

GTP 光纤通信测试例程

黑金动力社区 2019-04-25

1 实验简介

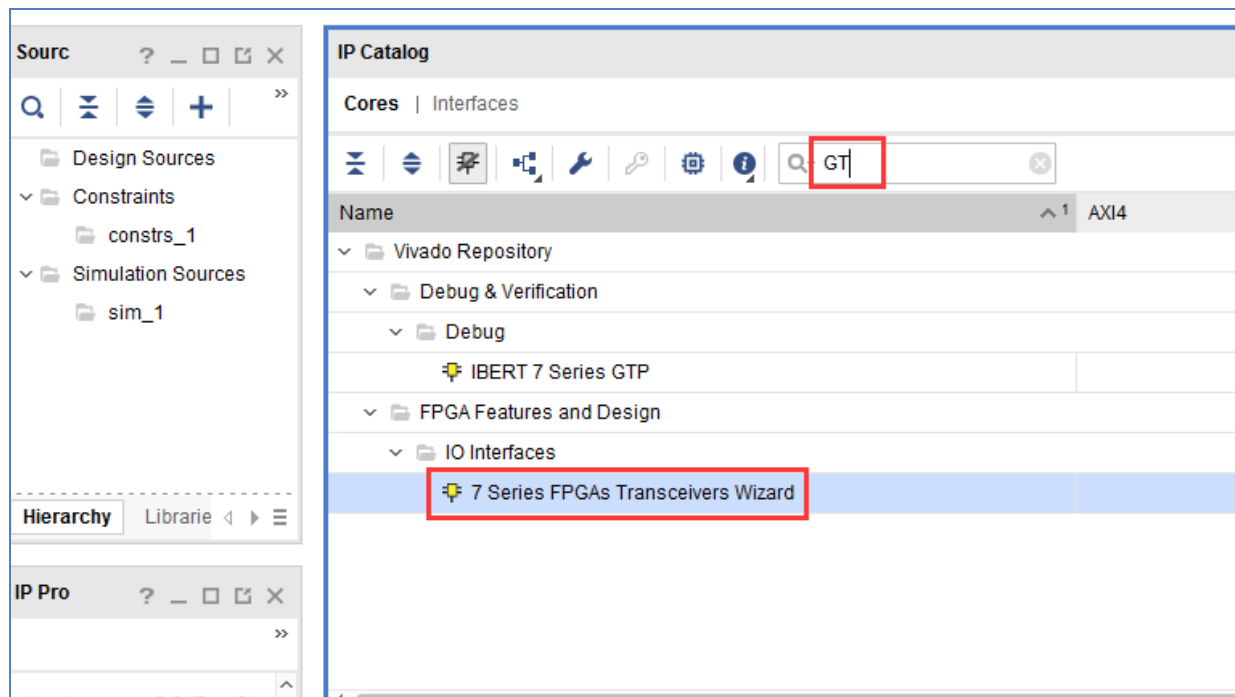
本实验将介绍通过光纤实现数据通信的传输，测试数据由 FPGA 自身产生，由 GTP 发送到第一路光纤口，然后通过光纤线环路到第二路光纤口，通过 GTP 接收数据进行校验。

2 实验原理

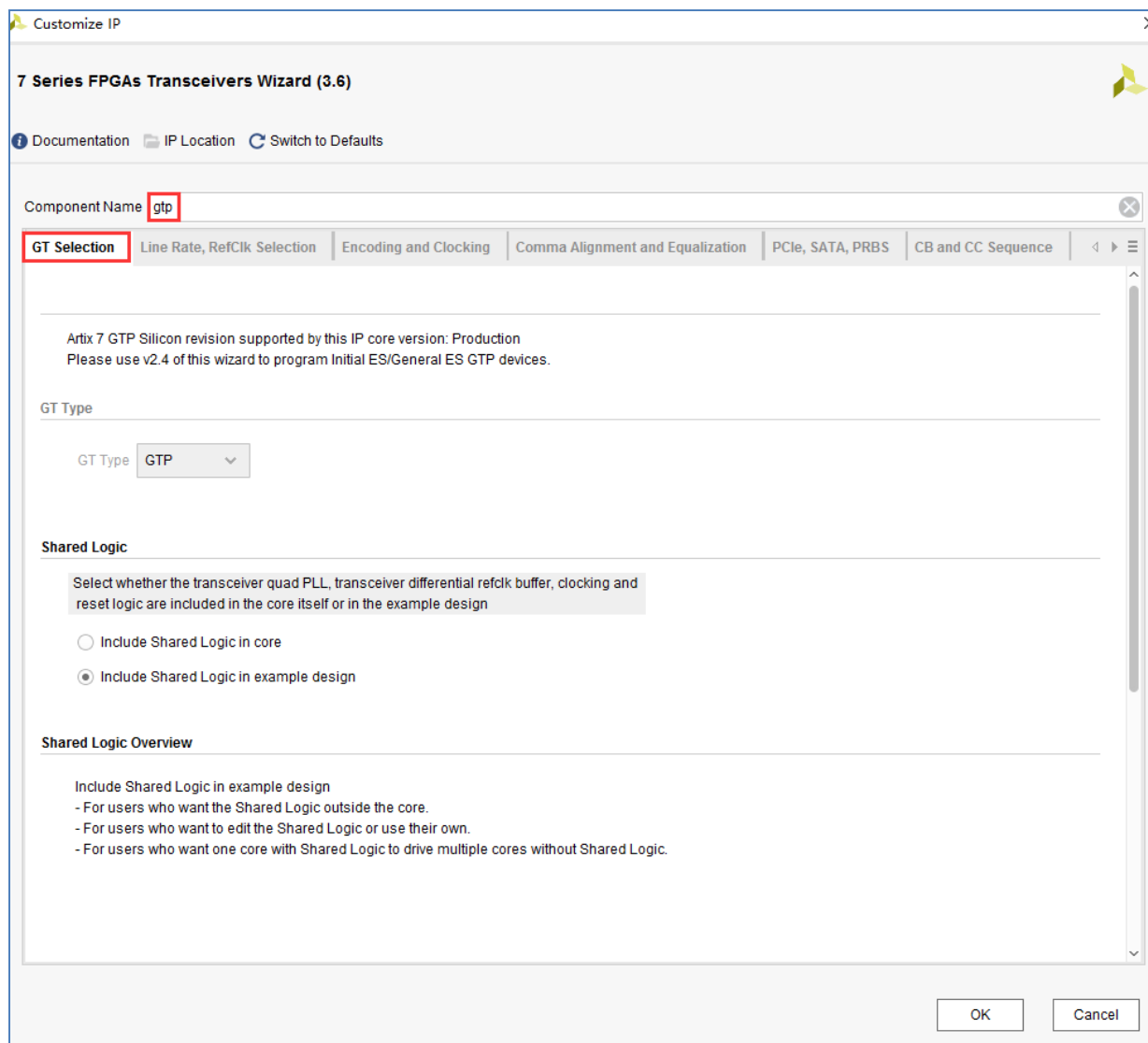
2.1 GTP IP 设计

XILINX 的 Vivado 软件已经为用户设计好了 GTP IP，用户无需关心 GTP 的内部具体工作就可以使用 IP 来实现 GTP 的高速的数据收发。下面我具体的 GTP IP 的生成和配置方法：

1. 在 IP Catalog 界面中双击 FPGA Features and Design\IO Interface 目录下的"7 Series FPGAs Transceivers Wizard"图标。



2. Component Name 栏输入"gtp"为取名，在 GT selection 界面里无需修改，保留默认。



3. 在 Line Rate, Refclk Selection 界面里, 首先设置 GTP 的传输协议位 “Aurora 8B10B single lane 4byte”, 我们在前面一章讲过, Xilinx 的 GTP 是支持很多种协议的, Aurora 8B/10B 协议是一个可扩展的、轻量级的链路层协议, 可以用于通过一条或多条串行链路将数据点到点传输。这里我们用的光模块传输是单路的, 所以选择 single lane, 数据接口为 4byte, 就是 32 位数据。再选择 TX 和 RX 的 Line Rate 速度, 这个 Line Rate 速度是需要是 GTP 参考时钟的整数倍, 开发板上的 GTP 参考时钟为 125Mhz, 这里我们 Line Rate 为参考时钟的 10 倍, 所以 Line Rate 设置为 1.25Gbps, 如果用户需要设置其它时钟比如 5Gbps, 只需要直接修改。Reference Clock 为 125Mhz。

Customize IP

7 Series FPGAs Transceivers Wizard (3.6)

Documentation IP Location Switch to Defaults

Component Name

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

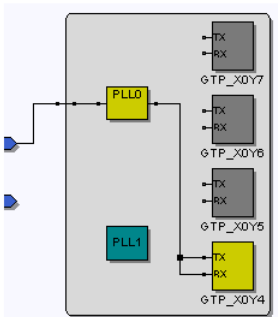
Protocol **aurora 8b10b single lane 4byte**

TX **RX**

Line Rate (Gbps) [0.5 - 3.75] ☐ TX off Line Rate (Gbps) [0.5 - 3.75] ☐ RX off

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

gt row ☐ Use Common DRP



PLL Selection

TX RX ☐ Extend reset to 3 ms

Transceiver Selection

☐ Use GTP X0Y5

TX Clock Source

RX Clock Source

☐ Advanced Clocking Option

默认软件只有使能一个 Channel (GTP_X0Y4) 的 GTP, 这里要分别选中 GTP_X0Y5, GTP_X0Y6, GTP_X0Y7, 然后选择右边的"Use GTP X0Y5/6/7"前面的钩, 这样 PLL0 都连接到了 4 个 GTP Channel 模块。

7 Series FPGAs Transceivers Wizard (3.6)

Documentation IP Location Switch to Defaults

Component Name

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

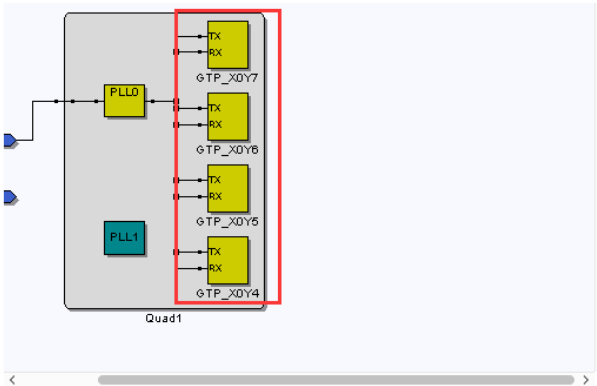
Protocol

TX **RX**

Line Rate (Gbps) [0.5 - 3.75] ☐ TX off Line Rate (Gbps) [0.5 - 3.75] ☐ RX off

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

gt row ☐ Use Common DRP



PLL Selection

TX RX ☐ Extend reset to 3 ms

Transceiver Selection

☒ Use GTP X0Y7

TX Clock Source

RX Clock Source

☐ Advanced Clocking Option

☐ PRBS pattern generator and checker

☐ Vivado Lab Tools

Active Transceivers = 4

另外 TX Clock Source 和 RX Clock Source 需要都选择 REFCLK0 Q0, 因为电路设计上 125Mhz 的参考时钟是连接到 REFCLK0 的管脚上的。

Component Name: gtp

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): 1.25 [0.5 - 3.75] ☐ TX off

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

gt row: Top Row ☐ Use Common DRP

RX

Line Rate (Gbps): 1.25 [0.5 - 3.75] ☐ RX off

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

PLL Selection

TX: PLL0 RX: PLL0 ☐ Extend reset to

Transceiver Selection

☒ Use GTP X0Y5

TX Clock Source: REFCLK0 Q0

RX Clock Source: REFCLK0 Q0

☐ Advanced Clocking Option

☐ PRBS pattern generator and checker

☐ Vivado Lab Tools

Diagram: A block diagram showing a Quad1 component with four transceivers (GTP_X0Y4, GTP_X0Y5, GTP_X0Y6, GTP_X0Y7) connected to PLL0 and PLL1. The PLL0 is connected to the TX and RX of all four transceivers. The PLL1 is connected to the TX and RX of GTP_X0Y4 and GTP_X0Y5.

4. 在 Encoding and Clocking 界面里，设置 TX 和 RX 的外部数据宽度，8B/10B 使能，内部数据宽度等信息，这里保留默认配置。因为外部数据宽度是 32，是内部数据宽度的 2 倍，所以这里 GTP 的内部时钟频率是外部接口的 2 倍。

Customize IP

7 Series FPGAs Transceivers Wizard (3.6)

Documentation IP Location Switch to Defaults

Component Name: gtp

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

TX **RX**

External Data Width (Bits) 32 External Data Width (Bits) 32

Encoding 8B/10B Decoding 8B/10B

Internal Data Width (Bits) 20 Internal Data Width (Bits) 20

☒ Use DRP DRP/System Clock Frequency (MHz) 100 [0.0 - 156.0]

Optional Ports

☐ TXBYPASS8B10B ☒ TXCHARDISPMODE ☒ TXCHARDISPVAL

☒ RXCHARISCOMMA ☒ RXCHARISK ☐ RXSTARTOFSEQ

Synchronization and Clocking

TX **RX**

☒ Enable TX Buffer ☒ Enable RX Buffer

TXUSRCLK Source TXOUTCLK RX Buffer Bypass Mode Auto

TXOUTCLK Source ☐ Use TXPLLREFCLK RXUSRCLK Source TXOUTCLK

RXOUTCLK Source ☐ Use RXPLLREFCLK

Optional Ports

☒ TXPCSRESET ☒ TXPMARESET ☐ TXSYSCLKSEL ☐ TXRATE ☒ TXBUFSTATUS ☐ TX8B10BEN

☒ RXPCSRESET ☒ RXPMARESET ☐ RXSYSCLKSEL ☐ RXRATE ☒ RXBUFSTATUS

☒ RXBLERRESET ☒ RXCDRPHOLD ☐ SIGNALIDCLK ☐ PLL0PD ☐ PLL1PD

4. 在 Comma Alignment and Equalization 界面里，选择默认设置。这里选择 Comma 值为 K28.5, K28.5 是一种用以表示 Fibre Channel 操作开始的特殊 10 比特字符。8B/10B 编码中将 K28.5 作为 K 码的控制字符，称为“comma”，所以可以用 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。

Customize IP

7 Series FPGAs Transceivers Wizard (3.6)

Documentation IP Location Switch to Defaults

Component Name:

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

☒ Use comma detection Comma Value: Comma Mask:

☐ Decode valid comma only Plus Comma: Align to:

☐ Combine plus/minus commas (double-length comma) Minus Comma:

Optional Ports

☒ ENPCOMMAALIGN (Enables positive Comma Alignment) ☒ ENMCOMMAALIGN (Enables negative Comma Alignment)

☐ RXSLIDE ☒ RXBYTEISALIGN ☒ RXBYTEREALIGN ☒ RXCOMMADET

Termination and Equalization

Differential Swing and Emphasis Mode:

RX Equalization

Mode: Automatic Gain Control:

RX Termination

Voltage: Trim Value (mV):

Optional Ports

☒ TXPOLARITY ☐ TXINHIBIT ☒ TXDIFFCTRL ☒ TXPOSTCURSOR ☒ TXPRECURSOR ☒ TXMAINCURSOR

☒ RXPOLARITY

OK

5. PCIe,SATA, PRBS 页面无需修改，保持默认设置。

Customize IP

7 Series FPGAs Transceivers Wizard (3.6)

[Documentation](#) [IP Location](#) [Switch to Defaults](#)

Component Name:

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

PCIe Express and SATA

☐ Enable PCI Express

SATA COM sequence

Bursts: [0 - 7] Idles: [0 - 7]

PCI Express Parameters

Transition Time

To P2: [0 - 255] From P2: [0 - 4095] To/From Non P2: [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

☐ Use RX OOB Signal Detection

PRBS

☒ Use PRBS Detector ☒ Use Port TXPRBSSEL ☒ Use Port TXPRBSFORCEERR ☐ RXPRBS_LOOPBACK

6. 在 CB and Sequence 页面中不用修改。

Component Name

gtp

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

1

Sequence length

1

Clock correction

☒ Use Clock Correction

PPM Offset +/-

100

[-1250 - 1250]

☐ Use Two Clock Correction Sequences

Periodicity of the CC sequence (bytes)

5000

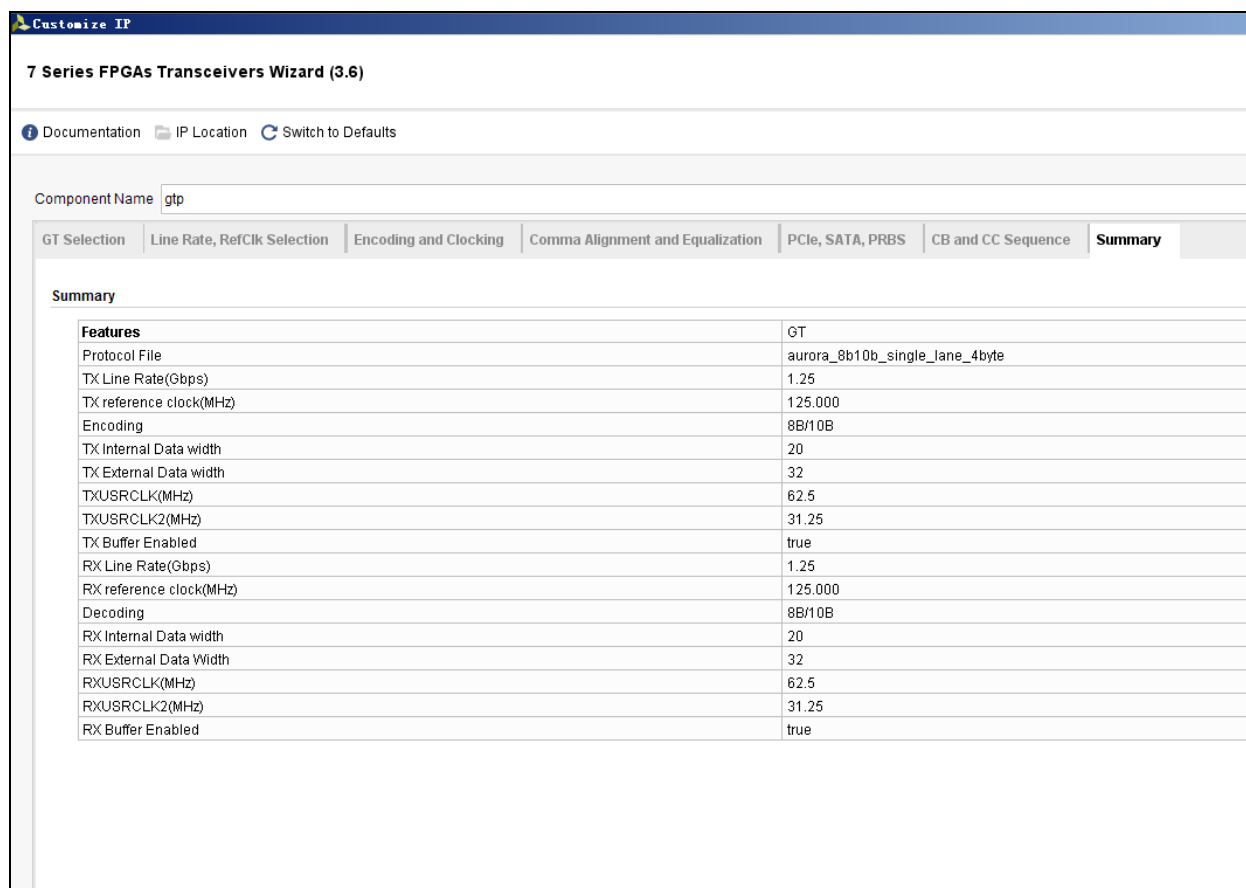
Sequence length

4

Sequence Definition

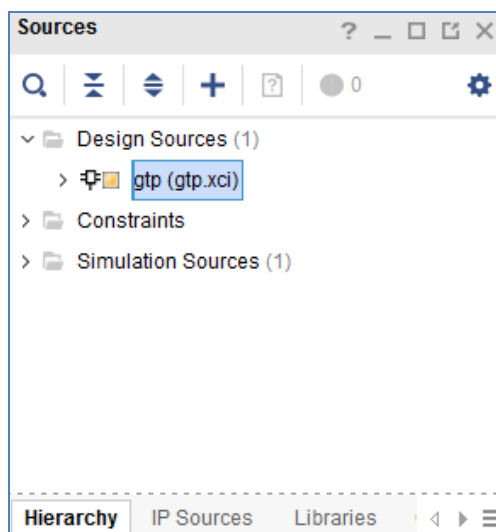
	Sequence	K Character	Inverted Disparity	Don't Care
Sequence1, Byte1	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sequence1, Byte2	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sequence1, Byte3	11110111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

7. 在 Summary 界面中检查一下配置情况，点击 OK 完成。

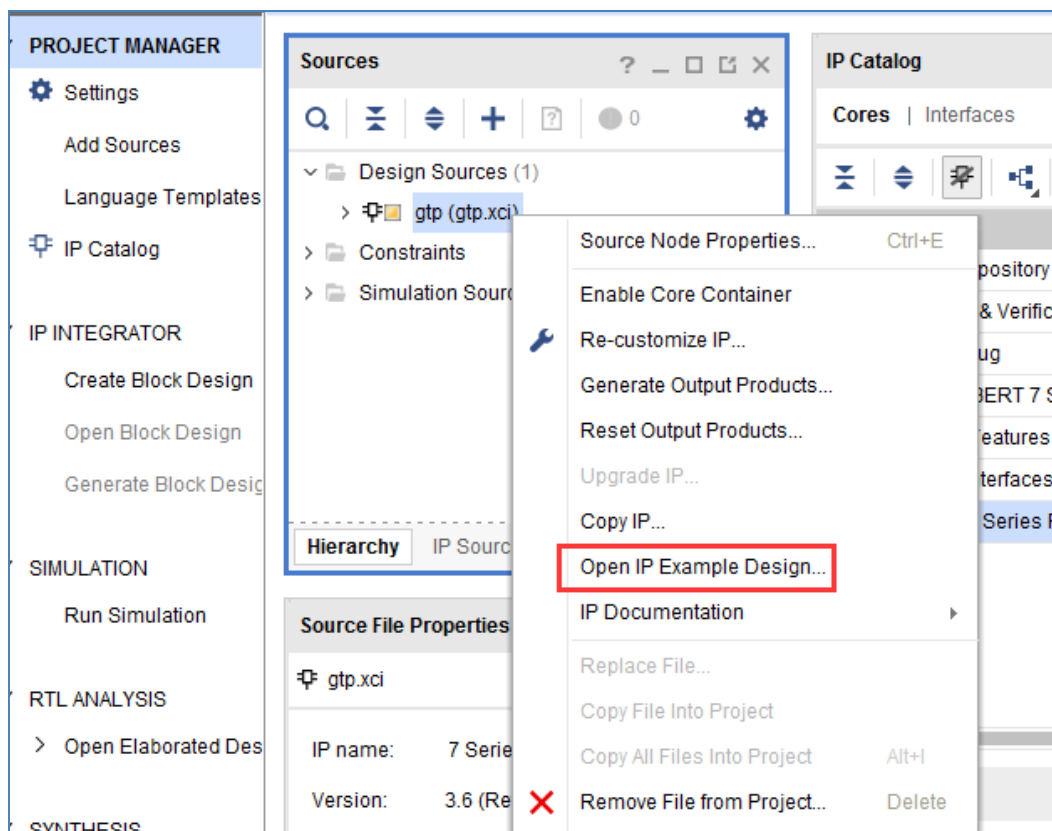


关于更多 GTP IP 的配置信息，大家请参考 Xilinx 提供的文档“pg168-gtwizard.pdf”和“ug482_7Series_GTP_Transceivers.pdf”，这两个文档上有对 GTP IP 各个配置参数做了详细的介绍。

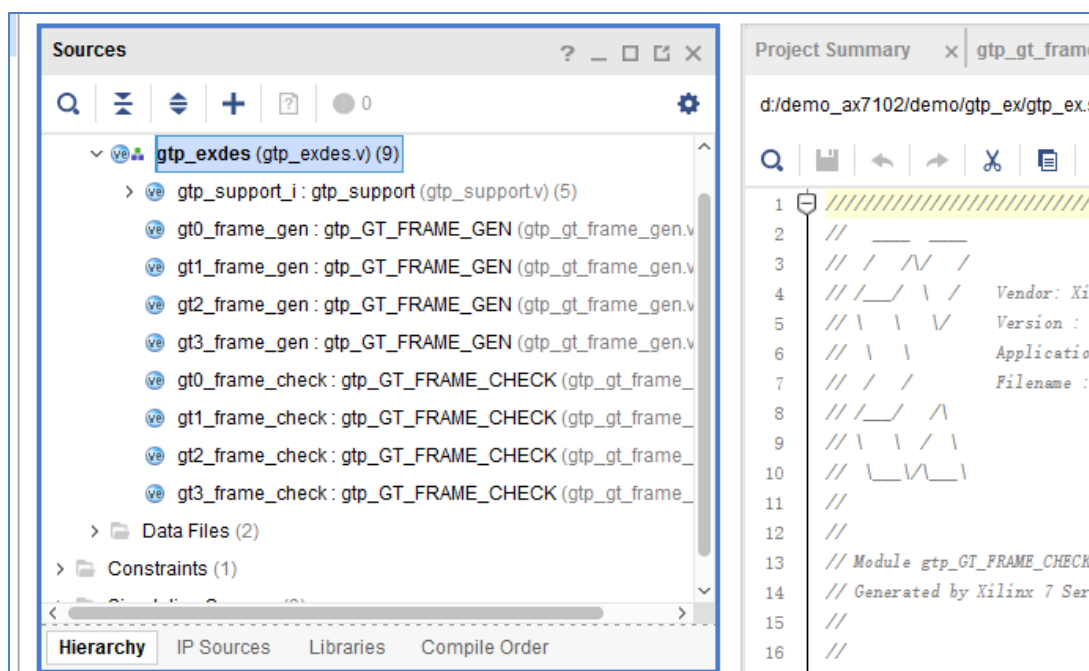
8. 配置完成在工程中自动添加刚刚生成 gtp 的 IP。



9. 我们来生成 GTP IP 的 example 工程，右键选择 gtp,在下拉菜单里选择"Open IP Example Design..."。



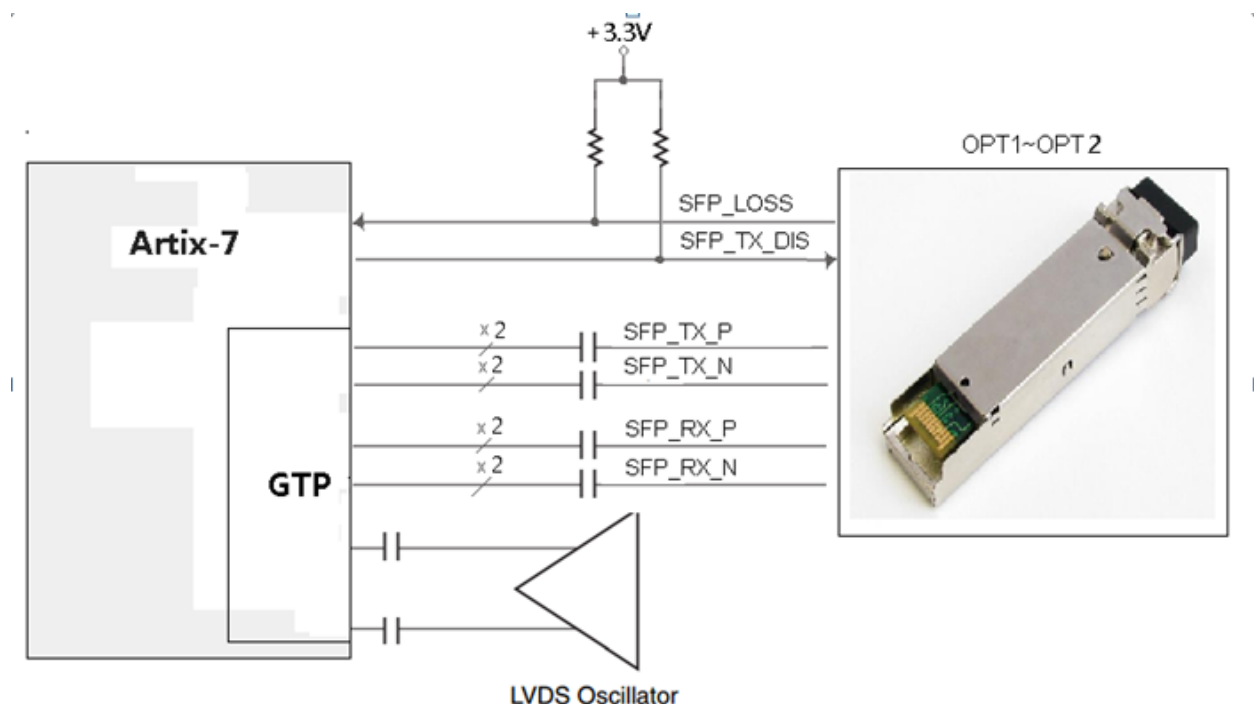
10. 生成的 example 工程如下图所示，在这个例子工程中，程序会在 gt0_frame_gen 模块中产生测试数据进行 GTP 的数据传输，在 gt0_frame_check 模块接收并检查是否正确，如果不正确，错误统计值增加。



关于 example 的工程代码和测试我们这里不做介绍，大家自己去看去测试就可以了。在后面的 GTP 数据传输的软件开发中我们会用到这里 example 工程中的一些文件和 IP。

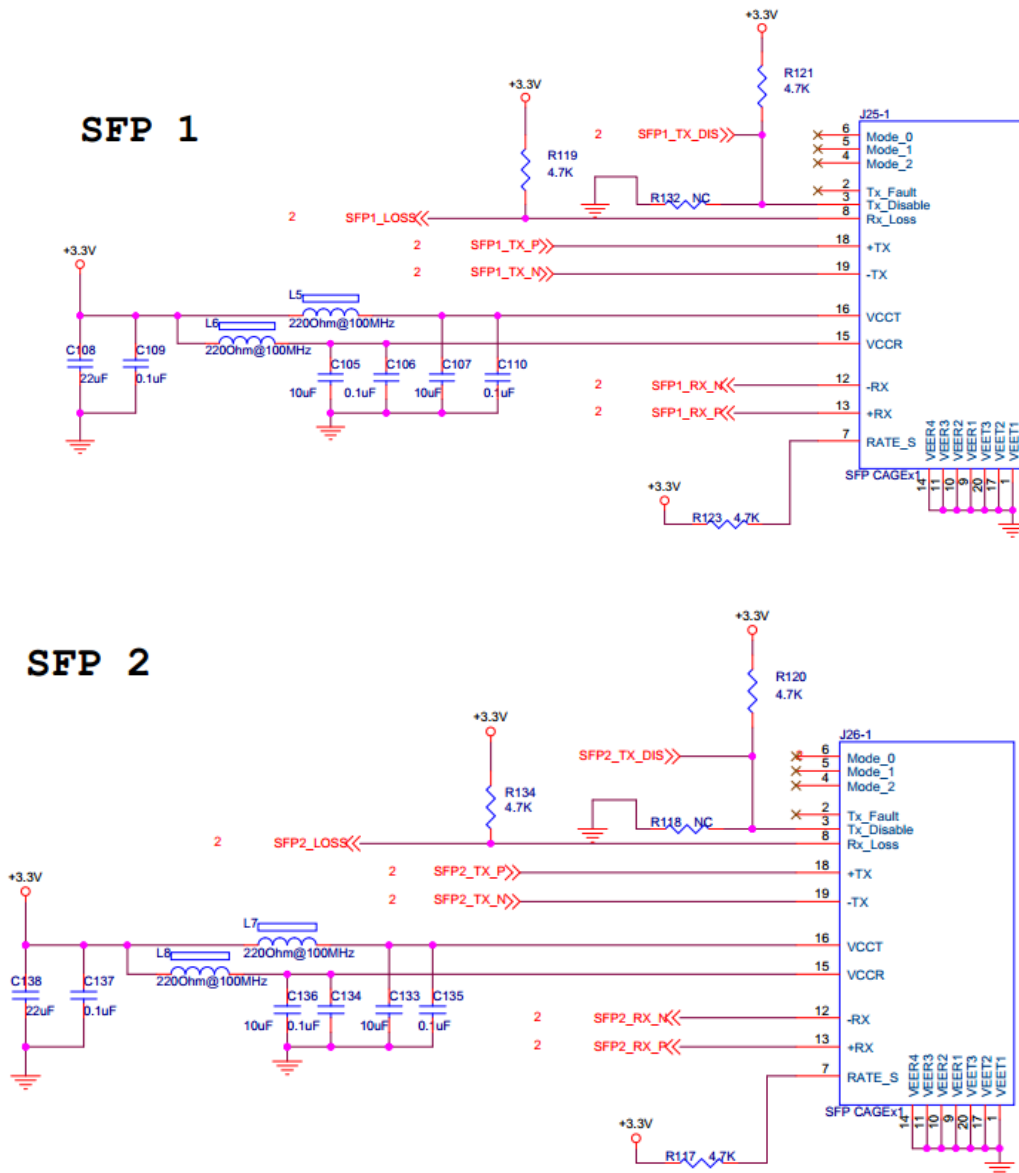
2.2 硬件介绍

在 AX7102 开发板上，有 2 路光纤接口 OPT1~OPT2, 分别连接到 FPGA 芯片的 GTP 的通道上。FPGA 和光纤连接的设计示意图如下图所示：



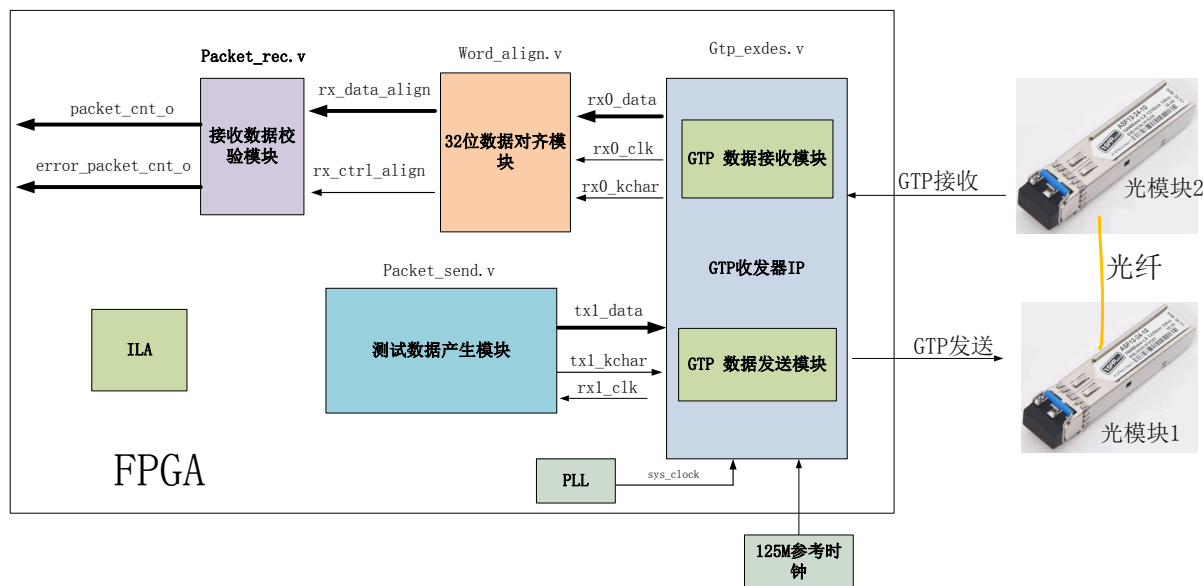
其中 OPT1 光模块接口连接到 GTP 的 Channel0 上，OPT2 跟 GTP 的 Channel1 相连。光模块和 FPGA 之间用 0.1uf 的电容隔开，使用 AC Couple 的模式。

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：



3 程序设计

光纤数据传输的 FPGA 程序设计是在 example 的工程代码的基础上添加了一个 TOP 文件和 3 个.v 文件，工程的逻辑框图如下图所示：



程序产生数据使用 GTP IP 发送给外部的光模块 1，光模块 1 把电信号转换成光信号通过光纤传输到光模块 2，光模块 2 又把光信号转换成电信号输入到 FPGA 的 GTP 接收，GTP 接收到的数据需要做一个 32 位数据对齐之后，并解析出数据信号和控制数据信号，校验模块对这些数据信号和控制信号进行校验计算判断接收数据和发送数据是否一致。

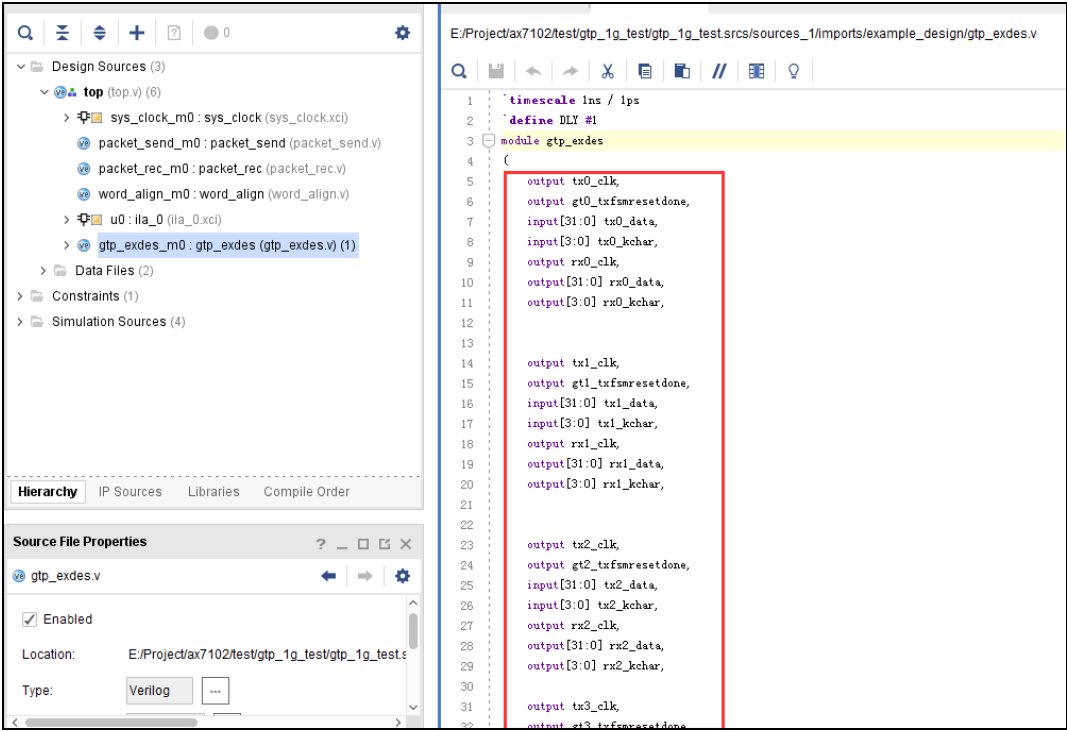
光纤数据传输设计好的工程如下图所示：



这里工程添加了 ILA 工具可以用来查看接收到的数据。

1) gtp 数据通信模块

在前面我们已经生成过 gtp IP 的 example 工程，这里删除了 `gt0_frame_gen.v` 模块和 `gt0_frame_check.v` 模块。因为这两个是测试数据的产生和检查模块，本例程用不上。另外对 `gtp_exdes.v` 文件进行修改，主要是删除 `gt0_frame_gen.v` 模块和 `gt0_frame_check.v` 模块的例化，再在模块的 Port 处添加以下的 4 个 channel 的用户接口信号，添加后的四个通道的信号如下图所示：



再把端口定义的四个 Channel 的信号和 gtp_support 子模块里的信号进行连接。

```

650
651 assign tx0_clk = gtp0_txusrclk2_i;
652 assign rx0_clk = gtp0_rxusrclk2_i;
653 assign rx0_data = gtp0_rxdata_i;
654 assign rx0_kchar = gtp0_rxcharisk_i;
655 assign gtp0_txfsresetdone = gtp0_txfsresetdone_i;
656
657
658 assign tx1_clk = gtp1_txusrclk2_i;
659 assign rx1_clk = gtp1_rxusrclk2_i;
660 assign rx1_data = gtp1_rxdata_i;
661 assign rx1_kchar = gtp1_rxcharisk_i;
662 assign gtp1_txfsresetdone = gtp1_txfsresetdone_i;
663
664 assign tx2_clk = gtp2_txusrclk2_i;
665 assign rx2_clk = gtp2_rxusrclk2_i;
666 assign rx2_data = gtp2_rxdata_i;
667 assign rx2_kchar = gtp2_rxcharisk_i;
668 assign gtp2_txfsresetdone = gtp2_txfsresetdone_i;
669
670 assign tx3_clk = gtp3_txusrclk2_i;
671 assign rx3_clk = gtp3_rxusrclk2_i;

```

下面对在 gtp_exdes.v 端口中添加的几个 GTP 用户接口信号做一下介绍，以下以 channel0 的 GTP 接口为例：

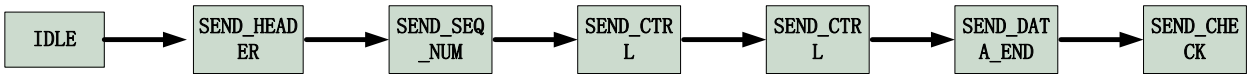
信号名	位数	输入输出	说明
tx0_clk	1	输出	数据发送时钟，也就是第十四部分介绍的 TXUSRCLK2，

			频率为 GTP 的参考时钟 125Mhz。数据在上升沿有效。
tx0_data	32	输入	GTP 发送数据。
tx0_kchar	4	输入	<p>GTP 发送的 K 控制字，用来指示发送的数据是 K 码控制字符还是正常传输数据。高电平表明是 K 码控制字符，4 位对应发送数据的 4 个 Byte。</p> <p>tx0_kchar [3] 对应 tx0_data [31:24]</p> <p>tx0_kchar [2] 对应 tx0_data [23:16]</p> <p>tx0_kchar [1] 对应 tx0_data [15:8]</p> <p>tx0_kchar [0] 对应 tx0_data [7:0]</p>
rx0_clk	1	输出	数据接收时钟，也就是第十四部分介绍的 RXUSRCLK2，频率为 GTP 的参考时钟 125Mhz。数据在上升沿有效。
rx0_data	32	输出	GTP 接收数据。
rx0_kchar	4	输出	<p>GTP 接收 K 控制字，用来指示接收的数据是 K 码控制字符还是正常传输数据。高电平表明是 K 码控制字符，4 位对应接收数据 32 位的 4 个 Byte。</p> <p>rx0_kchar [3] 对应 rx0_data [31:24]</p> <p>rx0_kchar [2] 对应 rx0_data [23:16]</p> <p>rx0_kchar [1] 对应 rx0_data [15:8]</p> <p>rx0_kchar [0] 对应 rx0_data [7:0]</p>
gt0_txfsmresetdone	1	输出	GTP 初始化完成信号。

知道了 GTP 用户接口信号的含义，我们就能通过用户接口实现光纤数据的收发了。

2) gtp 数据包发送模块 packet_send.v

在 packet_send.v 中会由一个状态机来发送测试数据，首先一包数据开始传输前，会先发送同步包头信号，再发送数据包的序号和控制信号。然后把这测试数据通过 GTP 发送出去。发送流程如下图：



所有的 GTP 发送的数据位数为 32 位，同步信号包头信号定义为 32 位的“ff_00_00_bc”，低 8 位"bc"是 K28.5 码控制字符。K 码特征字定义在 Xilinx 的“ug482_7Series_GTP_Transceivers.pdf”文档里有描述。

Table C-2: Valid Control K Characters

Special Code Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
K28.0	000 11100	001111 0100	110000 1011
K28.1	001 11100	001111 1001	110000 0110
K28.2	010 11100	001111 0101	110000 1010
K28.3	011 11100	001111 0011	110000 1100
K28.4	100 11100	001111 0010	110000 1101
K28.5	101 11100	001111 1010	110000 0101
K28.6	110 11100	001111 0110	110000 1001
K28.7 ⁽¹⁾	111 11100	001111 1000	110000 0111
K23.7	111 10111	111010 1000	000101 0111
K27.7	111 11011	110110 1000	001001 0111
K29.7	111 11101	101110 1000	010001 0111
K30.7	111 11110	011110 1000	100001 0111

向 GTP 发送 K28.5 码控制字符时，需要拉高 gt_tx_ctrl 信号的对应位，标示发送数据里的某个字节位为 K 码控制字。所以这里在向 GTP 发送同步信号时，gt_tx_ctrl 信号设置为 0001，发送其它数据的时候则置为 0000。

```
47 end
48 SEND_HEADER:
49 begin
50   gt_tx_data <= 32'hff_00_00_bc;
51   gt_tx_ctrl <= 4'b0001;
52   state <= SEND_SEQ_NUM;
53   check_sum <= 32'd0;
54 end
55 SEND_SEQ_NUM:
56 begin
57   gt_tx_data <= sequence_number;
58   gt_tx_ctrl <= 4'b0000;
59   state <= SEND_CTRL;
```

3) 位数据对齐模块 word_align.v

GTP 收发器外部用户数据接口的宽度为 32 位，内部数据宽度为 20 位(8b/10b 转换)。在实际测试过程中发现，发送的 32 位数据会有可能出现 16 位的数据的移位，就是说发送的数据和接收到的数据会有 16 位的错位，下表演示 GTP 发送数据和接收数据移位的情况：

GTP 发送的数据		GTP 接收的数据	
数据 1	11111111	数据 1	11112222
数据 2	22222222	数据 2	22223333

数据 3	33333333	数据 3	33334444
数据 4	44444444	数据 4	44445555
数据 5	55555555	数据 5	5555.....
.....

因为我们在 GTP 发送同步信号和无用数据的时候加入了 K 码控制字，并且设置 `gt_tx_ctrl` 信号为 0001, 如果出现 16 位数据移位的情况，接收到的同步信号和无用数据时，K 码控制字也会跟着移位，`gt_tx_ctrl` 的信号就会变成 0100。所以我们在程序可以通过判断 `gt_tx_ctrl` 信号的值来判断接收到的 GTP 数据是否移位，如果接收到的 `gt_tx_ctrl` 为 0001，跟我们发送的时候一样，说明数据没有移位；如果接收到的 `gt_tx_ctrl` 为 0100，接收到的数据移位，需要重新组合，在 `word_align.v` 模块里完成。

```

24 always@(posedge rx_clk)
25 begin
26     case (align_bit)
27     4'b0001:
28         rx_data_align <= gt_rx_data;
29     4'b0100:
30         rx_data_align <= {gt_rx_data[15:0], gt_rx_data_d0[31:16]};
31     default:
32         rx_data_align <= 32'd0;
33     endcase
34 end

```

4) GTP 数据解析模块 `packet_rec.v`

因为接收到的 32 位数据中只有一部分是有效的数据，其它的是同步包头，序列数据，控制数据和 Checksum，在 `packet_rec.v` 模块里会计算数据的 checksum，然后跟接收到的 checksum 值进行对比，如果不正确，会产生数据错误信号。

程序的一个功能是检测 GTP 数据中的同步包头信号（数据为 `ff_00_02_bc`），如果接收到同步包头信号，开始一包数据的接收。

```

WAIT_HEADER:
begin
    check_sum <= 32'd0;
    if(gt_rx_ctrl[0] == 1'b1 && gt_rx_data[7:0] == 8'hbc)
        state <= SEQ_NUM;
end
SEQ_NUM:

```

程序的另一个功能是判断统计数据的 checksum 并和接收到的 checksum 判断。

```

3   else if(state == CHECK)
4   begin
5       packet_cnt <= packet_cnt + 1;
6       if(check_sum != gt_rx_data || sequence_number != (last_sequence_number + 1))
7           error_packet_cnt <= error_packet_cnt + 1;
8   end
9 end

```

5) 管脚约束

这里的管脚约束是在 gtp IP 的 example 工程中的 gtp_exdes.xdc 文件中修改而来，比如 GTP 的参考时钟输入管脚，这里需要跟开发板上的管脚对应。另外还需要添加 SFP 光模块的发送控制管脚的定义如

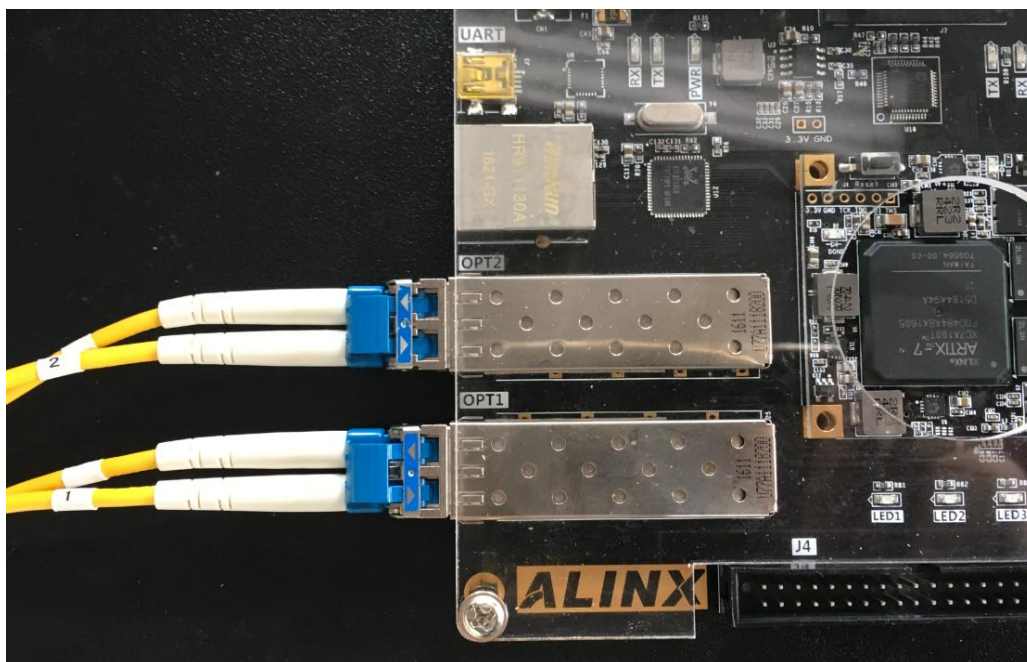
```

93 ##### ReClk Location constraints #####
94 set_property LOC E6 [get_ports Q0_CLK0_GTREFCLK_PAD_N_IN ]
95 set_property LOC F8 [get_ports Q0_CLK0_GTREFCLK_PAD_P_IN ]
96
97 ## LOC constrain for SYS_CLK_P/N
98 set_property IOSTANDARD DIFF_SSTL15 [get_ports SYS_CLK_IN_P]
99 set_property IOSTANDARD DIFF_SSTL15 [get_ports SYS_CLK_IN_N]
100 set_property PACKAGE_PIN R4 [get_ports SYS_CLK_IN_P]
101 set_property PACKAGE_PIN T4 [get_ports SYS_CLK_IN_N]
102
103 set_property IOSTANDARD LVCMOS33 [get_ports {tx_disable[1]}]
104 set_property IOSTANDARD LVCMOS33 [get_ports {tx_disable[0]}]
105 set_property PACKAGE_PIN C22 [get_ports {tx_disable[0]}]
106 set_property PACKAGE_PIN C20 [get_ports {tx_disable[1]}]

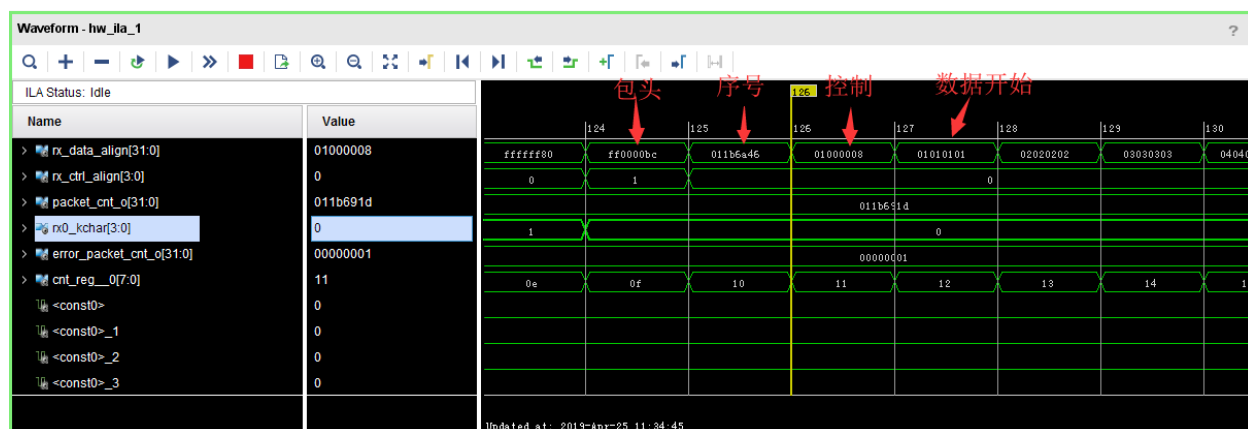
```

4 光纤数据传输测试

编译项目通过后我们就可以开始光纤数据传输的实验了。光模块插入到 AX7102 开发板的 OPT1~OPT2 接口上，再连接光纤。AX7101 的用户，如果只有两个光模块，那只要插入到 OPT1 和 OPT2 的接口上（RX 和 TX 相连接）。AX7102 硬件连接后如下图所示：

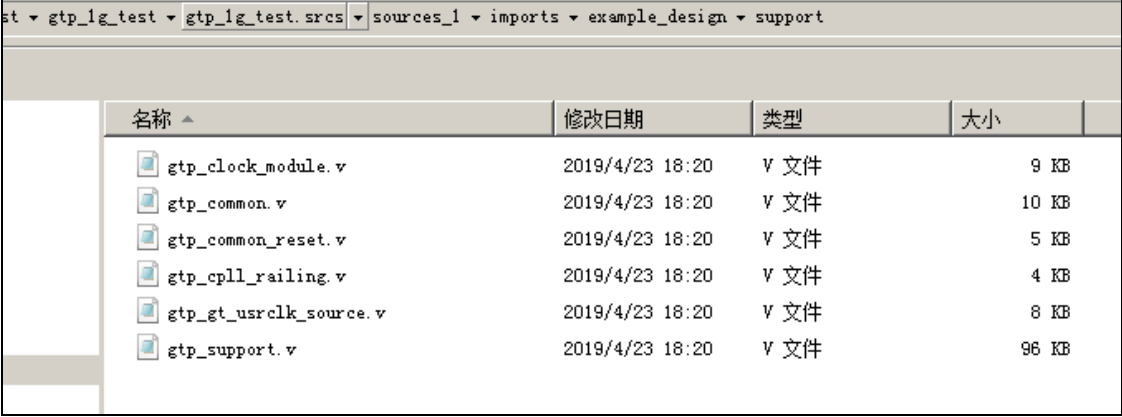








再下载 bit 文件到 FPGA, 我们可以在 ILA 里看到接收的测试数据了。



如果有接收的数据包错误, error_packet_cnt_o 的值会加 1。

这里为止 GTP 光纤数据传输例程就介绍完了。如果用户需要对工程中的 gtp IP 重新配置的话，如果只是修改工程里的 gtp IP 配置的话，编译的时候会报错。用户需要在修改工程中的 gtp IP 配置的同时，还是需要重新生成 gtp IP 的 example 文件，然后用 example 工程中生成的文件去替换工程中的以下文件。



名称 ^	修改日期	类型	大小
 gtp_clock_module.v	2019/4/23 18:20	V 文件	9 KB
 gtp_common.v	2019/4/23 18:20	V 文件	10 KB
 gtp_common_reset.v	2019/4/23 18:20	V 文件	5 KB
 gtp_cpll_railing.v	2019/4/23 18:20	V 文件	4 KB
 gtp_gt_usrclk_source.v	2019/4/23 18:20	V 文件	8 KB
 gtp_support.v	2019/4/23 18:20	V 文件	96 KB