

GTX Fiber Optic Data Communication Experiment

Technical Email: alinx@aithtech.com

Sales Email: rachel.zhou@alinx.com

1 Experiment Introduction

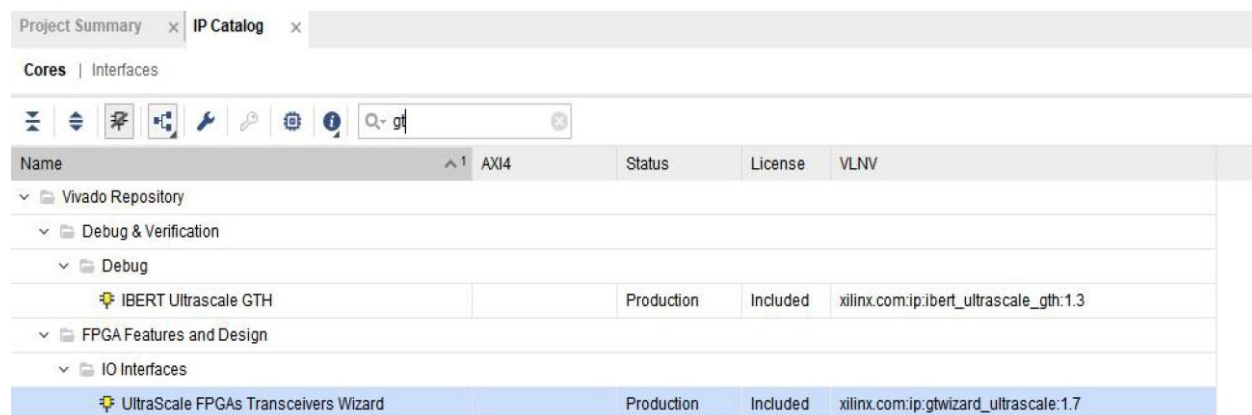
This experiment will introduce the transmission of data communication through optical fiber. The test data is generated by the FPGA itself and transmit by GTX to the first optical fiber port. Then, through the fiber loop to the second fiber port, the GTX receives data for verification.

2 Experimental principle

2.1 GTX IP Design

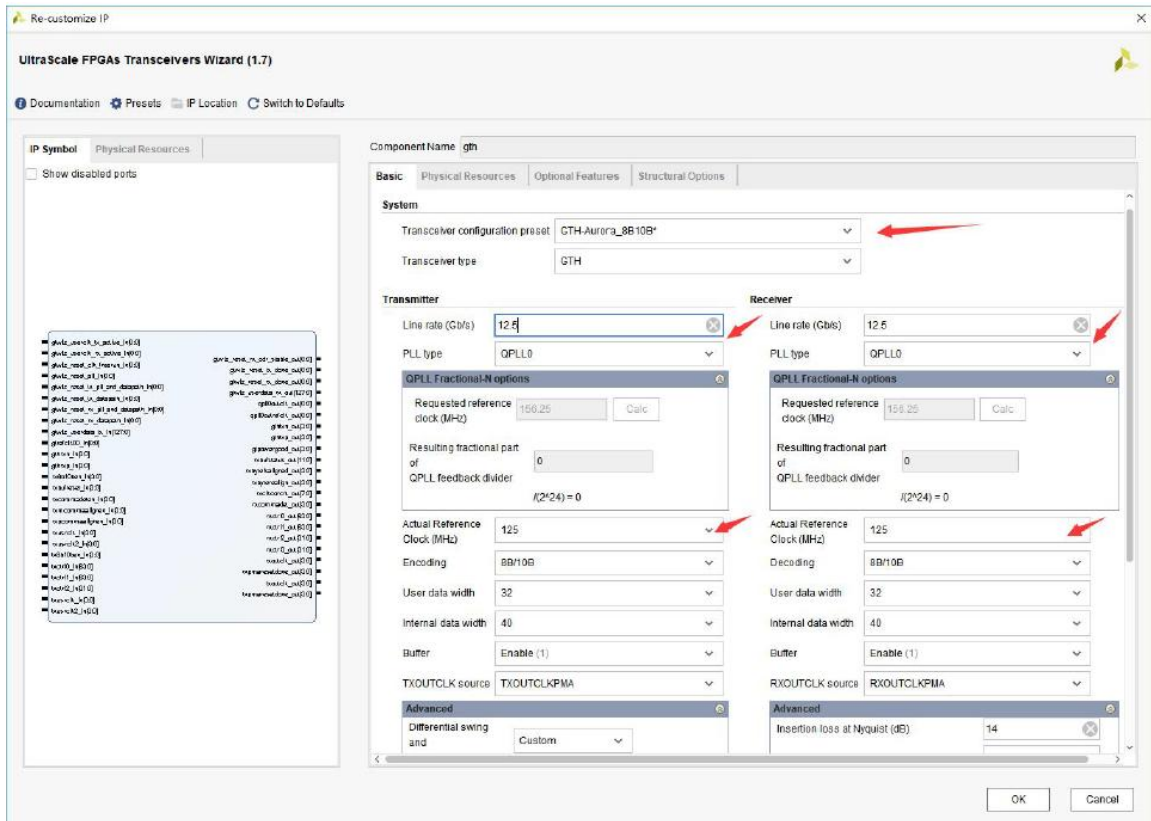
XILINX Vivado software has designed GTX IP for users, users can use IP to achieve high-speed data transmission and reception of GTX without paying attention to the internal work of GTX. The following is the specific GTX IP generation and configuration method.

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

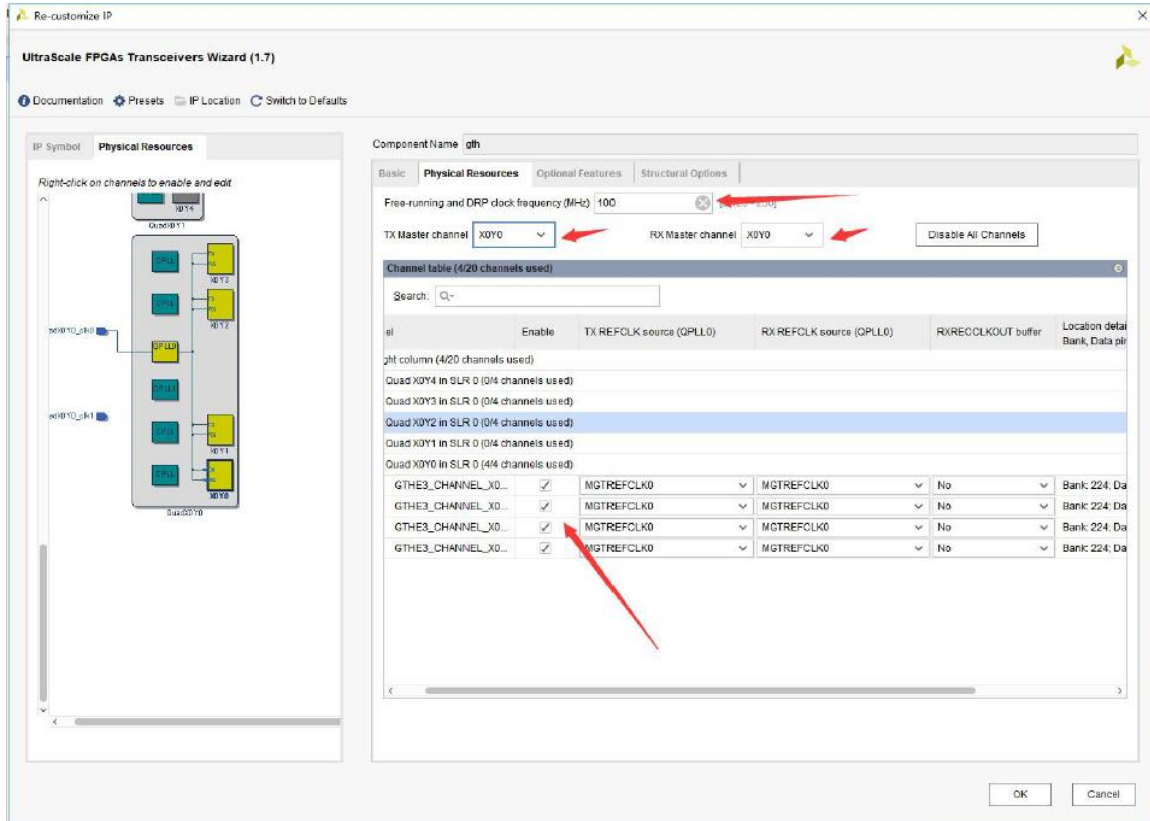


- 2) Enter "gtx" as the name in the "Component Name". First set the GTX transmission protocol bit "GTH-Aurora 8B10B single lane 4byte". As we mentioned in the previous chapter, Xilinx's GTX supports many protocols. The Aurora 8B/10B protocol is a scalable and lightweight link layer protocol can be used to transmit data point-to-point through one or more serial links. The optical module we use here is single-channel transmission, and the data interface is 4byte, which is 32-bit data. Then select the Line Rate speed of TX and RX. The Line Rate speed needs to be an integer multiple of the GTX reference clock. The GTP reference clock on the

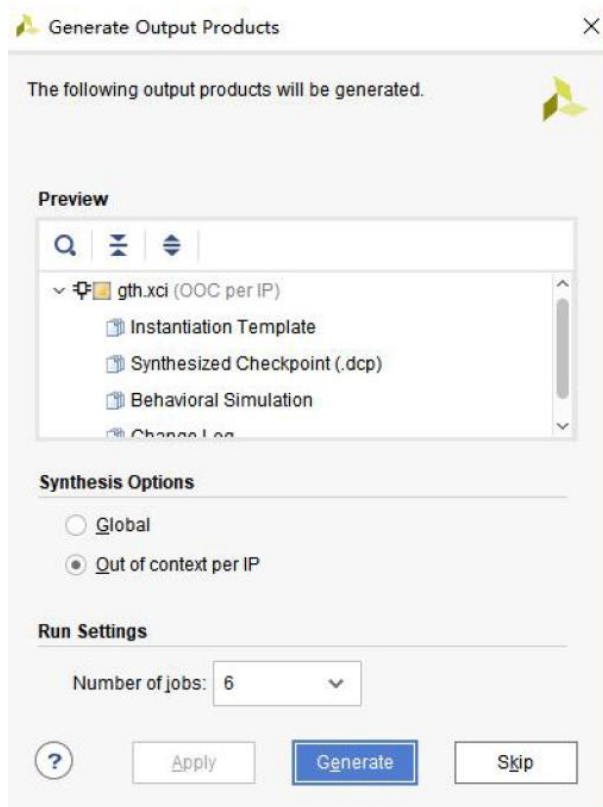
development board is 125Mhz. Here, our Line Rate is 10 times the reference clock, so the Line Rate is set to 1.25Gbps, if the user needs to set other rates such as 10Gbps, just modify it directly. Reference Clock is 125Mhz. PLL is set to QPLL and other defaults.



- 3) The 4 transceivers of the AXKU040 development board are all mounted on Bank224, the clock is 100M, and 4 channels are checked by default.



- 4) The other options are ok by default, generate ip. For more information about the configuration of GTX IP, please refer to the documents provided by Xilinx.

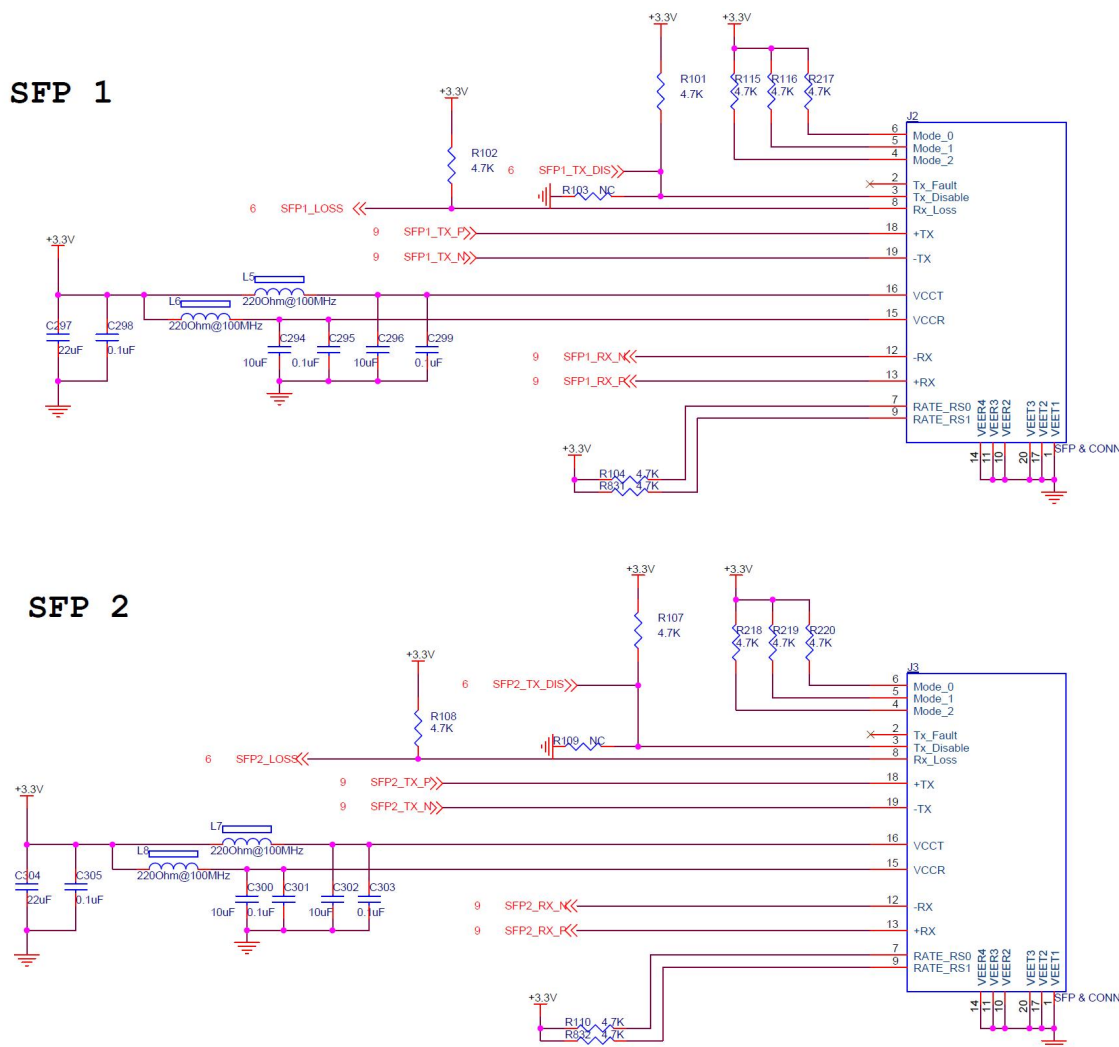


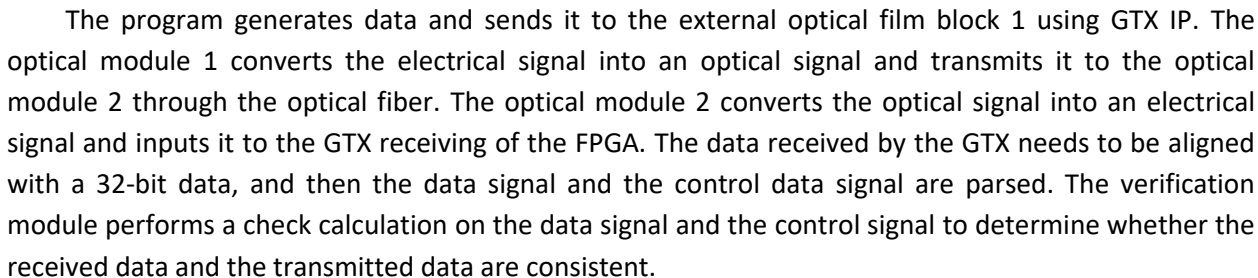
2.2 Hardware Introduction

On the AXKU040 FPGA development board, there are four fiber optic interfaces OPT1~OPT4 and one four-in-one QSFP fiber optic interface, which are respectively connected to the GTX channel of the FPGA chip.

The OPT1 optical module interface is connected to GTX Channel0, OPT2 is connected to GTX Channel1, OPT3 is connected to GTX Channel2, and OPT4 is connected to GTX Channel3. The QSFP optical module is connected to Channel0~Channel3 of the GTX of the BANK118. The optical module and the FPGA are separated by a 0.1uf capacitor, and the AC Couple mode is used.

The optical module LOSS signal and TX_Disable signal are connected to the normal IO of the FPGA. The LOSS signal is used to detect whether the optical reception of the optical module is lost. If no optical fiber or link is inserted, the LOSS signal is high, otherwise it is low. The TX_Disable signal is used to enable or disable the light emission of the optical module. If the TX_Disable signal is high, the light emission is turned off. Otherwise, the optical transmission is enabled. When normal use, the signal need to be pulled low. The hardware schematic is as follows:





Sources

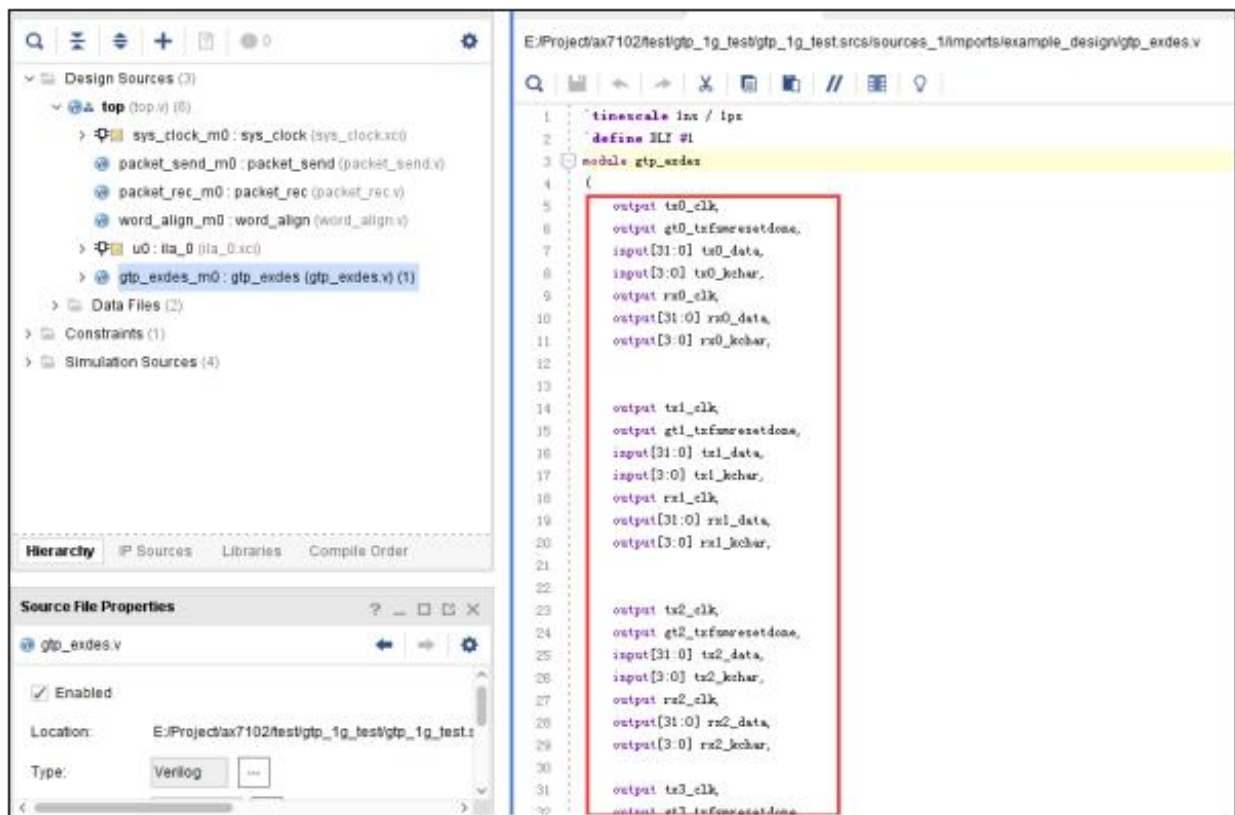
- Design Sources (4)
 - Verilog Header (1)
 - top (top.v) (6)
 - sys_clock_m0 : sys_clock (sys_clock.xci)
 - packet_send_m0 : packet_send (packet_send.v)
 - packet_rec_m0 : packet_rec (packet_rec.v)
 - word_align_m0 : word_align (word_align.v)
 - u0 : ila_0 (ila_0.xci)
 - gtx_exdes_m0 : gth_example_top (gth_example_top.v)
 - gth_example_checking_8b10b (gth_example_checking_8b10b.v)
 - gth_example_stimulus_8b10b (gth_example_stimulus_8b10b.v)
 - Constraints (1)

Hierarchy | IP Sources | Libraries | Compile Order

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

3.1 GTX data communication module

In the previous example we have generated the GTX IP example project, where the `gt0_frame_gen.v` module and the `gt0_frame_check.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, and then add the following four channels of user interface signals at the Port of the module. The signals of the four channels after the addition are as shown in the figure below



Then connect the signals of the four channels defined by the port with the signals in the “GTX_support” submodule:

```

650
651 assign tx0_clk = gt0_txusrclk2_i;
652 assign rx0_clk = gt0_rxusrclk2_i;
653 assign rx0_data = gt0_rxdata_i;
654 assign rx0_kchar = gt0_rxcharisk_i;
655 assign gt0_txfsmresetdone = gt0_txfsmresetdone_i;
656
657
658 assign tx1_clk = gt1_txusrclk2_i;
659 assign rx1_clk = gt1_rxusrclk2_i;
660 assign rx1_data = gt1_rxdata_i;
661 assign rx1_kchar = gt1_rxcharisk_i;
662 assign gt1_txfsmresetdone = gt1_txfsmresetdone_i;
663
664 assign tx2_clk = gt2_txusrclk2_i;
665 assign rx2_clk = gt2_rxusrclk2_i;
666 assign rx2_data = gt2_rxdata_i;
667 assign rx2_kchar = gt2_rxcharisk_i;
668 assign gt2_txfsmresetdone = gt2_txfsmresetdone_i;
669
670 assign tx3_clk = gt3_txusrclk2_i;
671 assign rx3_clk = gt3_rxusrclk2_i;

```

The following is a description of several GTX user interface signals added in the GTX_exdes.v port. The following takes the channel0 GTX interface as an example:

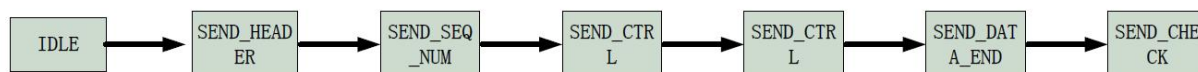
Signal Name	Bits	Input/Output	Description
tx0_clk	1	Output	The data transmission clock, which is the TXUSRCLK2 described in Section 14, has a GTP reference clock of 125Mhz. The data is valid on the rising edge.
tx0_data	32	Input	GTP Transmits data
tx0_kchar	4	Input	<p>The K control word transmitted by GTP 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.</p> <p>Tx0_kchar [3] corresponds to tx0_data [31:24]</p> <p>Tx0_kchar [2] corresponds to tx0_data [23:16]</p> <p>Tx0_kchar [1] corresponds to tx0_data [15:8]</p> <p>Tx0_kchar [0] corresponds to tx0_data [7:0]</p>
rx0_clk	1	Output	The data receive clock, which is the RXUSRCLK2 described in Section 14, has a GTX reference clock of 125Mhz. Data is valid on the rising edge
rx0_data	32	Output	GTP receives data.
rx0_kchar	4	Output	GTP receives the K control word to indicate whether the

			<p>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.</p> <p>Rx0_kchar [3] corresponds to rx0_data [31:24]</p> <p>Rx0_kchar [2] corresponds to rx0_data [23:16]</p> <p>Rx0_kchar [1] corresponds to rx0_data [15:8]</p> <p>Rx0_kchar [0] corresponds to rx0_data [7:0]</p>
gt0_txsmresetdone	1	Output	GTP initialization completion signal

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

3.2 GTX 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 GTX. The sending process is as follows:



The number of data bits transmitted by all GTXs is 32 bits, the sync signal header signal is defined as 32 bits of "ff_00_00_bc", and the lower 8 bits "bc" are K28.5 code control characters. The K code feature definition is described in Xilinx's "ug482_7Series_GTX_Transceivers.pdf" document.

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

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.

```

47
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;

```

3.3 Bit data alignment module `word_align.v`

The external user data interface of the GTX 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 GTX transmit data and receive data shifts:

GTX Transmitted Data		GTX 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 GTX 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 GTX 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.

```
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
```

3.4 GTX 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 GTX 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
```

3.5 Pin constraints

```

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
# -----

# Location constraints for differential reference clock buffer
# Note: the IP core-level XDC constrains the transceiver channel
# -----

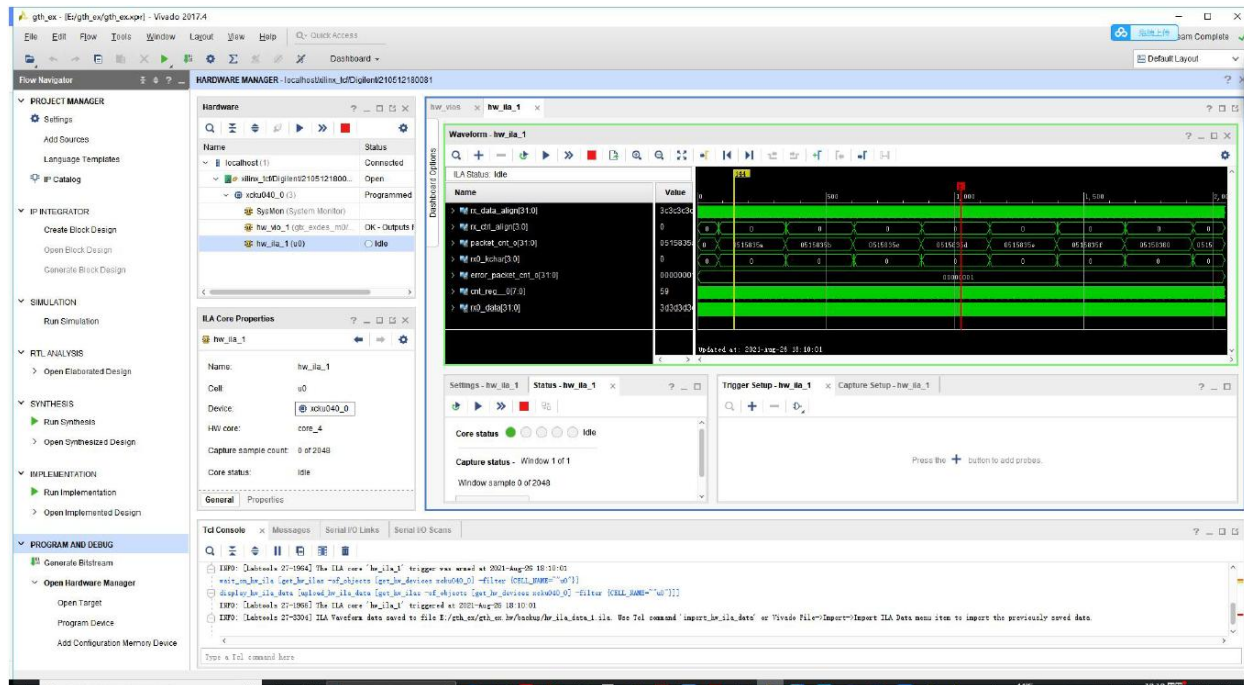
set_property PACKAGE_PIN AF5 [get_ports mgtrefclk_n]
set_property PACKAGE_PIN AF6 [get_ports mgtrefclk_p]

```

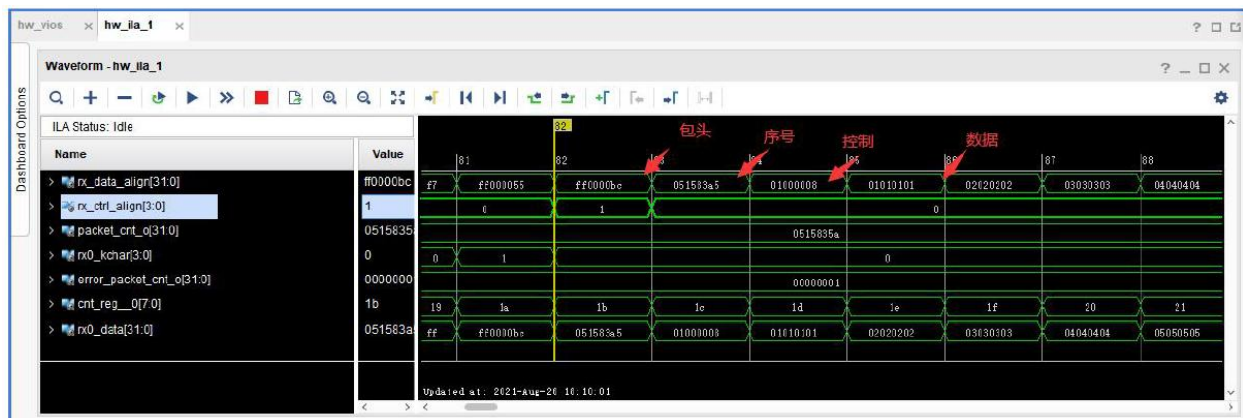
4 Optical Fiber Data Transmission Test

The screenshot displays the Xilinx Vivado Hardware Manager interface. The main window shows the 'Hardware' tab with a tree view of the device hierarchy. The selected device is 'xc7u040-0' (3), which is in a 'Programmed' state. The 'Hardware Device Properties' panel on the left provides details for 'xc7u040_0', including its name, part number, ID code (13822093), IR length (6), and status (Programmed). The 'Programming file' is set to 'Elgh_exigh_ex_runsimpl_1/top.bit'. The 'Waveform' window on the right shows a table of signals and their values, with a waveform viewer below it. The 'Settings' panel at the bottom left shows the 'Core status' as 'Idle' and the 'Capture status' as 'Window 1 of 1'. The 'Trigger Setup' and 'Capture Setup' panels are also visible at the bottom right.

Click “Refresh Device”








At this point, the ila interface will appear, and 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.

So far, the GTX fiber data transfer experiment has been introduced. If the user needs to reconfigure the “GTX IP” in the project, if you only modify the “GTX IP” configuration in the project, you will get an error when compiling. Users need to re-generate the GTX IP example file while modifying the GTX IP configuration in the project, and then replace the following files in the project with the files generated in the example project.

_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	