KCU105 DAC38RF82 JESD REFERENCE DESIGN USER GUIDE

Table of Contents

I. **DESCRIPTION……2**

II. **HARDWARE SETUP…….....2**

III. **COMPILING VIVADO PROJECT……...3**

1. **Restore the Vivado project…….……..……3**

2. **Compiling the project…….…….....3**

IV. **TESTING WITH DAC38RF82…….…….....….3**

1. **Configure DAC…….……3**

2. **Programming using Vivado Hardware Manager……...…4**

3. **Setting the FMC voltage…….....9**

4. **Checking Results in Chipscope…….……11**

5. **Scope and Spectrum Analyser……..13**

V. **FIRMWARE INFORMATION……14**

1. **Generation of Sinewave……...……..14**

2. **Changing the Sinewave frequency……15**

VI. **STATUS LEDS…….……18**

I. DESCRIPTION

The design "KCU105_DAC38RF82_7p68G.zip" is developed for KCU105 board for the mode: 84111 of DAC38RF82. It has JESD Base IP and JESD PHY IP to get the tone generated within the firmware and sends it to the DAC38RF82 after conversion to conform to the JESD protocol. The design is compiled for 7.68G lane rate. The design has a simple transport layer specific for that mode (84111) that receives the samples generated by the DDS compiler in firmware and, re-order the bits and give out 16 samples every clock cycle. The results can be verified using Chipscope or using a Scope (or a spectrum analyzer). 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 7.68G. For any other line rate the firmware needs to be recompiled for that specific linerate.

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 DAC data in Chipscope. Section V describes the technical information about the firmware. Section VI 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 DAC38RF82 EVM	
sysref	SYSREF Signal from the DAC38RF82 EVM	
Txp_in/Txn_in	Serial Data to the DAC (LVDS lines)	
tx_dataout	Transport Layer data out (20 samples for every link clock)	Output from the transport module
tx_tready	Ready signal indicating JESD BASE IP is ready for data	Output from JESD BASE IP

DAC38RF82 RevE EVM is used for testing.

II. **HARDWARE SETUP**

Connect the KCU105 board with DAC38RF82. 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.

Connect a clock of 384Mhz of 6dBm to SMA J4 and remove the shunt connecting pin1 and 2 of jumper JP10. Keep all other hardware settings in the default configuration.

Also connect a Scope or Spectrum Analyzer to the DAC EVM.

III. **COMPILING VIVADO PROJECT**

1. Extracting the zip file.

- Right click the KCU105_DAC38RF82_7p68G.xpr.zip file and then press "Extract All". After extracting, open the project using **Vivado 2016.1**. The project file is
  KCU105_ DAC38RF82_7p68G.xpr\prj_MyKcu105_TI\ prj_MyKcu105_TI.xpr

2. Compiling the project.

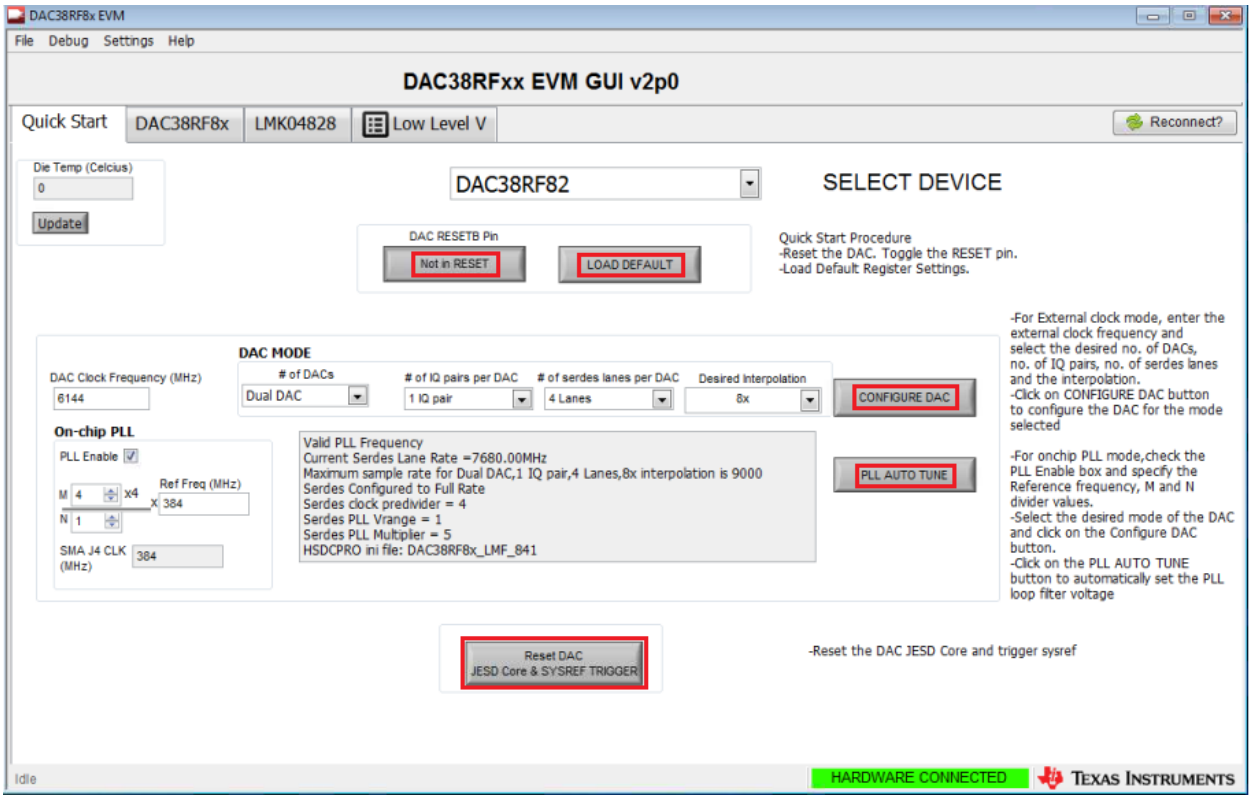
- In Vivado, on the left you will see the Flow Navigator. Press the generate Bitstream option under "Program and Debug" to generate the output ".bit" file.
- Once the generation is over a .bit file will be generated, along with with .ltx file (required for chipscope).
- The bit file generated is "mySystem_wrapper.bit". 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_DAC38RF82_7p68G.xpr\prj_MyKcu105_TI\prj_MyKcu105_TI.runs\impl_1\

IV. **TESTING WITH DAC38RF82**

1. Configure DAC

- The reference clock from DAC should be stable before downloading firmware. So, the DAC has to be configured first before we program the board.
- Open DAC38RF82 GUI v2p0 and perform the follow actions:

- Reset the DAC. Toggle the Reset Pin.
- Load the Default Register Settings.
- Set the following values:
    - DAC Clock Frequency: 6144 Mhz
    - # of DACs : Dual DAC
    - # of IQ pairs per DAC : 1 IQ pair
    - # of serdes lanes per DAC : 4 lanes
    - Desired Interpolation : 8x
    - Check the PLL enable checkbox for On-chip PLL
- Press the Configure DAC button.
- Press on PLL Auto tune
- Finally press on Reset DAC JESD Core & SYSREF TRIGGER.

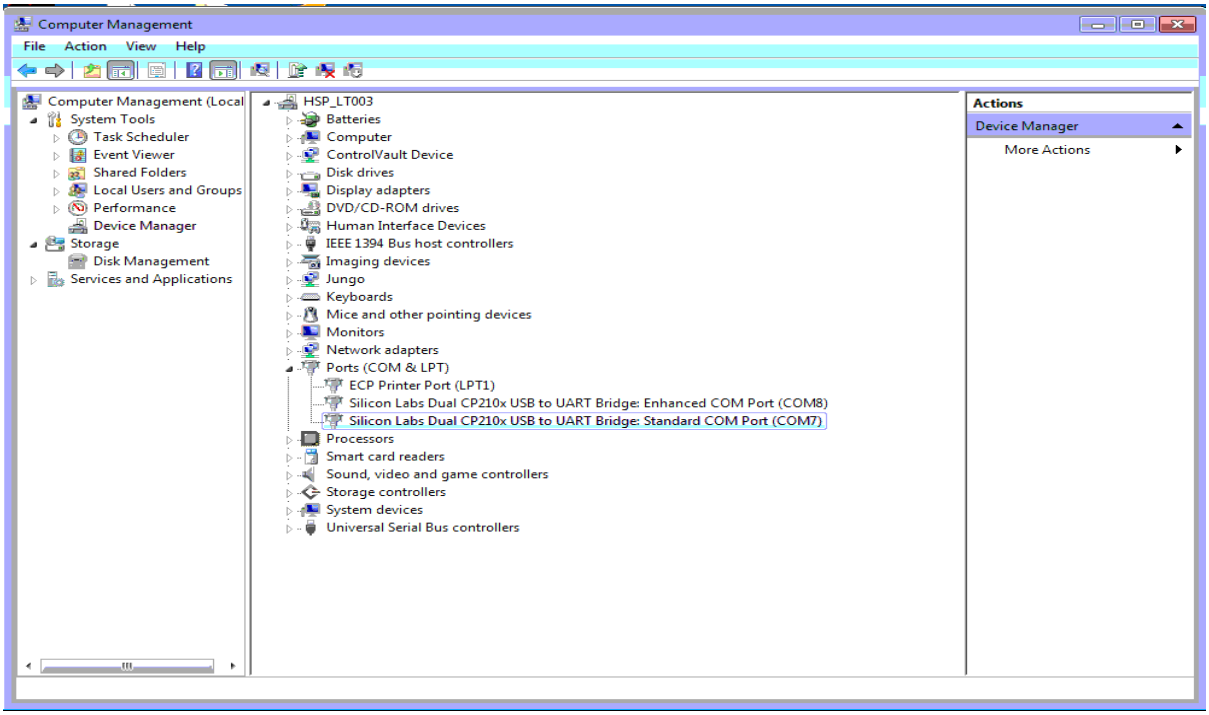


2. Programming with Vivado Hardware Manager

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

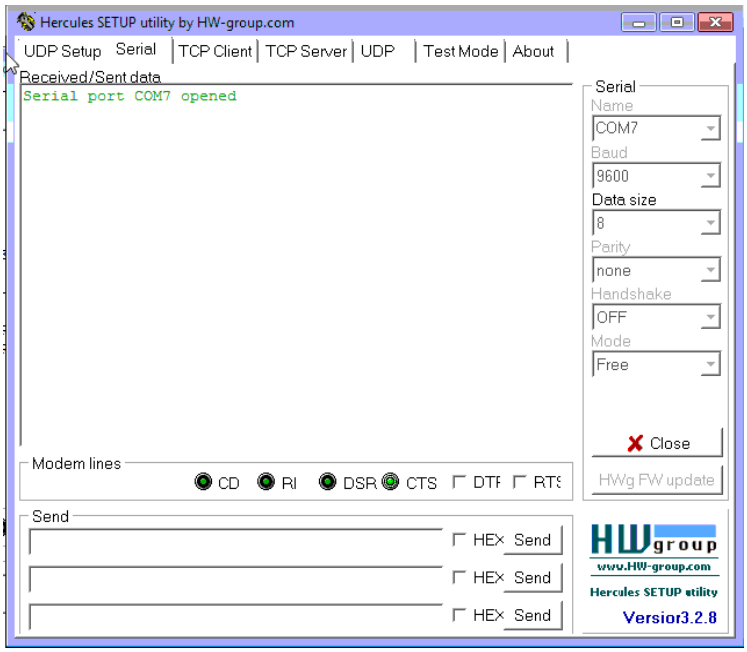
1) Make sure to configure the DAC EVM before programming the device.

2) Open Device manager and check for the COM ports. (Make sure you have connected to the UART port of the board)

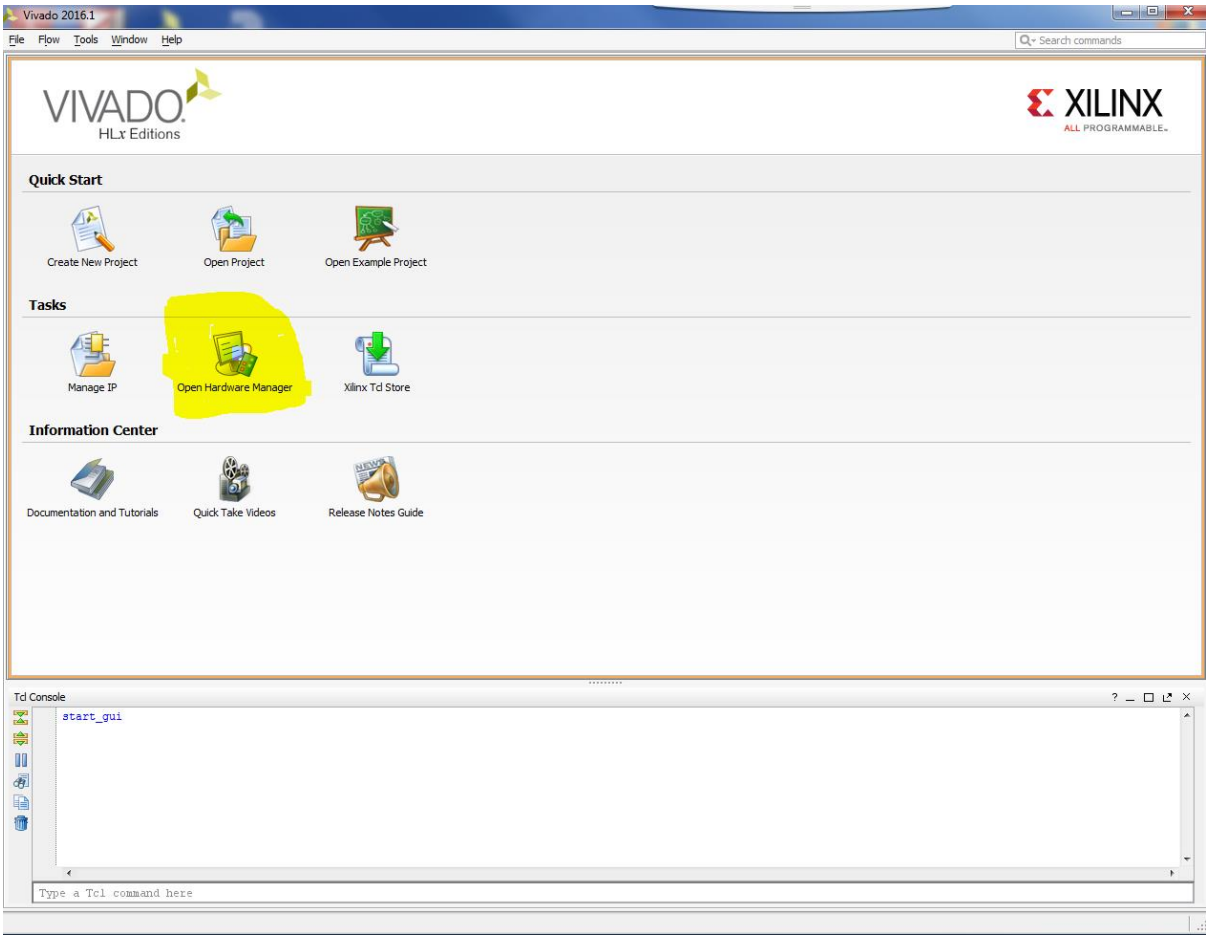


In this case, we have COM Port 8 (Enhanced) and COM Port 7 (Standard).

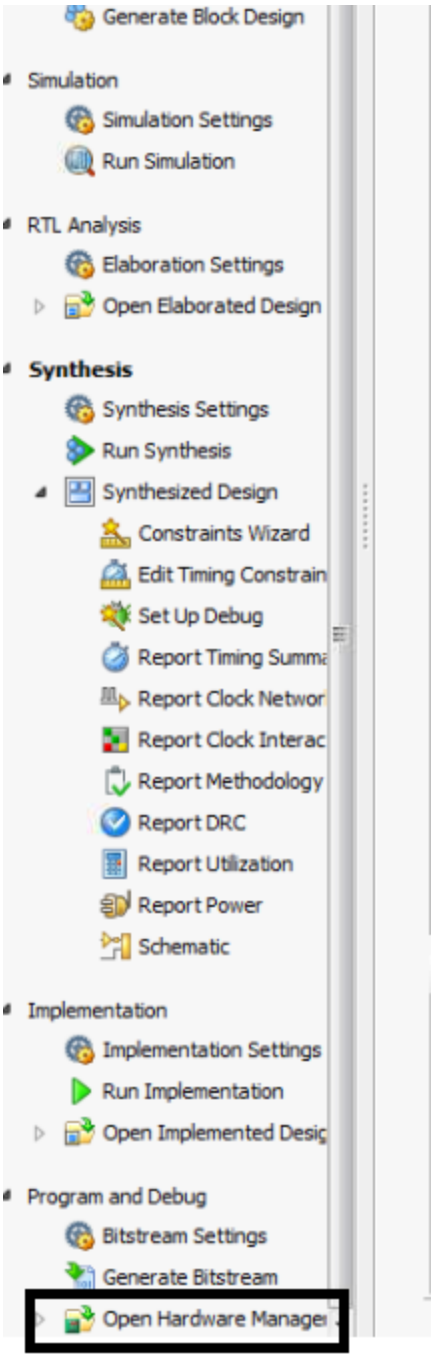
3) Open both the COM ports using any Utility (like Hercules). For **Standard COM** Port(in this case COM port 7), make sure of the below settings while opening the port.



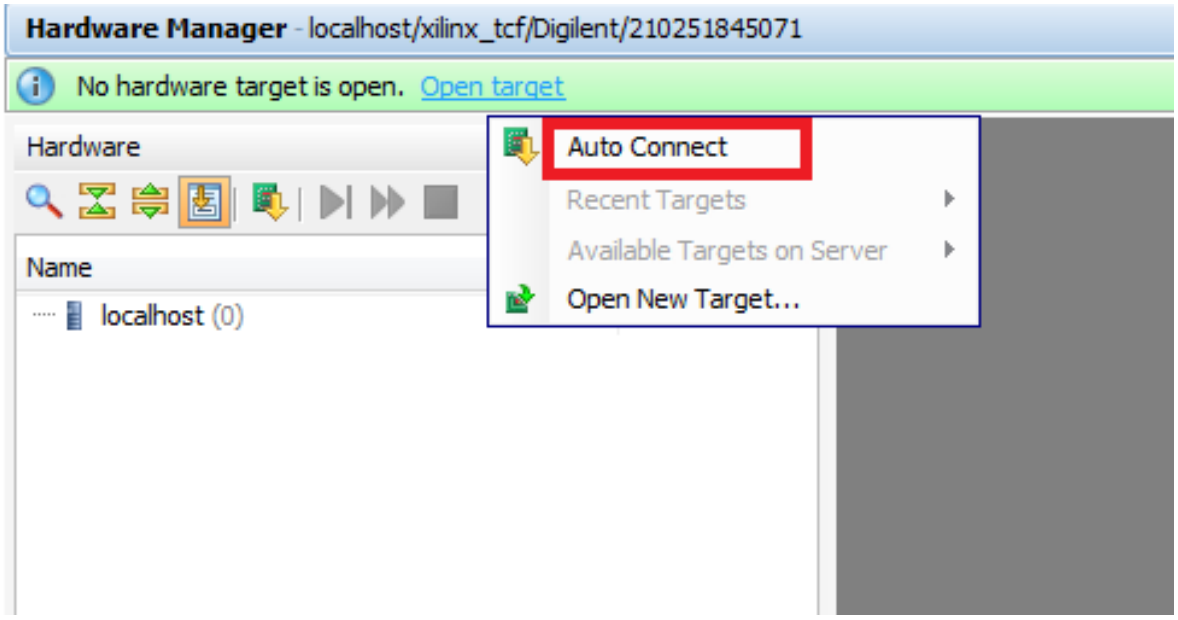
4) Open the Vivado Hardware manager. It can be opened from the GUI start screen (or) the Vivado flow navigator under “Program and Debug”.



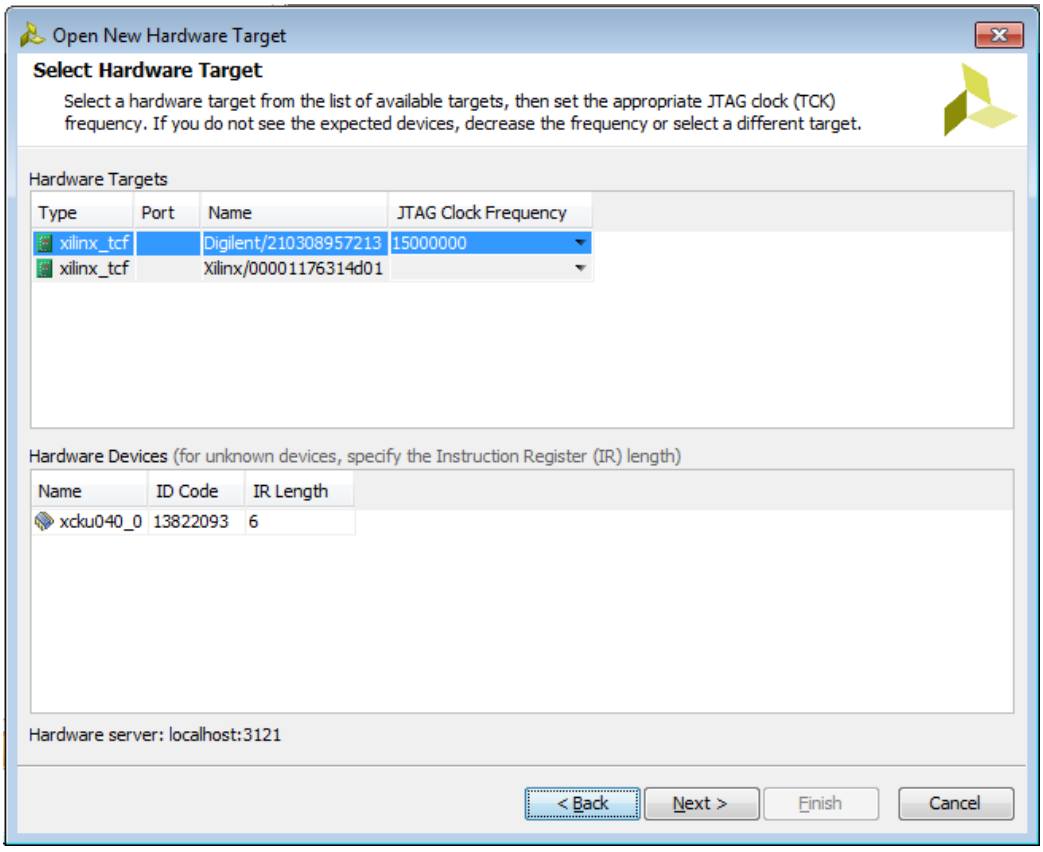
(OR)


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

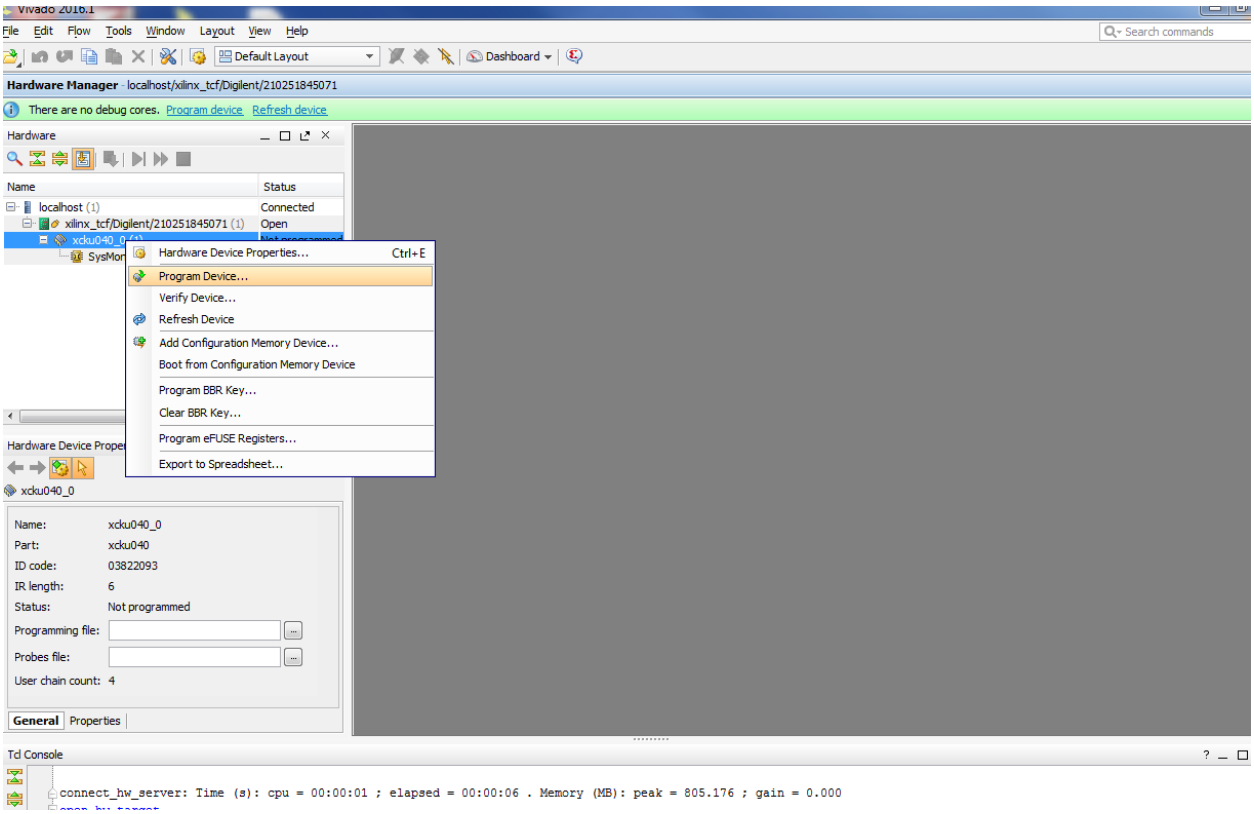
5) Press on Open Target and then press on Auto connect.



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.



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



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**".

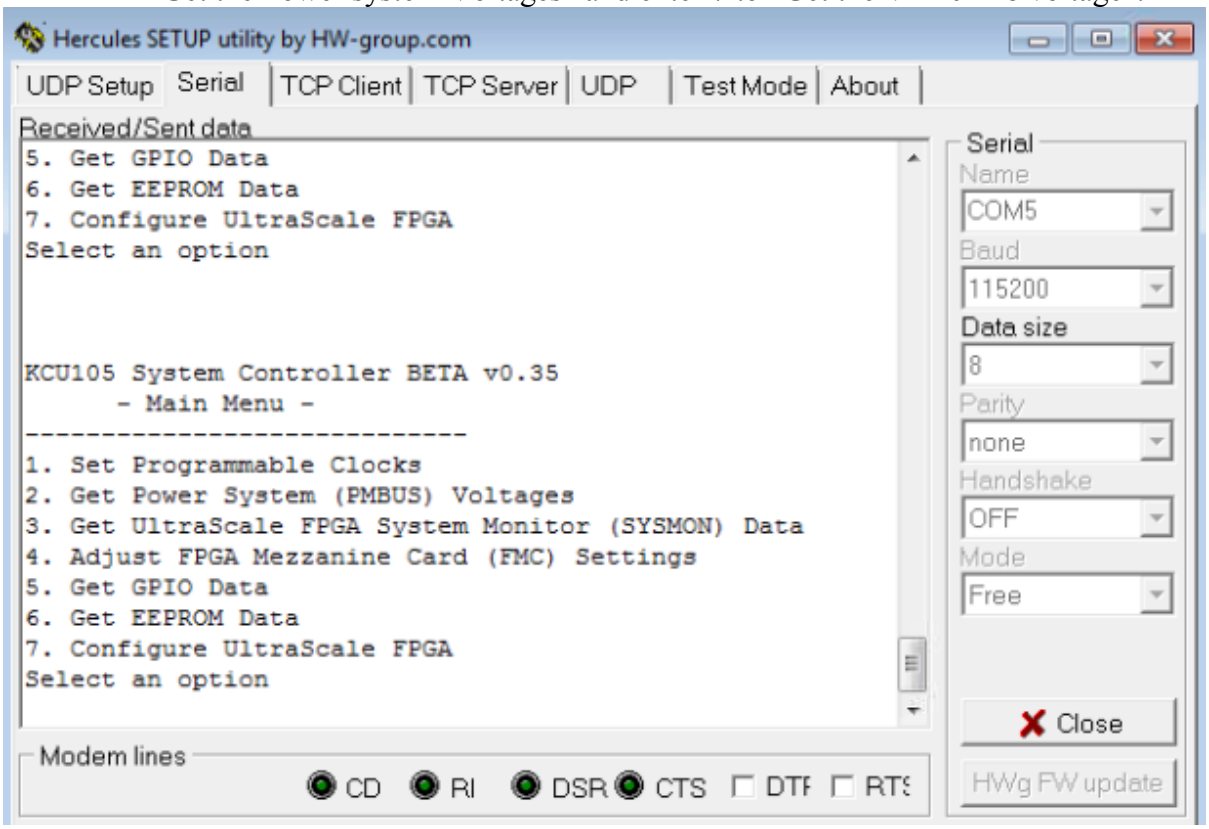
3. **Setting the FMC voltage**

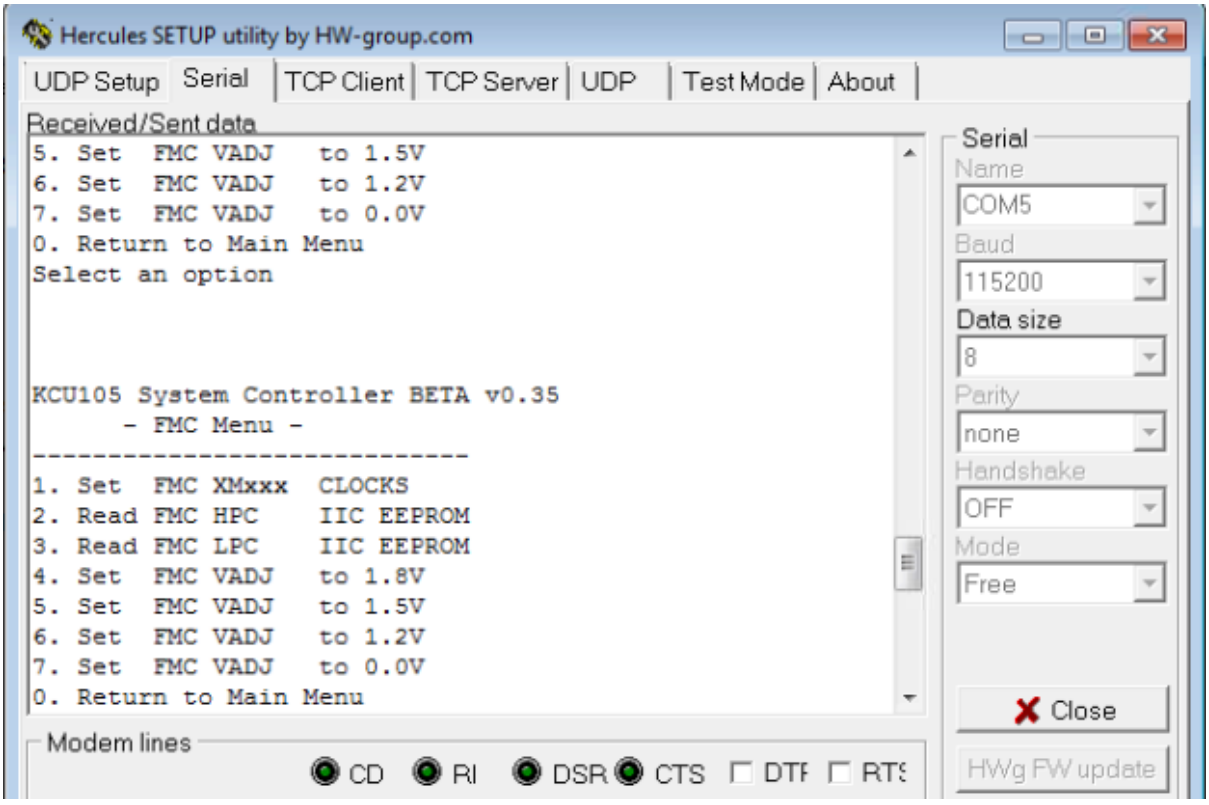
We need to set the FMC voltage of the KCU to 1.8V before we capture. In order to set this please make sure that you have connected the UART of the board to the PC. This setting needs to be done only once after the KCU has been connected and turned ON. For subsequent captures, this section can be skipped.

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 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 Mezzanine Card (FMC) settings
- Again enter 4 to Set FMC VADJ to 1.8V.

- To read the voltage value, 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”.





4. **Checking Results in Chipscope**

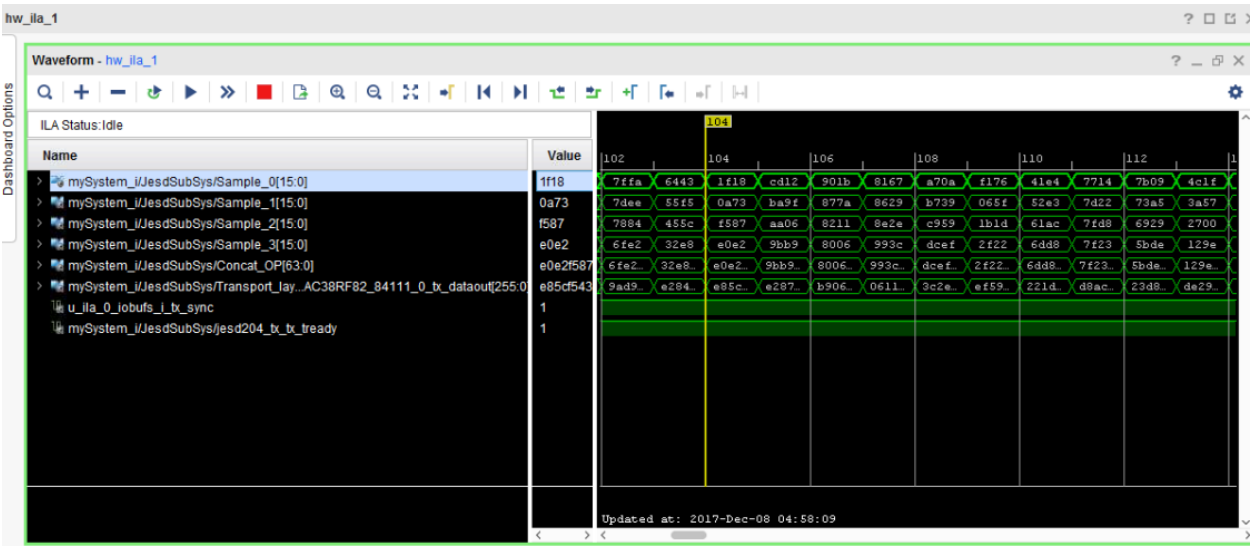
Once the development kit is programmed, user can view the results in Chipscope (Vivado Hardware Manager) which probes signals from the board.

Signals which are currently probed are

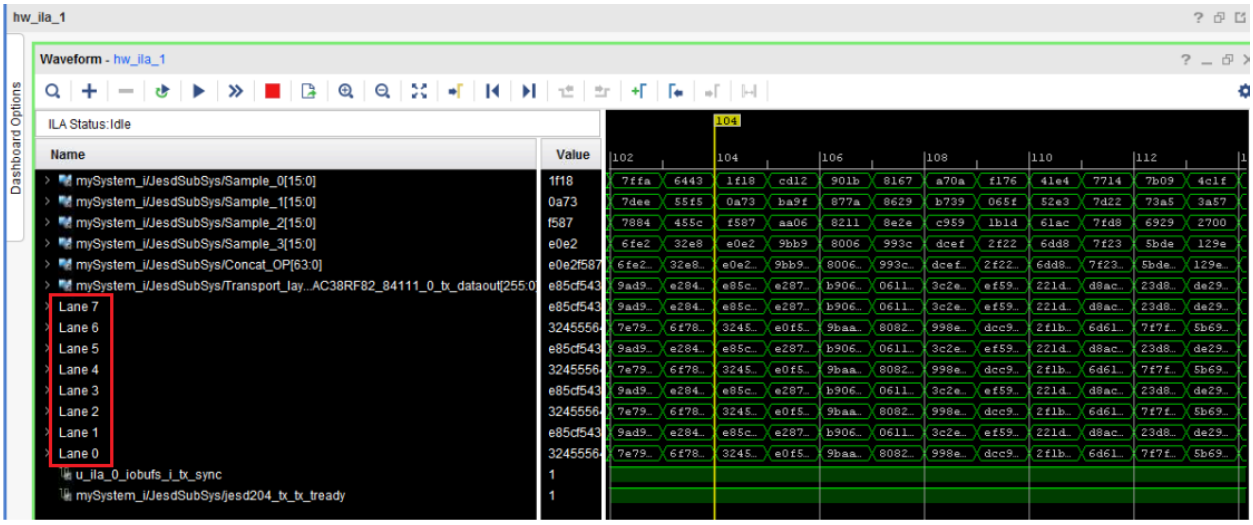
- **Tx_sync -** Active low SYNC signal from JESD Base IP. If SYNC is established, this signal will be high.
- **Transport_layer_DAC38RF82_84111_0_tx_dataout[255...0]** – A 256 bit bus of data coming from the Transport layer. It contains sixteen 16-bit samples in each clock cycle.
- **Sample_0….Sample_3 –** The output samples coming from each DDS compiler (see Firmware Information section).
- **Concat_OP** – The concatenated signals of Sample_0 to Sample_3 signals to create a 64-bit bus. This bus consists of four 16-bit samples, with the Sample_0 output being in the LSB position.

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

The hw_ila_1 window is shown below.



The **Transport_layer_DAC38RF82_84111_0_tx_dataout[255...0]** 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.

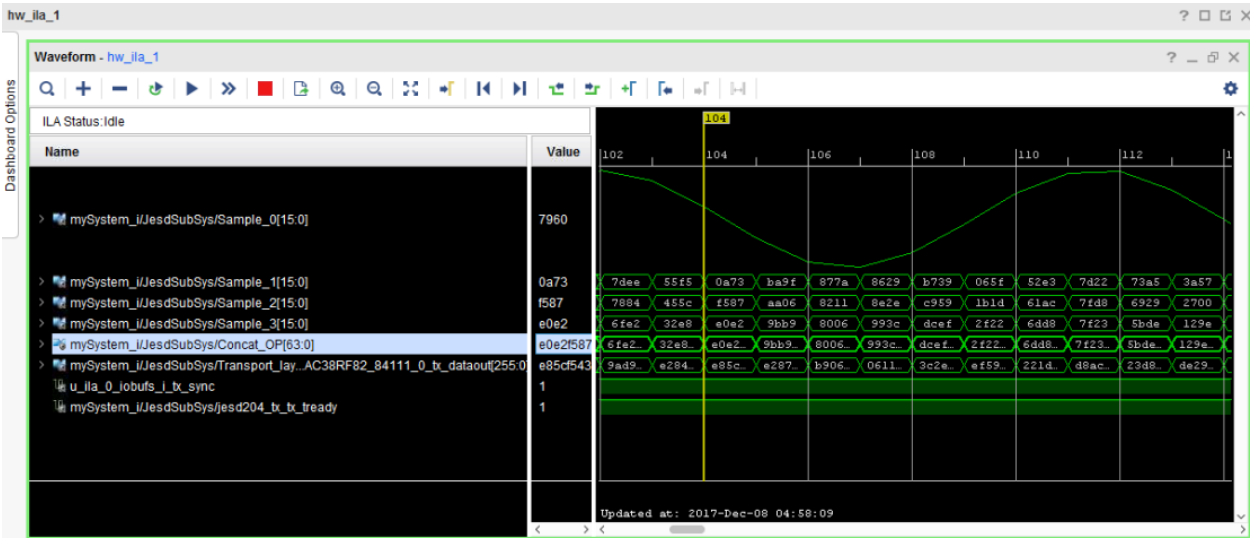


The Concat_OP is a 64-bit bus consisting of four 16-bit samples. In this mode 4 samples are divided across 2 lanes for every clock cycle. Hence the output of the transport layer is a 256 bit bus, which contains sixteen 16-bit samples for 8 lanes.

It is possible to view the output of each DDS as a 20Mhz wave, within the chipscope.

The procedure:

1. Right click -> Radix -> Signed decimal
2. Right click -> Waveform Style -> Analog



The transport layer is implemented only for the mode 84111.

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

Sample Pattern for one frame of 84111

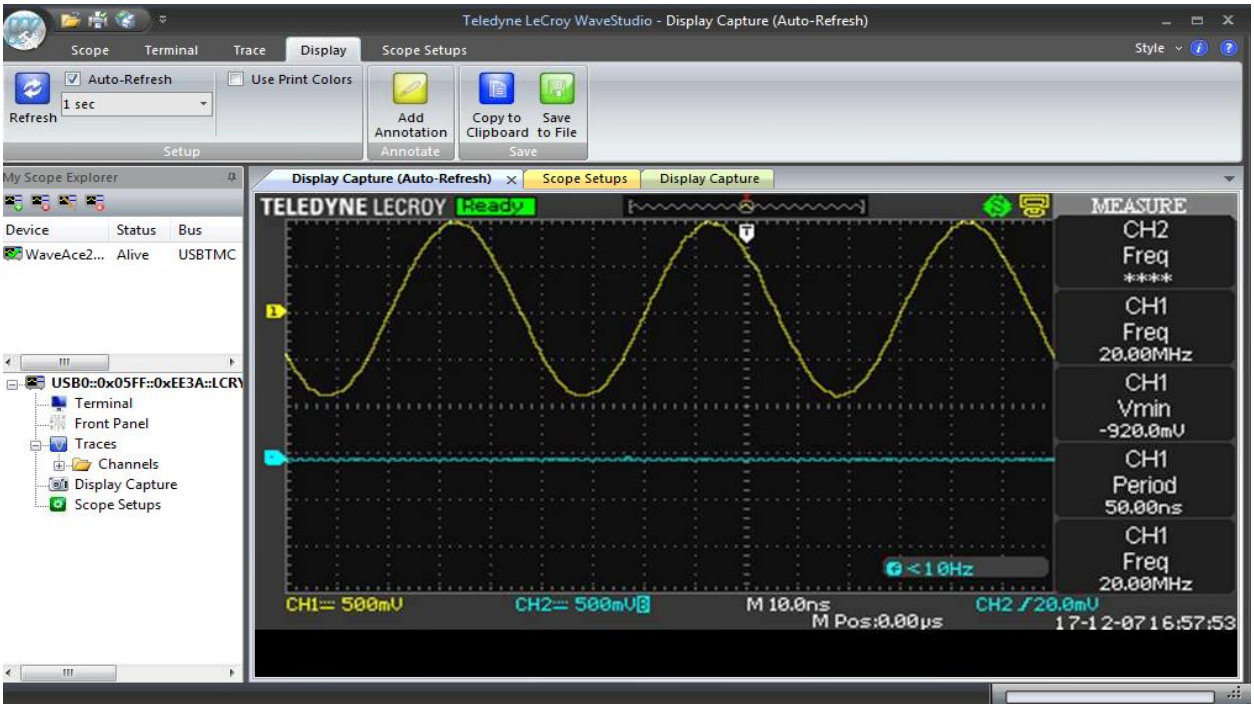
Bytes	**0**	
Nibbles	**0**	**1**
Lane RX0	A-i0[15:8]	
Lane RX1	A-i0[7:0]	
Lane RX2	A-q0[15:8]	
Lane RX3	A-q0[7:0]	
Lane RX4	B-i0[15:8]	
Lane RX5	B-i0[7:0]	
Lane RX6	B-q0[15:8]	
Lane RX7	B-q0[7:0]	

Transport_layer_DAC38RF82_84111_0_tx_dataout follows the following sequence

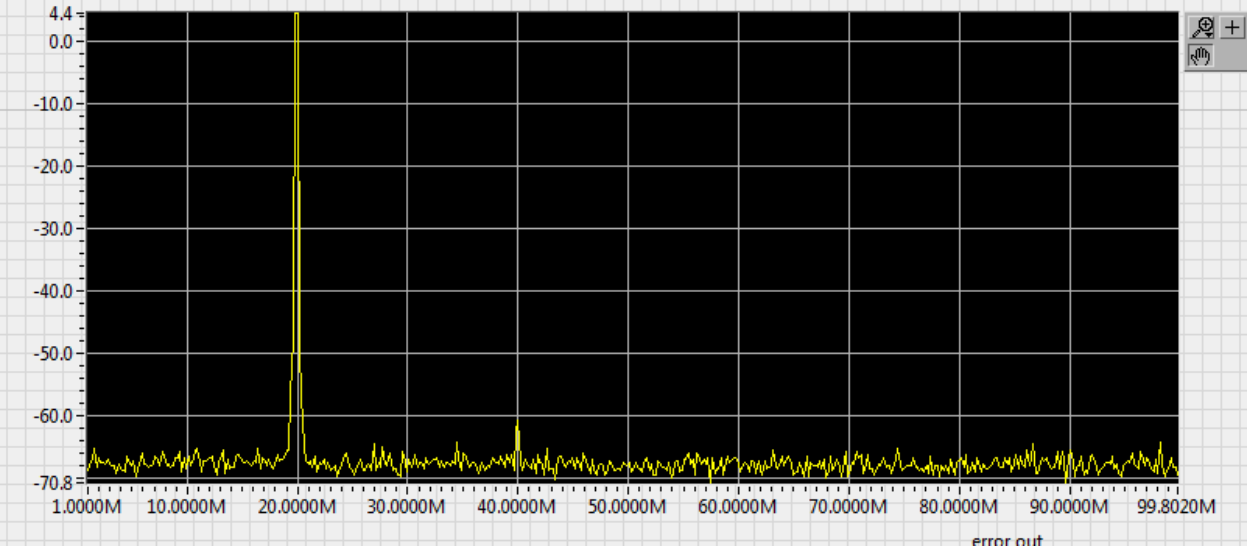
- The first link clock , **Transport_layer_DAC38RF82_84111_0_tx_dataout** signal contains 16 samples. All the even lanes (0,2,4,6) contain MSB 8 bits of a sample (15:8) and all the Odd lanes (1,3,5,7) contain the LSB 8 bits of a sample (7:0)
- Each lane contains MSB/LSB of 4 samples with the 1st sample of each cycle being the LSB for each lane.

5. Scope and Spectrum Analyzer

The 20 Mhz tone as seen in the Scope output:



The 20 Mhz tone as seen in the Spectrum Analyser:



V. FIRMWARE INFORMATION

This FW (and Transport Layer) has been made specifically for 84111 mode. A sinewave is being generated within the FW. This Sine wave is being sent continuously (free-running) to the DAC from the FPGA.

1. Generation of Sinewave:

- A DDS Compiler is being used within the FW in order to generate the Sinewave. In this design, a Sine wave of frequency of 20 Mhz is being generated.
- The DDS compiler will require a sampling clock as input. In this firmware, the sampling frequency of the DDS compiler is 192 Mhz.
- For every link clock cycle we need to send 32 bits of data to the JESD Base IP. Hence we use 4 instances of DDS compiler modules to generate 4 waves of 20 Mhz each. However each instance is offset by an equal value (for this mode the phase offset is one-fourth of the output wave i.e., 20/4 = 5Mhz) such that in each link clock 4 samples of a 20 Mhz Sinewave is generated. All four of these samples are concatenated (Concat_OP signal) and given as an input to the transport layer.
- The concatenated output (Concat_OP) consists of four 16-bit samples. In this mode 4 samples are divided across 2 lanes for every clock cycle. Hence the output of the transport layer is a 256 bit bus, which contains sixteen 16-bit samples for 8 lanes.
- The DAC sampling rate for this lane rate (7.68G) is 768 Msps. Each DDS Compiler instance generates a 20Mhz wave w.r.t a sampling rate of 192 MHz. Therefore when we

combine the outputs of 4 of such instances for a single cycle, a sine wave of 20 Mhz is generated for a sampling rate of 768 Msps.

- In this design, the DDS compilers generate a free running sine wave output. Hence every clock cycle is a start of frame. Thus the output of the transport layer is directly connected to the JESD TX Base IP.

2. Changing the Sinewave frequency:

- In order to change the generated Sinewave frequency, we need to change the phase offset and phase increment values.

  (calculation has been presented for 20 Mhz sinewave case)

  - Phase offset = [2^Phase width /(Sampling frequency/Required frequency)]/4
    = [2^16/(192/20)]/4
    = [65536/9.6]/4
    = 6826.667/4
    = 1706.6667

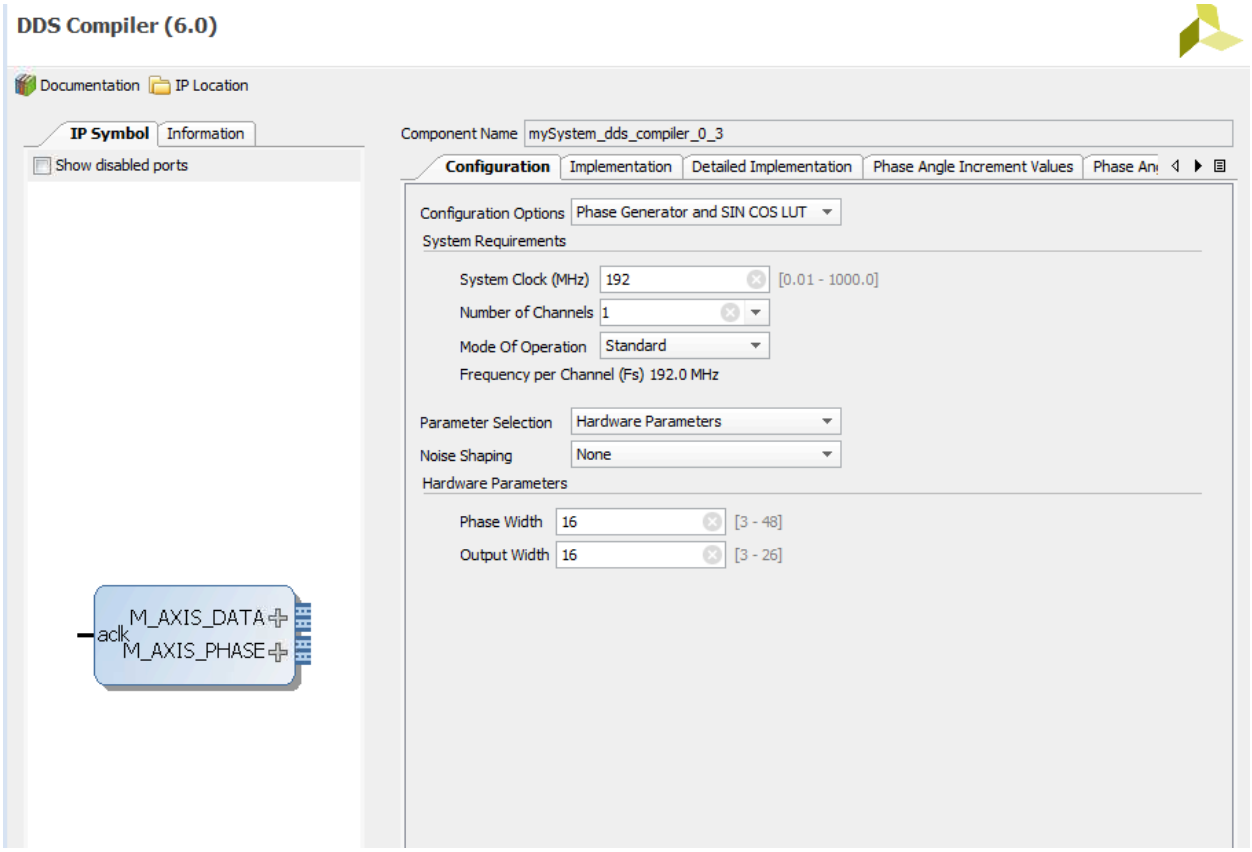
    By rounding off to the nearest integer we get 1707 which is 6AB in hexadecimal or 11010101011 in binary.

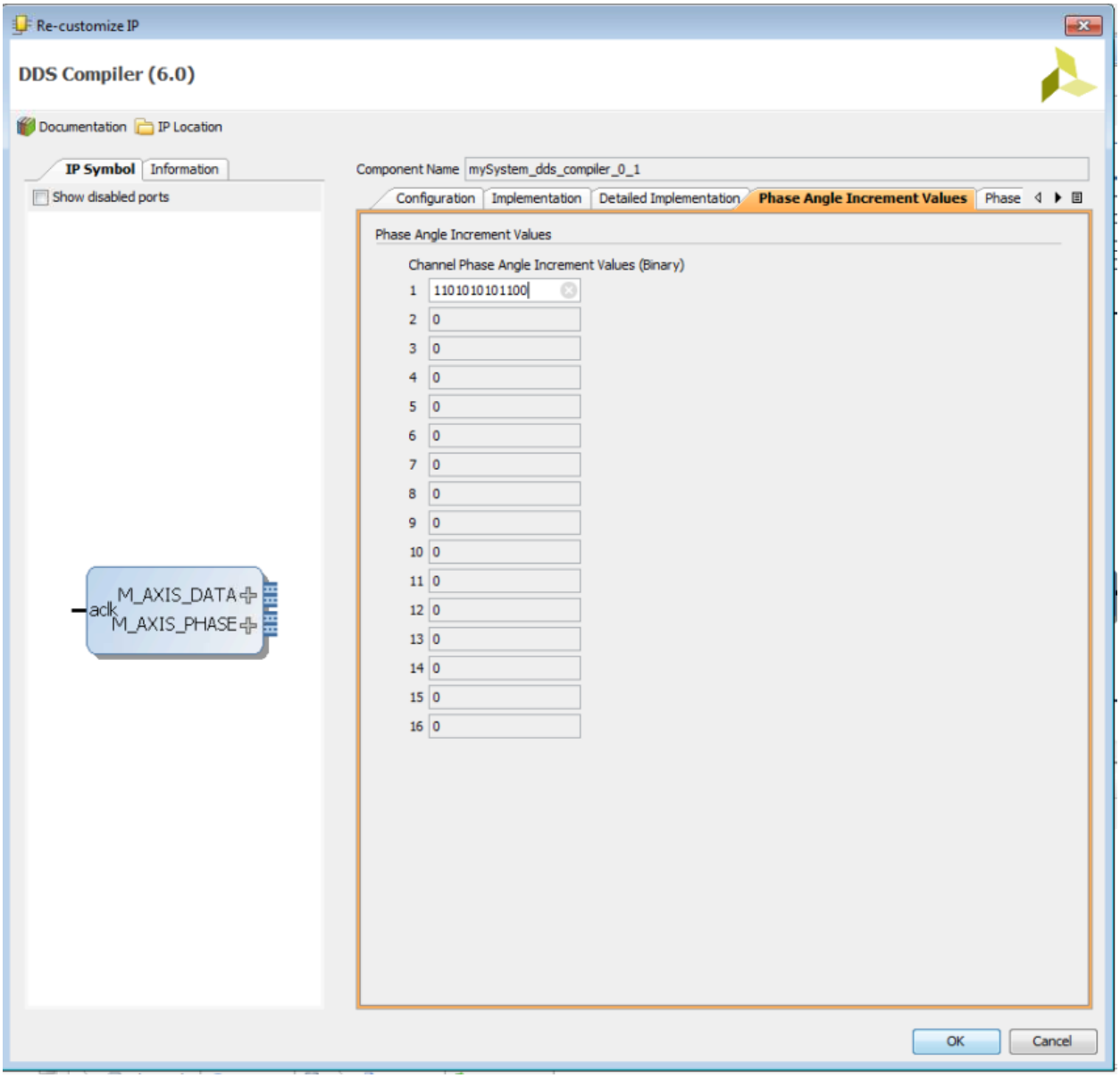
  - Phase increment = 2^Phase width /(Sampling frequency/Required frequency)
    = 2^16/(192/20)
    = 65536/9.6
    = 6826.667

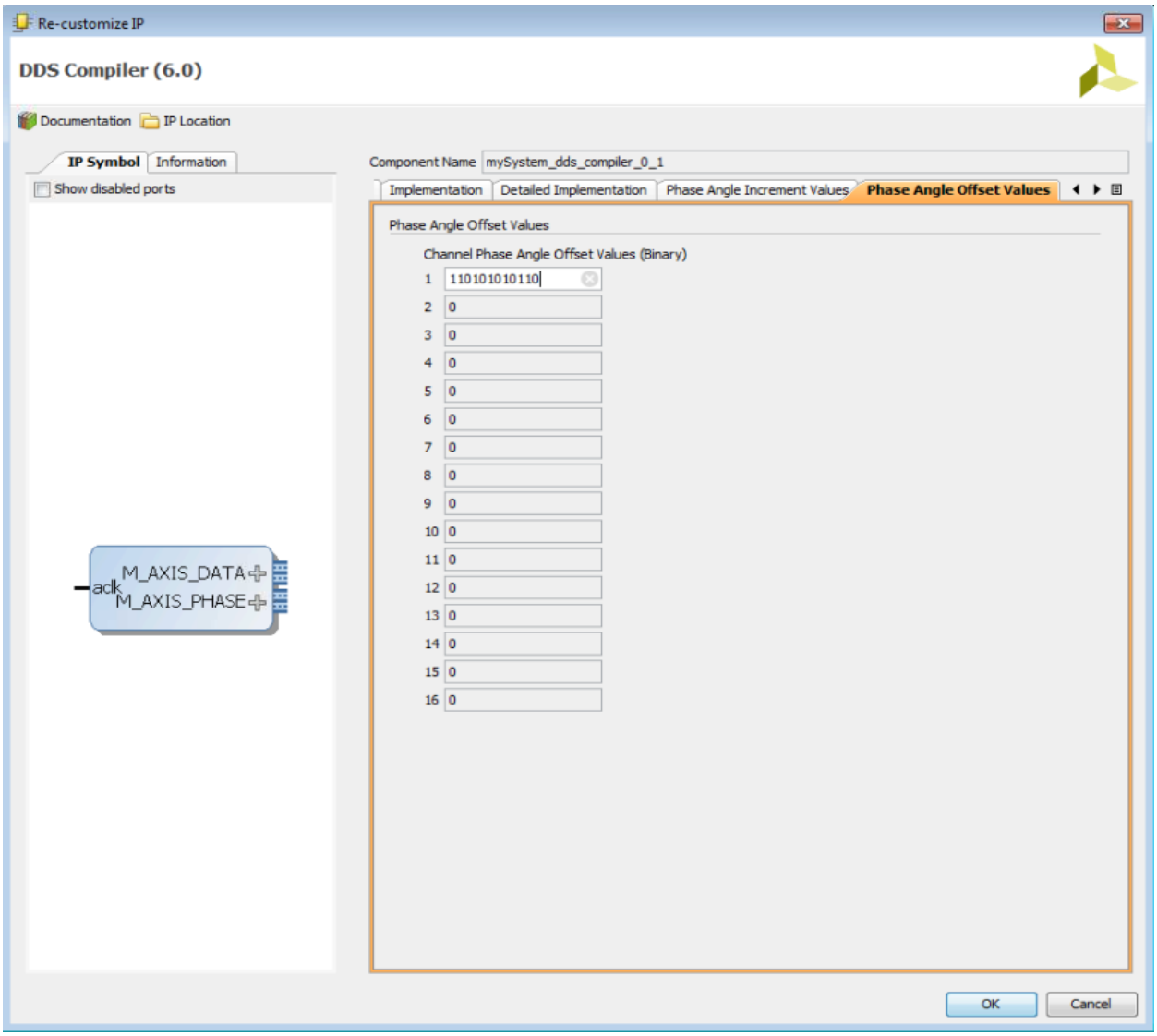
    By rounding off to the nearest integer and making it a multiple of phase offset, we get 6828 which is 1AAC in hexadecimal or 1101010101100 in binary.

For more detailed information about the DDS compiler module please refer to the Xilinx document Pg141 available at the following link:

https://www.xilinx.com/support/documentation/ip_documentation/dds_compiler/v6_0/pg141-dds-compiler.pdf


DDS Compiler (6.0)
Documentation
IP Location
IP Symbol
Information
Show disabled ports
Component Name
mySystem_dds_compiler_0_3
Configuration
Implementation
Detailed Implementation
Phase Angle Increment Values
Configuration Options
Phase Generator and SIN COS LUT
System Requirements
System Clock (MHz)
192
[0.01 - 1000.0]
Number of Channels
1
Mode Of Operation
Standard
Frequency per Channel (Fs) 192.0 MHz
Parameter Selection
Hardware Parameters
Noise Shaping
None
Hardware Parameters
Phase Width
16
[3 - 48]
Output Width
16
[3 - 26]
aclk
M_AXIS_DATA
M_AXIS_PHASE


Re-customize IP
DDS Compiler (6.0)
Documentation
IP Location
IP Symbol
Information
Show disabled ports
aclk
M_AXIS_DATA
M_AXIS_PHASE
Component Name
mySystem_dds_compiler_0_1
Configuration
Implementation
Detailed Implementation
Phase Angle Increment Values
Phase
Phase Angle Increment Values
Channel Phase Angle Increment Values (Binary)
1 1101010101100
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0
13 0
14 0
15 0
16 0
OK
Cancel



VI. **STATUS LEDS**

Two signals have been added in the top module for debugging

tx_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

txoutclk: 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 (if any) are assigned to LED mainly to prevent logic deletion by Fitter tool and it can be ignored

Note: Both the LEDs are active high