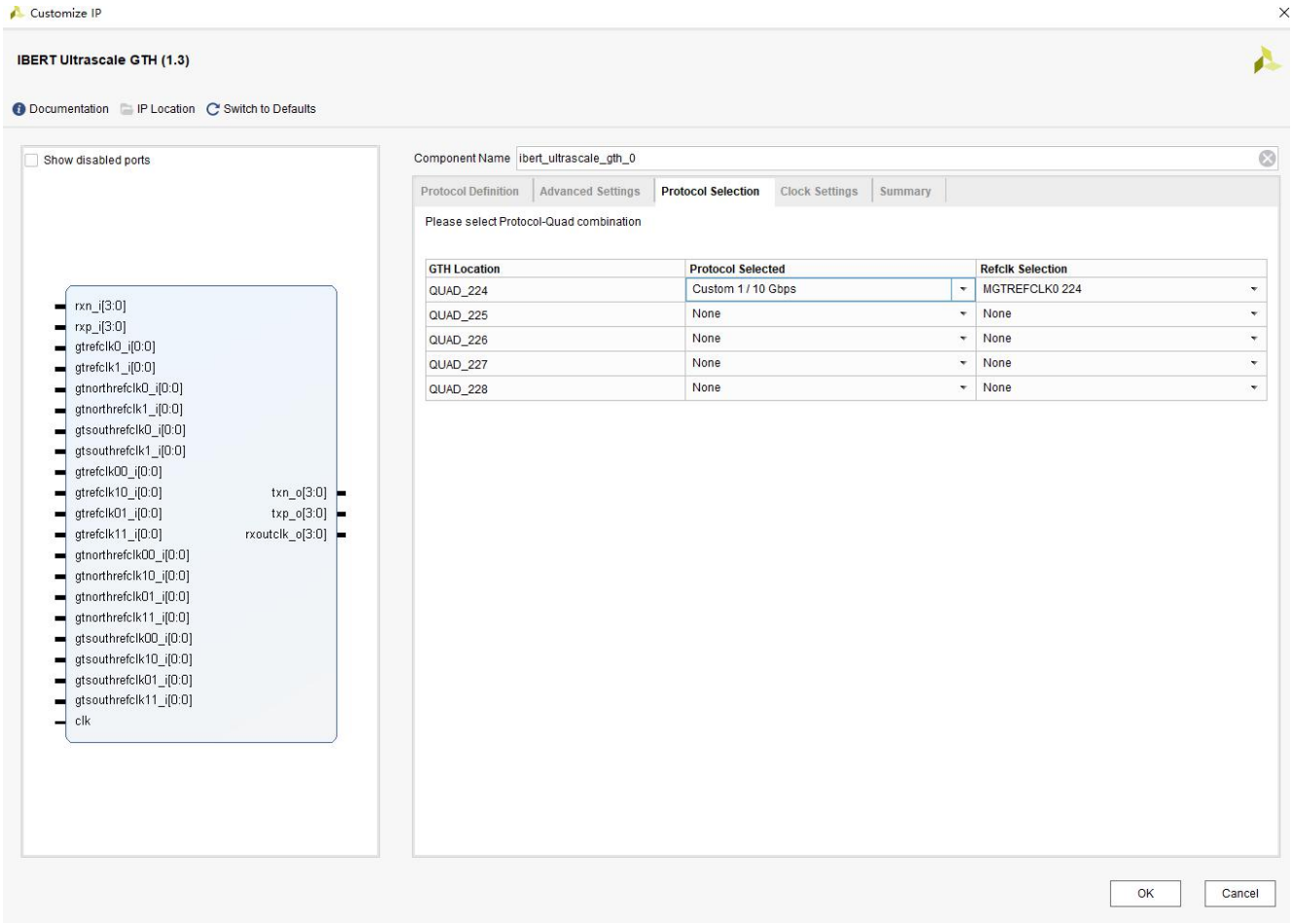
2. 在 Protocol Definition 页面中输入 LineRate 的速度,如果用户使用的是 10G 的光模块，这里可以最高的速率 12.5Gbps, 参考时钟为 125Mhz，LineRate 的频率为参考时钟的整数 80 倍。



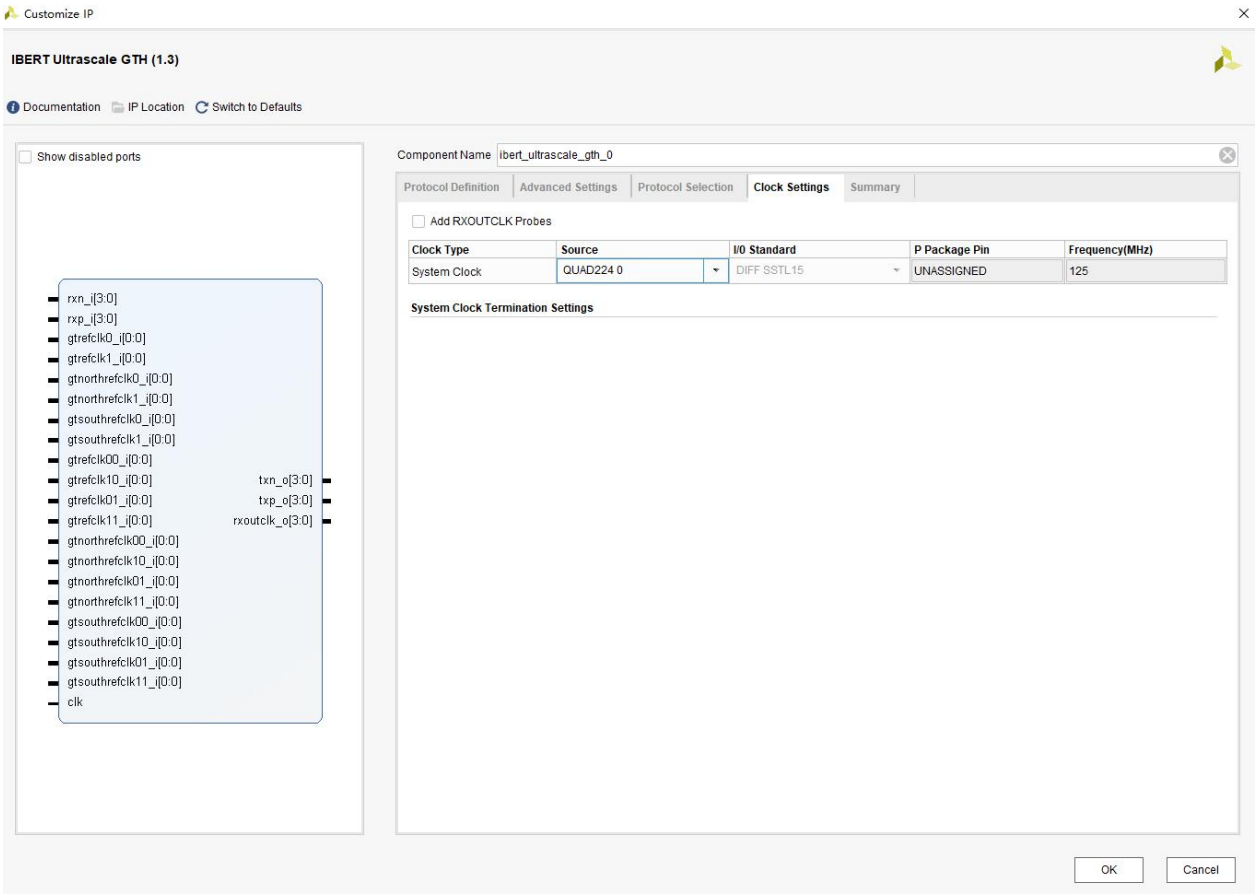
如果用户使用的是 10G或者1.25G 的光模块，这里的 LineRate 的速度选择为 1.25Gbps，10Gbps LineRate 的频率为参考时钟的整数 10 倍。



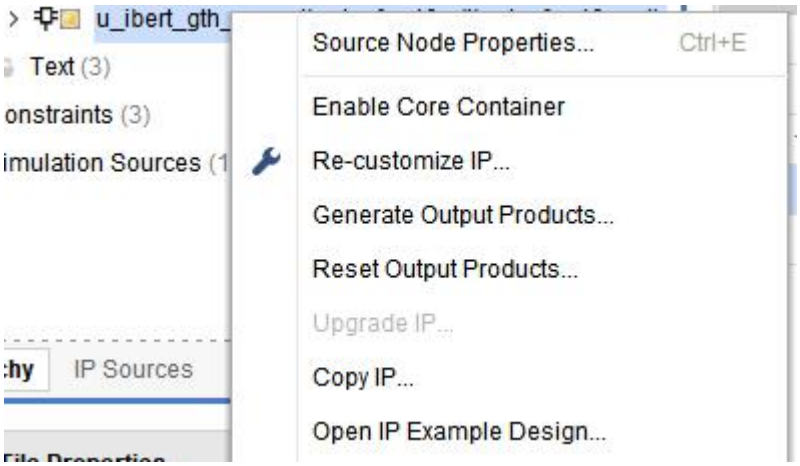
3. 在 Protocol Selection 界面里，选择 QUAD_224因为这里的4路光口挂到我们bank224的收发器通道上所以这里选择对应quad。



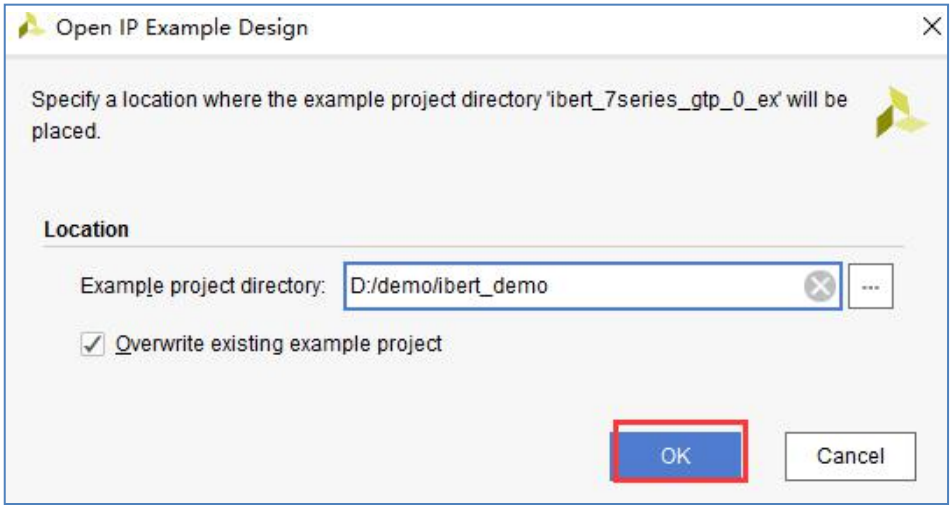
4. 最后clock settings 选择 quad224的的参考时钟其余选项默认 选择ok 进行创建



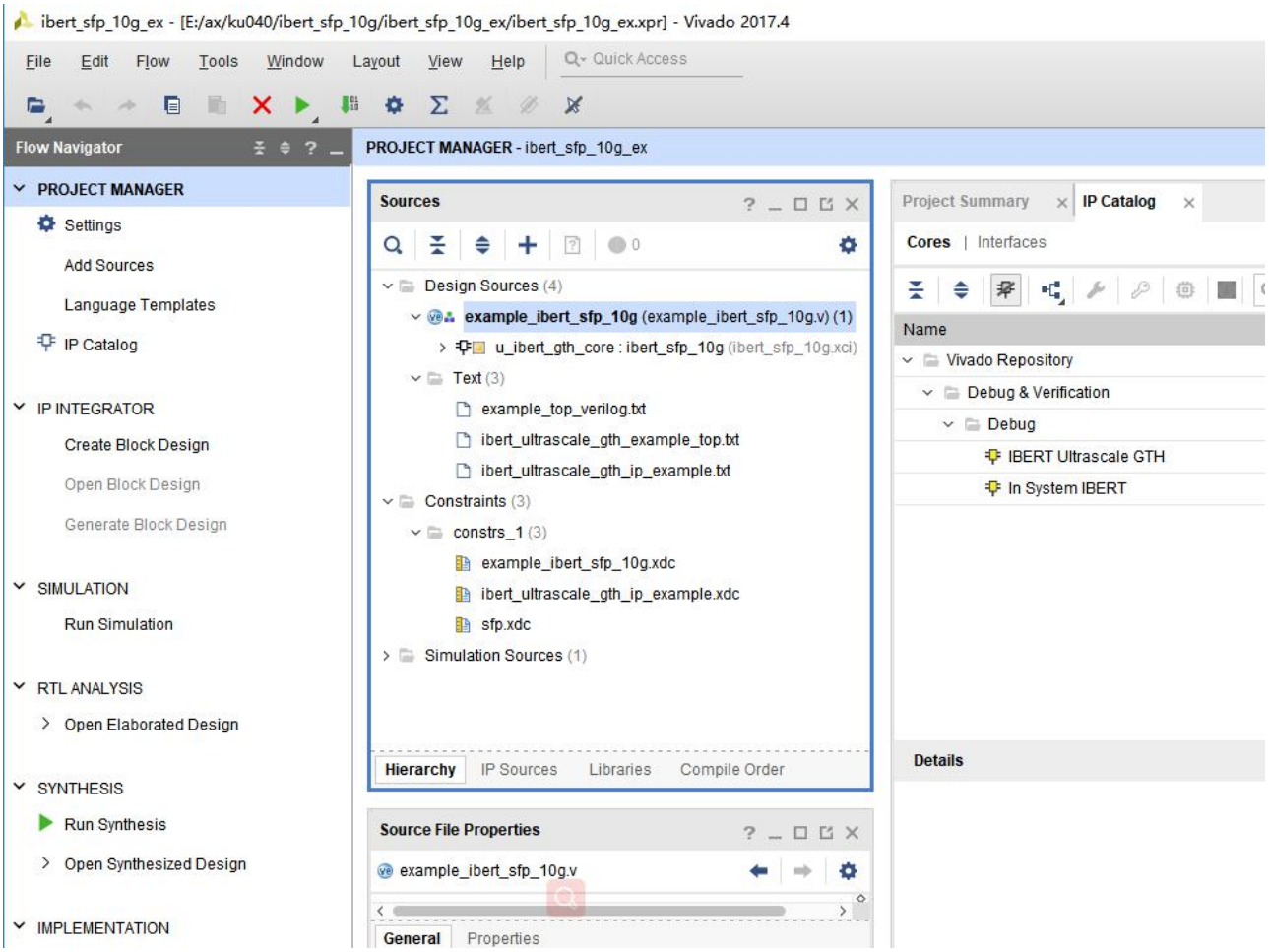
5. 右键选择 ibert IP，在弹出的下拉框中选择 Open IP Example Design。



6.再选择 Example project 的放置目录。



7.软件自动生成 example_ibert 的新项目，在这个项目中 verilog 程序和管脚约束文件都已经配置好了。



8.因为在 AXKU040 的硬件电路中，我们使用了 tx_disable 信号来允许/禁止 SFP 光模块的发送，所打开example_ibert 这里的文件名跟前面创建ip名称有关系，工程这里需要在 TOP 程序中定义 4个 tx_disable 信号并赋值为 0，一直使能 SFP 光模块的发送，

```
  output [3:0]                        tx_disable,
  // GT top level ports
  output [(4*`C_NUM_GTH_QUADS)-1:0]   gth_txn_o,
  output [(4*`C_NUM_GTH_QUADS)-1:0]   gth_txp_o,
  input  [(4*`C_NUM_GTH_QUADS)-1:0]     gth_rxn_i,
  input  [(4*`C_NUM_GTH_QUADS)-1:0]     gth_rxp_i,
  input  [`C_GTH_REFCLKS_USED-1:0]      gth_refclk0p_i,
  input  [`C_GTH_REFCLKS_USED-1:0]      gth_refclk0n_i,
  input  [`C_GTH_REFCLKS_USED-1:0]      gth_refclk1p_i,
  input  [`C_GTH_REFCLKS_USED-1:0]      gth_refclk1n_i
);
```

9. 再在 xdc 文件中添加如下管脚的约束。

```
set_property IOSTANDARD LVCMOS18 [get_ports {disable1[0]}]
set_property PACKAGE_PIN AM10 [get_ports  {disable1[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {disable1[1]}]
set_property PACKAGE_PIN AL10 [get_ports  {disable1[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {disable1[2]}]
set_property PACKAGE_PIN AP9 [get_ports  {disable1[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {disable1[3]}]
set_property PACKAGE_PIN AN9 [get_ports  {disable1[3]}]
```

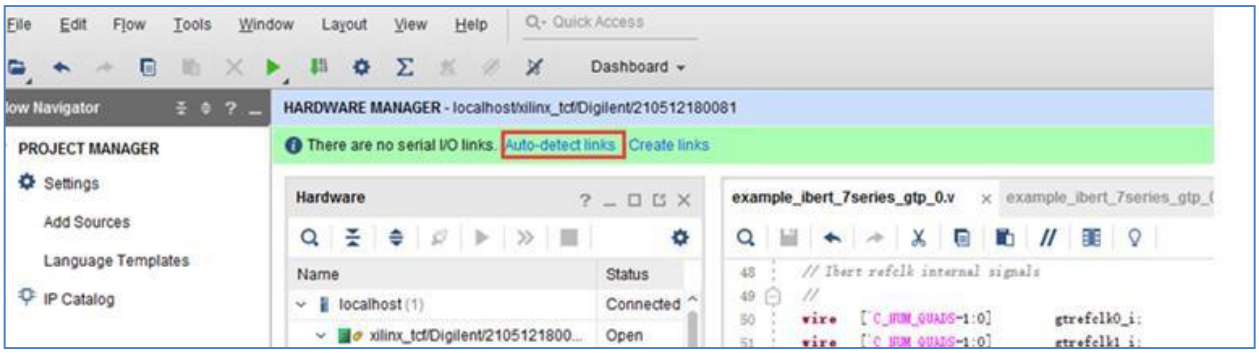
4 GTH 的眼图测试现象

因为 AXKU040 开发板自身是不带 SFP 的光模块和光纤的，所以测试之前需要自己准备 SFP 的光模块和光纤。因为光纤传输至少需要 2 个光模块，用户需要准备 2 个 SFP 光模块才是做光纤通信实验。10G 或者 1.25G SFP 的光模块和光纤再淘宝上都能购买到，在购买 SFP 光模块的同时，同时让商家提供配套的光纤就可以了。

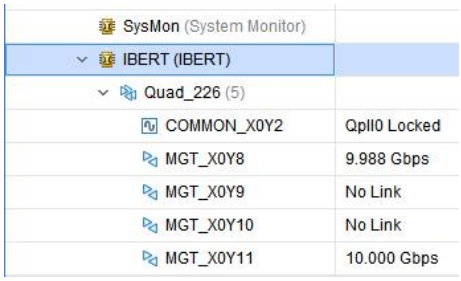
4.1 SFP 眼图测试现象

测试之前我们把 SFP 的光模块分别插到2个光模块的接口上

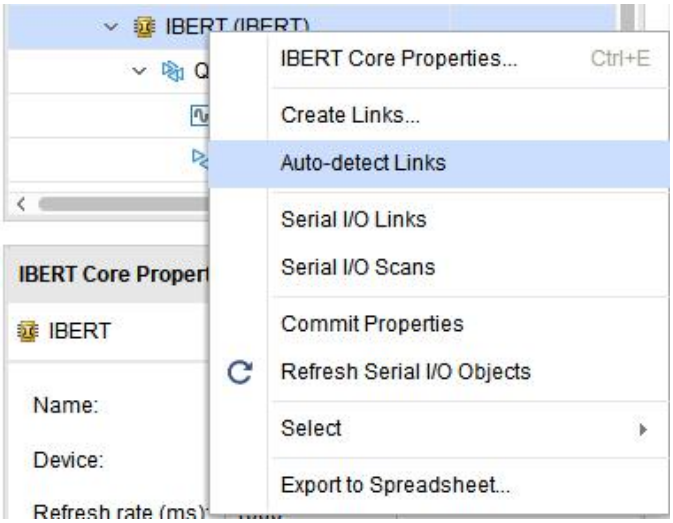
在 Vivado 软件里下载.bit 文件到 FPGA，下载后选择 Auto-detect links 软件会自动检测 Serial I/O Links。

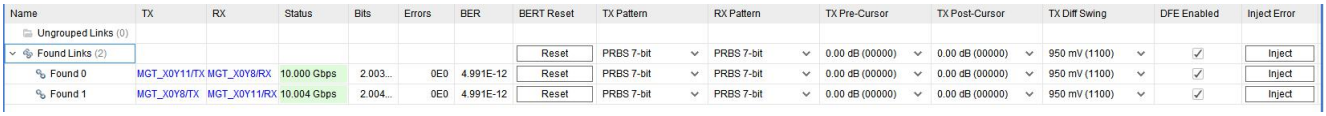


在 Serial I/O Links 界面会出现 2 路数据通信的情况，下图为 10 Gbps 连接速度的界面。



右键IBERT选择 Auto-detect Links，还显示链接的通道





Name	TX	RX	Status	Bits	Errors	BER	BERT Reset	TX Pattern	RX Pattern	TX Pre-Cursor	TX Post-Cursor	TX Diff Swing	DFE Enabled	Inject Error
Ungrouped Links (0)														
Found Links (2)							Reset	PRBS 7-bit	PRBS 7-bit	0.00 dB (00000)	0.00 dB (00000)	950 mV (1100)	✓	Inject
Found 0	MGT_X0Y11/TX	MGT_X0Y8/RX	10.000 Gbps	2.003...	0E0	4.991E-12	Reset	PRBS 7-bit	PRBS 7-bit	0.00 dB (00000)	0.00 dB (00000)	950 mV (1100)	✓	Inject
Found 1	MGT_X0Y8/TX	MGT_X0Y11/RX	10.004 Gbps	2.004...	0E0	4.991E-12	Reset	PRBS 7-bit	PRBS 7-bit	0.00 dB (00000)	0.00 dB (00000)	950 mV (1100)	✓	Inject

对这个界面，我们这里简单介绍一下，比如第一个 Found 0 Link，它由 MGT_X0Y11 通道的 TX 发送数据，由 MGT_X0Y8 通道的 RX 接收，数据 Link 的速度为 10Gbps。Bits 这列为发送的数据量

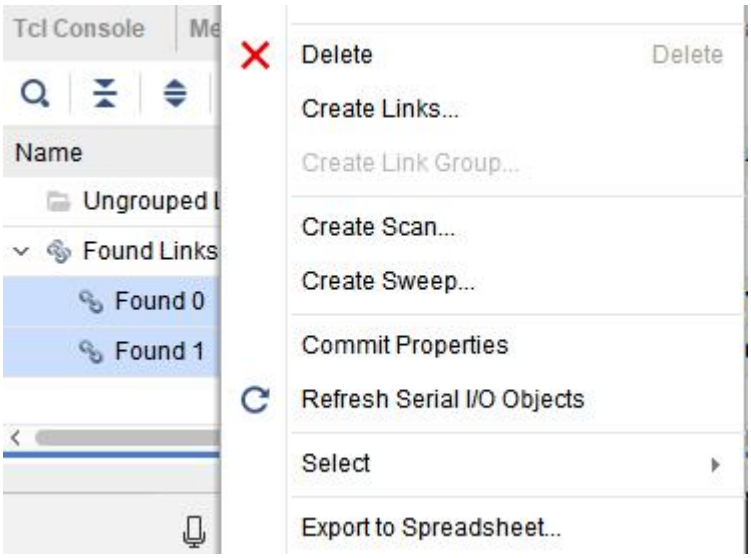
比如 1.21E-12 就是发送了 1.21x10 的 12 次方个数据），这个值随着时间的增加会不断增加。Error 项为错误的数据，这里我们看到的是 0，说明没有数据接收错误。BER 为误码率，Errors 的数量除以传输的数据数量就等于 BER 误码率。BERT Reset 是复位统计的数据，重新计数。TX Pattern 为发送的测试数据，默认为 PRBS 7bit，这里我们也可以选择其它的测试数据来测试光纤数据的传输。

可能大家还不太清楚 MGT_X0Y8~MGT_X0Y11 各自代表那个光纤通道，在.xdc 文件里，有如下的定义：

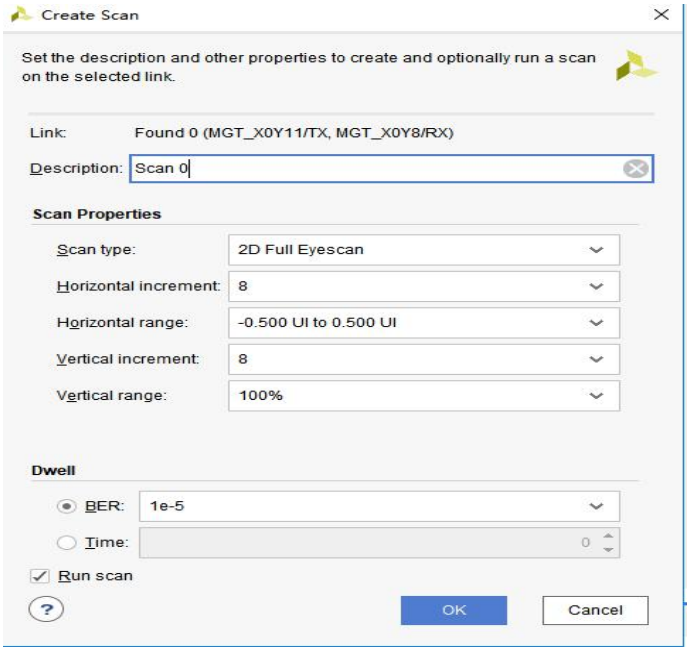
```
# GT X0Y8
set_clock_groups -asynchronous -group [get_clocks -of_objects [get_pins {u_ibert_gth_core/inst/QUAD[0].u_q/CH[0].u_ch/u_gthe3_channel/RXOUTCLK}] -include_generated_clocks]
set_clock_groups -asynchronous -group [get_clocks -of_objects [get_pins {u_ibert_gth_core/inst/QUAD[0].u_q/CH[0].u_ch/u_gthe3_channel/TXOUTCLK}] -include_generated_clocks]
# GT X0Y9
set_clock_groups -asynchronous -group [get_clocks -of_objects [get_pins {u_ibert_gth_core/inst/QUAD[0].u_q/CH[1].u_ch/u_gthe3_channel/RXOUTCLK}] -include_generated_clocks]
set_clock_groups -asynchronous -group [get_clocks -of_objects [get_pins {u_ibert_gth_core/inst/QUAD[0].u_q/CH[1].u_ch/u_gthe3_channel/TXOUTCLK}] -include_generated_clocks]
# GT X0Y10
set_clock_groups -asynchronous -group [get_clocks -of_objects [get_pins {u_ibert_gth_core/inst/QUAD[0].u_q/CH[2].u_ch/u_gthe3_channel/RXOUTCLK}] -include_generated_clocks]
set_clock_groups -asynchronous -group [get_clocks -of_objects [get_pins {u_ibert_gth_core/inst/QUAD[0].u_q/CH[2].u_ch/u_gthe3_channel/TXOUTCLK}] -include_generated_clocks]
# GT X0Y11
set_clock_groups -asynchronous -group [get_clocks -of_objects [get_pins {u_ibert_gth_core/inst/QUAD[0].u_q/CH[3].u_ch/u_gthe3_channel/RXOUTCLK}] -include_generated_clocks]
set_clock_groups -asynchronous -group [get_clocks -of_objects [get_pins {u_ibert_gth_core/inst/QUAD[0].u_q/CH[3].u_ch/u_gthe3_channel/TXOUTCLK}] -include_generated_clocks]
```

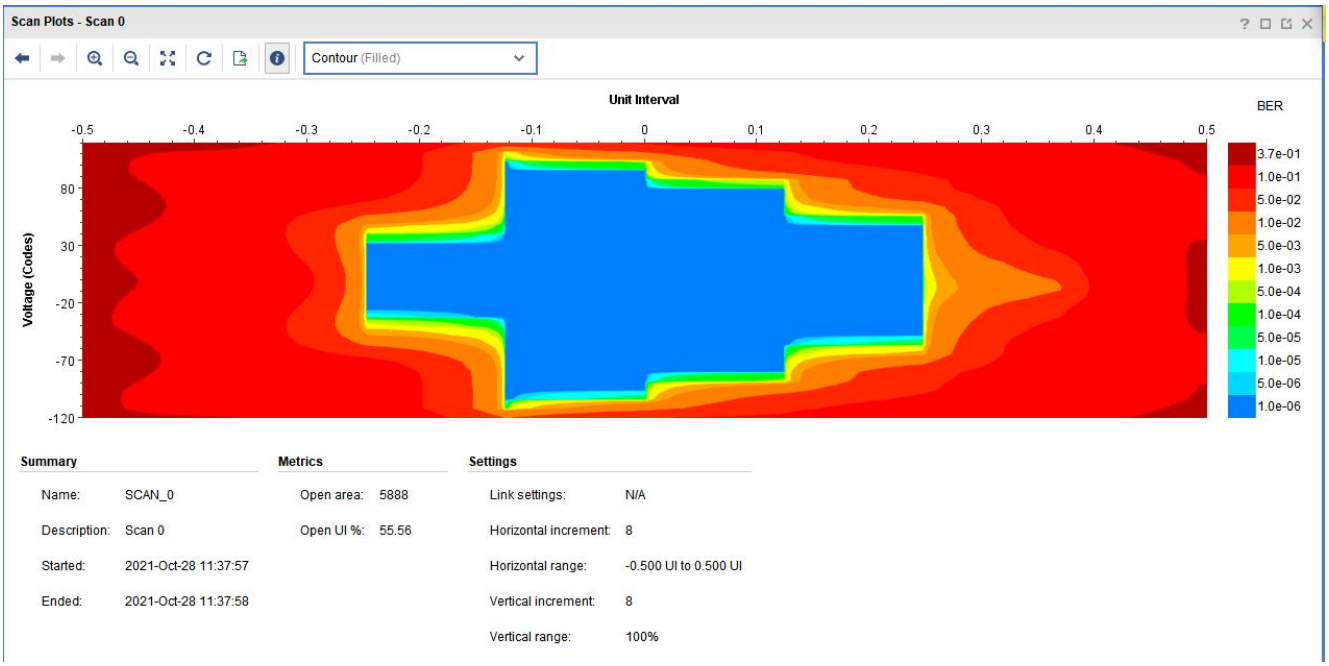
接下去我们来测试光纤通信的电眼图，一般情况，测试板上的电眼图需要配合高端示波器和差分探头才能测量。但这里我们不需要外接任何设备或仪器，就可以测量光纤数据通信的眼图情

况，极大的方便了 FPGA 高速串行通信的软硬件调试。右键点击我们想看的通道，比如我们这里选择第一路 Found 1(MGT_ X0Y11 发，MGT_ X0Y8 接收），再在下拉菜单里选择 Create Scan...



Create scan这里默认设置就好了选择ok





眼图中颜色越蓝的地方，BER值越小，说明这个区域误码率越低，或者几乎没有误码率。颜色越红，表示这个区域误码率越高。一般来讲，这个眼图的眼睛张的越开，说明数据传输信号越好。从上张图片可以看出，Link的速度越低，对应的眼图也会更好，Link的速度越高，对应的眼图会下降，这对硬件设计和PCB设计提出了更高的要求。对眼图的介绍我们这里不做多讲，大家自己百度搜索一下查找相关的资料。用同样的方法大家也可以分别看一下其它几路Link的眼图情况。