

GTH 光纤通信测试例程

黑金动力社区 2021-10-30

1 实验简介

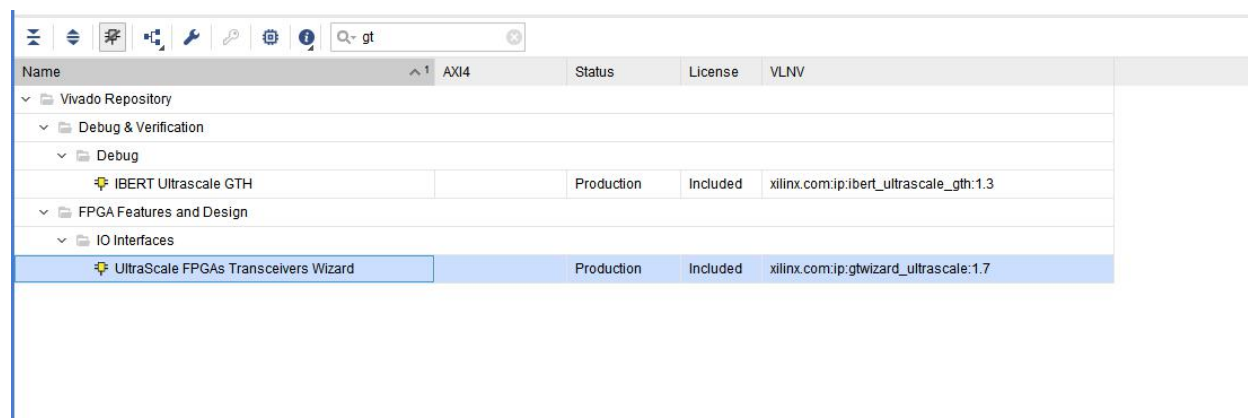
本实验将介绍通过光纤实现数据通信的传输, 测试数据由 FPGA 自身产生, 由 GTH 发送到第一路光纤口, 然后通过光纤线环路到第二路光纤口, 通过 GTH 接收数据进行校验。

2 实验原理

2.1 GTH IP 设计

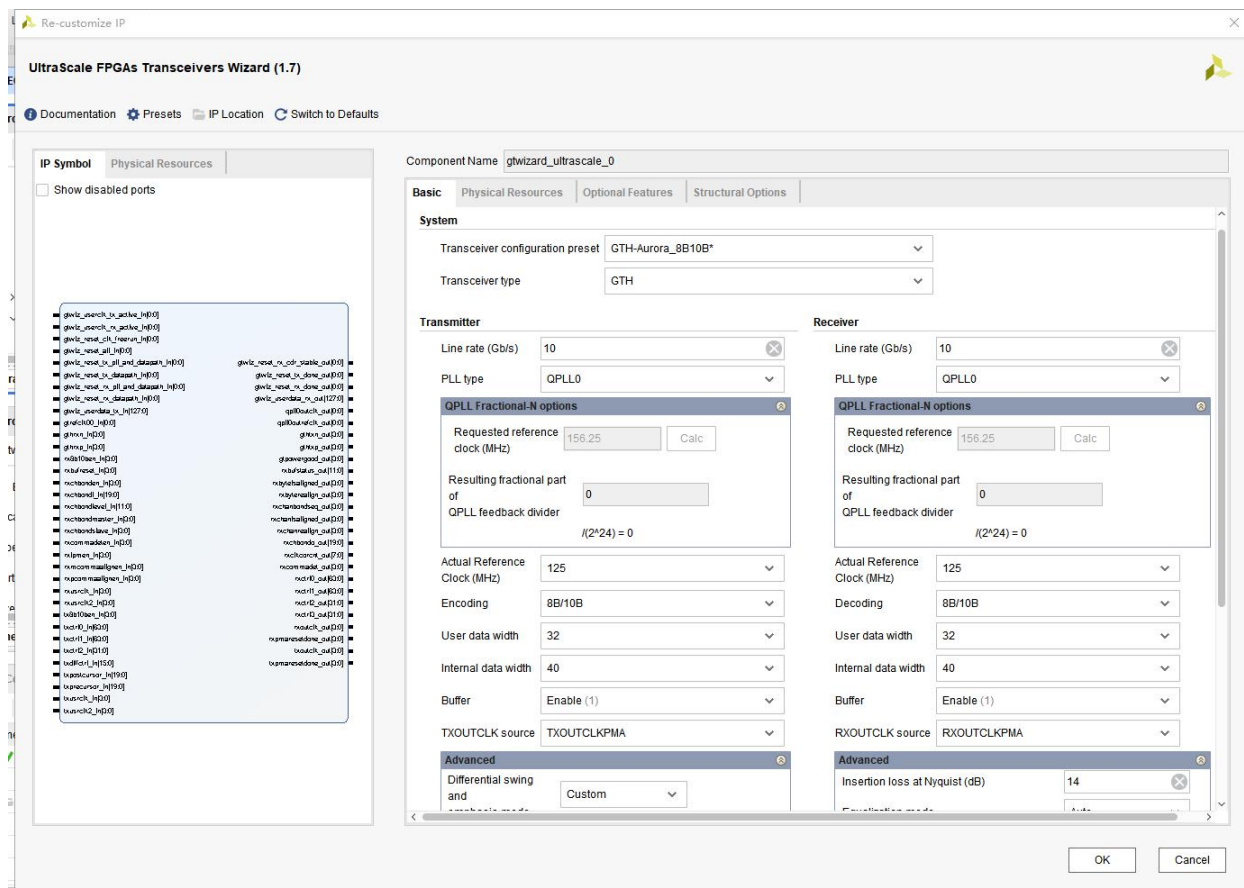
XILINX 的 Vivado 软件已经为用户设计好了 GTH IP, 用户无需关心 GTH 的内部具体工作就可以使用 IP 来实现 GTH 的高速的数据收发。下面我具体的 GTH IP 的生成和配置方法:

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 Line Rate, Refclk Selection 界面里, 首先设置 GTH 的传输协议位 "Aurora 8B10B single lane 4byte ", 我们在前面一章讲过, Xilinx 的 GTH 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 GTH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 100 倍，所以 Line Rate 设置为 1.25Gbps, 如果用户需要设置其它速率比如 1.25Gbps,只需要直接修改。Reference Clock 为 125Mhz。这里设置内部数据宽度为 20。因为外部数据宽度是 32，是内部数据宽度的 2 倍，所以这里 GTH 的内部时钟频率是外部接口的 2 倍。



因为 XCKU040FFVA1156 芯片有 4 个 BANK 的 GTX 收发器，在 AXKU040 硬件设计上 BANK224连接了 4 路 SFP+的光模块，所以这里需要选中 GTX_X0Y8, GTX_X0Y9, GTX_X0Y10, GTX_X0Y11, 然后选择右边的"Use GTX X0Y8/9/10/11"前面的钩，这样 QPLL0 都连接到了 4 个 GTX Channel 模块。TX Master channel 与 RX Master channel 这里可以选择其中一个通道，Free_running and DRP clock frequency 设置位 100M 其他默认即可 选择 ok 生成我们的 ip 核

Basic

Physical Resources

Optional Features

Structural Options

Free-running and DRP clock frequency (MHz) 100 [3.125 - 250]

TX Master channel X0Y9 RX Master channel X0Y9

Disable All Channels


Channel table (4/20 channels used)

Search: Q-

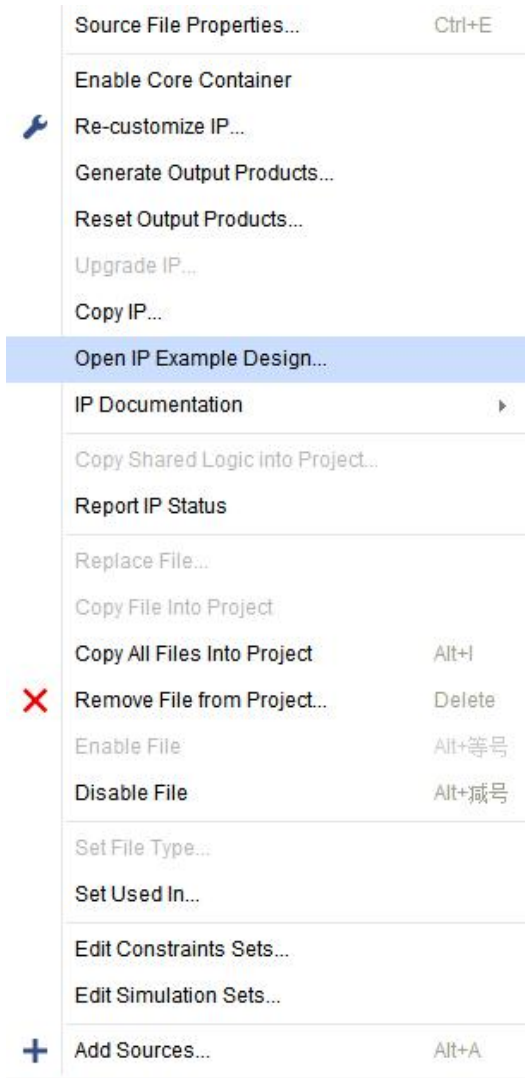
Channel	Enable	TX REFCLK source (QPLL0)	RX REFCLK source (QPLL0)	RXRECCLKOUT buffer	Location details: Bank, Data pins (RXN, RXP, TXN, TXP)
Right column (4/20 channels used)					
Quad X0Y4 in SLR 0 (0/4 channels used)					
Quad X0Y3 in SLR 0 (0/4 channels used)					
Quad X0Y2 in SLR 0 (4/4 channels used)					
GTHE3_CHANNEL_X0...	<input checked="" type="checkbox"/>	MGTREFCLK1 of Quad X0Y4 ...	MGTREFCLK1 of Quad X0Y4 ...	No	Bank: 226; Data pins :P1, P2, R3, R4
GTHE3_CHANNEL_X0...	<input checked="" type="checkbox"/>	MGTREFCLK1 of Quad X0Y4 ...	MGTREFCLK1 of Quad X0Y4 ...	No	Bank: 226; Data pins :T1, T2, U3, U4
GTHE3_CHANNEL_X0...	<input checked="" type="checkbox"/>	MGTREFCLK1 of Quad X0Y4 ...	MGTREFCLK1 of Quad X0Y4 ...	No	Bank: 226; Data pins :V1, V2, W3, W4
GTHE3_CHANNEL_X0...	<input checked="" type="checkbox"/>	MGTREFCLK1 of Quad X0Y4 ...	MGTREFCLK1 of Quad X0Y4 ...	No	Bank: 226; Data pins :Y1, Y2, AA3, AA4
Quad X0Y1 in SLR 0 (0/4 channels used)					
Quad X0Y0 in SLR 0 (0/4 channels used)					

关于更多 GTH IP 的配置信息，大家请参考 Xilinx 提供的文档“pg182”和“UG576Series FPGAs GTX_GTH Transceivers User Guide.pdf”，这两个文档上有对 GTH IP 各个配置参数做了详细的介绍。

3. 配置完成在工程中自动添加刚刚生成的 IP。

>  gtwizard_ultrascale_0 (70)

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



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GTH 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。

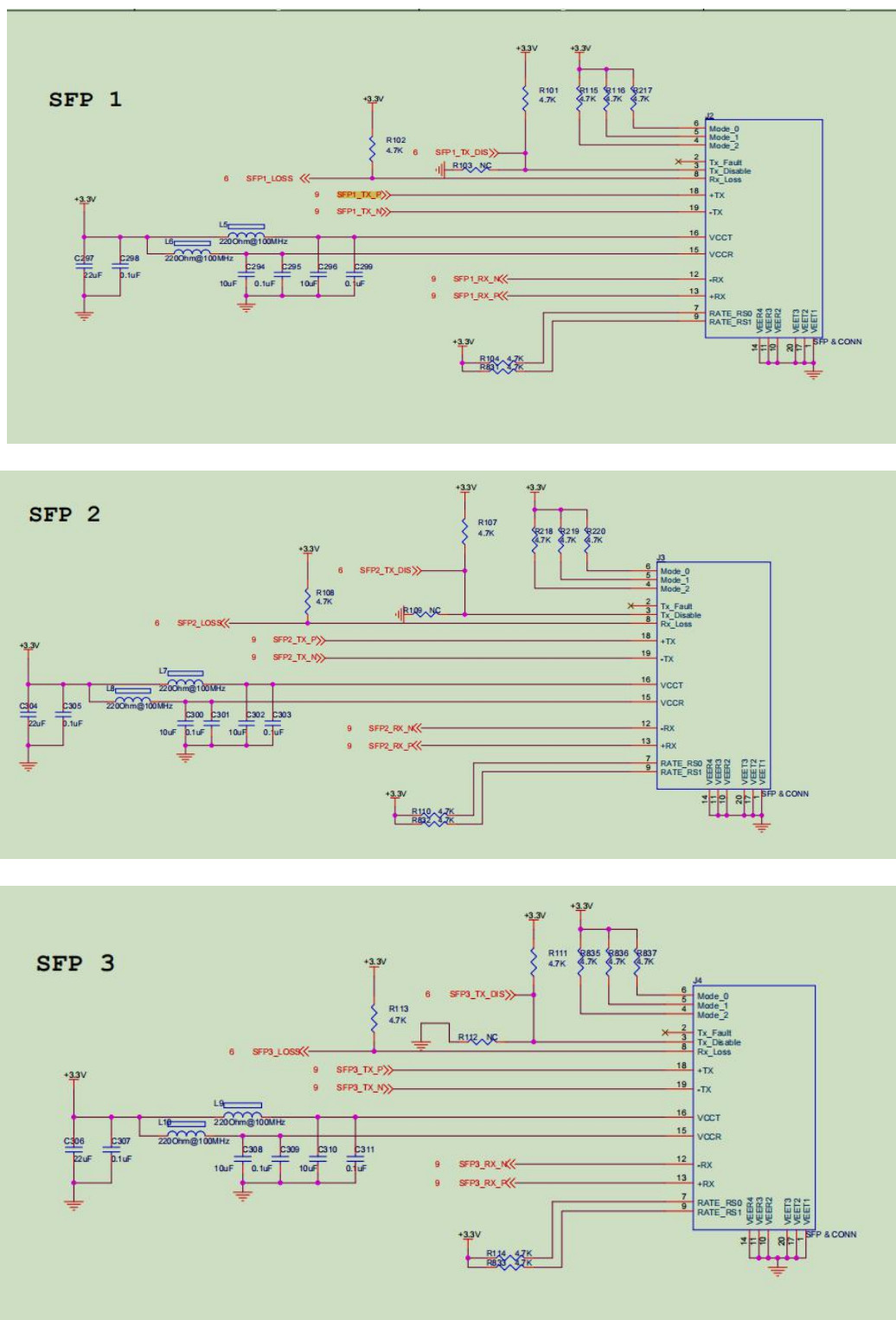
2.2 硬件介绍

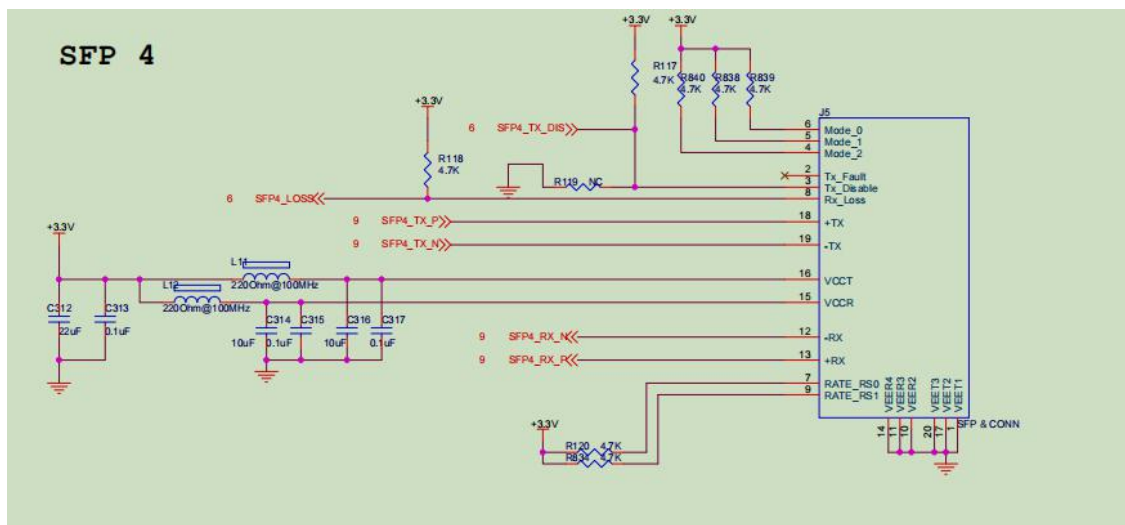
在 AXKU040 开发板上，有 4 路光纤接口 FPGA 芯片的 GTH 的通道

上。光模块和 FPGA 之间用 0.1uf 的电容器隔开，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

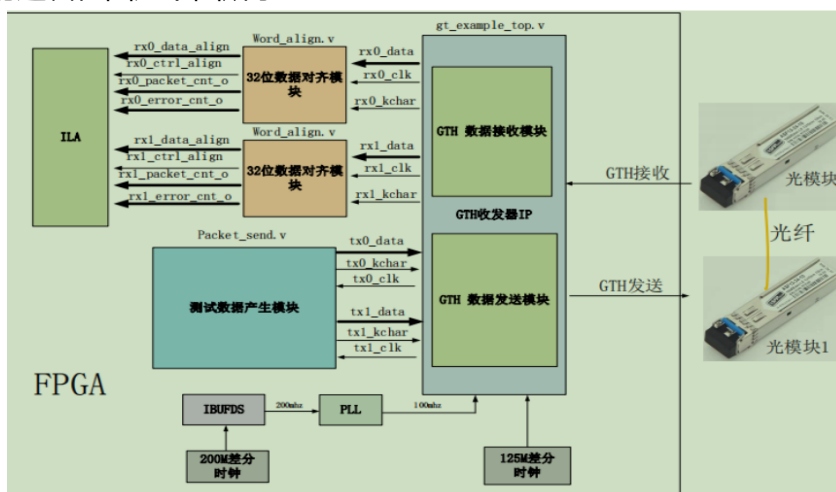
信号用来使能或者不使能光模块的光发射，如果 TX_Disable 信号为高，光发射关闭，否则光发送使能，正常使用的时候需要拉低此信号。硬件原理图如下：





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 GTH IP 发送给外部的的光模块 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。

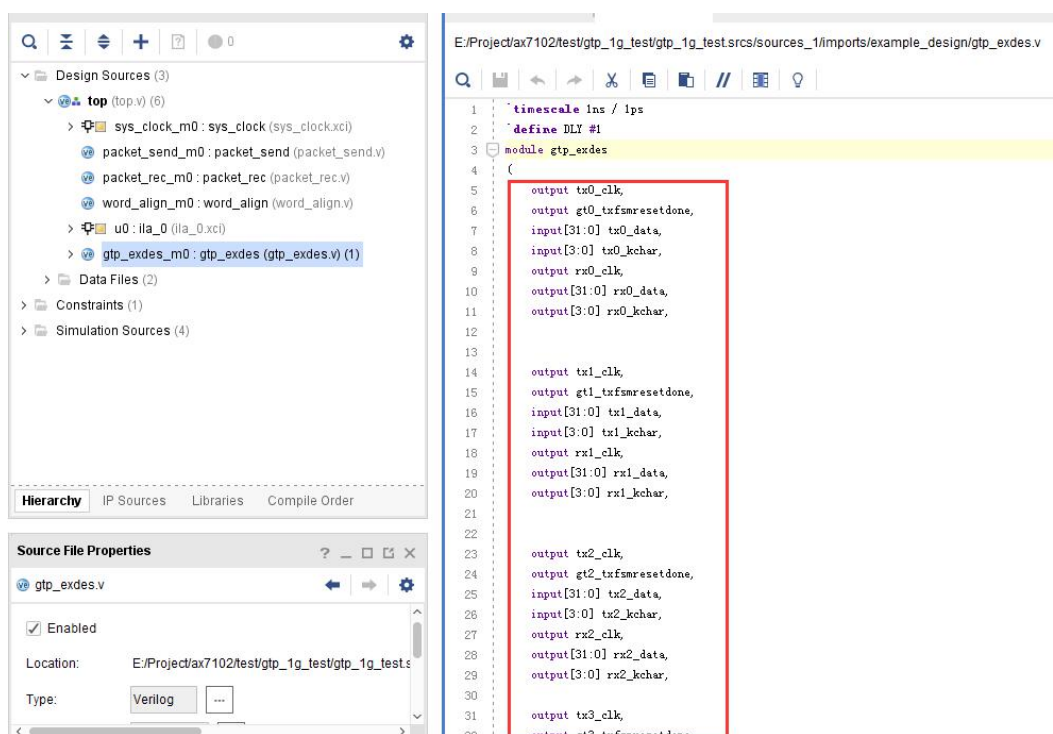
光纤数据传输设计好的工程如下图所示：



这里工程添加了 ILA 工具可以用来查看接收到的数据。

1) gtx 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 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：



其中标注有“ modify ”都是需要修改的地方

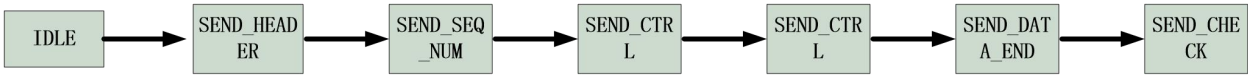
下面对在 gtx_exdes.v 端口中添加的几个 GTH 用户接口信号做一下介绍，以下以 channel0 的GTH接口为例：

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 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	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。 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 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	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。 rx0_kchar [3] 对应 rx0_data [31:24] rx0_kchar [2] 对应 rx0_data [23:16] rx0_kchar [1] 对应 rx0_data [15:8] rx0_kchar [0] 对应 rx0_data [7:0]
gt0_txfsmresetdone	1	输出	GTH 初始化完成信号。

知道了 GTH 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发了。

2) gtx 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，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 “7 Series FPGAs GTX_GTH Transceivers User Guide.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 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。

```

48
49
50
51
52
53
54
55
56
57
58
59

SEND_HEADER:
begin
    gt_tx_data <= 32'hff_00_00_bc;
    gt_tx_ctrl <= 4'b0001;
    state <= SEND_SEQ_NUM;
    check_sum <= 32'd0;
end
SEND_SEQ_NUM:
begin
    gt_tx_data <= sequence_number;
    gt_tx_ctrl <= 4'b0000;
    state <= SEND_CTRL;
end

```

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 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 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 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) 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 GTX 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 gtwizard_ultrascale IP 的 example 工程中的 gtx_exdes.xdc 文件中修改而来，比如 GTH 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

```

set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]

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

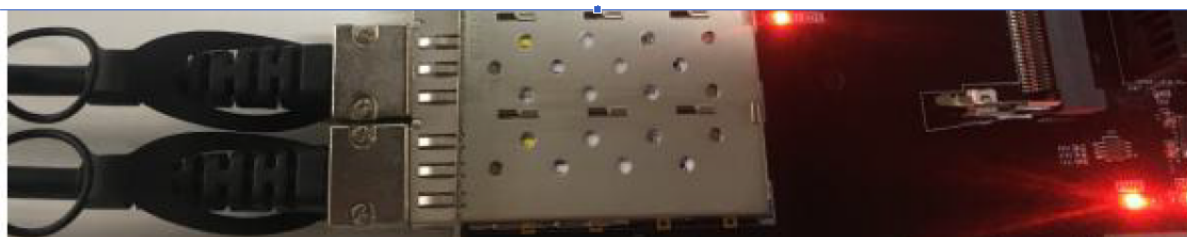
create_clock -period 5.000 [get_ports sys_clk_p]
set_property PACKAGE_PIN AK17 [get_ports sys_clk_p]
set_property IOSTANDARD DIFF_SSTL12 [get_ports sys_clk_p]
# UltraScale FPGAs Transceivers Wizard IP example design-level XDC file
# -----

# Location constraints for differential reference clock buffers
# Note: the IP core-level XDC constrains the transceiver channel data pin
# -----
set_property PACKAGE_PIN AF5 [get_ports mgtrefclk_n]
set_property PACKAGE_PIN AF6 [get_ports mgtrefclk_p]

```

4 光纤数据传输测试

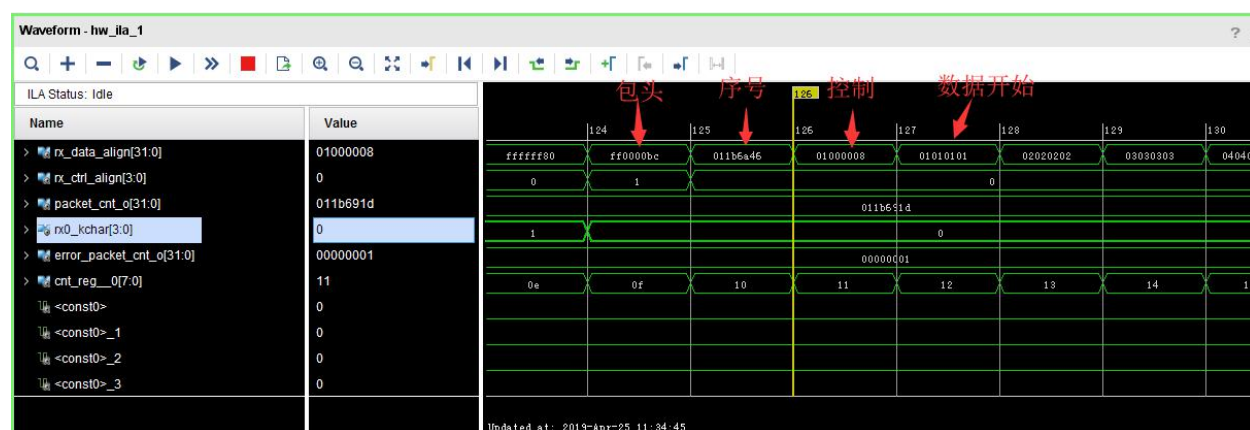
编译项目通过后我们就可以开始光纤数据传输的实验了。光模块插入到 AXKU040 开发板的 sfp1+与 sfp2+接口上 AXKU040 硬件连接后如下图所示：



再下载 bit 文件到 FPGA, 另外也没有弹出 ila 的测试界面。

点一下 Refresh device 按钮。

这时 ila 界面就会出现，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。

这里为止 GTH 光纤数据传输例程就介绍完了。如果用户需要对工程中的 gtH IP 重新配置的话，如果只是修改工程里的 gtH IP 配置的话，编译的时候会报错。用户需要在修改工程中的 gtH IP

配置的同时，还是需要重新生成 gtH 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。

_test ▾ gtx_lg_test ▾ gtx_lg_test.srcs ▾ sources_1 ▾ imports ▾ example_design ▾ support				
名称 ▲	修改日期	类型	大小	
 gtx_clock_module.v	2019/5/5 13:17	V 文件	9 KB	
 gtx_common.v	2019/5/5 13:17	V 文件	9 KB	
 gtx_common_reset.v	2019/5/5 13:17	V 文件	5 KB	
 gtx_gt_usrclk_source.v	2019/5/5 13:17	V 文件	8 KB	
 gtx_support.v	2019/5/5 13:17	V 文件	103 KB	