Stratix 10 L-Tile and H-Tile Transceiver Demo Design (Quartus Prime 17.1)

Author: James Murray

Department: European Application Engineering, Customer Experience Group, Intel PSG

Example Design Scope

This demo design is intended to enable quick and easy transceiver system generation. It can be used to implement a transceiver skeleton designs for transceiver placement and clocking verification. The demo design has ADME enabled so when used with the Quartus Prime Pro Transceiver Toolkit it can be used for hardware transceiver bring-up and signal integrity tuning.

Opening the design and generating IP

- Choose the project whose speed grade (-1, -2, -3) whose speed grade is appropriate for your application. The *-1 project supports up to 28.3Gbps, *-2 supports up to 25.8Gbps, and *-3 up to 17.4Gbps. The project will not compile if an inappropriate device is chosen to support too high a data rate
- Extract the .zip file and open the XCVR_Top.qpf project file in Quartus Prime Pro.
- Ensure that the option to regenerate all IP is selected in Quartus Settings
    - Quartus Settings > IP Settings > Always regenerate IP
- Synthesise the design. The first time you synthesize the design, it will take a long time. This is because the design includes lots of Quartus generated IP that must be generated.
- When synthesis is complete you can switch off the “Always regenerate IP” option to reduce future compile time
    - Quartus Settings > IP Settings > Never regenerate IP
- You are now ready to begin configuring the example designs for your system.

Quick Start Guide

The XCVR_Top.vhd file uses a series of generic groups to configure the transceiver architecture.

The following entry will instantiate duplex wrapper 0 containing four PHYs clocked from a single ATX PLL using the x1 serial clock line. The datarate will be 12.5Gbps. The “H” denotes this is the H-Tile device example design. The signal integrity pattern mux is enabled. The Reference clock from bank L1K, Bottom location will be used.

- NumLHDx0:integer:=4; -- The number of PHYs you want to instantiate
- LHDx0_TxPLL:string:="ATX"; -- The Tx PLL type you want to use.
- LHDx0_SCLK:string:="x1"; -- The Tx serial clock line you want to use
- LHDx0_Rate:string:="12G5"; -- The predefined data rate you want to use
- LHDx0_EnSIPatterns:boolean:= TRUE; -- This enables Step, Impulse, and CLK patterns
- LHDx0_refclk_sel:string:="L1KB"; -- This selects the REFCLK on the device

Similarly, the following will instantiate Tx simplex wrapper 2 containing eight duplex (Tx) PHYs clocked from a single FPLL using the x24 serial clock line. The datarate will be 10.3125Gbps. The “H” denotes this is the H-Tile device example design. The Reference clock from bank R4C, Top location will be used.

- NumLHTx2:integer:=8; -- The number of PHYs you want to instantiate
- LHTx2_TxPLL:string:="FPLL"; -- The Tx PLL type you want to use.

- LHTx2_SCLK:string:="x24"; -- The Tx serial clock line you want to use
- LHTx2_Rate:string:="10G3"; -- The predefined data rate you want to use
- LHTx2_EnSIPatterns:boolean:= TRUE; -- This enables Step, Impulse, and CLK patterns
- LHTx2_refclk_sel:string:="R4CT"; -- This selects the REFCLK on the device

After configuring the generics, run Quartus Prime Pro Analysis and Synthesis, pin out the design, and then fully compile. Standard Stratix 10 device fitting rules apply.

Comprehensive Guide

Example Design Nomenclature

Throughout the demo design, the following nomenclature is used.

- "LH" denotes a combined L-Tile ES2 or H-Tile transceiver IP in Quartus versions 17.1ir2 or later.
- "Dx", "Tx", "Rx", "GT" denotes a Duplex, Tx simplex, Rx simplex, or duplex GXT channel type.
- "PHY", "ATX", "FPLL", "CMU" denotes a PHY, ATX PLL, FPLL, or CMU Tx PLL.
- "9G8" or "12G5", etc, denotes data rates of 9.8Gbps or 12.5Gbps. Other data rates are included in the designs

For example:

- LH_PHYDx10G3_x1.ip – An L or H-Tile transceiver PHY, configured for duplex (Dx) operation at a datarate of 10.3125Gbp. The PHY is configured for one channel.
- LH_PHYRx12G5_x1.ip – An L or H-Tile transceiver PHY, configured for Rx simplex operation at a datarate of 12.5Gbp. The PHY is configured for one channel.
- LH_ATX17G4.ip – An L or H-Tile ATX PLL configured for 17.4Gbps operation.

This nomenclature extends to the VHDL files too.

- LH_Dx[n]_Wrap.vhd – A VHDL wrapper file for an L-Tile ES2 and H-Tile Duplex channel
- LH_Tx[n]_Wrap.vhd – A VHDL wrapper file for an an L-Tile ES2 and H-Tile Simplex Tx channel
- LH_GT[n]_Wrap.vhd – A VHDL wrapper file for an L-Tile ES2 and H-Tile Duplex GXT channel

Configuring the XCVR_Top.vhd File Generics

The XCVR_Top.vhd file includes groups of VHDL generics that make it easy to configure the PHY, Tx PLL, and serial clock line requirements, REFCLK choice, and datarate for your application,

The generics are listed in groups. Each group instantiates a wrapper that contains transceiver PHYs, reset controllers, and a single Tx PLL. Rx Simplex wrappers do not use a Tx PLL. The wrapper also includes a data mux to select useful signal integrity patterns.

Each group can include any number of channels. The following image shows two groups of generics for duplex channels.

```
NumLHDx0:integer:=4;        -- Total number of Channels
LHDx0_TxPLL:string:="FPLL";    -- Enter "ATX, "CMU" or "
LHDx0_SCLK:string:="x1";       -- Enter "x1" or x24. > 6
LHDx0_Rate:string:="10G3";     -- Enter "9G8, "10G1", "1
LHDx0_EnSIPatterns:boolean:= TRUE;  -- Enter TRUE of FA
LHDx0_refclk_sel:string:="R4FB"; -- Enter REFCLK choice

NumLHDx1:integer:=24;          -- Total number of Channe
LHDx1_TxPLL:string:="ATX";     -- Enter "ATX, "CMU" or "
LHDx1_SCLK:string:="xN";       -- Enter "x1" or x24. > 6
LHDx1_Rate:string:="10G1";     -- Enter "9G8, "10G1", "1
LHDx1_EnSIPatterns:boolean:= TRUE;  -- Enter TRUE of FA
LHDx1_refclk_sel:string:="L1DT"; -- Enter REFCLK choice
```

The demo design includes the following quantities of generic groups:

- Eight groups of duplex channel types
- Four groups of Tx simplex channel types
- Four groups of Rx simplex channel types
- Eight groups of GXT channel types

Selecting the number of PHYs

You can use as many or as few of the groups in the design as you like. This is done by entering the appropriate integer in the NumLH[D,T,R]x[n] generic string. For example:

- Entering NumLHDx0:integer:=1, instantiates Duplex wrapper 0 containing a single channel PHY and Reset Controller IP. A single Tx PLL IP will be instantiated.
- Entering NumLHDx3:integer:=8, instantiates Duplex wrapper 3 containing eight single channel PHYs and Reset Controller IPs. A single Tx PLL IP will be instantiated for all channels.
- Entering “0” disables that group and no components are instantiated.

Selecting the Tx PLL Type

Entering “ATX”, “FPLL”, or “CMU” in the LH[D,T,R]x[n]_TxPLL generic string selects which PLL type will be used. For example:

- LHTx2_TxPLL:string:="FPLL"; -- selects an fPLL for Tx simplex group 2

or

- LHDx7_TxPLL:string:="ATX"; -- selects an ATX PLL for Duplex group 7

The string entry is case sensitive.

Selecting the serial clock lines

Entering “x1” or “x24” in the LH[D,T,R]x[n]_SCLK generic selects whether the x1 clock line or x24 clock lines are used to clock the PHY from the Tx PLL. For example:

- LHDx0_SCLK:string:="x1"; -- selects the x1 line

or

- LHDx1_SCLK:string:="x24"; -- selects the x6 or x24 line via an mcgb.

The string entry is case sensitive.

Rx Simplex PHYs do not require a Tx PLL so this parameter is ignored for Rx Simplex groups.

Selecting the Data Rate

The demo design includes some predefined data rates that are selected by entering the appropriate string for the LH[D,T,R]x[n]_Rate generic. For example

LHDx0_Rate:string:="10G3" -- a 10.3Gbps PHY and Tx PLL are instantiated

Predefined data rates include:

- "9G8" - 9.8304Gbps GX Channel clocked from a 307.2MHz reference clock
- "10G1" - 10137.6Gbps GX Channel clocked from a 307.2MHz reference clock
- "10G3" - 10312.5Gbps GX Channel clocked from a 644.53125MHz reference clock
- "11G3" - 11.3Gbps GX Channel clocked from a 706.25MHz reference clock
- "12G5" - 12.5Gbps GX Channel clocked from a 625MHz reference clock
- "17G4" - 17.4Gbps GX Channel clocked from a 725MHz reference clock
- "24G3" – 24.33024Gbps GXT Channel clocked from a 184.32MHz reference clock
- "25G8" – 25.78Gbps GXT Channel clocked from a 644.53125MHz reference clock
- "28G0" – 28.05Gbps GXT Channel clocked from a 701.25MHz reference clock
- "28G3" – 28.25Gbps GXT Channel clocked from a 706.25MHz reference clock

Note: GXT channels are not supported in the -3 transceiver speed grade version of the design.

If you need a different frequency REFCLK to those fixed datarates listed above, you can open the PHY and Tx PLL IP and change it.

User Data Rates

The example designs include four user PHYs and Tx PLL IP that are selected by entering USR[n] in the LH[D,T,R]x[n]_Rate string. For example:

- LHDx0_Rate:string:="USR1"; -- The predefined data rate you want to use

The string entry is case sensitive.

When using the "USR[3:0]" option, you should open the appropriate PHY IP and Tx PLL and edit these for your own required datarate and reference clock frequency.

For example, to use a duplex USR1 PHY clocked from an ATX PLL you must open and edit the datarate and reference clock frequency in the following IP files located in <project location>\Sources\IP

- LH_ATXUSR1.ip
- LH_PHYDxUSR1_x1.ip

Similarly, to use a Tx simplex USR2 PHY clocked from an FPLL you must open and edit the datarate and reference clock frequency in the following IP files located in <project location>\Sources\IP

- LH_FPLLUSR2.ip
- LH_PHYTxUSR2_x1.ip

There are no USR GXT options with this design.

Selecting the Reference Clock

The demo design includes a full complement of REFCLK pins for a 4 tile device. The REFCLK signals are assigned to pins in the project.

Entering the appropriate transceiver bank and location string into the "LH[D,T,R]x[n]_refclk_sel" generic string selects which physical REFCLK your PHY and Tx PLL will be connected to.

- LHTx0_refclk_sel:string:="L1KB"; -- Enter REFCLK choice here.

Examples of the string nomenclature are:

- REFCLK_GXB**L1K**_CH**B** = The **B**ottom REFCLK of transceiver bank **L1K =** "L1KB"
- REFCLK_GXB**R4C**_CH**T** = The **T**op REFCLK of transceiver bank **R4C =** "R4CT"

The string entry is case sensitive.

Instead of implementing a mux which would be synthesized in the FPGA fabric that would degrade Tx jitter performance, the REFCLK is chosen by way of generate statements, and the connection made at compile time.

If you use a Stratix 10 device that has a different pinout to this example design, you should ensure the top level REFCLK pins are assigned correctly for your device.

Using the Signal Integrity Pattern Mux

To enable the signal integrity pattern selector, set the "LH[D,T,R]x[n]_EnSIPatterns" Boolean generic to TRUE. For example

LHDx0_EnSIPatterns:boolean:= TRUE;

This feature enables step, single bit, and clock patterns that can be used to assess channel performance. It can be used to complement the Transceiver Toolkit PRBS patterns and tune channel signal integrity. The Mux is controlled by an In System Source and Probes IP (ISSP).

When using this feature with many channels in a design, the number of ISSP IP instances can become too numerous and cumbersome to use. Consider setting the Boolean generic to FALSE for any channels that you don't need the pattern generator for.

The patterns are selected using a 4-bit In System Source and Probes (ISSP) IP source. The encoding is shown below.

- b"0000" transmits x"5555,5555,5555,5555" – Fast clock pattern
- b"0001" transmits x"3333,3333,3333,3333" -- x33 clock pattern
- b"0010" transmits x"F0F0,F0F0,F0F0,F0F0" -- F0 clock pattern
- b"0011" transmits x"FF00,FF00,FF00,FF00" -- FF00 clock pattern
- b"0100" transmits x"FFFF,0000,FFFF,0000" -- FFFF0000 clock pattern
- b"0101" transmits x"FFFF,FFFF,0000,0000" – Step pattern
- b"0110" transmits x"8000,0000,0000,0000" – Single bit (impulse) pattern
- b"0111" transmits x"8000,8000,8000,8000" – Multiple impulse pattern
- b"1000" transmits x"8080,8080,8080,8080" – Multiple fast impulse pattern
- b"1001" transmits x"EFFF,FFFF,FFFF,FFFF" – Negative impulse pattern
- b"1010" transmits x"0000,1000,00FF,FFFF" -- Consecutive Step & Impulse pattern
- b"1011" transmits x"FFFF,EFFF,FF00,0000" – Negative Consecutive Step & Impulse pattern

- b"1100" transmits x"FFFF,5555,0000,AAAA" – Alternating fast slow clock pattern
- b"1101" transmits a user pattern defined from a 64 bit In System Source and Probes IP
- b"1110" Connects Rx parallel dataout to Tx Parallel data in using a DC FIFO.
- b"1111" transmits Traffic_Tx_parallel_datain

Configuring the Mux using (ISSP)

When the device is configured, open the 4 bit ISSP IP in Quartus Prime and select one of the binary select codes from above.

Configuring the User Pattern using (ISSP)

Open the 64 bit ISSP IP in Quartus Prime and configure the data for the pattern you want to transmit. For example, transmitting a step pattern with a random single bit impulse could be implemented with the following code.

x"FFFF,FFFF,0001,0000"

Using the Simplex Channels

The simplex channels are configured and instantiated in the same way as the duplex channels. However, the simplex wrappers expose the PHY Avalon Memory Mapped (AVMM) interface ports at the top level of the wrapper file. This is so that you can easily common up the following Tx and Rx Simplex PHY IP AVMM signals in the XCVR_Top.vhd file required when merging into a physical duplex channel.

reconfig_write
reconfig_read
reconfig_address
reconfig_writedata

In addition to connecting the above signals between Tx and Rx interfaces, you should follow the merging guidelines listed in the Stratix 10 Transceiver PHY IP user guide.

Using GXT channels in the H-Tile Example design

The -1 and -2 versions of the demo design supports duplex GXT transceivers. They use the “Main” and “Adjacent” ATX PLL clocking architecture.

GXT capable channels are in physical locations 0, 1, 3, & 4 of a transceiver bank. Channels configured for GXT mode must be driven by the ATX PLL directly adjacent to it. Within a bank, the bottom ATX PLL drives channels 0 & 1, and the top ATX PLL drives channels 3 & 4.

Driving more than 2 channels requires ATX PLL cascading using the dedicated GXT cascade lines.

A physical bottom ATX PLL can cascade to the physical top ATX PLL in that bank, and the physical top ATX PLL in the bank below.

A physical top ATX PLL can cascade to the physical bottom ATX PLL in that bank, and the physical bottom ATX PLL in the bank above.

The cascaded ATX PLLs in GXT mode use the notation of a logical main, above, and below ATX PLL.

The main ATX PLL is always the source PLL that you cascade from. The above and below ATX PLLs are always the ATX PLLs that you cascade to.

The above ATX PLL is always physically above the main ATX PLL, and the below ATX PLL is always physically below the main ATX PLLs.

The demo design includes eight GXT generic groups that can be configured for up to 6 channels. The following images shows groups 0 and 1.

```
NumLHGT0abv:integer:=0; -- Total number of GXT channels clocked from the above ATX. The at
NumLHGT0main:integer:=0;    -- Total number of GXT channels clocked from the Main ATX PLL.
NumLHGT0blw:integer:=0; -- Total number of GXT channels clocked from the below ATX. The be
LHGT0_refclk_sel:string:="R4DB";
LHGT0_Rate:string:= "28G0";    -- Enter "24G3, "25G8", "28G0"

NumLHGT1abv:integer:=2; -- Total number of GXT channels clocked from the above ATX. The at
NumLHGT1main:integer:=2;    -- Total number of GXT channels clocked from the Main ATX PLL.
NumLHGT1blw:integer:=0; -- Total number of GXT channels clocked from the below ATX. The be
LHGT1_refclk_sel:string:="R4EB";
LHGT1_Rate:string:= "25G8";    -- Enter "24G3, "25G8", "28G0"
```

Any combination of main, above, or below channels are allowed in the design if the main group has at least one channel. The main group must be populated first because this group is clocked from the source ATX PLL.

Example for one GXT channel

- NumLHGTCH0abv:integer:=0; -- Number of GXT channels clocked from the above ATX PLL.
- NumLHGTCH0ctr:integer:=1; -- Number of GXT channels clocked from the main ATX PLL.
- NumLHGTCH0blw:integer:=0; -- Number of GXT channels clocked from the below ATX PLL.

Example for four GXT channels

- NumLHGTCH0abv:integer:=0; -- Number of GXT channels clocked from the above ATX PLL.
- NumLHGTCH0ctr:integer:=2; -- Number of GXT channels clocked from the main ATX PLL.
- NumLHGTCH0blw:integer:=2; -- Number of GXT channels clocked from the below ATX PLL.

The following configuration would be illegal because no main channels are used.

- NumLHGTCH0abv:integer:=0; -- Number of GXT channels clocked from the above ATX PLL.
- NumLHGTCH0ctr:integer:=0; -- Number of GXT channels clocked from the main ATX PLL.
- NumLHGTCH0blw:integer:=2; -- Number of GXT channels clocked from the below ATX PLL.

To disable the group completely, enter 0 for main, above and below channels.

When using the GXT clock lines you should use the HQ reference clock lines. You can use the following assignment to do this.

- set_instance_assignment -name XCVR_USE_HQ_REFCLK ON -to pin_name

Timing

The design has minimal constraints for timing. They include a constraint for the PHY and Tx PLL AVMM clocks, and some example constraints for REFCLKs too.

```
#**************************************************************
# Create Clock
#**************************************************************
create_clock -period 125MHz -name AVMM100 [get_ports {clk_100[0]}]

# Banks 1C/1D/1E/1F
create_clock -period 644.53125MHz -name REF644M_L1CT [get_ports {REFCLK_L1CT}]
create_clock -period 644.53125MHz -name REF644M_L1DT [get_ports {REFCLK_L1DT}]
create_clock -period 644.53125MHz -name REF644M_L1DB [get_ports {REFCLK_L1DB}]
create_clock -period 614.40000MHz -name REF614M_L1ET [get_ports {REFCLK_L1ET}]
create_clock -period 614.40000MHz -name REF614M_L1EB [get_ports {REFCLK_L1EB}]
create_clock -period 625.00000MHz -name REF625M_L1FT [get_ports {REFCLK_L1FT}]

# Banks 1K/1L/1M/1N
create_clock -period 644.53125MHz -name REF644M_L1KT [get_ports {REFCLK_L1KT}]
create_clock -period 614.40000MHz -name REF614M_L1KB [get_ports {REFCLK_L1KB}]
create_clock -period 625.00000MHz -name REF625M_L1LT [get_ports {REFCLK_L1LT}]
create_clock -period 644.53125MHz -name REF644M_L1NT [get_ports {REFCLK_L1NT}]

# Banks 4C/4D/4E/4F
create_clock -period 706.25000MHz -name REF706M_R4CT [get_ports {REFCLK_R4CT}]
create_clock -period 706.25000MHz -name REF706M_R4DT [get_ports {REFCLK_R4DT}]
create_clock -period 644.53125MHz -name REF644M_R4DB [get_ports {REFCLK_R4DB}]
create_clock -period 706.25000MHz -name REF706M_R4ET [get_ports {REFCLK_R4ET}]
create_clock -period 644.53125MHz -name REF644M_R4EB [get_ports {REFCLK_R4EB}]
create_clock -period 706.25000MHz -name REF706M_R4FT [get_ports {REFCLK_R4FT}]
create_clock -period 644.53125MHz -name REF644M_R4FB [get_ports {REFCLK_R4FB}]

# Banks 4K/4L/4M/4N
create_clock -period 644.53125MHz -name REF644M_R4LB [get_ports {REFCLK_R4LB}]
create_clock -period 706.25000MHz -name REF706M_R4NB [get_ports {REFCLK_R4NB}]
```

If you fail to constrain your design, the channels may not appear in the Transceiver Toolkit or you may see other typical timing related problems.

Transceiver Toolkit

The design has ADME enabled so should work with the Quartus Transceiver Toolkit. All PRBS generators, checkers, and PMA settings will work with this design.

To start the Transceiver Toolkit select it from the Tools Menu in Quartus Prime (with the device programmed loaded) using the appropriate version of the Quartus Prime Pro software. When the Transceiver Toolkit is open, load the SOF in the System Console. You should then see your channels and can control them using the Transceiver Toolkit GUI.

Stratix 10 Helper TCL script

Written by Peter Schepers, this Stratix 10 Helper TCL script is also compatible with this design. This tcl file contains several procedures to do additional configuration and status updates for Stratix 10 devices.

http://www.alterawiki.com/wiki/File:ttk_helper_s10.tcl

When the Transceiver Toolkit System Console is open, you can source the ttk_helper_s10.tcl script located in the scripts folder. This script contains several useful routines for advanced debug and setup. For instance, calibration, full register access, etc. See Appendix A for a list of commands.

Adaption script

Written by Peter Schepers, the New_adapt_values.tcl TCL script lets you set maximum initial values for the Rx PMA adaption that is usually required for lossy channels.

To run this script, you must first source the ttk_helper_s10.tcl script.

Trouble Shooting

The design uses standard Quartus Stratix 10 fitting rules. A design failing to fit may be due to a violation of the clocking rules detailed in the Stratix 10 L-Tile and H-Tile PHY IP user guides.

Generic String Entry is Case and Input sensitive

The generic string entries are case and entry sensitive. Failing to enter the exact required string can cause the following problems

Entering “X24”, or “xN”, or “XN” instead of “x24” will cause the mcgb_serial_clk connection between the Tx PLL and PHY to be missing. The design may not compile.

Entering “X1” instead of “x1” will cause the tx_serial_clkclock connection between the Tx PLL and PHY to be missing. The design may not compile.

Entering “atx” instead of “ATX” will cause the Tx PLL to not be instantiated, though the design may compile.

Entering “fpll” or “fPLL” instead of “FPLL” will cause the Tx PLL to not be instantiated, though the design may compile.

Incorrectly entering the Rate string will cause the PHY and Tx PLL to not be instantiated. For example, entering “10g3” or “10G” instead of “10G3”. The design may compile.

Other case, or input sensitivity problems may occur. Refer to the RTL comments for correct entry.

Enter Generics According to Device Clocking Rules

Not all generic combinations are valid. For example, due to device clocking restrictions, a x1 serial clock line cannot clock greater than 6 channels. Similarly, a CMU does not have access to the x24 clock lines. You should select your generics according to the device clocking rules.

Location Constraints

When adding location constraints to the channels in your design, also consider the location of the Tx PLLs. For example, you may need to lock these down to avoid x24 clock line contention or to respect PLL spacing rules.

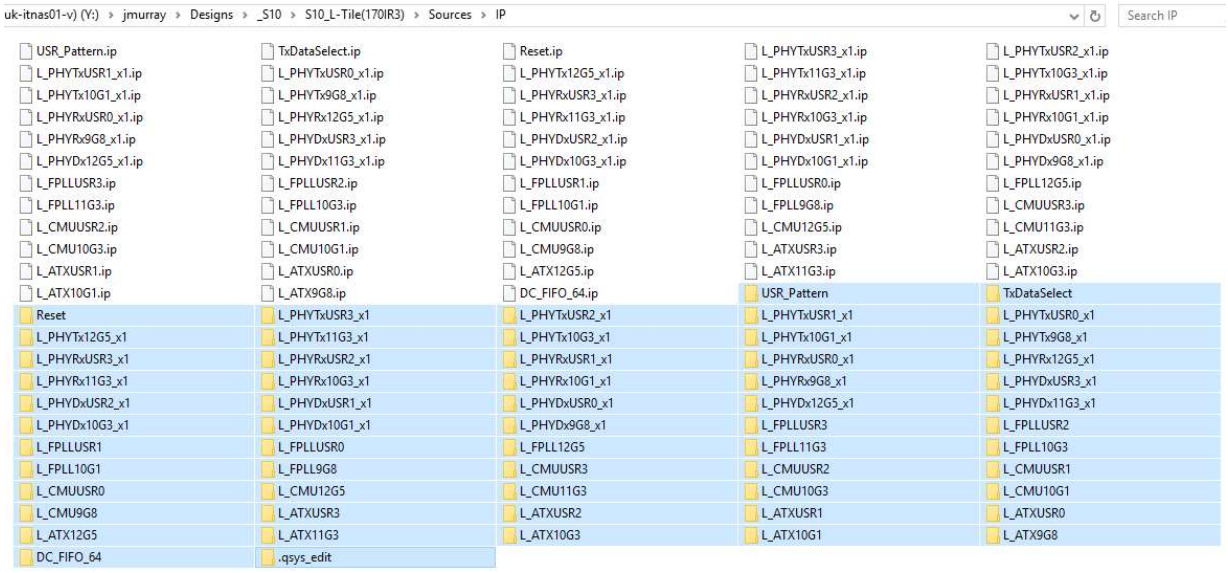
Transceiver Voltage Settings

The GX and GXT channel PHY and PLL IP use 1.1v by default. Stratix 10 devices must have common side-wide transceiver power. If you need 1.0v settings for your application you will need to change these in the PHY and ATX PLL IP. This design assumes the whole device is powered at the same voltage.

Speed Grades

This design supports the highest datarates as well as more generic datarates. When generating IP, the device speed grade is detected. If the speed grade is too slow for a given transceiver IP, it will not generate. You will need to temporarily select a fast-enough speed grade to generate the IP, even if you don’t plan to use it in your design. Once generated, you can reselect your slower speed grade device.

If you see “out of date” IP errors, consider deleting all of the IP folders such that only the top level *.ip file remains.



Then in Quartus settings select the following to regenerate the IP for your specific device next time you synthesize.

- Quartus Settings > IP Settings > Always regenerate IP

The transceivers are not visible in the Quartus Transceiver Toolkit.

Check that the reset to all IP is not asserted

Check that the Avalon Memory Mapped clock is present and connected

Check that the REFCLK is properly connected

Check that your AVMM clock is properly timing constrained

Appendix A – List commands in helper_ttk.tcl file

```
================================================
This tcl file contains a number of procedures to do additional configuration and status updates for Stratix 10
Revision 1.6
Date : 17 May 2017
Author : Peter Schepers
Note that for DFE the nios triggered DFE is still in debug, so for the moment use freeze =0
Only function DFE per channel in a certain phy is currently up to date, the others are not
Optimize AC gain is still wip
================================================
rmw_all {offset mask newval}
example useage rmw_all 0x160 0x0E 0x03

rmw_channel {phy index channel offset mask newval}
example useage rmw_channel $phy 1 3 0x134 0x0F 0x03

rd_channel {phy index channel offset}
example useage rd_channel $phy 0 3 0x134

rd_all {offset}
example useage rd_all 0x160

rd_atxpll {atx_pll index offset}
example rd_atxpll $atx_pll 0 0x480

rmw_atxpll {atx_pll index offset mask newval}
example rmw_atxpll $atx_pll 0 0x480 0x01 0x01

rd_fpll {fpll index offset}
example rd_atxpll $fpll 0 0x480

rmw_fpll {fpll index offset mask newval}
example rmw_fpll $fpll 0 0x480 0x01 0x01

set_vga_gain_all {vga_gain}
example useage set_vga_gain_all 5

get_vga_gain_all {}
example useage set_vga_gain_all

set_vga_gain_phy {phy index vga_gain}
example useage set_vga_gain_phy $phy 1 5

get_vga_gain_phy {phy index}
example useage get_vga_gain_phy $phy 1

set_vga_gain_channel {phy index channel vga_gain}
```

example useage set_vga_gain_channel $phy 1 2 5

set_ac_gain_all {ac_gain}
example useage set_ac_gain_all 5

get_ac_gain_all {}
example useage set_ac_gain_all

set_ac_gain_phy {phy index ac_gain}
example useage set_ac_gain_phy $phy 1 5

get_ac_gain_phy {phy index}
example useage get_ac_gain_phy $phy 1

set_ac_gain_channel {phy channel index ac_gain}
example useage set_ac_gain_channel $phy 1 2 5

set_cdr_gear_all {dyn fix ipd}
example useage set_cdr_gear_all 31 3 3

get_cdr_gear_all {}
example useage get_cdr_gear_all

set_cdr_gear_phy {phy index dyn fix ipd}
example useage set_cdr_gear_phy $phy 0 31 3 3

get_cdr_gear_phy {phy index}
example useage get_cdr_gear_phy $phy 0

set_cdr_gear_channel {phy index channel dyn fix ipd}
example useage set_cdr_gear_channel $phy 1 4 31 3 3

rx_polarity_inv_on {phy index channel}
example useage rx_polarity_inv_on $phy 0 2

rx_polarity_inv_off {phy index channel}
example useage rx_polarity_inv_off $phy 0 2

status_dump_all{}
example useage status_dump_all

status_dump_phy{phy index}
example useage status_dump_phy $phy 1

status_dump_channel{phy index channel}
example useage status_dump_channel $phy 1 3

raw_dump_channel {phy index ch start_addr stop_addr}

example useage raw_dump_channel $phy 0 0 0x000 0x300

compare_xcvr_channels {phy index ch1 ch2 start_addr stop_addr}
example useage compare_xcvr_channels $phy 0 3 5 0x100 0x250

set_serial_loop_all {loopback}
example useage set_serial_loop_all 1

set_serial_loop_phy {phy index loopback}
example useage set_serial_loop_phy $phy 1 1

enable_rev_serial_post {phy index channel}
example useage enable_rev_serial_post $phy 0 1

enable_rev_serial_pre {phy index channel}
example useage enable_rev_serial_pre $phy 0 1

disable_rev_serial {phy index channel}
example useage disable_rev_serial $phy 0 1

dfe_enable_channel {phy index channel number_of_taps freeze_mode}
example useage : dfe_enable_channel $phy 0 0 11 1
info:number_of_taps : 3,7 or 11
info:freeze_mode : 1 is default operation, 0 is for debug capabilities

dfe_disable_channel {phy index channel}
example useage : dfe_disable_channel $phy 0 3

dfe_enable_phy {phy index number_of_taps freeze_mode}
example useage : dfe_enable_phy $phy 0 11 1
info:number_of_taps : 3,7 or 11
info:freeze_mode : 1 is default operation, 0 is for debug capabilities

dfe_disable_phy {phy index}
example useage : dfe_disable_phy $phy 0

dfe_enable_all {number_of_taps freeze_mode}
example useage : dfe_enable_all 11 1
info:number_of_taps : 3,7 or 11
info:freeze_mode : 1 is default operation, 0 is for debug capabilities

dfe_disable_all {}
example useage : dfe_disable_all

triggered_ctle {phy index channel}
example triggered_ctle $phy 1 3

get_converged_all {}

example useage get_converged_all

show_dfe_taps_all {}
example useage show_dfe_taps_all

show_dfe_taps_channel {phy index channel}
example useage show_dfe_taps_channel $phy 1 2

show_dfe_taps_channel_loop {phy index channel loopcount}
example useage show_dfe_taps_channel_loop $phy 1 2 1000

enable_floating_taps {phy index channel}
example useage enable_floating_taps $phy 0 1

get_offset_all {}
example useage get_offset_all

get_pma_settings_all {}
example get_pma_settings_all

get_pma_settings_phy {phy index}
example get_pma_settings_phy $phy 0

get_pma_settings_channel {phy index channel }
example get_pma_settings_channel $phy 0 1

recalibrate_channel {phy index channel rate_switch}
example recalibrate_channel $phy 1 0 1

recalibrate_receiver {phy index channel rate_switch}
example recalibrate_receiver $phy 1 0 1

recalibrate_transmitter {phy index channel rate_switch}
example recalibrate_transmitter $phy 1 0 1

recalibrate_atxpll {atx_pll index}
example recalibrate_atxpll $atx_pll 0

recalibrate_fpll {fpll index}
example recalibrate_fpll $fpll 0

find_optimum_ac_gain_algo {phy index channel verbose}
example find_optimum_ac_gain_algo $phy 0 1 1

start_check_serial_loop {}
example useage start_check_serial_loop

start_check_prbs31 {}

example useage start_check_prbs31

Revision Control

V1.0, 16 May 2017. Initial release

V1.1, 29 June 2017. Converted design to combined L-Tile ES2 and H-Tile IP type.

Added “rate” generic to GXT channels

L-Tile ES2 now supports GXT channels

Added guidance for speed grade and IP generation problems

V1.2, 03 July 2017. Added SI Pattern Mux to GXT channels

Added Adaption script.

Added case sensitivity troubleshooting tip

V1.3, 23 August 2017. Enabled ADME settings in Tx and Rx simplex PHYs

Corrected datarate on 9G8 Rx Simplex PHY

Corrected a bug in Refclk_Select.vhd line 628

V1.4, 03 November 2017. Aligned with Quartus Prime Pro 17.1

Quartus 17.1 does not accept IP configured for rates that can’t be supported by the device OPN. Broke the single design out into three different speed grade designs to support different data rates.

Added support for 28.3Gbps GXT channels