GTH Fiber Optic Communication Test Experiment

Contact Email: support@aithtech.com

1 Experiment Introduction

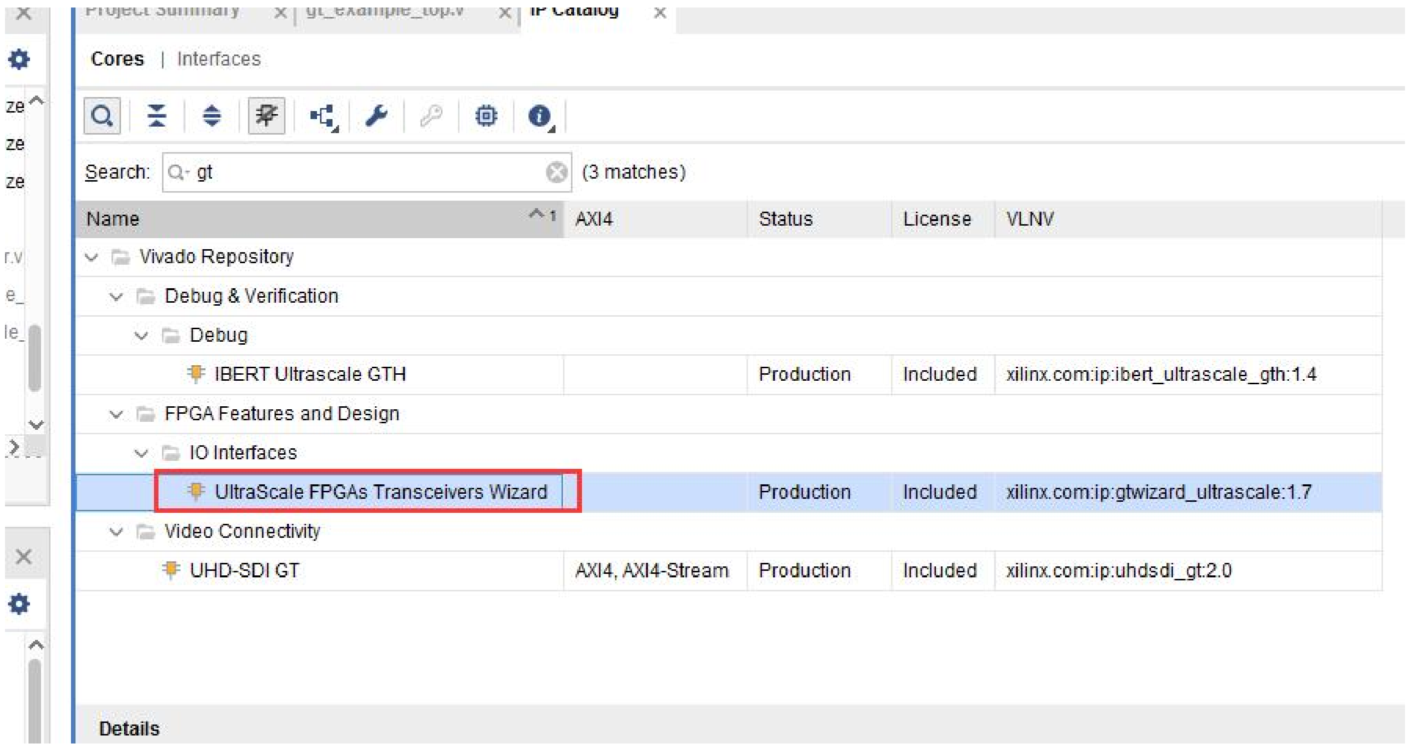
This experiment will introduce the transmission of data communication through optical fiber. The test data is generated by the FPGA itself and transmit by GTH to the first optical fiber port. Then, through the fiber loop to the second fiber port, the GTH/GTY receives data for verification.

2 Experimental principle

2.1 GTH IP Design

XILINX Vivado software has designed GTH IP for users, users can use IP to achieve high-speed data transmission and reception of GTH without paying attention to the internal work of GTH. The following is the specific GTH IP generation and configuration method.

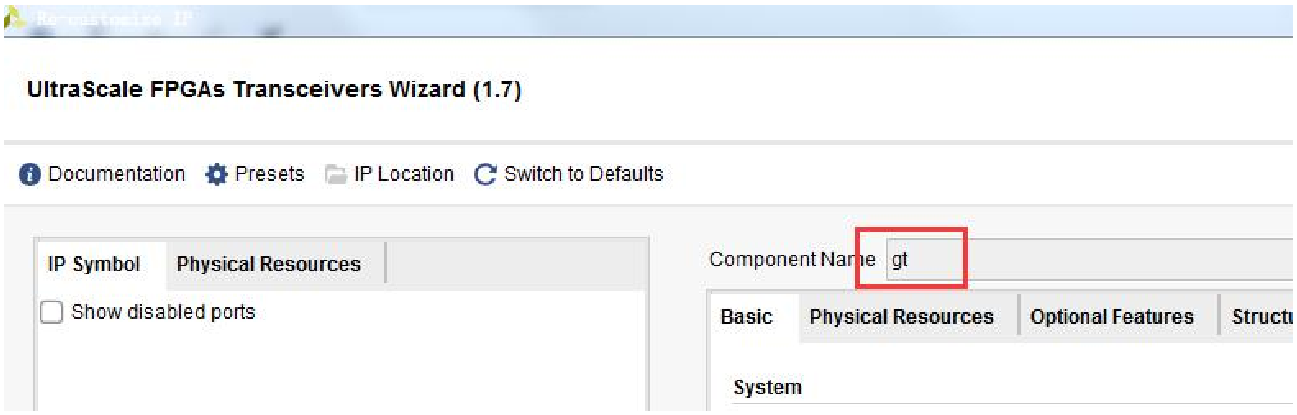
1) Double-click the "Ultrascale FPGAs Transceivers Wizard" icon in the “FPGA Features and Design\IO Interface” directory in the “IP Catalog” interface.



Search: gt (3 matches)

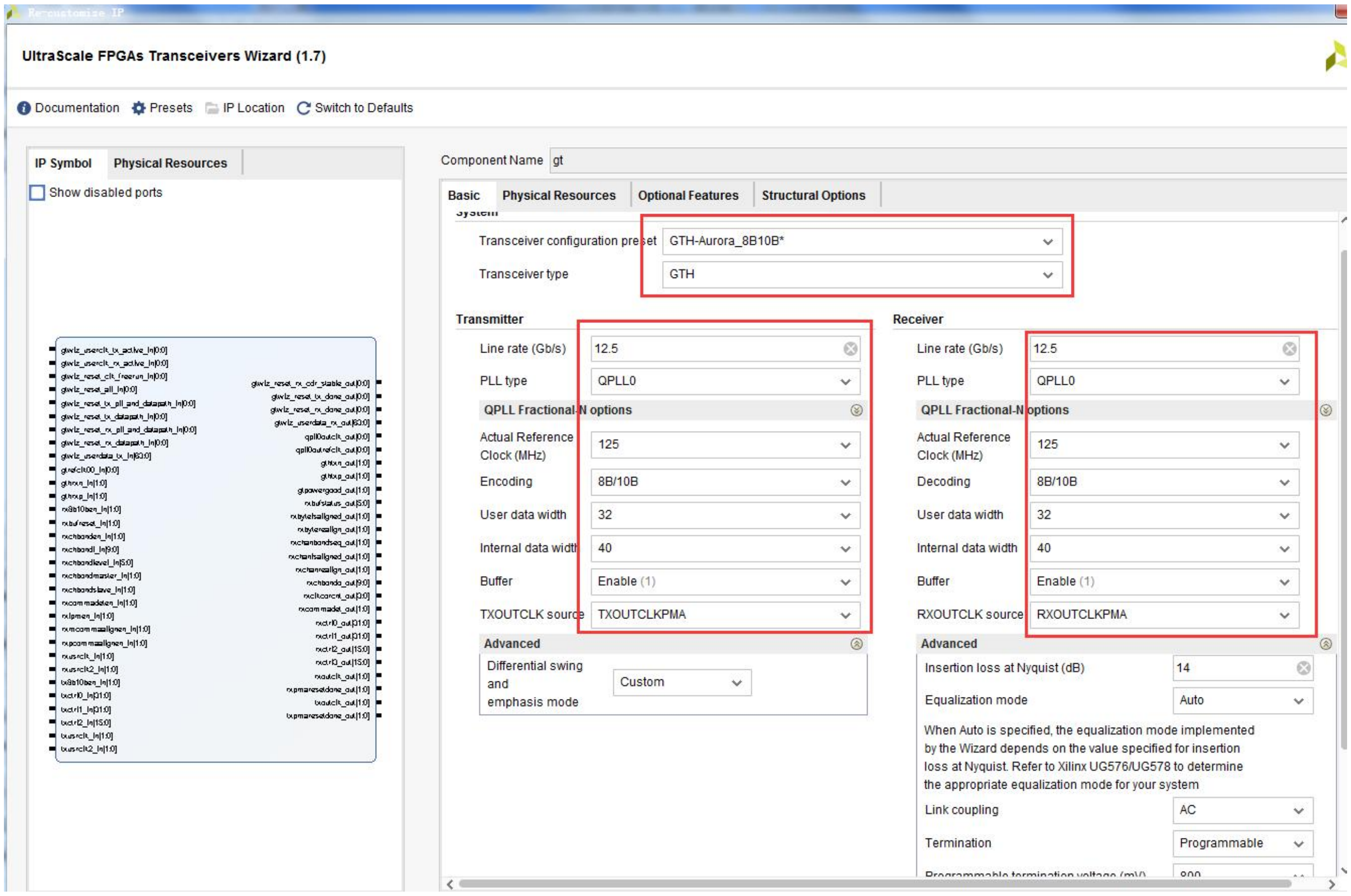
Name	AXI4	Status	License	VLNV
Vivado Repository				
Debug & Verification				
Debug				
IBERT Ultrascale GTH		Production	Included	xilinx.com:ip:ibert_ultrascale_gth:1.4
FPGA Features and Design				
IO Interfaces				
UltraScale FPGAs Transceivers Wizard		Production	Included	xilinx.com:ip:gtwizard_ultrascale:1.7
Video Connectivity				
UHD-SDI GT	AXI4, AXI4-Stream	Production	Included	xilinx.com:ip:uhdsdi_gt:2.0

2) In the Basic interface, Enter "gt" as the name in the "Component Name"



Component Name: gt

3) In the Basic interface, first set the transmission protocol bit "GTH-Aurora 8B10B", the Aurora 8B/10B protocol is an extensible, lightweight link layer protocol that can Data point-to-point transmission. The optical module transmission we use here is single-channel, so choose single lane, and the data interface is 4byte, which is 32-bit data. Then select the Line Rate speed of TX and RX and the type of PLL used, here select 12.5G and QPLL0. The reference clock selects 125Mhz according to the actual crystal oscillator on the board, the user data width is 32 bits, and it is 40 bits after the internal conversion through 8B/10B.



Component Name: gt

Transceiver configuration preset: GTH-Aurora_8B10B*
Transceiver type: GTH

Transmitter

Line rate (Gb/s)	12.5
PLL type	QPLL0
QPLL Fractional-N options	
Actual Reference Clock (MHz)	125
Encoding	8B/10B
User data width	32
Internal data width	40
Buffer	Enable (1)
TXOUTCLK source	TXOUTCLKPMA
Advanced	
Differential swing and emphasis mode	Custom

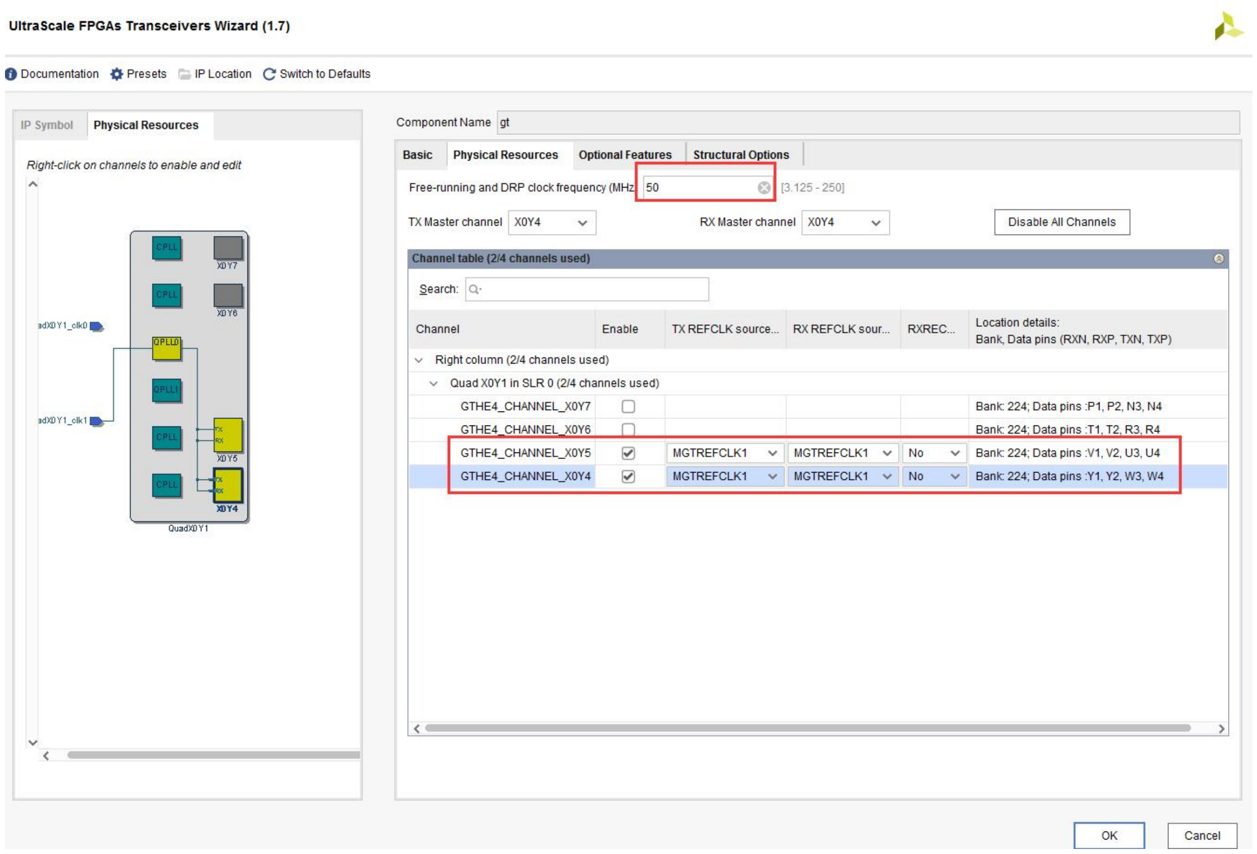
Receiver

Line rate (Gb/s)	12.5
PLL type	QPLL0
QPLL Fractional-N options	
Actual Reference Clock (MHz)	125
Decoding	8B/10B
User data width	32
Internal data width	40
Buffer	Enable (1)
RXOUTCLK source	RXOUTCLKPMA
Advanced	
Insertion loss at Nyquist (dB)	14
Equalization mode	Auto

When Auto is specified, the equalization mode implemented by the Wizard depends on the value specified for insertion loss at Nyquist. Refer to Xilinx UG576/UG578 to determine the appropriate equalization mode for your system

Link coupling	AC
Termination	Programmable
Programmable termination voltage (mV)	800

4) In the Physical Resources page, set the DRP clock to 50Mhz. Taking the AXU4EV-P development board as an example, the XCZU4EV-SFVC784 chip has only one BANK GTH transceiver (BANK224). In the AXU4EV-P hardware design, X0Y4 and X0Y5 of the BANK224 are connected to 2-way SFP+ optical modules, and the reference clock is 125Mhz connection to MGTREFCLK1. So here you need to select GTHE_CHANNEL_X0Y4, GTHE_CHANNEL_X0Y5, and select MGTREFCLK1 for the reference clock.



Component Name: gt

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

TX Master channel: X0Y4

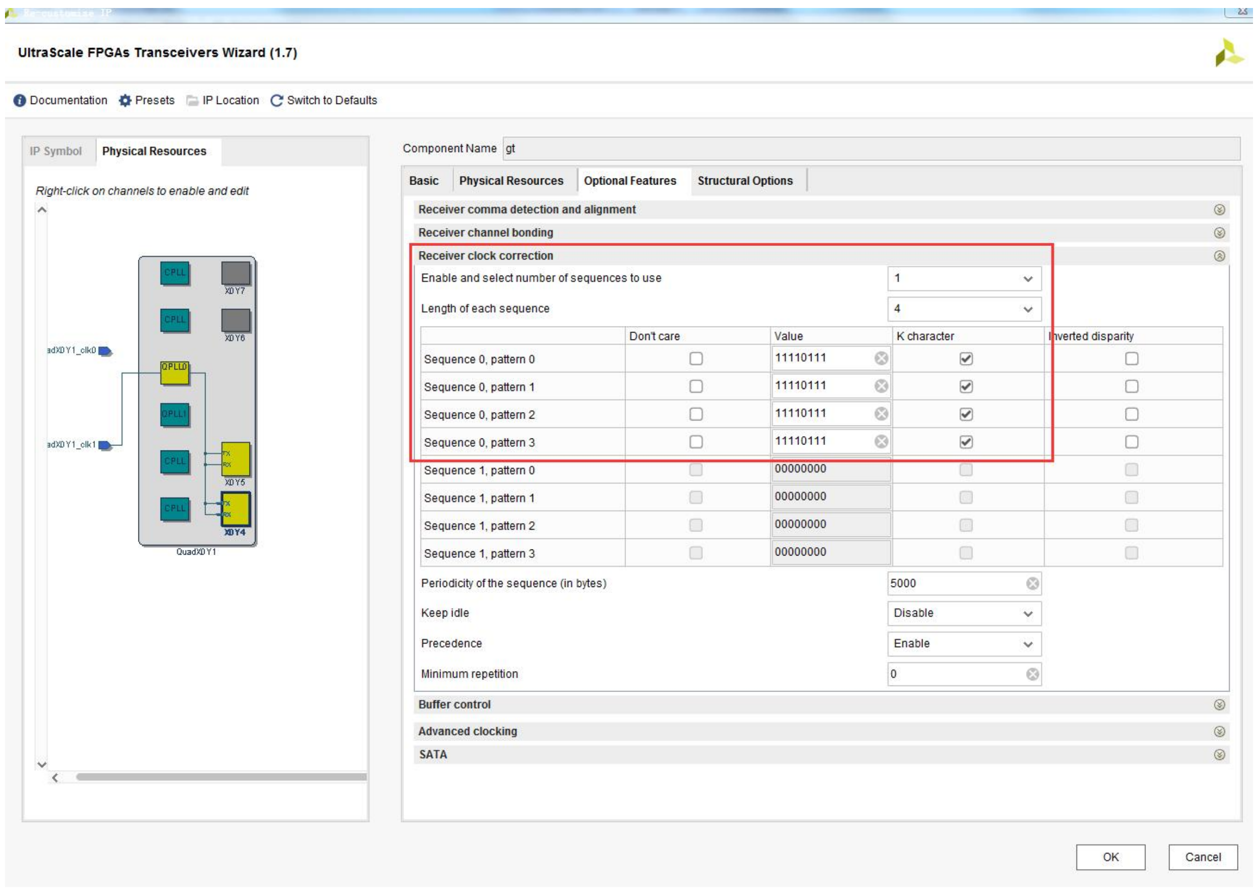
RX Master channel: X0Y4

Channel table (2/4 channels used)

Search:

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	☐				Bank: 224; Data pins :P1, P2, N3, N4
GTHE4_CHANNEL_X0Y6	☐				Bank: 224; Data pins :T1, T2, R3, R4
GTHE4_CHANNEL_X0Y5	✓	MGTREFCLK1	MGTREFCLK1	No	Bank: 224; Data pins :V1, V2, U3, U4
GTHE4_CHANNEL_X0Y4	✓	MGTREFCLK1	MGTREFCLK1	No	Bank: 224; Data pins :Y1, Y2, W3, W4

5) In the **Optional Features** interface, set the receiving clock check, where four **F7 K** codes are set as the receiving clock check. Therefore, when the optical fiber data is sent later, the clock check code of these 4 bytes needs to be added.



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

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	☐	11110111	✓	☐
Sequence 0, pattern 1	☐	11110111	✓	☐
Sequence 0, pattern 2	☐	11110111	✓	☐
Sequence 0, pattern 3	☐	11110111	✓	☐
Sequence 1, pattern 0	☐	00000000	☐	☐
Sequence 1, pattern 1	☐	00000000	☐	☐
Sequence 1, pattern 2	☐	00000000	☐	☐
Sequence 1, pattern 3	☐	00000000	☐	☐

Periodicity of the sequence (in bytes): 5000

Keep idle: Disable

Precedence: Enable

Minimum repetition: 0

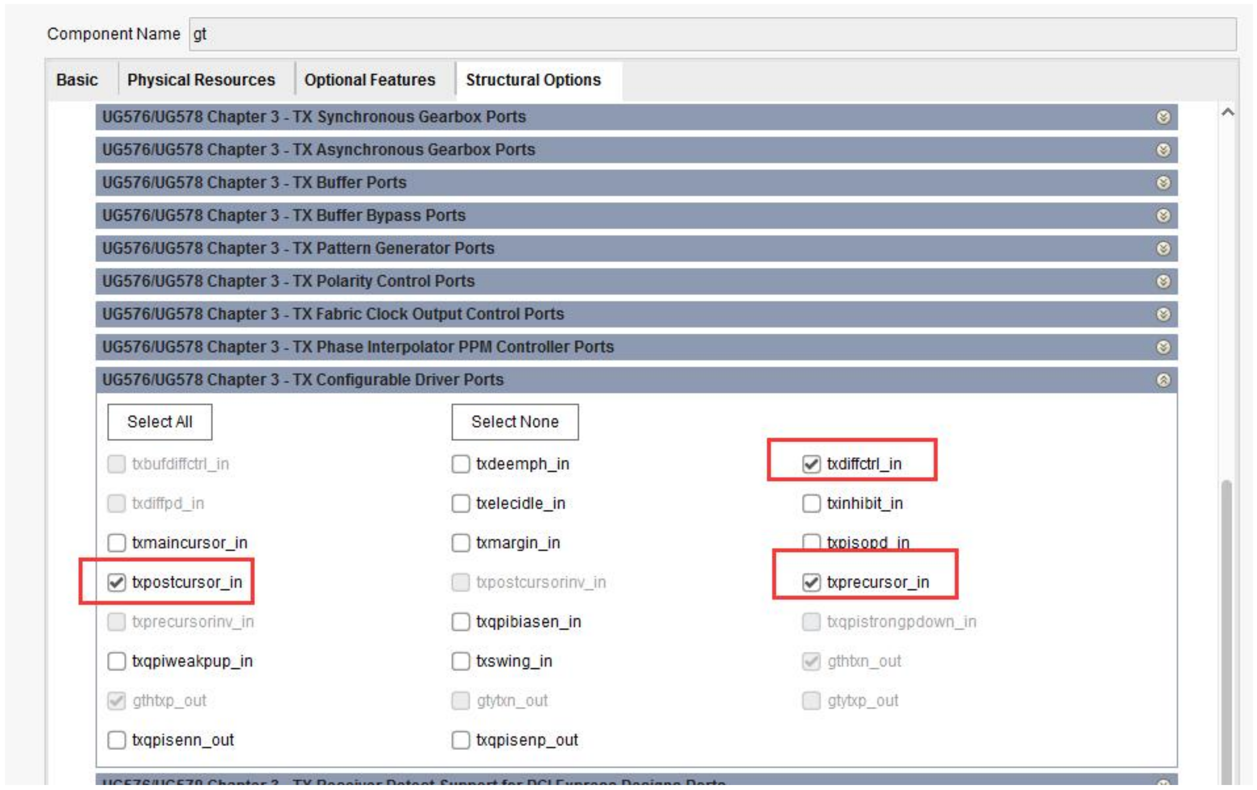
Buffer control

Advanced clocking

SATA

OK | Cancel

6) In the Structural Options interface, select the TX Configurable Driver Ports item, select the following items, and the IP will add some more signals to configure the drive capability of the sender.



Component Name: gt

Basic | Physical Resources | Optional Features | Structural Options

UG576/UG578 Chapter 3 - TX Synchronous Gearbox Ports

UG576/UG578 Chapter 3 - TX Asynchronous Gearbox Ports

UG576/UG578 Chapter 3 - TX Buffer Ports

UG576/UG578 Chapter 3 - TX Buffer Bypass Ports

UG576/UG578 Chapter 3 - TX Pattern Generator Ports

UG576/UG578 Chapter 3 - TX Polarity Control Ports

UG576/UG578 Chapter 3 - TX Fabric Clock Output Control Ports

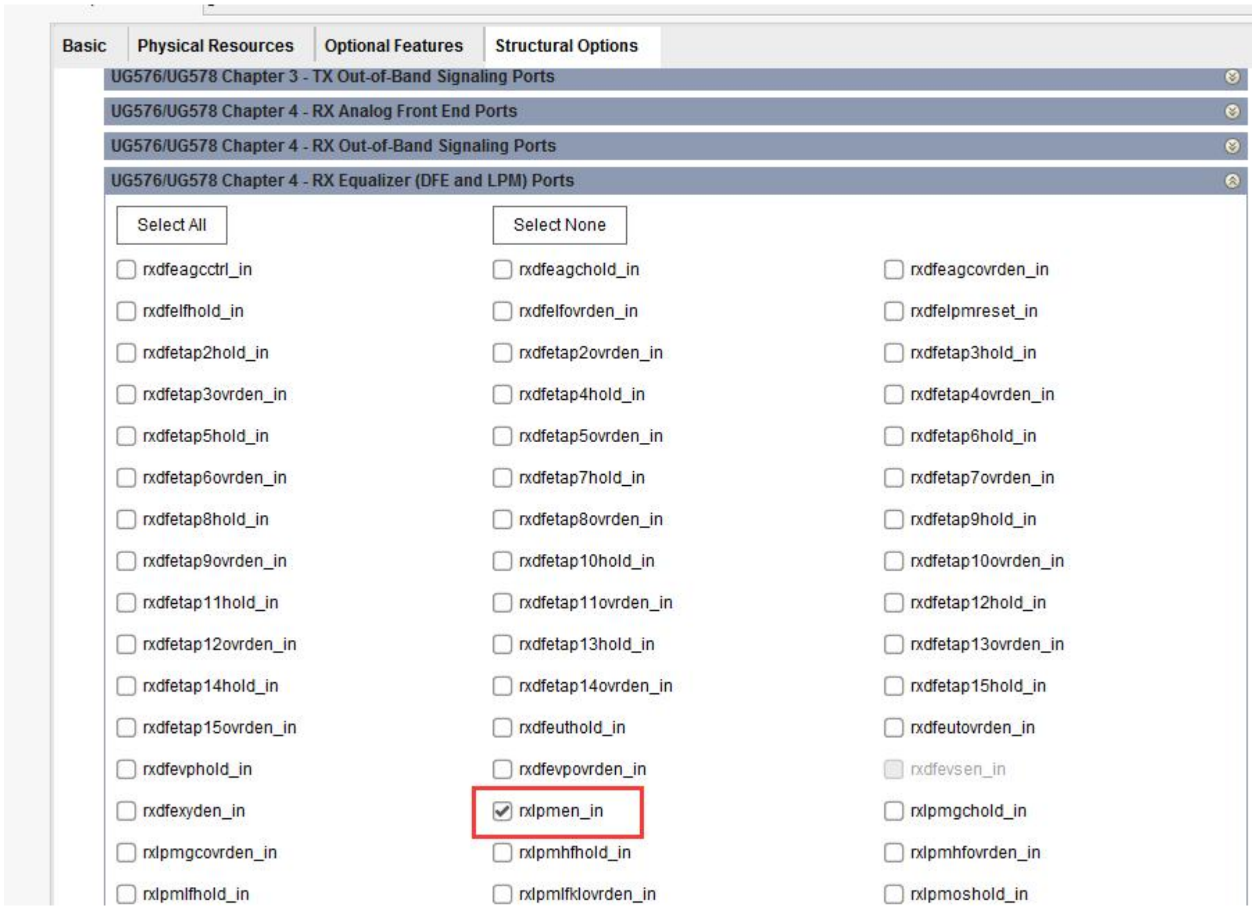
UG576/UG578 Chapter 3 - TX Phase Interpolator PPM Controller Ports

UG576/UG578 Chapter 3 - TX Configurable Driver Ports

Select All | Select None

☐ txbufdiffctrl_in	☐ txdeemph_in	✓ txdiffctrl_in
☐ txdiffpd_in	☐ txelecidle_in	☐ txinhibit_in
☐ txmaincursor_in	☐ txmargin_in	☐ txpisopd_in
✓ txpostcursor_in	☐ txpostcursorinv_in	✓ txprecursor_in
☐ txprecursorinv_in	☐ txqpibiasen_in	☐ txqpistrongpdown_in
☐ txqpiweakpup_in	☐ txswing_in	✓ gthtxn_out
✓ gthtxp_out	☐ gtytxn_out	☐ gtytxp_out
☐ txqpisenn_out	☐ txqpisenp_out	

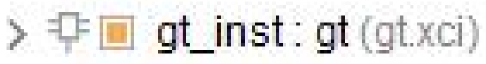
7) Select RX Equalizer(DFE and LPM) Ports item, select rxlpmen_in item, this pin can control DFE mode or LPM mode (default is DFE mode)


Basic
Physical Resources
Optional Features
Structural Options
UG576/UG578 Chapter 3 - TX Out-of-Band Signaling Ports
UG576/UG578 Chapter 4 - RX Analog Front End Ports
UG576/UG578 Chapter 4 - RX Out-of-Band Signaling Ports
UG576/UG578 Chapter 4 - RX Equalizer (DFE and LPM) Ports
Select All
Select None

☐ rxdfeagcctrl_in	☐ rxdfeagchold_in	☐ rxdfeagcovrden_in
☐ rxdfelfhold_in	☐ rxdfelfovrden_in	☐ rxdfelpmreset_in
☐ rxdfetap2hold_in	☐ rxdfetap2ovrden_in	☐ rxdfetap3hold_in
☐ rxdfetap3ovrden_in	☐ rxdfetap4hold_in	☐ rxdfetap4ovrden_in
☐ rxdfetap5hold_in	☐ rxdfetap5ovrden_in	☐ rxdfetap6hold_in
☐ rxdfetap6ovrden_in	☐ rxdfetap7hold_in	☐ rxdfetap7ovrden_in
☐ rxdfetap8hold_in	☐ rxdfetap8ovrden_in	☐ rxdfetap9hold_in
☐ rxdfetap9ovrden_in	☐ rxdfetap10hold_in	☐ rxdfetap10ovrden_in
☐ rxdfetap11hold_in	☐ rxdfetap11ovrden_in	☐ rxdfetap12hold_in
☐ rxdfetap12ovrden_in	☐ rxdfetap13hold_in	☐ rxdfetap13ovrden_in
☐ rxdfetap14hold_in	☐ rxdfetap14ovrden_in	☐ rxdfetap15hold_in
☐ rxdfetap15ovrden_in	☐ rxdfeuthold_in	☐ rxdfeutovrden_in
☐ rxdfevphold_in	☐ rxdfevpovrden_in	☐ rxdfevsen_in
☐ rxdfexyden_in	✓ rxlpmen_in	☐ rxlpmgchold_in
☐ rxlpmgcovrden_in	☐ rxlpmhfhold_in	☐ rxlpmhfovrden_in
☐ rxlpmlfhold_in	☐ rxlpmlfklovrden_in	☐ rxlpmoshold_in

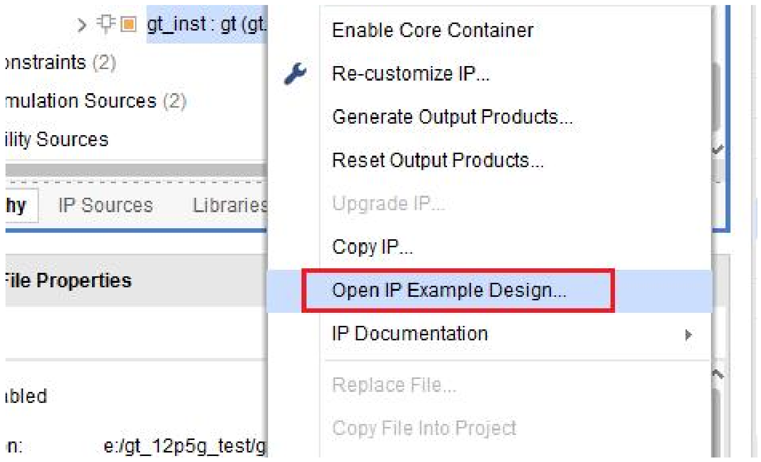
For more information on GTH IP configuration, please refer to the documentation provided by Xilinx "PG182-UltraScale FPGA Transceivers Wizard" and "ug576-ultrascale-gth-transceivers". These two documents have detailed introduction on various configuration parameters for GTH IP.

8) After the configuration is completed, the IP of the gt just generated is automatically added to the project.

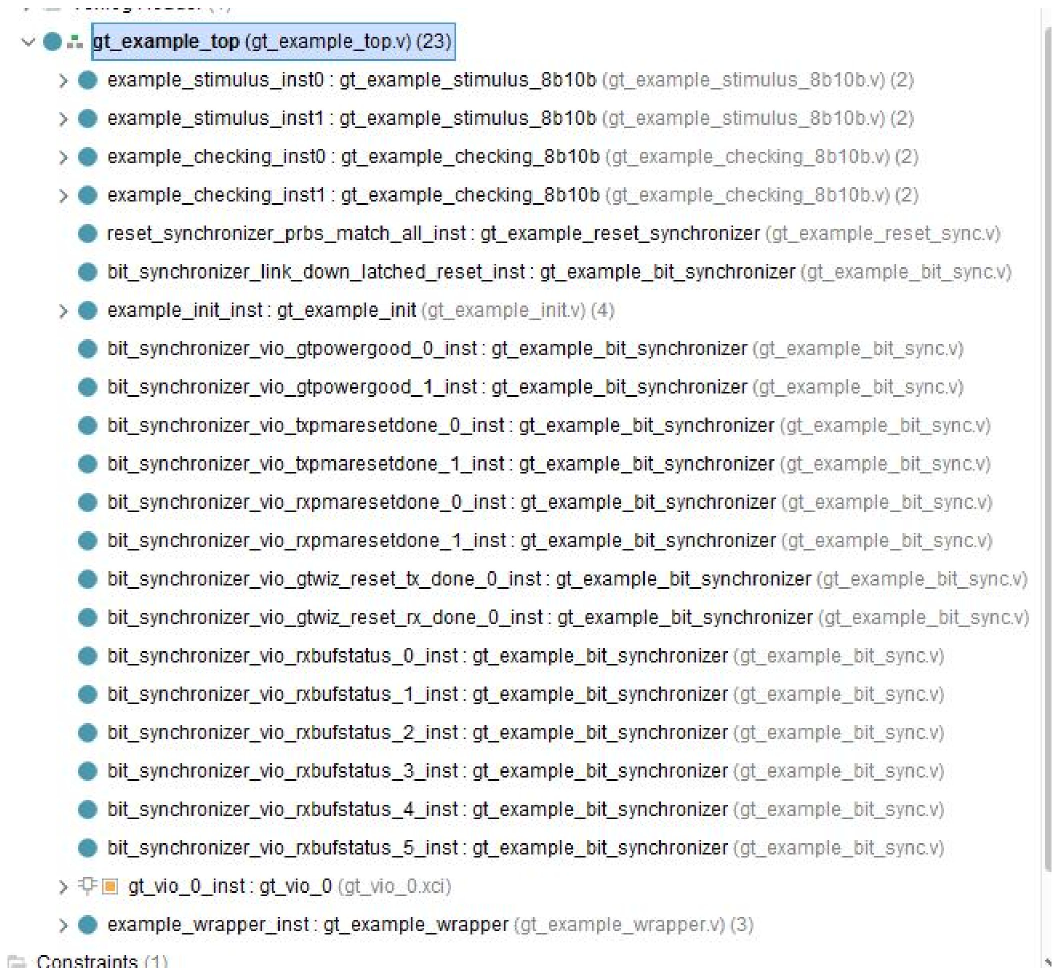


gt_inst : gt (gt.xci)

9) Let's generate an example project of GT IP, right-click to select gt, and select "Open IP Example Design..." in the drop-down menu.



10) The generated example project is shown in the following figure:



gt_example_top (gt_example_top.v) (23)
example_stimulus_inst0 : gt_example_stimulus_8b10b (gt_example_stimulus_8b10b.v) (2)
example_stimulus_inst1 : gt_example_stimulus_8b10b (gt_example_stimulus_8b10b.v) (2)
example_checking_inst0 : gt_example_checking_8b10b (gt_example_checking_8b10b.v) (2)
example_checking_inst1 : gt_example_checking_8b10b (gt_example_checking_8b10b.v) (2)
reset_synchronizer_prbs_match_all_inst : gt_example_reset_synchronizer (gt_example_reset_sync.v)
bit_synchronizer_link_down_latched_reset_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
example_init_inst : gt_example_init (gt_example_init.v) (4)
bit_synchronizer_vio_gtpowergood_0_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
bit_synchronizer_vio_gtpowergood_1_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
bit_synchronizer_vio_txpmaresetdone_0_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
bit_synchronizer_vio_txpmaresetdone_1_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
bit_synchronizer_vio_rxpmaresetdone_0_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
bit_synchronizer_vio_rxpmaresetdone_1_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
bit_synchronizer_vio_gtwiz_reset_tx_done_0_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
bit_synchronizer_vio_gtwiz_reset_rx_done_0_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
bit_synchronizer_vio_rxbufstatus_0_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
bit_synchronizer_vio_rxbufstatus_1_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
bit_synchronizer_vio_rxbufstatus_2_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
bit_synchronizer_vio_rxbufstatus_3_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
bit_synchronizer_vio_rxbufstatus_4_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
bit_synchronizer_vio_rxbufstatus_5_inst : gt_example_bit_synchronizer (gt_example_bit_sync.v)
gt_vio_0_inst : gt_vio_0 (gt_vio_0.xci)
example_wrapper_inst : gt_example_wrapper (gt_example_wrapper.v) (3)
Constraints (1)

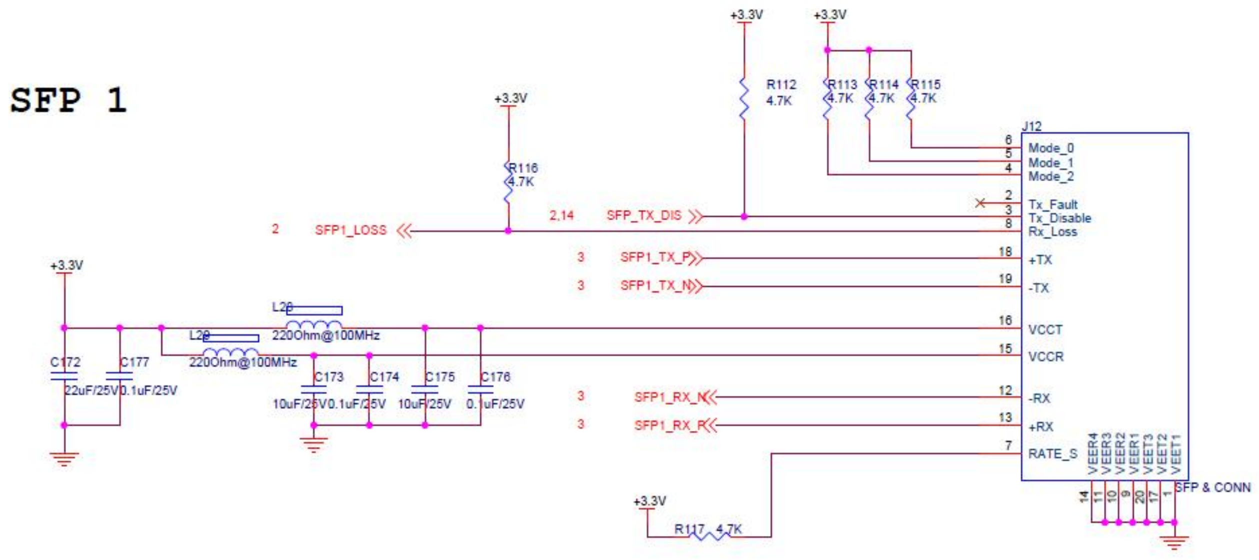
In this example project, the gt_example_stimulus_8b10b program is the data module for generating the tested PRBS31, and the data of PRBS31 is used for GTH transmission. The program gt_example_checking_8b10b checks whether the received PRBS31 data is correct. We will not introduce the engineering code and test of the example here. You can read the PG182 document for the functions of some modules and signals. In the following GTH data transmission, we will make modifications on the basis of this example project.

2.2 Hardware Introduction

On the AXU4EV-P FPGA development board, there are two fiber optic interfaces SFP1~SFP4 , which are respectively connected to the GTH channel of the FPGA chip.

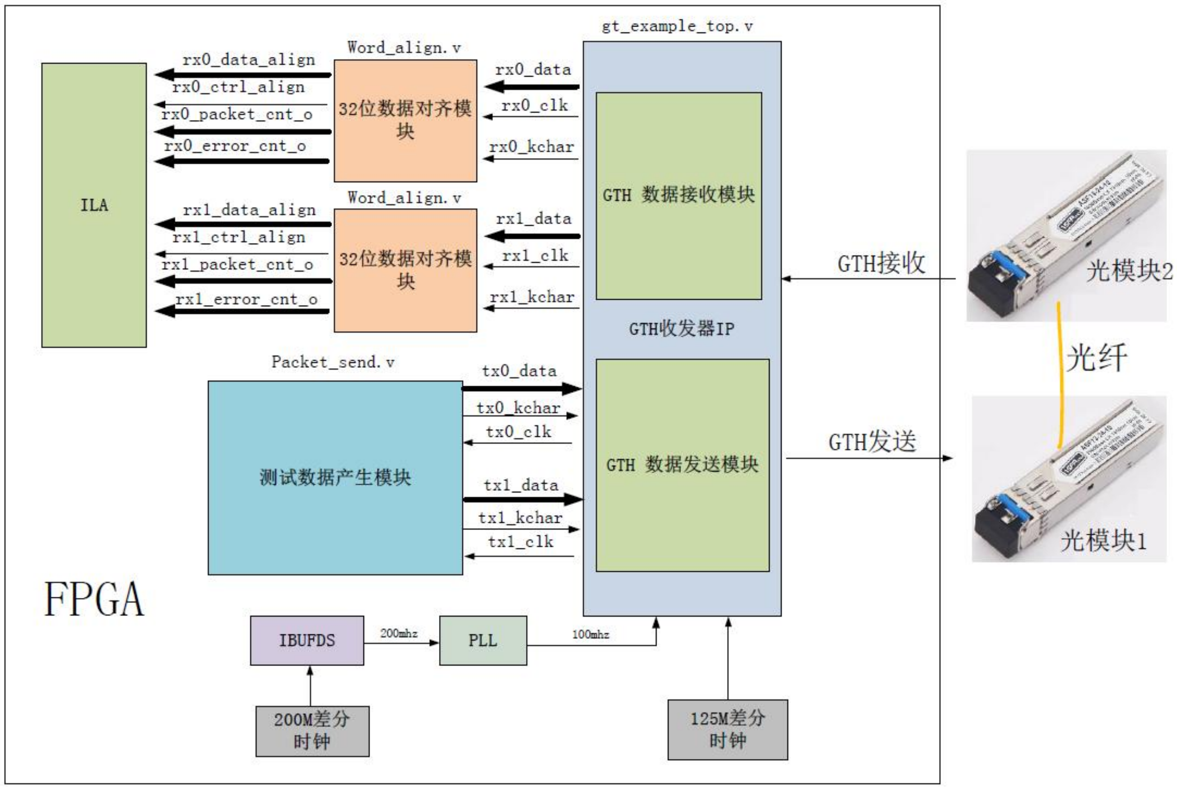
The SFP1 optical module interface is connected to GTH Channel0, SFP2 is connected to GTH Channel1.

The TX_Disable signal and RX_LOSS of the optical module are connected to the common IO of the FPGA. The TX_Disable signal is used to enable or disable the optical transmission of the optical module. If the TX_Disable signal is high, the optical transmission is turned off, otherwise the optical transmission is enabled, and this signal needs to be pulled down during normal use. The hardware schematic diagram of SFP1 optical module is as follows:



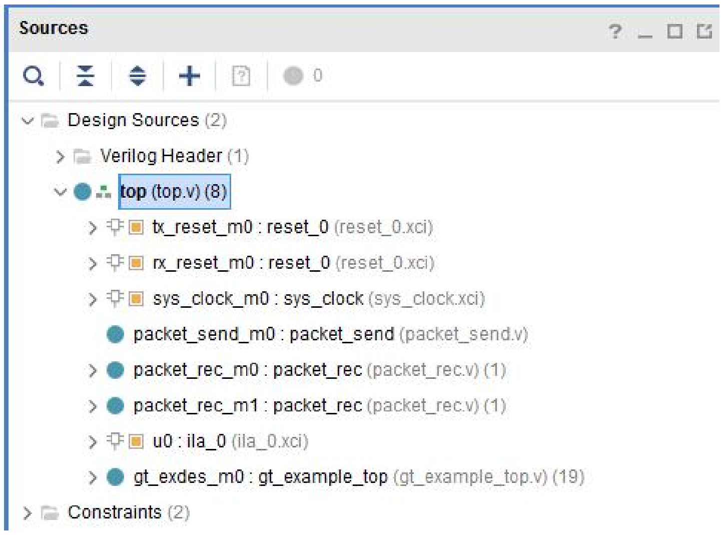
3 Programming

The FPGA programming of optical fiber data transmission is to add a TOP file and 3 .v files on the basis of the project code of the example. The logical block diagram of the project is shown in the following figure:



The program generates data and sends it to the external optical film block 1 using GTH IP. The optical module 1 converts the electrical signal into an optical signal and transmits it to the optical module 2 through the optical fiber. The optical module 2 converts the optical signal into an electrical signal and inputs it to the GTH receiving of the FPGA. The data received by the GTH needs to be aligned with a 32-bit data, and then the data signal and the control data signal are parsed. The verification module performs a check calculation on the data signal and the control signal to determine whether the received data and the transmitted data are consistent.

The design of the fiber optic data transmission is as follows:

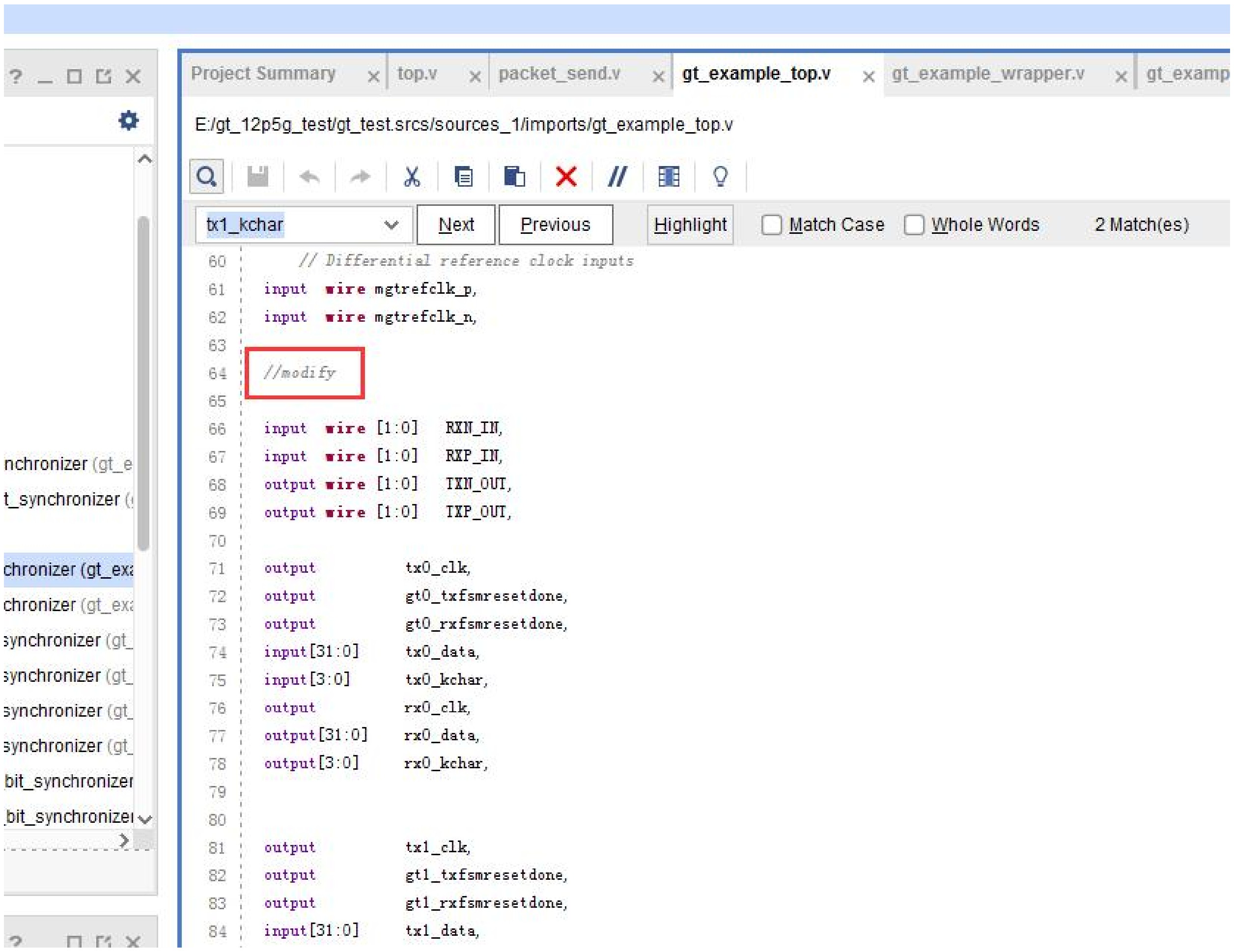

```
Sources
Design Sources (2)
  Verilog Header (1)
  top (top.v) (8)
    tx_reset_m0 : reset_0 (reset_0.xci)
    rx_reset_m0 : reset_0 (reset_0.xci)
    sys_clock_m0 : sys_clock (sys_clock.xci)
    packet_send_m0 : packet_send (packet_send.v)
    packet_rec_m0 : packet_rec (packet_rec.v) (1)
    packet_rec_m1 : packet_rec (packet_rec.v) (1)
    u0 : ila_0 (ila_0.xci)
    gt_exdes_m0 : gt_example_top (gt_example_top.v) (19)
Constraints (2)
```

The project here adds an ILA tool that can be used to view the received data.

1) gth data communication module

In the previous example we have generated the gth IP example project, where the **gt_example_stimulus_8b10b.v** module and the **gt_example_checking_8b10b.v** module have been removed. Because these two are the test data generation and inspection modules, this experiment is not used. In addition, the “gtp_exdes.v” file is modified, mainly to delete the instantiation of the “gt0_frame_gen.v” module and the “gt0_frame_check.v” module,Then the interface signal is connected with the signal of the original data generating module and the checking module. We have added //Modify for the specific modification, you can compare it with the source file.

➢ For example, the following user interface signal is added to the Port of the module:



E:/gt_12p5g_test/gt_test.srcs/sources_1/imports/gt_example_top.v

```
    // Differential reference clock inputs
input  wire mgtrefclk_p,
input  wire mgtrefclk_n,

//modify

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

output          tx0_clk,
output          gt0_txfsmresetdone,
output          gt0_rxfsmresetdone,
input[31:0]     tx0_data,
input[3:0]      tx0_kchar,
output          rx0_clk,
output[31:0]    rx0_data,
output[3:0]     rx0_kchar,


output          tx1_clk,
output          gt1_txfsmresetdone,
output          gt1_rxfsmresetdone,
input[31:0]     tx1_data,
```

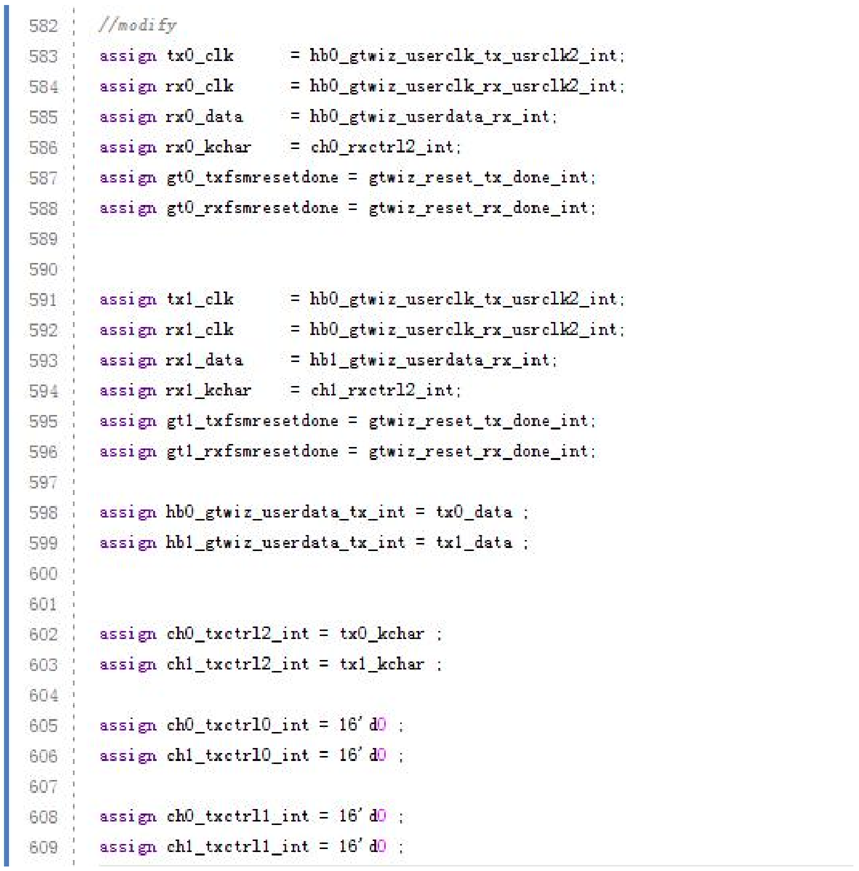
The following is an introduction to the user interface signals. The following takes the GTH interface of channel0 as an example:

Signal Name	Bits	Input/Output	Description
tx0_clk	1	Output	The data transmission clock, which is the TXUSRCLK2 described in Section 14, has a GTH reference clock of 312.5Mhz. The data is valid on the rising edge.

tx0_data	32	Input	GTP Transmits data
tx0_kchar	4	Input	The K control word transmitted by GTH to indicate whether the transmitted data is a K code control character or a normal transmission data. A high level indicates a K code control character and 4 bits correspond to 4 Bytes of the transmitted data. Tx0_kchar [3] corresponds to tx0_data [31:24] Tx0_kchar [2] corresponds to tx0_data [23:16] Tx0_kchar [1] corresponds to tx0_data [15:8] Tx0_kchar [0] corresponds to tx0_data [7:0]
rx0_clk	1	Output	The data receive clock, which is the RXUSRCLK2 described in Section 14, has a GTH reference clock of 312.5Mhz. Data is valid on the rising edge
rx0_data	32	Output	GTH receives data.
rx0_kchar	4	Output	GTH receives the K control word to indicate whether the received data is a K code control character or a normal transmission data. A high level indicates a K code control character, and 4 bits correspond to 4 Bytes of 32 bits of received data. rx0_kchar [3] corresponds to rx0_data [31:24] rx0_kchar [2] corresponds to rx0_data [23:16] rx0_kchar [1] corresponds to rx0_data [15:8] rx0_kchar [0] corresponds to rx0_data [7:0]
gt0_txfsmresetdone	1	Output	GTH initialization completion signal

Knowing the meaning of the GTH user interface signal, we can send and receive fiber data through the user interface.

➢ Complete the connection of user interface signals and GTH IP through the following connections.

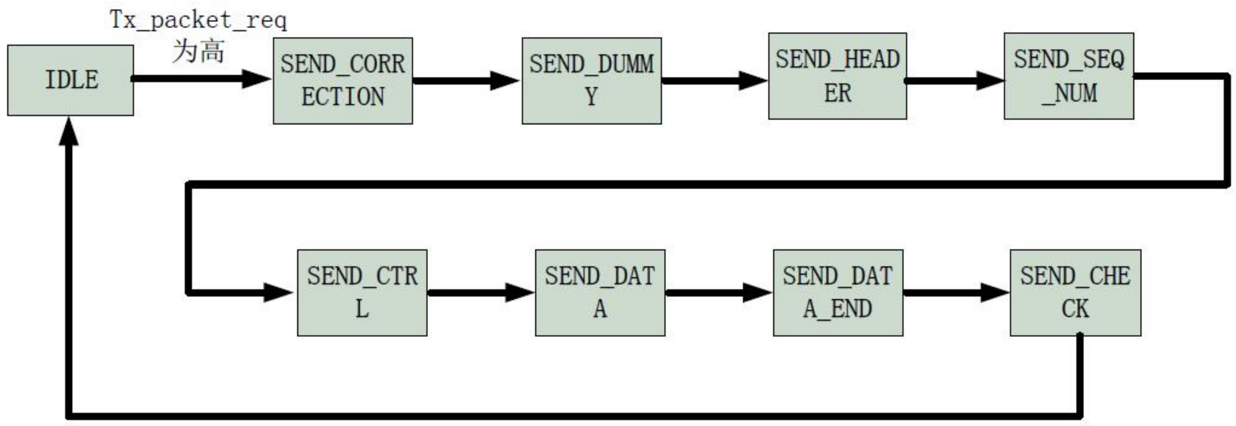


```
//modify
assign tx0_clk      = hb0_gtwiz_userclk_tx_usrclk2_int;
assign rx0_clk      = hb0_gtwiz_userclk_rx_usrclk2_int;
assign rx0_data     = hb0_gtwiz_userdata_rx_int;
assign rx0_kchar    = ch0_rxctrl2_int;
assign gt0_txfsmresetdone = gtwiz_reset_tx_done_int;
assign gt0_rxfsmresetdone = gtwiz_reset_rx_done_int;


assign tx1_clk      = hb0_gtwiz_userclk_tx_usrclk2_int;
assign rx1_clk      = hb0_gtwiz_userclk_rx_usrclk2_int;
assign rx1_data     = hb1_gtwiz_userdata_rx_int;
assign rx1_kchar    = ch1_rxctrl2_int;
assign gt1_txfsmresetdone = gtwiz_reset_tx_done_int;
assign gt1_rxfsmresetdone = gtwiz_reset_rx_done_int;

assign hb0_gtwiz_userdata_tx_int = tx0_data ;
assign hb1_gtwiz_userdata_tx_int = tx1_data ;


assign ch0_txctrl2_int = tx0_kchar ;
assign ch1_txctrl2_int = tx1_kchar ;

assign ch0_txctrl0_int = 16'd0 ;
assign ch1_txctrl0_int = 16'd0 ;

assign ch0_txctrl1_int = 16'd0 ;
assign ch1_txctrl1_int = 16'd0 ;
```

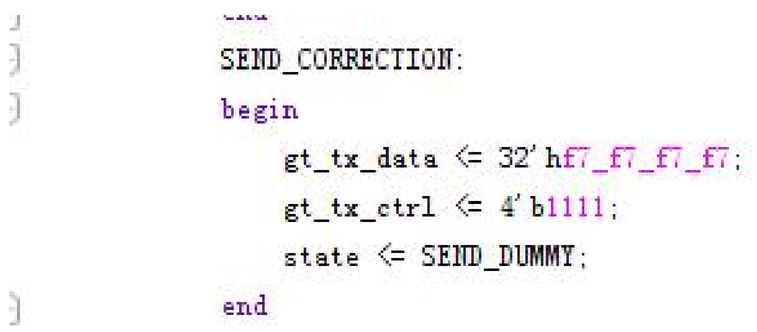
For the signals used in the program, users can refer to the "ug576-ultrascale-gth-transceivers.pdf" document for query. For example, page P117 introduces the function of the 8-bit txctrl2 signal as the K code corresponding to 64-bit data.

2) gth packet transmitting module packet_send.v

In the packet_send.v, the test data is transmitted by a state machine. First, before the data is transmitted, the synchronization packet header signal is transmit first, and then the sequence number and control signal of the data packet are transmit. The test data is then transmit out via GTH. The sending process is as follows:

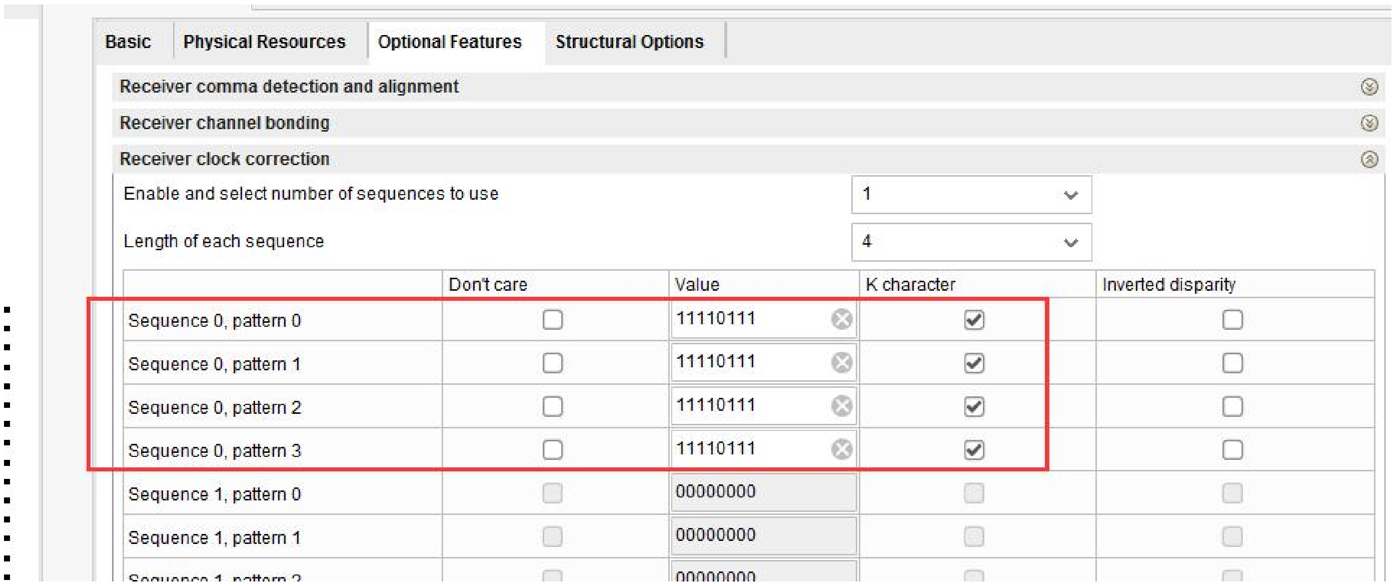


All data bits transmitted by GTH are 32 bits. Before transmitting data packets, 32-bit data correction control word needs to be sent. The correction control word is used to eliminate the frequency error caused by different system clocks. The gt_tx_ctrl signal is set to 1111, indicating that these 32-bit data are all control words



```
        SEND_CORRECTION:
        begin
            gt_tx_data <= 32'hf7_f7_f7_f7;
            gt_tx_ctrl <= 4'b1111;
            state <= SEND_DUMMY;
        end
```

The 32-bit correction word (F7F7F7F7) has been configured in the GTH IP. So when GTH receives data packets, it will automatically make corrections



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	☐	11110111	✓	☐
Sequence 0, pattern 1	☐	11110111	✓	☐
Sequence 0, pattern 2	☐	11110111	✓	☐
Sequence 0, pattern 3	☐	11110111	✓	☐
Sequence 1, pattern 0	☐	00000000	☐	☐
Sequence 1, pattern 1	☐	00000000	☐	☐
Sequence 1, pattern 2	☐	00000000	☐	☐

The synchronization signal packet header signal is defined as 32-bit "ff_00_00_bc", and the lower 8-bit "bc" is the K28.5 code control character. The K-code feature word definition is described in Xilinx's "ug576-ultrascale-gth-transceivers.pdf" document.

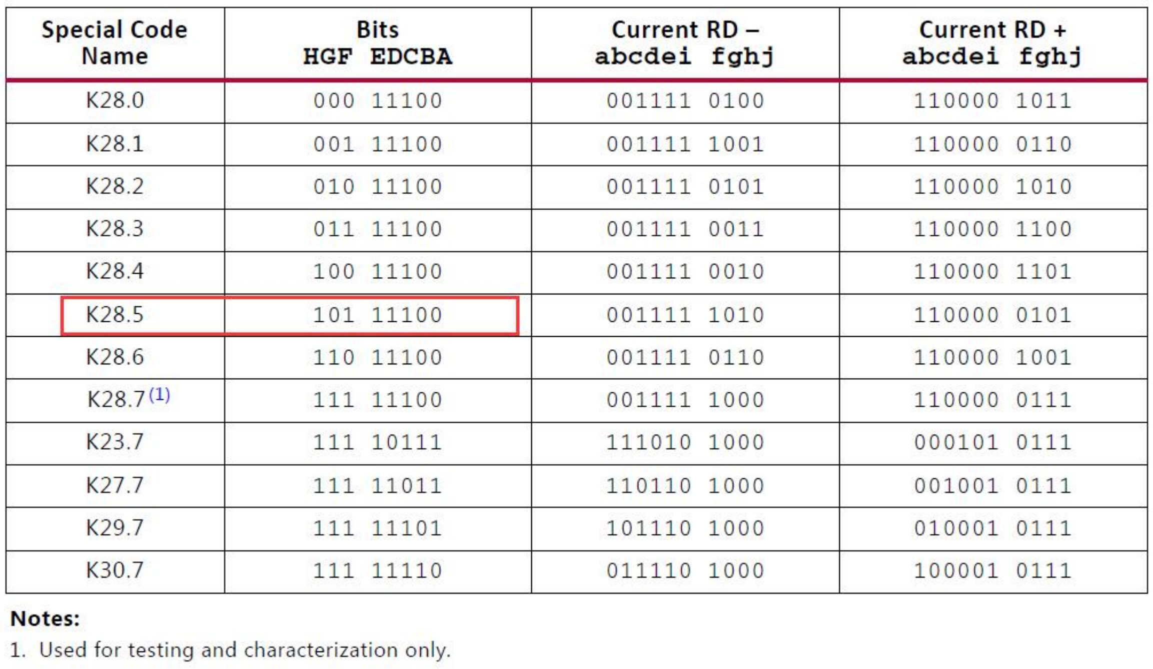


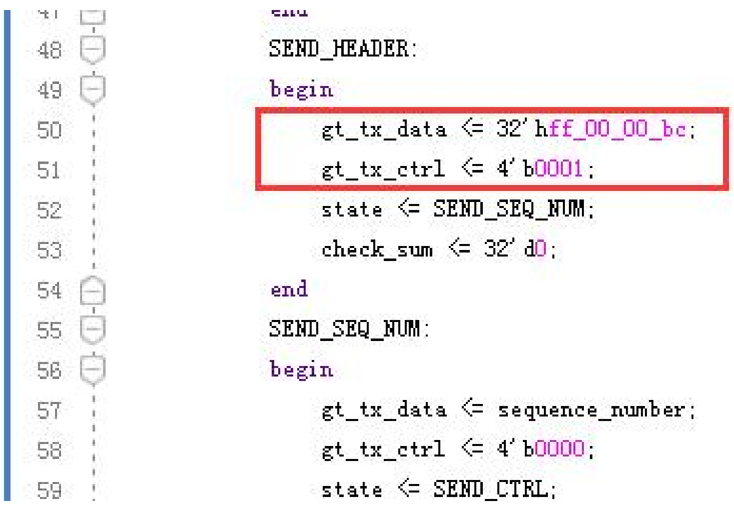
Table A-2: **Valid Control K Characters**

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.

When transmitting K28.5 code control characters to GTP, you need to raise the corresponding bit of the gt_tx_ctrl signal to indicate that a certain byte in the transmitted data is the K code control word. So here when transmitting a sync signal to GTP, the gt_tx_ctrl signal is set to 0001, and when other data is transmitted, it is set to 0000.



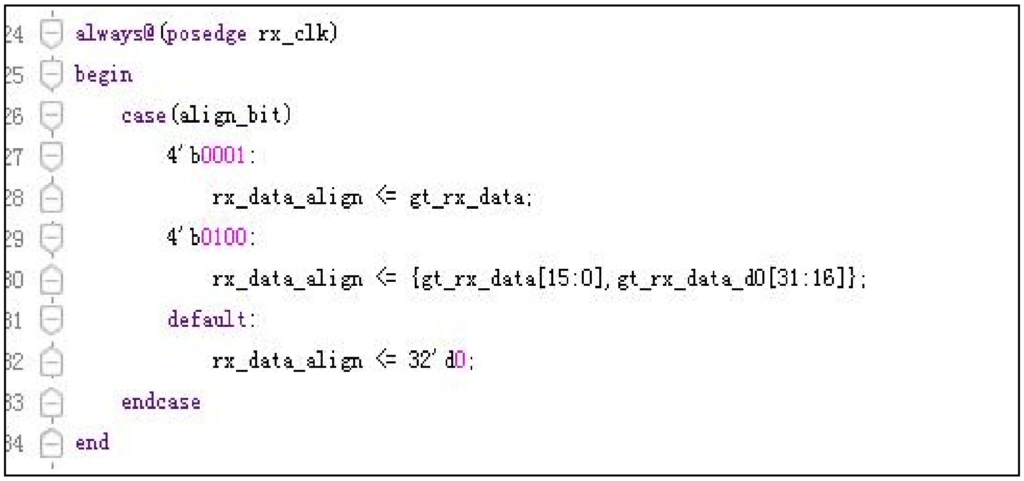
```
end
SEND_HEADER:
begin
    gt_tx_data <= 32'hff_00_00_bc;
    gt_tx_ctrl <= 4'b0001;
    state <= SEND_SEQ_NUM;
    check_sum <= 32'd0;
end
SEND_SEQ_NUM:
begin
    gt_tx_data <= sequence_number;
    gt_tx_ctrl <= 4'b0000;
    state <= SEND_CTRL;
```

3) Bit data alignment module word_align.v

The external user data interface of the GTH transceiver has a width of 32 bits and an internal data width of 20 bits (8b/10b conversion). During the actual test, it is found that the 32-bit data transmitted may have 16-bit data shift, that is, the transmitted data and the received data will have 16-bit misalignment. The following table shows the GTH transmit data and receive data shifts:

GTH Transmitted Data		GTH Received data	
Data 1	11111111	Data 1	11112222
Data 2	22222222	Data 2	22223333
Data 3	33333333	Data 3	33334444
Data 4	44444444	Data 4	44445555
Data 5	55555555	Data 5	5555.....
.....	.....	.....	.....

Because we added the K code control word when the GTH transmits the sync signal and the useless data, and sets the gt_tx_ctrl signal to 0001, if there is a 16-bit data shift, the received sync signal and the useless data, the K code control word It will also shift, and the signal of gt_tx_ctrl will become 0100. So we can judge whether the received GTH data is shifted by judging the value of the gt_tx_ctrl signal. If the received gt_tx_ctrl is 0001, it is the same as when we transmit it, indicating that the data is not shifted; if the received gt_tx_ctrl is 0100 .The received data is shifted and needs to be recombined and done in the word_align.v module.

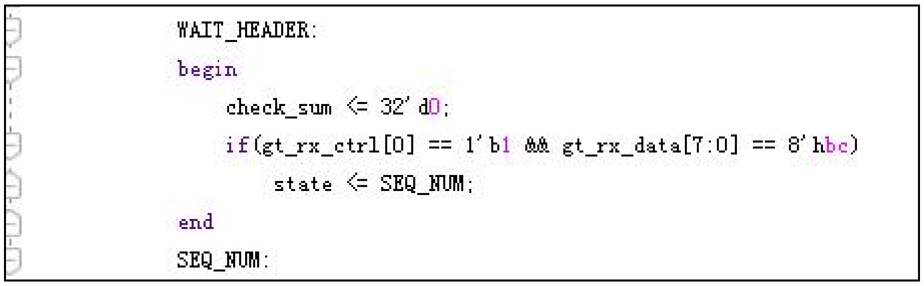


```
always@(posedge rx_clk)
begin
    case(align_bit)
        4'b0001:
            rx_data_align <= gt_rx_data;
        4'b0100:
            rx_data_align <= {gt_rx_data[15:0],gt_rx_data_d0[31:16]};
        default:
            rx_data_align <= 32'd0;
    endcase
end
```

4) GTH Data Parsing Module packet_rec.v

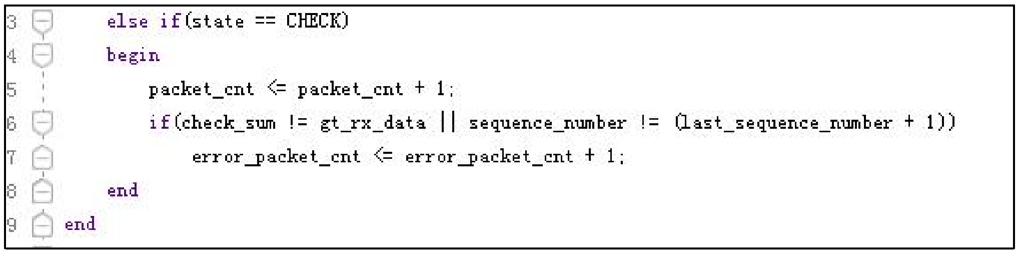
Because only a part of the received 32-bit data is valid data, the other is the synchronization header, sequence data, control data and Checksum. The checksum of the data is calculated in the packet_rec.v module, and then compared with the received checksum value. If it is not correct, it will generate a data error signal.

One function of the program is to detect the synchronization header signal in the GTH data (data is ff_00_02_bc), and if a synchronization header signal is received, start receiving a packet of data.



```
            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:
```

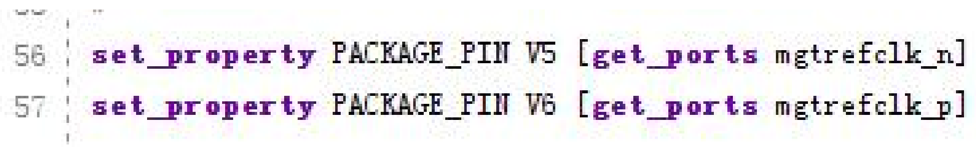
Another function of the program is to determine the checksum of the statistics and the received checksum.



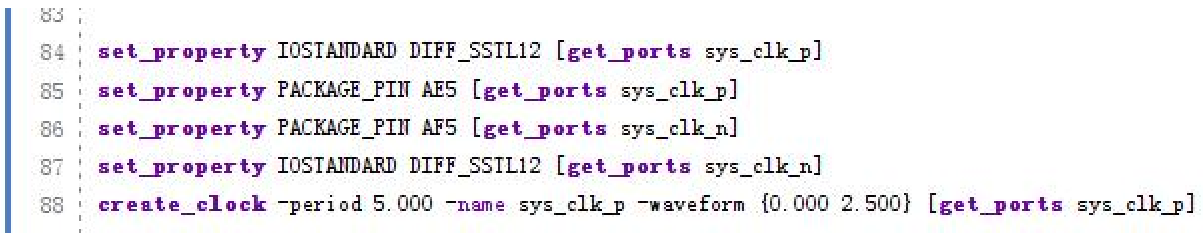
```
    else if(state == CHECK)
    begin
        packet_cnt <= packet_cnt + 1;
        if(check_sum != gt_rx_data || sequence_number != (last_sequence_number + 1))
            error_packet_cnt <= error_packet_cnt + 1;
    end
end
```

5) Pin constraints

The pin constraints here are modified from the gt_example_top.xdc file in the example project of GTH IP. For example, the reference clock of GTH is 125Mhz and the system clock is 200Mhz, which needs to correspond to the pins on the development board.

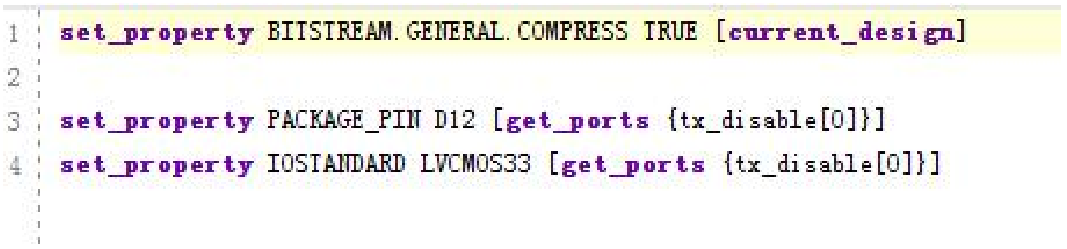


```
set_property PACKAGE_PIN V5 [get_ports mgtrefclk_n]
set_property PACKAGE_PIN V6 [get_ports mgtrefclk_p]
```



```
set_property IOSTANDARD DIFF_SSTL12 [get_ports sys_clk_p]
set_property PACKAGE_PIN AE5 [get_ports sys_clk_p]
set_property PACKAGE_PIN AF5 [get_ports sys_clk_n]
set_property IOSTANDARD DIFF_SSTL12 [get_ports sys_clk_n]
create_clock -period 5.000 -name sys_clk_p -waveform {0.000 2.500} [get_ports sys_clk_p]
```

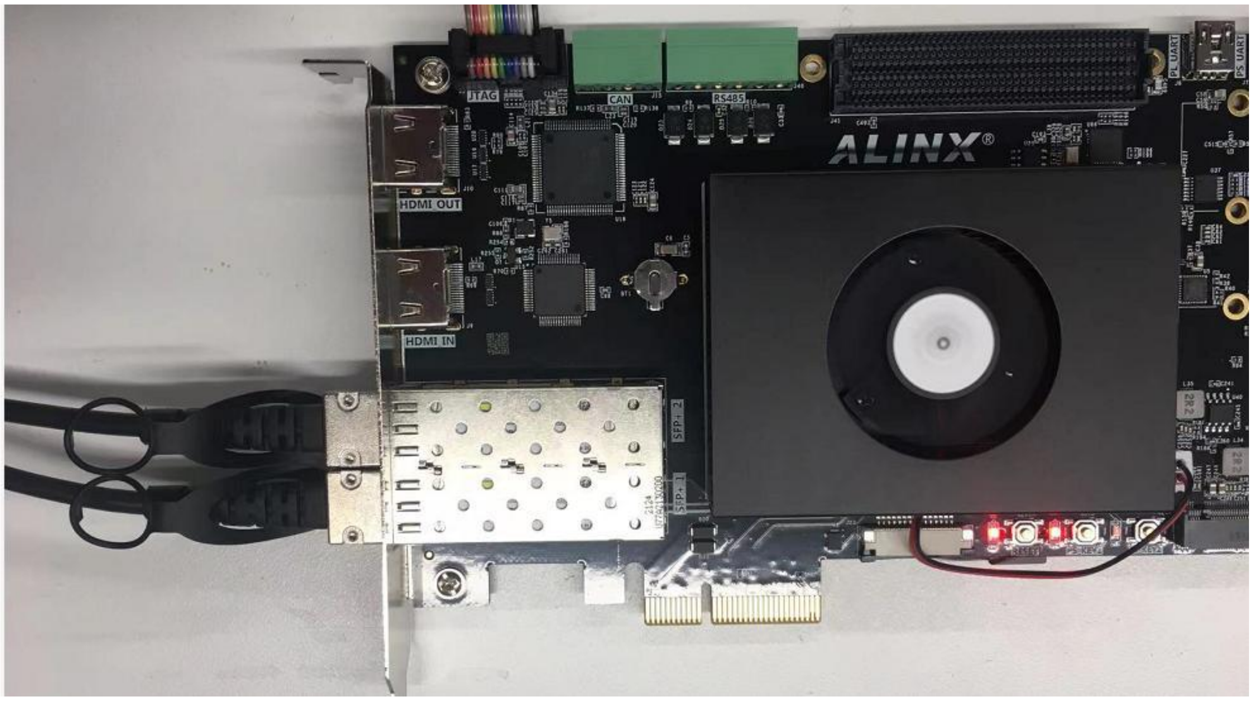
In addition, it is necessary to add the definition of the transmission control pin of the SFP optical module, such as



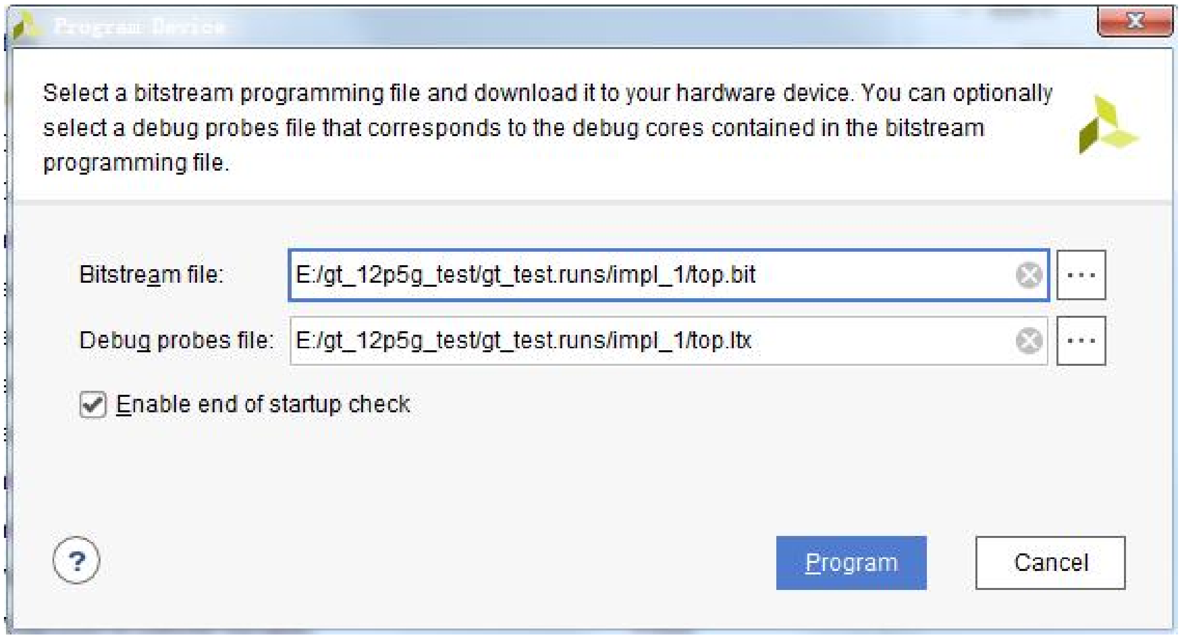
```
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]

set_property PACKAGE_PIN D12 [get_ports {tx_disable[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {tx_disable[0]}]
```

4 Optical Fiber Data Transmission Test

After the compilation project is passed, we can start the experiment of fiber data transmission. The optical module is plugged into the SFP1~SFP2 interface of the AXU4EV-P FPGA development board, and then connected to the optical fiber. The AXU4EV-P hardware is connected as shown below:



Then download the bit file to the FPGA.



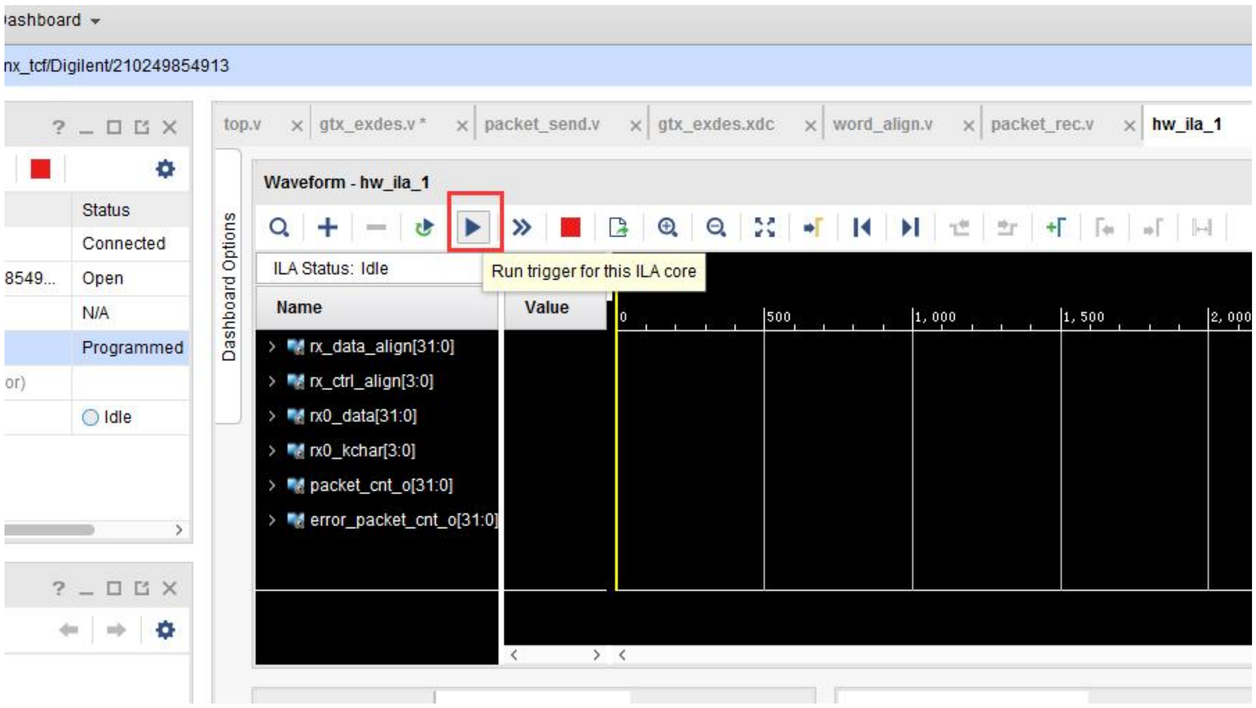
Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: E:/gt_12p5g_test/gt_test.runs/impl_1/top.bit

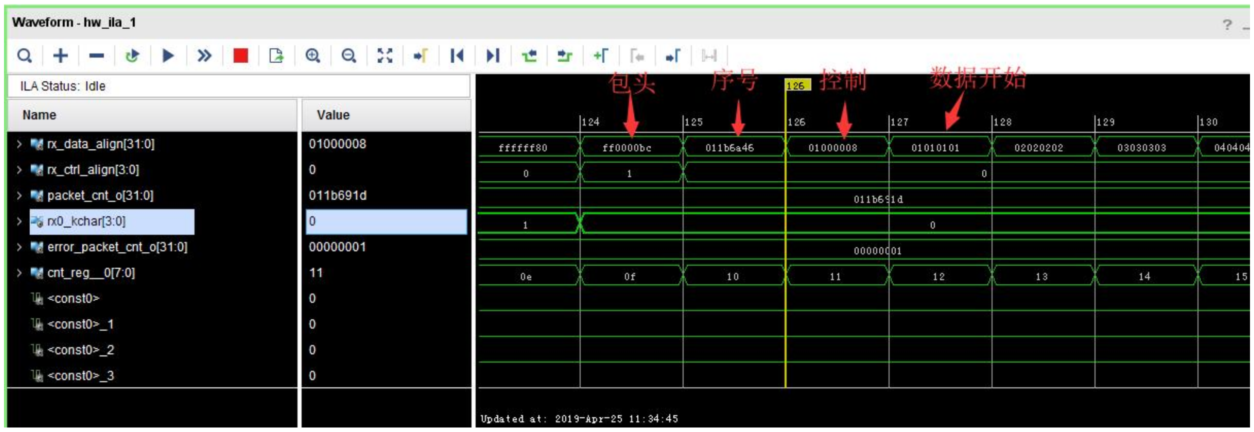
Debug probes file: E:/gt_12p5g_test/gt_test.runs/impl_1/top.ltx

☑ Enable end of startup check

Click the Run Trigger for this ILA core button



We can see the test data received in the ILA.



The error_packet_cnt_o value is incremented by 1 if there is a received packet error.