

Targeted for ACDS 21.4

Intel® Agilex® I-Series F-Tile Superlite II Demo designs Supporting Documentation V1.0



Introduction & Motivation

- This document serves as supporting documentation for the various F-tile Superlite II demo designs.
- All F-tile demo designs (including Superlite II/IV variants) follow a similar configuration and can have different number of channels, different clocking, using FEC or not, FGT or FHT etc. but the look and feel is always the same, both from RTL implementation as well from software implementation.
- The software is written in such a way that there is a single main.c file and supporting files that is being re-used by all designs (whether FEC is used or not) and the parameterization is done using the parameters.h file. Those software files are also maintained on the Intel Forum Intel Agilex I-Series Demo page <https://community.intel.com/t5/FPGA-Wiki/High-Speed-Transceiver-Demo-Designs-Agilex-I-Series-F-Tile/ta-p/1315123>
- This allows to maintain a single set of source files to be used across all designs and only very little parameterization is required. E.g. to move from a design that is using 2 PHY's with 4 channels each with PAM4 without FEC to a design that is using 3 PHY's with FEC only very few lines need to be modified in the parameters.h file. Examples of those are also maintained in the software section on the Intel Agilex I-Series Demo page.

Introduction & Motivation

- This design illustrates how to use the Superlite II V4 concept on Intel Agilex using the F-Tile
- The V4 version of Superlite II is based on the V2+V3 version.
- The V2 version uses a valid signal in combination with the transmit data and there is also a ready signal provided which can potentially halt the transmit traffic for a short interval (based on filling level of Transmit FIFO). This interface is equivalent to the Avalon Streaming Interface.
- The V3 version adds additional functionality :
 - Link information is exchanged between the local and remote side and results in a LinkUp when both local and remote side are aligned.
 - XOFF is used in order to allow for backpressure or Flow Control: ask remote side to stop sending traffic (when local side e.g. is not capable of receiving the data).
 - Because of the exchanges between local and remote side the protocol is now a fully bidirectional protocol (V1 and V2 versions could be used fully unidirectional)

Introduction & Motivation

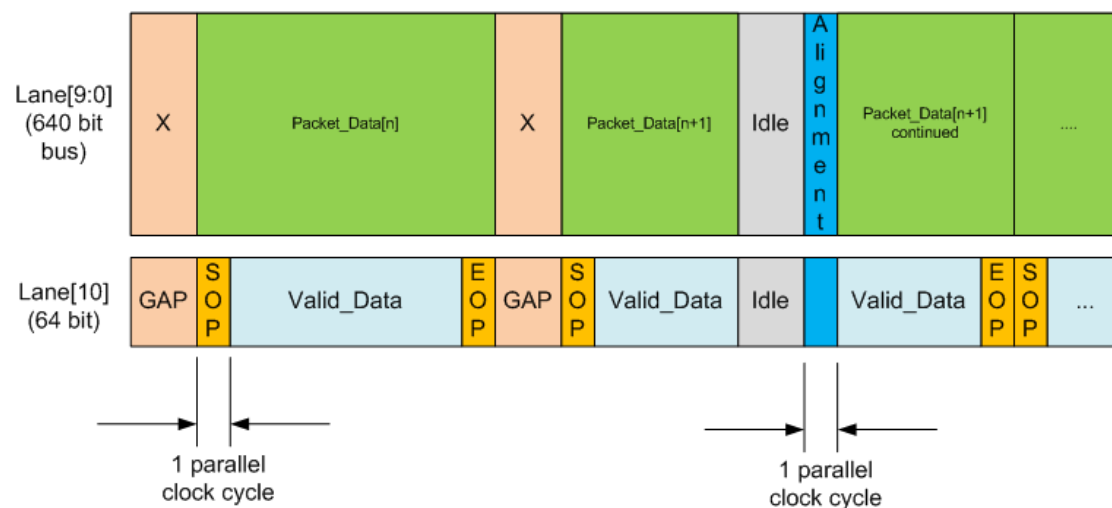
The V4 version is the version where transmit datapath is clocked with a single clock derived from the transceiver. This simplifies the Tx statemachine and FIFO implementation and has a significant reduction of the latency.

Additional features added in the V4 version :

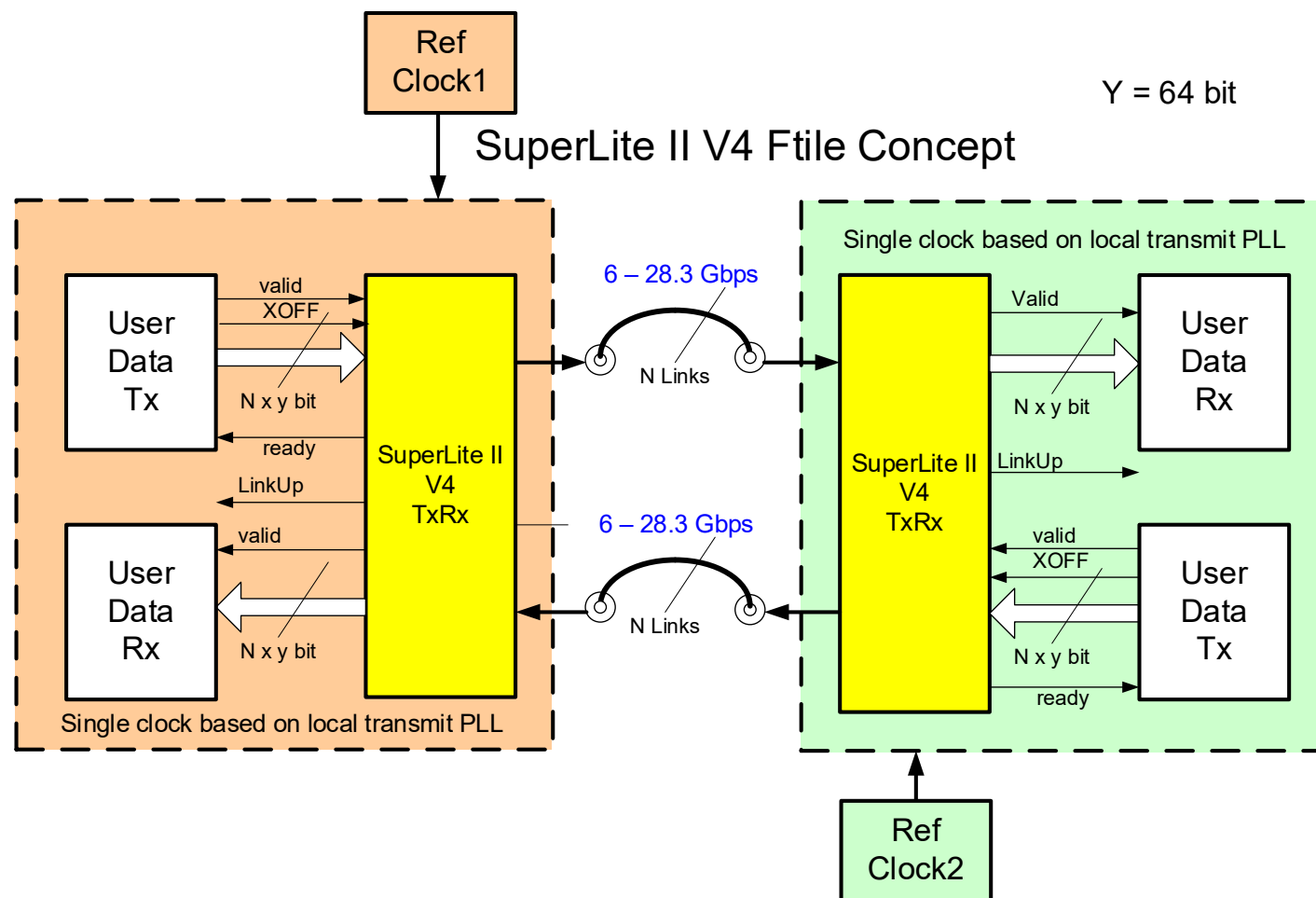
- Insertion and detection of Lane Identification: each lane has its own unique identifier. If lane swap or lane re-ordering occurs the information received in terms of lane identification allows to automatically recreate the original data pattern. This helps in not having to worry about how the lanes are routed in layout or how cables are connected.
- Added GENERIC parameter SIMPLEX which when set to true does not perform handshaking with the remote side and allows SIMPLEX operation (even when using Duplex transceiver). When parameter SIMPLEX is set to true, Linkup will be declared when LaneAligned is achieved locally.
- Simplified exchange of control characters for flow control.
- Tested interoperability with equivalent Superlite II V4 versions on A10 and S10 GX (H-tile).
- Latency measurement is inherently part of the protocol now.
- This specific version is using 2 instances of the Superlite II V4 demo, each with 4 lanes running at 24.78125 Gbps where one is connected to the lower lanes of both QSFP-DD modules. Using DAC or Fiber Optic connection a full system test can be done between the 2 modules, illustrating link-up, reset behaviour etc.

How to use Superlite II V4 in Packet mode?

- By nature the Superlite II V4 is using streaming data (Avalon[®] streaming interface based) so not framed or packetized. One could however support also a framed mode (or packet mode) by using one of the lanes to transport the packet related information (SOP, EOP, etc). This does require an additional lane to provide the packet overhead information. This concept has been successfully implemented on Stratix V GX (with 11 lanes) and can easily be repeated on Stratix10 (with higher rates and including KR-FEC).



'Superlite II V4' Concept



'SuperLite II V4' Concept

- Transport data from point A to point B as simple as possible multiple bidirectional serial links. Note that the number of lanes in each direction do not have to be equal.
- V3 version of the protocol exchanges information between local and return side (handshaking) as well as XON/XOFF control (backpressure)
- On the transmit side data will only be written in the FIFO when the valid signal is asserted, the fifo also can issue a backpressure signal (indicated by the 'ready' signal to indicate if data can be transmitted. This is equivalent to the Avalon Streaming interface.
- When XOFF is received from the remote side, all traffic will be halted on the local side (link remains up though).
- All the logic is clocked on one single clock (i.e. both transmit and receive core logic), this clock is the clock derived from the transmit PLL and is basically the line rate/64.
- This implies that the combination of data valid with the locally generated clock from the local transmit PLL is frequency locked to the clock used at the transmit side (which will typically vary in ppm).
- Uses 64/66 Encoding in combination with scrambling.
- Multiple lanes are bonded, there is no real limit on the number of lanes that can be used, apart from potential limitations to compile very large number of lanes. Typically 4 or 10 lanes will be used. Also a single lane is possible.

‘Superlite II V4’ Concept (continued)

- The Superlite II V4 TX module will automatically pause Tx user data if needed to insert idle and alignment characters and will also pause Tx user data based on the “ready” received from the Native PHY (as it is working in overspeed). So the user data needs to be able to immediately stop sending traffic. If this is not possible one can add an additional FIFO with a slower running write clock.
- User data is always 64 bit per lane because of the 64b66b encoding.

'SuperLite II V4 2x4 lanes Demo Design

■ Goals :

- Demonstrate the Superlite II V4 concept on the Intel Agilex F-Tile
 - Use Intel Agilex I-Series PCIe dev kit as platform for demonstration.
 - Uses 2 Superlite II V4 demo instances (each with their own packet generator and checker). One instance (or "phy") is connected to the QSFPDD0 module (lower lanes) and the other one is connected to the QSFPDD1 module (lower lanes)
 - Full system validation can be done using either a DAC cable (various length) or Fiber Optic connection using the 2 QSFP-DD modules.
- Line rate in this demo is 25.78125 Gbps per lane by default but it could run up to 28.3 by changing the software.
- 4 lanes will be used for each phy, therefore aggregate line rate is 100 Gbps per phy
- User data width is 256-bits (64 bits per lane) per phy
- Actual throughput depends on the configured "read time" and "idle time" parameters. In the demo the "read time" is 1056 clock cycles and "idle time" is 7 clock cycles but these can be easily changed. The "read time" needs to be an integer multiple of 33. The "read time" would normally be set to a higher value like e.g. 67584 clock cycles in order to increase the efficiency.
- Actual throughput is measured in the design by measuring the data_valid received in combination with the clock.







Goals (continued)

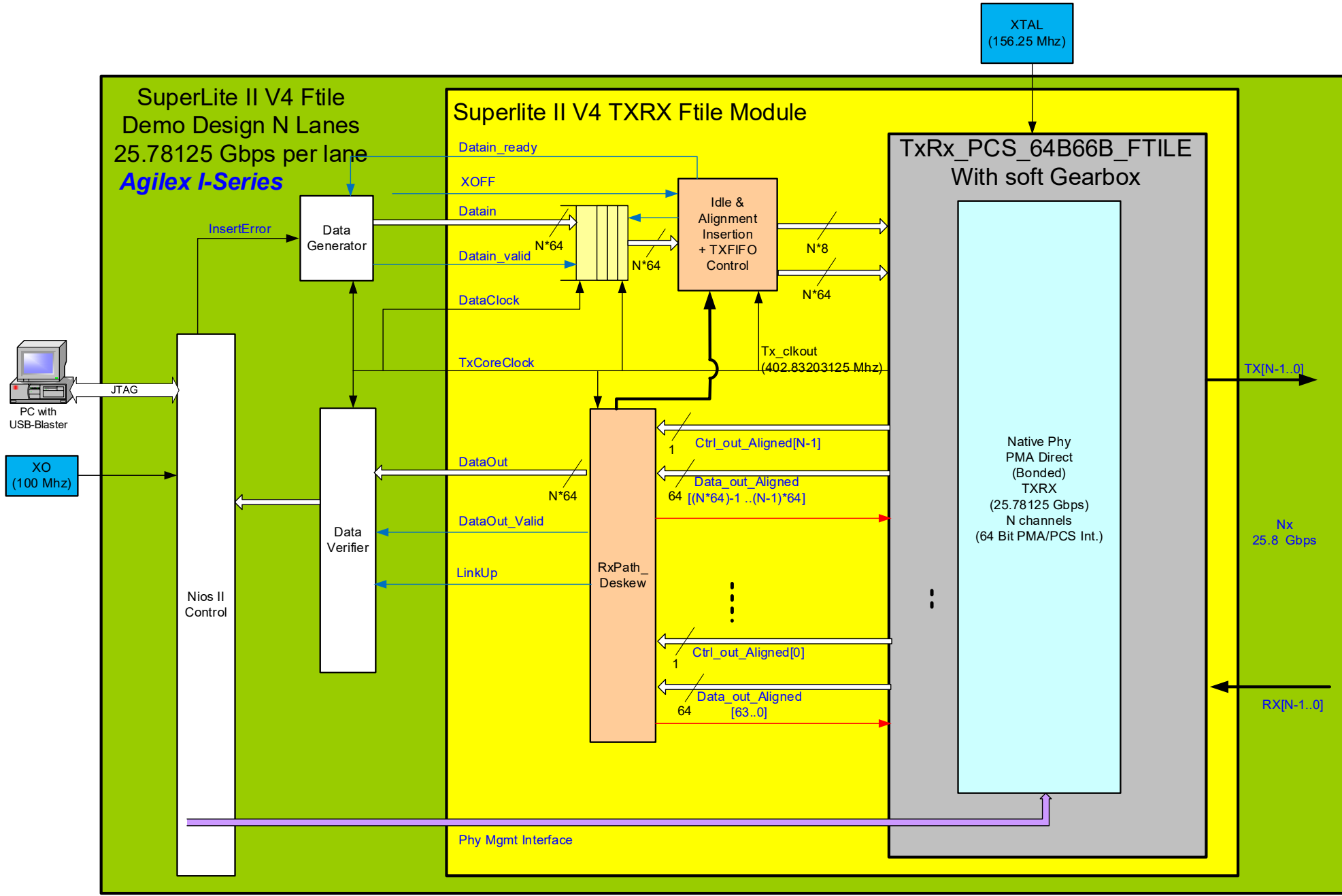
■ Goals :

- The user data is generated based on the line_rate/64 Mhz clock generated by the Native PHY . As it receives backpressures from the Superlite IV Tx module the data will be halted at specific times.
- To create the 256 bit user data in each phy a combination of different counterpatterns and prbpatterns are used across the 4 lanes of each phy
- Through NDME Transceiver Toolkit support is automatically available.
- Use Nios II processor as controller/monitor.
- The RAM in the QSYS is programmed with the latest software, so one can directly connect to the system using JTAG and run the software.
- Reports errorcount and calculates BER based on the received data.
- Includes Testbench

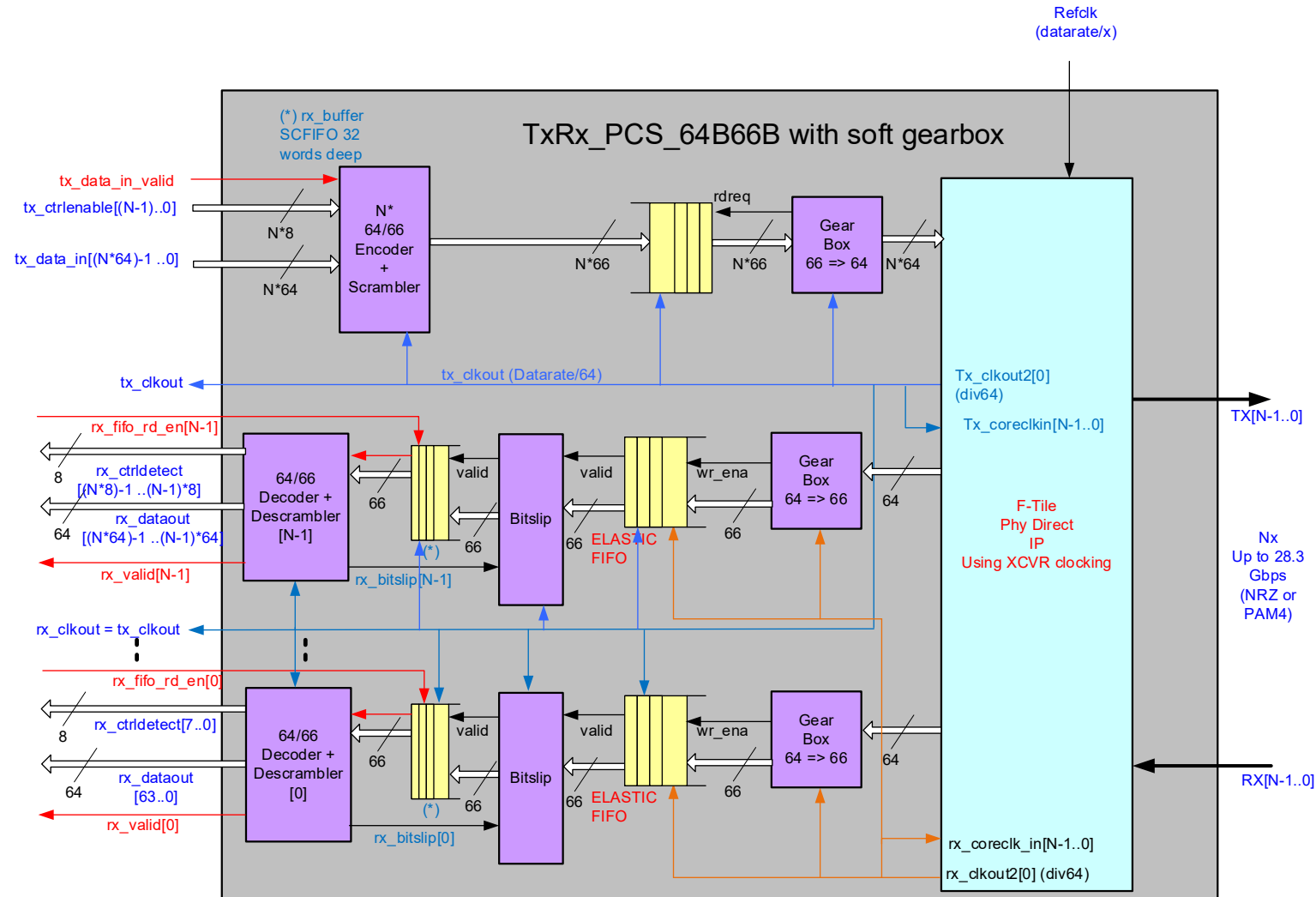
'Superlite II V4' Resources (one phy)

- Superlite II V4 Ftile TxRx Module Resources 4 lanes at 25.78125 Gbps targeting Intel Agilex
 - ~ 13700 ALUT's
 - ~ 12450 ALM's
 - ~ 25500 Dedicated logic registers
 - 27 M20K (for the receive and transmit buffers)
 - 19008 Block Memory bits
 - Note that the amount of resources is higher than normal because a soft gearbox implementation has to be used as the PHY direct IP does not support the gearbox function yet in this mode (should be supported by the PHY direct IP in a later Intel® Quartus® Prime Software release)

Instance	Entity	ALMs needed [=A-B+C]	ALMs used for memory	Combinational ALUTs	Dedicated Logic Registers	Block Memory Bits	M20Ks
▼  superliteii_txrx_module_inst	superliteii_txrx_module	12459.0 (81.3)	160.0 (0.0)	13665 (36)	25329 (311)	19008	27
 Reset_Synchro_inst1	reset_synchro	2.0 (2.0)	0.0 (0.0)	1 (1)	4 (4)	0	0
 Reset_Synchro_inst2	reset_synchro	1.5 (1.5)	0.0 (0.0)	0 (0)	4 (4)	0	0
 Rx_Path_inst	Rx_Path_Deskew	116.4 (116.4)	0.0 (0.0)	181 (181)	107 (107)	0	0
▶  Tx_Ratematcher_Control_inst	Tx_Ratematcher_Control	194.9 (179.5)	0.0 (0.0)	352 (326)	352 (323)	2048	7
▼  txrx_pcs_64b66b_inst	txrx_pcs_64b66b	12062.8 (3.6)	160.0 (0.0)	13095 (12)	24551 (8)	16960	20



'txrx_pcs_64b66b' with soft Gearbox Detailed blockdiagram



Tx Path

- A datastream is generated at a clock of 402.83203125 Mhz @ 256 bits when DataIn_ready is high. The datapattern is a combination of 4 60-bit Prbs-23 (each with their own starting value) and one 16-bit counterpattern.
 - The 16-bit counterpattern is spread across the 16 lanes, 4 bits per lane. This is to make sure any deskew on the lanes would be immediately spotted.
- Since idle +alignment characters have to be inserted at regular intervals and because of the overclocking the pcs is generating periodic tx_pcs_ready pulses the data stream will be halted at regular intervals. This is indicated by the Superlite IV TX module by the DataIn_ready signal.
- Single biterrors can be inserted using inserterror input. As there is one PRBSGenerator per virtual lane, the biterror insert will produce 4 bit-errors at a time.
- This data is presented to the Superlite II V4 TxRx Module
- The data is written into the TxFifo (single clocked fifo) using the 402.83 Mhz as clock and using the DataIn_Valid as write-enable.
- The read_enable is pulsed at the same pace as the write-enable.
- If no valid data is present idle characters will be sent.

Tx Path (continued)

- The 64b66 encoding is very straightforward : bits 65..64 are set to “10” for control data and “01” for data (so neutral disparity)
- In terms of idle character, a value of “1C1C...1CBC” is being used. The alignment character is sent as the last control character. The lower 16 bits of this control word (at the alignment position) encodes the status of the link in addition to the alignment indication.
- If no local alignment has been found : the last control word sent is “..7C7C”
- If local word alignment has been found : the last control word sent is “..FD7C”
- If XOFF is being sent to the remote side : the last control word sent is “..5C7C”
- Lane Alignment coding (B<i>i</i> is the byte)

B7	B6	B5	B4	B3	B2	B1	B0
Lane Identity	Latency Count Tx	7C	7C	7C	7C	FD/5C/7C	7C

Tx Path (continued)

- In order to make sure there is DC balance and enough transitions present on the serial links a selfsynchronizing scrambler is used (the same as being used in 10GbE). The scrambler will scramble only the 64 databits (the framing bits on bits [65..64] are not scrambled since they are required for the framing. The polynomial used for the scrambler is $1 + x^{39} + x^{58}$ according to the IEEE.
- The resulting 66-bit is sent to a buffer where it is being read by a gearbox implemented in soft where the gearbox converts it from 66-bit to 64-bits and sent it to the PHY direct IP.
- In the PHY direct IP the 64-bit is serialized (using 64-bit serializer) and sent out as serial datastreams (25.8 Gbps per lane).

Rx Path

- After deserialization of the data the 64-bit data is passed to a gearbox (using the recovered clock). The output from the gearbox is then written into an elastic FIFO where the readclock is the tx_clkout, from this point all Rx data is clocked on the same clock domain as the tx clock.
- The resulting 66 bits data per lane from the elastic FIFO is then presented to the 64b66b decoder module (through the rx_buffer) which will periodically assign bitlips until correct frame alignment has been achieved.
- This is all happening on all lanes at the same time and everything is clocked based on the single clock out derived from the transmitter (the overspeed clock at datarate/64).
- The rx_buffers are used for the deskew.

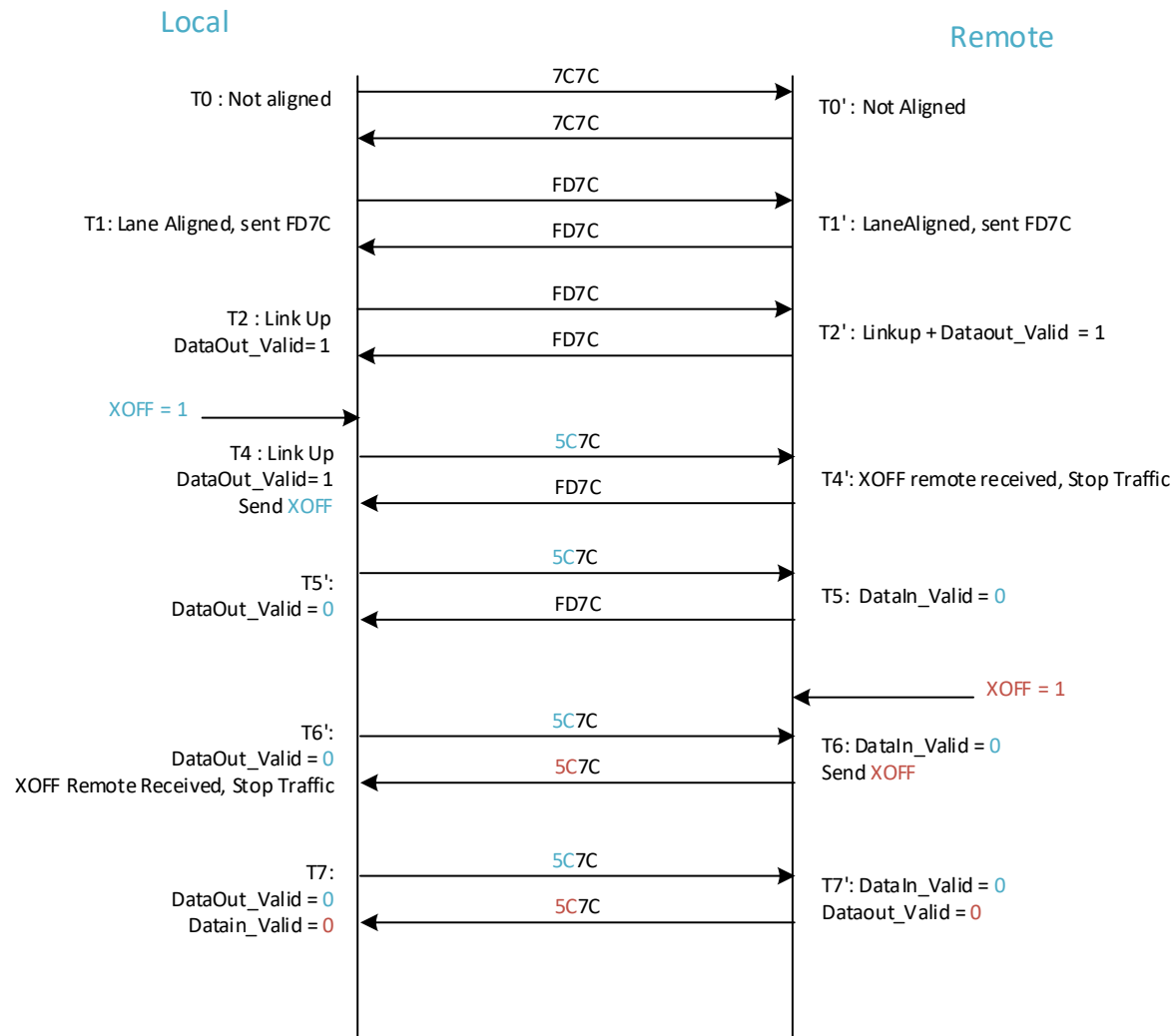
Rx Path (continued)

- Based on the position of the alignment word, the delay for each lane is measured, the statemachine inside the Rx_path_deskew module will use the information of that measured delay to control a special Rx FIFO signal “rx_fifo_rd_en” to all rx_buffers to deskew all the lanes.
- Once lane alignment is achieved the 256-bit reconstructed data based on the received Lane Identification in addition with a data valid signal will be sent to the data verifier.
- Here PRBS and counterverification will take place and single biterrors will be counted.
- Prbslocked is asserted when 128 consecutive valid Prbs-23 60-bit words have been received on each of the lanes and de-asserted when 128 consecutive non-valid Prbs-23 60-bit words have been received.
- Countlocked is asserted when 128 consecutive valid 16-bit counter words have been received on the parts of the lanes they were received.
- There are 4 60 bit PRBS Verifiers (one for every lane) and 1 16-bit CountVerifiers to validate the 256 bit data.b

Rx Path (continued)

- The V3 version adds an additional Rx State machine to the logic to determine the local Rx state and it depends on the incoming “alignment” control word
- If last control word received is “...FD7C” it means the remote side has reached Wordalignment, if local side also has reached wordalignment this results in Linkup to be asserted, indicating that both sides are up and running. This can be used as the trigger to sent data across e.g.
- If last control word received is “...5C7C” it means an XOFF has been received from the remote side, this will be forwarded to the local transmitter to stop generating traffic on the local side.
- (See Flow chart on next slide for more details)

Control Characters used in combination with XOFF



F-Tile Phy Direct IP (Common Datapath)

General	
Number of system copies:	1
Common Datapath Options	
PMA type:	FGT
FGT PMA configuration rules:	Basic
Number of PMA lanes:	4
Datapath clocking mode:	PMA
System PLL frequency:	830.078125 MHz
PMA mode:	Duplex
PMA modulation type:	NRZ
PMA data rate:	25781.25 Mbps
PMA parallel clock frequency:	805.6640625 MHz
PMA width:	32
<input type="checkbox"/> Enable RX de-skew when available	
<input type="checkbox"/> Provide separate interface for each PMA	

F-Tile Phy Direct IP (Tx Datapath Options)

TX PMA Interface

TX PMA interface FIFO mode: Phase compensation

☐ Enable tx_pmaif_fifo_empty port

☐ Enable tx_pmaif_fifo_pempty port

☐ Enable tx_pmaif_fifo_pfull port

TX Core Interface

TX Core Interface FIFO

☐ Enable custom cadence generation ports and logic

☐ Enable tx_cadence_slow_clk_locked port

TX core interface FIFO Mode: Phase compensation

TX tile interface FIFO Mode: Phase compensation

☒ Enable TX double width transfer

TX core interface FIFO partially full threshold:

TX core interface FIFO partially empty threshold:

TX FGT PMA

☐ Enable Gray coding

☐ Enable precoding

PRBS generator mode: disable

☐ Enable fgt_tx_beacon port

TX FGT PLL Settings

Output frequency: MHz

VCO frequency: MHz

☐ Enable TX FGT PLL cascade mode

☒ Enable TX FGT PLL fractional mode

TX FGT PLL integer mode reference clock frequency: 156.250000 MHz

TX FGT PLL fractional mode reference clock frequency: 156.25 MHz

☐ Enable Core PLL mode

F-Tile Phy Direct IP (Tx Datapath Options cont.)

TX Clock Options	
Selected tx_clkout clock source:	User Clock1
Frequency of tx_clkout:	390.625 MHz
<input checked="" type="checkbox"/> Enable tx_clkout2 port	
Selected tx_clkout2 clock source:	Word Clock
tx_clkout2 clock div by:	2
Frequency of tx_clkout2:	402.832031 MHz
<input type="checkbox"/> Enable tx_clkout_hioint port	
<input type="checkbox"/> Enable tx_clkout2_hioint port	
Selected tx_coreclkln clock network:	Dedicated Clock
Selected tx_coreclkln2 clock network:	Dedicated Clock

F-Tile Phy Direct IP (Rx Datapath Options)

RX Core Interface FIFO

RX core interface FIFO mode: Phase compensation

☒ Enable RX double width transfer

RX core interface FIFO partially full threshold: 10

RX core interface FIFO partially empty threshold: 2

☐ Enable rx_fifo_full port

☐ Enable rx_fifo_empty port

☐ Enable rx_fifo_pfull port

☐ Enable rx_fifo_pempty port

☐ Enable rx_fifo_rd_en port

☐ Enable rx_pcs_fifo_full port

☐ Enable rx_pcs_fifo_empty port

RX FGT CDR Settings

Output frequency: 12890.625000 MHz

VCO frequency: 12890.625000 MHz

RX FGT CDR calculated reference frequency: 322.265625 MHz

CDR lock mode: auto

☐ Enable fgt_rx_set_locktoref port

☐ Enable fgt_rx_cdr_freeze port

RX User Clock Setting

☒ Enable RX user clock

RX user clock div by: 33

F-Tile Phy Direct IP (Rx Datapath Options cont.)

RX Clock Options

Selected rx_clkout clock source:

User Clock1

Frequency of rx_clkout:

390.625

MHz

☒ Enable rx_clkout2 port

Selected rx_clkout2 clock source:

Word Clock

rx_clkout2 clock div by:

2

Frequency of rx_clkout2:

402.832031

MHz

Selected rx_coreclk clock network:

Dedicated Clock

☐ Enable rx_clkout_hioint port

☐ Enable rx_clkout2_hioint port

F-Tile Phy Direct IP (Avalon Memory-Mapped Interface)

TX Datapath Options RX Datapath Options RS-FEC **Avalon Memory-Mapped Interface** Example Design

▼ Datapath Avalon Memory-Mapped Interface

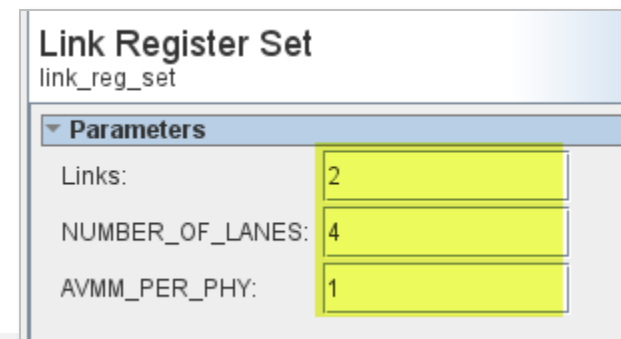
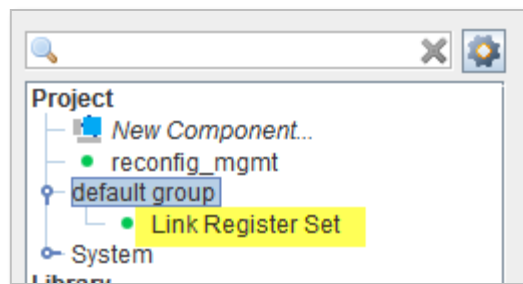
- ☒ Enable datapath Avalon interface
- ☐ Enable Direct PHY soft CSR
- ☐ Enable readdatavalid port on datapath Avalon interface
- ☐ Enable separate Avalon interface per fracture
- ☒ Enable Debug Endpoint on datapath Avalon interface

▼ PMA Avalon Memory-Mapped Interface

- ☒ Enable PMA Avalon interface
- ☐ Enable readdatavalid port on PMA Avalon interface
- ☐ Enable separate Avalon interface per PMA
- ☒ Enable Debug Endpoint on PMA Avalon interface

Link Register Set

- The design is using a custom .qsys component that can be parameterized. The component is called “Link Register Set” (using link_reg_set_hw.tcl) and is available in the default group section of the Qsys library.
- This component can be parameterized to the number of links, the number of LANES per Link and the number of PHY instances per link you want to instantiate in your design and makes the generation of the Qsys system much more streamlined.
- As in this design 2 Link's with each one PHY direct IP is being and each link has 4 lanes you end up with the configuration below (see screenshot)
- For every Link that you use in your design it will generate a number of registers (to control/readout the measured clocks) as well as all the required AVMM reconfiguration interfaces for the Native PHY (which is for every PHY instance : 1 AVMM for the transceiver configuration interface and 1 AVMM for the PDP (Parallel Datapath configuration interface).
- For the AVMM interfaces the Link Register set is calling another custom component : “reconfig_mgmt” (reconfig_mgmt_hw.tcl)



Timing Closure (Setup)

- The design is fully constraint without any timing violations. Note that in this design multiple clocks are being used (because of the soft gearbox implementation)
- Compiled for AGIB027R29A1E2VR0

Setup Summary					
Show:	Visible ▾	Hide	Q <<Filter>>		
	Clock	Slack	End Point TNS	Failing End Points	Worst-Case Operating Conditions
1	Generate_Superlite_II_Instances[0].inst...ect_inst directphy_f_0 rx_clkout2 ch11	0.261	0.000	0	Slow vid2 100C Model
2	Generate_Superlite_II_Instances[1].inst...ect_inst directphy_f_0 tx_clkout2 ch19	0.400	0.000	0	Slow vid2 100C Model
3	Generate_Superlite_II_Instances[0].inst...ect_inst directphy_f_0 tx_clkout2 ch11	0.401	0.000	0	Slow vid2 100C Model
4	Generate_Superlite_II_Instances[1].inst...ect_inst directphy_f_0 rx_clkout2 ch19	0.547	0.000	0	Slow vid2b 100C Model
5	altera_int_osc_clk	1.644	0.000	0	Slow vid2 100C Model
6	clock_divider_inst intelclkctrl_0 clkdiv_inst clock_div2	2.118	0.000	0	Slow vid2b 100C Model
7	src_divided_osc_clk	4.084	0.000	0	Slow vid2b 100C Model
8	altera_reserved_tck	12.269	0.000	0	Slow vid2 100C Model

Timing Closure (Hold)

- The design is fully constraint without any timing violations
- Compiled for AGIB027R29A1E2VR0

Hold Summary					
Show: Visible ▾ Hide <<Filter>> ...					
	Clock	Slack	End Point TNS	Failing End Points	Worst-Case Operating Conditions
1	clock_divider_inst intelclkctrl_0 clkdiv_inst clock_div2	0.009	0.000	0	Fast vid2 100C Model
2	Generate_Superlite_II_Instances[0].ins...ect_inst directphy_f_0 rx_clkout2 ch11	0.028	0.000	0	Fast vid2a 100C Model
3	Generate_Superlite_II_Instances[0].inst...rect_inst directphy_f_0 tx_clkout2 ch11	0.033	0.000	0	Fast vid2 100C Model
4	Generate_Superlite_II_Instances[1].inst...rect_inst directphy_f_0 tx_clkout2 ch19	0.033	0.000	0	Fast vid2 100C Model
5	altera_int_osc_clk	0.043	0.000	0	Fast vid2 100C Model
6	Generate_Superlite_II_Instances[1].ins...ect_inst directphy_f_0 rx_clkout2 ch19	0.052	0.000	0	Fast vid2 100C Model
7	altera_reserved_tck	0.053	0.000	0	Fast vid2a 100C Model
8	src_divided_osc_clk	0.062	0.000	0	Fast vid2 100C Model

Native Phy Debug Master Endpoint Interface

- In addition to the AVMM interface the Native PHY also enable the Native Phy Debug Master Endpoint interface which can also be accessed through the Qsys system due to the addition of the jtag_debug_module.
- Access to the Native Phy Debug Master Endpoint interface can be done through System Console totally in parallel while the Nios controller(s) are being used.
- The Native Phy Debug Master Endpoint interface can be used for advanced debugging and reporting (see further) .
- The major advantage of having the Native Phy Debug Master Endpoint interface is that it allows to run directly the Universal Toolkit (see further).

Nios II Control & Monitoring

The Nios II controller is used to control and monitor the design using the Nios II terminal.

- Display status signals for each of the channels
 - Tile/Quad information
 - Encoding (PAM4/NRZ and gray and/or 1/1+D encoding)
 - Link Up
 - DataLocked (combination of the PrbsLocked and Countlocked)
 - LaneAligned
 - Effective Datarate
 - Latency (only when looped back on its own)
 - Lock Alarm
 - Freq_Locked
 - Tx_ready/Rx_ready
 - Serial Loop + Reverse Parallel Loop
 - Channel Type
 - Connection Type
 - Rx/Tx Polarity Inversion
 - Number of biterrors.
- Displays the bitrate based on measured reference clock frequency
- Displays also the recovered clock frequency (of lane 0 only). Note that if lane 0 has no connection or is not in serial loopback the measured recovered clock frequency is a random number as the CDR is not locked to anything.

Nios II Control & Monitoring (continued)

- Control Serial Loopback on all lanes and per phy
- Control Reverse Parallel Loopback on all lanes and per phy
- Error insertion : Soft Biterror (which results in one error per virtual link)
- Displays the number of biterrors and the BER.
- Reads out and print out PMA settings for all channels
- Control Invert Tx and Rx Polarity on all lanes and per phy
- PMA tuning : Allows to find the optimum Transmit PMA settings (VOD, Pre-emphasis) using PMA sweep function.
- Stress Tests : 9 different stress tests are available (see dedicated slide) including latency measurement
- Displays the time the test has been running on the Transceiver Block.

Nios II Control & Monitoring (to be added)

- Measures the temperatures of the core (supported) and the transceiver tile (to be added)
- Dumps I2C information of connected modules and cables like Vendor, part number, cable length, temperature of the module. (due to a hardware issue on the PCIe devkit (ES Version) the I2C interface does not work for some modules, so this functionality is disabled for software compiled for the PCIe devkit)
- Allows to dump the I2C page Lower page and Page 00h registers of the selected module/cable (not available for the PCIe Devkit ES version)

Nios 2 Output in Nios II SDK Shell

```
Agilex I-series PCIe Devkit Superlite II Demo
-----
|Board Revision      : Agilex I-Series PCIe Devkit F-tile (ES Version)
|Hardware Revision   : 01/08/2022 variant 01
|Clocking of PHY's   : PMA clocking
|FGT Firmware version : 0x3cb40195
|Software Build Date  : Jan 10 2022 15:53:00
-----
|PHY                |      0 |      1 |
|RefClock Measured (Mhz) |156.2500|156.2500|
|Line rate (Gbps)      | 25.7812| 25.7812|
|Number Of Lanes       |      4 |      4 |
|Aggregate rate (Gbps)  |103.1250|103.1250|
|Selected Channel      |      0 |      0 |
|Number of Phy's       |      2 |
|Total Aggregate rate (Gbps) | 206.250
|Selected Phy         |      0 |
-----
|Superlite Related statistics:
|Read Length          |    1056 |    1056 |
|Idle Length          |      7 |      7 |
|Ratio                |0.993371|0.993371|
|Throttle Datarate     |      0 |      0 |
|Net Received Bandwidth (Gbps) | 0.0000 | 0.0000 |
|Efficiency (%)        | 0.00000 | 0.00000 |
|Measured Max Latency (ns) |    178 |    312 | (only valid with loopback (int or ext))
|Measured Max Latency (clocks) |     72 |    126 | (only valid with loopback (int or ext))
-----
```

Test with QSFP-DD 2.5 Meter cable connected between PHY0 And PHY1

```
PHY 0 F-Tile 12A QSFPDD0 bottom
=====
Channel          :      0      1      2      3
-----
Transceiver Type :      FGT      FGT      FGT      FGT
FGT Quad         :      0      0      0      0
FGT Lane         :      3      2      1      0
Native Phy Mode  :  PMA-DIR  PMA-DIR  PMA-DIR  PMA-DIR
Mute Tx          :      0      0      0      0
Serial Loop      :      0      0      0      0
Reverse // Loop  :      0      0      0      0
Line Encoding    :      NRZ      NRZ      NRZ      NRZ
Invert Tx Polarity :
Invert Rx Polarity :
Connection Type  :  QSFPDD  QSFPDD  QSFPDD  QSFPDD
PrbsPattern      :      23      23      23      23
tx_ready         :      1      1      1      1
rx_ready         :      1      1      1      1
FreqLocked 1ms   :      1      1      1      1

XOFF Sent        :      0
XOFF Received    :      0
Throttle Data    :      0
Error Deskew     :      0
Fifo Error       :      0
WordAligned      :      1
LaneAligned      :      1
LinkUp           :      1
DataLocked       :      1
DataLocked Alarm :
Errorcount       :      0

BER (CL=0.95) Link : 4.351426e-15

Test Time PHY 0      : 1h 51m 25s
Tx Clkout Frequency  : 402.83203 MHz
DataClock_out Frequency : 388.33618 Mhz
Recovered Clock Frequency (Lane 0) : 402.83203 MHz
Measured ppm difference : 0
```

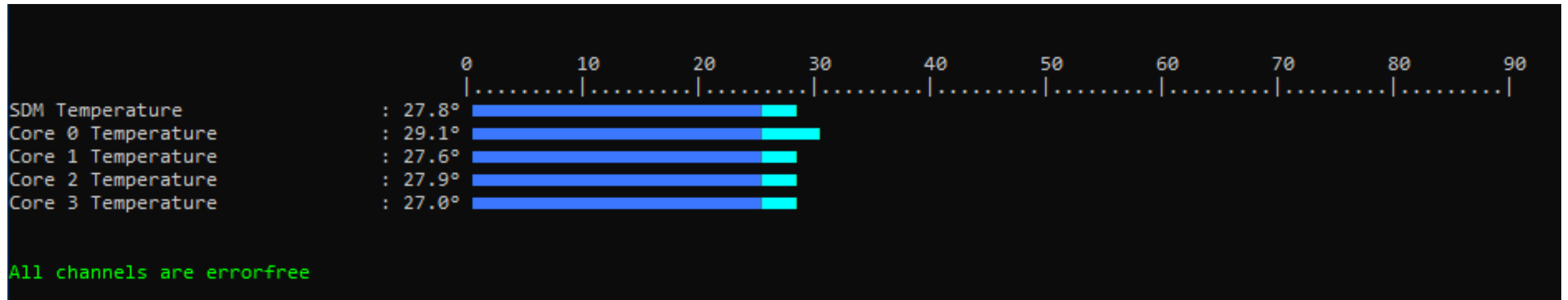
```
PHY 1 F-Tile 12A QSFPDD1 bottom
=====
Channel          :      0      1      2      3
-----
Transceiver Type :      FGT      FGT      FGT      FGT
FGT Quad         :      2      2      2      2
FGT Lane         :      3      2      1      0
Native Phy Mode  :  PMA-DIR  PMA-DIR  PMA-DIR  PMA-DIR
Mute Tx          :      0      0      0      0
Serial Loop      :      0      0      0      0
Reverse // Loop  :      0      0      0      0
Line Encoding    :      NRZ      NRZ      NRZ      NRZ
Invert Tx Polarity :
Invert Rx Polarity :
Connection Type  :  QSFPDD  QSFPDD  QSFPDD  QSFPDD
PrbsPattern      :      23      23      23      23
tx_ready         :      1      1      1      1
rx_ready         :      1      1      1      1
FreqLocked 1ms   :      1      1      1      1

XOFF Sent        :      0
XOFF Received    :      0
Throttle Data    :      0
Error Deskew     :      0
Fifo Error       :      0
WordAligned      :      1
LaneAligned      :      1
LinkUp           :      1
DataLocked       :      1
DataLocked Alarm :
Errorcount       :      0

BER (CL=0.95) Link : 4.351491e-15

Test Time PHY 1      : 1h 51m 25s
Tx Clkout Frequency  : 402.83203 MHz
DataClock_out Frequency : 388.33618 Mhz
Recovered Clock Frequency (Lane 0) : 402.83203 MHz
Measured ppm difference : 0
```

Temperature measurement



Available commands

```
Select Action :
=====
P. Toggle Display to show info for All Phy's or only Selected Phy
.. Set all channels in serial loopback except selected channel
[. Mute all FGT transmitters
]. Unmute all FGT transmitters
(. Reset all channels except selected channel
). De-assert Reset on all channels
1. Perform a reset on the Selected Phy
~. Perform a reset on the Selected channel
$. Perform a rx_reset on the Selected channel
^. Perform a tx_reset on the Selected channel
2. Select Phy
3. Select Channel To Control
5. Insert Biterrors on Link (4 at a time)
6. Reset ErrorCounter
7. Show Status
8. Control Serial loopback in Selected Phy
{. Control Reverse Parallel loopback in Selected Phy
C. continuously update BER and errorcount every 2 seconds
D. Input new BER Time Interval
E. Show Transceiver PMA Settings on all channels
G. Change TX PMA settings on the Selected Phy
M. Control Tx Polarity in Selected Phy
N. Control Rx Polarity in Selected Phy
Q. Clear Errorcounters on all Phys (including RSFEC counters)
R. Rerun adaptation on all channels in Selected Phy
S. Store register space of Selected Channel
T. Readout register space of Selected Channel and compare with previously stored register space and list differences
U. Sweep PMA settings and program best setting on selected channel
K. Toggle Throttle Datarate (set to 50% throughput) on Selected Phy
X. Toggle XOFF to partner to stop/start sending traffic at remote side
}. Dump I2C registers of selected module (Lower Page and Page 00h)
!. Read Register of PHY Direct in Selected Phy using rd_channel
:. Read Register of PHY Direct in Selected Phy using rd_channel_ftile
L. Stress test functions including latency measurement
Z. Dump address space
To stop the test press CTRL-C (this will automatically create output.log file

Enter Choice :
```

Stress Tests

Option 'L' provides a submenu to perform the following “stress tests”:

```
Press 1 if you want to reset only the Selected PHY and check it status
Press 2 if you want to reset the Selected PHY and check if both PHY's come up properly (requires connection between the PHY's)
Press 3 if you want to do rx_reset of the Selected PHY and check it status
Press 4 if you want to stress test AVMM accesses to the reconfiguration interface and the PDP interface
Press 5 to reset the rx of all lanes of the Selected Phy until the link is up on all lanes of the Selected Phy
Press 6 to reset the Selected Rx channel until the link is up on that Channel
Press 7 to run a test mimicing cable pull/plug-in using Tx mute option on local phy and verify if remote PHY comes up properly
Press 8 to run the same test as test 1 but specifically when used in combination with SiPh which take seconds to come up
Press 9 to measure latency across 1000 resets
```

Measuring Latency

- Stress test '9' is measuring the maximum latency across 1000 resets and provides a histogram of this maximum latency being measured.
- As can be seen maximum latency is on average 94 parallel clock cycles which is ~ 233 ns at a line rate of 25.78125 Gbps. (This is with external loopback on 4 of the physical lanes) using QSFP-DD electrical loopback.

```
Number of cases Maximum Latency is equal to 93 parallel clocks is : 329
Number of cases Maximum Latency is equal to 94 parallel clocks is : 610
Number of cases Maximum Latency is equal to 95 parallel clocks is : 61
```

Show PMA settings (option 'e') using 2.5 meter DAC cable

```
phy 0

Channel      : 0 | 1 | 2 | 3 |
              -----
FGT Quad     : 0 | 0 | 0 | 0 |
FGT Lane     : 3 | 2 | 1 | 0 |
Transceiver Type : FGT | FGT | FGT | FGT |
Line Encoding : NRZ | NRZ | NRZ | NRZ |
tx_ready     : 1 | 1 | 1 | 1 |
rx_ready     : 1 | 1 | 1 | 1 |
rx_freqlocked_1ms : 1 | 1 | 1 | 1 |
Serial Loop  : 0 | 0 | 0 | 0 |
Reverse Parallel : 0 | 0 | 0 | 0 |
Main Tap     : 35 | 35 | 35 | 35 |
Pre Tap 2 Level : 0 | 0 | 0 | 0 |
Pre Tap 1 Level : 5 | 5 | 5 | 5 |
Post Tap 1 Level : 0 | 0 | 0 | 0 |
RXEQ_HF_BOOST : 0 | 0 | 0 | 0 |
RXEQ_VGA_GAIN : 0 | 0 | 0 | 0 |
RXEQ_DFE_TAP 1 : 8 | 8 | 8 | 25 |
RXEQ_DFE_TAP 2 : -14 | -14 | -13 | -8 |
RXEQ_DFE_TAP 3 : -7 | -7 | -5 | -2 |
RXEQ_DFE_TAP 4 : 3 | -2 | 3 | -1 |
RXEQ_DFE_TAP 5 : -3 | 2 | -7 | -2 |
RXEQ_DFE_TAP 6 : -1 | -15 | -3 | -7 |
RXEQ_DFE_TAP 7 : -3 | 12 | 7 | 6 |
RXEQ_DFE_TAP 8 : 3 | -3 | -1 | 0 |
RXEQ_DFE_TAP 9 : 3 | 3 | 3 | 3 |
RXEQ_DFE_TAP10 : -1 | -3 | -1 | -1 |
RXEQ_DFE_TAP11 : -2 | -1 | -1 | -3 |
RXEQ_DFE_TAP12 : 1 | 1 | 0 | -1 |
RXEQ_DFE_TAP13 : -1 | 1 | 0 | 1 |
RXEQ_DFE_TAP14 : 3 | -1 | -3 | 3 |
RXEQ_DFE_TAP15 : -1 | -1 | 0 | -2 |
RXEQ_DFE_TAP16 : -2 | -2 | -3 | -2 |
FOM          : 237 | 239 | 233 | 233 |
VGA          : 21 | 23 | 22 | 21 |
CTLE         : 26 | 27 | 25 | 26 |
Convergence time : 136 | 112 | 142 | 94 | (ms) (only LSB portion)
Rx Flow State : 8 | 6 | 6 | 11 |
```

```
phy 1

Channel      : 0 | 1 | 2 | 3 |
              -----
FGT Quad     : 2 | 2 | 2 | 2 |
FGT Lane     : 3 | 2 | 1 | 0 |
Transceiver Type : FGT | FGT | FGT | FGT |
Line Encoding : NRZ | NRZ | NRZ | NRZ |
tx_ready     : 1 | 1 | 1 | 1 |
rx_ready     : 1 | 1 | 1 | 1 |
rx_freqlocked_1ms : 1 | 1 | 1 | 1 |
Serial Loop  : 0 | 0 | 0 | 0 |
Reverse Parallel : 0 | 0 | 0 | 0 |
Main Tap     : 35 | 35 | 35 | 35 |
Pre Tap 2 Level : 0 | 0 | 0 | 0 |
Pre Tap 1 Level : 5 | 5 | 5 | 5 |
Post Tap 1 Level : 0 | 0 | 0 | 0 |
RXEQ_HF_BOOST : 0 | 0 | 0 | 0 |
RXEQ_VGA_GAIN : 0 | 0 | 0 | 0 |
RXEQ_DFE_TAP 1 : 24 | 9 | 30 | 25 |
RXEQ_DFE_TAP 2 : -24 | -10 | -24 | -10 |
RXEQ_DFE_TAP 3 : 1 | -3 | -7 | -2 |
RXEQ_DFE_TAP 4 : -5 | 0 | -2 | -3 |
RXEQ_DFE_TAP 5 : -2 | -14 | -12 | -4 |
RXEQ_DFE_TAP 6 : 1 | 5 | 1 | 6 |
RXEQ_DFE_TAP 7 : 2 | 1 | 6 | 0 |
RXEQ_DFE_TAP 8 : -3 | -3 | 0 | -1 |
RXEQ_DFE_TAP 9 : 2 | 1 | 0 | 1 |
RXEQ_DFE_TAP10 : -1 | -1 | -6 | 0 |
RXEQ_DFE_TAP11 : -6 | -1 | -1 | 0 |
RXEQ_DFE_TAP12 : 1 | -1 | 0 | -3 |
RXEQ_DFE_TAP13 : 0 | -1 | 0 | 0 |
RXEQ_DFE_TAP14 : 0 | 0 | 1 | 0 |
RXEQ_DFE_TAP15 : -6 | 3 | -2 | -1 |
RXEQ_DFE_TAP16 : -1 | -6 | -2 | -3 |
FOM          : 233 | 236 | 239 | 234 |
VGA          : 20 | 22 | 19 | 18 |
CTLE         : 29 | 29 | 30 | 27 |
Convergence time : 106 | 110 | 107 | 135 | (ms) (only LSB portion)
Rx Flow State : 6 | 6 | 11 | 9 |
```


Sweep PMA function (electrical loopback) (option 'u')

```
Wait time after setting TX PMA setting : 10 seconds
Wait time after Rx reset                : 5 seconds

setting 0 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 0 , rx_ready : 0, no_lock, Vertical Eye : -1^[[19~
setting 1 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 1 , rx_ready : 0, no_lock, Vertical Eye : -1
setting 2 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 2 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 120
setting 3 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 3 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 124
setting 4 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 4 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 145
setting 5 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 5 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 154
setting 6 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 6 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 164
setting 7 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 7 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 166
setting 8 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 8 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 187
setting 9 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 9 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 195
setting 10 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 10 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 206
setting 11 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 11 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 217
setting 12 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 12 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 223
setting 13 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 13 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 228
setting 14 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 14 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 231
setting 15 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 15 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 165
setting 16 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 16 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 226
setting 17 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 17 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 229
setting 18 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 18 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 167
setting 19 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 19 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 228
setting 20 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 20 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 225
setting 21 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 21 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 223
setting 22 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 22 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 222
setting 23 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 23 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 223
setting 24 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 24 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 225
setting 25 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 25 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 233
setting 26 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 26 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 226
setting 27 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 27 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 234
setting 28 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 28 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 231
setting 29 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 29 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 236
setting 30 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 30 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 234
setting 31 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 31 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 235
setting 32 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 32 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 233
setting 33 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 33 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 240
setting 34 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 34 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 233
setting 35 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 35 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 242
setting 36 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 36 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 242
setting 37 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 37 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 235
setting 38 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 38 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 235
setting 39 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 39 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 232
setting 40 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 40 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 237
setting 41 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 41 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 237
setting 42 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 42 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 235
setting 43 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 43 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 238
setting 44 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 44 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 239
setting 45 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 45 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 239
setting 46 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 46 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 236
setting 47 Tx_phy 1 Rx_phy 0 Ch 0 : VOD : 47 , BER : 0.000000e+00, ErrorCount : 0.000000e+00 Vertical Eye : 243
Range : 47
Optimum range : 46
Ber_Minimum : 0.000000e+00
Start_value : 2
End_value : 47
sweep_low : 0
best_setting BER : 25
```

Continuous Update of Errorcount statistics (option 'c')

```
Phy 0 Ch 0 BER related statistics every 2 seconds
Press Enter to stop the loop
```

```
=====
>>>> HH:MM:SS,Core_Temp[0],Core_Temp[1],Core_Temp[2],Core_Temp[3],PrbsLockAlarm,BER,Errorcount(63:32),ErrorCount(31:0),Errorcount(float),Incremental Errorcount
>>>> 0h: 0m: 1s, 28.5, 27.6, 27.5, 27.0, 0, 0.000000e+00, 0, 0, 0.000000e+00, 0
>>>> 0h: 0m: 3s, 28.5, 27.6, 27.5, 27.2, 0, 0.000000e+00, 0, 0, 0.000000e+00, 0
>>>> 0h: 0m: 5s, 28.5, 27.2, 27.5, 27.2, 0, 0.000000e+00, 0, 0, 0.000000e+00, 0
>>>> 0h: 0m: 7s, 28.5, 27.5, 27.5, 27.0, 0, 0.000000e+00, 0, 0, 0.000000e+00, 0
>>>> 0h: 0m: 9s, 28.5, 27.6, 27.9, 27.2, 0, 0.000000e+00, 0, 0, 0.000000e+00, 0
>>>> 0h: 0m:11s, 28.9, 27.6, 27.9, 27.2, 0, 0.000000e+00, 0, 0, 0.000000e+00, 0
>>>> 0h: 0m:13s, 28.5, 27.5, 27.5, 27.2, 0, 0.000000e+00, 0, 0, 0.000000e+00, 0
```

Readout of the I2C information (not available on the PCIe devkit ES-version)

- Every time a status update is done all QSFP-DD modules will be checked if anything is plugged in and if this is the case the I2C information will be readout.
- See screenshot below showing various module and cables connected in the system.
- Information provided is Vendor, Part Number, temperature for modules and cable length for QSFP-DD cables or QSFP cables.
- One can dump the entire I2C Lower Page and Page 00h using option '{'

```
QSFPDD0 plugged in Vendor : Molex      Part : 2015911010      QSFP-DD Passive Cable with length 1.00 m
QSFPDD1 plugged in Vendor : Molex      Part : 2015911010      QSFP-DD Passive Cable with length 1.00 m
```

Board setup

- Make connections to the QSFPDD0 and QSFPDD1 (using loopback modules, DAC cable, optical modules etc.
- Connect micro-USB Cable to J8 in order to use onboard USB-Blaster II.
- Power on the board
- All designs so far use the default clock frequency of 156.25 Mhz. Not required to use clock control GUI to change clocks.
- Download 'Devkit_Demo.sof' to the AGIB027R29A1E2VR0 device using QII 21.4 programmer

Running the software on Windows10

This is when you want to run the software provided as .elf file (part of the deliverable) Do the following :

1. Start a Nios II command Shell (can be found in the Intel® Quartus® Prime Software installation (e.g. C:\quartus_21_4\nios2eds) and go to the directory where the .elf and run1.bsh files are located.
2. Since 21.4 Nios II terminal is using WSL, a quick way to go to the selected directory is to copy the complete path using windows explorer and in the Nios 21.4 terminal you type `cd "$(wslpath " => then you paste the path you copied (using right mouse click) => and end with typing ")"`
e.g. `cd "$(wslpath "C:\ALTERA\Reference_Designs\SUPERLITEII\Agilex_F-Tile\Agilex_PCie_Kit_SuperliteII_V4_2x4\software\app\Agilex_xCh")"`
3. Since the Intel Quartus archive is restored in windows the files need to be converted from Dos to Linux. To do so either use `"dos2unix -k *"` or run the provided .bsh file `"/convert.bsh"` (note option `-k` keeps the original date of the file intact).
4. Type In The Following Command : `"/run1.bsh"` This will set the JtagSpeed to 16M ,download the .elf file and start the downloaded Nios II code inside the main program memory.

Rebuilding the software on Windows 10

In order to rebuild the software follow the below steps (using QII 21.4 or later):

- Start a Nios II 21.4 Command Shell
 1. go to the `../software/app/Agilex_xCh` directory (which is part of the archive) using the method described on the previous slide
 2. Run the bash file `./convert.bsh` (this will also convert the files in the bsp folder)
 3. Run `./create-this-app`. This will compile the system libraries (in the bsp folder), compile the software code (main.c and a number of other .c files in this case) and produce a Nios II executable.
- If you make any changes to the software you need to rebuild by typing “make” in the Nios II 21.4 shell (after having created the software a first time)
- To run the software : type In The Following Command : `./run1.bsh` This will set the JtagSpeed to 16M and download the Nios II code to the FPGA and open the terminal.

Running the software on Linux

This is when you want to run the software provided as .elf file (part of the deliverable) Do the following:

1. Start a konsole with enough memory (e.g. `"arc submit -i node/"memory>=128000" --konsole"`)
2. Make sure all restored files are unix files and are executable (use `"/convert.bsh"`)
3. Open an acds shell (e.g. `"arc shell acds/21.4"`)
4. Start a Nios Command Shell (e.g. `"/p/psg/swip/releases/acds/21.4/67/linux64/nios2eds/nios2_command_shell.*"`) One can figure out the path using `"which quartus"` while being in the acds shell
5. Download the .elf file using command `"nios2-download -r -c 1 -d 1 -i "0" -g Agilex_xCh.elf"`
Note that this assumes the device is connected to USB-Cable with index 1. If another index is used, change accordingly.
6. Next use `"nios2-terminal -c 1 -d 1 -i "0" |tee output.log"` to run the software from program memory.

Rebuilding the software on Linux

In order to rebuild the software follow the below steps (using QII 21.4 or later):

- Start a Nios II 21.4 Command Shell using the steps on the previous slide
 1. go to the `../software/app/Agilex_xCh` directory (which is part of the archive)
 2. Run the bash file `./convert.bsh` (should already have been done)
 3. Run `./create-this-app`. This will compile the system libraries (in the bsp folder), compile the software code (main.c and a number of other .c files in this case) and produce a Nios II executable.
- If you make any changes to the software you need to rebuild by typing “make” in the Nios II 21.4 shell (after having created the software a first time)

To run the software :

1. Download the .elf file using command “`nios2-download -r -c 1 -d 1 -i "0" -g Agilex_xCh.elf`”. Note that this assumes the device is connected to USB-Cable with index 1. If another index is used, change accordingly.
2. Next use “`nios2-terminal -c 1 -d 1 -i "0" |tee output.log`” to run the software from program memory.

Using Transceiver Toolkit (need patch on top of 21.4)

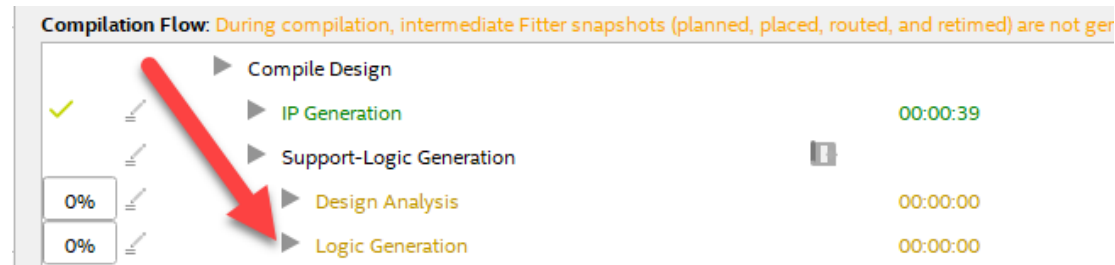
- To start the Transceiver Toolkit simply start it from the Tools Menu in the Intel Quartus Prime Software (with the project loaded). This will automatically detect the design and start up the Transceiver Toolkit which allows you to run further tests in optimizing the PMA settings, do autosweeps etc.
- Make sure to stop all traffic running in TTK and close the TTK before going back to the Nios terminal to continue with the design (If the hard PRBS generator and checker have been enabled during testing those channels will show errors (which is normal). Using option 'q' these errors can be cleared

Signaltap II Debug Example

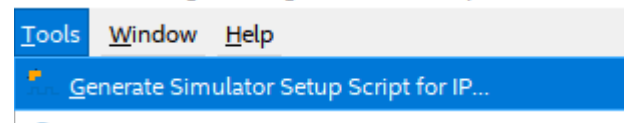
- The design contains a number of signaltap instances
- Signaltap II acquisition can be done simultaneously with Nios II control.

How to run a simulation using Questasim/Modelsim (1/4)

1. Open the project using Intel Quartus Prime Software and in directory <restored_project>/simulation/mentor (DO NOT USE THE ORIGINAL PROJECT AFTER RESTORING THE ARCHIVE), the simulation has its own project
2. Make sure the IP settings are set correctly (see slide (3/4). IP settings are not fully governed by .qsf only, hence the additional check to be done to make sure they are correctly set
3. Press “Logic Generation” (see screenshot below) (this generates the __tiles files which are mandatory for the simulation) and also regenerates all .ip files and .spd files required for the simulation.

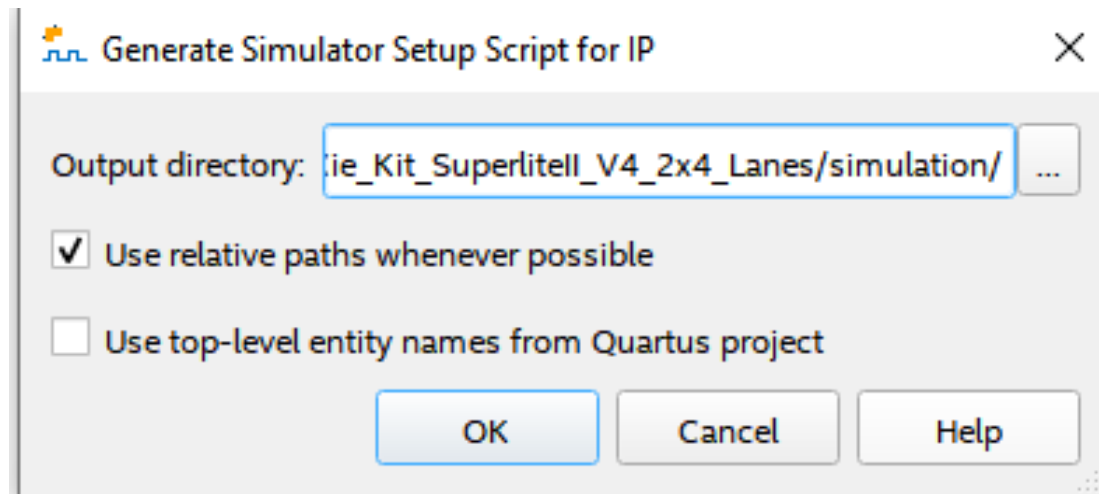


4. From the Intel Quartus Prime Software Menu Start Tools => Generate Simulator Setup Script for IP...

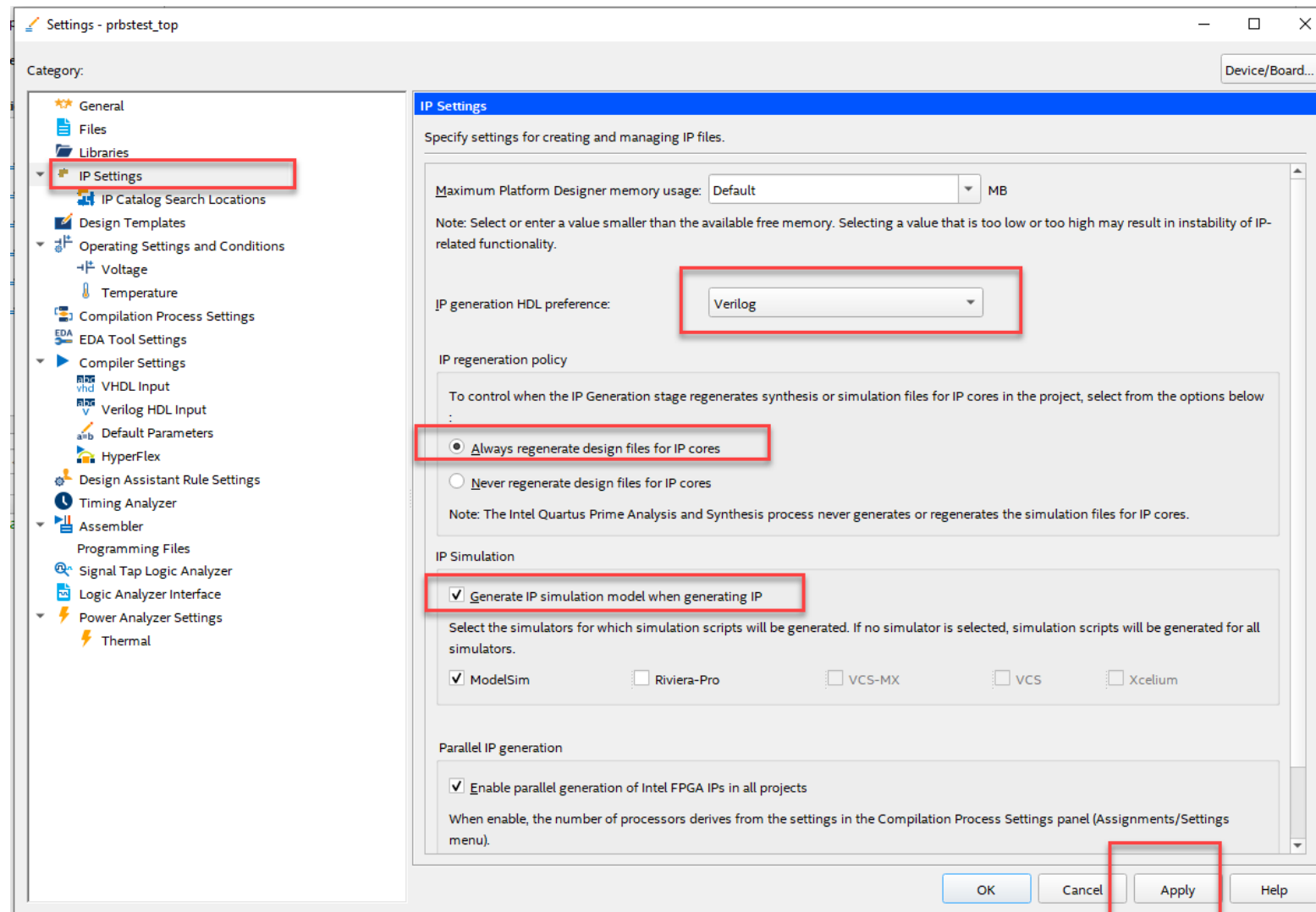


How to run a simulation using Questasim/Modelsim (2/4)

5. In the window that pops-up, remove the mentor in the path name so that you get <restored_project>/simulation/
 - Make sure “use top-level entity names from Intel Quartus Prime Software project” is not selected (in 21.4 this is already remove as an option)



How to run a simulation using Questasim/Modelsim (3/4)



How to run a simulation using Questasim/Modelsim (4/4)

5. Go to the simulation/mentor folder and launch questasim e.g. using the following command *arc submit -i questasim/2020.4 questasim_sver-lic priority=100 -- "vsim -i -64"*
6. Type "sim.do" => this will do the entire simulation.
7. In case modifications are made to the source files you can rerun the simulation using "resim.do"

Deliverables

- Intel® Quartus® 21.4 B67 archive which contains the demo design files, the software source files, files required to run the simulation
- The testbenches and project files required for simulation can be found after restoring the archive: <restored_project>/simulation/mentor
- The software source files can be found after restoring the archive in the <restored_project>/software/app and <restored_project>/software/bsp folders/
- SOF File
- Devkit_demo.elf File (Software)
- Readme.txt file

Status & Revision

- V21.4 : January 10th, 2022
 - Modified RTL implementation to use the same control and status bits as the SuperliteIV demo design so that the same main.c file used for all other designs (PRBS (with and without FEC), Superlite IV and Superlite II can be used).
- V21.4 : November 29th , 2021
 - Initial release of the design
 - Successfully tested with 2.5 meter DAC cable and electrical loopback

