Kintex-7 Based SATA 3.0 Host Controller IP Integration Manual

Table of Content

List Of Figures

List Of Tables

1 Introduction

1.1 Purpose

The purpose of this document is to describe the details of SATA 3.0 Host Controller Integration with Kintex-7 Development kit.

1.2 Overview

SATA 3.0 Host Controller interfaces the Micro blaze Processor through AXI4-Bus enabling the data transfers between each other. The Micro blaze Processor will send the response to the GPIO's depending on the command issued.

1.3 Acronyms & Abbreviations

Table 1: Acronyms & Abbreviations

Term	Meaning
FPGA	Field Programmable Gate Array
GPIO	General purpose input output
FMC	FPGA Mezzanine Card
LUT	Look Up Table
IO	Input and Output
FF	Flip Flop

2 SATA 3.0 Host Controller

The below figure represents the test setup for the Kintex-7 Development Board with SATA FMC Daughter card and Intel SSD Device.

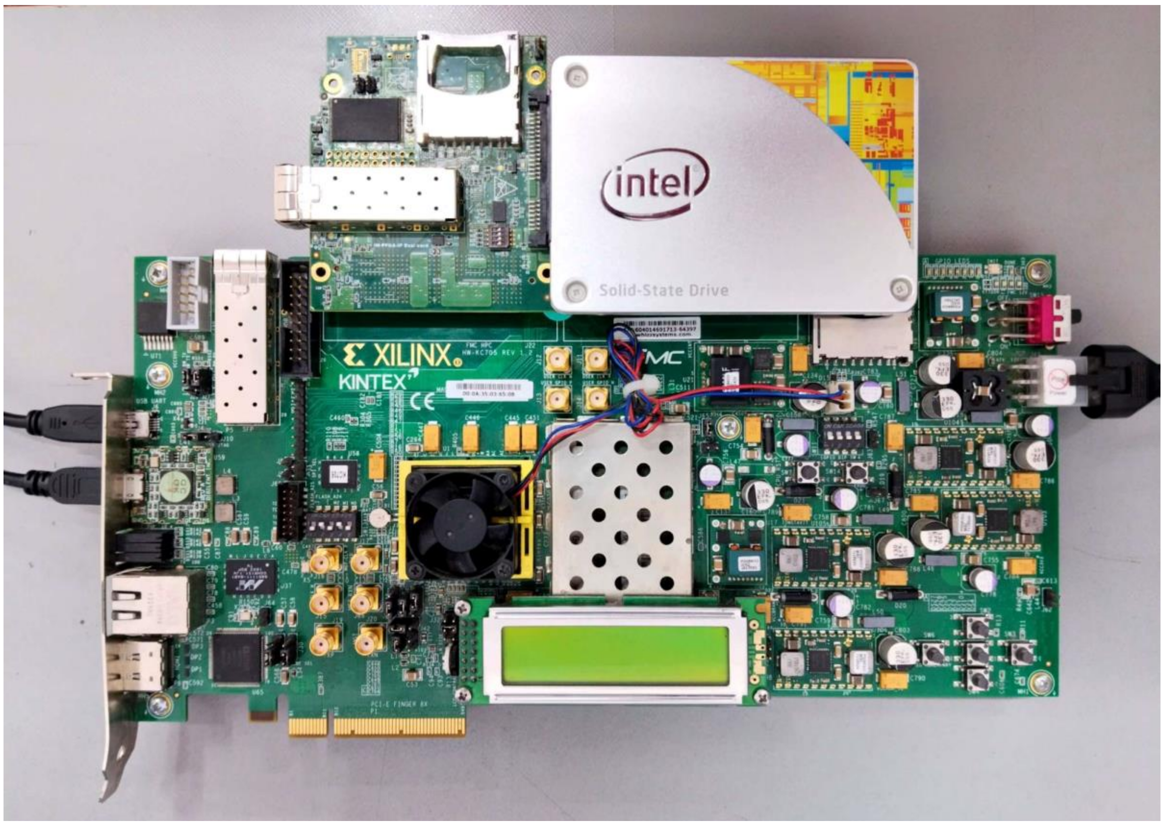


Figure 1: SATA 3.0 Host Controller setup

2.1 SATA FMC Daughter Card

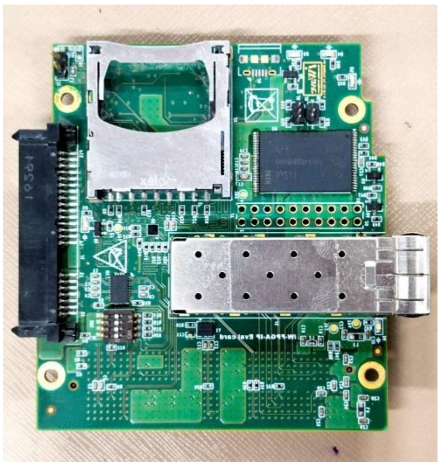


Figure 2: SATA FMC Daughter Card

3 IP Configuration and Instantiation

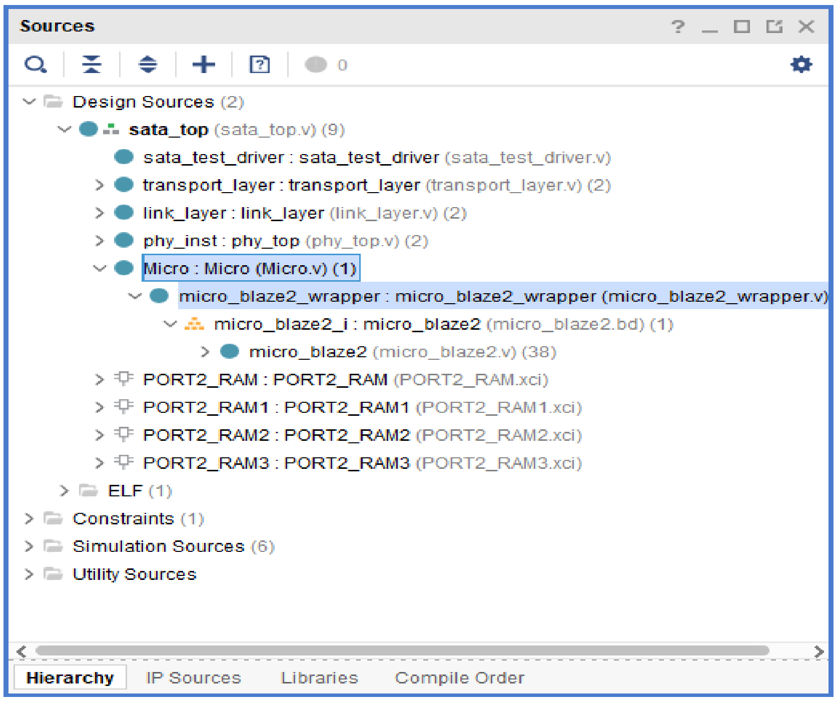
3.1 Example Design

The SATA 3.0 Host Controller example design mainly consists of

1. **Test Driver:** Test Driver responsible for issuing commands for read and write operations towards through the transport layer. And its responsible for overall command execution, including control of Register accesses.
2. **Micro blaze processor:** Micro blaze Processor is used to configure GPIO through the AXI-4 Interconnect mainly access to read and write operation to the device.
3. **Transport Layer:** The Transport layer is responsible for placing control information and data to be transferred between the host and device in a packet/frame, known as a Frame Information Structure (FIS).
4. **Link Layer:** The Link layer is responsible for taking data from the constructed frames, scramble or de-scramble and perform CRC check.
5. **Physical Layer:** The Physical layer is responsible for transmitting and receiving the (8B/10B) encoded information as a serial data stream on the line. This layer consists of 3 blocks. That was Speed negotiation, Transceiver wizard, OOB signaling and control.
    a. **GTX Transceiver wizard:** responsible for highspeed serial communication in the physical layer.
    b. **Speed negotiation:** responsible for changing the line rate (Ex: 6GB/s to 3GB/s).
    c. **OOB signaling:** responsible for OOB signal generation and detection, provide status to link layer.

3.2 SATA 3.0 Host Controller IP Instantiation

The SATA 3.0 Host Controller block design (Micro.v) is instantiated in the design as shown in the below Figure 3.


Sources

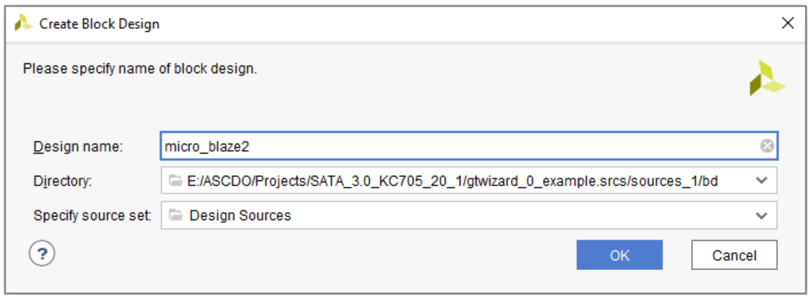
Design Sources (2)
sata_top (sata_top.v) (9)
sata_test_driver : sata_test_driver (sata_test_driver.v)
transport_layer : transport_layer (transport_layer.v) (2)
link_layer : link_layer (link_layer.v) (2)
phy_inst : phy_top (phy_top.v) (2)
Micro : Micro (Micro.v) (1)
micro_blaze2_wrapper : micro_blaze2_wrapper (micro_blaze2_wrapper.v)
micro_blaze2_i : micro_blaze2 (micro_blaze2.bd) (1)
micro_blaze2 (micro_blaze2.v) (38)
PORT2_RAM : PORT2_RAM (PORT2_RAM.xci)
PORT2_RAM1 : PORT2_RAM1 (PORT2_RAM1.xci)
PORT2_RAM2 : PORT2_RAM2 (PORT2_RAM2.xci)
PORT2_RAM3 : PORT2_RAM3 (PORT2_RAM3.xci)
ELF (1)
Constraints (1)
Simulation Sources (6)
Utility Sources

Figure 3: Instantiation Module of SATA 3.0 Host Controller Block design

Module sata_top was the top module of the project which integrates with physical layer, link layer, transport layer, test driver along with SATA 3.0 Host Controller module.

3.3 Steps to Configuring the SATA 3.0 Host Controller

- Install the required Vivado Design Suite for the host PC adding the license path. Downloads (xilinx.com)
- Open the SATA 3.0 Host Controller project. Go to the Flow navigator → Project Manager → IP Integrator → open "Create Block design".
- Then give block design name (micro_blaze2) and directory as E:\ASCDO\Projects\SATA_3.0_KC705_20_1\gtwizard_0_example.srcs\sources_1\bd\micro_blaze2/ micro_blaze2.bd in the below Figure 4.


Create Block Design

Please specify name of block design.

Design name: micro_blaze2
Directory: E:/ASCDO/Projects/SATA_3.0_KC705_20_1/gtwizard_0_example.srcs/sources_1/bd
Specify source set: Design Sources

Figure 4: SATA 3.0 Host Controller Block Design Configuration step 1

- Diagram and Address Editor windows will be opened. In Diagram window, click ‘+’ to add the IP into the block deign. What is the required IP need for the design that should be added.

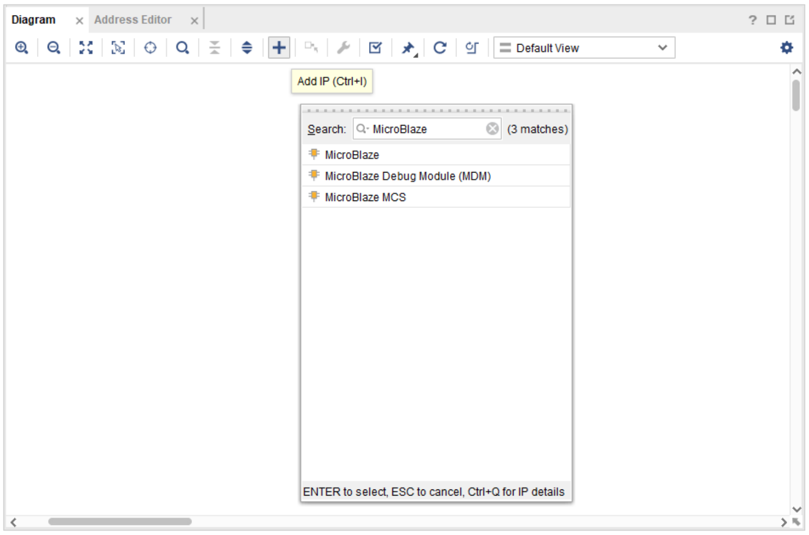


Diagram
Address Editor
Default View
Add IP (Ctrl+I)
Search: MicroBlaze (3 matches)
MicroBlaze
MicroBlaze Debug Module (MDM)
MicroBlaze MCS
ENTER to select, ESC to cancel, Ctrl+Q for IP details

Figure 5: SATA 3.0 Host Controller Block Design Configuration step 2

- After adding IP like Micro blaze, processor system reset, AXI4-interconnect, Micro blaze Debug Module, clocking wizard and GPIO, the required GPIO’s are connected to the Micro blaze by using AXI4-stream interconnect to the block design as per shown in below Figure 6. Micro blaze was configured by as per the SATA design requirements.

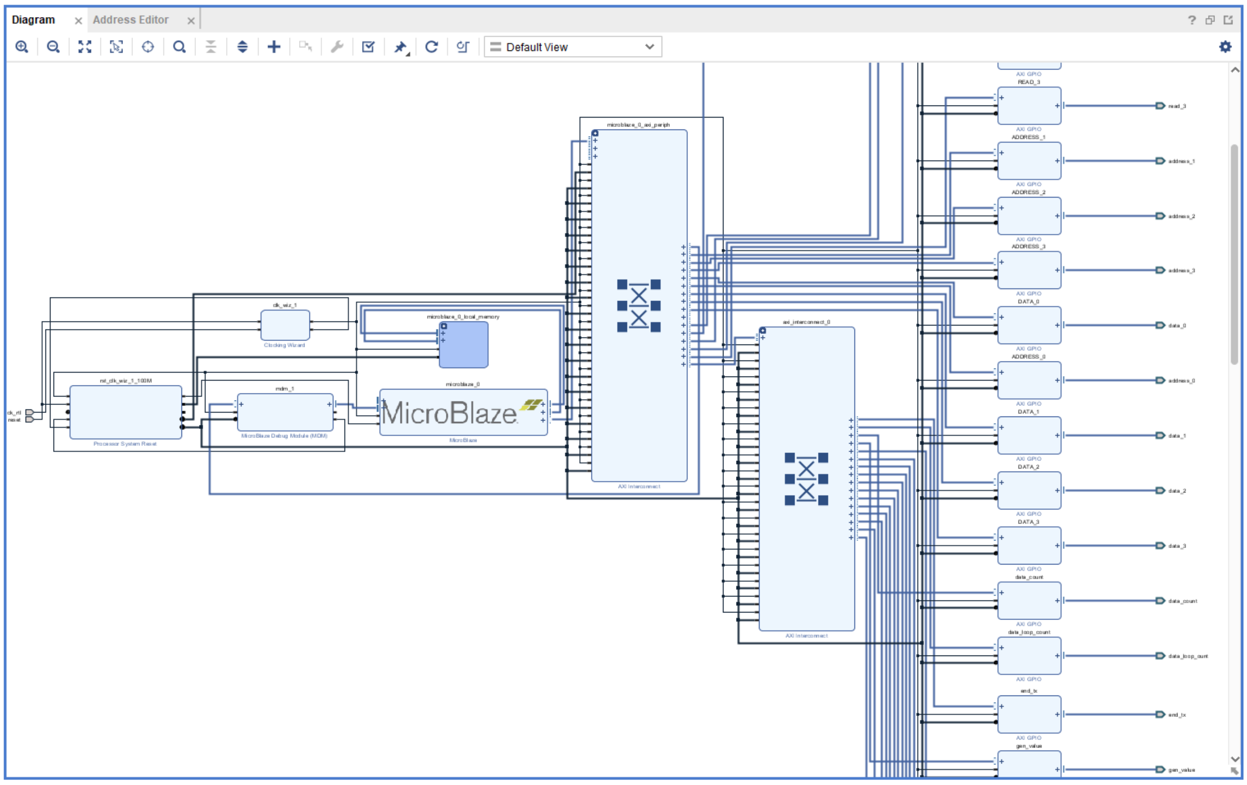
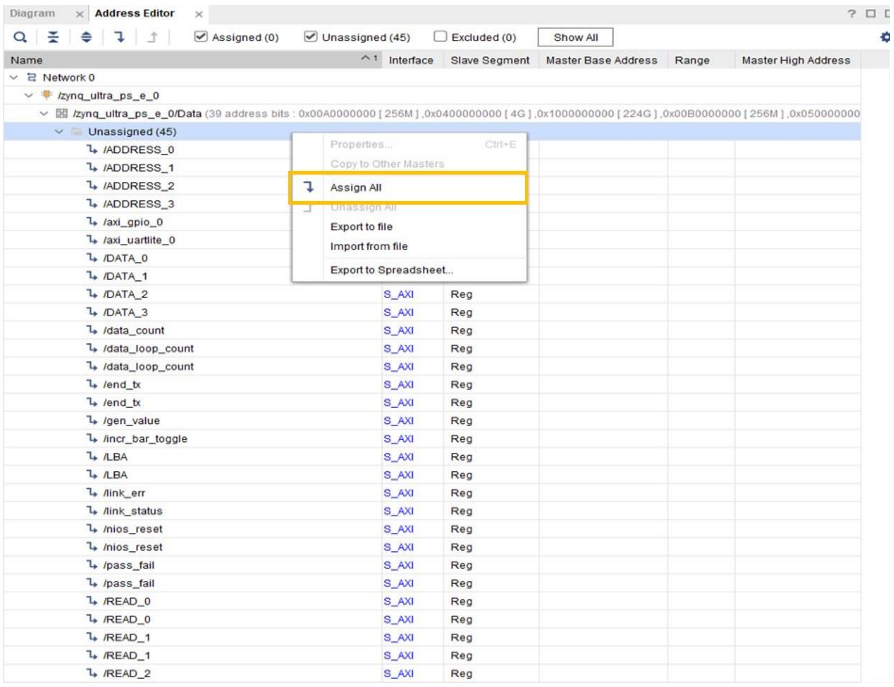


Figure 6: SATA 3.0 Host Controller Block Design Configuration step 3

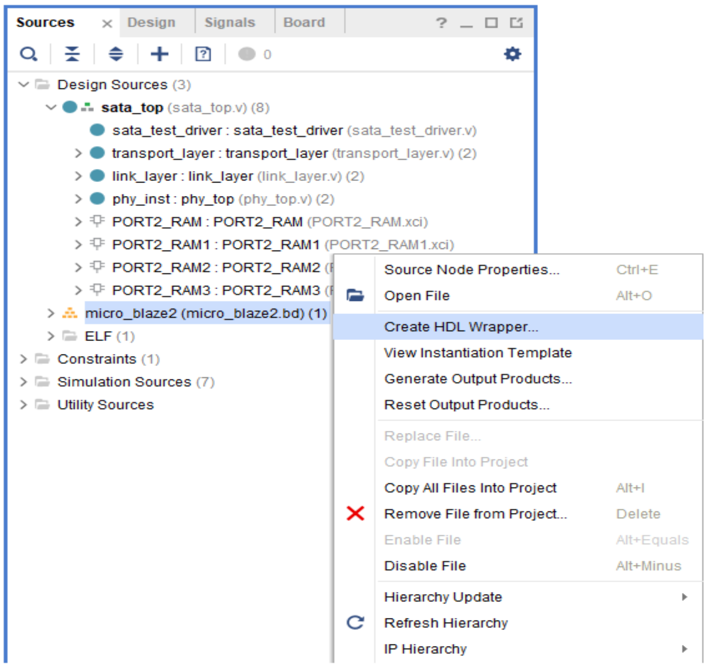
- Once the block design was created, then go to Address Editor window to assign the address to all the GPIO's as shown in Figure 7.



Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
/zynq_ultra_ps_e_0					
/zynq_ultra_ps_e_0/Data (39 address bits : 0x00A0000000 [256M] ,0x0400000000 [4G] ,0x1000000000 [224G] ,0x00B0000000 [256M] ,0x050000000					
Unassigned (45)					
/ADDRESS_0					
/ADDRESS_1					
/ADDRESS_2					
/ADDRESS_3					
/axi_gpio_0					
/axi_uartlite_0					
/DATA_0					
/DATA_1					
/DATA_2	S_AXI	Reg			
/DATA_3	S_AXI	Reg			
/data_count	S_AXI	Reg			
/data_loop_count	S_AXI	Reg			
/data_loop_count	S_AXI	Reg			
/end_tx	S_AXI	Reg			
/end_tx	S_AXI	Reg			
/gen_value	S_AXI	Reg			
/incr_bar_toggle	S_AXI	Reg			
/LBA	S_AXI	Reg			
/LBA	S_AXI	Reg			
/link_err	S_AXI	Reg			
/link_status	S_AXI	Reg			
/nios_reset	S_AXI	Reg			
/nios_reset	S_AXI	Reg			
/pass_fail	S_AXI	Reg			
/pass_fail	S_AXI	Reg			
/READ_0	S_AXI	Reg			
/READ_0	S_AXI	Reg			
/READ_1	S_AXI	Reg			
/READ_1	S_AXI	Reg			
/READ_2	S_AXI	Reg			

Figure 7: SATA 3.0 Host Controller Block Design Configuration step 4

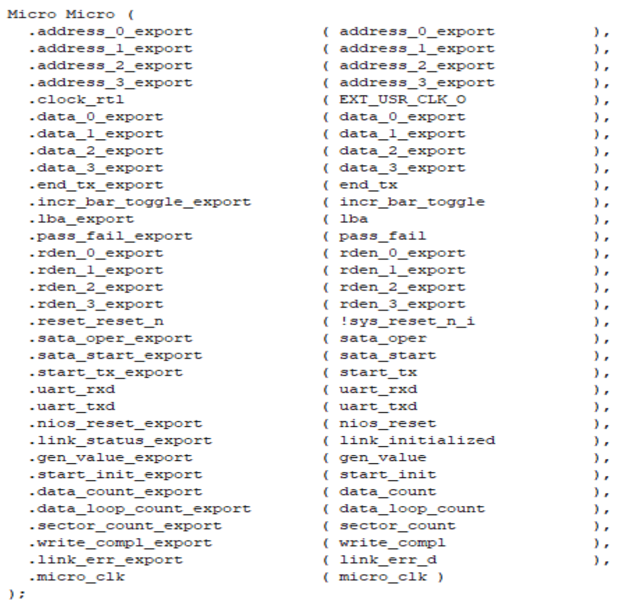
- Synthesize the block design first by validating the design (press key 'F6') and generate the outputs. Once the design validation was completed then generate wrapper module of the block design by right click the .bd file and click Create HDL wrapper as shown in Figure


Sources
Design
Signals
Board
Design Sources (3)
sata_top (sata_top.v) (8)
sata_test_driver : sata_test_driver (sata_test_driver.v)
transport_layer : transport_layer (transport_layer.v) (2)
link_layer : link_layer (link_layer.v) (2)
phy_inst : phy_top (phy_top.v) (2)
PORT2_RAM : PORT2_RAM (PORT2_RAM.xci)
PORT2_RAM1 : PORT2_RAM1 (PORT2_RAM1.xci)
PORT2_RAM2 : PORT2_RAM2
PORT2_RAM3 : PORT2_RAM3
micro_blaze2 (micro_blaze2.bd) (1)
ELF (1)
Constraints (1)
Simulation Sources (7)
Utility Sources
Source Node Properties... Ctrl+E
Open File Alt+O
Create HDL Wrapper...
View Instantiation Template
Generate Output Products...
Reset Output Products...
Replace File...
Copy File Into Project
Copy All Files Into Project Alt+I
Remove File from Project... Delete
Enable File Alt+Equals
Disable File Alt+Minus
Hierarchy Update
Refresh Hierarchy
IP Hierarchy

8. After the wrapper file creation then Instantiate inside the top module of design as shown in Figure 3.

Figure 8: SATA 3.0 Host Controller Block Design Configuration step 5

- And the Micro file instantiation inside the sata_top module as shown in Figure 9.

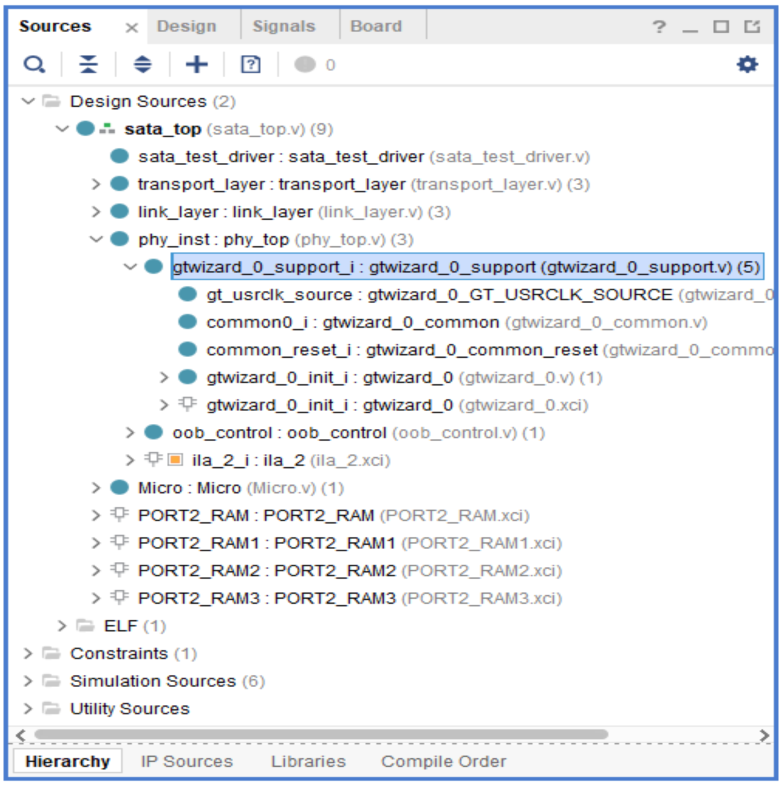


```
Micro Micro (
   .address_0_export            ( address_0_export          ),
   .address_1_export            ( address_1_export          ),
   .address_2_export            ( address_2_export          ),
   .address_3_export            ( address_3_export          ),
   .clock_rtl                   ( EXT_USR_CLK_O             ),
   .data_0_export               ( data_0_export             ),
   .data_1_export               ( data_1_export             ),
   .data_2_export               ( data_2_export             ),
   .data_3_export               ( data_3_export             ),
   .end_tx_export               ( end_tx                    ),
   .incr_bar_toggle_export      ( incr_bar_toggle           ),
   .lba_export                  ( lba                       ),
   .pass_fail_export            ( pass_fail                 ),
   .rden_0_export               ( rden_0_export             ),
   .rden_1_export               ( rden_1_export             ),
   .rden_2_export               ( rden_2_export             ),
   .rden_3_export               ( rden_3_export             ),
   .reset_reset_n               ( !sys_reset_n_i            ),
   .sata_oper_export            ( sata_oper                 ),
   .sata_start_export           ( sata_start                ),
   .start_tx_export             ( start_tx                  ),
   .uart_rxd                    ( uart_rxd                  ),
   .uart_txd                    ( uart_txd                  ),
   .nios_reset_export           ( nios_reset                ),
   .link_status_export          ( link_initialized          ),
   .gen_value_export            ( gen_value                 ),
   .start_init_export           ( start_init                ),
   .data_count_export           ( data_count                ),
   .data_loop_count_export      ( data_loop_count           ),
   .sector_count_export         ( sector_count              ),
   .write_compl_export          ( write_compl               ),
   .link_err_export             ( link_err_d                ),
   .micro_clk                   ( micro_clk )
);
```

Figure 9: Instantiation of SATA 3.0 Host Controller Block Design

3.4 GTX transceiver IP Instantiation

Create one example design of GTX Transceiver wizard as per SATA the protocol in the Vivado. And configure GTX Transceiver wizard settings as per design. Then instantiate the example design of GTX Transceiver wizard to the physical layer module as shown in below Figure 10.



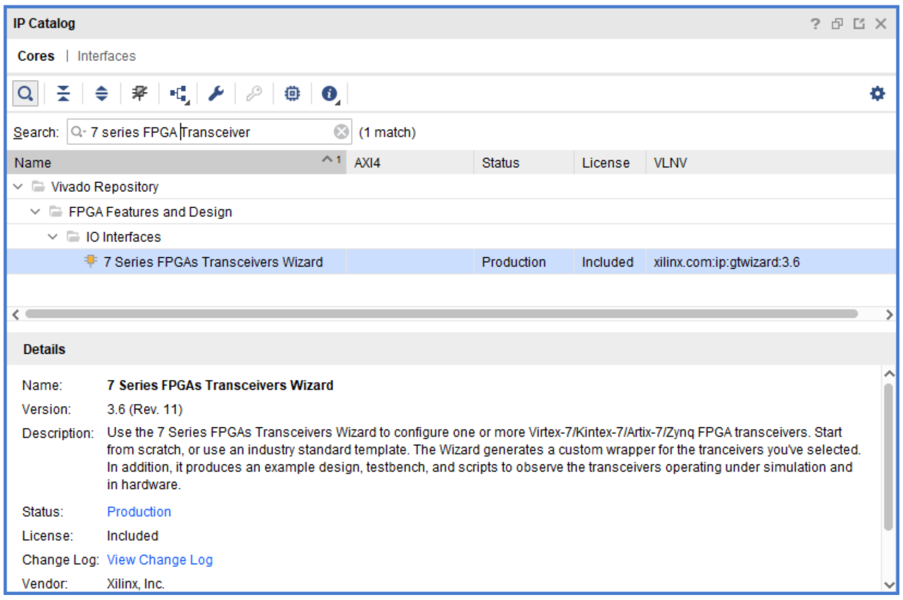
```
Design Sources (2)
  sata_top (sata_top.v) (9)
    sata_test_driver : sata_test_driver (sata_test_driver.v)
    transport_layer : transport_layer (transport_layer.v) (3)
    link_layer : link_layer (link_layer.v) (3)
    phy_inst : phy_top (phy_top.v) (3)
      gtwizard_0_support_i : gtwizard_0_support (gtwizard_0_support.v) (5)
        gt_usrclk_source : gtwizard_0_GT_USRCLK_SOURCE (gtwizard_0
        common0_i : gtwizard_0_common (gtwizard_0_common.v)
        common_reset_i : gtwizard_0_common_reset (gtwizard_0_commo
        gtwizard_0_init_i : gtwizard_0 (gtwizard_0.v) (1)
        gtwizard_0_init_i : gtwizard_0 (gtwizard_0.xci)
      oob_control : oob_control (oob_control.v) (1)
      ila_2_i : ila_2 (ila_2.xci)
    Micro : Micro (Micro.v) (1)
    PORT2_RAM : PORT2_RAM (PORT2_RAM.xci)
    PORT2_RAM1 : PORT2_RAM1 (PORT2_RAM1.xci)
    PORT2_RAM2 : PORT2_RAM2 (PORT2_RAM2.xci)
    PORT2_RAM3 : PORT2_RAM3 (PORT2_RAM3.xci)
  ELF (1)
Constraints (1)
Simulation Sources (6)
Utility Sources
```

Figure 10: Instantiation Module of GTH Transceiver wizard

The Figure 10 represents the gtwizard_0_support.v module of GTX Transceiver wizard instantiated inside the physical layer module (phy_top).

3.5 Steps to Configure the GTH Transceiver IP

- Go to the Flow navigator → Project Manager → IP Catalog. In IP catalog, add the required 7 series FPGA Transceiver wizard as shown in below Figure 11.



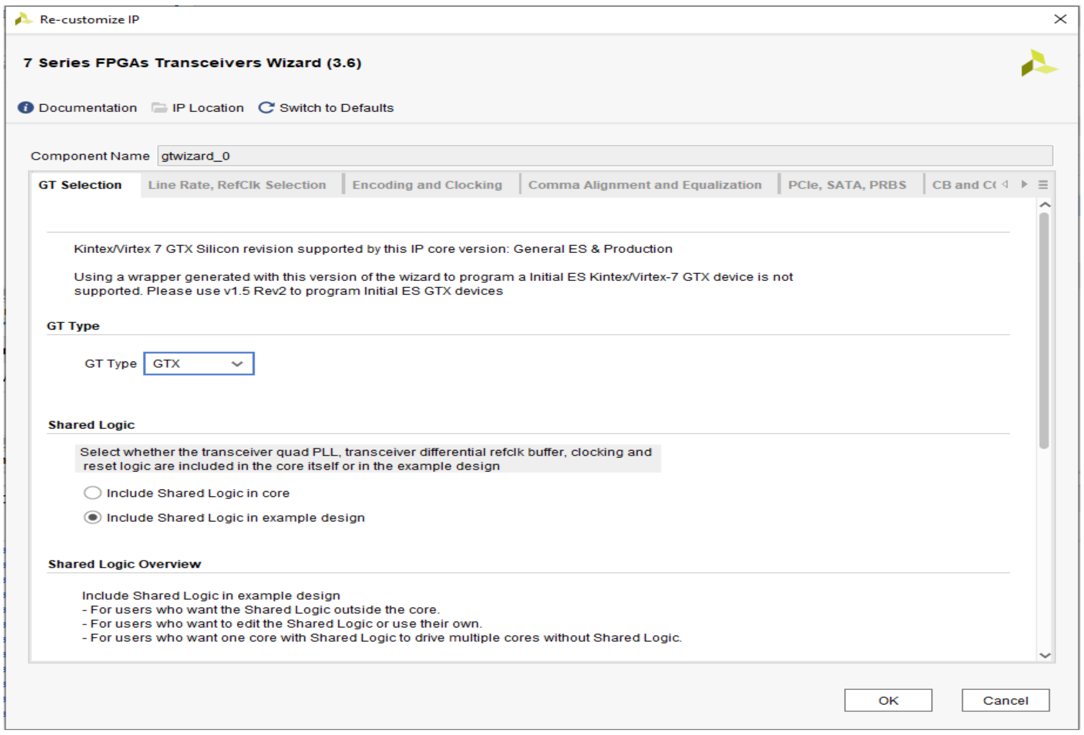
Name	AXI4	Status	License	VLNV
Vivado Repository				
FPGA Features and Design				
IO Interfaces				
7 Series FPGAs Transceivers Wizard		Production	Included	xilinx.com:ip:gtwizard:3.6

Details

Name:	7 Series FPGAs Transceivers Wizard
Version:	3.6 (Rev. 11)
Description:	Use the 7 Series FPGAs Transceivers Wizard to configure one or more Virtex-7/Kintex-7/Artix-7/Zynq FPGA transceivers. Start from scratch, or use an industry standard template. The Wizard generates a custom wrapper for the tranceivers you've selected. In addition, it produces an example design, testbench, and scripts to observe the transceivers operating under simulation and in hardware.
Status:	Production
License:	Included
Change Log:	View Change Log
Vendor:	Xilinx, Inc.

Figure 11: GTX Transceiver Configuration step 1

- After that 7 series FPGA Transceiver wizard (3.6) will be opened and set the transceiver configuration GT type as "GTX". And configure the GTX-Transceiver wizard IP as per SATA 3.0 Host Controller design. The Line rate, RefClk selection and PLL type and rest of the settings should be configured as per the design.



Re-customize IP

7 Series FPGAs Transceivers Wizard (3.6)

Documentation | IP Location | Switch to Defaults

Component Name: gtwizard_0

GT Selection | Line Rate, RefClk Selection | Encoding and Clocking | Comma Alignment and Equalization | PCIe, SATA, PRBS | CB and C(

Kintex/Virtex 7 GTX Silicon revision supported by this IP core version: General ES & Production

Using a wrapper generated with this version of the wizard to program a Initial ES Kintex/Virtex-7 GTX device is not supported. Please use v1.5 Rev2 to program Initial ES GTX devices

GT Type

GT Type: GTX

Shared Logic

Select whether the transceiver quad PLL, transceiver differential refclk buffer, clocking and reset logic are included in the core itself or in the example design

Include Shared Logic in core [✓ Include Shared Logic in example design]

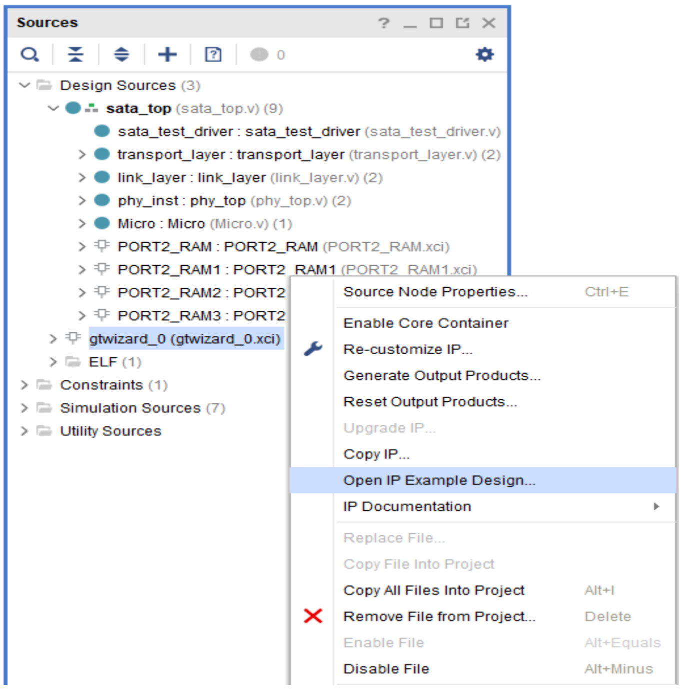
Shared Logic Overview

Include Shared Logic in example design
- For users who want the Shared Logic outside the core.
- For users who want to edit the Shared Logic or use their own.
- For users who want one core with Shared Logic to drive multiple cores without Shared Logic.

OK | Cancel

Figure 12: GTX Transceiver Configuration step 2

- Once the configuration was completed, then generate outputs of GTX Transceiver wizard. After right click on the GTX Transceiver wizard to open the IP example design.

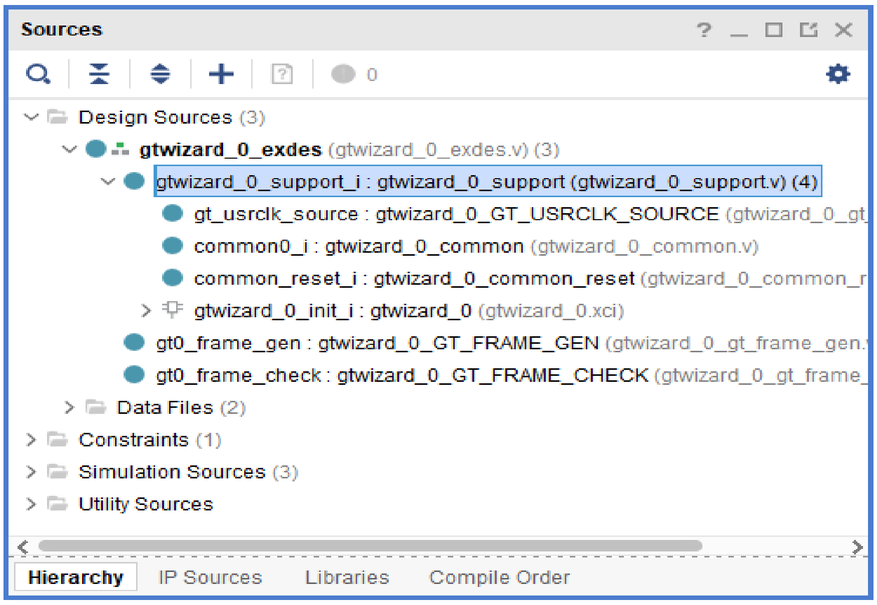


```
Sources
Design Sources (3)
  sata_top (sata_top.v) (9)
    sata_test_driver : sata_test_driver (sata_test_driver.v)
    transport_layer : transport_layer (transport_layer.v) (2)
    link_layer : link_layer (link_layer.v) (2)
    phy_inst : phy_top (phy_top.v) (2)
    Micro : Micro (Micro.v) (1)
    PORT2_RAM : PORT2_RAM (PORT2_RAM.xci)
    PORT2_RAM1 : PORT2_RAM1 (PORT2_RAM1.xci)
    PORT2_RAM2 : PORT2
    PORT2_RAM3 : PORT2
  gtwizard_0 (gtwizard_0.xci)
  ELF (1)
Constraints (1)
Simulation Sources (7)
Utility Sources
```

```
Source Node Properties...        Ctrl+E
Enable Core Container
Re-customize IP...
Generate Output Products...
Reset Output Products...
Upgrade IP...
Copy IP...
Open IP Example Design...
IP Documentation
Replace File...
Copy File Into Project
Copy All Files Into Project      Alt+I
Remove File from Project...      Delete
Enable File                      Alt+Equals
Disable File                     Alt+Minus
```

Figure 13: GTX Transceiver Configuration step 3

- Give the path as E:\ASCDO\Projects\SATA_3.0_KC705_20_1\gtwizard_0_ex for open the example design of GTX Transceiver wizard. The Example design of GTX Transceiver wizard was containing gtwizard_0_exdes and other modules.



Sources

Design Sources (3)
- gtwizard_0_exdes (gtwizard_0_exdes.v) (3)
  - gtwizard_0_support_i : gtwizard_0_support (gtwizard_0_support.v) (4)
    - gt_usrclk_source : gtwizard_0_GT_USRCLK_SOURCE (gtwizard_0_gt
    - common0_i : gtwizard_0_common (gtwizard_0_common.v)
    - common_reset_i : gtwizard_0_common_reset (gtwizard_0_common_r
    - gtwizard_0_init_i : gtwizard_0 (gtwizard_0.xci)
  - gt0_frame_gen : gtwizard_0_GT_FRAME_GEN (gtwizard_0_gt_frame_gen.
  - gt0_frame_check : gtwizard_0_GT_FRAME_CHECK (gtwizard_0_gt_frame_
- Data Files (2)

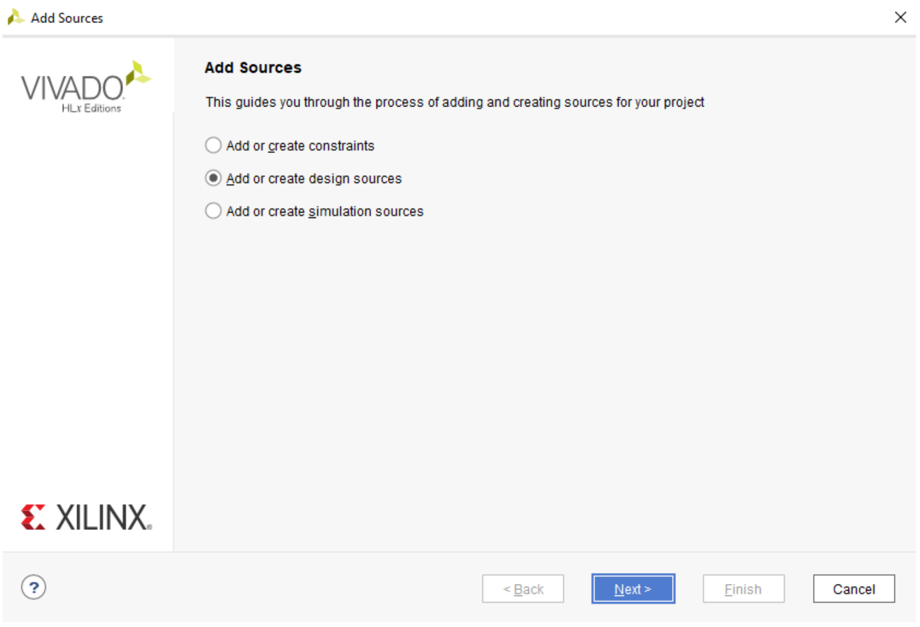
Constraints (1)

Simulation Sources (3)

Utility Sources

Figure 14: GTX Transceiver Configuration step 4

- Then go to the SATA 3.0 Host Controller design, right click the "+" inside the source window and give "Add or create design sources". Then go to Add files → gtwizard_0_ex → imports.



Add Sources

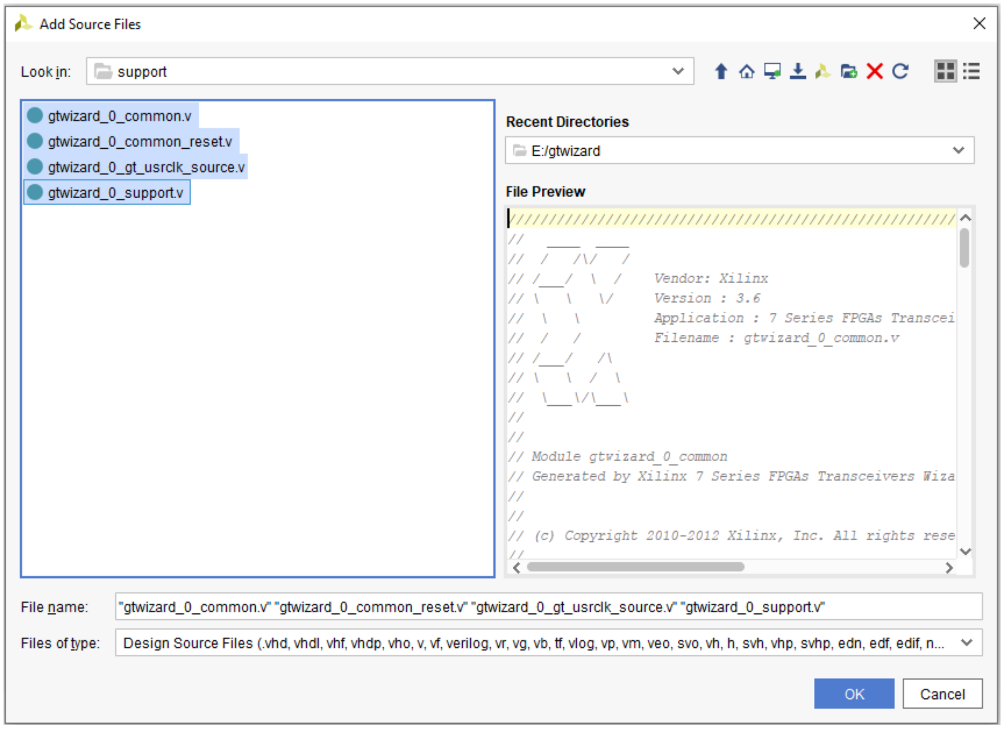
Add Sources

This guides you through the process of adding and creating sources for your project

- ( ) Add or create constraints
- (●) Add or create design sources
- ( ) Add or create simulation sources

Figure 15: GTX Transceiver Configuration step 5

- Add the support.v module and required modules of GTX Transceiver wizard to the SATA 3.0 Host Controller design.



gtwizard_0_common.v
gtwizard_0_common_reset.v
gtwizard_0_gt_usrclk_source.v
gtwizard_0_support.v

```
/////////////////////////////////////////////////////
//   ____  ____
//  /   /\/   /
// /___/  \  /    Vendor: Xilinx
// \   \   \/     Version : 3.6
//  \   \         Application : 7 Series FPGAs Transcei
//  /   /         Filename : gtwizard_0_common.v
// /___/   /\
// \   \  /  \
//  \___\/\___\
//
//
// Module gtwizard_0_common
// Generated by Xilinx 7 Series FPGAs Transceivers Wiza
//
//
// (c) Copyright 2010-2012 Xilinx, Inc. All rights rese
//
```

File name: "gtwizard_0_common.v" "gtwizard_0_common_reset.v" "gtwizard_0_gt_usrclk_source.v" "gtwizard_0_support.v"

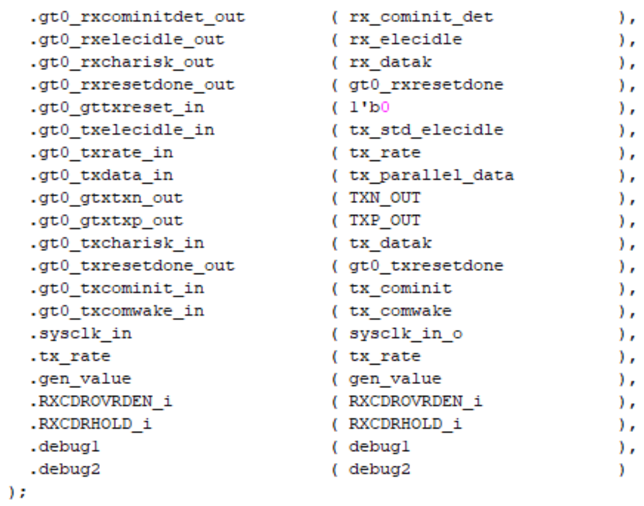
Files of type: Design Source Files (.vhd, vhdl, vhf, vhdp, vho, v, vf, verilog, vr, vg, vb, tf, vlog, vp, vm, veo, svo, vh, h, svh, vhp, svhp, edn, edf, edif, n...

Figure 16: GTX Transceiver Configuration step 6

- And the gtwizard_0_support instantiation of GTX Transceiver wizard inside the phy_top module as shown in Figure 17.

```
gtwizard_0_support #
(
  .EXAMPLE_SIM_GTRESET_SPEEDUP  ( "TRUE" ),
  .STABLE_CLOCK_PERIOD          ( 16     )
)
gtwizard_0_support_i
(
  .soft_reset_tx_in             ( sys_reset_i || reset_tx   ),
  .soft_reset_rx_in             ( sys_reset_i || reset_tx   ),
  .dont_reset_on_data_error_in  ( 1'b0                      ),
  .q3_clk0_gtrefclk_pad_n_in    ( Q3_CLK1_GTREFCLK_PAD_N_IN ),
  .q3_clk0_gtrefclk_pad_p_in    ( Q3_CLK1_GTREFCLK_PAD_P_IN ),
  .gt0_data_valid_in            ( 1'b1                      ),
  .gt0_txusrclk_out             (                           ),
  .gt0_txusrclk2_out            ( tx_std_clkout_o           ),
  .gt0_rxusrclk_out             (                           ),
  .gt0_rxusrclk2_out            ( rx_std_clkout_o           ),
  .gt0_cpllreset_in             ( 1'b0                      ),
  .gt0_rxrate_in                ( tx_rate_d2                ),
  .gt0_eyescanreset_in          ( 1'h0                      ),
  .gt0_rxuserrdy_in             ( 1'h0                      ),
  .gt0_eyescantrigger_in        ( 1'h0                      ),
  .gt0_rxdata_out               ( rx_parallel_data          ),
  .gt0_gtxrxp_in                ( RXP_IN                    ),
  .gt0_gtxrxn_in                ( RXN_IN                    ),
  .gt0_rxdfelpmreset_in         ( 1'h0                      ),
  .gt0_rxmonitorsel_in          ( 2'h0                      ),
  .gt0_gtrxreset_in             ( 1'b0                      ),
  .gt0_rxcomwakedet_out         ( rx_comwake_det            ),
```

Figure 17: Instantiation of GTX transceiver wizard (i)



```
  .gt0_rxcominitdet_out         ( rx_cominit_det            ),
  .gt0_rxelecidle_out           ( rx_elecidle               ),
  .gt0_rxcharisk_out            ( rx_datak                  ),
  .gt0_rxresetdone_out          ( gt0_rxresetdone           ),
  .gt0_gttxreset_in             ( 1'b0                      ),
  .gt0_txelecidle_in            ( tx_std_elecidle           ),
  .gt0_txrate_in                ( tx_rate                   ),
  .gt0_txdata_in                ( tx_parallel_data          ),
  .gt0_gtxtxn_out               ( TXN_OUT                   ),
  .gt0_gtxtxp_out               ( TXP_OUT                   ),
  .gt0_txcharisk_in             ( tx_datak                  ),
  .gt0_txresetdone_out          ( gt0_txresetdone           ),
  .gt0_txcominit_in             ( tx_cominit                ),
  .gt0_txcomwake_in             ( tx_comwake                ),
  .sysclk_in                    ( sysclk_in_o               ),
  .tx_rate                      ( tx_rate                   ),
  .gen_value                    ( gen_value                 ),
  .RXCDROVRDEN_i                ( RXCDROVRDEN_i             ),
  .RXCDRHOLD_i                  ( RXCDRHOLD_i               ),
  .debug1                       ( debug1                    ),
  .debug2                       ( debug2                    )
);
```

Figure 18: Instantiation of GTX transceiver wizard (ii)

- The Constraint file (.xdc) provided in the design is for Xilinx Kintex-7 development Board and should be changed for custom boards.
- Give the required clock, Pin/IO constraints for SATA 3.0 Host Controller in the .xdc file and compile the custom design with SATA 3.0 Host Controller IP.

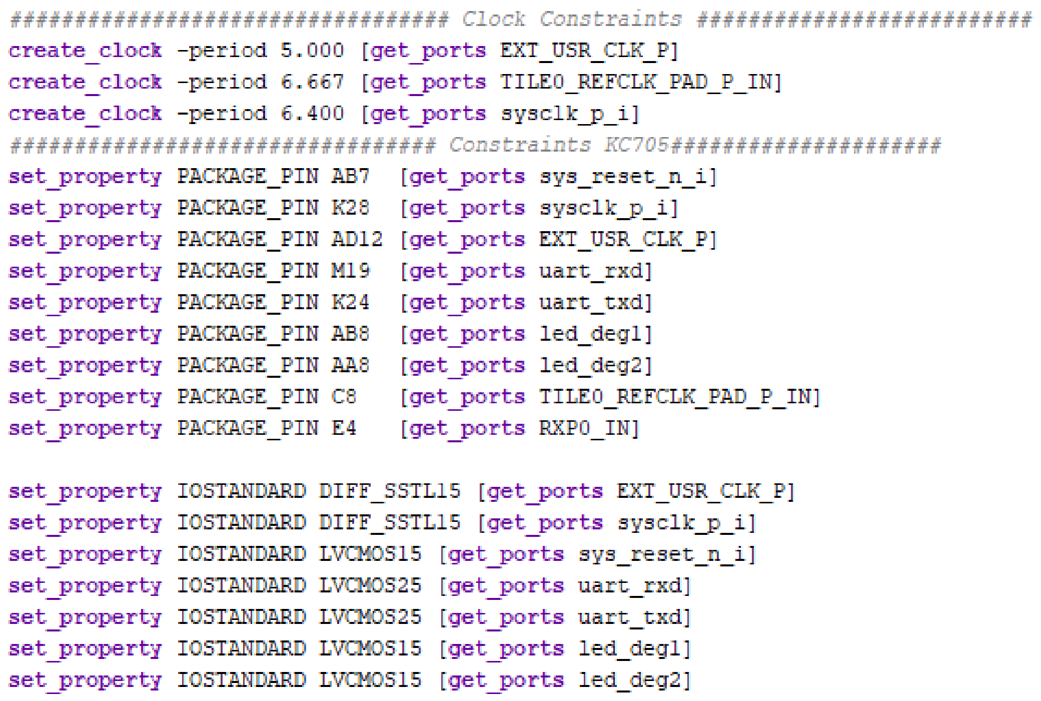
4 Implementation Details

4.1 Clock Domain

In SATA 3.0 Host Controller, the actual system clock (EXT_USR_CLK_P) was 200MHz and it was coming from the Kintex-7 Development kit. The transceiver reference clock (TILE0_REFCLK_PAD_P_IN) was 150MHz and transceiver system clock was 156.25MHz. The gt0_txusrclk2_out clock of GTX transceiver wizard was used to the input clock of all other modules of SATA 3.0 Host Controller design.

4.2 Constraints

Figure 18 shows the pin constraints in the .xdc file of SATA 3.0 Host Controller design



```
################################## Clock Constraints ##########################
create_clock -period 5.000 [get_ports EXT_USR_CLK_P]
create_clock -period 6.667 [get_ports TILE0_REFCLK_PAD_P_IN]
create_clock -period 6.400 [get_ports sysclk_p_i]
################################# Constraints KC705#####################
set_property PACKAGE_PIN AB7  [get_ports sys_reset_n_i]
set_property PACKAGE_PIN K28  [get_ports sysclk_p_i]
set_property PACKAGE_PIN AD12 [get_ports EXT_USR_CLK_P]
set_property PACKAGE_PIN M19  [get_ports uart_rxd]
set_property PACKAGE_PIN K24  [get_ports uart_txd]
set_property PACKAGE_PIN AB8  [get_ports led_deg1]
set_property PACKAGE_PIN AA8  [get_ports led_deg2]
set_property PACKAGE_PIN C8   [get_ports TILE0_REFCLK_PAD_P_IN]
set_property PACKAGE_PIN E4   [get_ports RXP0_IN]

set_property IOSTANDARD DIFF_SSTL15 [get_ports EXT_USR_CLK_P]
set_property IOSTANDARD DIFF_SSTL15 [get_ports sysclk_p_i]
set_property IOSTANDARD LVCMOS15 [get_ports sys_reset_n_i]
set_property IOSTANDARD LVCMOS25 [get_ports uart_rxd]
set_property IOSTANDARD LVCMOS25 [get_ports uart_txd]
set_property IOSTANDARD LVCMOS15 [get_ports led_deg1]
set_property IOSTANDARD LVCMOS15 [get_ports led_deg2]
```

Figure 19: Constraints of SATA 3.0 Host Controller

5 Design modification to be done for Custom Board

- Update the FPGA part number/board according to the FPGA device used
- Update the complete design for the selected FPGA device
- Updated the Transceiver wizard for the selected board.
- Update the pin constraints for SATA interface, clock, reset and UART pins
- Update the clock constraint according to the input clock frequency for the selected FPGA device
- Recompile the design to generate the new binaries and use the XSA file to create the new application project in Vitis

6 Resource Utilization

The table below shows the resource utilization summary for Kintex-7 development kit for SATA 3.0 Host Controller IP.

Table 2 :Resource Utilization for Kintex-7 Development Kit device.

Resource	Utilization	Available
LUT	14055	203800
LUTRAM	1646	64000
FF	18813	407600
BRAM	91	445