WWW.ESISTREAM.COM

ESIstream

The **E**fficient **S**erial **I**nterface

ESIstream XILINX KU040 V2-1

Version 2.1

ESIstream

The **E**fficient **S**erial **I**nterface

ESIstream is an Open-source high efficiency serial interface protocol based on 14b/16b encoding.

Its main benefits are low overhead and ease of hardware implementation.

Reference documents

ESIstream Protocol specification V2.0: www.ESIstream.com
EV12AQ600 datasheet: website link, download link

Document aim and comment

This document aims at explaining the architecture, the design,the environment of simulation and the environment of compilation of ESIstream transmitter (TX) and receiver (RX) IP package.

To make the adoption of ESIstream easier and to speed up the integration into customers' products, the ESIstream IP package include all the source code for both ends of the link transmitter (Tx) and receiver (Rx). A fully implementable project supporting Xilinx FPGA Kintex Ultrascale (KU040). Furthermore, the ESIstream IP package include detailed test-bench simulation for both the Tx and Rx.

This package allows testing of the communication with a lane rate up to 12.8Gbps.
Although the current IP requires a Xilinx FPGA KU040, the modular architecture allows an easy migration of the ESIstream IP to a different Xilinx target replacing the FPGA specific IPs.

Following the user guide section of this document, user will be able to generate two implementable projects. One using a FMC XM107 loopback test board and the other one using a FMC EV12AQ600 test board, both project are based on a Xilinx KCU105 evaluation kit.

A test-bench simulation is also available with the FMC XM107 loopback test board project using Vivado simulator.

For technical support on ESIstream, contact: GRE-HOTLINE-BDC@Teledyne.com

Packages supported by this document

ESISTREAM_XILINX_KU040_V2-1_32b
ESISTREAM_XILINX_KU040_V2-1_64b

Introduction to ESIstream

ESIstream protocol initiated by Teledyne-e2v is born from a severe need of the following combination:

- Reduced data overhead on serial links, as low as possible.
- Increased rate of useful data when linking ADCs operating at GSPS speeds with FPGAs on a serial interface.
- Simplified hardware implementation, simple enough to be built on RF SiGe technologies.

ESIstream provides an efficient High-Speed serial 14b/16b interface. It is open-source and supports in particular serial communication between FPGAs and High-Speed data converters.

An ESIstream system is made up of a transmitter and a receiver.

- A transmitter can be an ADC or an FPGA or an ASIC
- A receiver can be a DAC or an FPGA or an ASIC
- A number of lanes (L ≥ 1) to transmit serial data
- A synchronization signal (sync) to initialize the communication. Clock embedded in each lane data stream (need to be recovered by RX).

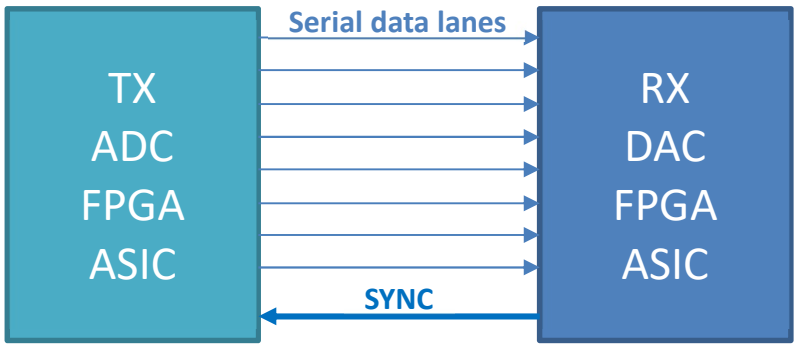


Figure 1: Basic ESIstream system

The main keys benefits are:

- FLEXIBILITY: Open-source and license free
- EFFICIENCY: 87.5%, with 14 bit of useful data and 2 bit overhead (clock bit and disparity bit).
- SIMPLICITY: Minimal hardware implementation.

The main specifications of ESIstream are:

- Deterministic latency
- Multi-device synchronization
- Demonstrated lane rate up to 12.8Gbps, depending on device abilities
- Multi-lanes synchronization
- Guaranteed DC balance transmission, ±16 bit running disparity
- Synchronization monitoring, using the clock bit (overhead bit)
- Sufficient number of transitions for CDR, max run length of 32.

Disclaimer

TABLE OF CONTENTS

FIGURES

Issue	Date	Comments
1.0	June 2019	Creation
2.1	October 2019	Publication

1. DESIGN GUIDE

1.1 Transmitter IP (TX)

1.1.1 Description

1.1.1.1 Architecture

The transmitter IP allows generation of the synchronization sequence on a sync pulse event (alignment and prbs) and encoding of the input data according to the ESIstream protocol specification (scrambling processing, clock bit concatenation, disparity bit processing and concatenation).
The encoding block encapsulates the encoding ESIstream protocol layer and makes the interface between ESIstream TX IP incoming data to encode and the encoded data to transmit through the transceiver IP.

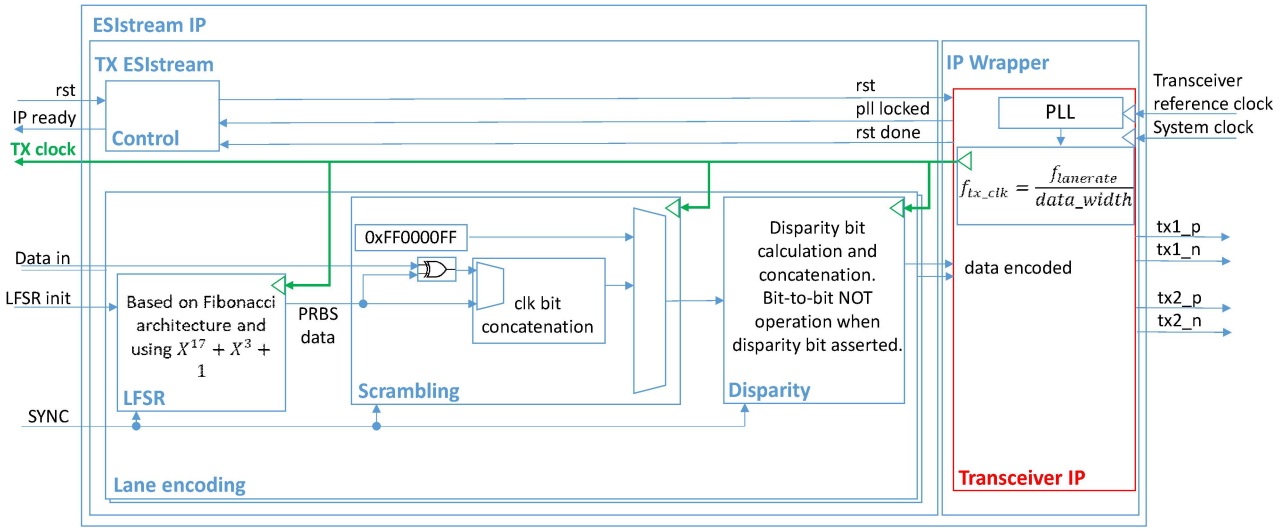


Figure 2: Block diagram of ESIstream TX IP with 2 lanes

The FPGA manufacturer provides the transceiver IP (in **red** in the above block diagram), and the transceiver IP can change depending on the FPGA target device manufacturer and reference.

Module	.vhd file name	Description
ESIstream IP	tx_esistream + rx_tx_xcvr_wrapper	Serializes and transmits data, using an Intel FPGA transceiver IP, on the differential serial lane outputs (tx_n / tx_p).
TX ESIstream	tx_esistream	For each lane, encodes useful data (data_in signal) into an ESIstream frame vector (data_encoded signal).
Control	tx_control	Manages and monitors transceiver PLL(s) lock, reset, reset done, user ready and ip ready signals.
Encoding	tx_encoding	When SER_WIDTH = 32 : Encodes useful data 2x14-bits (data_in signal) into a 2x16-bits ESIstream frame vector (data_encoded signal) according to the ESIstream protocol specification. When SER_WIDTH = 64 : Encodes useful data 4x14-bits (data_in signal) into a 4x16-bits ESIstream frame vector (data_encoded signal) according to the ESIstream protocol specification.

LFSR	lfsr	Generates pseudo-random binary sequence (PRBS) to scramble data using a linear feedback shift register (LFSR) based on a Fibonacci architecture and using the polynomial $X^{17} + X^3 + 1$. Applying scrambling ensures a statistical DC balanced transmission. In case of multi-lane interfaces in order to reduce correlation between lanes, each lane may have different initial values for scrambling units (LFSR init input signal)
Scrambling	tx_scrambling	When a sync event occurs, generates the synchronization sequence for receiver frame alignment and PRBS initialization. When SER_WIDTH = 32 : it scrambles 2x14-bit ESIstream data with a pseudo-random binary sequence generated by the LFSR module and concatenates the overhead clock bit to each data stream. When SER_WIDTH = 64 : it scrambles 4x14-bit ESIstream data with a pseudo-random binary sequence generated by the LFSR module and concatenates the overhead clock bit to each data stream.
Disparity	tx_disparity	When SER_WIDTH = 32: Calculates and processes disparity (bit-to-bit NOT) for 2x16-bit ESIstream frames concatenated with the overhead disparity bit. When SER_WIDTH = 64: Calculates and processes disparity (bit-to-bit NOT) for 4x16-bit ESIstream frames concatenated with the overhead disparity bit. (ESIstream protocol specification - ANNEX C: Example of disparity bit implementation).
Disparity submodule	tx_disparity_word_16b	Calculates the current word disparity for a 16-bit ESIstream frame. (ESIstream protocol specification - ANNEX C: Example of disparity bit implementation).
IP Wrapper	rx_tx_xcvr_wrapper	Contains transceiver IP and provides a generic entity interface with it.

1.1.1.2 Synchronization sequence

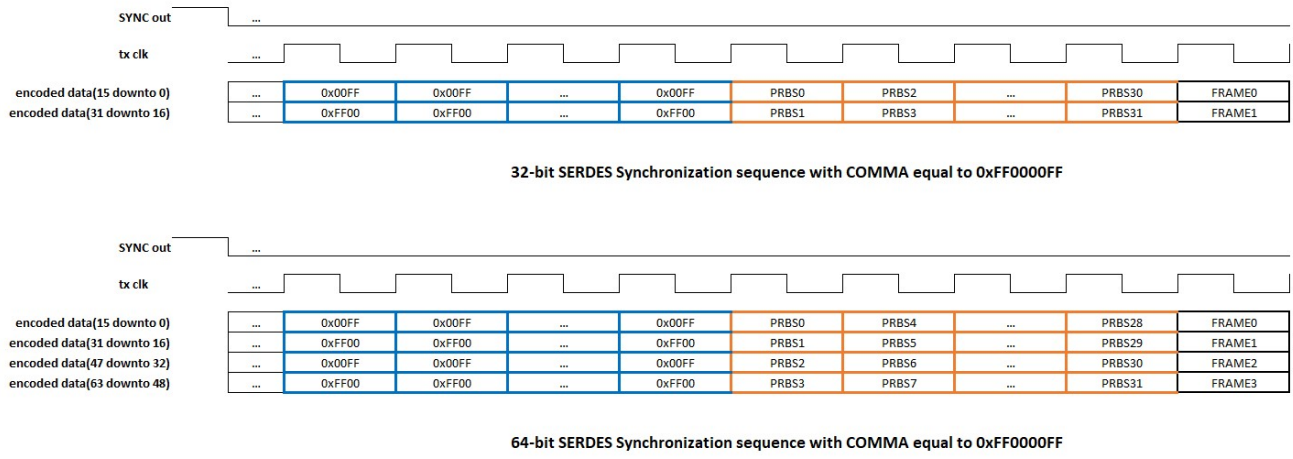


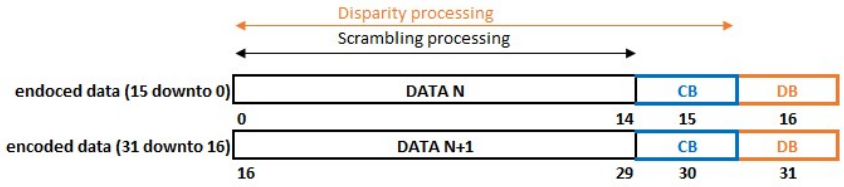
Figure 3: Synchronization sequence

When the transmitter receives a synchronization pulse, it will send a frame alignment sequence of 32x16-bit frames alternating between 0x00FF and 0xFF00 if COMMA equal to 0xFF0000FF with no scrambling and no disparity processing applied. The receiver can used the alignment pattern 0xFF0000FF or 0x00FFFF00 (COMMA) to align the received data and then to rebuild the frames sent by the transmitter.

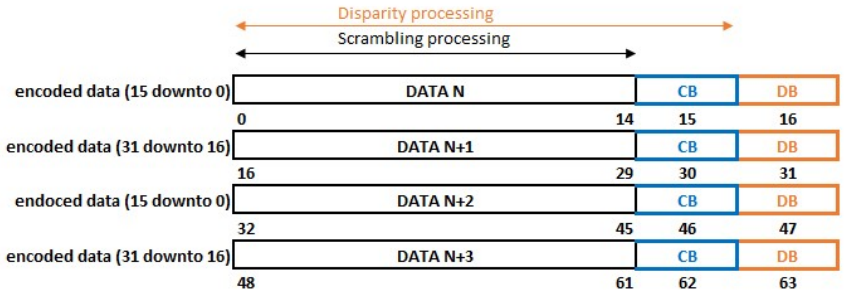
During the second part of the sequence it will send 32x16-bit additional frames containing PRBS values in the 14-bit data field plus two undefined overhead bits. This sequence allows initializing the receiver LFSR to generate the PRBS used by the descrambling process.

1.1.1.3 Encoded data

After the synchronization sequence, transmitter sends scrambled data plus overhead bits (clock bit & disparity bit).



32-bit SERDES encoded data



64-bit SERDES encoded data

Figure 4: ESIstream encoded data

1.1.2 Transmitter entity

1.1.2.1 Package

The ESIstream IP and sub-modules use a common package (*esistream_pkg.vhd*) to share signal types, functions (LFSR…), and constants (DESER_WIDTH, SER_WIDTH…).

1.1.2.2 Generic description

Generic	Type	Values	Description
NB_LANES	natural	1 to (FPGA maximum number of transceivers)	Number of lanes
COMMA	std_logic_vector	x"00FFFF00" or x"FF0000FF"	Frame alignment synchronization pattern.

1.1.2.3 Port description

Port	IN / OUT	width	Description
rst	Input	1-bit	Active high asynchronous reset.
rst_xcvr	Ouput	NB_LANES array of 1-bit	Active high transceiver asynchronous reset
xcvr_pll_lock	Input	NB_LANES array of 1-bit	Active high transceiver pll lock.
tx_rstdone	Input	NB_LANES array of 1-bit	Active high transceiver reset done
tx_usrclk	Input	1-bit	Transceiver user clock (frame clock) from transceiver.
xcvr_data_tx	Output	SER_WIDTH*NB_ LANES bit vector	Transceiver user data to transceiver.
tx_usrrdy	Output	NB_LANES array of 1-bit	Indicates that tx_esistream (user) is ready. Not used
sync_in	Input	1-bit	Active high pulse. Generates the synchronization sequence for the receiver, frame alignment and PRBS initialization.
prbs_en	Input	1-bit	Active high, enables scrambling processing.
disp_en	Input	1-bit	Active high, enables disparity processing.
lfsr_init	Input	NB_LANES array of 17-bit	Selects LFSR initialization value for each lane.
data_in	Input	NB_LANES array of 14-bit x SER_WIDTH/16	Useful data to encode. - When SER_WIDTH = 32: 2x14-bit at each tx_clk cycle. o data_in(NB_LANES_index)(0) : Data N o data_in(NB_LANES_index)(1) : Data N+1 - When SER_WIDTH = 64: 4x14-bit at each tx_clk cycle. o data_in(NB_LANES_index)(0) : Data N o data_in(NB_LANES_index)(1) : Data N+1 o data_in(NB_LANES_index)(2) : Data N+2 o data_in(NB_LANES_index)(3) : Data N+3
ip_ready	output	1-bit	Active high, when transceiver PLL(s) locked and transceiver reset done. Indicates that IP is ready to receive a sync pulse.

1.2 ReceiveR IP (RX)

1.2.1 Description

1.2.1.1 Architecture

The receiver IP sends the sync event to the transmitter and waits for the synchronization sequence to align the lanes and to initialize the LFSR. Then the receiver decodes the deserialized transceiver raw data according to the ESIstream protocol specification (descrambling processing, disparity bit processing).
The lane decoding block encapsulates the decoding ESIstream protocol layer and makes the interface between the encoded data received through the transceiver IP and ESIstream RX IP decoded data at the buffer output.

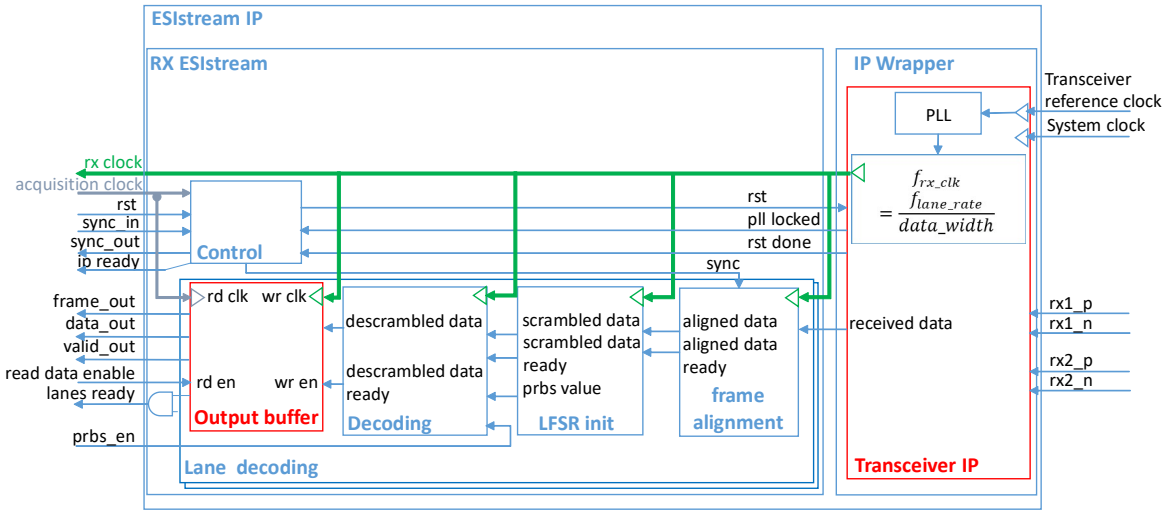


Figure 5: Block diagram of ESIstream RX IP with 2 lanes

The FPGA manufacturer provides the transceiver IP and the output buffer IP (in **red** in the above block diagram).
User should replace these two IP blocks according to the targeted FPGA.

Module	.vhd file name	Description
ESIstream IP	rx_esistream_with_xcvr	Deserializes and receives data using an Intel FPGA transceiver IP from the differential serial lane inputs (rx_n / rx_p).
RX ESIstream	rx_esistream	For each lane: - When DESER_WIDTH=32, Decodes 32-bits raw data (2x16-bits ESIstream encoded frame vector) to provide decoded useful data 2x14-bits (data_out signal) and related overhead clock bit and disparity bit (through frame_out) when valid_out is high. - When DESER_WIDTH=64, Decodes 64-bits raw data (4x16-bits ESIstream encoded frame vector) to provide decoded useful data 4x14-bits (data_out signal) and related overhead clock bit and disparity bit (through frame_out) when valid_out is high.
Control	rx_control	Manages and monitors sync, transceiver PLL(s) lock, reset, reset done, user ready and ip ready signals.
Lane decoding	rx_lane_decoding	- When DESER_WIDTH = 32, Decodes useful data 2x14-bits, from a 2x16-bits ESIstream frame vector (32-bits) from transceiver output

		according to the ESIstream - When DESER_WIDTH = 64, Decodes useful data 4x14-bits, from a 4x16-bits ESIstream frame vector (64-bits) from transceiver output according to the ESIstream.
Frame alignment	rx_frame_alignment	After a sync event, waits and detects frame alignment synchronization sequence. Indicates to the LFSR init module to start PRBS initialization sequence when frames aligned.
LFSR init	rx_lfsr_init	After a sync event, waits for frames aligned event to initialize and synchronize the PRBS with scrambled data.
Decoding	rx_decoding	Applies descrambling and reverse disparity processing on each frame.
IP Wrapper	rx_xcvr_wrapper	Contains transceiver IP and provides a generic entity interface with it.
output buffer	rx_output_buffer_wrapper	Provides a generic interface for the output buffer. Embeds Intel FPGA IP (dual clock FIFO). Frames buffering until all lanes ready and read data enable signal asserted. Frames buffering always begins with the first frame of the PRBS initialization sequence. Independent clocks (rx_clk, wr_clk) built-in fifo, 16-bit width. rd_clk and wr_clk should be at the same synchronous frequency. Fifo not empty indicates synchronized data are available at buffer outputs. Acquisition clock (acq_clk) and rx_clk must be synchronous. Default rx_clk can be connected

1.2.1.2 Frame alignment

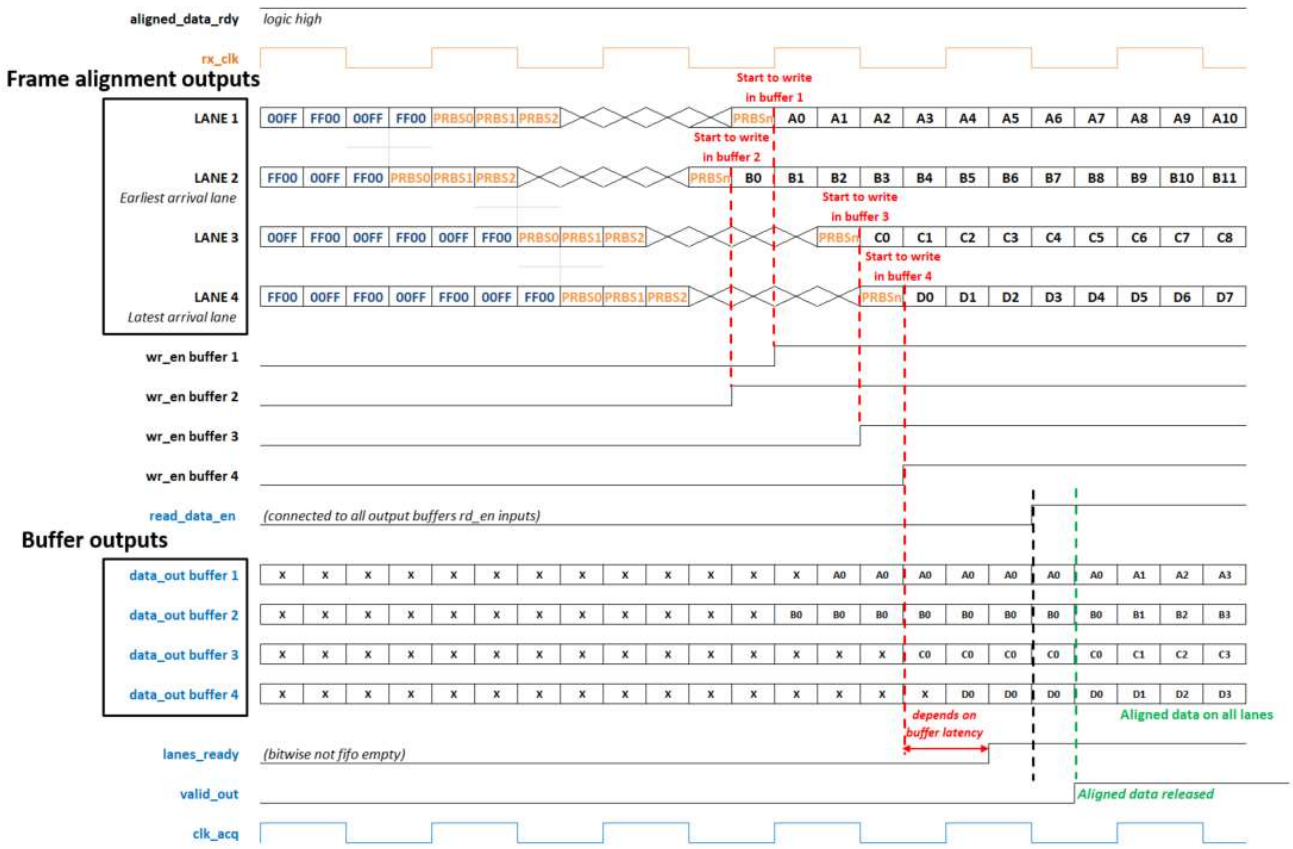


Figure 6: Frame alignment and output buffers alignment

1.2.1.3 Synchronization sequence

See section 2.1.2.

1.2.1.4 Received data

After the synchronization sequence, transmitter send scrambled data plus overhead bits (clock bit & disparity bit).

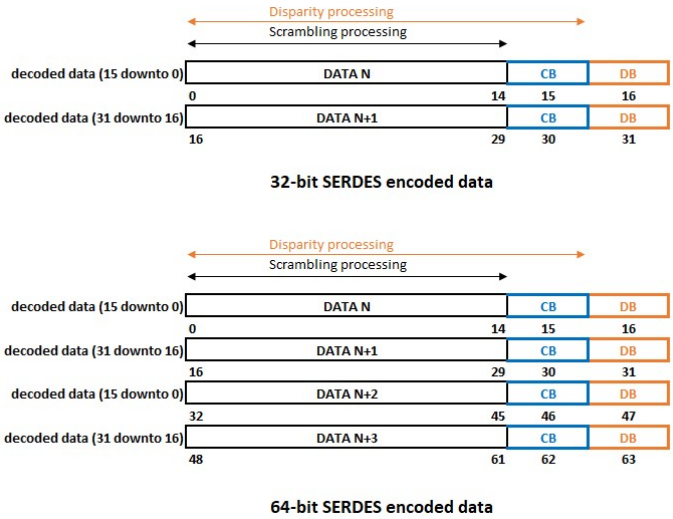


Figure 7: ESIstream received data

1.2.2 Receiver entity

1.2.2.1 Package

The ESIstream IP and sub-modules use a common package (*esistream_pkg.vhd*) to share signal types, functions (LFSR…), and constants (DESER_WIDTH, SER_WIDTH…).

1.2.2.2 Generic description

Generic	Type	Values	Description
NB_LANES	natural	1 to (FPGA maximum number of transceivers)	Number of lanes
SYNC_DELAY	natural	1 to n	Delay to add between sync output and sync process in lane decoding module.
COMMA	std_logic_vector	x”00FFFF00” or x”FF0000FF”	Frame alignment synchronization pattern.

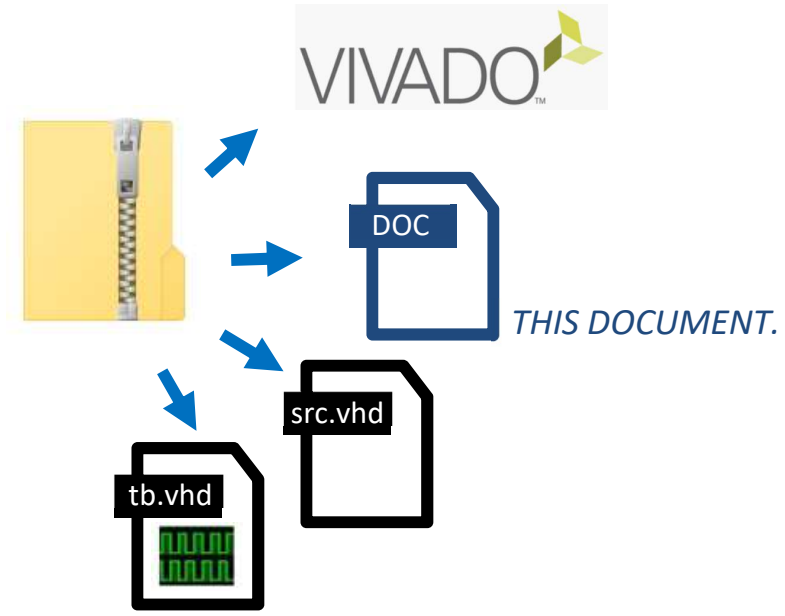
1.2.2.3 Port description

Port	IN / OUT	width	Description
rst	Input	1	Active high transceiver PLL asynchronous reset
rst_xcvr	Output	1	Active high transceiver asynchronous reset
rx_rstdone	Input	NB_LANES array of 1-bit	Active high transceiver reset done.
xcvr_pll_lock	Input	NB_LANES array of 1-bit	Active high transceiver pll lock.

rx_usrclk	Input	1	Transceiver user clock (frame clock) from transceiver.
xcvr_data_rx	Inpit	SER_WIDTH*NB_ LANES bit vector	Transceiver user data from transceiver.
sync_in	input	1	Active high pulse. Generates the synchronization sequence for the receiver, frame alignment and PRBS initialization.
prbs_en	input	1	Active high, enables scrambling processing.
lanes_on	Input	NB_LANES array of 1	Active high, enable lane. If lane disabled, then no data is available in corresponding lane output buffer.
read_data_en	input	1	Active high, enables read of data at buffers outputs. When enabled and when valid_out output is high, received data is streamed on frame_out and data_out.
clk_acq	input	1	Acquisition clock, output buffer read port clock, should be the same frequency with no phase drift with respect to receive clock (default: clk_acq should take rx_clk).
sync_out	output	1	Active high pulse. Connect to transmitter sync_in to generate the synchronization sequence for the receiver, frame alignment and PRBS initialization
frame_out	output	NB_LANES array of 16 x DESER_WIDTH/16	Decoded ESIstream frame: disparity bit (15) & clk bit (14) & data (13 down to 0) = frame_out(15 downto 0).
data_out	output	NB_LANES array of 14 x SER_WIDTH/16	Decoded useful data. data_out = frame_out(13 downto 0).
valid_out	output	NB_LANES array of 1	Active high one rx_clk period later than read_data_en, frame_out and data_out stream valid output.
ip_ready	output	1	Active high, when transceiver PLL(s) locked and transceiver reset done. Indicates that IP is ready to receive a sync pulse.
lanes_ready	output	1	Active high, indicates lanes synchronization status. When high, all enabled lanes are synchronized and data are available at output_buffer(s) outputs.

2. USER GUIDE

2.1 Package content



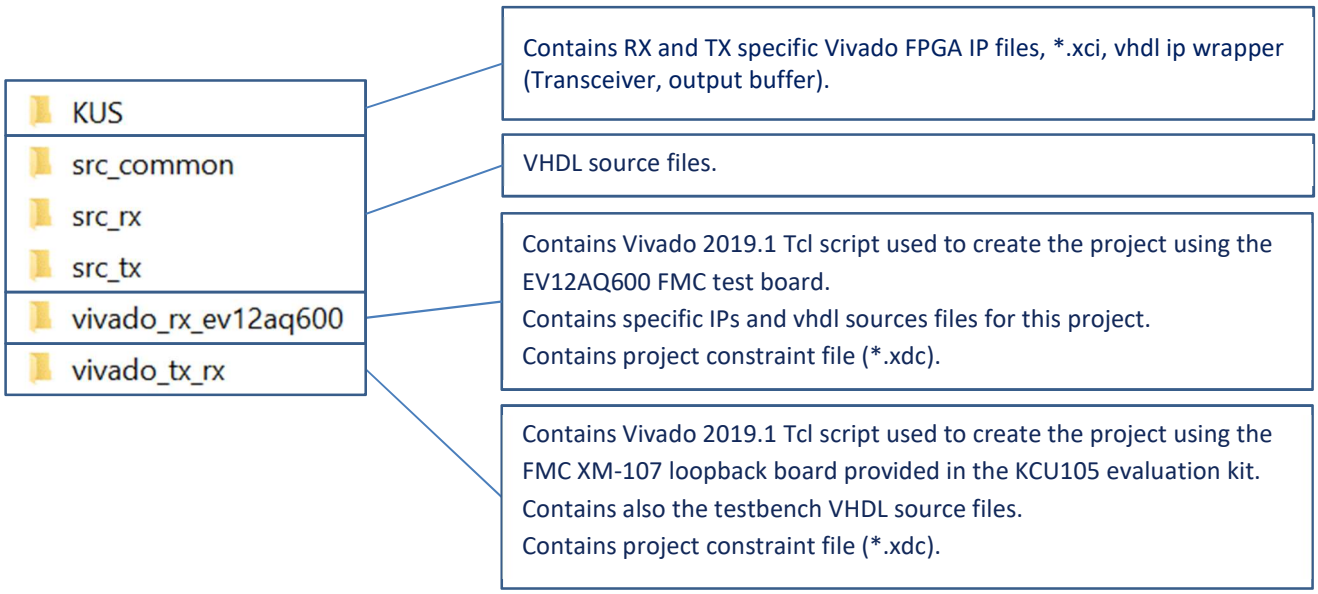
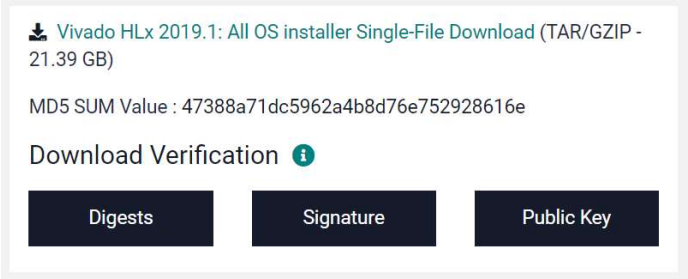


Figure 8: ESIstream package overview

2.2 Softwares and download links

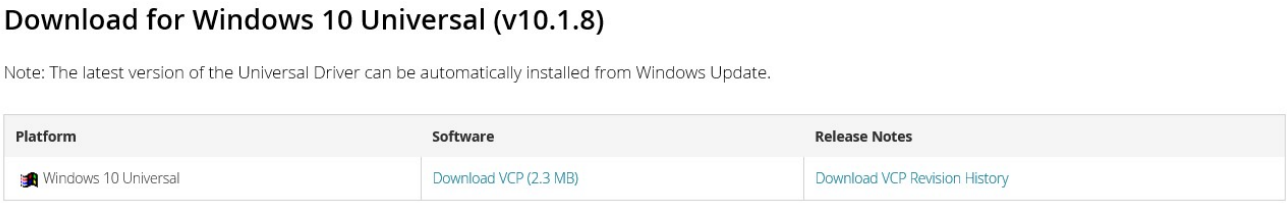
2.2.1 Vivado

- Donwload and install **Vivado Design Suite – HLx Editions - 2019.1 Full product Installation.**
  - Use the link: https://www.xilinx.com/support/download.html

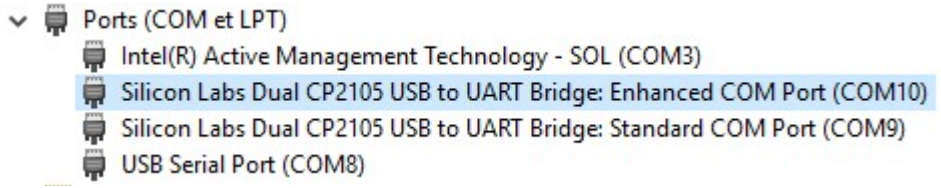


2.2.2 CP201x USB to UART Bridge VCP Drivers

- Download and install **CP210x USB to UART Bridge VCP Drivers**.
  - Use the link: https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers



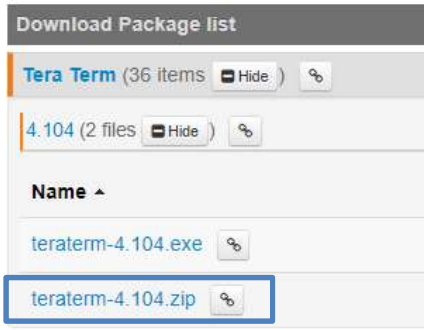
  - To determine the port COM number of the CP210x :
    - Open the device manager:



  - There will be two “Silicon Labs Dual CP210x” COM ports, Standard and Enhanced
    - The Standard COM Port is the FPGA UART COM Port
      - **not used** in this project
    - The **Enhanced** COM Port is the System Controller COM Port
      - **Used** in this project to configure the **Si570** and **Si5328** frequencies.
      - The Si5328 output is connected to the transceivers reference clock input.
  - The COM Port numbers will vary from system to system

2.2.3 Tera Term

- Download Tera Term terminal
  - Use the link: https://osdn.net/projects/ttssh2/releases/
    - Download the *.zip file.

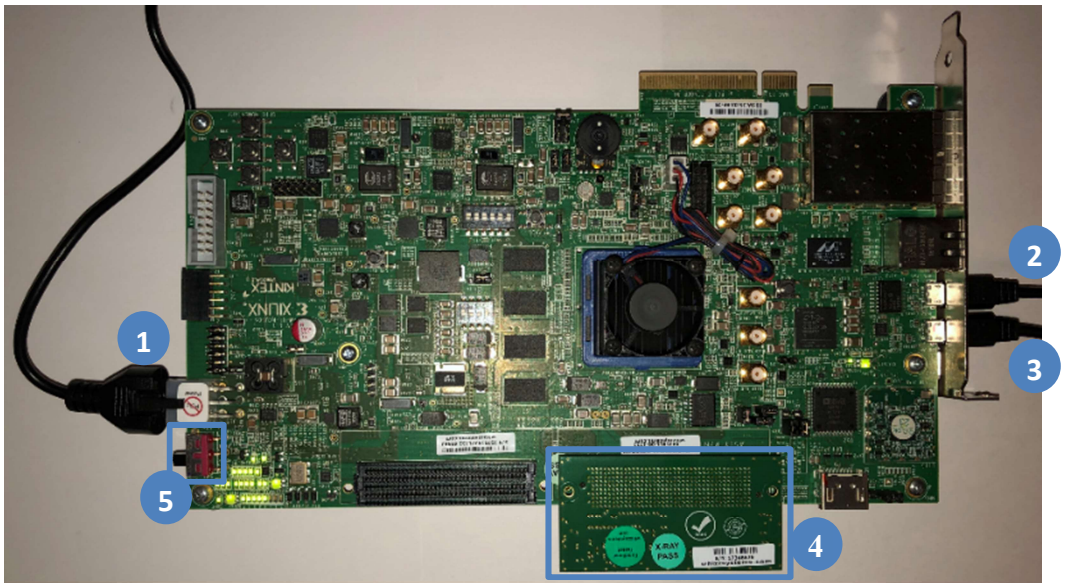


    - No installation required, just unzip the file.

2.3 FMC XM107 Loopback project setup

2.3.1 Hardware setup

- **Warning: To avoid power supplies hot plug, check all power supplies switches positions are OFF.**
- **Warning: Use the correct hardware configuration for each bitstream.**



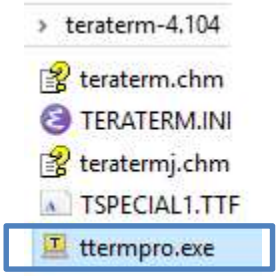
1- Connect the power supply cable (+12V / 5A) to the power supply connector on the KCU105 board.
2- Connect a USB Type-A to Micro-B cable to the USB JTAG (J87) connector on the KCU105 board and to your PC.
3- Connect a USB Type-A to Micro-B cable to the USB UART connector (J4) on the KCU105 board and to your PC.
4- Connect the FMC XM107 board to the FMC HPC connector (J22).
5- Power-on the KCU105 board.

2.3.2 Software setup

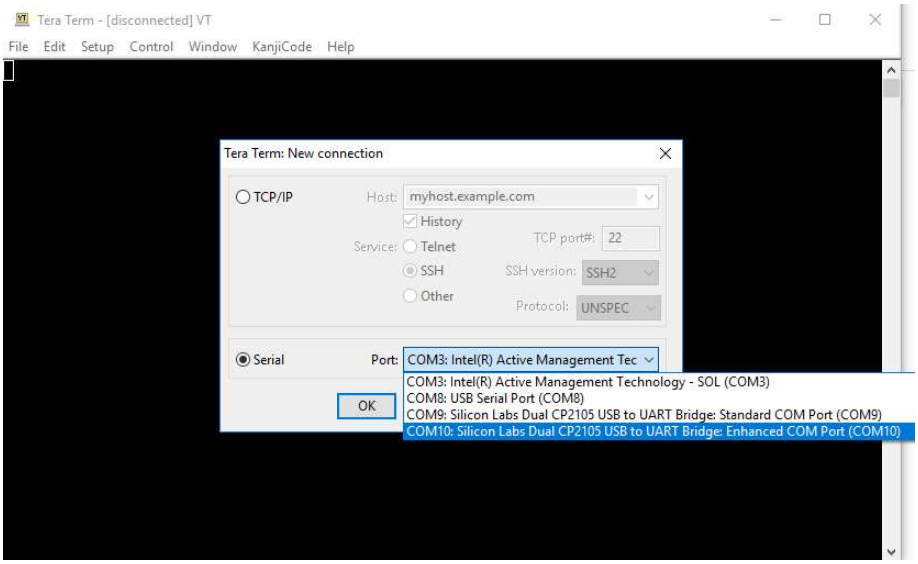
2.3.2.1 Transceiver reference clock configuration

Do this procedure before programming the FPGA when working with the FMC XM107 loopback vivado project (*vivado_tx_rx/script_common_gth.tcl*).

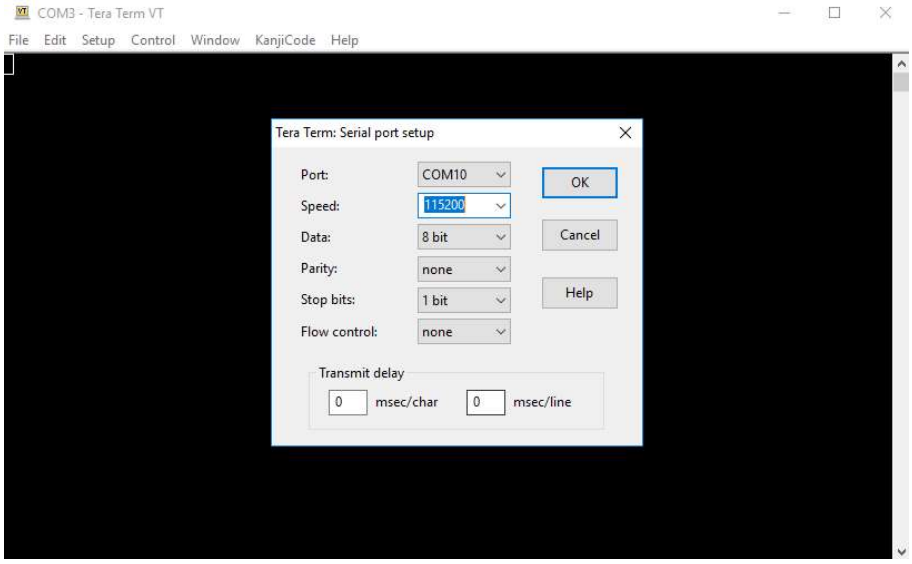
- Open a Tera Term terminal
  - Using *ttermpro.exe* in the unzipped directory (see section 4.1.3 Tera Term)



- Select the new connection as *Serial* and use the *Enhanced COM Port*

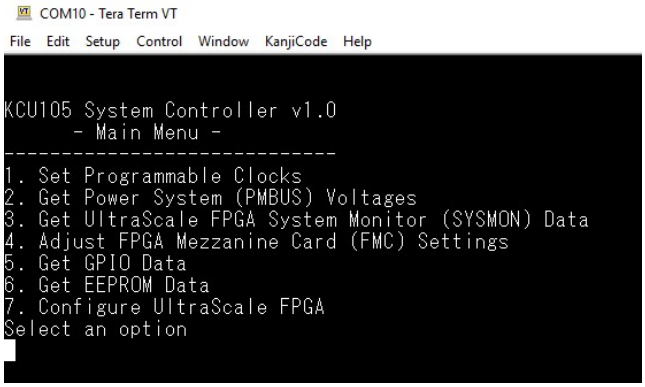


- Configure UART settings :
  - Go to Setup > Serial Port…
  - Set parameters as show below:

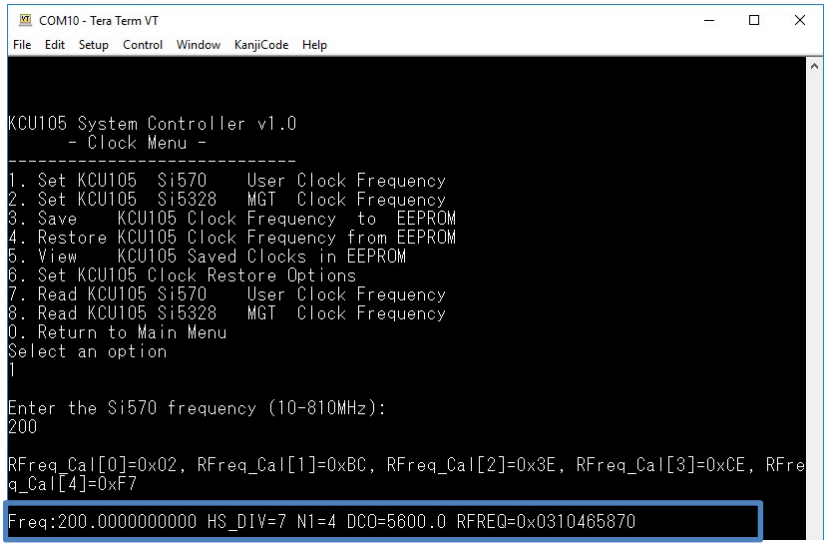


  - OK

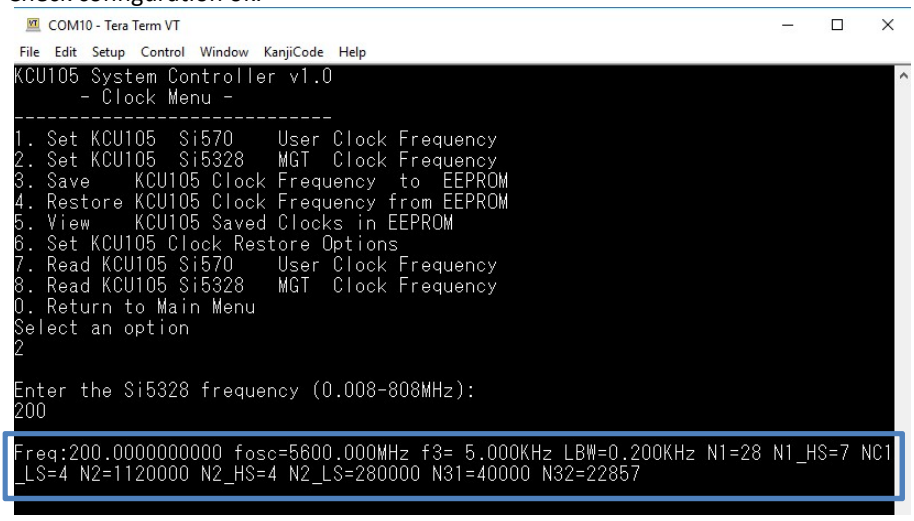
- Press Enter to display the Main Menu:



- Press 1
- Press 1 again
  - Enter 200 to configure Si570 frequency.
  - Check configuration ok.



- Press 2
  - Enter 200 to configure Si5328 frequency.
  - Check configuration ok.



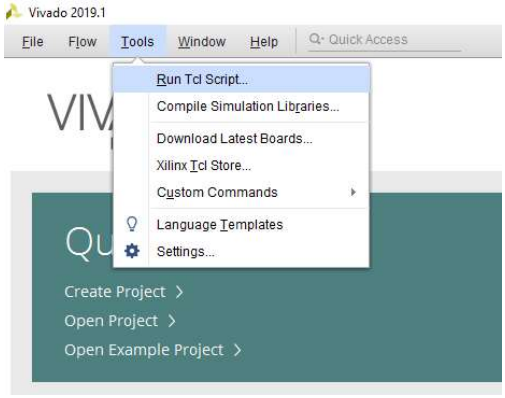
- End of clocks configuration.

2.3.2.2 Generate FMC XM107 loopback vivado project

A tcl script ("script.tcl") is provided with the sources and can be used to generate the project using Vivado Xilinx software. It is recommended to use Vivado 2019.1 as updating the IP (in particular the GTH) to a more recent version may cause issue.

- Open Vivado 2019.1

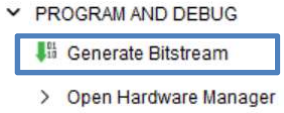
- Tools > Run Tcl Scripts…
  - Launch the package Tcl script located in the directory ***vivado_tx_rx/script_common_gth.tcl***



When the project has been created and the IP generated, you can start compiling, simulating or modifying the example design like any Vivado project.
Note: If the TCL script is moved from its original folder it will not work as it uses relative path.

2.3.2.3 Generate bitstream

- Click on *Generate Bitstream* in the *Flow Navigator* panel (*PROGRAM AND DEBUG*).



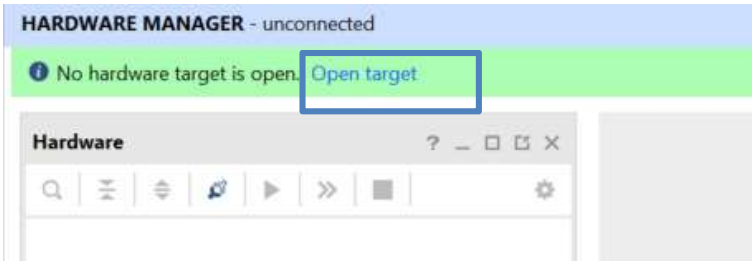
- Wait end of bitstream generation.
  - Check Synthesis, Implementation end without critical warnings.
  - Check there is no Timing error.

2.3.2.4 Test procedure

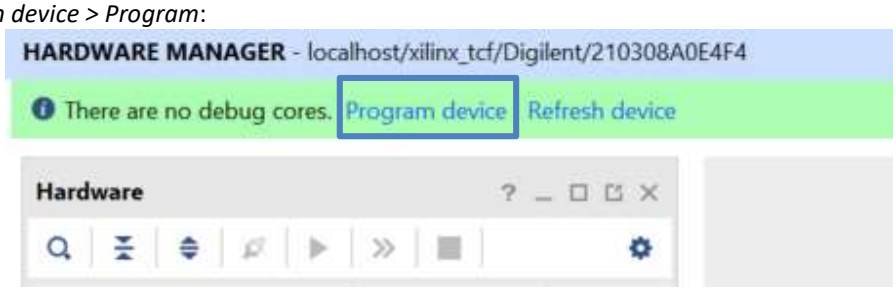
Once the bitstream is generated, the KU040 FPGA embedded on the KCU105 board can be loaded using the Vivado Hardware Manager.

- Program the FPGA: *Flow Navigator > PROGRAM AND DEBUG*
  - Click on *Open Hardware Manager*
    - KCU105 must be powered ON.
    - JTAG cable must be connected.

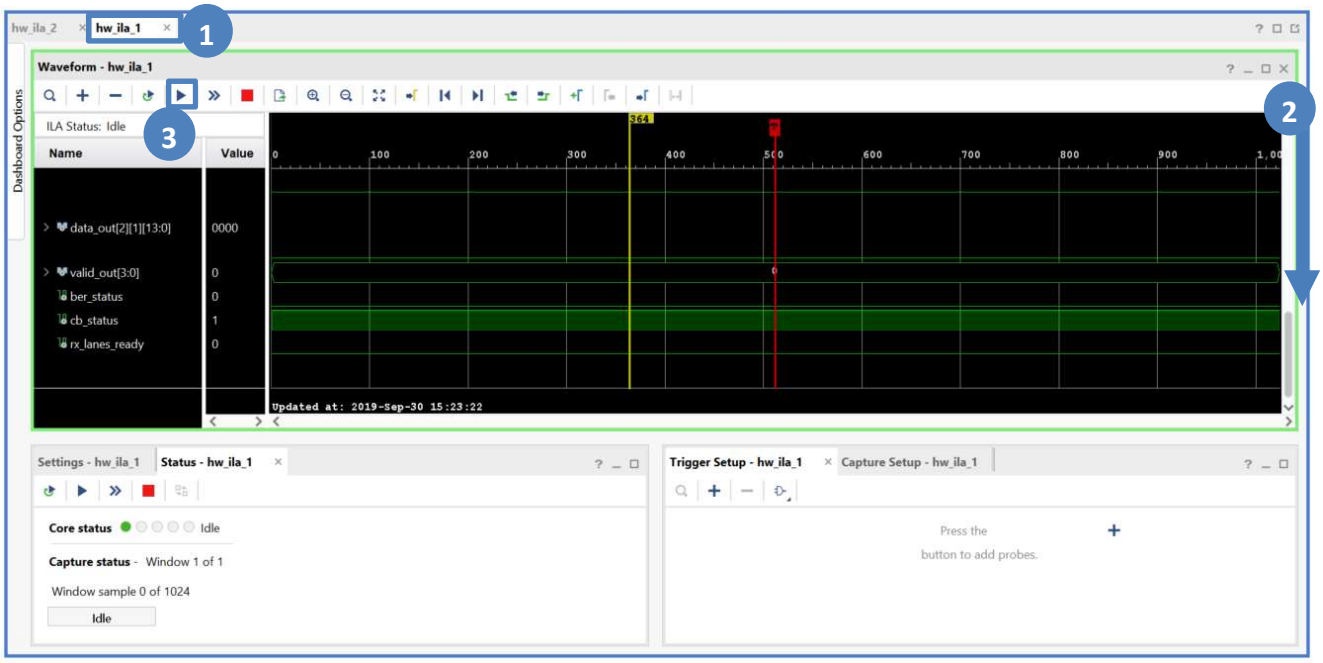
- Click on *Open target > Auto connect*



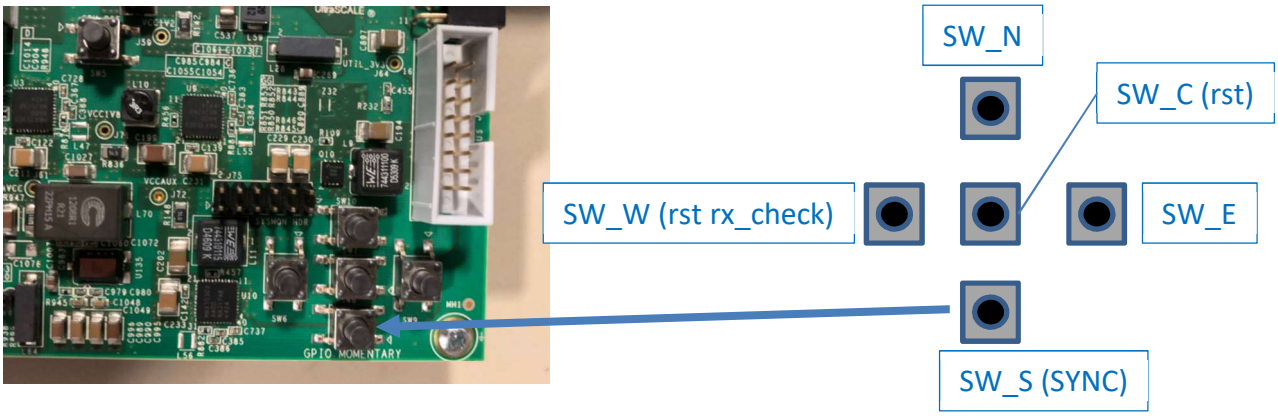
- Click on *Program device > Program*:



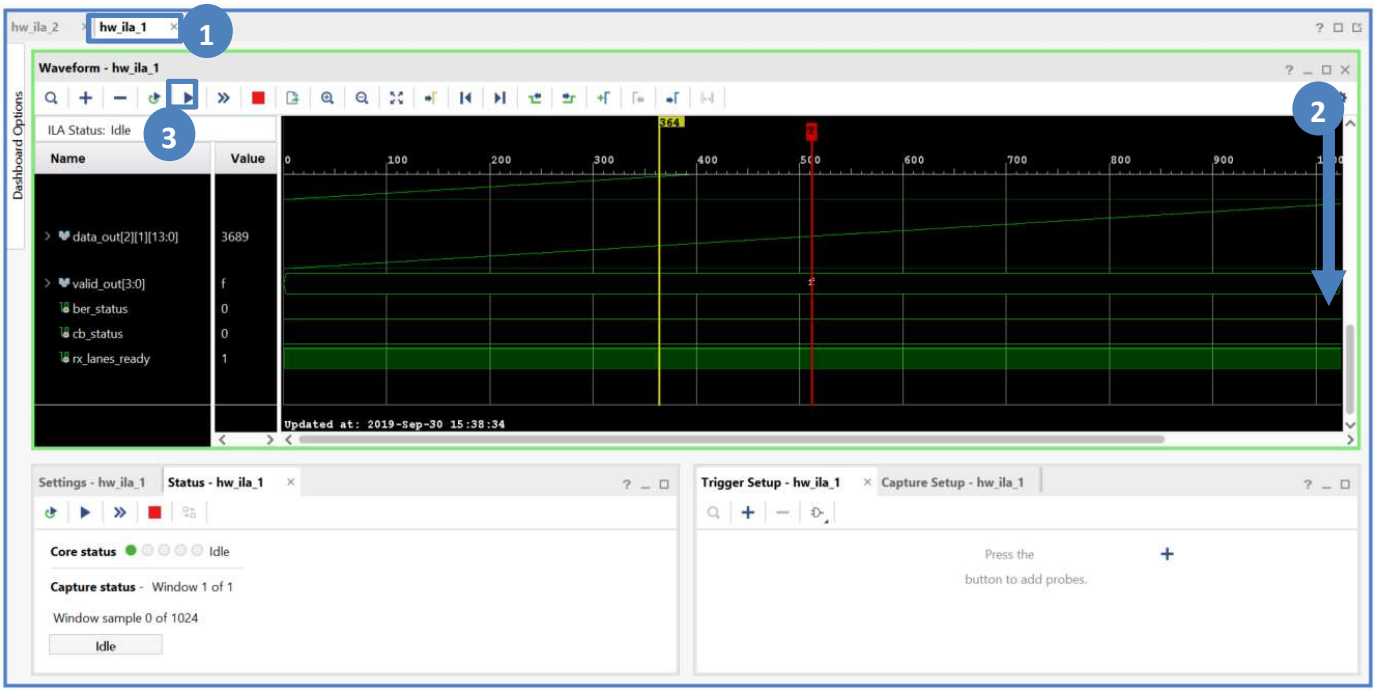
- *hw_ila_1*:
  - after clicking on Run trigger for this ILA core (Play icon):
    - rx lanes are not ready (rx_lanes_ready signal logic low)
    - Data not valid



- Now on the KCU105 board:
  - Push rst push-button, SW_C
  - Push SYNC push-button, SW_S
  - Push SYNC push-button, SW_W
    - LEDs 0, 1 and 2 should on (tx ip ready, rx ip ready, lanes ready).
    - Others LEDs should OFF (No bit error and No Clock bit error).



- *hw_ila_1*:
  - after clicking on Run trigger for this ILA core (Play icon)
    - rx lanes are ready (rx_lanes_ready signal logic high)
    - There is no bit error on all lanes (ber_status signal logic low)
    - There is no clock bit error on all lanes (ber_status signal logic low)
    - Positive ramp pattern received on all data (14-bit).
    - Data valid



2.3.3 Testbench

The testbench instantiates the ESIstream encoding ip (tx_esistream), the ESIstream decoding ip (rx_esistream) and the transceivers (rx_tx_xcvr_wrapper).

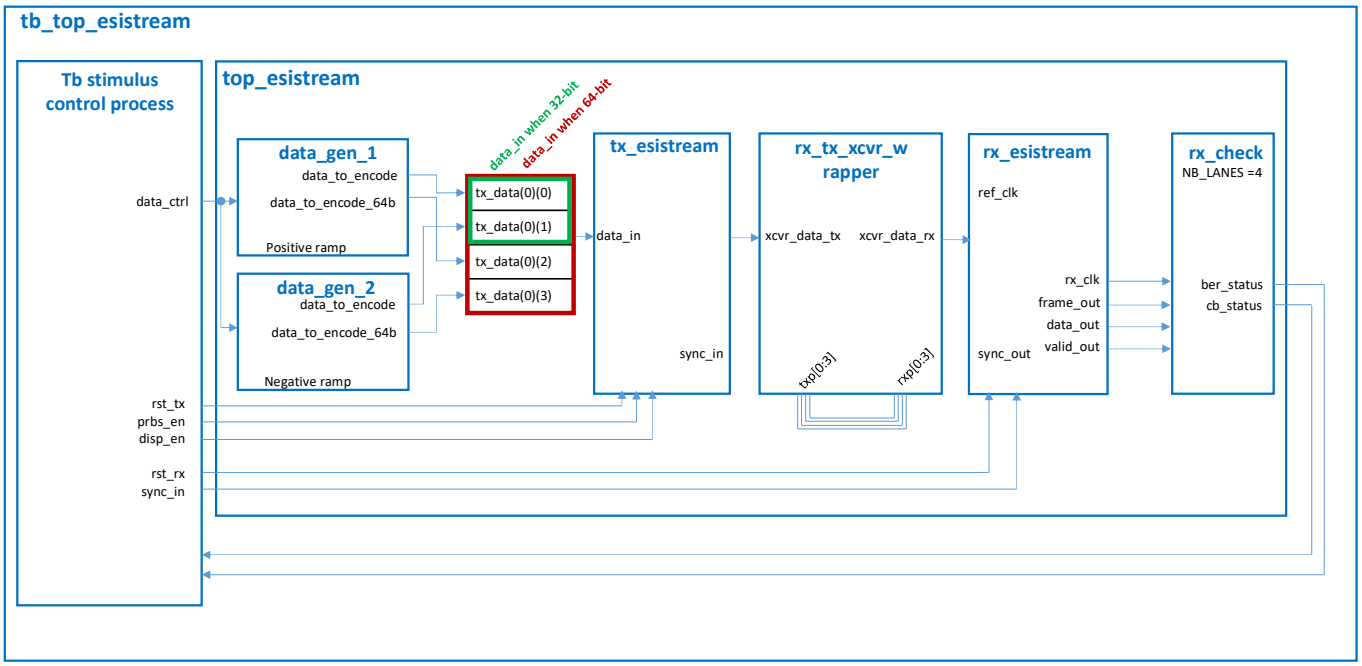


Figure 9: 64-bit SERDES testbench Block diagram

Two data generator modules (data_gen) generate known data pattern at the transmitter input.
According to the value of the data generator control input signal *d_ctrl,* the data to encode can be a 14-bit positive ramp, a 14-bit negative ramp, zeros only or ones only.

d_ctrl value	waveform
"00"	"00000000000000"
"01"	Positive ramp For 64-bit implementation: - Odd values came out on *data_out* - Even values came out on *data_out_64b*
"10"	Negative ramp For 64-bit implementation: - Odd values came out on *data_out* - Even values came out on *data_out_64b*
"11"	"11111111111111"

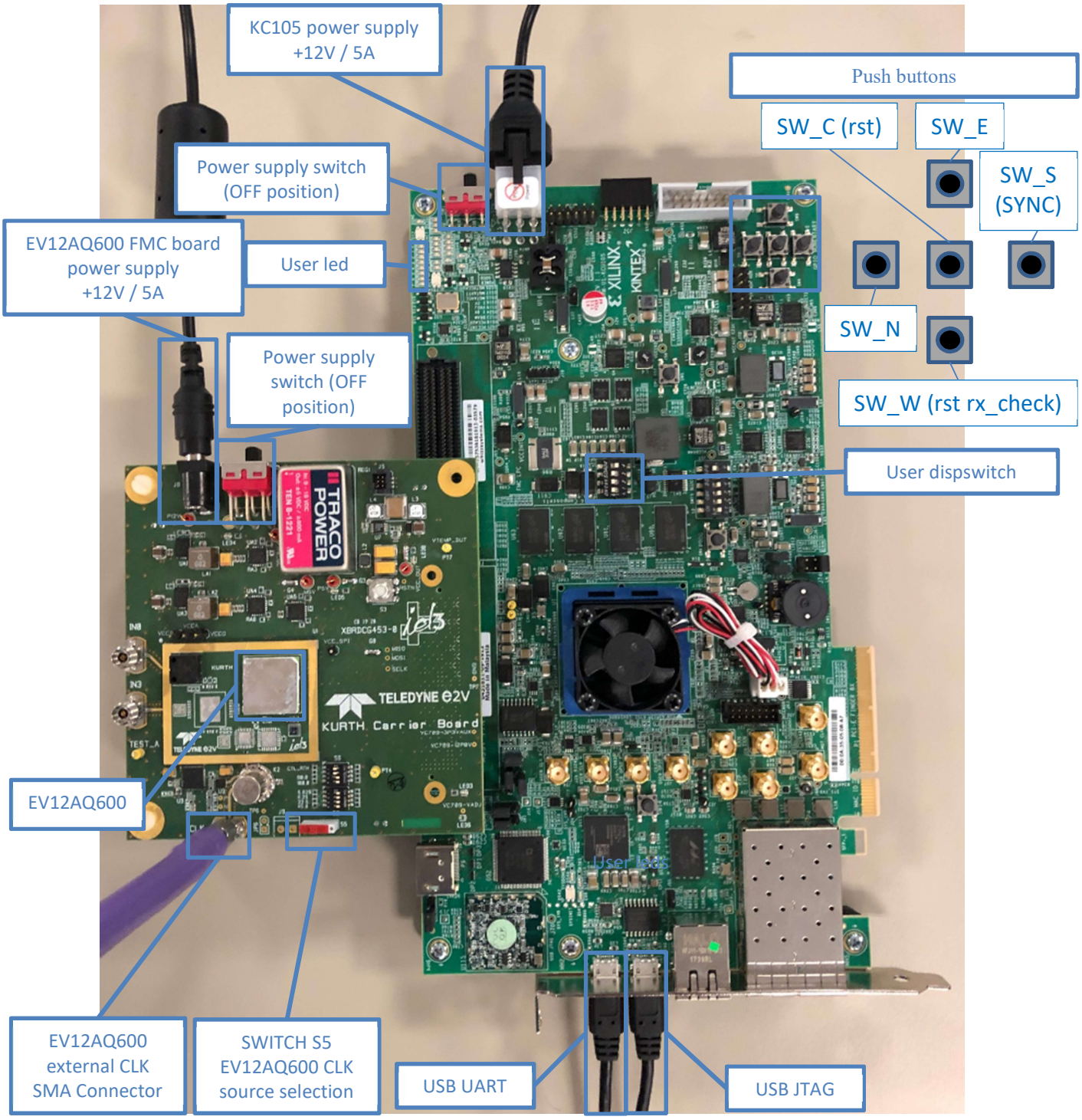
The validation module (rx_check) allows checking received data values. It reports an error when there is a bad value in one of the received data field or in one of the clock bit field of the received ESIstream frames.

BER status	'0': No error '1': Data bit error
CB status	'0': No error '1': Clock bit error

2.4 FMC EV12AQ600 project setup

2.4.1 Hardware setup

- **Warning:** To avoid power supplies hot plug, check all power supplies switches positions are OFF.
- **Warning:** Use the correct hardware configuration for each bitstream.



For more information on the EV12AQ600 FMC test board, contact GRE-HOTLINE-BDC@Teledyne.com.

2.4.1.1 Dipswitch configuration

Dipswitch	Function	Description
1	prbs enable	RX ESIstream IP scrambling processing enabled when ON.
2	data enable	EV12AQ600 data enabled when ON else all data bits set to logic 0 (a SPI write of DATA_MODE_SEL register must be done).
3	External PLL enable	When ON external PLL located on the EV12AQ600 FMC test board is used to clock the ADC. Else an external clock should be provided on CLK SMA + S5 switch right position.
4	DC balance enable	EV12AQ600 disparity processing enabled when ON (a SPI write of DATA_MODE_SEL register must be done).

2.4.1.2 Leds status

GPIO LEDs	Output	Description
0	User reset status	'0': reset pushbutton SW_C on '1': reset pushbutton SW_C off
1	User SYNC status	'0': SYNC pushbutton SW_S on '1': SYNC pushbutton SW_S off
2	Transceivers ip ready status	'0': PLL unlocked or reset not done. '1': PLL locked and reset done
3	Lanes ready status	'0': Lanes synchronization failed '1': Lanes synchronized succeed
4	Isrunning status	'0': aq600_interface.vhd isrunning fsm state not reached. '1': RX check tests can be launched pushing SW_W and data can be monitored using vivado ILA.
5	None	Always '0'
6	BER status	'0': No error '1': Bit error After lanes synchronized, rst check push button (SW_W) should be pressed for 1-second minimum to start the test and get a valid status.
7	CB status	'0': No error '1': Clock bit error After lanes synchronized, rst check push button (SW_W) should be pressed for 1-second minimum to start the test and get a valid status.

2.4.1.3 Push-buttons description

Push buttons	Control	Description
SW_N	Change ADC data mode	Not used for this test. FPGA AQ600 interface FSM should be in isrunning state to use it.
SW_S	SYNC signal generation	
SW_W	RX check module reset	Push button when led 0-1-2-3-4 are ON
SW_E	Start / Stop	Push button to start or stop the test.
SW_C	FPGA logic reset	

2.4.1.4 S5 switch configuration

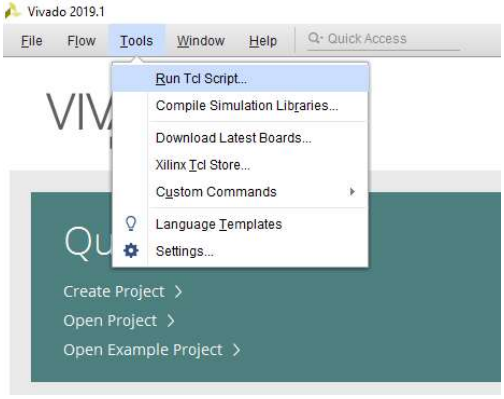
S5 switch position	Function	Description
Left	SMA external CLK selected	
Right	PLL external CLK selected	

2.4.2 Software setup

A tcl script ("script.tcl") is provided with the sources and can be used to generate the project using Vivado Xilinx software. It is recommended to use Vivado 2019.1 as updating the IP (in particular the GTH) to a more recent version may cause issue.

- Open Vivado 2019.1

- Tools > Run Tcl Scripts…
    - Launch the package Tcl script located in the directory ***vivado_rx_ev12aq600/script.tcl***



When the project has been created and the IP generated, you can start compiling, simulating or modifying the example design like any Vivado project.
Note: If the TCL script is moved from its original folder it will not work as it uses relative path.

- Generate bitstream.
- Program the FPGA
    - Click on *Open target > Auto connect.*
    - Click on *Program device > Program*.

2.4.3 RAMP mode test

The ADC EV12AQ600 can be configured in ramp mode. In ramp mode, the data encoded corresponds to a 12-bit ramp value with only the even values on lane 1 and the odd values on lane 2.

See below the chronogram of the ramp test mode. The data shown in the following figure only presents the 14 bits data from the ADC (12 bits sample value plus 2 control bits CB1 and CB2) and does not include the encoding of the ESIstream protocol which is used on the serial interface.

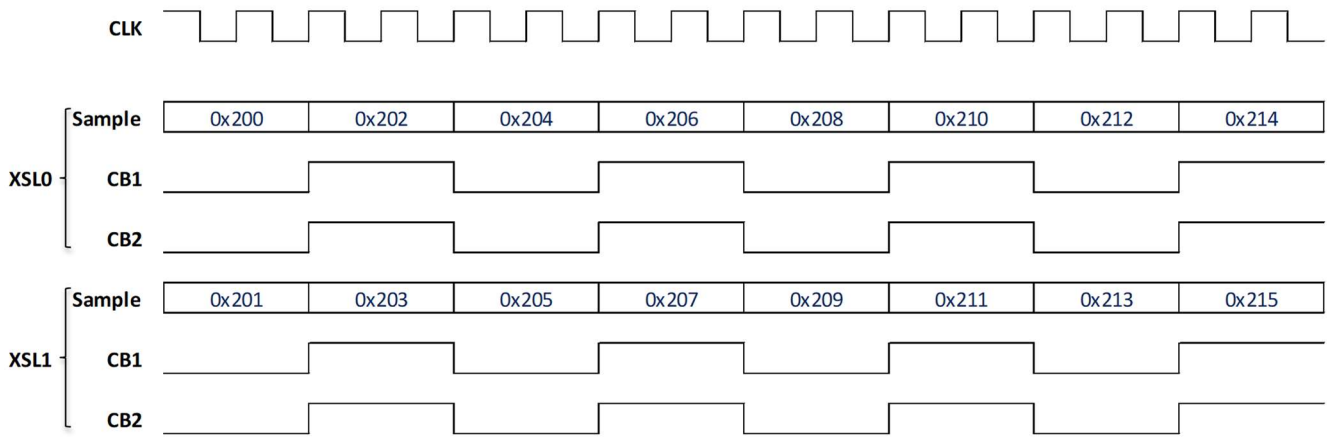


Figure 10: Chronogram of the Ramp test mode

The rx_check module allows checking that the received data match this pattern and that the clock bit is valid.

The rx_check module allows checking that the received data match this pattern and that the clock bit is valid.

A state machine located in the aq600_interface.vhd file allows configuring the EV12AQ600 ADC in ramp mode and launching the ESIstream serial link synchronization (SYNC signal).

This state machine starts when the user pushes the “Start / Stop” push-button.

- Functional State Machine (FSM) description:

  1- Write external PLL registers.
  2- Wait external PLL lock signal
  3- Reset EV12AQ600 and FPGA transceivers (10 µs min). During the RSTN pulse, CSN must be held high and SCLK held low. The CLK must be provided before the RSTN pulse. The CLK can start before or after the power-up.
  4- Wait OTP memory wakes up (1 ms min).
  5- Send SPI instruction WRITE @0x0001 =0b1 to EV12AQ600 ADC. The EV12AQ600 OTP memory cells are loaded into registers. There must be at least 1 ms between the RSTN pulse and this instruction.
  6- Send SPI instruction WRITE @0x80BA =0b1 to enable ramp mode.
  7- Send SYNCTRIG pulse to the ESIstream IP and to the ADC (default: the trigger mode is disabled).
  8- Normal operation when the FSM reaches the *isrunning* state). Data monitoring at the output of the RX ESIstream IP is possible using the Vivado ILA interface.

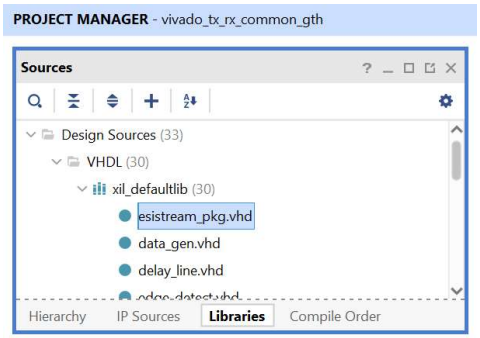
- Ramp mode test procedure:

1- Connect the EV12AQ600 FMC board to FPGA evaluation board FMC connector.
2- Check that the EV12AQ600 FMC board power switch is on OFF position.
3- Connect EV12AQ600 FMC board +12V / 5A power supply cable.
4- Check that the FPGA evaluation board power switch is on OFF position.
5- Connect FPGA evaluation board +12V / 5A power supply cable.
6- Connect JTAG cable between the FPGA evaluation board and the PC.
7- Slide EV12AQ600 FMC board switch (S5) to the position allowing the use of the SMA external CLK.
    a. Use a signal generator to generate the EV12AQ600 CLK signal (6.4 GHz / +12 dBm).
8- FPGA evaluation board dipswitch positions **SW12.3 should be OFF**, SW12.1, SW12.2 and SW12.4 positions should be all ON.
9- Power on EV12AQ600 FMC board and the FPGA evaluation board.
10- Load FPGA using Quartus programmer. The bitstream must be generated before opening the hardware manager.
11- Push-button (2 second) “FPGA logic reset” (SW_C).
12- Push-button (2 second) “Start / Stop” (SW_E).
13- Check all leds are ON excepted led 5 (OFF).
14- Push button (2 second) “RX check module reset” (SW_W)
15- Vivado Integrated Logic Analyzer (ILA) acquisition can be performed after refreshing device through hardware manager.

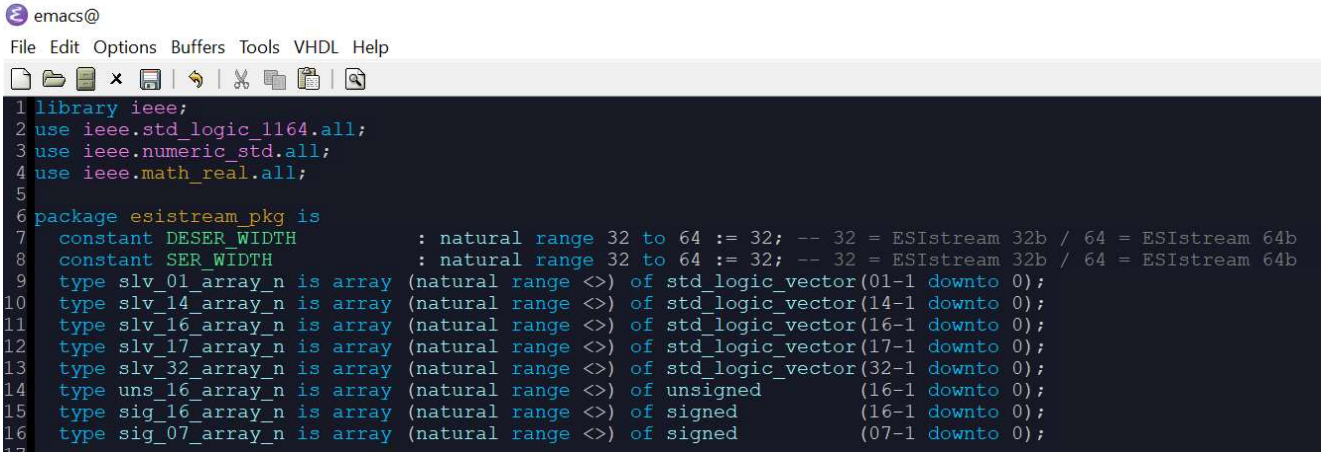
3. ANNEX A: SWITCH PROJECT TO A 32-BIT SERDES

Moving to 32b SERDES allows reducing the ESIstream IP latency.

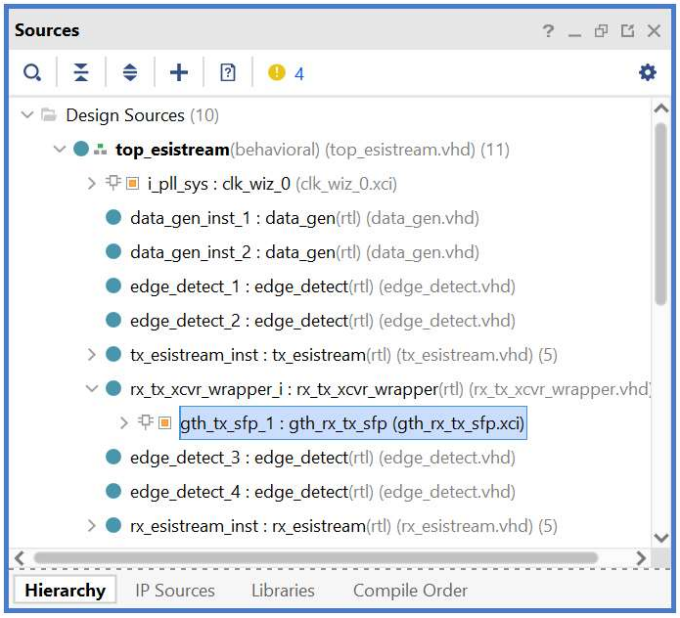
- Generate the project using the Tcl script or open the project.
- Open the file *esistream_pkg.vhd* located in the *src_common* directory.



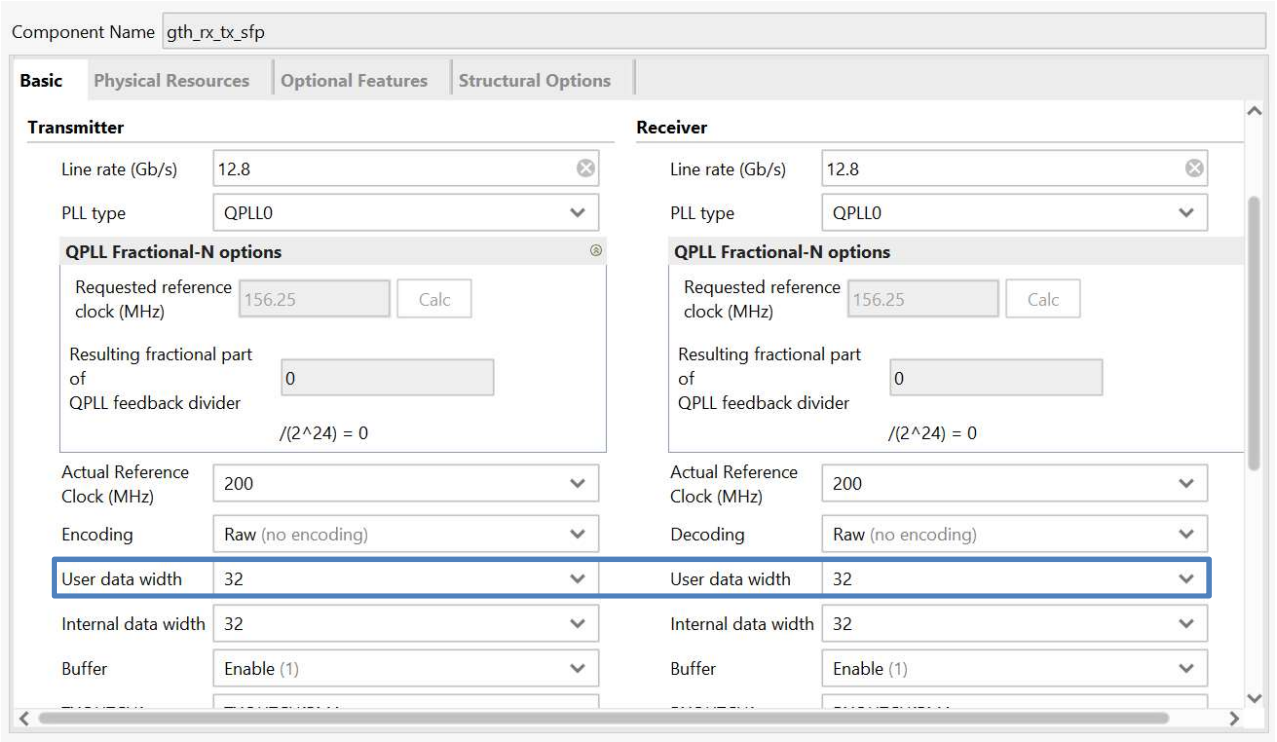
- Change *DESER_WIDTH* and *SER_WIDTH* constants value from 64 to 32 in *esistream_pkg.vhd.*



- In Sources > Hierarchy open the IP *gth_rx_tx_sfp.xci*.



- In *Basic,* change User data width from 64 to 32 for both the Transmitter and the Receiver.



- Click on OK and wait end of IP Re-customization.
- Click on Generate and wait the end of IP generation.