GTX Fiber Optic Communication Testing Experiment

Technical Email: alinx@aithtech.com Sales Email: rachel.zhou@alinx.com

1 Experiment Introduction

The Xilinx Kintex UltraScale family of FPGAs integrates a GTX serial high-speed transceiver for high-speed serial data communication. On the AXKU040 FPGA development board, the four transceiver channels of the 20 GTX of the FPGA are connected to the interface of the four SFP optical modules. Users only need to purchase the SPF optical module to realize the data transmission of the optical fiber. This lab will introduce data transmission and reception and eye diagram testing between optical modules through fiber optic connections.

2 Experimental principle

2.1 GTX Introduction

The AXKU040 FPGA development board FPGA chip (XCKU040_ffva1156) comes with 20 GTX serial high-speed transceiver channels, each with a transceiver speed up to 12.5 Gb/s. The GTX transceivers support different serial transmission interfaces or protocols, such as PCIE 1.1/2.0/3.0 interface, 10G network XUAI interface, OC-48, serial Rapid IO interface, SATA (Serial ATA) interface, Serial Digital Interface (SDI) and etc.

Xilinx groups serial high-speed transceivers in Quad, four serial high-speed transceivers and a COMMOM (QPLL) to form a Quad, each serial high-speed transceiver called a Channel.

The specific internal logic block diagram of GTX is shown below. It consists of four transceiver channels GTXE2_CHANNEL and one GTXE2_COMMON. Each GTXE2_CHANNEL contains the transmit circuit TX and the receive circuit RX.

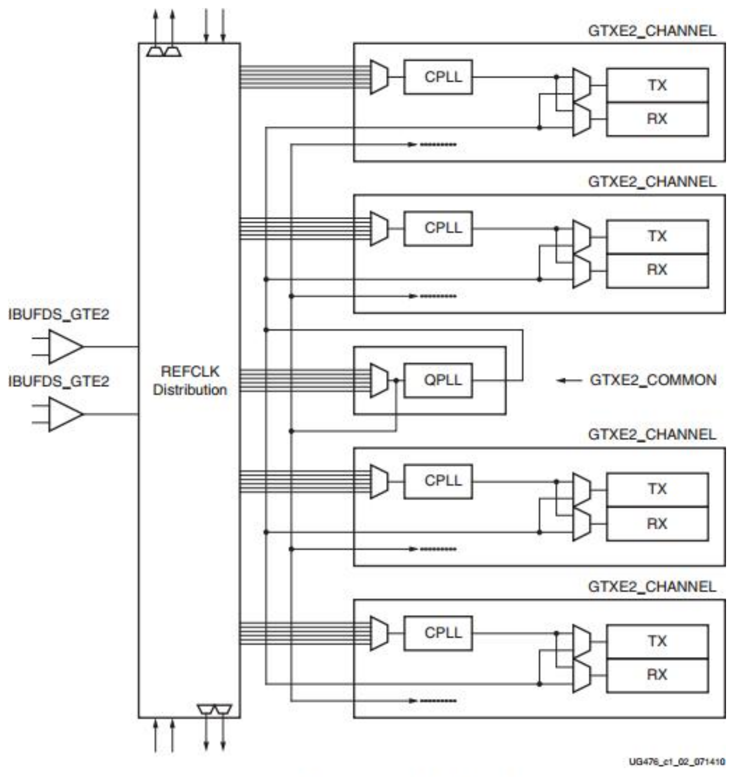
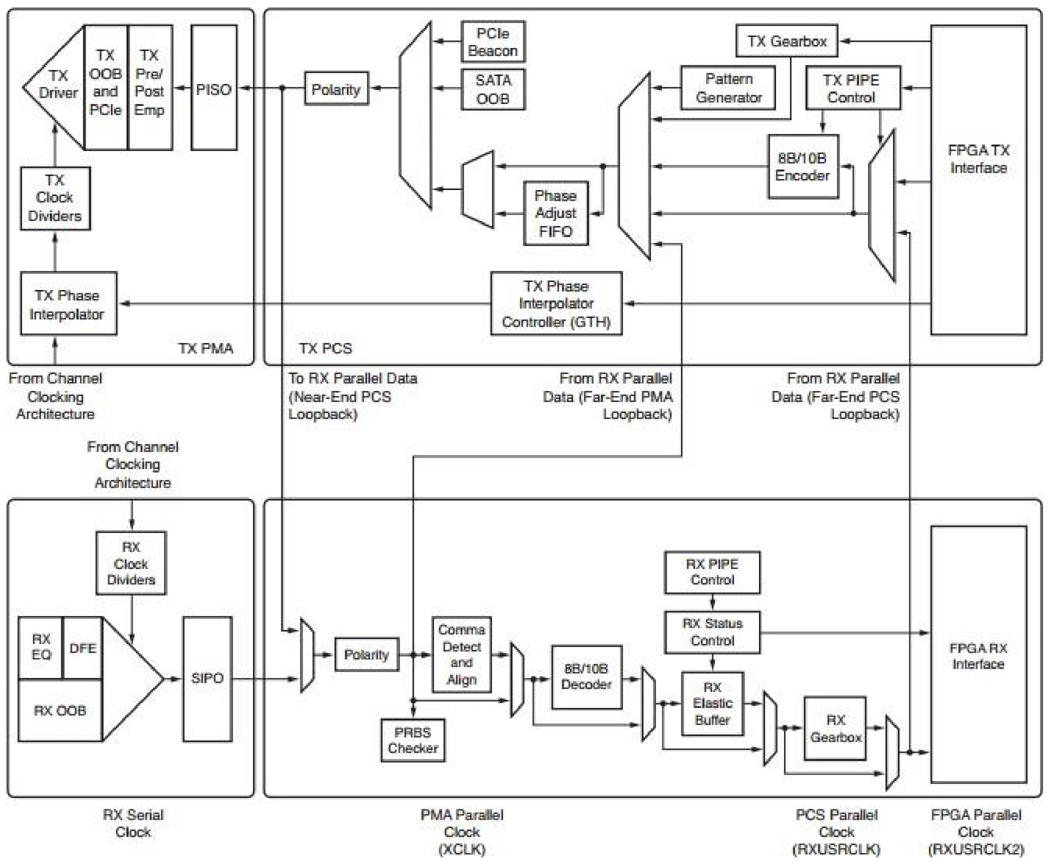


Figure 1-2: **GTX Transceiver Quad Configuration**

The logic of each GTXE2_CHANNEL is shown below:



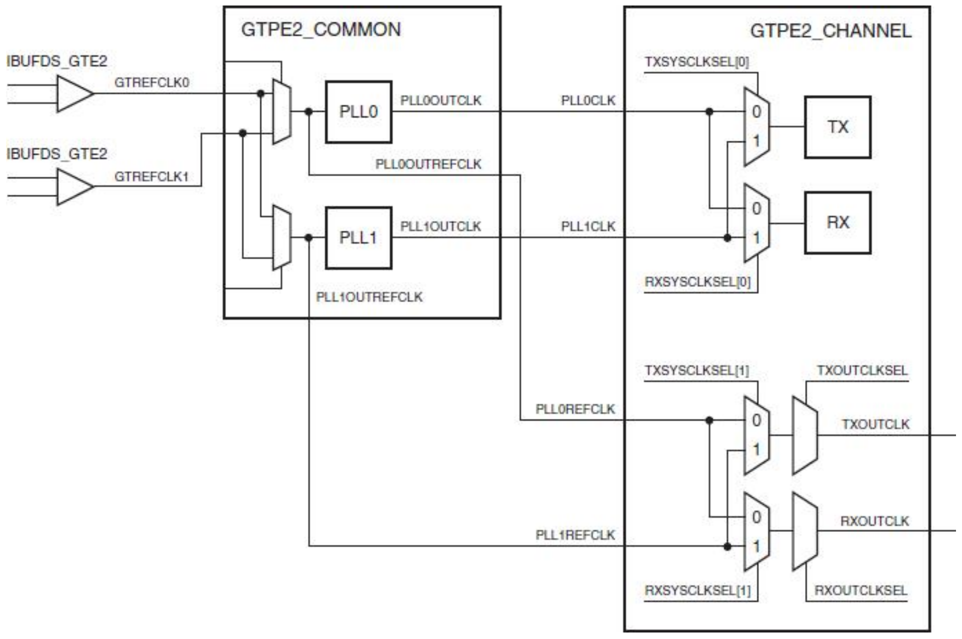
The function of the transceiver and receiver of GTXE2_CHANNEL is independent. It consists of two sublayers: PMA (Physical Media Attachment) and PCS (Physical Coding Sublayer). The PMA sublayer contains circuits such as high-speed serial-to-parallel conversion (Serdes), pre/post emphasis, receive equalization, clock generator, and clock recovery. The PCS sublayer contains 8B/10B codec, buffer, channel bonding and clock correction circuits

2.1.1 GTX transmit and receive processing flow:

First, after the user logic data is encoded by 8B/10B, it enters a transmit buffer (Phase Adjust FIFO). The buffer is mainly isolated from the clock domain of the PMA sublayer and the PCS sublayer to solve the clock rate matching and phase. The problem of difference is finally subjected to parallel-to-serial conversion (PISO) by high-speed Serdes, and if necessary, pre-emphasis (TX Pre-emphasis) and post-emphasis can be performed. It is worth mentioning that if the TXP and TXN differential pins are accidentally cross-connected during PCB design, this design error can be compensated for by Polarity. The receiving end and the transmitting end process are reversed, and there are many similarities. I will not go into details here. It is necessary to pay attention to the elastic buffer of the RX receiving end, which has clock correction and channel binding.

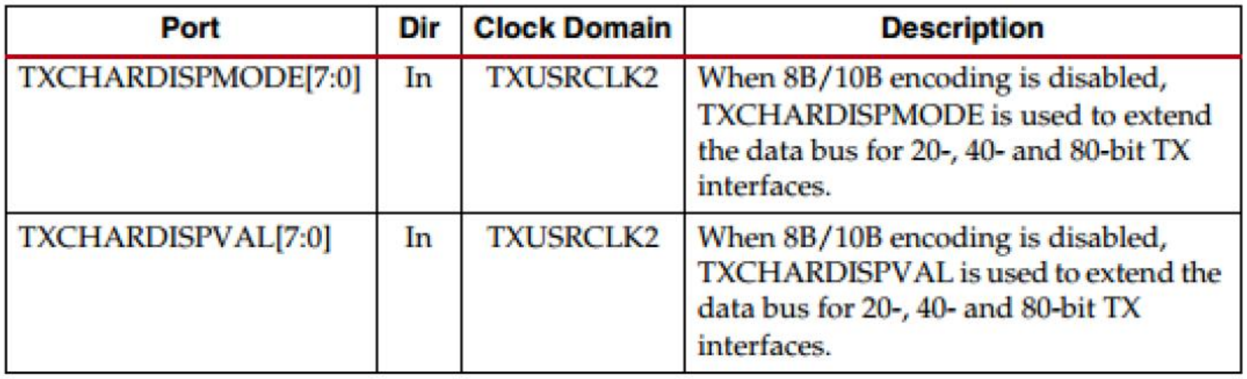
2.1.2 GTX reference clock

The GTX module has four differential reference clock input pins (MGTREFCLK0P/N and MGTREFCLK1P/N) that serve as the reference clock source for the GTX module and are user-selectable. On the core board of the AX7325, a 125Mhz GTX reference clock is connected to MGTREFCLK0P/N as the reference clock for GTX. The differential reference clock is converted to a single-ended clock signal by the IBUFDS module into PLL0 and PLL1 of GTXE2_COMMOM, producing the desired clock frequency in the TX and RX circuits. If the TX and RX transceivers are at the same speed, the TX and RX circuits can use the same PLL-generated clock. If the TX and RX transceivers are not the same speed, a different PLL clock-generated clock is required.

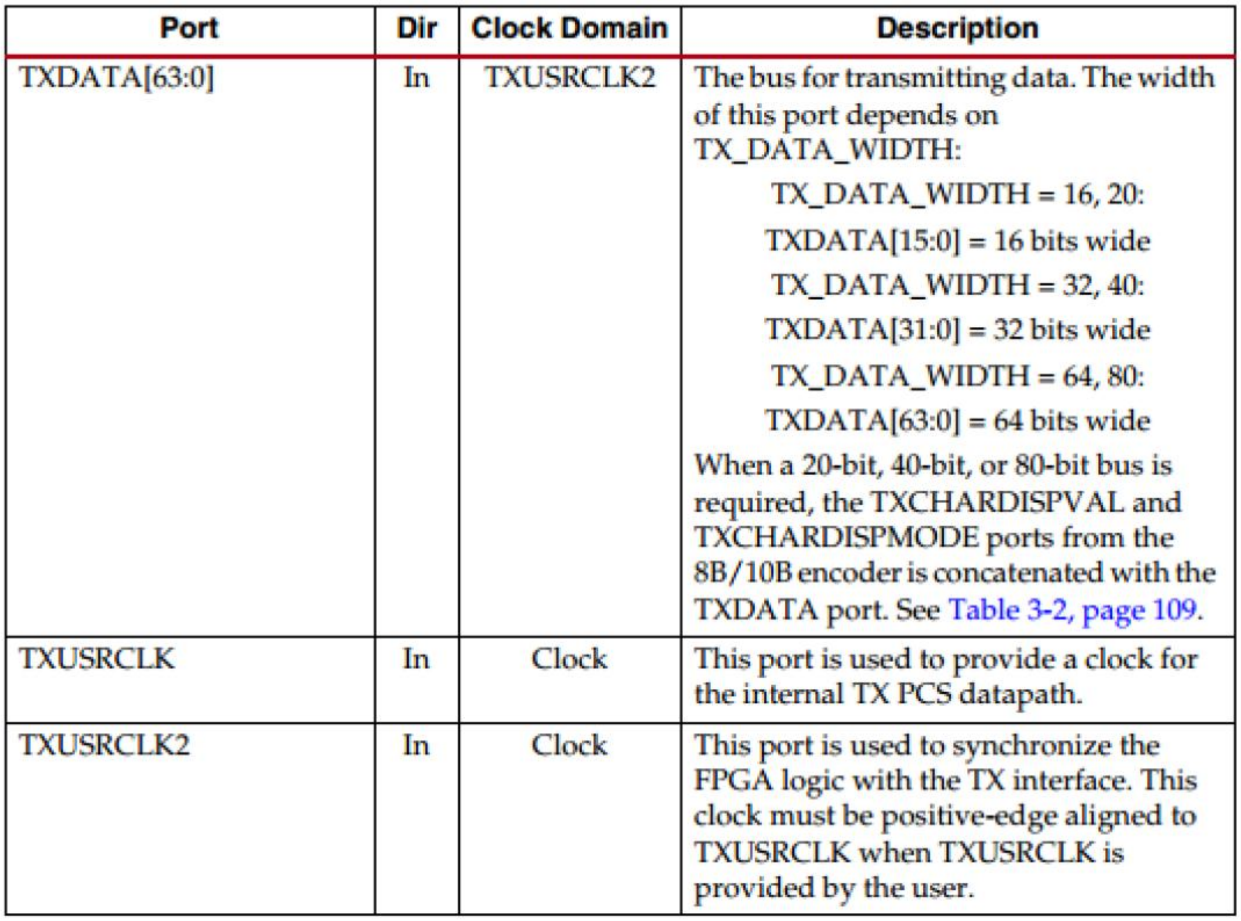


2.1.3 GTX FPGA TX interface signal

The TX interface signal is the interface signal sent by the user data of the FPGA to the GTX. The name and description of the interface signal are shown in the following table:



Port	Dir	Clock Domain	Description
TXCHARDISPMODE[7:0]	In	TXUSRCLK2	When 8B/10B encoding is disabled, TXCHARDISPMODE is used to extend the data bus for 20-, 40- and 80-bit TX interfaces.
TXCHARDISPVAL[7:0]	In	TXUSRCLK2	When 8B/10B encoding is disabled, TXCHARDISPVAL is used to extend the data bus for 20-, 40- and 80-bit TX interfaces.



Port	Dir	Clock Domain	Description
TXDATA[63:0]	In	TXUSRCLK2	The bus for transmitting data. The width of this port depends on TX_DATA_WIDTH: TX_DATA_WIDTH = 16, 20: TXDATA[15:0] = 16 bits wide TX_DATA_WIDTH = 32, 40: TXDATA[31:0] = 32 bits wide TX_DATA_WIDTH = 64, 80: TXDATA[63:0] = 64 bits wide When a 20-bit, 40-bit, or 80-bit bus is required, the TXCHARDISPVAL and TXCHARDISPMODE ports from the 8B/10B encoder is concatenated with the TXDATA port. See Table 3-2, page 109.
TXUSRCLK	In	Clock	This port is used to provide a clock for the internal TX PCS datapath.
TXUSRCLK2	In	Clock	This port is used to synchronize the FPGA logic with the TX interface. This clock must be positive-edge aligned to TXUSRCLK when TXUSRCLK is provided by the user.

The functions of TXCHARDISPMODE[7:0] and TXCHARFDISPVAL[7:0] are determined by whether TX8B10BEN is enabled. When 8B/10B is enabled, these signals are used as TX data signals.

TXDATA[63:0] transmits the data interface signal. The data width is determined by the TX_DATA_WIDTH parameter. When TX_DATA_WIDTH is 16, or 20, the TXDATA data width is 16; when TX_DATA_WIDTH is 32, or 40, the TXDATA data width is 32. The TX8B10BEN and TX_DATA_WIDTH parameter settings can be configured to different FPGA interface data bit widths and GTX internal data widths, as shown in the following table:

TX8B10BEN	TX_DATA_WIDTH	TX_INT_DATAWIDTH	FPGA Interface Width	Internal Data Width
0	16	0	16	16
	20	0	20	20
	32	0	32	16
	32	1	32	32
	40	0	40	20
	40	1	40	40
	64	1	64	32
	80	1	80	40

When TX8B10BEN is not enabled, the 20-bit, 40-bit, and 80-bit data signals are combined by the TXCHARDISPMODE[7:0], TXCHARFDISPVAL[7:0] signals, and TXDATA.

	< < < Data Transmission Order is Right to Left (LSB to MSB) < < <											
	39	38	37 36 35 34 33 32 31 30	29	28	27 26 25 24 23 22 21 20	19	18	17 16 15 14 13 12 11 10	9	8	7 6 5 4 3 2 1 0
Data Transmitted	TXCHARDISPMODE[3]	TXCHARDISPVAL[3]	TXDATA[31:24]	TXCHARDISPMODE[2]	TXCHARDISPVAL[2]	TXDATA[23:16]	TXCHARDISPMODE[1]	TXCHARDISPVAL[1]	TXDATA[15:8]	TXCHARDISPMODE[0]	TXCHARDISPVAL[0]	TXDATA[7:0]

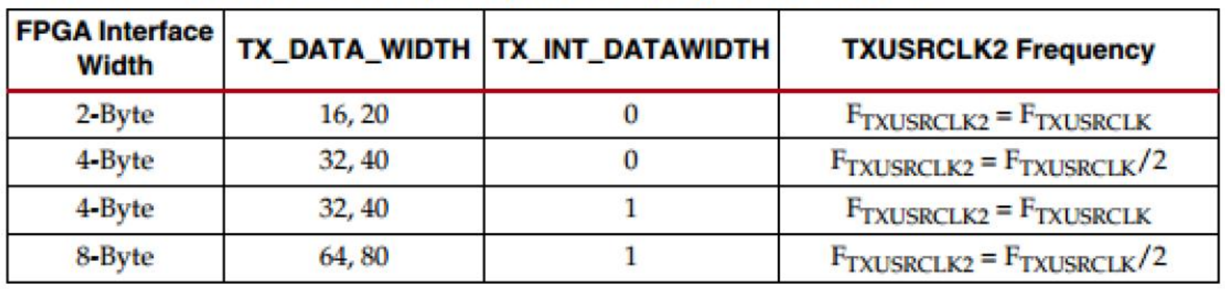
	< < < Data Transmission Order is Right to Left (LSB to MSB) < < <											
	79	78	77 76 75 74 73 72 71 70	69	68	67 66 65 64 63 62 61 60	59	58	57 56 55 54 53 52 51 50	49	48	47 46 45 44 43 42 41 40
Data Transmitted	TXCHARDISPMODE[7]	TXCHARDISPVAL[7]	TXDATA[63:56]	TXCHARDISPMODE[6]	TXCHARDISPVAL[6]	TXDATA[55:48]	TXCHARDISPMODE[5]	TXCHARDISPVAL[5]	TXDATA[47:40]	TXCHARDISPMODE[4]	TXCHARDISPVAL[4]	TXDATA[39:30]

The sampling clock for all FPGA interface signals is TXUSRCLK2, which samples TXDATA on the rising edge of TXUSRCLK2. TXUSRCLK is the PCS logic and data transmission provided to the GTX module. The clock frequency of TXUSRCLK is determined by the GTX serial transmission speed and the internal data width. The formula for the calculation is as follows:

$$TXUSRCLK\ Rate = \frac{Line\ Rate}{Internal\ Datapath\ Width}$$

The frequency of TXUSRCLK2 is related to TXUSRCLK, which is determined by the frequency of TXUSRCLK and the value of TX_DATA_WIDTH. The clock frequency of TXUSRCLK2 is calculated as shown in the following table:

Table 3-3: **TXUSRCLK2 Frequency Relationship to TXUSRCLK**



FPGA Interface Width	TX_DATA_WIDTH	TX_INT_DATAWIDTH	TXUSRCLK2 Frequency
2-Byte	16, 20	0	$F_{TXUSRCLK2} = F_{TXUSRCLK}$
4-Byte	32, 40	0	$F_{TXUSRCLK2} = F_{TXUSRCLK}/2$
4-Byte	32, 40	1	$F_{TXUSRCLK2} = F_{TXUSRCLK}$
8-Byte	64, 80	1	$F_{TXUSRCLK2} = F_{TXUSRCLK}/2$

The following two criteria should be followed for the two clocks, TXUSRCLK2 and TXUSRCLK:

1) TXUSRCLK and TXUSRCLK2 must be aligned on the rising edge. The smaller the deviation, the better. Therefore, BUFGs or BUFRs should be used to drive these two clocks.

2) Even if the reference clocks of TXUSRCLK, TXUSRCLK2, and GTX operate at different clock frequencies, it must be ensured that the same clock must be used. Therefore, the TXUSRCLK and TXUSRCLK2 clock frequencies must be multiplied or divided by the GTX reference clock.

2.1.4 GTX FPGA RX interface signal

The RX interface signal is the interface signal transmits by the GTX module to the FPGA user data. The name and description of the interface signal are shown in the following table:

Port	Dir	Clock Domain	Description
RXDISPERR[7:0]	Out	RXUSRCLK2	When 8B/10B decoding is disabled, RXDISPERR is used to extend the data bus for 20-bit, 40-bit and 80-bit RX interfaces.
RXCHARISK[7:0]	Out	RXUSRCLK2	When 8B/10B decoding is disabled, RXCHARISK is used to extend the data bus for 20-bit, 40-bit and 80-bit RX interfaces.
RXDATA[63:0]	Out	RXUSRCLK2	The bus for transmitting data. The width of this port depends on RX_DATA_WIDTH: RX_DATA_WIDTH = 16, 20: RXDATA[15:0] = 16 bits wide RX_DATA_WIDTH = 32, 40: RXDATA[31:0] = 32 bits wide RX_DATA_WIDTH = 64, 80: RXDATA[63:0] = 64 bits wide When a 20-bit, 40-bit, or 80-bit bus is required, the RXCHARISK and RXDISPERR ports from the 8B/10B encoder are concatenated with the RXDATA port. See Table 4-52, page 296.
RXUSRCLK	In	Clock	This port is used to provide a clock for the internal RX PCS datapath.
RXUSRCLK2	In	Clock	This port is used to synchronize the FPGA logic with the RX interface. This clock must be positive-edge aligned to RXUSRCLK when RXUSRCLK is provided by the user.

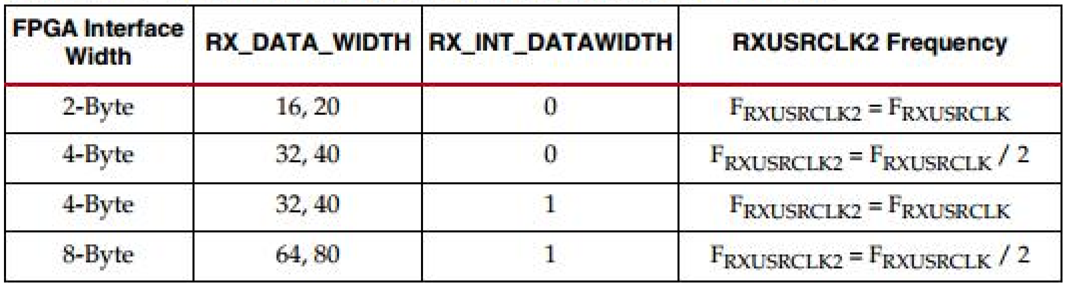
The RX interface signal is basically similar to the TX interface signal. The user port can be set to the data interface width of the FPGA and the GTX internal parallel data width according to different RX8B10BEN and RX_DATA_WIDTH parameters as shown in the following table:

Table 4-51: **FPGA RX Interface Datapath Configuration**

RX8B10BEN	RX_DATA_WIDTH	RX_INT_DATAWIDTH	FPGA Interface Width	Internal Data Width
1	20	0	16	20
	40	0	32	20
	40	1	32	40
	80	1	64	40
0	16	0	16	16
	20	0	20	20
	32	0	32	16
	32	1	32	32
	40	0	40	20
	40	1	40	40
	64	1	64	32
	80	1	80	40

Similarly, the frequency of RXUSRCLK and the frequency relationship of RXUSERCLK2 are also related to RX_DATA_WIDTH.

Table 4-53: **RXUSRCLK2 Frequency Relationship to RXUSRCLK**



FPGA Interface Width	RX_DATA_WIDTH	RX_INT_DATAWIDTH	RXUSRCLK2 Frequency
2-Byte	16, 20	0	$F_{RXUSRCLK2} = F_{RXUSRCLK}$
4-Byte	32, 40	0	$F_{RXUSRCLK2} = F_{RXUSRCLK} / 2$
4-Byte	32, 40	1	$F_{RXUSRCLK2} = F_{RXUSRCLK}$
8-Byte	64, 80	1	$F_{RXUSRCLK2} = F_{RXUSRCLK} / 2$

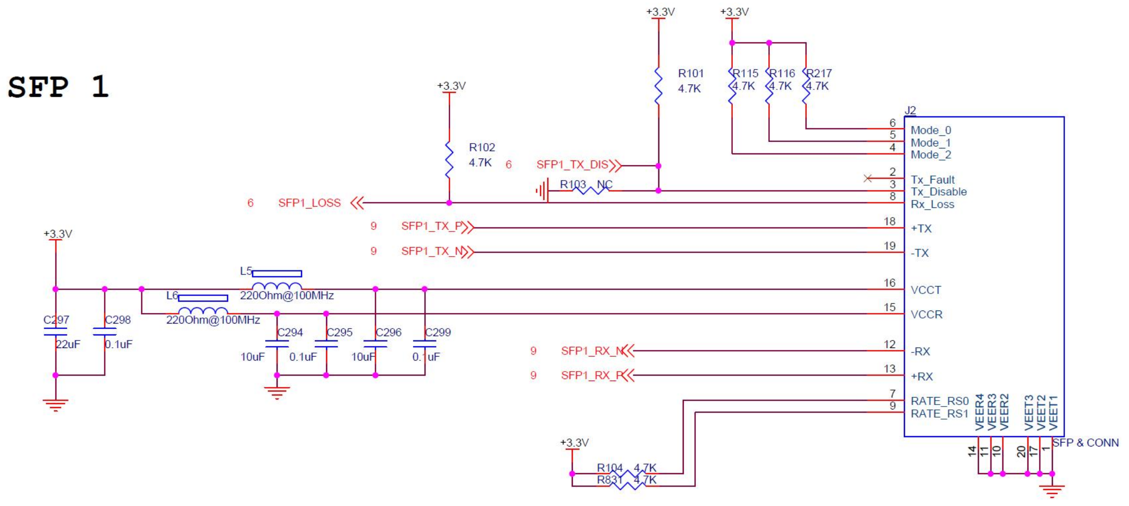
The knowledge about GTX is here. For more information on GTX hardware and applications, please refer to the document " ug476_7Series_Transceivers.pdf " provided by Xilinx.

2.2 Hardware Introduction

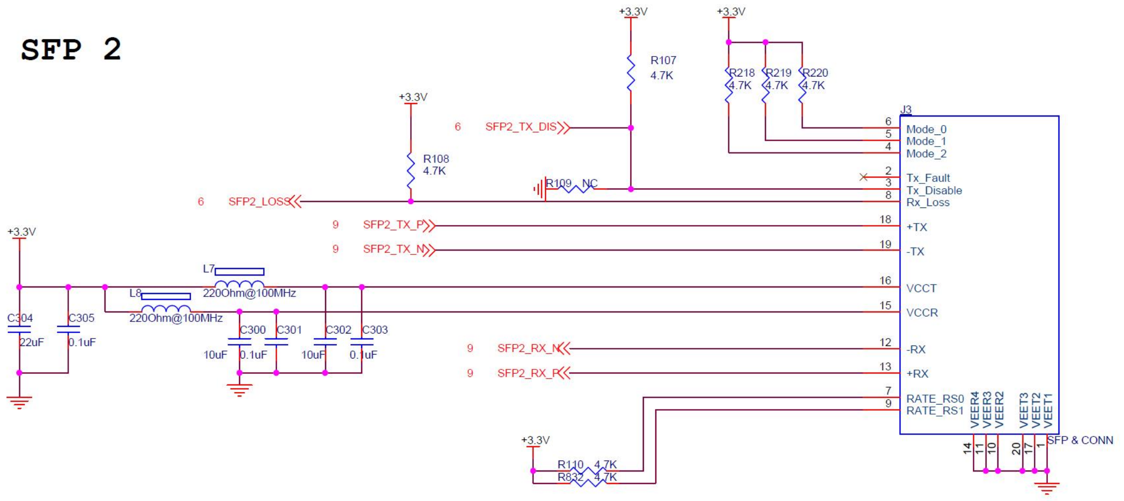
On the AXKU040 FPGA development board, there are four optical interfaces OPT1~OPT4, which are respectively connected to the GTX channels of the FPGA chip.

The OPT1 optical module interface is connected to GTX Channel0, OPT2 is connected to GTX Channel1, and the optical module and FPGA are separated by 0.1uf capacitor, using AC Couple mode.

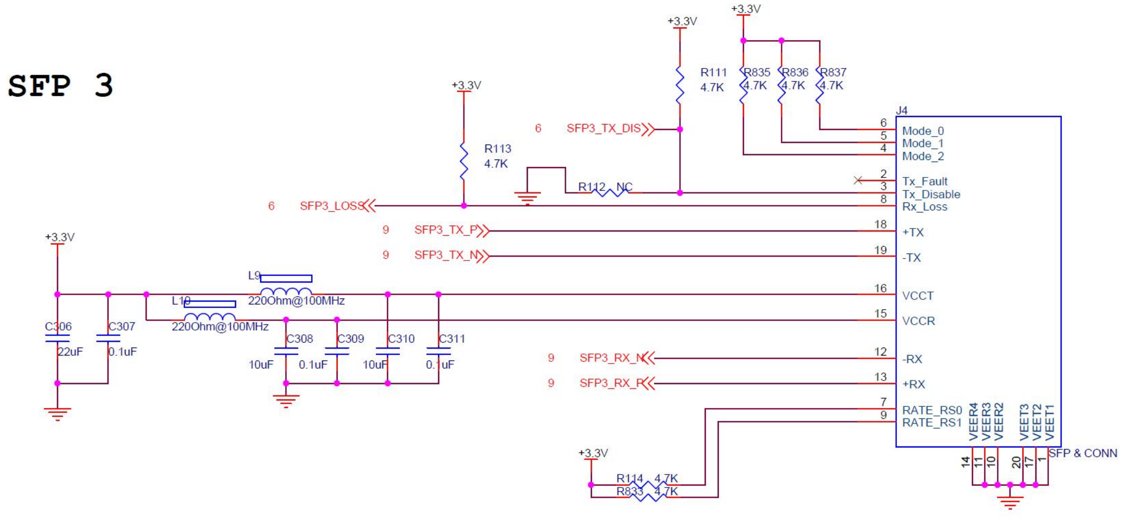
The optical module's LOSS signal and TX_Disable signal are connected to the normal IO of the FPGA. The LOSS signal is used to detect whether the optical reception of the optical module is lost. If no optical fiber or link is inserted, the LOSS signal is high, otherwise it is low. The TX_Disable signal is used to enable or disable the 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. This signal needs to be pulled down during normal use. The hardware schematic is as follows:



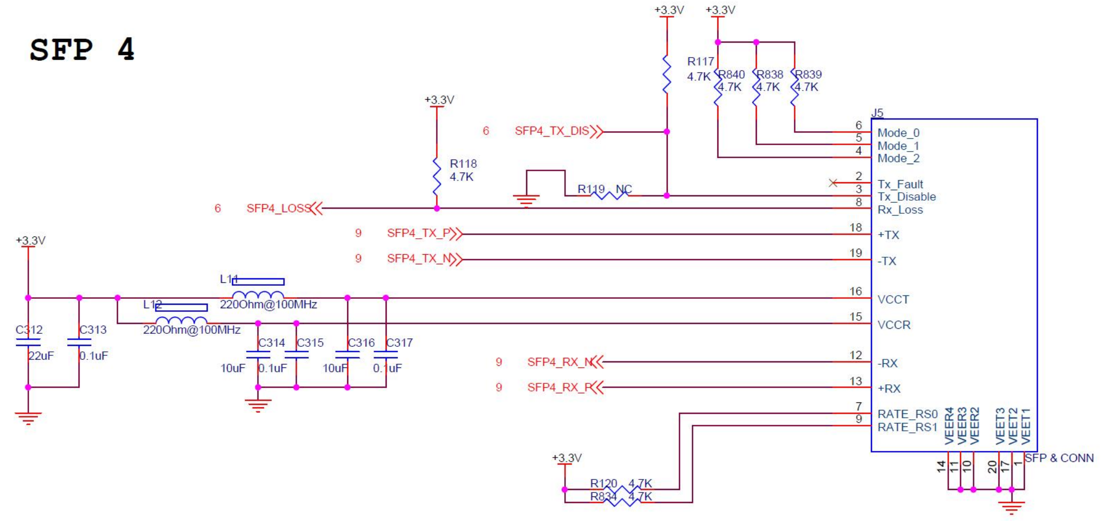
SFP 2



SFP 3



SFP 4



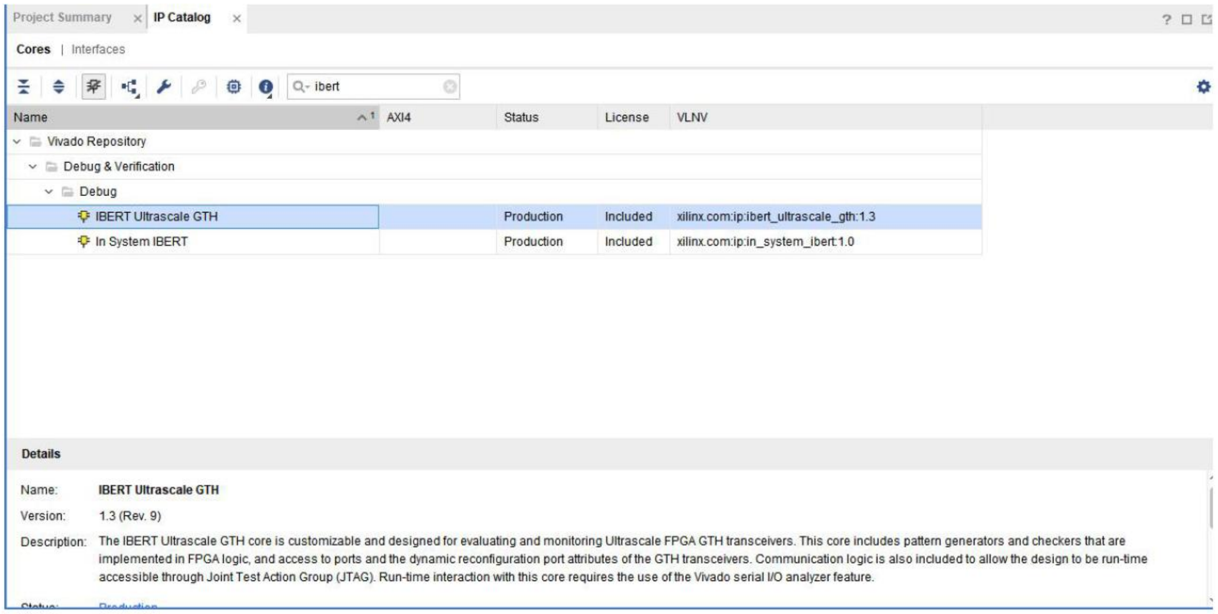
3 Programming

Before using GTX, let's test whether the GTX module on the development board is working properly. The following are the specific test steps for GTX data communication:

3.1 4-channel SFP fiber optic module communication test procedure

Before using GTX, let's test if the GTX module on the development board is working properly. The following are specific test steps for GTX data communication:

1) Create a new project, search for ibert in the IP Catalog



Name	AXI4	Status	License	VLNV
Vivado Repository				
Debug & Verification				
Debug				
IBERT Ultrascale GTH		Production	Included	xilinx.com:ip:ibert_ultrascale_gth:1.3
In System IBERT		Production	Included	xilinx.com:ip:in_system_ibert:1.0

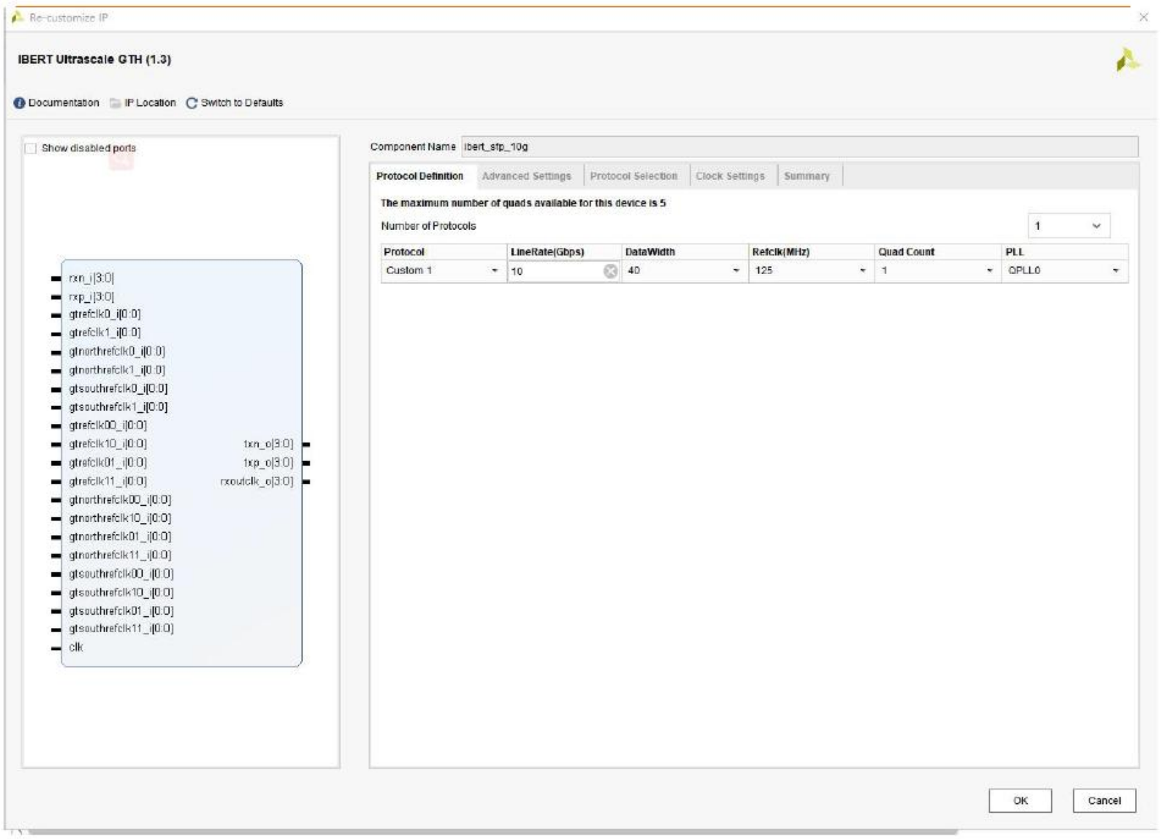
Details

Name: IBERT Ultrascale GTH

Version: 1.3 (Rev. 9)

Description: The IBERT Ultrascale GTH core is customizable and designed for evaluating and monitoring Ultrascale FPGA GTH transceivers. This core includes pattern generators and checkers that are implemented in FPGA logic, and access to ports and the dynamic reconfiguration port attributes of the GTH transceivers. Communication logic is also included to allow the design to be run-time accessible through Joint Test Action Group (JTAG). Run-time interaction with this core requires the use of the Vivado serial I/O analyzer feature.

2) Enter the speed of “LineRate” on the Protocol Definition page. If you are using a 10G optical module, you can select the highest rate of 10Gbps, the reference clock is 125Mhz, and the “LineRate” frequency is 80 times the integer of the reference clock.



IBERT Ultrascale GTH (1.3)

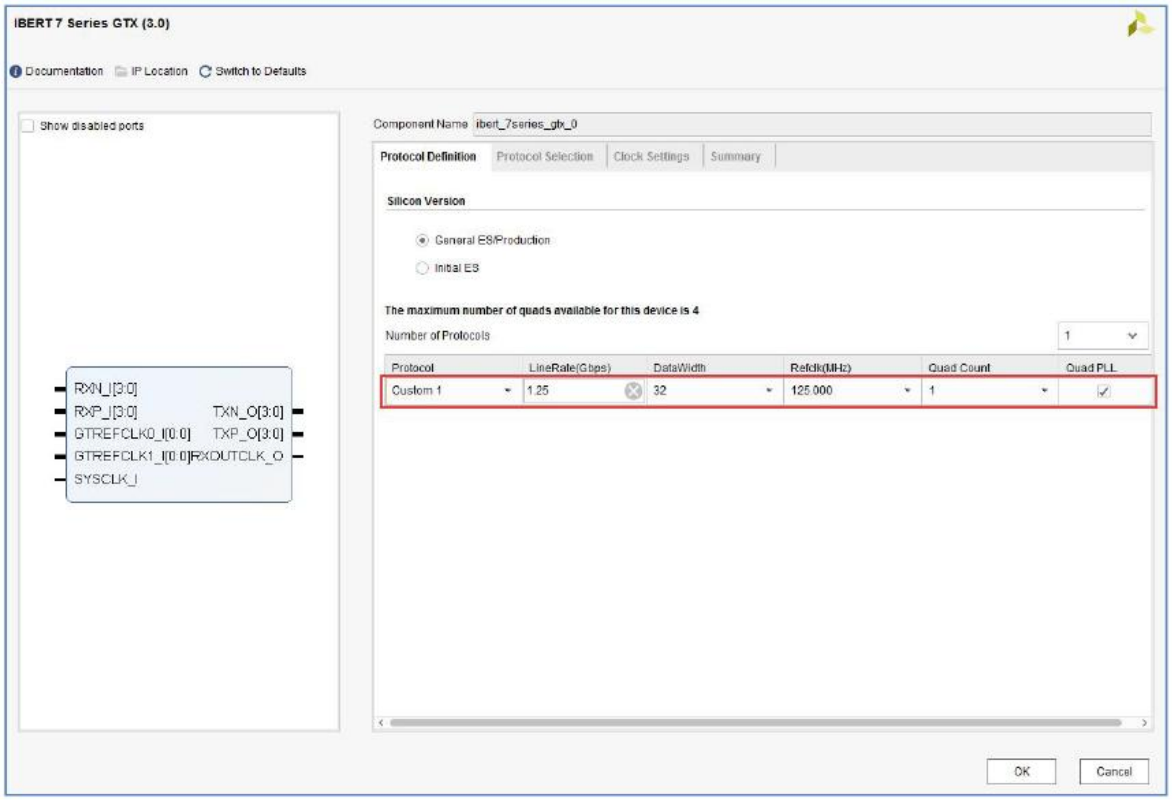
Component Name ibert_sfp_10g

The maximum number of quads available for this device is 5

Number of Protocols 1

Protocol	LineRate(Gbps)	DataWidth	Refclk(MHz)	Quad Count	PLL
Custom 1	10	40	125	1	QPLL0

If the user is using a 1.25G optical module, the speed of the LineRate is 1.25Gbps, and the frequency of the LineRate is 10 times the integer of the reference clock.



IBERT 7 Series GTX (3.0)

Component Name ibert_7series_gtx_0

Silicon Version

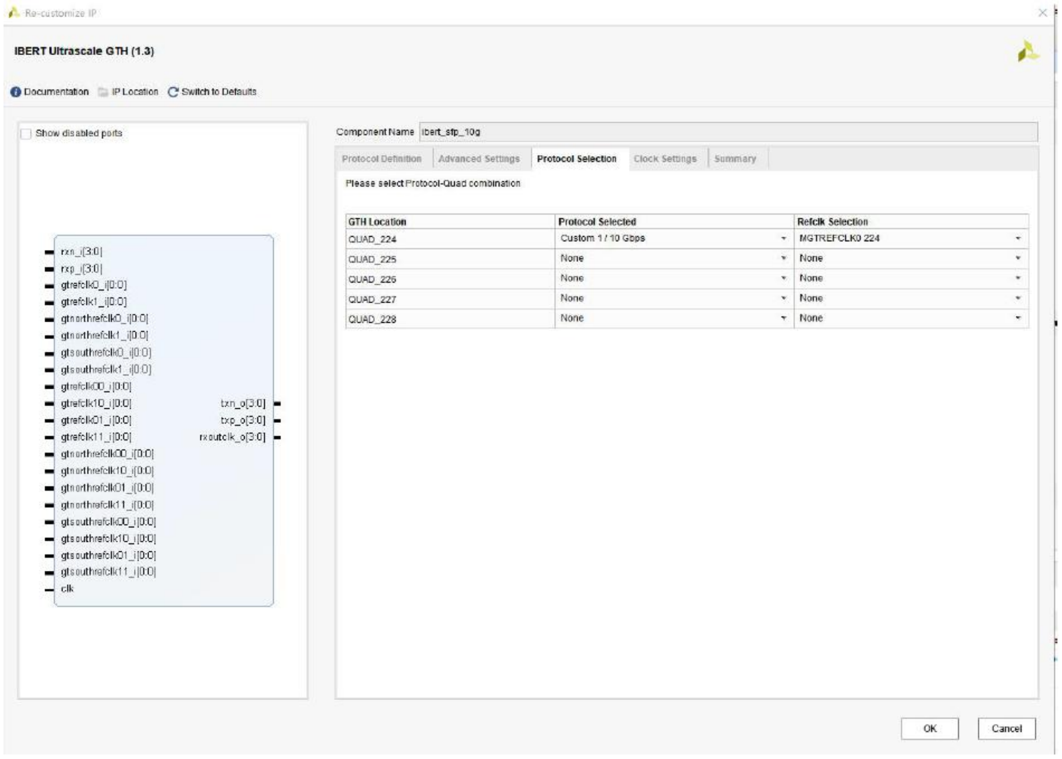
(●) General ES/Production
( ) Initial ES

The maximum number of quads available for this device is 4

Number of Protocols 1

Protocol	LineRate(Gbps)	DataWidth	Refclk(MHz)	Quad Count	Quad PLL
Custom 1	1.25	32	125.000	1	✓

3) In the “Protocol Selection” interface, select the “Protocol Selection” item as “Custom 1/10 Gbps” (“Custom 1/1.25Gbps” for 1.25G optical modules).

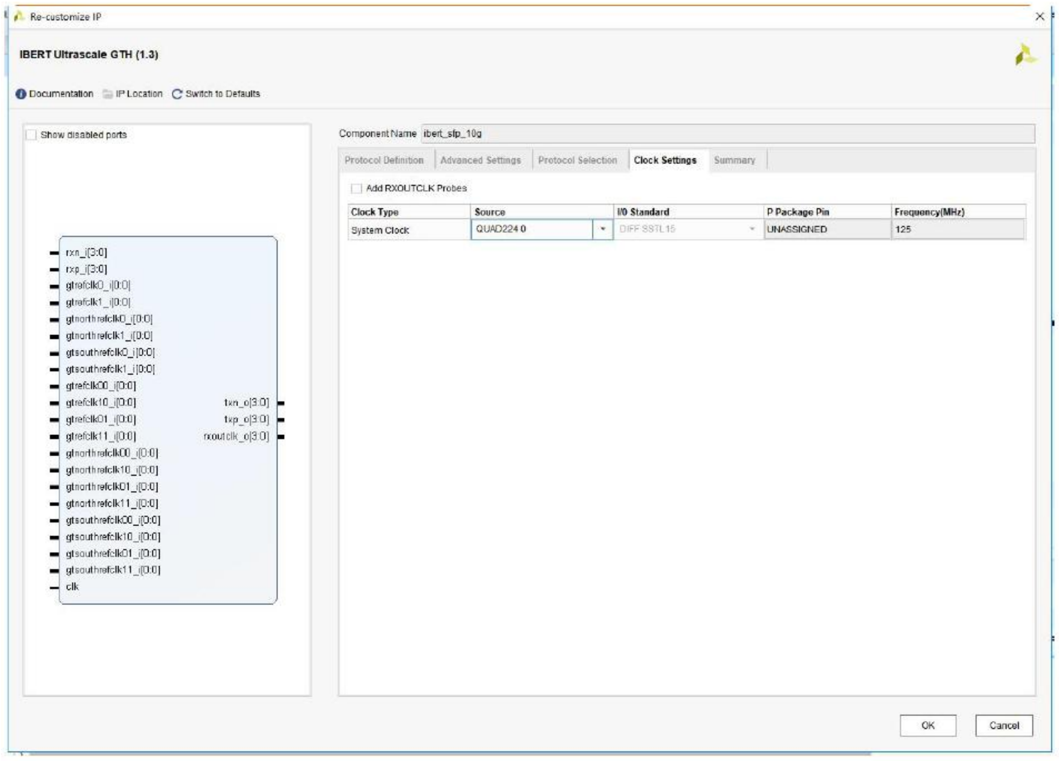


Component Name ibert_sfp_10g

Please select Protocol-Quad combination

GTH Location	Protocol Selected	Refclk Selection
QUAD_224	Custom 1 / 10 Gbps	MGTREFCLK0 224
QUAD_225	None	None
QUAD_226	None	None
QUAD_227	None	None
QUAD_228	None	None

4) In the “Clock Settings” screen, select the pin of the system clock. The pin settings here need to be consistent with the FPGA development board.

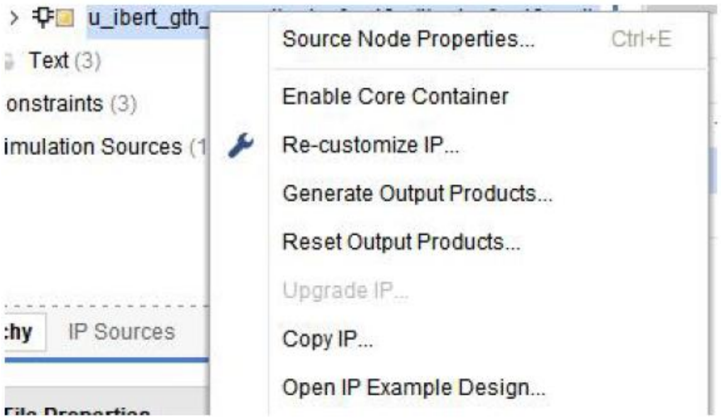


Component Name ibert_sfp_10g

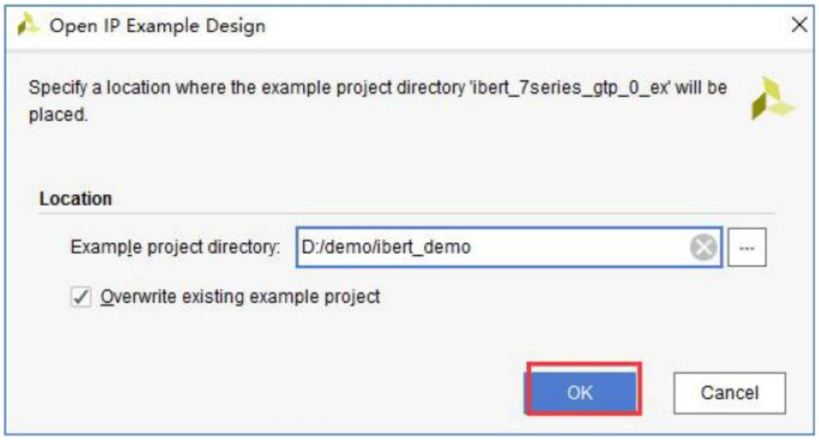
Add RXOUTCLK Probes

Clock Type	Source	I/O Standard	P Package Pin	Frequency(MHz)
System Clock	QUAD224 0	DIFF SSTL15	UNASSIGNED	125

5) Right-click and select ibert IP, and select Open IP Example Design in the pop-up drop-down box.



6) Then select the directory where Example project is placed.



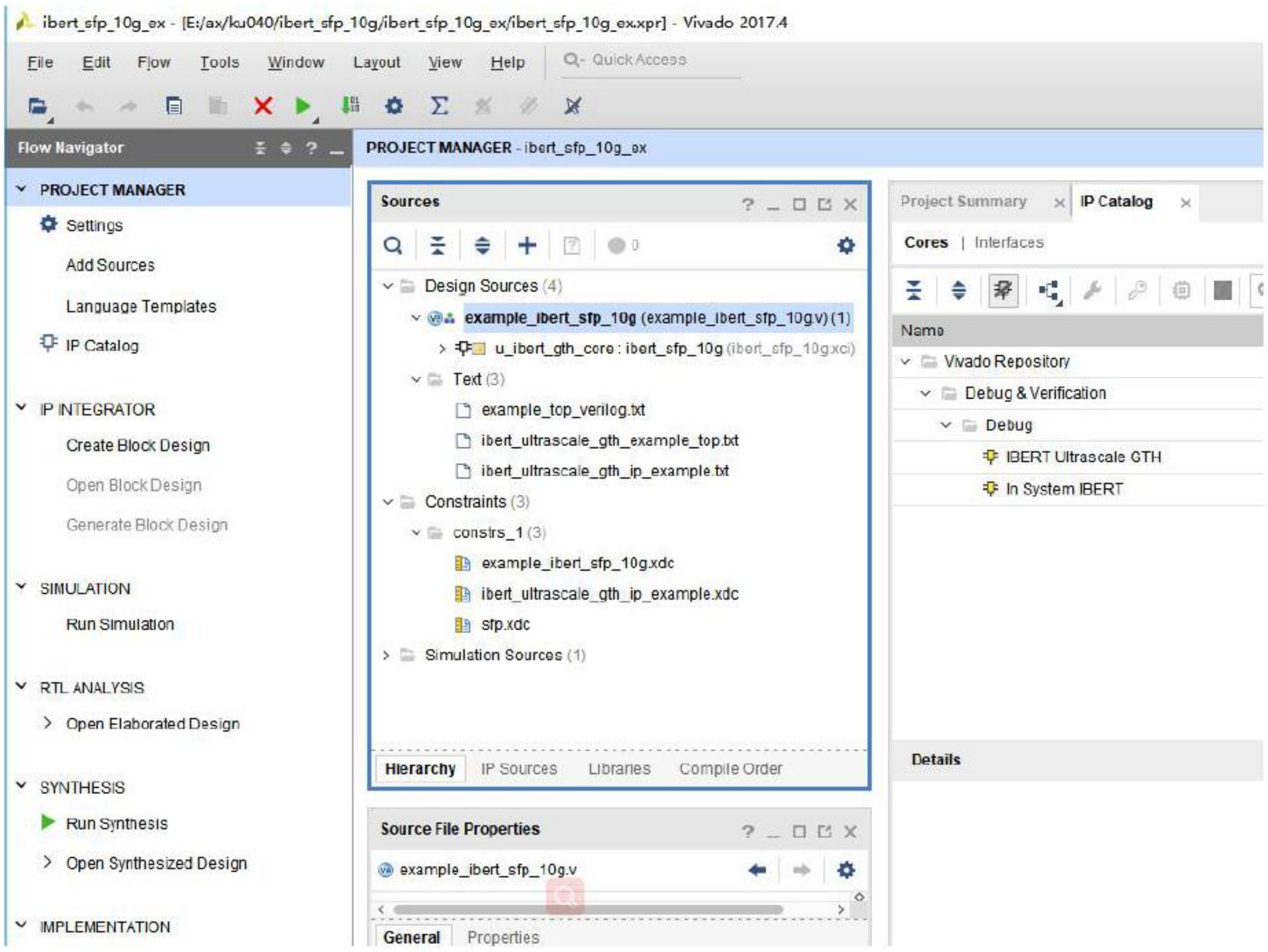
Specify a location where the example project directory 'ibert_7series_gtp_0_ex' will be placed.

Location

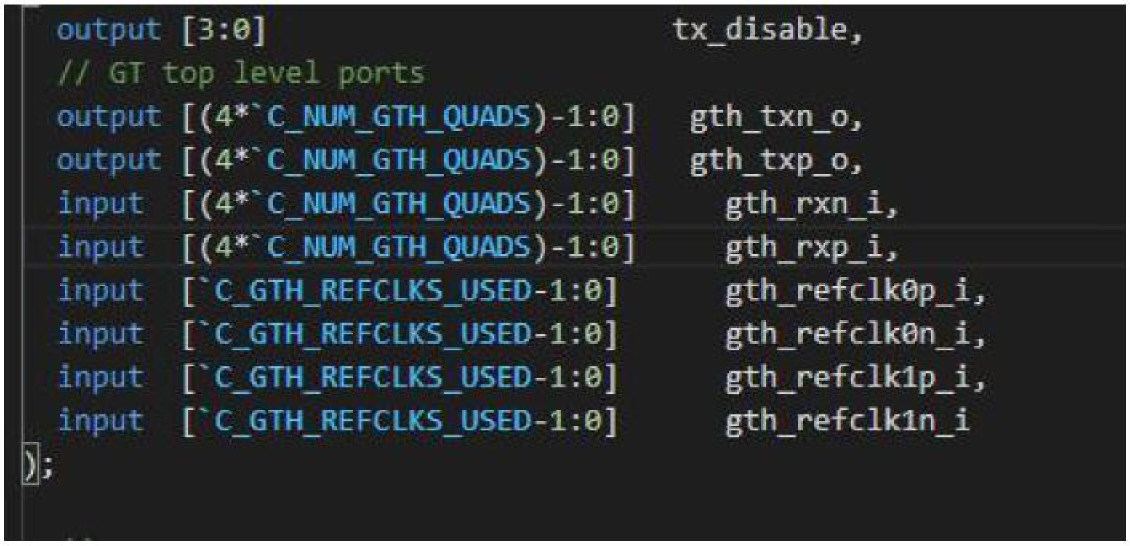
Example project directory: D:/demo/ibert_demo

✓ Overwrite existing example project

7) The software automatically generates a new project of example_ibert, in which the verilog program and pin constraint file have been configured.



8) In the AXKU040 hardware circuit, use the tx_disable signal to enable/disable the transmission of the SFP optical module. Open example_ibert. The file name here is related to the ip name created earlier. In the project, you need to define 4 tx_disable signals in the TOP program and assign them to 0. Always enable the transmission of the SFP optical module


```
    output [3:0]                      tx_disable,
    // GT top level ports
    output [(4*`C_NUM_GTH_QUADS)-1:0]   gth_txn_o,
    output [(4*`C_NUM_GTH_QUADS)-1:0]   gth_txp_o,
    input  [(4*`C_NUM_GTH_QUADS)-1:0]     gth_rxn_i,
    input  [(4*`C_NUM_GTH_QUADS)-1:0]     gth_rxp_i,
    input  [`C_GTH_REFCLKS_USED-1:0]      gth_refclk0p_i,
    input  [`C_GTH_REFCLKS_USED-1:0]      gth_refclk0n_i,
    input  [`C_GTH_REFCLKS_USED-1:0]      gth_refclk1p_i,
    input  [`C_GTH_REFCLKS_USED-1:0]      gth_refclk1n_i
);
```

9) Add the following pin constraints in the xdc file

```
set_property IOSTANDARD LVCMOS18 [get_ports {disable1[0]}]
set_property PACKAGE_PIN AM10 [get_ports  {disable1[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {disable1[1]}]
set_property PACKAGE_PIN AL10 [get_ports  {disable1[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {disable1[2]}]
set_property PACKAGE_PIN AP9 [get_ports  {disable1[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {disable1[3]}]
set_property PACKAGE_PIN AN9 [get_ports  {disable1[3]}]
```

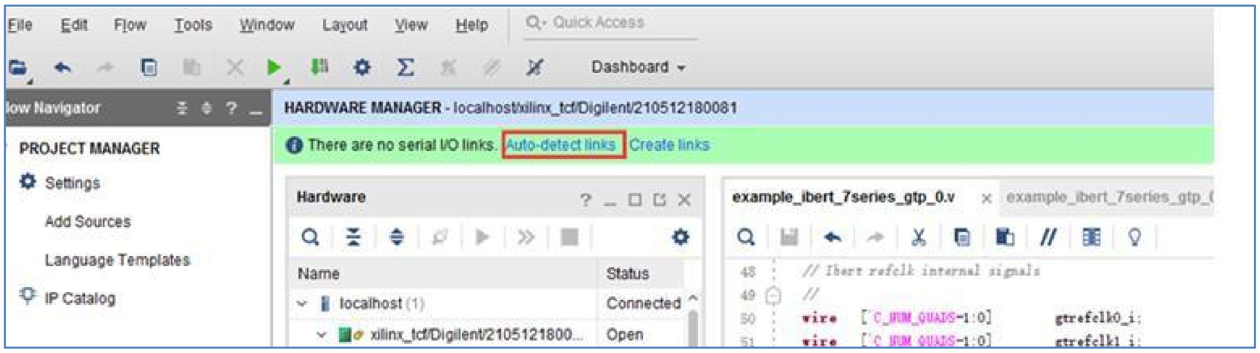
4 GTX Eye Diagram Test

Because the AXKU040 FPGA development board itself does not have SFP optical modules and optical fibers, you need to prepare SFP optical modules and optical fibers before testing. Because optical fiber transmission requires at least 2 optical modules, users need to prepare 2 SFP optical modules to do optical fiber communication experiments. 10G or 1.25G SFP optical modules and optical fibers can be purchased on Store. When purchasing an SFP optical module, it is enough for the merchant to provide supporting optical fibers.

4.1 4-way SFP communication test programs

Before the test, we inserted the optical modules of the SFP into the interfaces of the four optical modules, and then connected the optical modules OPT1 and OPT2 with optical fibers, and connected the optical module OPT3 and the optical module OPT4. Because the optical modules and optical fibers we use here are separate from TX and RX, so that the optical module RX needs to be connected to the TX of another optical module, and the TX needs to be connected to the RX of another optical module.

Download the .bit file to the FPGA in the Vivado software. After downloading, select Auto-detect links software to automatically detect the Serial I/O Links.



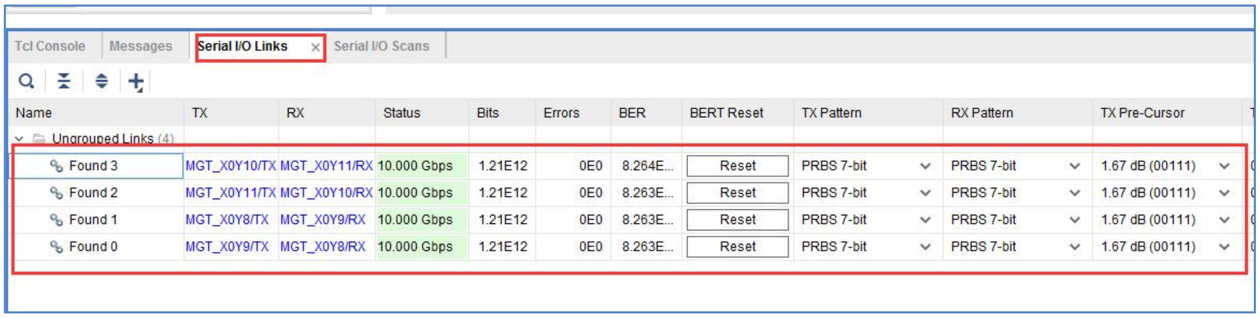
HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210512180081

There are no serial I/O links. Auto-detect links Create links

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/2105121800...	Open

```
// Ibert refclk internal signals
//
wire  [`C_NUM_QUADS-1:0]      gtrefclk0_i;
wire  [`C_NUM_QUADS-1:0]      gtrefclk1_i;
```

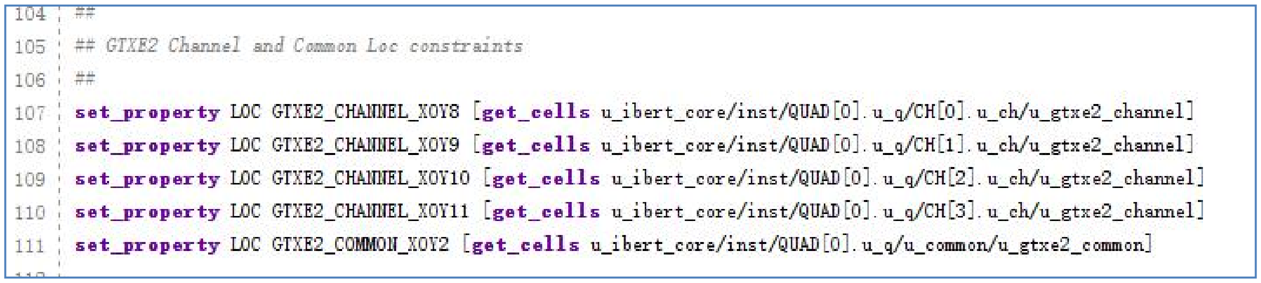
In the Serial I/O Links interface, there will be 4 channels of data communication. The following figure shows the interface of 10 Gbps connection speed.



Name	TX	RX	Status	Bits	Errors	BER	BERT Reset	TX Pattern	RX Pattern	TX Pre-Cursor
Ungrouped Links (4)										
Found 3	MGT_X0Y10/TX	MGT_X0Y11/RX	10.000 Gbps	1.21E12	0E0	8.264E...	Reset	PRBS 7-bit	PRBS 7-bit	1.67 dB (00111)
Found 2	MGT_X0Y11/TX	MGT_X0Y10/RX	10.000 Gbps	1.21E12	0E0	8.263E...	Reset	PRBS 7-bit	PRBS 7-bit	1.67 dB (00111)
Found 1	MGT_X0Y8/TX	MGT_X0Y9/RX	10.000 Gbps	1.21E12	0E0	8.263E...	Reset	PRBS 7-bit	PRBS 7-bit	1.67 dB (00111)
Found 0	MGT_X0Y9/TX	MGT_X0Y8/RX	10.000 Gbps	1.21E12	0E0	8.263E...	Reset	PRBS 7-bit	PRBS 7-bit	1.67 dB (00111)

For this interface, let's briefly introduce here, for example, the first Found 0 Link, which is sent by the TX of the MGT_X0Y0 channel, received by the RX of the MGT_X0Y8 channel, and the speed of the data link is 10Gbps. Bits is listed as the amount of data sent (for example, 1.21E-12 is sent by 1.2x10 of 12th power), and this value will increase with time. The Error item is the wrong data. Here we see 0, indicating no data reception error. BER is the bit error rate, and the number of Errors divided by the amount of data transmitted is equal to the BER error rate. The BERT Reset is the data of the reset statistics and is recounted. TX Pattern is the test data sent. The default is PRBS 7bit. Here we can also select other test data to test the transmission of fiber data.

It may not be clear to you which Fibre Channel each of MGT_X0Y8~MGT_X0Y11 represents. In the .xdc file, the following definitions are available:

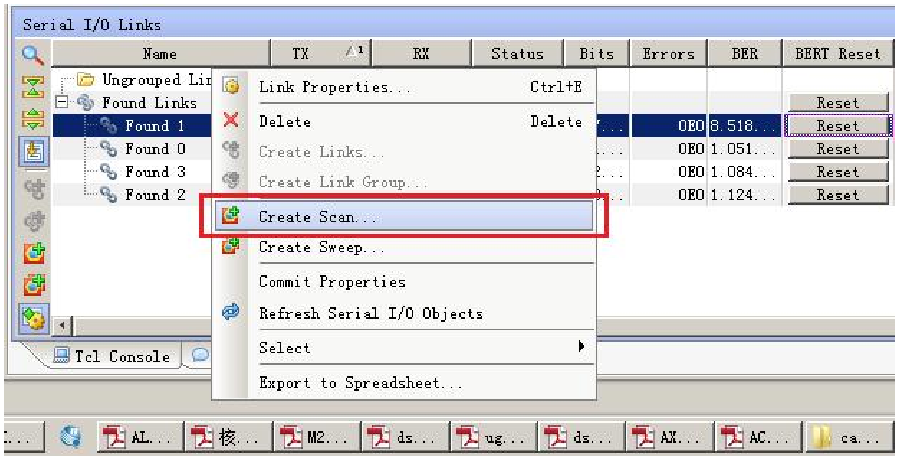


```
##
## GTXE2 Channel and Common Loc constraints
##
set_property LOC GTXE2_CHANNEL_X0Y8 [get_cells u_ibert_core/inst/QUAD[0].u_q/CH[0].u_ch/u_gtxe2_channel]
set_property LOC GTXE2_CHANNEL_X0Y9 [get_cells u_ibert_core/inst/QUAD[0].u_q/CH[1].u_ch/u_gtxe2_channel]
set_property LOC GTXE2_CHANNEL_X0Y10 [get_cells u_ibert_core/inst/QUAD[0].u_q/CH[2].u_ch/u_gtxe2_channel]
set_property LOC GTXE2_CHANNEL_X0Y11 [get_cells u_ibert_core/inst/QUAD[0].u_q/CH[3].u_ch/u_gtxe2_channel]
set_property LOC GTXE2_COMMON_X0Y2 [get_cells u_ibert_core/inst/QUAD[0].u_q/u_common/u_gtxe2_common]
```

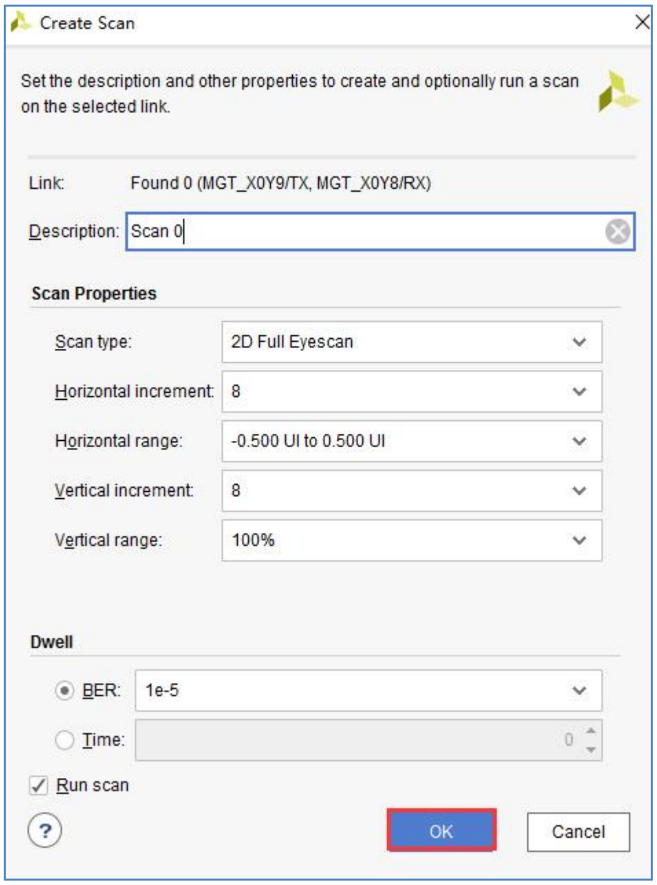
NOTE MGT_X0Y8~11 corresponds to GTX's Channel0~3 respectively. On the AX7325 FPGA development board, Channel0 of GTX is connected to OPT1, Channel1 is connected to OPT2. Channel2 is connected to OPT3, Channel4 is connected to OPT3. Therefore, the correspondence between MGT_X0Y8~11 and the OPT optical module on the development board is as follows:

MGT_ X0Yx	Channel	OPT Fiber Optical Module
MGT_ X0Y8	Channel0	OPT1
MGT_ X0Y9	Channel1	OPT2
MGT_ X0Y10	Channel2	OPT3
MGT_ X0Y11	Channel3	OPT4

Next, we will test the electrical eye diagram of fiber-optic communication. In general, the electrical eye diagram on the test board needs to be measured with a high-end oscilloscope and a differential probe. But here we do not need any external equipment or instruments, we can measure the eye diagram of fiber optic data communication, which greatly facilitates the software and hardware debugging of FPGA high-speed serial communication. Right click on the channel we want to see. For example, we select the first road Found 1 (MGT_ X0Y9 Transmit, MGT_ X0Y8 Receive), then select Create Scan from the drop-down menu.



No need to modify in the Create_scan interface, click OK


Create Scan

Set the description and other properties to create and optionally run a scan on the selected link.

Link: Found 0 (MGT_X0Y9/TX, MGT_X0Y8/RX)

Description: Scan 0

Scan Properties

Scan type:	2D Full Eyescan
Horizontal increment:	8
Horizontal range:	-0.500 UI to 0.500 UI
Vertical increment:	8
Vertical range:	100%

Dwell

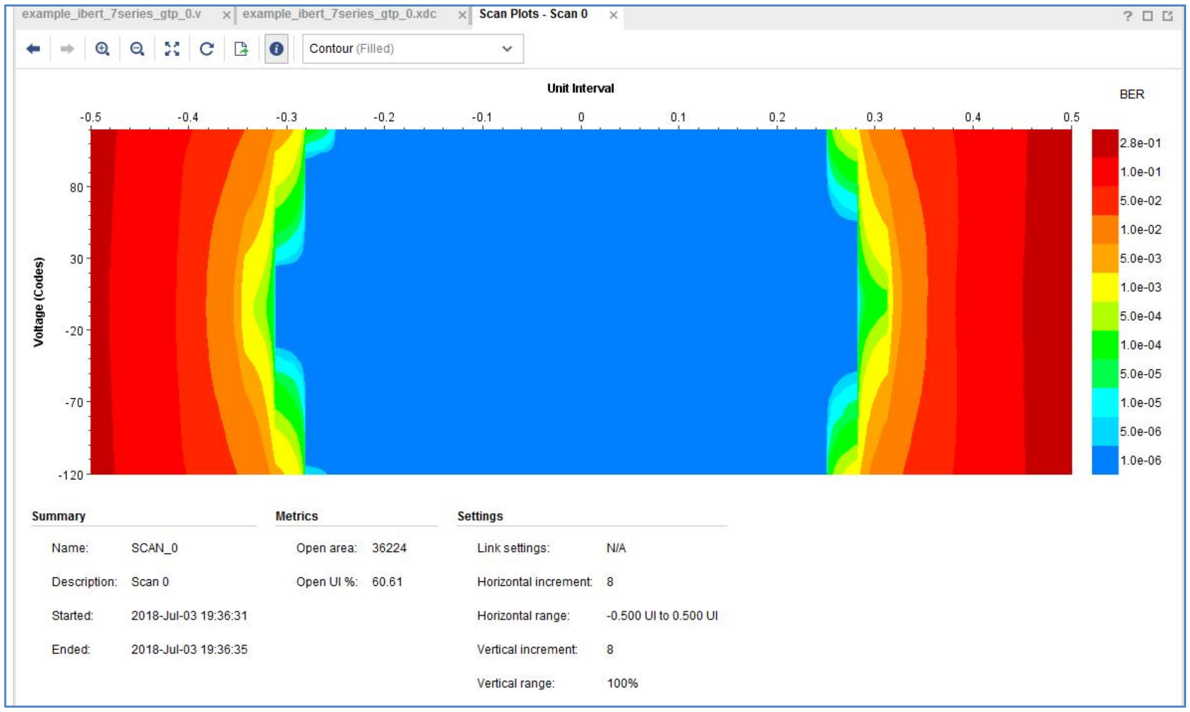
[✓ BER: 1e-5]

Time: 0

✓ Run scan

At this time, the eye diagram of this Link will be tested. The eye diagram with a link speed of 1.25Gbps is shown below:

The eye diagram with a Link speed of 1.25 Gbps is shown below:



Summary

Name: SCAN_0
Description: Scan 0
Started: 2018-Jul-03 19:36:31
Ended: 2018-Jul-03 19:36:35

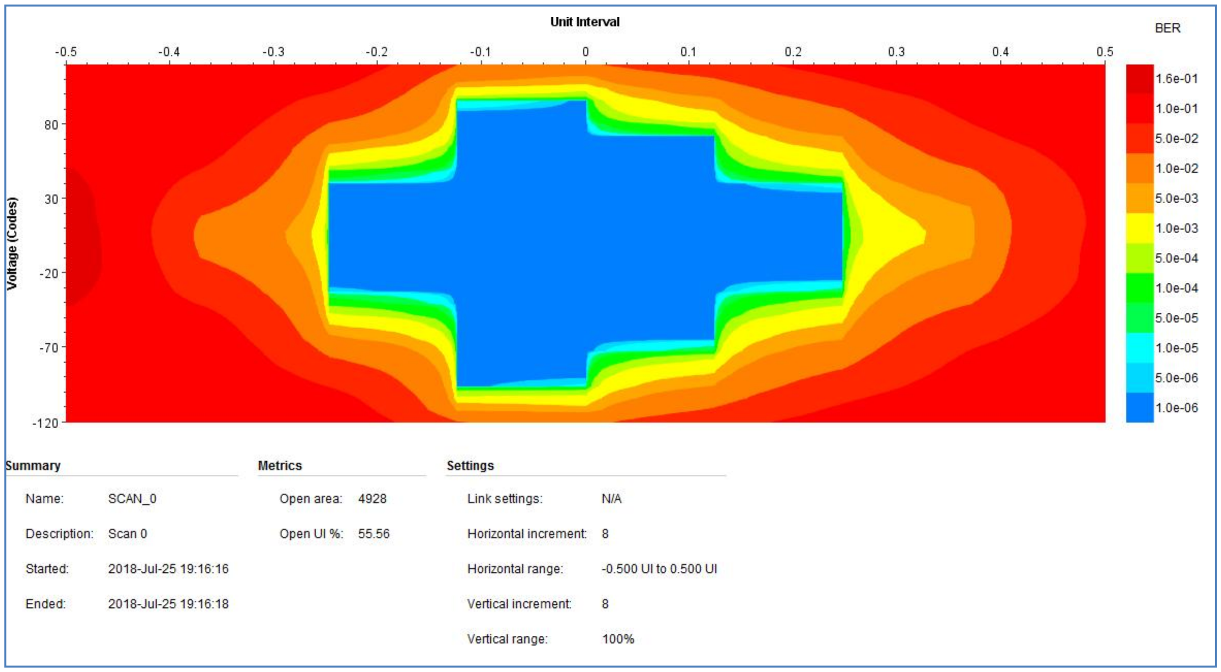
Metrics

Open area: 36224
Open UI %: 60.61

Settings

Link settings: N/A
Horizontal increment: 8
Horizontal range: -0.500 UI to 0.500 UI
Vertical increment: 8
Vertical range: 100%

The eye diagram with a Link speed of 10 Gbps is shown below:



Summary

Name: SCAN_0
Description: Scan 0
Started: 2018-Jul-25 19:16:16
Ended: 2018-Jul-25 19:16:18

Metrics

Open area: 4928
Open UI %: 55.56

Settings

Link settings: N/A
Horizontal increment: 8
Horizontal range: -0.500 UI to 0.500 UI
Vertical increment: 8
Vertical range: 100%

The bluer the color in the eye diagram, the smaller the BER value, indicating that the bit error rate is lower in this area, or there is almost no bit error rate. The redder the color, the higher the bit error rate in this area. In general, the more open the eye of this eye, the better the data transmission signal. As can be seen from the above two pictures, the lower the speed of Link, the better the eye diagram, the higher the speed of Link, the lower the corresponding eye diagram, which puts higher requirements on hardware design and PCB design. Don't talk much about the introduction of the eye diagram. Let's search for the relevant information. In the same way, you can also look at the eye diagrams of other links.