Zynq Ultrascale+ 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 Zynq Ultrascale+ MPSoC Development kit.

1.2 Overview

SATA 3.0 Host Controller interfaces the Zynq Ultrascale+ Processor through AXI4-Bus enabling the data transfers between each other. The Zynq Ultrascale+ MPSoC 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 Zynq Ultrascle+ MPSoC Development Board with SATA FMC Daughter card and Intel SSD Device.

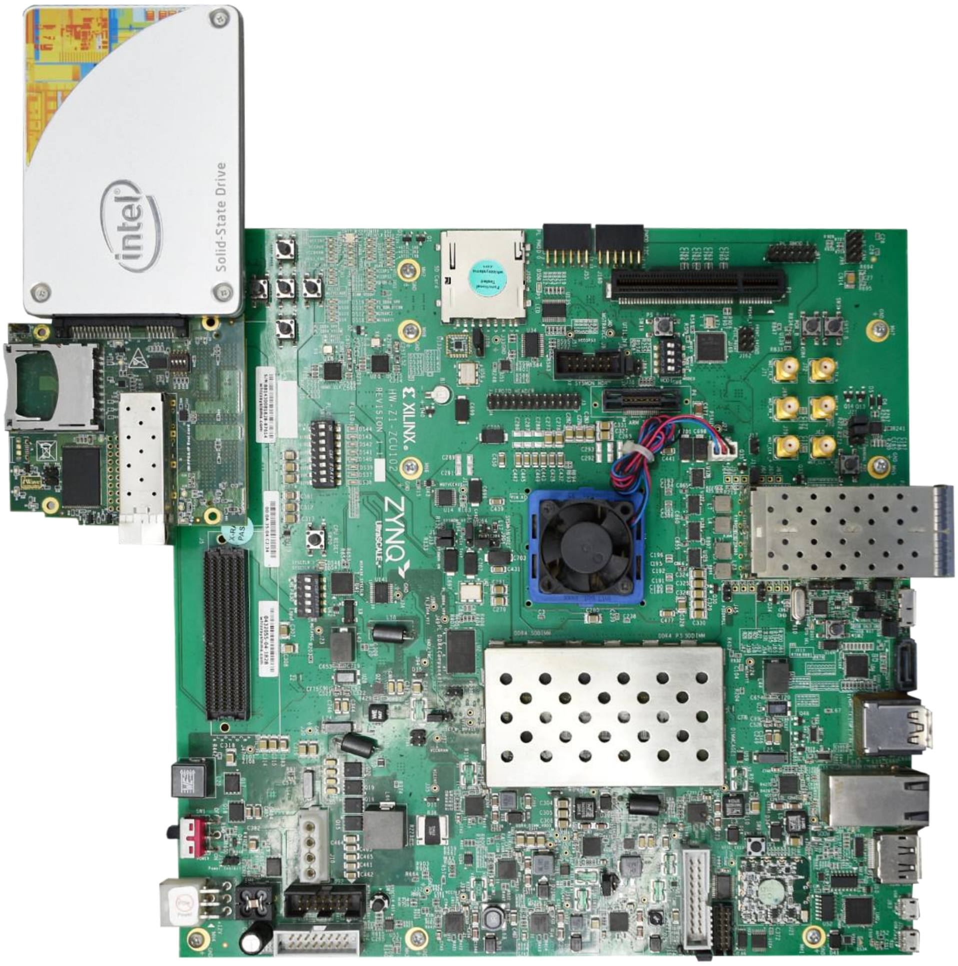


Figure 1: SATA 3.0 Host Controller setup

2.1 SATA FMC Daughter Card

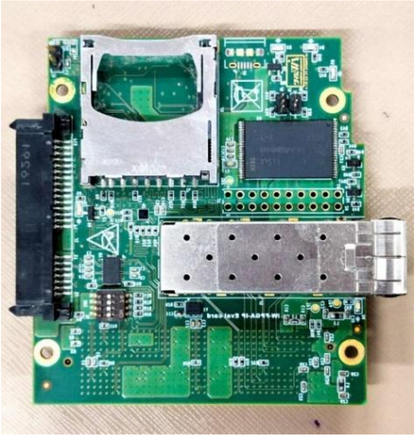


Figure 2: SATA FMC Daughter Card

3 IP Configuration and Instantiation

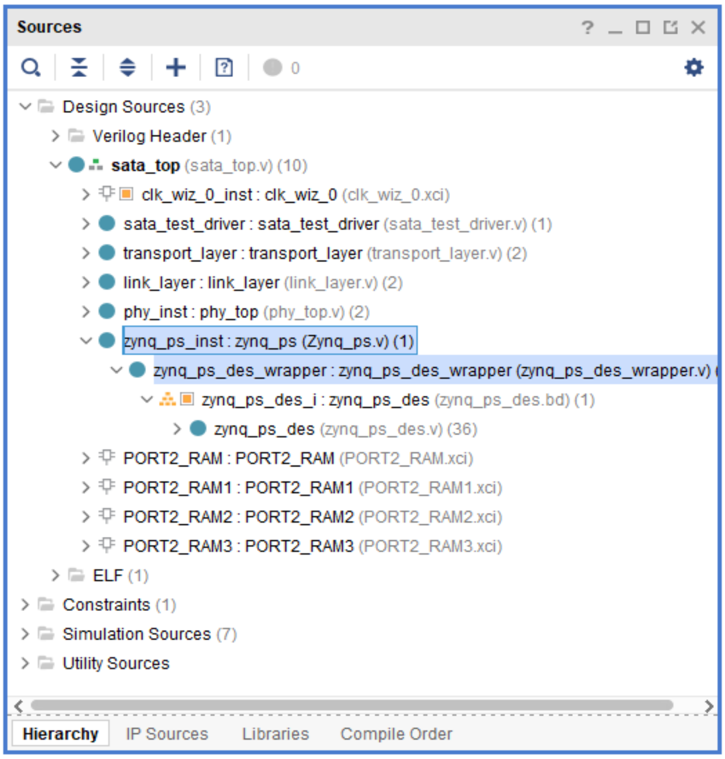
3.1 Example Design

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. **Zynq Ultrascale+ MPSoC processor:** Zynq Ultrascale+ MPSoC 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. **GTH 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 (zynq_ps.v) is instantiated in the design as shown in the below Figure 3.



```
Sources
Design Sources (3)
  Verilog Header (1)
  sata_top (sata_top.v) (10)
    clk_wiz_0_inst : clk_wiz_0 (clk_wiz_0.xci)
    sata_test_driver : sata_test_driver (sata_test_driver.v) (1)
    transport_layer : transport_layer (transport_layer.v) (2)
    link_layer : link_layer (link_layer.v) (2)
    phy_inst : phy_top (phy_top.v) (2)
    zynq_ps_inst : zynq_ps (Zynq_ps.v) (1)
      zynq_ps_des_wrapper : zynq_ps_des_wrapper (zynq_ps_des_wrapper.v) (
        zynq_ps_des_i : zynq_ps_des (zynq_ps_des.bd) (1)
          zynq_ps_des (zynq_ps_des.v) (36)
    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 (7)
Utility Sources
```

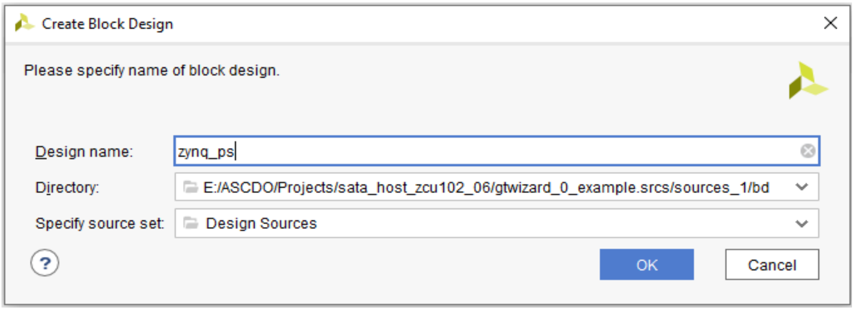
Figure 3: Instantiation Module of SATA 3.0 Host Controller IP

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 (zynq_ps) and directory as E:/ASCDO/Projects/sata_host_zcu102_06/gtwizard_0_example.srcs/sources_1/bd/zynq_ps_des.bd in the below Figure 4.



Create Block Design

Please specify name of block design.

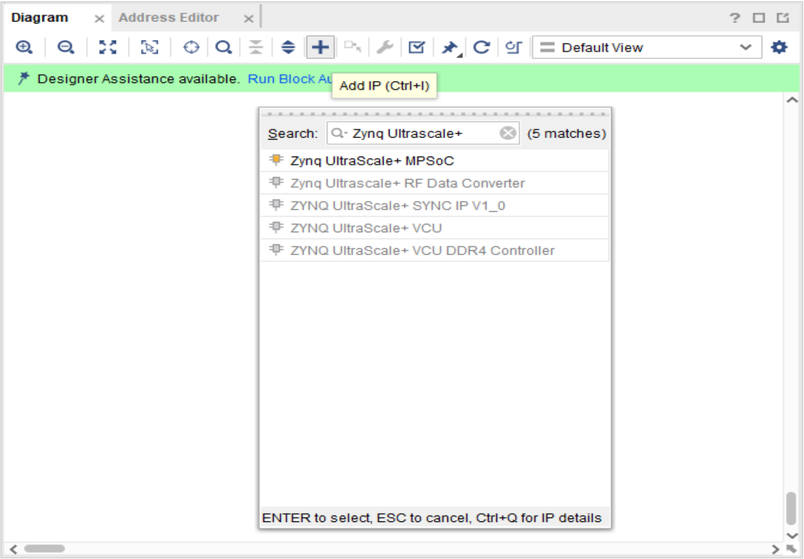
Design name: zynq_ps

Directory: E:/ASCDO/Projects/sata_host_zcu102_06/gtwizard_0_example.srcs/sources_1/bd

Specify source set: Design Sources

Figure 4: SATA 3.0 Host Controller IP 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.



Search: Zynq Ultrascale+ (5 matches)

Zynq UltraScale+ MPSoC
Zynq Ultrascale+ RF Data Converter
ZYNQ UltraScale+ SYNC IP V1_0
ZYNQ UltraScale+ VCU
ZYNQ UltraScale+ VCU DDR4 Controller

ENTER to select, ESC to cancel, Ctrl+Q for IP details

Figure 5: SATA 3.0 Host Controller IP Configuration step 2

- After adding IP like Zynq Ultrascale+, processor system reset, AXI4-interconnect and GPIO, the required GPIO's are connected to the Zynq Ultrascale+ by using AXI4-stream interconnect to the block design as per shown in below Figure 6. Zynq Ultrascale+ was configured by as per the SATA design requirements.

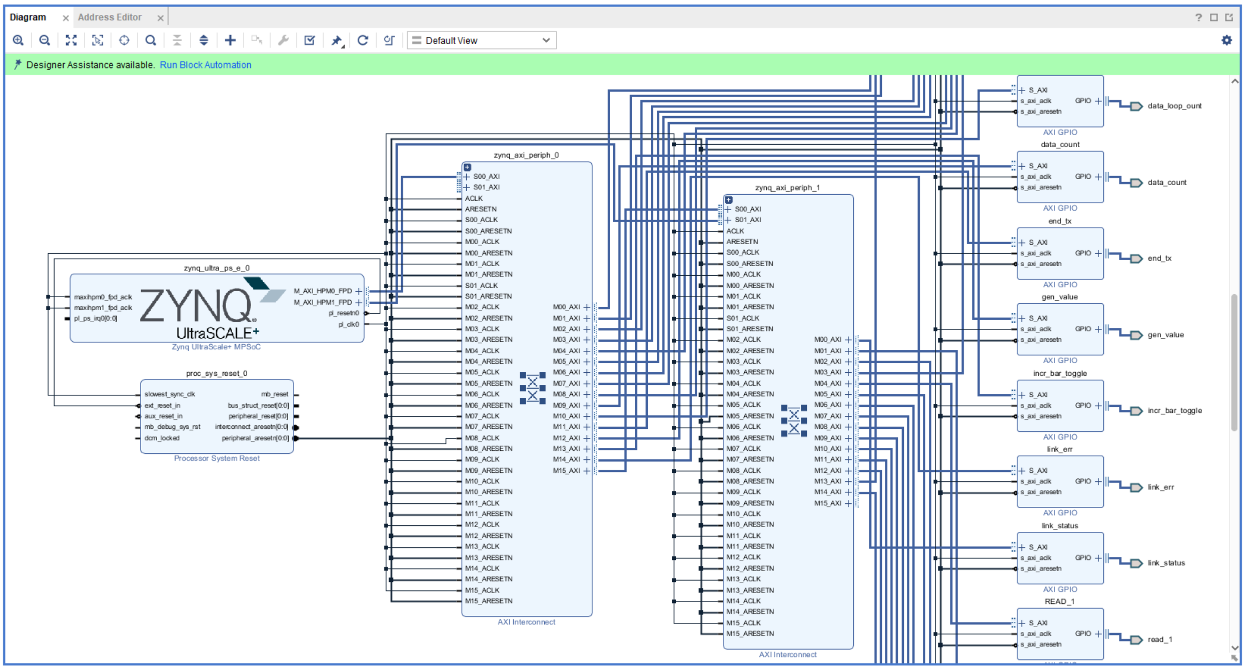
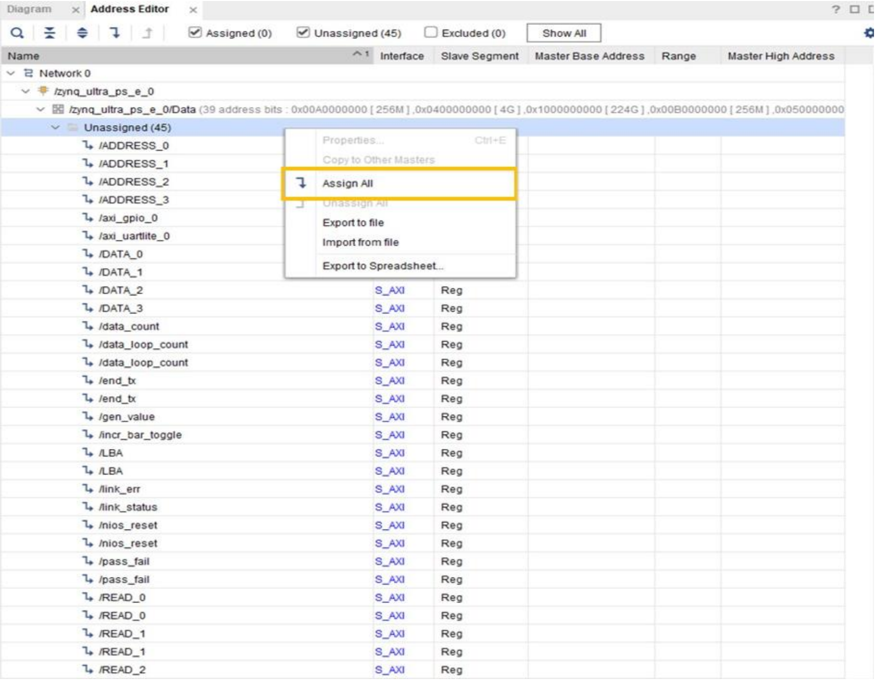


Figure 6: SATA 3.0 Host Controller IP Configuration step 3

- 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.



☑ Assigned (0) ☑ Unassigned (45) □ Excluded (0) Show All

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

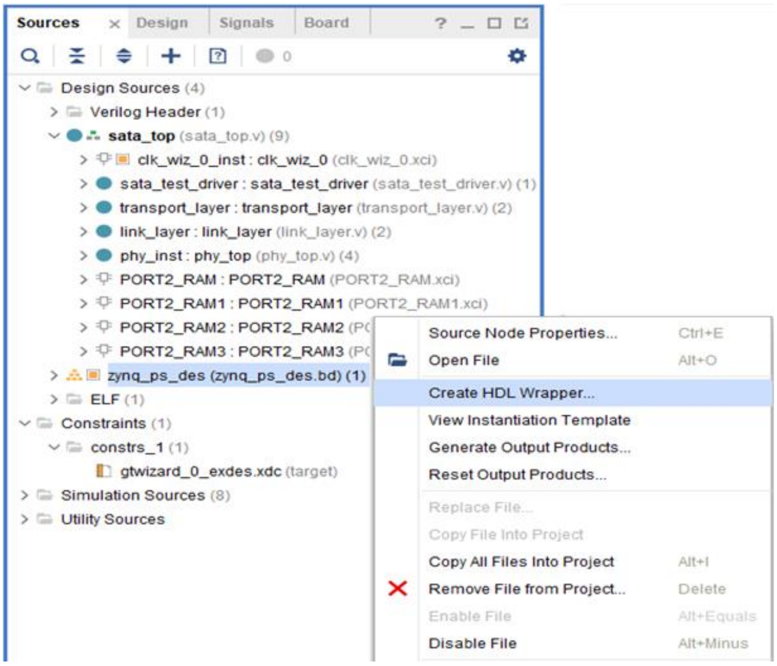
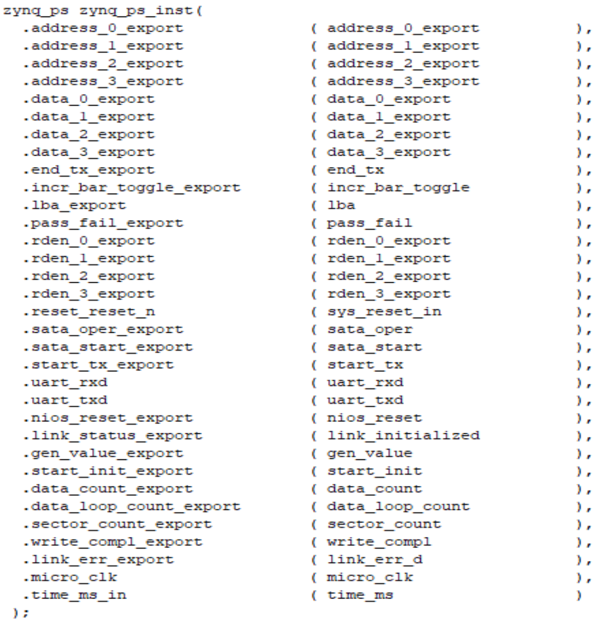

Sources
Design
Signals
Board
0
Design Sources (4)
Verilog Header (1)
sata_top (sata_top.v) (9)
clk_wiz_0_inst : clk_wiz_0 (clk_wiz_0.xci)
sata_test_driver : sata_test_driver (sata_test_driver.v) (1)
transport_layer : transport_layer (transport_layer.v) (2)
link_layer : link_layer (link_layer.v) (2)
phy_inst : phy_top (phy_top.v) (4)
PORT2_RAM : PORT2_RAM (PORT2_RAM.xci)
PORT2_RAM1 : PORT2_RAM1 (PORT2_RAM1.xci)
PORT2_RAM2 : PORT2_RAM2 (P(
PORT2_RAM3 : PORT2_RAM3 (P(
zynq_ps_des (zynq_ps_des.bd) (1)
ELF (1)
Constraints (1)
constrs_1 (1)
gtwizard_0_exdes.xdc (target)
Simulation Sources (8)
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

Figure 8: SATA 3.0 Host Controller IP Configuration step 5

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

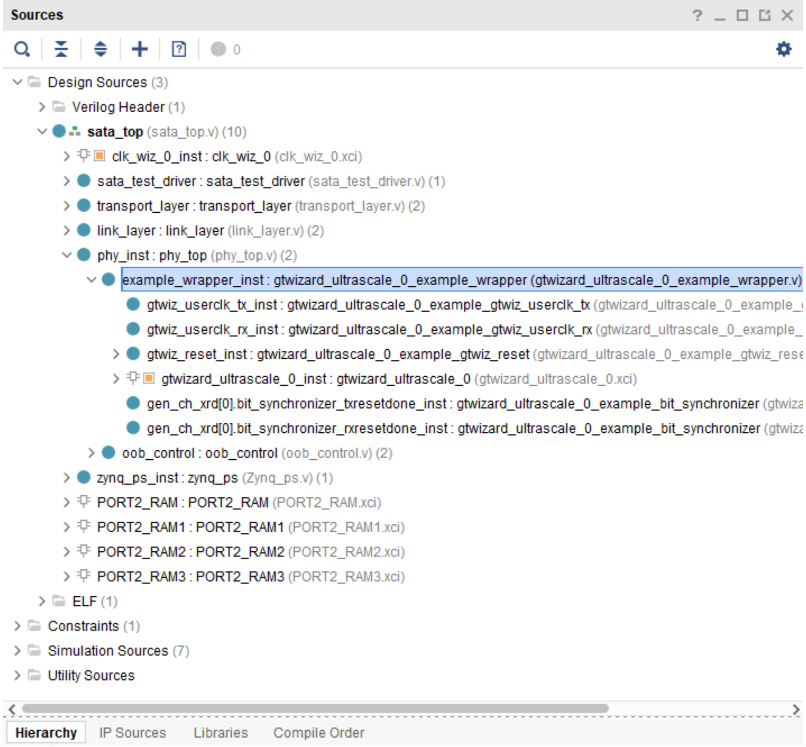


```
zynq_ps zynq_ps_inst(
  .address_0_export               ( address_0_export          ),
  .address_1_export               ( address_1_export          ),
  .address_2_export               ( address_2_export          ),
  .address_3_export               ( address_3_export          ),
  .data_0_export                  ( 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_in              ),
  .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                 ),
  .time_ms_in                     ( time_ms                   )
 );
```

Figure 9: Instantiation of SATA 3.0 Host Controller IP

3.4 GTH transceiver IP Instantiation

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



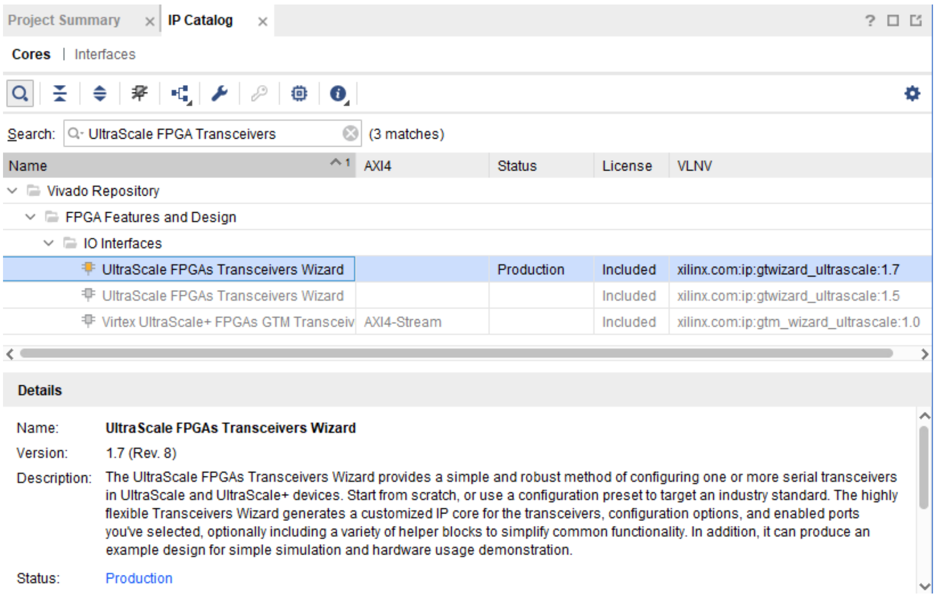
```
Design Sources (3)
  Verilog Header (1)
  sata_top (sata_top.v) (10)
    clk_wiz_0_inst : clk_wiz_0 (clk_wiz_0.xci)
    sata_test_driver : sata_test_driver (sata_test_driver.v) (1)
    transport_layer : transport_layer (transport_layer.v) (2)
    link_layer : link_layer (link_layer.v) (2)
    phy_inst : phy_top (phy_top.v) (2)
      example_wrapper_inst : gtwizard_ultrascale_0_example_wrapper (gtwizard_ultrascale_0_example_wrapper.v)
        gtwiz_userclk_tx_inst : gtwizard_ultrascale_0_example_gtwiz_userclk_tx (gtwizard_ultrascale_0_example_
        gtwiz_userclk_rx_inst : gtwizard_ultrascale_0_example_gtwiz_userclk_rx (gtwizard_ultrascale_0_example_
        gtwiz_reset_inst : gtwizard_ultrascale_0_example_gtwiz_reset (gtwizard_ultrascale_0_example_gtwiz_rese
        gtwizard_ultrascale_0_inst : gtwizard_ultrascale_0 (gtwizard_ultrascale_0.xci)
        gen_ch_xrd[0].bit_synchronizer_txresetdone_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtwiza
        gen_ch_xrd[0].bit_synchronizer_rxresetdone_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtwiza
      oob_control : oob_control (oob_control.v) (2)
    zynq_ps_inst : zynq_ps (Zynq_ps.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 (7)
Utility Sources
```

Figure 10: Instantiation Module of GTH Transceiver wizard

The Figure 10 represents the wrapper module of GTH 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 Ultrascale FPGA Transceiver wizard as shown in below Figure 11.



Search: UltraScale FPGA Transceivers (3 matches)

Name	AXI4	Status	License	VLNV
Vivado Repository				
FPGA Features and Design				
IO Interfaces				
UltraScale FPGAs Transceivers Wizard		Production	Included	xilinx.com:ip:gtwizard_ultrascale:1.7
UltraScale FPGAs Transceivers Wizard			Included	xilinx.com:ip:gtwizard_ultrascale:1.5
Virtex UltraScale+ FPGAs GTM Transceiv	AXI4-Stream		Included	xilinx.com:ip:gtm_wizard_ultrascale:1.0

Details

Name: UltraScale FPGAs Transceivers Wizard

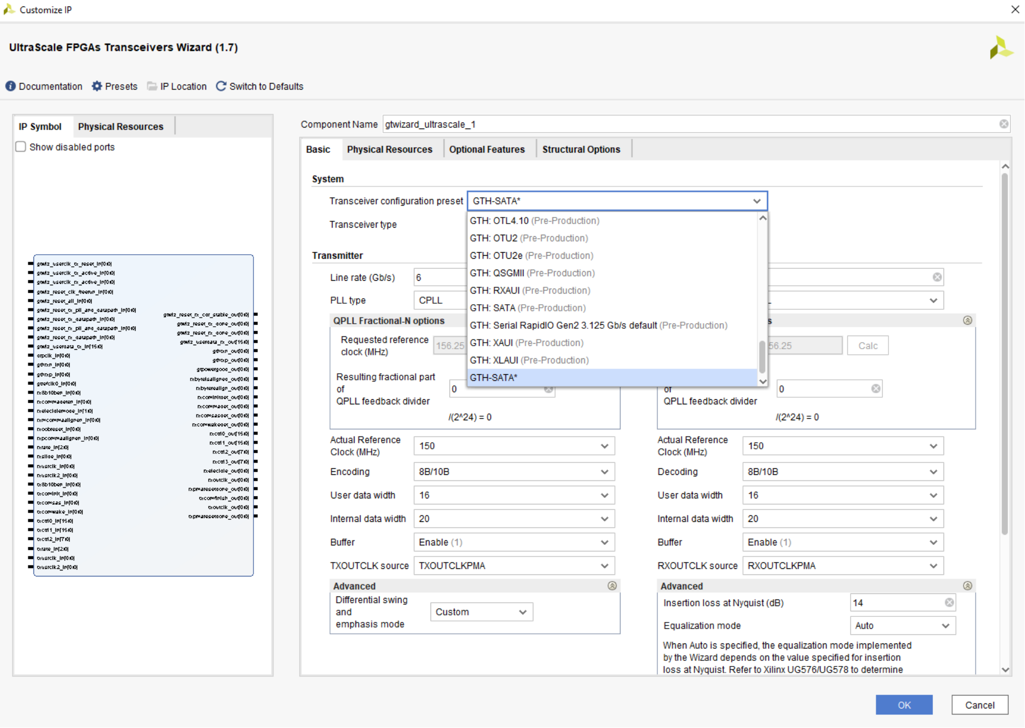
Version: 1.7 (Rev. 8)

Description: The UltraScale FPGAs Transceivers Wizard provides a simple and robust method of configuring one or more serial transceivers in UltraScale and UltraScale+ devices. Start from scratch, or use a configuration preset to target an industry standard. The highly flexible Transceivers Wizard generates a customized IP core for the transceivers, configuration options, and enabled ports you've selected, optionally including a variety of helper blocks to simplify common functionality. In addition, it can produce an example design for simple simulation and hardware usage demonstration.

Status: Production

Figure 11: GTH Transceiver Configuration step 1

- After that Ultrascale FPGA Transceiver wizard (1.7) will be opened and set the transceiver configuration preset as "GTH-SATA". And configure the GTH Transceiver wizard IP as per SATA 3.0 Host Controller design. The Line rate, Actual reference clock and PLL type was selected default by when the configuration preset was selected and rest of the settings should be configured as per the design.



Customize IP

UltraScale FPGAs Transceivers Wizard (1.7)

Documentation
Presets
IP Location
Switch to Defaults
IP Symbol
Physical Resources
□ Show disabled ports
Basic
Physical Resources
Optional Features
Structural Options
GTH: OTL4.10 (Pre-Production)
GTH: OTU2 (Pre-Production)
GTH: OTU2e (Pre-Production)
GTH: QSGMII (Pre-Production)
GTH: RXAUI (Pre-Production)
GTH: SATA (Pre-Production)
GTH: Serial RapidIO Gen2 3.125 Gb/s default (Pre-Production)
GTH: XAUI (Pre-Production)
GTH: XLAUI (Pre-Production)
GTH-SATA*
Calc
OK
Cancel

Component Name: gtwizard_ultrascale_1

System

Transceiver configuration preset: GTH-SATA*

Transceiver type

Transmitter

Line rate (Gb/s): 6

PLL type: CPLL

QPLL Fractional-N options

Requested reference clock (MHz): 156.25

Resulting fractional part of QPLL feedback divider: 0 /(2^24) = 0

Actual Reference Clock (MHz): 150

Encoding: 8B/10B

User data width: 16

Internal data width: 20

Buffer: Enable (1)

TXOUTCLK source: TXOUTCLKPMA

Advanced

Differential swing and emphasis mode: Custom

Receiver

Requested reference clock (MHz): 156.25

Resulting fractional part of QPLL feedback divider: 0 /(2^24) = 0

Actual Reference Clock (MHz): 150

Decoding: 8B/10B

User data width: 16

Internal data width: 20

Buffer: Enable (1)

RXOUTCLK source: RXOUTCLKPMA

Advanced

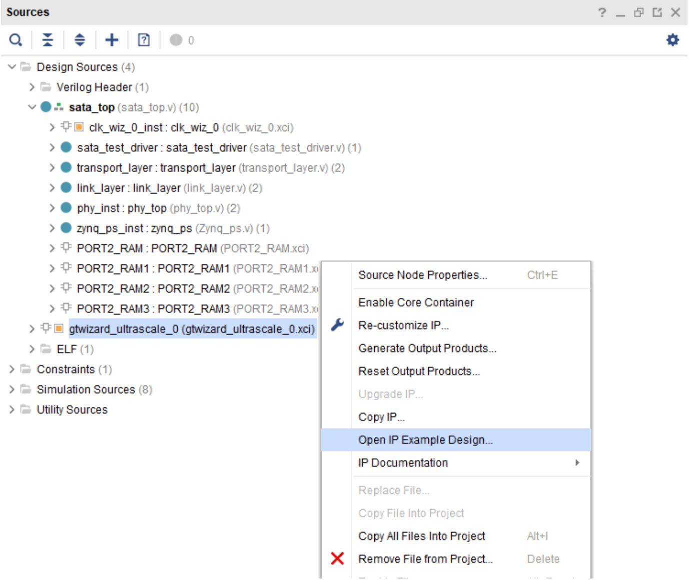
Insertion loss at Nyquist (dB): 14

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

Figure 12: GTH Transceiver Configuration step 2

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



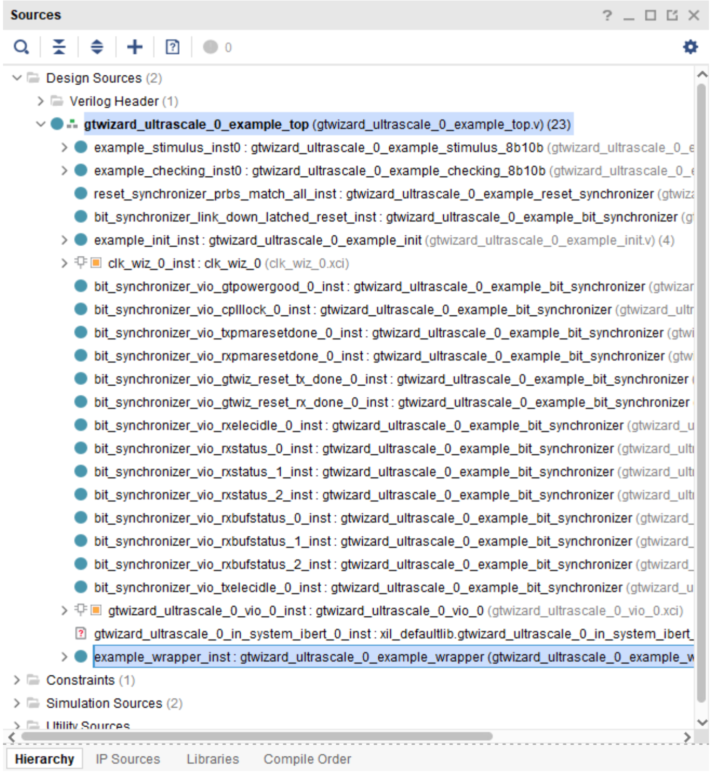
Sources

- Design Sources (4)
  - Verilog Header (1)
  - sata_top (sata_top.v) (10)
    - clk_wiz_0_inst : clk_wiz_0 (clk_wiz_0.xci)
    - sata_test_driver : sata_test_driver (sata_test_driver.v) (1)
    - transport_layer : transport_layer (transport_layer.v) (2)
    - link_layer : link_layer (link_layer.v) (2)
    - phy_inst : phy_top (phy_top.v) (2)
    - zynq_ps_inst : zynq_ps (Zynq_ps.v) (1)
    - PORT2_RAM : PORT2_RAM (PORT2_RAM.xci)
    - PORT2_RAM1 : PORT2_RAM1 (PORT2_RAM1.x
    - PORT2_RAM2 : PORT2_RAM2 (PORT2_RAM2.x
    - PORT2_RAM3 : PORT2_RAM3 (PORT2_RAM3.x
  - gtwizard_ultrascale_0 (gtwizard_ultrascale_0.xci)
  - ELF (1)
- Constraints (1)
- Simulation Sources (8)
- Utility Sources

0
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

Figure 13: GTH Transceiver Configuration step 3

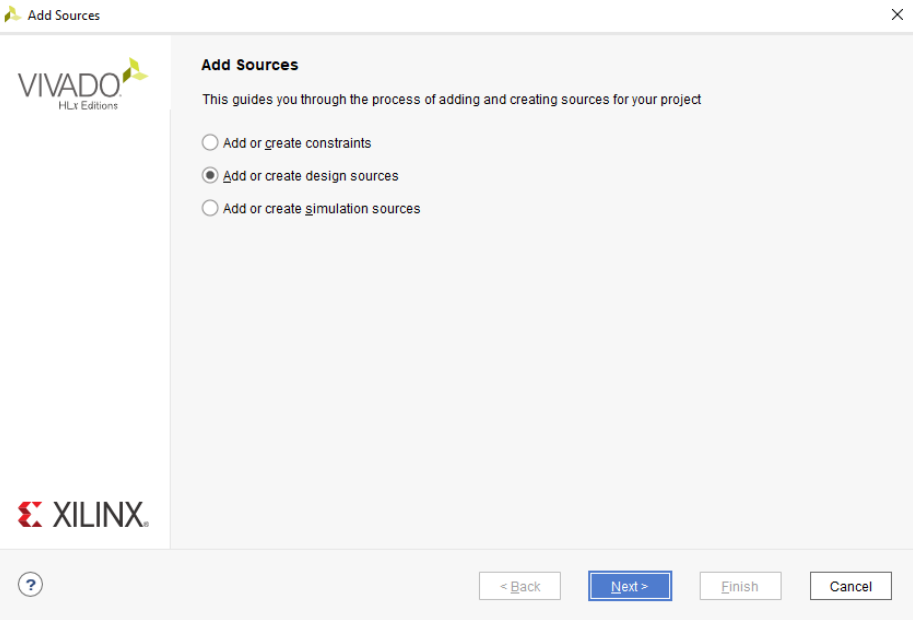
- Give the path as E:/ASCDO/projects/sata_host_zcu102_06/gtwizard for open the example design of GTH Transceiver wizard. The Example design of GTH Transceiver wizard was contains top module and other modules.



```
Design Sources (2)
  Verilog Header (1)
  gtwizard_ultrascale_0_example_top (gtwizard_ultrascale_0_example_top.v) (23)
    example_stimulus_inst0 : gtwizard_ultrascale_0_example_stimulus_8b10b (gtwizard_ultrascale_0_e
    example_checking_inst0 : gtwizard_ultrascale_0_example_checking_8b10b (gtwizard_ultrascale_0_e
    reset_synchronizer_prbs_match_all_inst : gtwizard_ultrascale_0_example_reset_synchronizer (gtwiz
    bit_synchronizer_link_down_latched_reset_inst : gtwizard_ultrascale_0_example_bit_synchronizer (g
    example_init_inst : gtwizard_ultrascale_0_example_init (gtwizard_ultrascale_0_example_init.v) (4)
    clk_wiz_0_inst : clk_wiz_0 (clk_wiz_0.xci)
    bit_synchronizer_vio_gtpowergood_0_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtwizar
    bit_synchronizer_vio_cplllock_0_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtwizard_ultr
    bit_synchronizer_vio_txpmaresetdone_0_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtwi
    bit_synchronizer_vio_rxpmaresetdone_0_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtw
    bit_synchronizer_vio_gtwiz_reset_tx_done_0_inst : gtwizard_ultrascale_0_example_bit_synchronizer (
    bit_synchronizer_vio_gtwiz_reset_rx_done_0_inst : gtwizard_ultrascale_0_example_bit_synchronizer
    bit_synchronizer_vio_rxelecidle_0_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtwizard_u
    bit_synchronizer_vio_rxstatus_0_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtwizard_ult
    bit_synchronizer_vio_rxstatus_1_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtwizard_ult
    bit_synchronizer_vio_rxstatus_2_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtwizard_ult
    bit_synchronizer_vio_rxbufstatus_0_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtwizard_
    bit_synchronizer_vio_rxbufstatus_1_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtwizard_
    bit_synchronizer_vio_rxbufstatus_2_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtwizard_
    bit_synchronizer_vio_txelecidle_0_inst : gtwizard_ultrascale_0_example_bit_synchronizer (gtwizard_u
    gtwizard_ultrascale_0_vio_0_inst : gtwizard_ultrascale_0_vio_0 (gtwizard_ultrascale_0_vio_0.xci)
    gtwizard_ultrascale_0_in_system_ibert_0_inst : xil_defaultlib.gtwizard_ultrascale_0_in_system_ibert_
    example_wrapper_inst : gtwizard_ultrascale_0_example_wrapper (gtwizard_ultrascale_0_example_w
Constraints (1)
Simulation Sources (2)
Utility Sources
```

Figure 14: GTH 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_ultrascale_0_ex → imports.



Add Sources

Add Sources

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

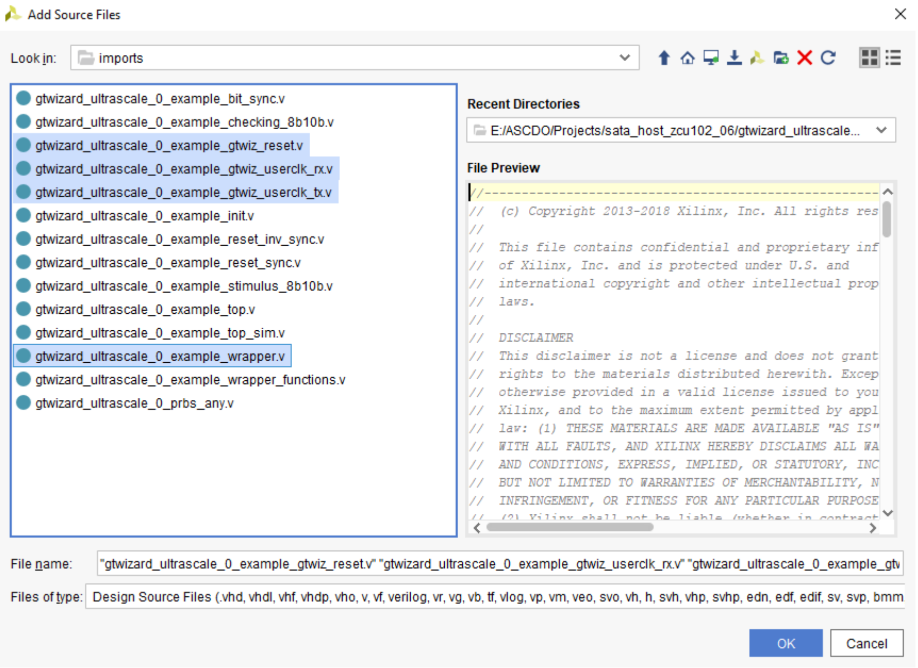
○ Add or create constraints

● Add or create design sources

○ Add or create simulation sources

Figure 15: GTH Transceiver Configuration step 5

- Add the wrapper module and required modules of GTH Transceiver wizard to the SATA 3.0 Host Controller design.



Add Source Files

Look in: imports

```
gtwizard_ultrascale_0_example_bit_sync.v
gtwizard_ultrascale_0_example_checking_8b10b.v
gtwizard_ultrascale_0_example_gtwiz_reset.v
gtwizard_ultrascale_0_example_gtwiz_userclk_rx.v
gtwizard_ultrascale_0_example_gtwiz_userclk_tx.v
gtwizard_ultrascale_0_example_init.v
gtwizard_ultrascale_0_example_reset_inv_sync.v
gtwizard_ultrascale_0_example_reset_sync.v
gtwizard_ultrascale_0_example_stimulus_8b10b.v
gtwizard_ultrascale_0_example_top.v
gtwizard_ultrascale_0_example_top_sim.v
gtwizard_ultrascale_0_example_wrapper.v
gtwizard_ultrascale_0_example_wrapper_functions.v
gtwizard_ultrascale_0_prbs_any.v
```

Recent Directories

E:/ASCDO/Projects/sata_host_zcu102_06/gtwizard_ultrascale...

File Preview

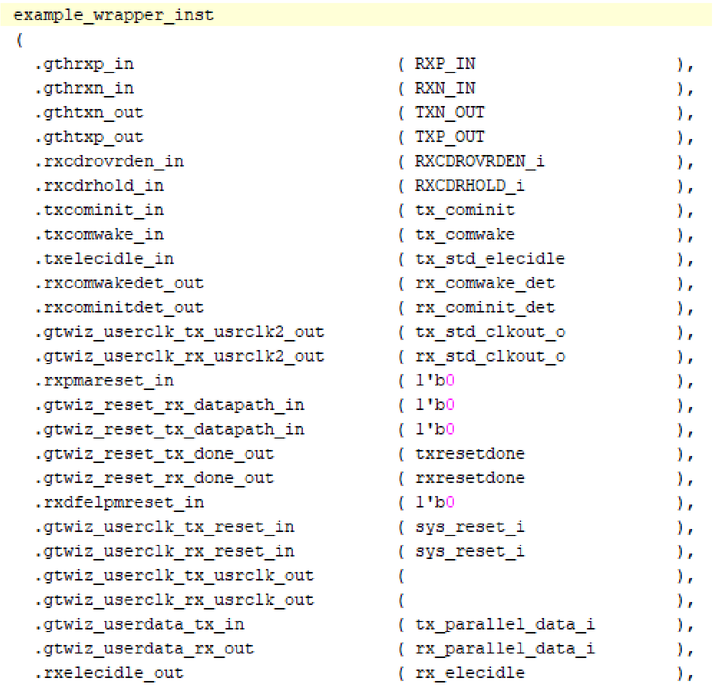
```
//----------------------------------------------------------------
//  (c) Copyright 2013-2018 Xilinx, Inc. All rights res
//
//  This file contains confidential and proprietary inf
//  of Xilinx, Inc. and is protected under U.S. and
//  international copyright and other intellectual prop
//  laws.
//
//  DISCLAIMER
//  This disclaimer is not a license and does not grant
//  rights to the materials distributed herewith. Excep
//  otherwise provided in a valid license issued to you
//  Xilinx, and to the maximum extent permitted by appl
//  law: (1) THESE MATERIALS ARE MADE AVAILABLE "AS IS"
//  WITH ALL FAULTS, AND XILINX HEREBY DISCLAIMS ALL WA
//  AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INC
//  BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, N
//  INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE
//  (2) Xilinx shall not be liable (whether in contract
```

File name: "gtwizard_ultrascale_0_example_gtwiz_reset.v" "gtwizard_ultrascale_0_example_gtwiz_userclk_rx.v" "gtwizard_ultrascale_0_example_gtv

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, sv, svp, bmm

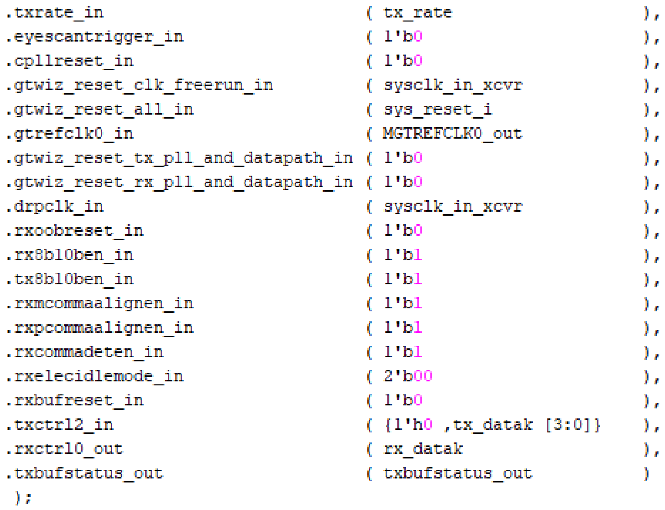
Figure 16: GTH Transceiver Configuration step 6

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



```
example_wrapper_inst
(
  .gthrxp_in                          ( RXP_IN                      ),
  .gthrxn_in                          ( RXN_IN                      ),
  .gthtxn_out                         ( TXN_OUT                     ),
  .gthtxp_out                         ( TXP_OUT                     ),
  .rxcdrovrden_in                     ( RXCDROVRDEN_i               ),
  .rxcdrhold_in                       ( RXCDRHOLD_i                 ),
  .txcominit_in                       ( tx_cominit                  ),
  .txcomwake_in                       ( tx_comwake                  ),
  .txelecidle_in                      ( tx_std_elecidle             ),
  .rxcomwakedet_out                   ( rx_comwake_det              ),
  .rxcominitdet_out                   ( rx_cominit_det              ),
  .gtwiz_userclk_tx_usrclk2_out       ( tx_std_clkout_o             ),
  .gtwiz_userclk_rx_usrclk2_out       ( rx_std_clkout_o             ),
  .rxpmareset_in                      ( 1'b0                        ),
  .gtwiz_reset_rx_datapath_in         ( 1'b0                        ),
  .gtwiz_reset_tx_datapath_in         ( 1'b0                        ),
  .gtwiz_reset_tx_done_out            ( txresetdone                 ),
  .gtwiz_reset_rx_done_out            ( rxresetdone                 ),
  .rxdfelpmreset_in                   ( 1'b0                        ),
  .gtwiz_userclk_tx_reset_in          ( sys_reset_i                 ),
  .gtwiz_userclk_rx_reset_in          ( sys_reset_i                 ),
  .gtwiz_userclk_tx_usrclk_out        (                             ),
  .gtwiz_userclk_rx_usrclk_out        (                             ),
  .gtwiz_userdata_tx_in               ( tx_parallel_data_i          ),
  .gtwiz_userdata_rx_out              ( rx_parallel_data_i          ),
  .rxelecidle_out                     ( rx_elecidle                 ),
```

Figure 17: Instantiation of GTH transceiver wizard (i)



```
 .txrate_in                          ( tx_rate                     ),
 .eyescantrigger_in                  ( 1'b0                        ),
 .cpllreset_in                       ( 1'b0                        ),
 .gtwiz_reset_clk_freerun_in         ( sysclk_in_xcvr              ),
 .gtwiz_reset_all_in                 ( sys_reset_i                 ),
 .gtrefclk0_in                       ( MGTREFCLK0_out              ),
 .gtwiz_reset_tx_pll_and_datapath_in ( 1'b0                        ),
 .gtwiz_reset_rx_pll_and_datapath_in ( 1'b0                        ),
 .drpclk_in                          ( sysclk_in_xcvr              ),
 .rxoobreset_in                      ( 1'b0                        ),
 .rx8b10ben_in                       ( 1'b1                        ),
 .tx8b10ben_in                       ( 1'b1                        ),
 .rxmcommaalignen_in                 ( 1'b1                        ),
 .rxpcommaalignen_in                 ( 1'b1                        ),
 .rxcommadeten_in                    ( 1'b1                        ),
 .rxelecidlemode_in                  ( 2'b00                       ),
 .rxbufreset_in                      ( 1'b0                        ),
 .txctrl2_in                         ( {1'h0 ,tx_datak [3:0]}      ),
 .rxctrl0_out                        ( rx_datak                    ),
 .txbufstatus_out                    ( txbufstatus_out             )
  );
```

Figure 18: Instantiation of GTH transceiver wizard (ii)

- The Constraint file (.xdc) provided in the design is for Xilinx Zynq Ultrascale+ 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.

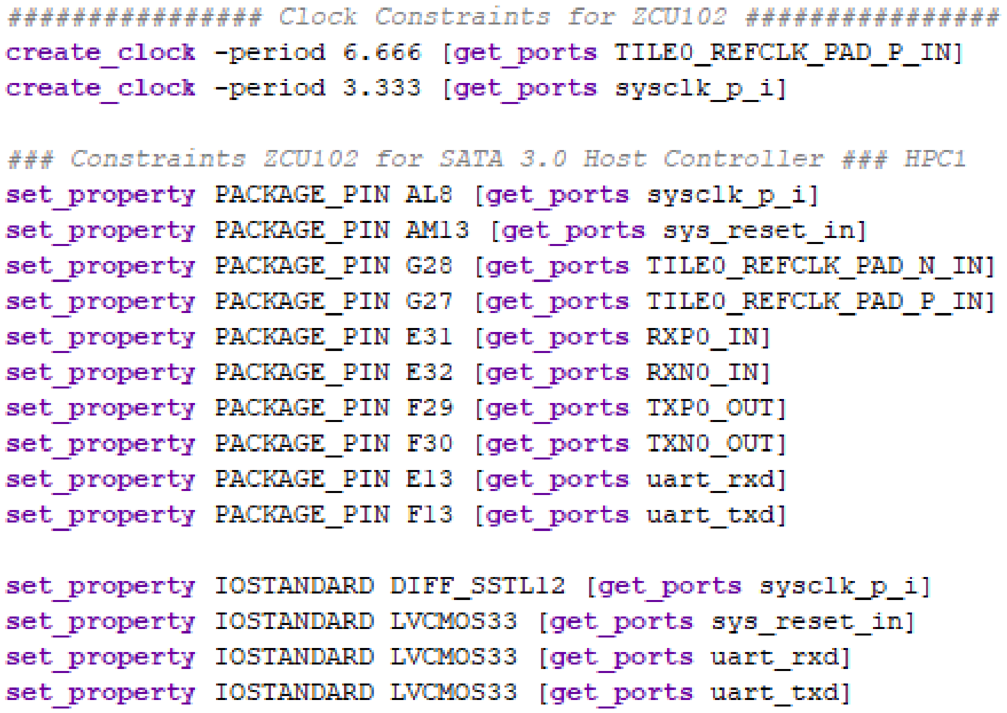
4 Implementation Details

4.1 Clock Domain

In SATA 3.0 Host Controller, the actual system clock was 300MHz and it was coming from the Zynq Ultrascale+ MPSoC Development kit. But the design required only 150MHz clock so clocking wizard to be added for generating a 150MHz output clock. This output clock was given to the GTH transceiver wizard and the gtwiz_userclk_tx_usrclk2_out of GTH transceiver wizard was used in the 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 for ZCU102 ################
create_clock -period 6.666 [get_ports TILE0_REFCLK_PAD_P_IN]
create_clock -period 3.333 [get_ports sysclk_p_i]

### Constraints ZCU102 for SATA 3.0 Host Controller ### HPC1
set_property PACKAGE_PIN AL8 [get_ports sysclk_p_i]
set_property PACKAGE_PIN AM13 [get_ports sys_reset_in]
set_property PACKAGE_PIN G28 [get_ports TILE0_REFCLK_PAD_N_IN]
set_property PACKAGE_PIN G27 [get_ports TILE0_REFCLK_PAD_P_IN]
set_property PACKAGE_PIN E31 [get_ports RXP0_IN]
set_property PACKAGE_PIN E32 [get_ports RXN0_IN]
set_property PACKAGE_PIN F29 [get_ports TXP0_OUT]
set_property PACKAGE_PIN F30 [get_ports TXN0_OUT]
set_property PACKAGE_PIN E13 [get_ports uart_rxd]
set_property PACKAGE_PIN F13 [get_ports uart_txd]

set_property IOSTANDARD DIFF_SSTL12 [get_ports sysclk_p_i]
set_property IOSTANDARD LVCMOS33 [get_ports sys_reset_in]
set_property IOSTANDARD LVCMOS33 [get_ports uart_rxd]
set_property IOSTANDARD LVCMOS33 [get_ports uart_txd]
```

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 Zynq Ultrascale+ MPSoC development kit for SATA 3.0 Host Controller IP.

Table 2 :Resource Utilization for Zynq Ultrascale+ MPSoC development Kit device.

Resource	Utilization	Available
LUT	11592	274080
LUTRAM	1024	144000
FF	14222	548160
BRAM	26.50	912