

GTH 光纤通信测试例程

黑金动力社区 2021-10-27

1 实验简介

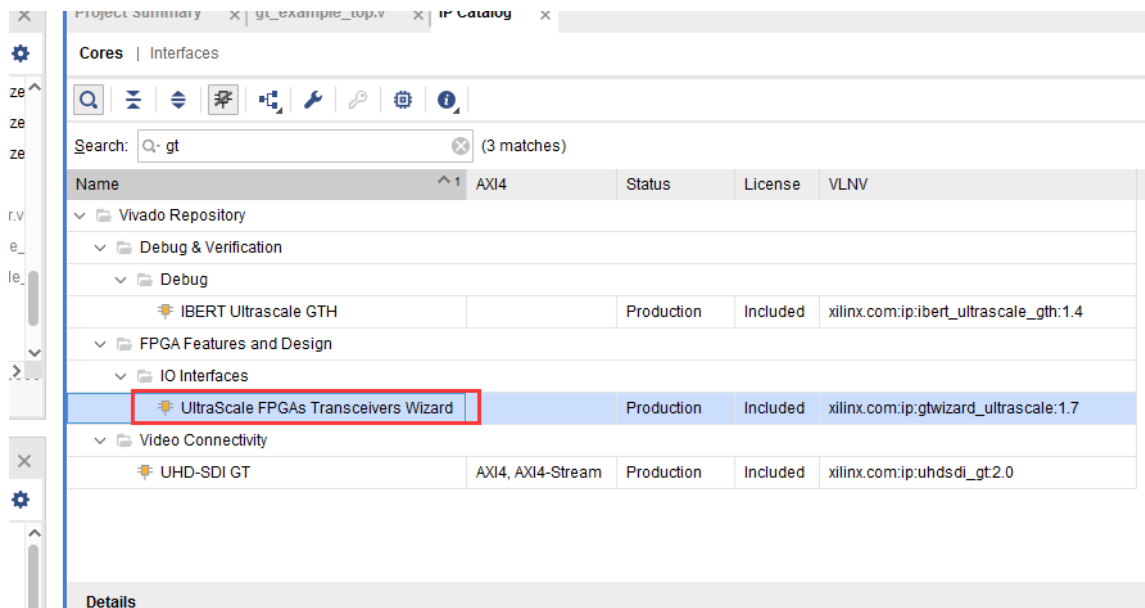
本实验将介绍通过光纤实现数据通信的传输，测试数据由 FPGA 自身产生，由 GTH 发送到第一路光纤口，然后通过光纤线环路到第二路光纤口，通过 GTH/GTY 接收数据进行校验。

2 实验原理

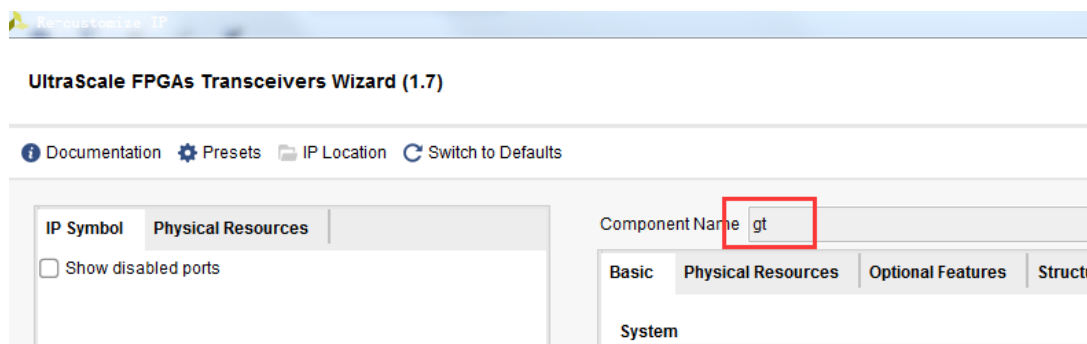
2.1 GTH IP 设计

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

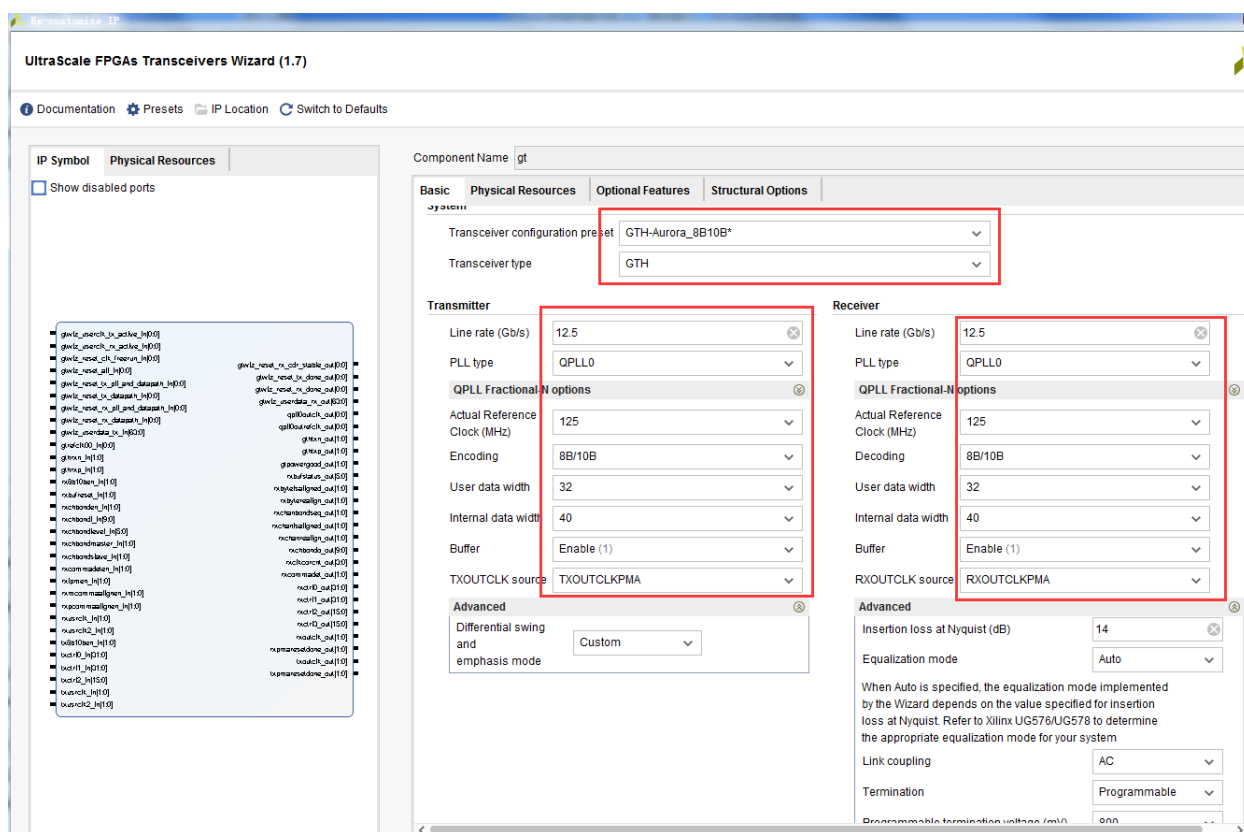
1. 在 IP Catalog 界面中双击 FPGA Features and Design\IO Interface 目录下的"UltraScale FPGAs Transceivers Wizard"图标。



2. 在 Basic 界面里，Component Name 栏输入"gt"为取名。



3. 在 Basic 界面里，首先设置传输协议位 “GTH-Aurora 8B10B”，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 速度和使用的 PLL 类型，这里选择 12.5G 和 QPLL0。参考时钟根据板上实际的晶振，选择 125Mhz，用户数据宽度为 32 位，内部通过 8B/10B 转换后就是 40 位。



4. 在 Physical Resources 页面里，设置 DRP 的时钟为 50Mhz。以 AXU4EV-P 开发板为例，XCZU4EV-SFVC784 芯片只有 1 个 BANK 的 GTH 收发器（BANK224），在 AXU4EV-P 硬件设计上 BANK224 的 X0Y4 和 X0Y5 连接了 2 路 SFP+ 的光模块，参考时钟 125Mhz 连接到 MGTREFCLK1。所以这里需要选中 GTHE_CHANNEL_X0Y4, GTHE_CHANNEL_X0Y5, 参考时钟选择 MGTREFCLK1。

UltraScale FPGAs Transceivers Wizard (1.7)

Documentation Presets IP Location Switch to Defaults

IP Symbol Physical Resources

Right-click on channels to enable and edit

Component Name: gt

Basic Physical Resources Optional Features Structural Options

Free-running and DRP clock frequency (MHz): 50 (3.125 - 250)

TX Master channel: X0Y4 RX Master channel: X0Y4 Disable All Channels

Channel table (2/4 channels used)

Search: Q-

Channel	Enable	TX REFCLK source...	RX REFCLK sour...	RXREC...	Location details: Bank, Data pins (RXN, RXP, TXN, TXP)
Right column (2/4 channels used)					
Quad X0Y1 in SLR 0 (2/4 channels used)					
GTHE4_CHANNEL_X0Y7	<input type="checkbox"/>				Bank: 224; Data pins :P1, P2, N3, N4
GTHE4_CHANNEL_X0Y6	<input type="checkbox"/>				Bank: 224; Data pins :T1, T2, R3, R4
GTHE4_CHANNEL_X0Y5	<input checked="" type="checkbox"/>	MGTREFCLK1	MGTREFCLK1	No	Bank: 224; Data pins :V1, V2, U3, U4
GTHE4_CHANNEL_X0Y4	<input checked="" type="checkbox"/>	MGTREFCLK1	MGTREFCLK1	No	Bank: 224; Data pins :Y1, Y2, W3, W4

OK Cancel

5. 在 **Optional Features** 界面里，设置接收时钟校验，这里设置 4 个的 F7 的 K 码作为接收时钟校验。所以后面光纤数据发送的时候需要把这 4 个字节的时钟校验码也加上。

Documentation Presets IP Location Switch to Defaults

IP Symbol Physical Resources

Right-click on channels to enable and edit

Component Name: gt

Basic Physical Resources Optional Features Structural Options

Receiver comma detection and alignment

Receiver channel bonding

Receiver clock correction

Enable and select number of sequences to use: 1

Length of each sequence: 4

	Don't care	Value	K character	Inverted disparity
Sequence 0, pattern 0	<input type="checkbox"/>	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sequence 0, pattern 1	<input type="checkbox"/>	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sequence 0, pattern 2	<input type="checkbox"/>	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sequence 0, pattern 3	<input type="checkbox"/>	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sequence 1, pattern 0	<input type="checkbox"/>	00000000	<input type="checkbox"/>	<input type="checkbox"/>
Sequence 1, pattern 1	<input type="checkbox"/>	00000000	<input type="checkbox"/>	<input type="checkbox"/>
Sequence 1, pattern 2	<input type="checkbox"/>	00000000	<input type="checkbox"/>	<input type="checkbox"/>
Sequence 1, pattern 3	<input type="checkbox"/>	00000000	<input type="checkbox"/>	<input type="checkbox"/>

Periodicity of the sequence (in bytes): 5000

Keep idle: Disable

Precedence: Enable

Minimum repetition: 0

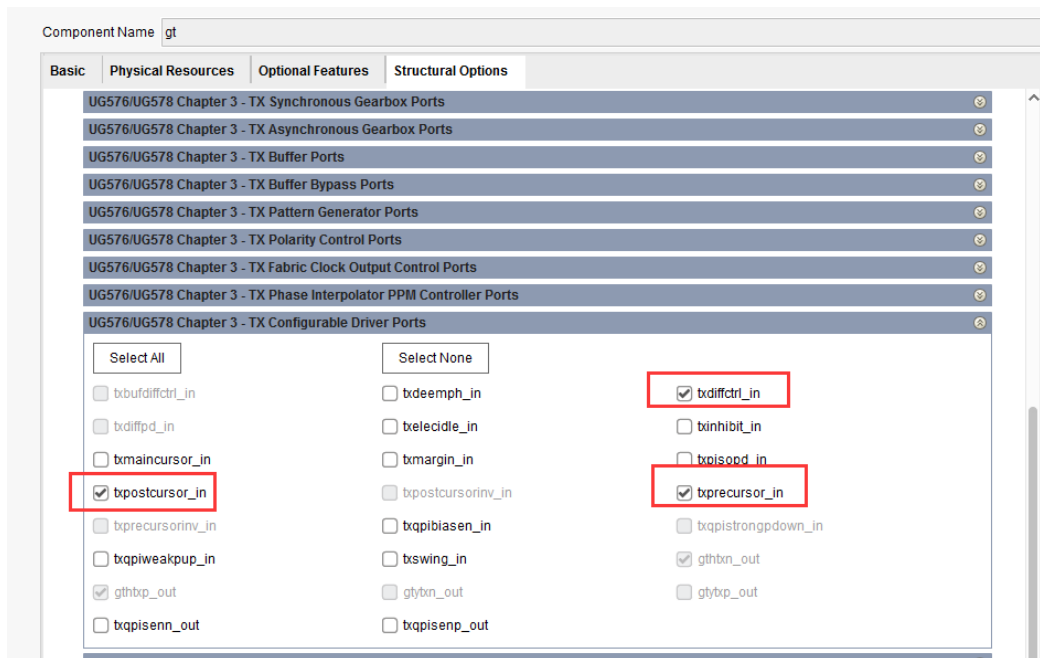
Buffer control

Advanced clocking

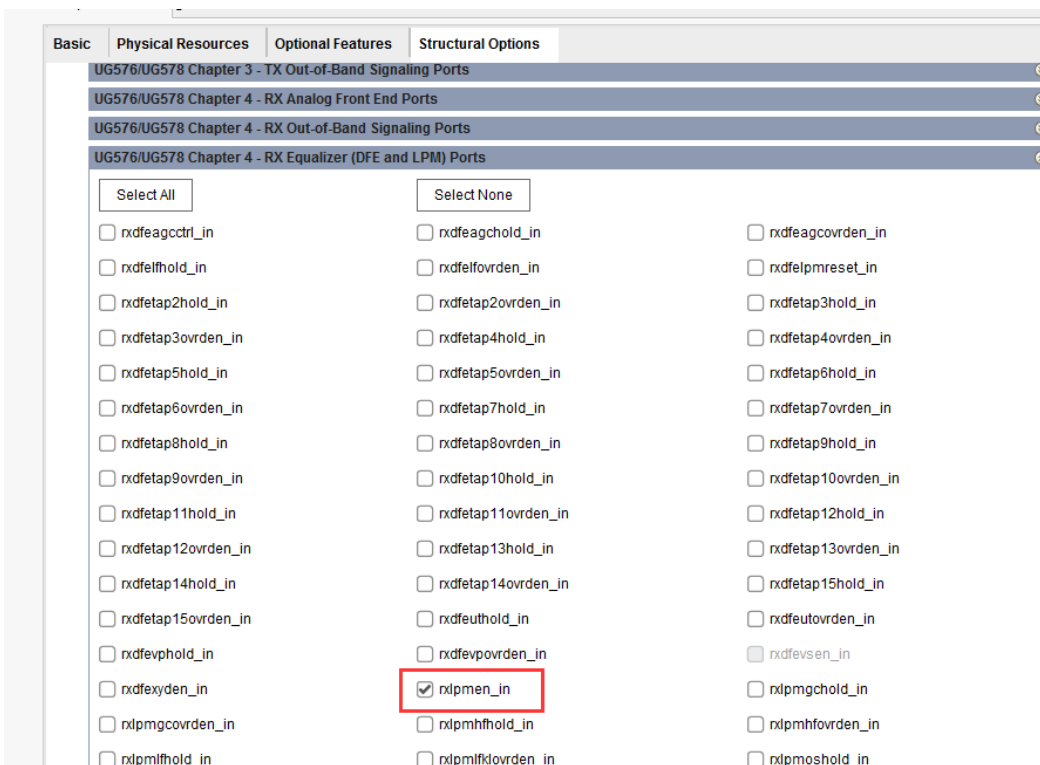
SATA

OK Cancel

6. Structural Options 界面里，选择 TX Configurable Driver Ports 项，选中以下几项，IP 会多出一些信号来配置发送端的驱动能力。




选择 RX Equalizer(DFE and LPM) Ports 项，选中 rxlpmen_in 项, 这个管脚可以控制 DFE 模式还是 LPM 模式（默认为 DFE 模式）

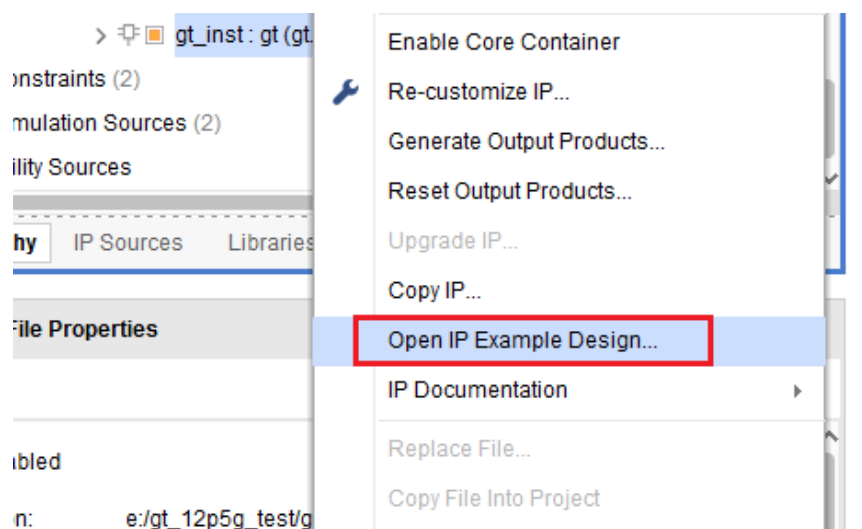


关于更多 GTH IP 的配置信息，大家请参考 Xilinx 提供的文档“ug576-ultrascale-gth-transceivers”和“PG182-UltraScale FPGA Transceivers Wizard”，这个文档上有对 GTH IP 各个配置参数做了详细的介绍。

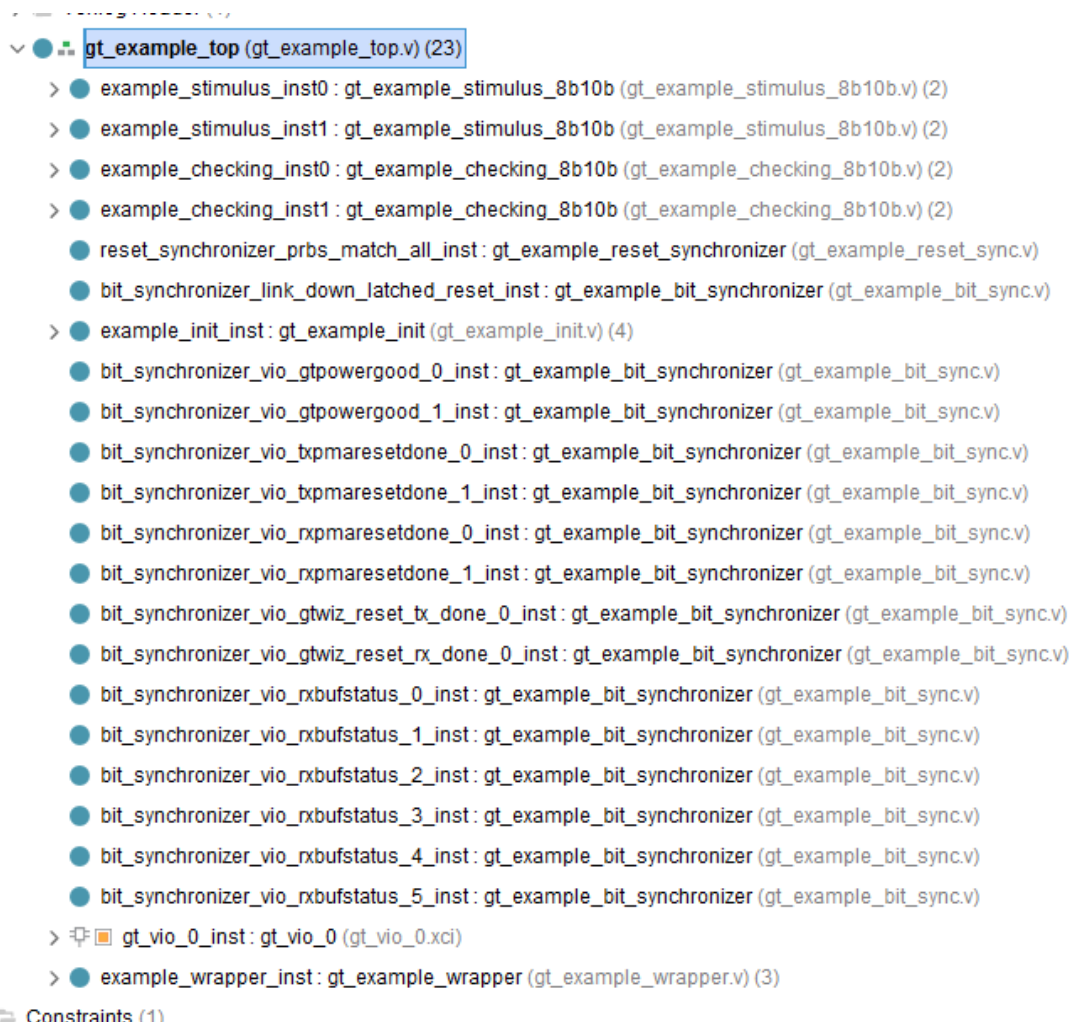
8. 配置完成在工程中自动添加刚刚生成 gt 的 IP。

>  gt_inst: gt (gt.xci)

9. 我们来生成 GT IP 的 example 工程，右键选择 gt,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，gt_example_stimulus_8b10b 程序是产生测试的 PRBS31 的数据模块，PRBS31 的数据供 GTH 发送使用。gt_example_checking_8b10b 程序是检查接收到的 PRBS31 的数据是否正确。关于 example 的工程代码和测试我们这里不做介绍，里面一些模块的功能和信号的功能大家自己去看 PG182 文档就可以了。在后面的 GTH 数据传输中我们会在这个例子工程的基础上进行修改。

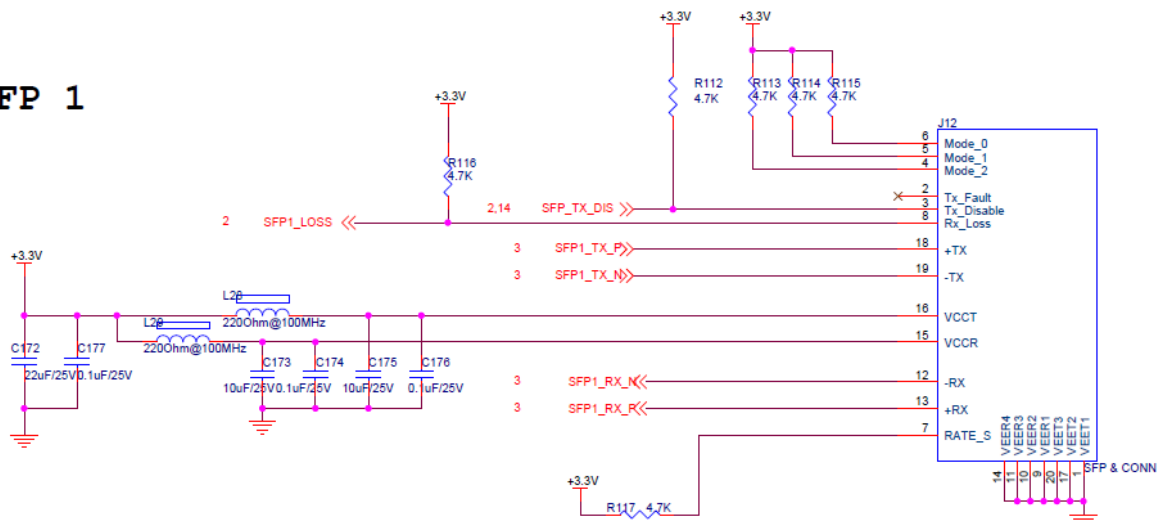
2.2 硬件介绍

在 AXU4EV-P 开发板上，有 2 路光纤接口 SFP1~SFP2，分别连接到 FPGA 芯片的 GTH 的通道上。

其中 SFP1 光模块接口连接到 GTH 的 Channel0 上，SFP2 跟 GTH 的 Channel1 相连。

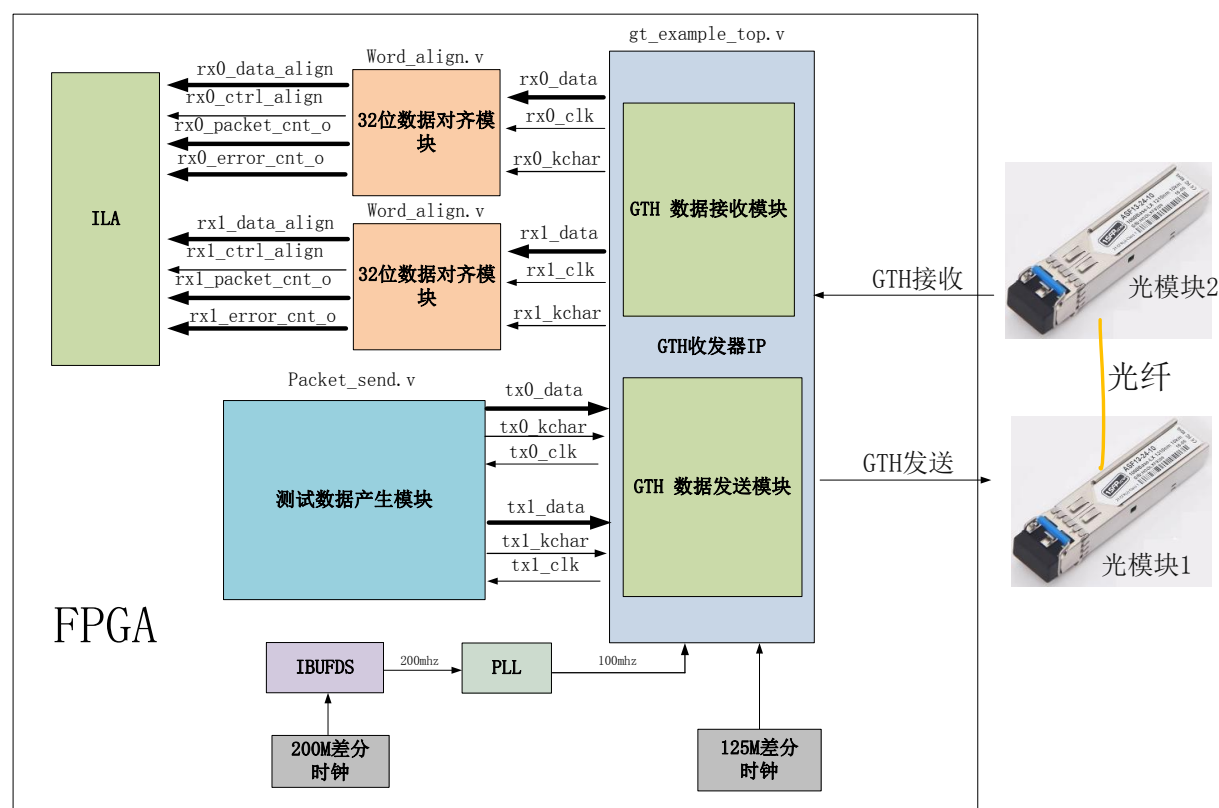
光模块的 TX_Disable 信号和 RX_LOSS 连接到 FPGA 的普通 IO 上。TX_Disable 信号用来使能或者不使能光模块的光发射，如果 TX_Disable 信号为高，光发射关闭，否则光发送使能，正常使用的時候需要拉低此信号。SFP1 光模块硬件原理图如下：

SFP 1



3 程序设计

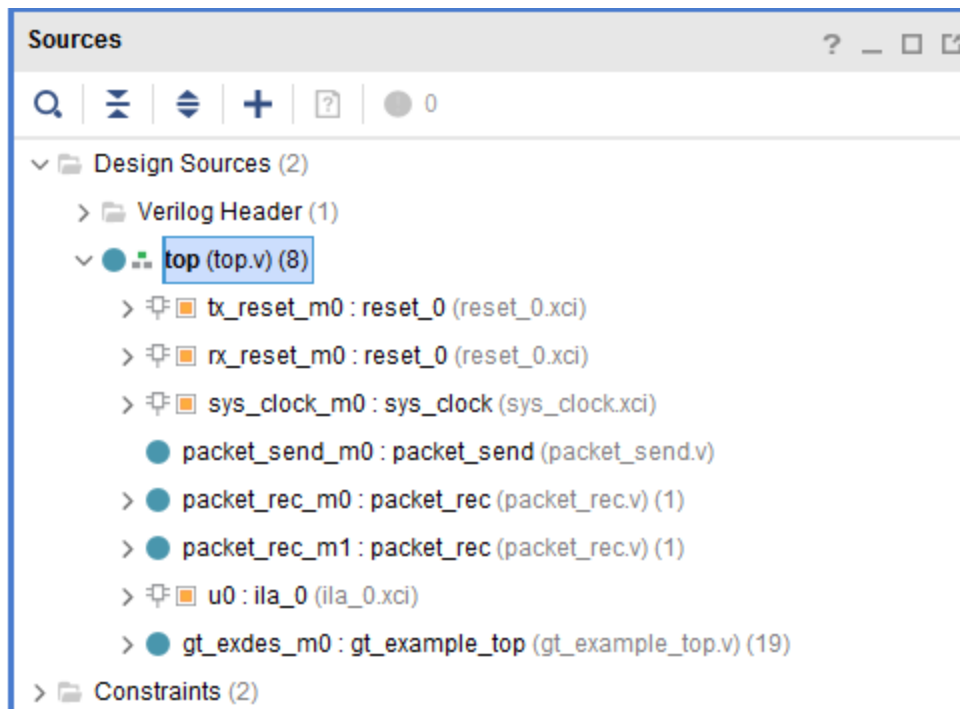
光纤数据传输的 FPGA 程序设计是在 example 的工程代码的基础上添加了一个 TOP 文件和 3 个.v 文件，工程的逻辑框图如下图所示：



程序产生数据使用 GTH 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 GTH 接收，GTH 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) gth 数据通信模块

在前面我们已经生成过 gth IP 的 example 工程，这里删除了 gt_example_stimulus_8b10b.v 模块和 gt_example_checking_8b10b.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，再就是接口信号跟原来数据产生模块和检查模块的信号连接。具体修改的地方我们都加了//Modify，大家可以跟源文件对比一下。

➤ 比如下面是在模块的 Port 处添加以下的用户接口信号：


```
60 // Differential reference clock inputs
61 input wire mgtrefclk_p,
62 input wire mgtrefclk_n,
63
64 //modify
65
66 input wire [1:0] RXN_IN,
67 input wire [1:0] RXP_IN,
68 output wire [1:0] TXN_OUT,
69 output wire [1:0] TXP_OUT,
70
71 output tx0_clk,
72 output gt0_txfsmresetdone,
73 output gt0_rxfsmresetdone,
74 input [31:0] tx0_data,
75 input [3:0] tx0_kchar,
76 output rx0_clk,
77 output [31:0] rx0_data,
78 output [3:0] rx0_kchar,
79
80
81 output tx1_clk,
82 output gt1_txfsmresetdone,
83 output gt1_rxfsmresetdone,
84 input [31:0] tx1_data,
```

下面对用户接口信号做一下介绍，以下以 channel0 的 GTH 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 GTH 的参考时钟 312.5Mhz。数据在上升沿有效。
tx0_data	32	输入	GTH 发送数据。
tx0_kchar	4	输入	GTH 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。 tx0_kchar [3] 对应 tx0_data [31:24] tx0_kchar [2] 对应 tx0_data [23:16] tx0_kchar [1] 对应 tx0_data [15:8] tx0_kchar [0] 对应 tx0_data [7:0]
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 GTH 的参考时钟 312.5Mhz。数据在上升沿有效。
rx0_data	32	输出	GTH 接收数据。
rx0_kchar	4	输出	GTH 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

			<p>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_tx fsmresetdone	1	输出	GTH 初始化完成信号。

知道了 GTH 用户接口信号的含义，后面我们需要通过这些用户接口实现光纤数据的收发。

➤ 通过下面的连接，完成用户接口信号和 GTH IP 的连接。

```

582 //modify
583 assign tx0_clk      = hb0_gtwiz_userclk_tx_usrclk2_int;
584 assign rx0_clk      = hb0_gtwiz_userclk_rx_usrclk2_int;
585 assign rx0_data     = hb0_gtwiz_userdata_rx_int;
586 assign rx0_kchar    = ch0_rxctrl2_int;
587 assign gt0_tx fsmresetdone = gt wiz_reset_tx_done_int;
588 assign gt0_rx fsmresetdone = gt wiz_reset_rx_done_int;
589
590
591 assign tx1_clk      = hb0_gtwiz_userclk_tx_usrclk2_int;
592 assign rx1_clk      = hb0_gtwiz_userclk_rx_usrclk2_int;
593 assign rx1_data     = hb1_gtwiz_userdata_rx_int;
594 assign rx1_kchar    = ch1_rxctrl2_int;
595 assign gt1_tx fsmresetdone = gt wiz_reset_tx_done_int;
596 assign gt1_rx fsmresetdone = gt wiz_reset_rx_done_int;
597
598 assign hb0_gtwiz_userdata_tx_int = tx0_data ;
599 assign hb1_gtwiz_userdata_tx_int = tx1_data ;
600
601
602 assign ch0_txctrl2_int = tx0_kchar ;
603 assign ch1_txctrl2_int = tx1_kchar ;
604
605 assign ch0_txctrl10_int = 16'd0 ;
606 assign ch1_txctrl10_int = 16'd0 ;
607
608 assign ch0_txctrl11_int = 16'd0 ;
609 assign ch1_txctrl11_int = 16'd0 ;

```

关于程序中用到的信号，用户可以参考“ug576-ultrascale-gth-transceivers.pdf”文档查询。比如 P117 页介绍 8 位 txctrl2 信号的功能就是对应 64 位数据的 K 码。

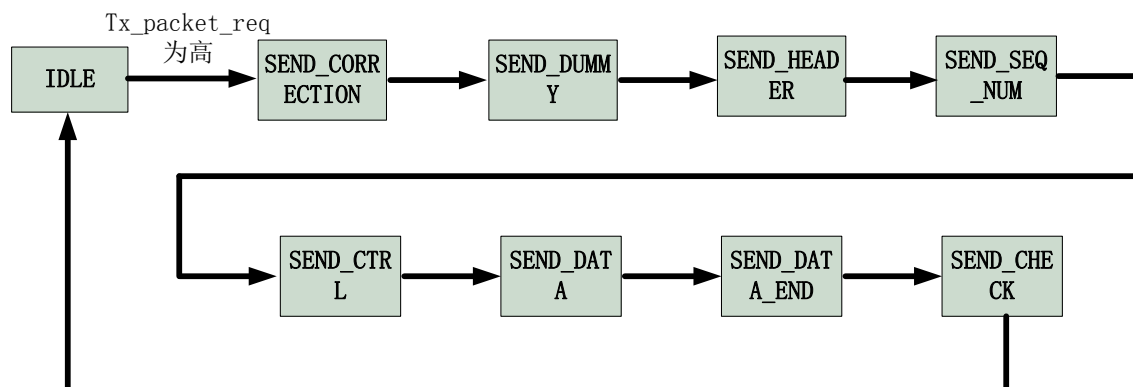
TXCTRL2[7:0]	In	TXUSRCLK2	<p>When High, indicates the corresponding data byte on TXDATA is a valid K character.</p> <p>TXCTRL2[7] corresponds to TXDATA[63:56] TXCTRL2[6] corresponds to TXDATA[55:48] TXCTRL2[5] corresponds to TXDATA[47:40] TXCTRL2[4] corresponds to TXDATA[39:32] TXCTRL2[3] corresponds to TXDATA[31:24] TXCTRL2[2] corresponds to TXDATA[23:16] TXCTRL2[1] corresponds to TXDATA[15:8] TXCTRL2[0] corresponds to TXDATA[7:0]</p> <p>A TXCTRL2 bit should be driven Low when the corresponding data byte from TXDATA is set to bypass the 8B/10B encoder.</p>
--------------	----	-----------	--

程序里定义了 TX 发送的驱动能力的参数配置，具体含义参考“ug576-ultrascale-gth-transceivers.pdf”文档

```
localparam TXDIFFCTRL = 5'b11000 //modify
localparam TXPRECURSOR = 5'b01110 //modify
localparam TXPOSTCURSOR = 5'b00111 //modify
```

2) gth 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 GTH 发送出去。发送流程如下图：



所有的 GTH 发送的数据位数为 32 位，在发送数据包之前需要发送 32 位的数据校正控制字，校正控制字用于消除不同的系统时钟之间造成的频率误差。gt_tx_ctrl 信号设置为 1111，表明这 32 位数据都是控制字。

```

    }
    SEND_CORRECTION:
    begin
        gt_tx_data <= 32'hF7_F7_F7;
        gt_tx_ctrl <= 4'b1111;
        state <= SEND_DUMMY;
    end
end
```

这个 32 位的校正字（F7F7F7F7）在 GTH 的 IP 里已经配置好了。所以 GTH 接收数据包的时候会自动做校正

	Don't care	Value	K character	Inverted disparity
Sequence 0, pattern 0	<input type="checkbox"/>	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sequence 0, pattern 1	<input type="checkbox"/>	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sequence 0, pattern 2	<input type="checkbox"/>	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sequence 0, pattern 3	<input type="checkbox"/>	11110111	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sequence 1, pattern 0	<input type="checkbox"/>	00000000	<input type="checkbox"/>	<input type="checkbox"/>
Sequence 1, pattern 1	<input type="checkbox"/>	00000000	<input type="checkbox"/>	<input type="checkbox"/>
Sequence 1, pattern 2	<input type="checkbox"/>	00000000	<input type="checkbox"/>	<input type="checkbox"/>

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“ug576-ultrascale-gth-transceivers.pdf”文档里有描述。

Table A-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

Notes:

1. Used for testing and characterization only.

向 GTH 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 GTH 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

GTH 收发器外部用户数据接口的宽度为 32 位，内部数据宽度为 40 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 GTH 发送数据和接收数据移位的情况：

GTH 发送的数据		GTH 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 GTH 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(algn_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) GTH 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 GTH 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 GTH IP 的 example 工程中的 gt_example_top.xdc 文件中修改而来，比如 GTH 的参考时钟 125Mhz 和系统时钟 200Mhz，这里需要跟开发板上的管脚对应。

```
56 set_property PACKAGE_PIN V5 [get_ports mgtrefclk_n]
57 set_property PACKAGE_PIN V6 [get_ports mgtrefclk_p]

83
84 set_property IOSTANDARD DIFF_SSIL12 [get_ports sys_clk_p]
85 set_property PACKAGE_PIN AE5 [get_ports sys_clk_p]
86 set_property PACKAGE_PIN AF5 [get_ports sys_clk_n]
87 set_property IOSTANDARD DIFF_SSIL12 [get_ports sys_clk_n]
88 create_clock -period 5.000 -name sys_clk_p -waveform {0.000 2.500} [get_ports sys_clk_p]
```

另外还需要添加 SFP 光模块的发送控制管脚的定义如

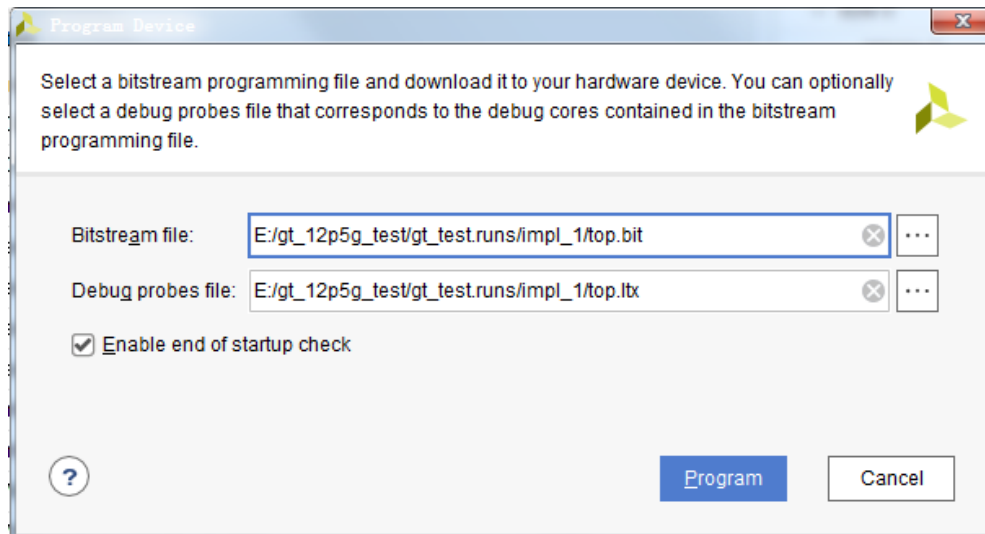
```
1 set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
2
3 set_property PACKAGE_PIN D12 [get_ports {tx_disable[0]}]
4 set_property IOSTANDARD LVCMOS33 [get_ports {tx_disable[0]}]
```

4 光纤数据传输测试

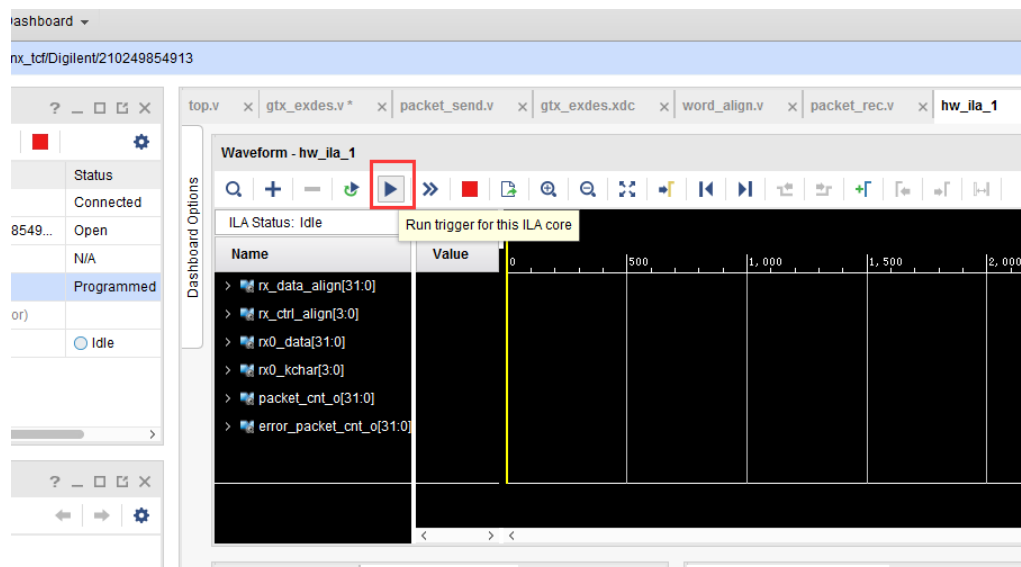
编译项目通过后我们就可以开始光纤数据传输的实验了。光模块插入到 AXU4EV-P 开发板的 SFP1~SFP2 接口上，再连接光纤。AXU4EV-P 硬件连接后如下图所示：



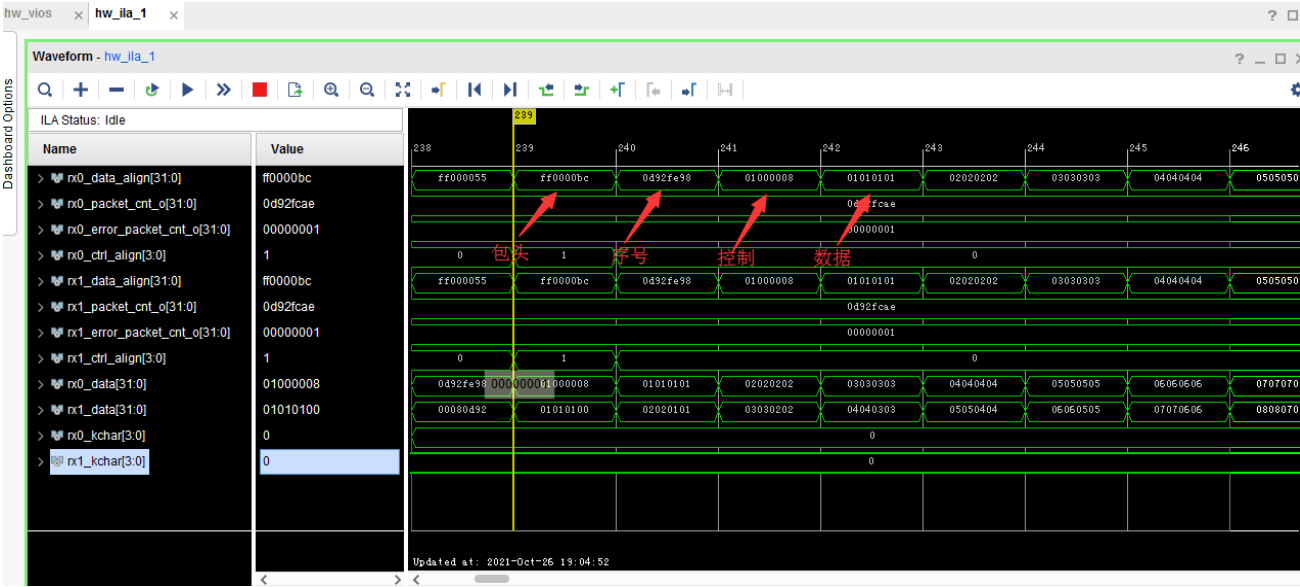
再下载 bit 文件到 FPGA,



点击 Run Trigger for this ILA core 按钮。



我们就可以在 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。