KCU105 ADC12DJxx00 JESD REFERENCE DESIGN USER GUIDE

Table of Contents

I. DESCRIPTION

The design “KCU105_ADC12DJxx_6G.zip” is developed for KCU105 board for the modes: JMODE0 and JMODE2. It has JESD Base IP and JESD PHY IP to get JESD data from the ADC12DJxx00 and is compiled for 6G lane rate. The design has a simple transport layer specific for the two modes that captures samples from the ADC, re-order the bits and give out 20 samples every clock cycle. The results can be verified using Chipscope. This document gives a brief on the compilation and verification process involved.

NOTE: This version of the FW is a fixed line rate firmware and hence will work only at 6G. For any other line rate the firmware needs to be recompiled for that specific linerate. This design has been validated upto 6.4G.

Section II discusses about how the hardware setup should be connected. Section III discusses on how to extract the project from .zip file and the compilation process involved. Section IV discusses on how to get the ADC data in Chipscope and validate the same. Section V discusses the LED debug signals added in project.

Signals	Description	Direction
CLK_IN1_D	Constant 300Mhz clock coming from an on-board crystal	Input
refclk	Reference clock from the ADC12DJxx00 EVM	
sysref	SYSREF Signal from the 12DJxx00 EVM	
Rxp_in/Rxn_in	Serial Data from ADC (LVDS lines)	
rx_dataout	Transport Layer data out (20 samples for every link clock)	Output from the transport module
rx_validout	SOMF aligned with rx_dataout	
rx_somfout	Data valid Signal aligned with rx_dataout	

ADC12DJ3200 RevA EVM is used for testing. In this EVM revision P & N pins of upper four SERDES lanes are swapped, hence Rx lane polarity inversion is implemented in the design to address that. Due to this, Rx lane polarity inversion constant given to the PHY module is 240.

II. HARDWARE SETUP

Connect the KCU105 board with 12DJ3200 RevA. Please use the **HPC** FMC to connect the devices. Connect the Digilent port or the JTAG cable to the PC in order to download the firmware. Additionally UART port must also be connected to the PC to make any configurations.

III. COMPILING VIVADO PROJECT

1. Extracting the zip file.

- Right click the KCU105_ADC12DJxx_6G.xpr.zip file and then press "Extract All". After extracting, open the project using **Vivado 2016.1**. The project file is KCU105_ADC12DJxx_6G.xpr\prj_MyKcu105_TI\ prj_MyKcu105_TI.xpr

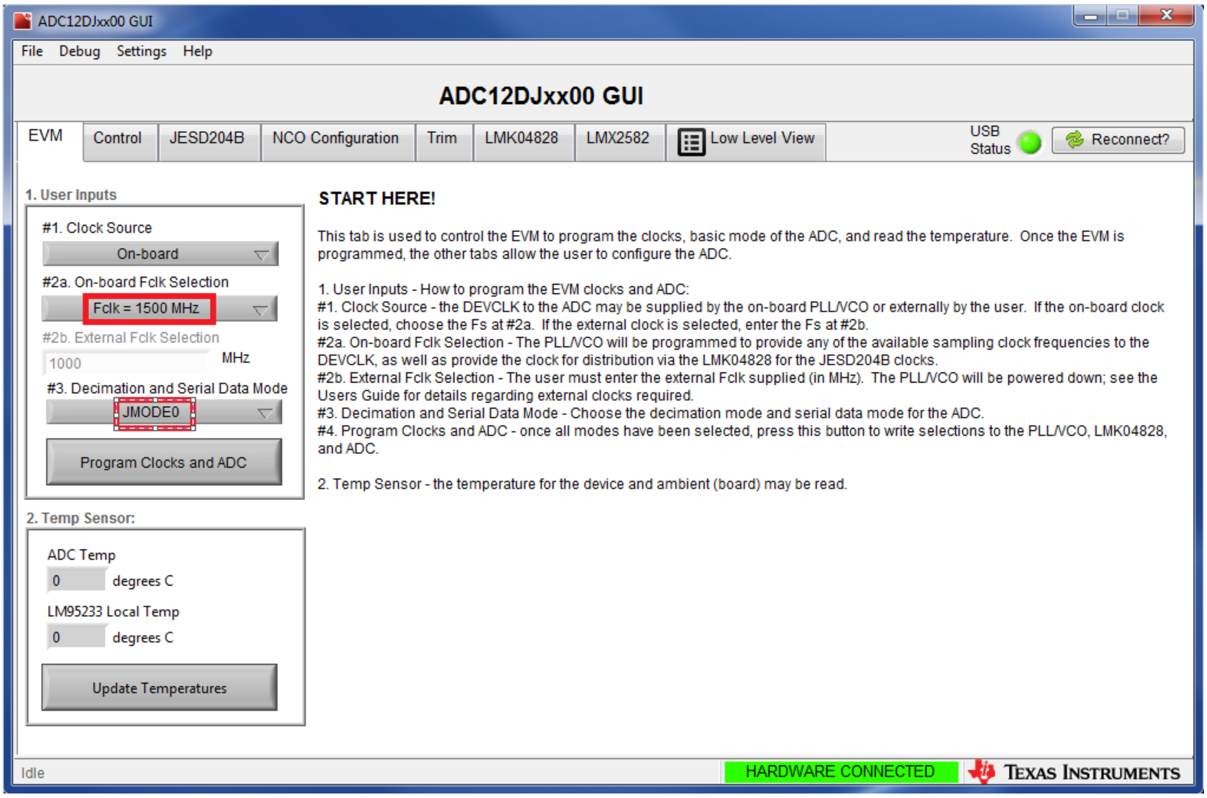
2. Compiling the project.

- In Vivado, on the left you will see the Flow Navigator. Press the generate Bitstream option under "Program and Debug" to generate the output ".bit" file.
- Once the generation is over a .bit file will be generated, along with with .ltx file (required for chipscope).
- The bit file generated is "mySystem_wrapper.bit". The name of the .ltx file is debug_nets.ltx. These are the files that must be loaded when the device is to be programmed.
- These files are generated at the location - KCU105_ADC12DJxx_6G.xpr\prj_MyKcu105_TI\prj_MyKcu105_TI.runs\impl_1\

IV. VALIDATING WITH ADC12DJxx00

1. Configure ADC

- The reference clock from ADC should be stable before downloading firmware. So, the ADC has to be configured first before we program the board.
- Open ADC12DJxx00 GUI, choose Fs as 1500Msps (equivalent to 6G lane rate) and choose either JMODE0 or JMODE2 in the EVM tab. Click "Program Clocks and ADC"
- Once it is successful, go to JESD204B tab, deselect "JESD Block Enable" and select the desired test Mode. Enable JESD Block again (This step can be skipped if the normal sine wave input signal is to be captured).



ADC12DJxx00 GUI

1. User Inputs

#1. Clock Source: On-board

#2a. On-board Fclk Selection: Fclk = 1500 MHz

#2b. External Fclk Selection: 1000 MHz

#3. Decimation and Serial Data Mode: JMODE0

2. Temp Sensor:

ADC Temp: 0 degrees C

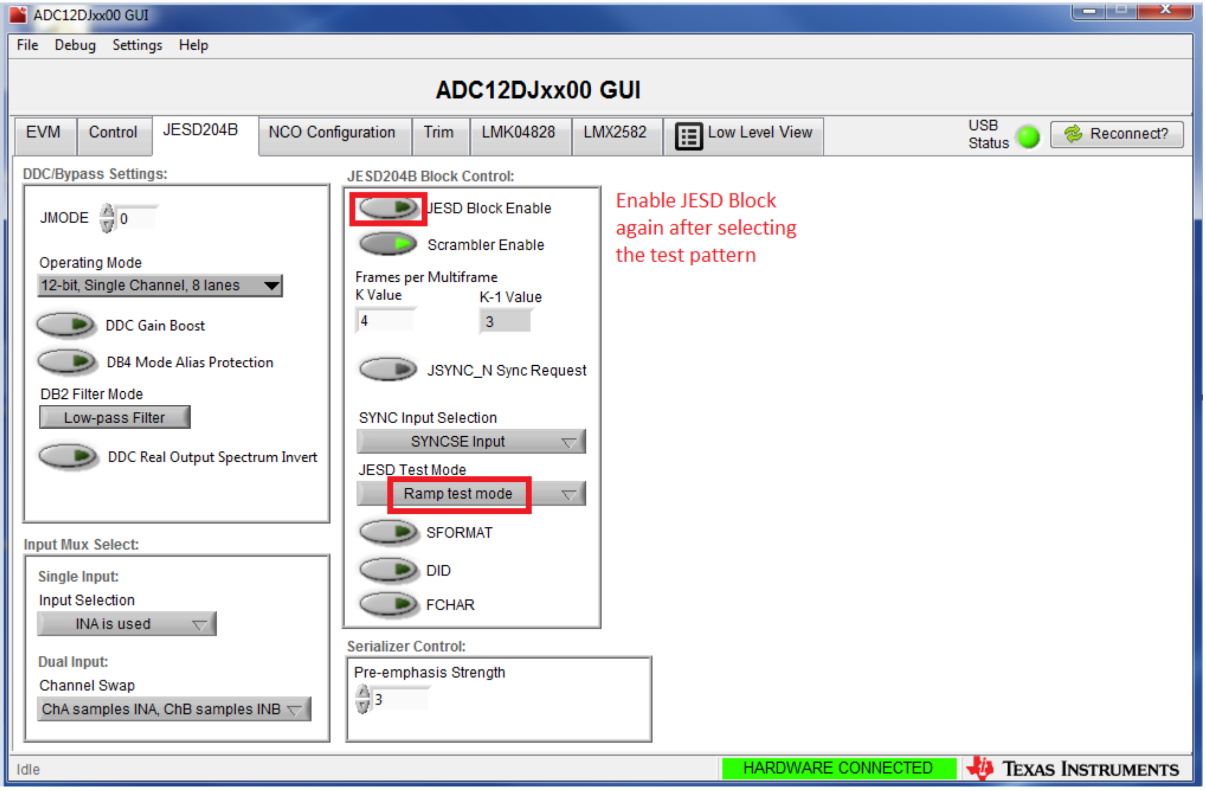
LM95233 Local Temp: 0 degrees C

START HERE!

This tab is used to control the EVM to program the clocks, basic mode of the ADC, and read the temperature. Once the EVM is programmed, the other tabs allow the user to configure the ADC.

1. User Inputs - How to program the EVM clocks and ADC:
#1. Clock Source - the DEVCLK to the ADC may be supplied by the on-board PLL/VCO or externally by the user. If the on-board clock is selected, choose the Fs at #2a. If the external clock is selected, enter the Fs at #2b.
#2a. On-board Fclk Selection - The PLL/VCO will be programmed to provide any of the available sampling clock frequencies to the DEVCLK, as well as provide the clock for distribution via the LMK04828 for the JESD204B clocks.
#2b. External Fclk Selection - The user must enter the external Fclk supplied (in MHz). The PLL/VCO will be powered down; see the Users Guide for details regarding external clocks required.
#3. Decimation and Serial Data Mode - Choose the decimation mode and serial data mode for the ADC.
#4. Program Clocks and ADC - once all modes have been selected, press this button to write selections to the PLL/VCO, LMK04828, and ADC.

2. Temp Sensor - the temperature for the device and ambient (board) may be read.



ADC12DJxx00 GUI

DDC/Bypass Settings:

JMODE: 0

Operating Mode: 12-bit, Single Channel, 8 lanes

DB2 Filter Mode: Low-pass Filter

Input Mux Select:

Single Input:
Input Selection: INA is used

Dual Input:
Channel Swap: ChA samples INA, ChB samples INB

JESD204B Block Control:

Frames per Multiframe
K Value: 4
K-1 Value: 3

SYNC Input Selection: SYNCSE Input

JESD Test Mode: Ramp test mode

Serializer Control:

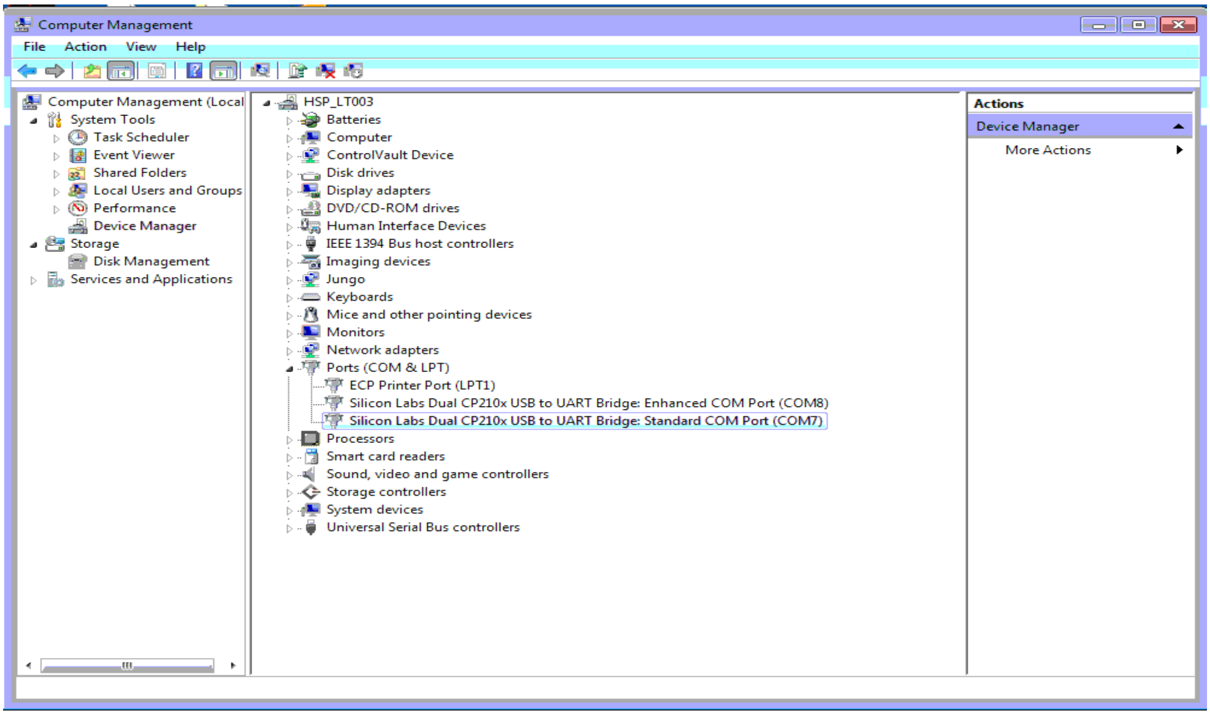
Pre-emphasis Strength: 3

Enable JESD Block again after selecting the test pattern

2. Programming with Vivado Hardware Manager

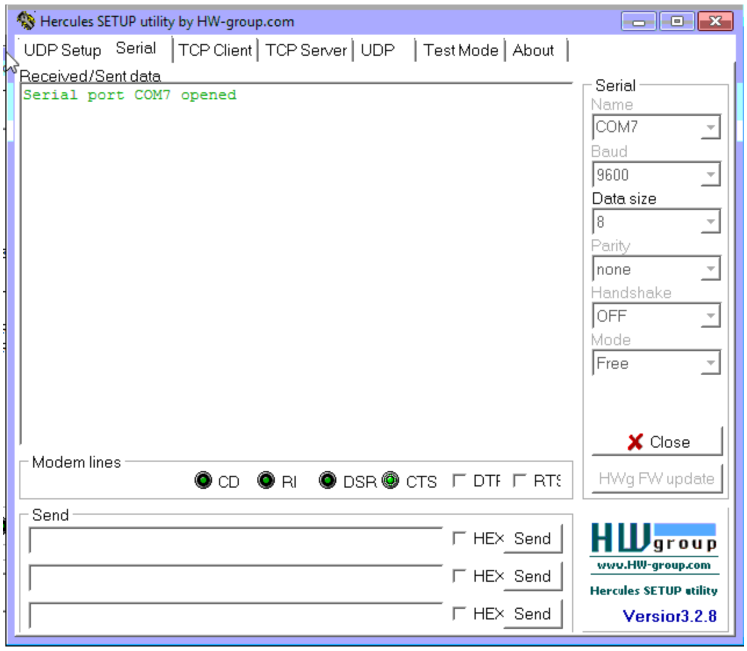
In order to program the KCU105 we need to follow a series of steps. Please make sure you follow the order while programming it.

1) Make sure to configure the ADC EVM before programming the device.
2) Open Device manager and check for the COM ports. (Make sure you have connected to the UART port of the board)



In this case, we have COM Port 8 (Enhanced) and COM Port 7 (Standard).

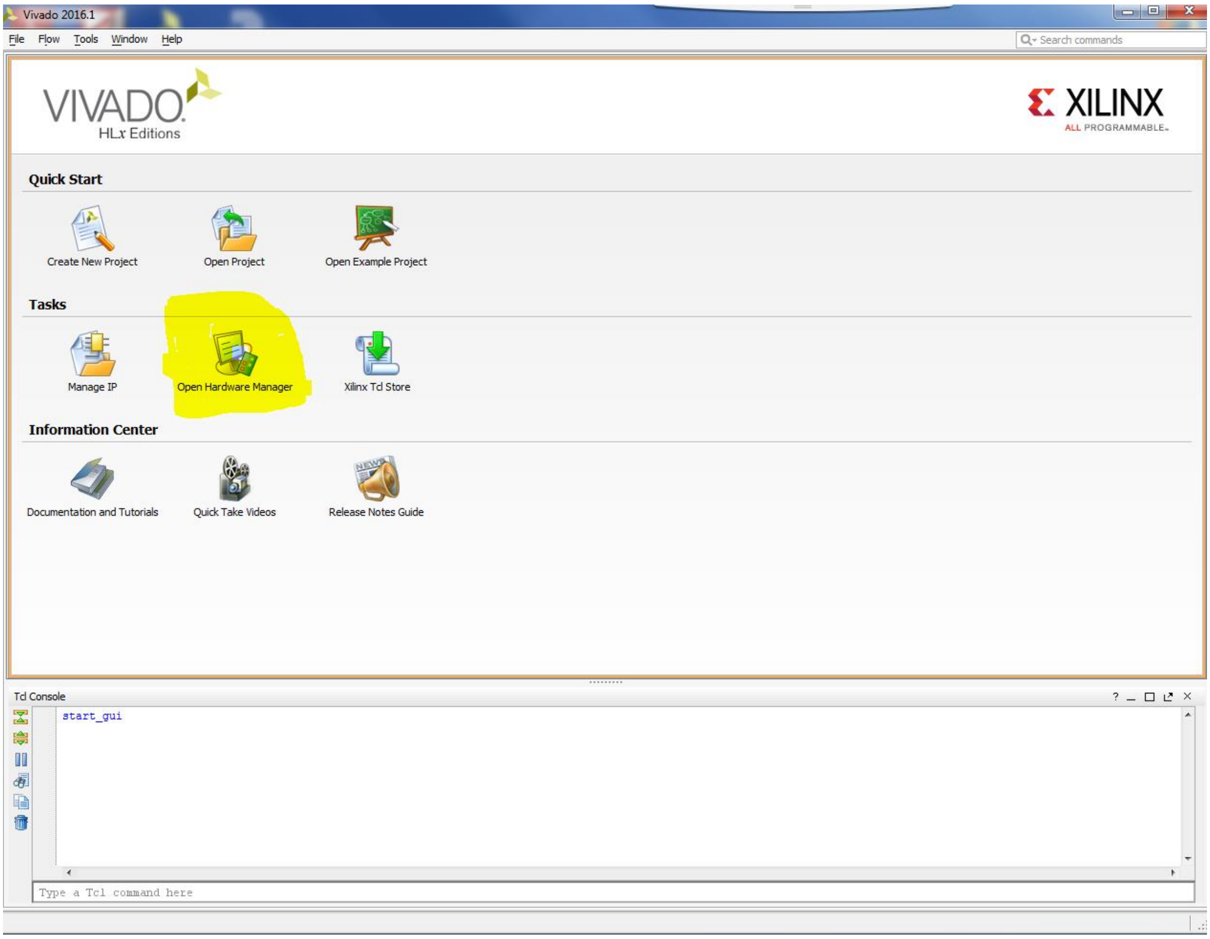
3) Open both the COM ports using any Utility (like Hercules). For **Standard COM** Port(in this case COM port 7), make sure of the below settings while opening the port.



```
Serial port COM7 opened
```

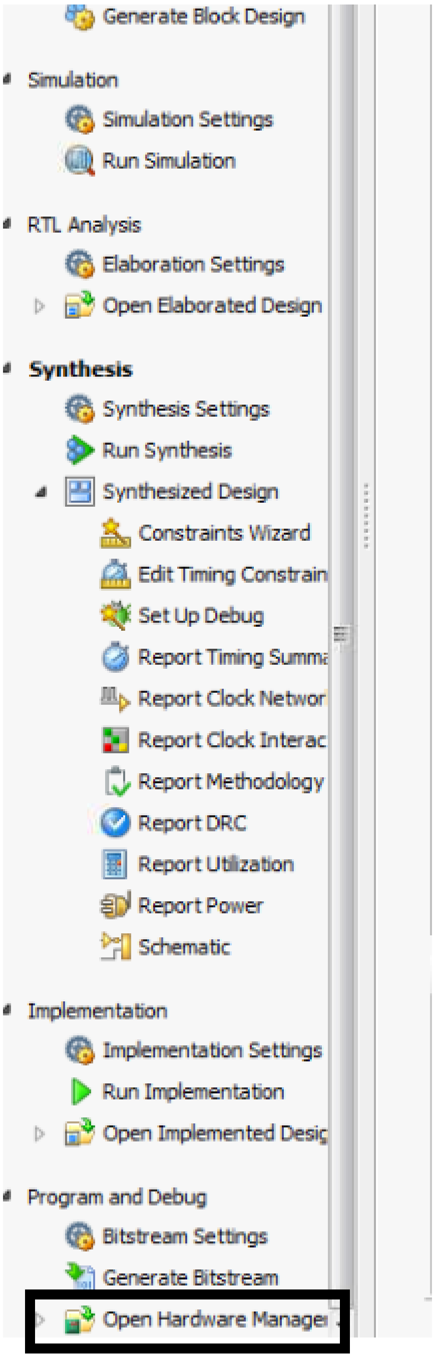
Serial
Name: COM7
Baud: 9600
Data size: 8
Parity: none
Handshake: OFF
Mode: Free

4) Open the Vivado Hardware manager. It can be opened from the GUI start screen (or) the Vivado flow navigator under “Program and Debug”.

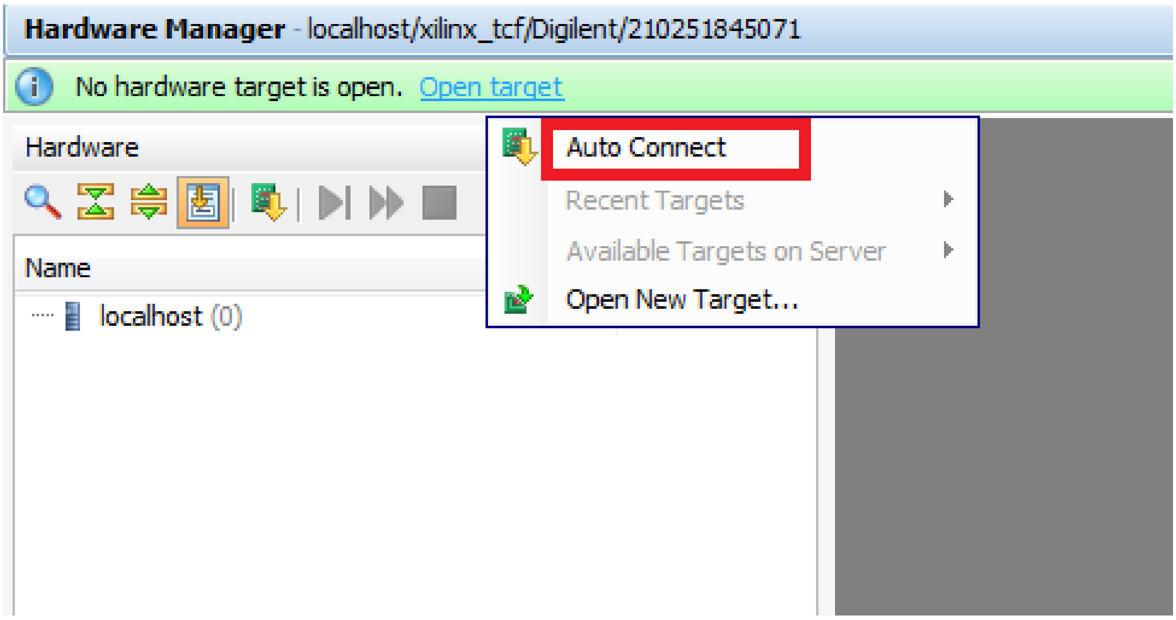


```
start_gui
```

(OR)


Generate Block Design
Simulation
Simulation Settings
Run Simulation
RTL Analysis
Elaboration Settings
Open Elaborated Design
Synthesis
Synthesis Settings
Run Synthesis
Synthesized Design
Constraints Wizard
Edit Timing Constrain
Set Up Debug
Report Timing Summa
Report Clock Networ
Report Clock Interac
Report Methodology
Report DRC
Report Utilization
Report Power
Schematic
Implementation
Implementation Settings
Run Implementation
Open Implemented Desig
Program and Debug
Bitstream Settings
Generate Bitstream
Open Hardware Manage

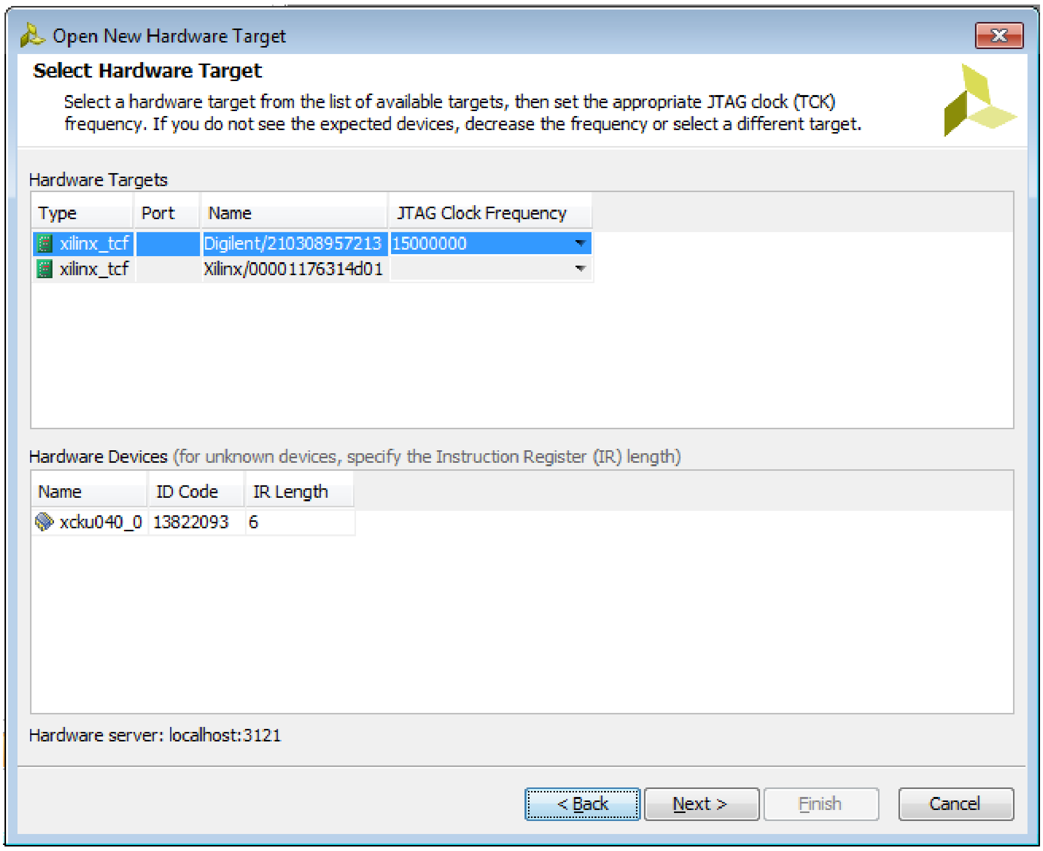
5) Press on Open Target and then press on Auto connect.



Hardware Manager - localhost/xilinx_tcf/Digilent/210251845071

No hardware target is open. Open target

If you cannot auto connect, then press on "Open New Target" from the drop down list. Click Next twice and you should see the following.



Open New Hardware Target

Select Hardware Target

Select a hardware target from the list of available targets, then set the appropriate JTAG clock (TCK) frequency. If you do not see the expected devices, decrease the frequency or select a different target.

Hardware Targets

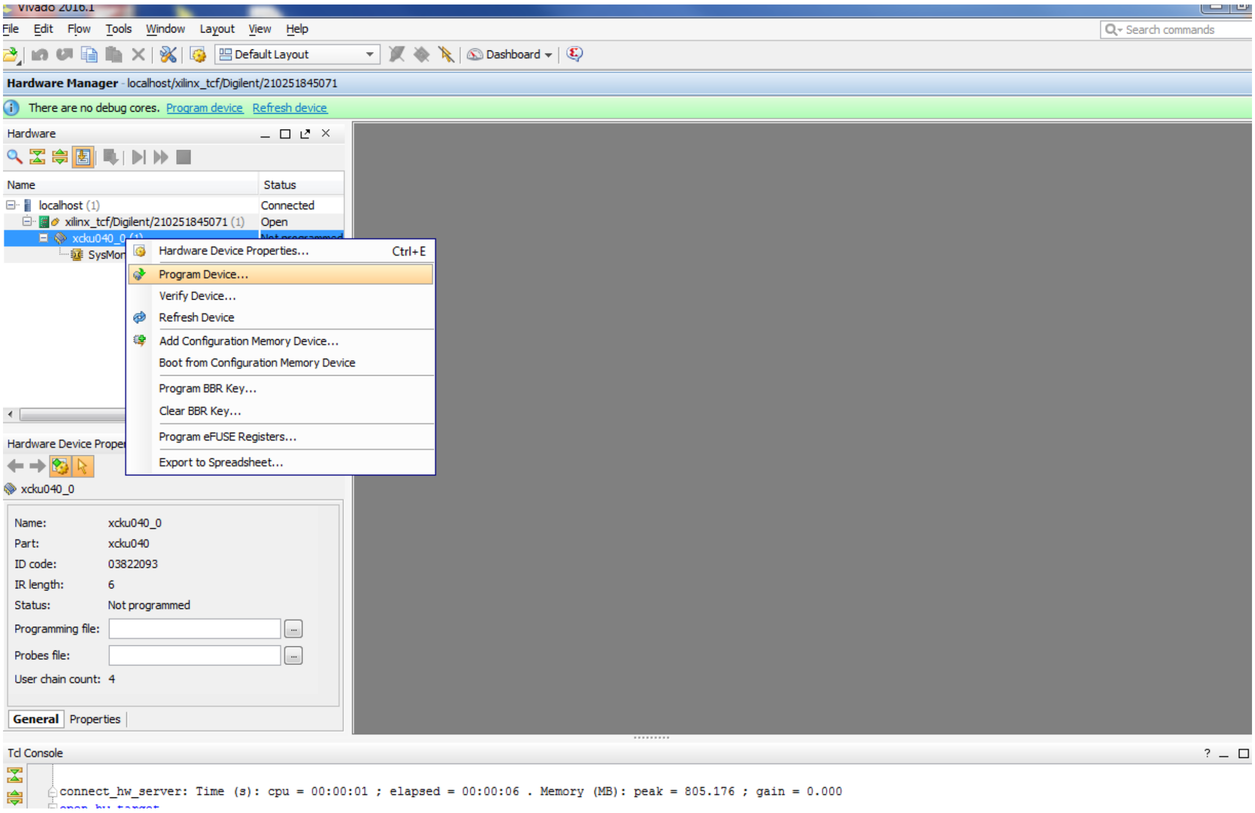
Type	Port	Name	JTAG Clock Frequency
xilinx_tcf		Digilent/210308957213	15000000
xilinx_tcf		Xilinx/00001176314d01	

Hardware Devices (for unknown devices, specify the Instruction Register (IR) length)

Name	ID Code	IR Length
xcku040_0	13822093	6

Hardware server: localhost:3121

6) After that Right click on the KCU105 device part number that is listed and press on Program Device.



Hardware Manager - localhost/xilinx_tcf/Digilent/210251845071

There are no debug cores. Program device Refresh device

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210251845071 (1)	Open

xcku040_0

Name: xcku040_0
Part: xcku040
ID code: 03822093
IR length: 6
Status: Not programmed
Programming file:
Probes file:
User chain count: 4

```
connect_hw_server: Time (s): cpu = 00:00:01 ; elapsed = 00:00:06 . Memory (MB): peak = 805.176 ; gain = 0.000
```

In the Pop-up. Select the “**mySystem_wrapper.bit**” generated earlier in the Bitstream file path. In the Debug Probes file path mention the “**debug_nets.ltx**” file which was also previously generated. Then press “**Program**”.

3. **Setting the FMC voltage**

We need to set the FMC voltage of the KCU to 1.8V before we capture. In order to set this please make sure that you have connected the UART of the board to the PC. This setting needs to be done only once after the KCU has been connected and turned ON. For subsequent captures, this section can be skipped.

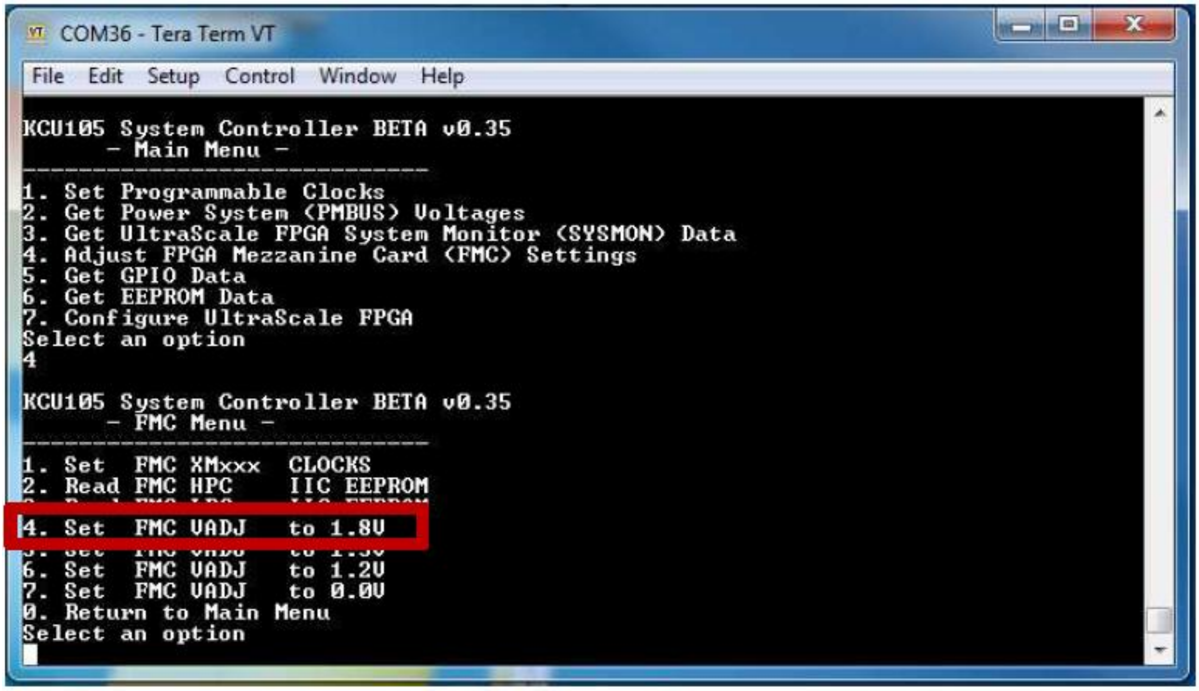
Enhanced COM Port

- Once firmware is downloaded, we need to set the FMC voltage to 1.8v in the enhanced port.
- To do that, in the Hercules (or any Terminal app to open COM port) where we have opened the Enhanced COM port, enter 0 to go the Main menu
- The main menu items will be displayed. Enter 4 to Adjust FPGA Mezzanini Card (FMC) settings
- Again enter 4 to Set FMC VADJ to 1.8V.

- To read the voltage value for verification, go back to the main menu by entering 0. Then enter 2 to “Get the Power system Voltages” and enter 7 to “Get the VADJ1D8 voltage”.

  Note: More detailed description can be found here in KCU105 EVM user manual http://www.ti.com/lit/ug/slau711/slau711.pdf

Setting the FMC VADJ to 1.8V in the Enhanced COM Port terminal.



```
KCU105 System Controller BETA v0.35
      - Main Menu -
-----------------------------
1. Set Programmable Clocks
2. Get Power System (PMBUS) Voltages
3. Get UltraScale FPGA System Monitor (SYSMON) Data
4. Adjust FPGA Mezzanine Card (FMC) Settings
5. Get GPIO Data
6. Get EEPROM Data
7. Configure UltraScale FPGA
Select an option
4

KCU105 System Controller BETA v0.35
      - FMC Menu -
-----------------------------
1. Set  FMC XMxxx  CLOCKS
2. Read FMC HPC    IIC EEPROM
3. Read FMC LPC    IIC EEPROM
4. Set  FMC VADJ   to 1.8V
5. Set  FMC VADJ   to 1.5V
6. Set  FMC VADJ   to 1.2V
7. Set  FMC VADJ   to 0.0V
0. Return to Main Menu
Select an option
```

Copyright © 2017, Texas Instruments Incorporated

4. **Checking Results in Signal Tap**

Once the development kit is programmed, user can view the results in Chipscope (Vivado Hardware Manager) which probes signals from the board.

In the “**hw_ila_1**” window on the right, you can see number of signals. The different signals have been explained below.

Signals which are currently probed are

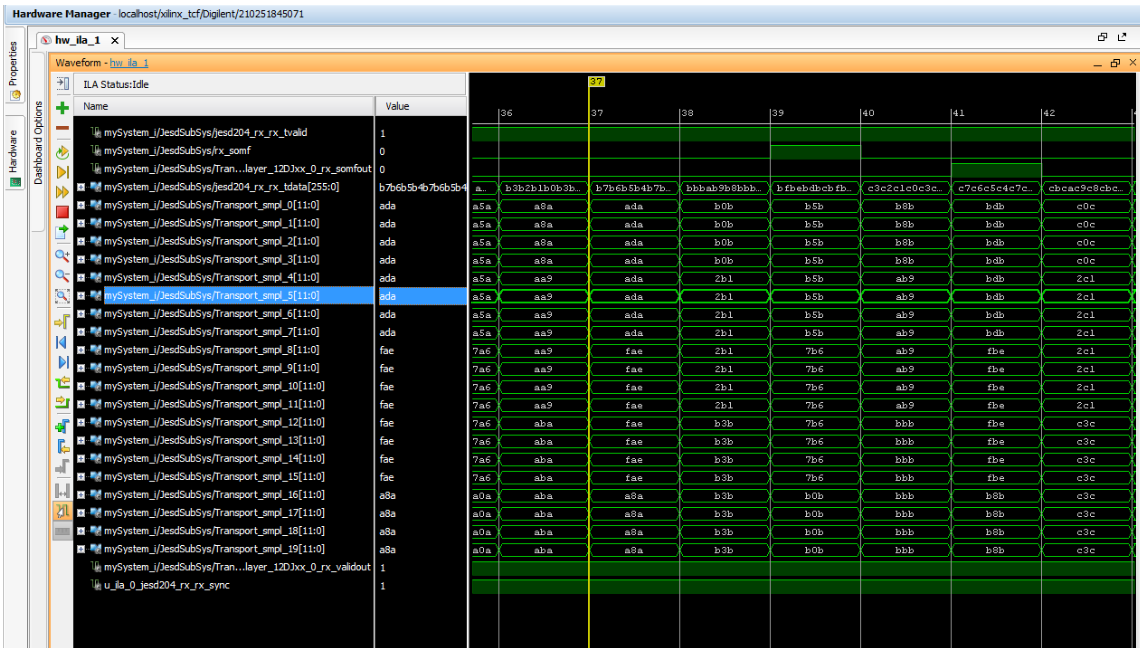
- **Rx_sync -** Active low SYNC signal from JESD Base IP. If SYNC is established, this signal will be high.
- **Jesd204_rx_rx_tdata[0…255]** – Data coming from the JESD BASE IP.
- **Jesd204_rx_rx_tvalid** – Indicates whether the output data coming from JESD Base IP is valid. If this signal is low then it is invalid.
- **Rx_somf** – Start of multiframe signal coming from the base.
- **Transport_smpl_1… Transport_smpl_19** – The output samples coming from the transport layer. The **Jesd204_rx_rx_tdata[0..255]** passes through the transport layer and is converted to a 240 signal bus (Transport_layer_12DJxx_0_rx_dataout) which contains 20 samples(12bit) per clock cycle. The “Transport_layer_12DJxx_0_rx_dataout” signal is then sliced into 20 12 bit buses that are probed (Transport_smpl_1… Transport_smpl_19). This has been done for better visibility in the chipscope for the user.

- **Transport_layer_12DJxx_0_rx_validout -** Indicates whether the output data coming from transport layer is valid. If this signal is low then it is invalid.
- **Rx_somfout** – Delayed SOMF signal coming as an output from transport layer.

Other signals can also be probed. Each time, signals are added/removed from signal tap, the project has to be compiled again.

The hw_ila_1 window is shown below.

Press the Play button on the left to capture the data. (data captured in the above image is with Ramp test mode).



Hardware Manager - localhost/xilinx_tcf/Digilent/210251845071

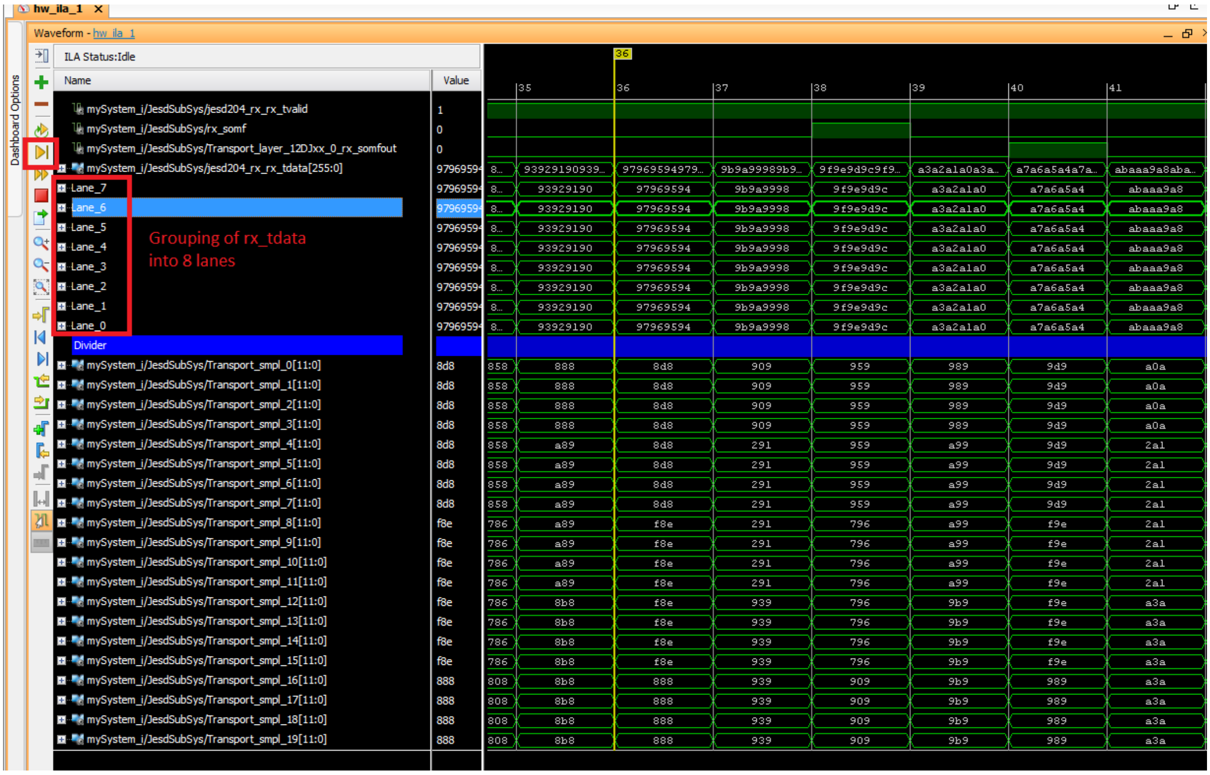
Waveform - hw_ila_1

ILA Status:Idle

Name	Value		36	37	38	39	40	41	42
mySystem_i/JesdSubSys/jesd204_rx_rx_tvalid	1								
mySystem_i/JesdSubSys/rx_somf	0								
mySystem_i/JesdSubSys/Tran…layer_12DJxx_0_rx_somfout	0								
mySystem_i/JesdSubSys/jesd204_rx_rx_tdata[255:0]	b7b6b5b4b7b6b5b4	a…	b3b2b1b0b3b…	b7b6b5b4b7b…	bbbab9b8bbb…	bfbebdbcbfb…	c3c2c1c0c3c…	c7c6c5c4c7c…	cbcac9c8cbc…
mySystem_i/JesdSubSys/Transport_smpl_0[11:0]	ada	a5a	a8a	ada	b0b	b5b	b8b	bdb	c0c
mySystem_i/JesdSubSys/Transport_smpl_1[11:0]	ada	a5a	a8a	ada	b0b	b5b	b8b	bdb	c0c
mySystem_i/JesdSubSys/Transport_smpl_2[11:0]	ada	a5a	a8a	ada	b0b	b5b	b8b	bdb	c0c
mySystem_i/JesdSubSys/Transport_smpl_3[11:0]	ada	a5a	a8a	ada	b0b	b5b	b8b	bdb	c0c
mySystem_i/JesdSubSys/Transport_smpl_4[11:0]	ada	a5a	aa9	ada	2b1	b5b	ab9	bdb	2c1
mySystem_i/JesdSubSys/Transport_smpl_5[11:0]	ada	a5a	aa9	ada	2b1	b5b	ab9	bdb	2c1
mySystem_i/JesdSubSys/Transport_smpl_6[11:0]	ada	a5a	aa9	ada	2b1	b5b	ab9	bdb	2c1
mySystem_i/JesdSubSys/Transport_smpl_7[11:0]	ada	a5a	aa9	ada	2b1	b5b	ab9	bdb	2c1
mySystem_i/JesdSubSys/Transport_smpl_8[11:0]	fae	7a6	aa9	fae	2b1	7b6	ab9	fbe	2c1
mySystem_i/JesdSubSys/Transport_smpl_9[11:0]	fae	7a6	aa9	fae	2b1	7b6	ab9	fbe	2c1
mySystem_i/JesdSubSys/Transport_smpl_10[11:0]	fae	7a6	aa9	fae	2b1	7b6	ab9	fbe	2c1
mySystem_i/JesdSubSys/Transport_smpl_11[11:0]	fae	7a6	aa9	fae	2b1	7b6	ab9	fbe	2c1
mySystem_i/JesdSubSys/Transport_smpl_12[11:0]	fae	7a6	aba	fae	b3b	7b6	bbb	fbe	c3c
mySystem_i/JesdSubSys/Transport_smpl_13[11:0]	fae	7a6	aba	fae	b3b	7b6	bbb	fbe	c3c
mySystem_i/JesdSubSys/Transport_smpl_14[11:0]	fae	7a6	aba	fae	b3b	7b6	bbb	fbe	c3c
mySystem_i/JesdSubSys/Transport_smpl_15[11:0]	fae	7a6	aba	fae	b3b	7b6	bbb	fbe	c3c
mySystem_i/JesdSubSys/Transport_smpl_16[11:0]	a8a	a0a	aba	a8a	b3b	b0b	bbb	b8b	c3c
mySystem_i/JesdSubSys/Transport_smpl_17[11:0]	a8a	a0a	aba	a8a	b3b	b0b	bbb	b8b	c3c
mySystem_i/JesdSubSys/Transport_smpl_18[11:0]	a8a	a0a	aba	a8a	b3b	b0b	bbb	b8b	c3c
mySystem_i/JesdSubSys/Transport_smpl_19[11:0]	a8a	a0a	aba	a8a	b3b	b0b	bbb	b8b	c3c
mySystem_i/JesdSubSys/Tran…layer_12DJxx_0_rx_validout	1								
u_ila_0_jesd204_rx_rx_sync	1								

The **Jesd204_rx_rx_tdata** can be grouped into 8 lanes of 32 bits each. This will make viewing much easier . This can be done by selecting the required signals (select 0 to 31 for lane0, 32 to 63 for lane_1 etc.,) and right-clicking and selecting "**New Virtual bus**". Then name the new bus. After grouping into 8 lanes you will get something similar to the image below.

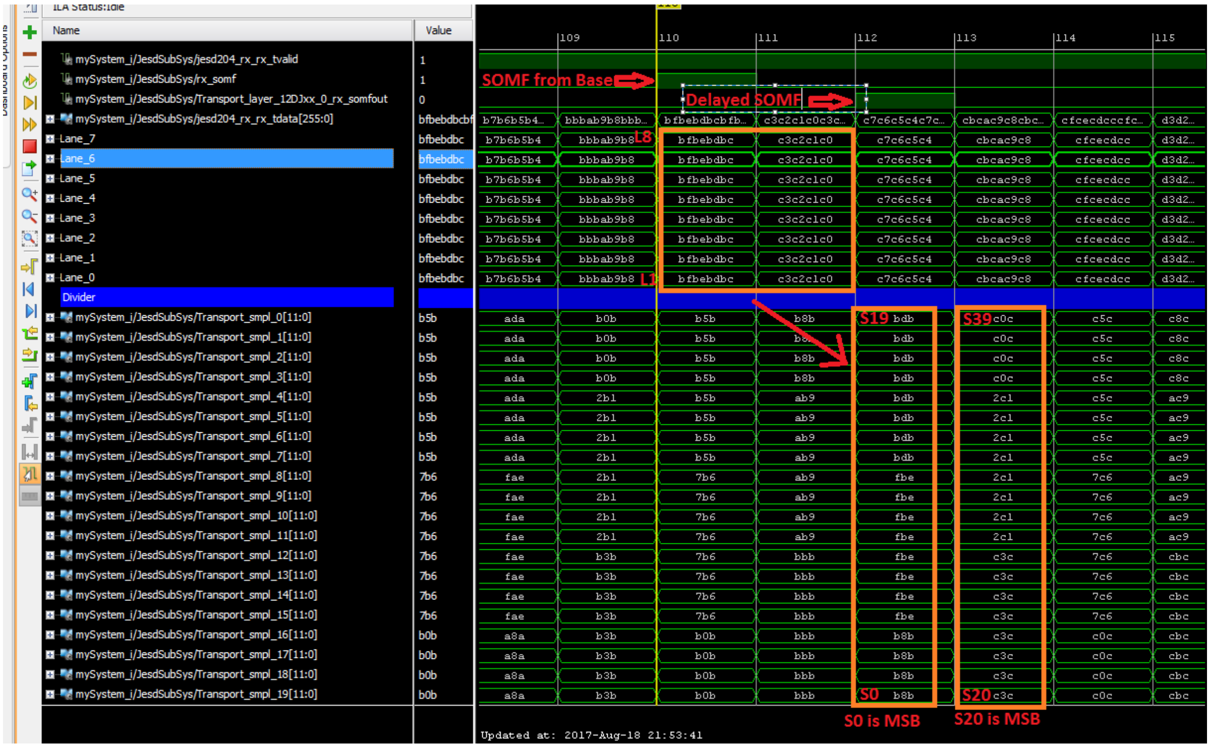
NOTE: The transport layer signals have been sliced in the firmware design itself, so that the user need not group the signals in the chipscope. This part of the block diagram can be removed if the user deems it unnecessary.



Grouping of rx_tdata into 8 lanes

Name	Value		35	36	37	38	39	40	41
mySystem_i/JesdSubSys/jesd204_rx_rx_tvalid	1								
mySystem_i/JesdSubSys/rx_somf	0								
mySystem_i/JesdSubSys/Transport_layer_12DJxx_0_rx_somfout	0								
mySystem_i/JesdSubSys/jesd204_rx_rx_tdata[255:0]	97969594	8...	93929190939...	97969594979...	9b9a99989b9...	9f9e9d9c9f9...	a3a2a1a0a3a...	a7a6a5a4a7a...	abaaa9a8aba...
Lane_7	97969594	8...	93929190	97969594	9b9a9998	9f9e9d9c	a3a2a1a0	a7a6a5a4	abaaa9a8
Lane_6	97969594	8...	93929190	97969594	9b9a9998	9f9e9d9c	a3a2a1a0	a7a6a5a4	abaaa9a8
Lane_5	97969594	8...	93929190	97969594	9b9a9998	9f9e9d9c	a3a2a1a0	a7a6a5a4	abaaa9a8
Lane_4	97969594	8...	93929190	97969594	9b9a9998	9f9e9d9c	a3a2a1a0	a7a6a5a4	abaaa9a8
Lane_3	97969594	8...	93929190	97969594	9b9a9998	9f9e9d9c	a3a2a1a0	a7a6a5a4	abaaa9a8
Lane_2	97969594	8...	93929190	97969594	9b9a9998	9f9e9d9c	a3a2a1a0	a7a6a5a4	abaaa9a8
Lane_1	97969594	8...	93929190	97969594	9b9a9998	9f9e9d9c	a3a2a1a0	a7a6a5a4	abaaa9a8
Lane_0	97969594	8...	93929190	97969594	9b9a9998	9f9e9d9c	a3a2a1a0	a7a6a5a4	abaaa9a8
Divider									
mySystem_i/JesdSubSys/Transport_smpl_0[11:0]	8d8	858	888	8d8	909	959	989	9d9	a0a
mySystem_i/JesdSubSys/Transport_smpl_1[11:0]	8d8	858	888	8d8	909	959	989	9d9	a0a
mySystem_i/JesdSubSys/Transport_smpl_2[11:0]	8d8	858	888	8d8	909	959	989	9d9	a0a
mySystem_i/JesdSubSys/Transport_smpl_3[11:0]	8d8	858	888	8d8	909	959	989	9d9	a0a
mySystem_i/JesdSubSys/Transport_smpl_4[11:0]	8d8	858	a89	8d8	291	959	a99	9d9	2a1
mySystem_i/JesdSubSys/Transport_smpl_5[11:0]	8d8	858	a89	8d8	291	959	a99	9d9	2a1
mySystem_i/JesdSubSys/Transport_smpl_6[11:0]	8d8	858	a89	8d8	291	959	a99	9d9	2a1
mySystem_i/JesdSubSys/Transport_smpl_7[11:0]	8d8	858	a89	8d8	291	959	a99	9d9	2a1
mySystem_i/JesdSubSys/Transport_smpl_8[11:0]	f8e	786	a89	f8e	291	796	a99	f9e	2a1
mySystem_i/JesdSubSys/Transport_smpl_9[11:0]	f8e	786	a89	f8e	291	796	a99	f9e	2a1
mySystem_i/JesdSubSys/Transport_smpl_10[11:0]	f8e	786	a89	f8e	291	796	a99	f9e	2a1
mySystem_i/JesdSubSys/Transport_smpl_11[11:0]	f8e	786	a89	f8e	291	796	a99	f9e	2a1
mySystem_i/JesdSubSys/Transport_smpl_12[11:0]	f8e	786	8b8	f8e	939	796	9b9	f9e	a3a
mySystem_i/JesdSubSys/Transport_smpl_13[11:0]	f8e	786	8b8	f8e	939	796	9b9	f9e	a3a
mySystem_i/JesdSubSys/Transport_smpl_14[11:0]	f8e	786	8b8	f8e	939	796	9b9	f9e	a3a
mySystem_i/JesdSubSys/Transport_smpl_15[11:0]	f8e	786	8b8	f8e	939	796	9b9	f9e	a3a
mySystem_i/JesdSubSys/Transport_smpl_16[11:0]	888	808	8b8	888	939	909	9b9	989	a3a
mySystem_i/JesdSubSys/Transport_smpl_17[11:0]	888	808	8b8	888	939	909	9b9	989	a3a
mySystem_i/JesdSubSys/Transport_smpl_18[11:0]	888	808	8b8	888	939	909	9b9	989	a3a
mySystem_i/JesdSubSys/Transport_smpl_19[11:0]	888	808	8b8	888	939	909	9b9	989	a3a

Transport layer data out for JMODE0, 6G Lane Rate with Ramp test pattern is below



The transport layer is implemented only for the modes JMODE0 and JMODE2. Both the modes follow similar interleaved pattern and have the same lane mapping

From the ADC datasheet, the sample format can be obtained as follows

Sample Pattern for one frame of JMODE0

Octet	0		1		2		3		4		5		6		7	
Nibble	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DA0	S0			S8			S16			S24			S32			T
DA1	S2			S10			S18			S26			S34			T
DA2	S4			S12			S20			S28			S36			T
DA3	S6			S14			S22			S30			S38			T
DA4	S1			S9			S17			S25			S33			T
DA5	S3			S11			S19			S27			S35			T
DA6	S5			S13			S21			S29			S37			T
DA7	S7			S15			S23			S31			S39			T

Sample Pattern for one frame of JMODE2

Octet	0		1		2		3		4		5		6		7	
Nibble	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DA0	A0			A4			A8			A12			A16			T
DA1	A1			A5			A9			A12			A17			T
DA2	A2			A6			A10			A14			A18			T
DA3	A3			A7			A11			A15			A19			T
DB0	B0			B4			B8			B12			B16			T
DB1	B1			B5			B9			B13			B17			T
DB2	B2			B6			B10			B14			B18			T
DB3	B3			B7			B11			B15			B19			T

User should take the transport layer data along with SOMF (Start of Multi-frame) “rx_somfout” signal (Transport_layer_12DJxx_0_rx_somfout) and rx_validout (Transport_layer_12DJxx_0_rx_validout) signals. These signals are in alignment with transport layer rx_dataout signal (Transport_layer_12DJxx_0_rx_dataout).

rx_dataout follows the following sequence for JMODE0

- The first link clock on the rising edge of rx_somfout, rx_dataout signal contains samples from first half of the frame [S0, S1,.., S19] with S0 as MSB
- On the second link clock, rx_dataout contains samples from second half of the frame [S20, S21,.., S39] with S20 as MSB
- On the third link clock, rx_dataout contains samples from the first half of successive frame [S0, S1, S2, …, S19] with S0 as MSB and the sequence repeats

rx_dataout follows the following sequence for JMODE2

- The first link clock on the rising edge of rx_somfout, rx_dataout signal contains samples from first half of the frame, both channels included

  [A0, B0, A1, B1, A2, B2.., A9, B9] with A0 as MSB

- On the second link clock, rx_dataout contains samples from second half of the frame, both channels included

  [A10, B10, A11, B11, …, A19, B19] with A10 as MSB

- On the third link clock, rx_dataout contains samples from the first half of successive frame, both channels included

  [A0, B0, A1, B1, A2, B2.., A9, B9] with A0 as MSB and the sequence repeats.

V. STATUS LEDS

Two signals have been added in the top module for debugging

rx_sync: This signal refers to the SYNC out from JESD Base IP and is given to LED D0 on board. It will be OFF if SYNC is lost. Under normal process, this LED will be ON

rxoutclk: This signal indicates if the link clock (lane rate/40 clock) generated from the PHY module. This is connected to LED D4.

Apart from the above two LEDs, few other signals are assigned to LED mainly to prevent logic deletion by Fitter tool and it can be ignored

Note: Both the LEDs are active high