# KCU105 DAC38RF82 JESD REFERENCE DESIGN USER GUIDE

# **Table of Contents**

I.	DESCRIPTION	2
II.	HARDWARE SETUP	2
III.	COMPILING VIVADO PROJECT 1. Restore the Vivado project	
	2. Compiling the project	3
IV.	TESTING WITH DAC38RF82         1. Configure DAC	
	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.		
	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, reorder 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	
CLK_IN1_D       Constant 300Mhz clock of cry         refclk       Reference clock from the system         systef       SYSREF Signal from the system         Txp_in/Txn_in       Serial Data (LVDS)         tx_dataout       Transport Layer data of link constant         tx_tready       Ready signal indicating	Reference clock from the DAC38RF82 EVM	Input
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.

DACSORFOXEVIVI					
File Debug Settings Help					
		DAC38RFxx EVM	GUI v2p0		
Quick Start DAC38RF8>	K LMK04828	Low Level V			Seconnect?
Die Temp (Celcius) 0		DAC38RF82	•	SELECT DEVICE	E
Update		DAC RESETB Pin	LOAD DEFAULT	Quick Start Procedure -Reset the DAC. Toggle the RESET -Load Default Register Settings.	pin.
	DAC MODE				-For External clock mode, enter the external clock frequency and select the desired no. of DACs, no. of IQ pairs, no. of serdes lanes
DAC Clock Frequency (MHz) 6144 On-chip PLL	# of DACs Dual DAC	1 IQ pair 💌 4 Lanes	es lanes per DAC Desired Inte	CONFIGURE DAC	and the interpolation. -Click on CONFIGURE DAC button to configure the DAC for the mode selected
PLL Enable M 4 3 X4 Ref Freq (M N 1 384 SMA J4 CLK 384 (MHz)	Hz) Maximum sar Serdes Confi Serdes clock Serdes PLL V Serdes PLL N	es Lane Rate =7680.00MHz mple rate for Dual DAC,1 IQ pair,4 L gured to Full Rate predivider = 4 /range = 1	anes,8x interpolation is 9000	PLL AUTO TURE	-For onchip PLL mode, check the PLL Enable box and specify the Reference frequency, M and N dvider values. -Select the desired mode of the D/ and cick on the Configure DAC button. -Click on the PLL AUTO TUNE button to automatically set the PLL loop filter voltage
		Reset DAC JESD Core & SYSREF	TRIGGER	-Reset the DAC JESD Core and	trigger sysref
lie				HARDWARE CONNECTE	🖸 🛛 🦊 Texas Instrument

### 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)

Computer Management		<b>-</b>
File Action View Help		
	全殿相	
System Tools  Other Task Scheduler  Other Task Scheduler  Other Task Scheduler  Other Tools  Ot	HSP_LT003 Batteries Computer Computer ControlVault Device Disk drives Display adapters DVD/CD-ROM drives Human Interface Devices Human Interface Devices Human Interface Devices Human Interface Devices Human Interface Devices Filler 1948 Bus host controllers Jungo Keyboards Monitors Monitors Monitors Monitors Ports (COM & LPT) 	Actions Device Manager More Actions

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

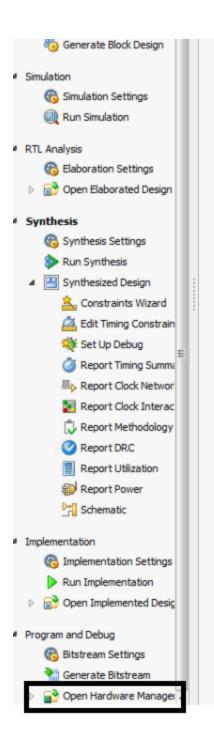
3) Open both the COM ports using any Utility (like Hercules). For **Standard COM** Port(in this case COM port 7), make sure of the below settings while opening the port.

Note: Set the set of t	
UDP Setup Serial TCP Client TCP Server UDP Test Mode About	
Received/Sent data	- Serial
Serial port COM7 opened	Name
	Baud 9600 -
	Data size
	8 -
	Parity
	none 🗾
	Handshake
	Free -
	l'
	🗙 Close
Modem lines	HWg FW update
Send	HWgroup
	www.HW-group.com
□ HE× Send	Hercules SETUP utility
☐	Versior3.2.8

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

Vivado 2016.1		
<u>Fi</u> le F <u>l</u> ow <u>T</u> ools <u>W</u> indow <u>H</u> elp	٩	<ul> <li>Search commands</li> </ul>
HLx Editions	1	E XILINX ALL PROGRAMMABLE.
Quick Start		
Create New Project Open Project Open Ex	mple Project	
Tasks		
Manage IP Open Hardware Manager Xilm	td Store	
Information Center		
Documentation and Tutorials Quick Take Videos Release	lotes Guide	
Td Console		? – 🗆 🗳 ×
start_gui       #       #       #       #       #		~
Type a Tcl command here		•
Type a fol command here		]

(OR)



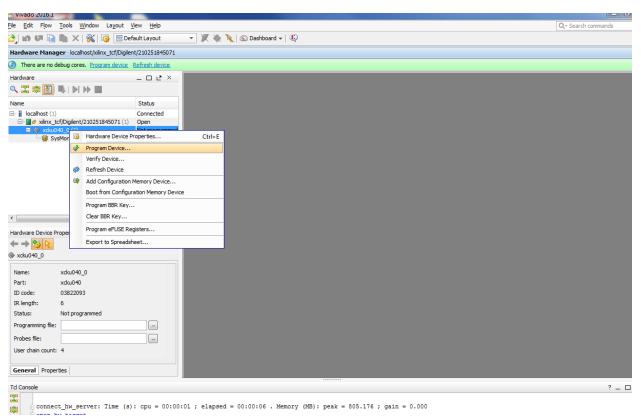
5) Press on Open Target and then press on Auto connect.

Hardware Manager - localhost/xilinx	_tcf/D	igilent/210251845071		
(i) No hardware target is open. Open	targe	<u>et</u>		
Hardware	<b>B</b>	Auto Connect		
🔍 🛣 🖨 🛃 💐 🕨 🕨 🔳		Recent Targets	►	
Name		Available Targets on Server	•	
····· localhost (0)	È	Open New 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 I	Hardware	Target			×
Select Hardy	ware Tar <u>q</u>	jet			
			available targets, then set t ed devices, decrease the fre		
Hardware Targe	ets				
Type P	Port Nar	ne	JTAG Clock Frequency		
🦉 xilinx_tcf		ent/210308957213			
xilinx_tcf	Xilin	x/00001176314d0	1 •		
Hardware Devic	es (for unk	nown devices, spe	cify the Instruction Register	(IR) length)	
Name	ID Code	IR Length			
xcku040 0	13822093	6			
Hardware serve	er: localhost	:3121			
			< <u>B</u>	ack Next >	Einish Cancel

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".

🎨 Hercules SETUP utility by HW-group.com	
UDP Setup Serial TCP Client TCP Server UDP Test Mode About	
Received/Sent data 5. Get GPIO Data 6. Get EEPROM Data 7. Configure UltraScale FPGA Select an option	Serial Name COM5 Baud 115200
<pre>KCU105 System Controller BETA v0.35</pre>	Data size       8       Parity       none       Handshake       OFF       Mode       Free
7. Configure UltraScale FPGA       Select an option       Modem lines       O CD O RI       O DSR O CTS	X Close
Harcular SETLIP utility by HW-group com	
W Hercules SETUP utility by HW-group.com	
Serial       TCP Client       TCP Server       UDP       Test Mode       About         Received/Sent data         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	Serial Name COM5 Baud 115200 Data size
UDP Setup Serial TCP Client TCP Server UDP Test Mode About Received/Sent data 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	Serial Name COM5 Baud 115200 Data size 8 Parity none
UDP Setup Serial TCP Client TCP Server UDP Test Mode About Received/Sent data 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 KCU105 System Controller BETA v0.35	Serial Name COM5 Baud 115200 Data size 8 Parity

# 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.

Naveform - hw_ila_1													? _ 0
Q   +   −   &   ►   ≫   ■   ⊡   @   Q   X   •     H   H	12 3	r +F	Fe   -	, <b>"</b>   [,,,]									
ILA Status: Idle				104									
Name	Value	102		104	1	106		108	1	110		112	
> 🏁 mySystem_i/JesdSubSys/Sample_0[15:0]	1f18	7ffa	6443	1f18	cd12	901b	8167	a70a	f176	41e4	7714	7509	4clf
> W mySystem_i/JesdSubSys/Sample_1[15:0]	0a73	7dee	5515	0a73	ba9f	877a	8629	Ъ739	065£	52e3	7422	73a5	3a57
> W mySystem_i/JesdSubSys/Sample_2[15:0]	f587	7884	455c	£587	aa06	8211	8e2e	c959	lbld	61ac	7148	6929	2700
> MySystem_i/JesdSubSys/Sample_3[15:0]	e0e2	6fe2	32e8	e0e2	9bb9	8006	993c	dcef	2f22	6dd8	7f23	5bde	129e
> mySystem_i/JesdSubSys/Concat_OP[63:0]	e0e2f587	6fe2	32e8	e0e2	9bb9	8006	993c		2 £ 2 2	6dd8	7£23	5bde	129e
> M mySystem_i/JesdSubSys/Transport_layAC38RF82_84111_0_tx_dataout[255:0] W u_ila_0_iobufs_i_tx_sync	e85cf543	9ad9	e284	e85c	e287	<u> </u>	0611	3c2e	ef59	221d	d8ac	2348	de29
W mySystem_i/JesdSubSys/jesd204_tx_tx_tready	1												
	· ·												

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.

v_ila_1													? 0
Waveform - hw_ila_1													? _ 6
Q   +   −   ♂   ▶   ≫   ■   ⊡   Q   Q   ∷   •         ▶	1 1 1 1 1	tr   +Γ	Fe   a	,   ,,,,									
ILA Status: Idle				104									
Name	Value	102	1	104		106		108		110		112	
> M mySystem_i/JesdSubSys/Sample_0[15:0]	1f18	7ffa	6443	1f18	cd12	901b	8167	a70a	f176	41e4	7714	7609	4clf
> M mySystem_i/JesdSubSys/Sample_1[15:0]	0a73	7dee	55£5	0a73	ba9f	877a	8629	b739	0651	52e3	7d22	73a5	3a57
> M mySystem_i/JesdSubSys/Sample_2[15:0]	f587	7884	455c	£587	aa06	8211	8e2e	c959	lbld	61ac	7fd8	6929	2700
> M mySystem_i/JesdSubSys/Sample_3[15:0]	e0e2	6fe2	32e8	e0e2	9bb9	8006	993c	dcef	2£22	6dd8	7f23	5bde	129e
> M mySystem_i/JesdSubSys/Concat_OP[63:0]	e0e2f587	6fe2	32e8	e0e2	9bb9	8006	993c	dcef	2f22	6dd8	7£23	5bde	129e
mySystem_i/JesdSubSys/Transport_layAC38RF82_84111_0_tx_dataout[255:0	e85cf543	9ad9	e284	e85c	e287	Ъ906	0611	3c2e	ef59	221d	d8ac	2348	de29
> Lane 7	e85cf543	9ad9	e284	e85c	e287	<b>b</b> 906	0611	3c2e	ef59	221d	d8ac	2348	de29
Lane 6	3245556	7e79	6£78	3245	e0f5	9baa	8082	998e	dec9	2flb	6d61	7£7£	5b69
> Lane 5	e85cf543	9ad9	e284	e85c	e287	<b>b</b> 906	0611	3c2e	ef59	221d	d8ac	23d8	de29
> Lane 4	3245556	7e79	6£78	3245	e0f5	9baa	8082	998e	dec9	2flb	6d61	7£7£	5b69
> Lane 3	e85cf543	9ad9	e284	e85c	e287	Ъ906	0611	3c2e	ef59	221d	d8ac	2348	de29
> Lane 2	3245556	7e79	6£78	3245	e0f5	9baa	8082	998e	dec9	2flb	6d61	7£7£	Sb69
> Lane 1	e85cf543	9ad9	e284	e85c	e287	<b>b</b> 906	0611	3c2e	ef59	221d	d8ac	2348	de29
> Lane 0	3245556	7e79	6f78	3245	e0f5	9baa	8082	998e	dec9	2f1b	6d61	7£7£	5b69
1 u_ila_0_iobufs_i_tx_sync	1												
14 mySystem_i/JesdSubSys/jesd204_tx_tx_tready	1												

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

Waveform - hw_ila_1													? _ 0
Q   +   −   &   ▶   ≫   ■   ⊡   @   Q   ⊠   +       ↓	12 3	tr +F	Fe   -	l III									
ILA Status: Idle				104									
Name	Value	102		104		106		108	1	110		112	
> 💐 mySystem_i/JesdSubSys/Sample_0[15:0]	7960												
> 📲 mySystem_i/JesdSubSys/Sample_1[15:0]	0a73	7dee	55f5	0a73	ba9f	877a	8629	b739	065f	52e3	7d22	73a5	3a57
> M mySystem_i/JesdSubSys/Sample_2[15:0]	f587	7884	455c	£587	aa06	8211	8e2e	c959	1b1d	61ac	7£48	6929	2700
> M mySystem_i/JesdSubSys/Sample_3[15:0]	e0e2	6fe2	32e8	e0e2	9669	8006	993c	dcef	2f22	6448	7f23	5bde	129e
mySystem_i/JesdSubSys/Concat_OP[63:0]	e0e2f587	61e2	32e8	e0e2	9bb9	8006	993c	dcef	2122	6dd8	7123	5bde	129e
mySystem_i/JesdSubSys/Transport_layAC38RF82_84111_0_tc_dataout(255:0)	e85cf543	9ad9	e284	e85c	e287	Ъ906	0611	3c2e	ef59	221d.,	d8ac	2348	de29_
1 u_ila_0_iobufs_i_tx_sync	1												
1 mySystem_i/JesdSubSys/jesd204_tx_tx_tready	1												
				•									
		Indated	at - 20	117-Dec-	08 04:5	8.09							

The transport layer is implemented only for the mode 84111.

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]		
	Bytes Nibbles Lane RX0 Lane RX1 Lane RX2 Lane RX3 Lane RX4 Lane RX5 Lane RX6	Bytes0Nibbles0Lane RX0A-i0[Lane RX1A-i0]Lane RX2A-q0[Lane RX3A-q0[Lane RX4B-i0[Lane RX5B-i0[Lane RX6B-q0[	Bytes $0$ Nibbles $0$ $1$ Lane RX0A-i0[15:8]Lane RX1A-i0[7:0]Lane RX2A-q0[15:8]Lane RX3A-q0[7:0]Lane RX4B-i0[15:8]Lane RX5B-i0[7:0]Lane RX6B-q0[15:8]	

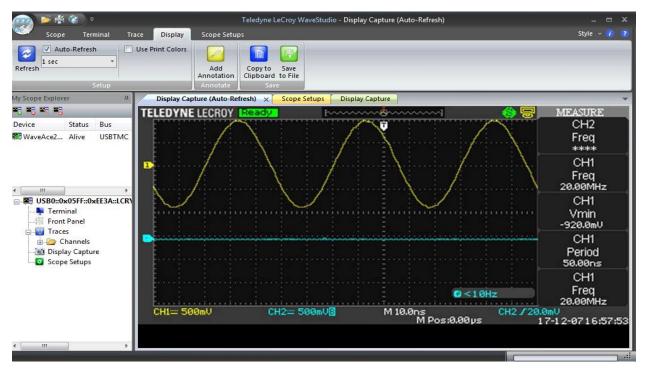
From the DAC datasheet, the sample format can be obtained as follows

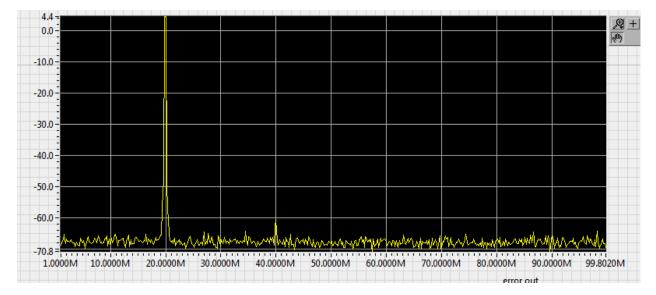
Transport\_layer\_DAC38RF82\_84111\_0\_tx\_dataout follows the following sequence

- 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 1<sup>st</sup> 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

PAGE 14

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.

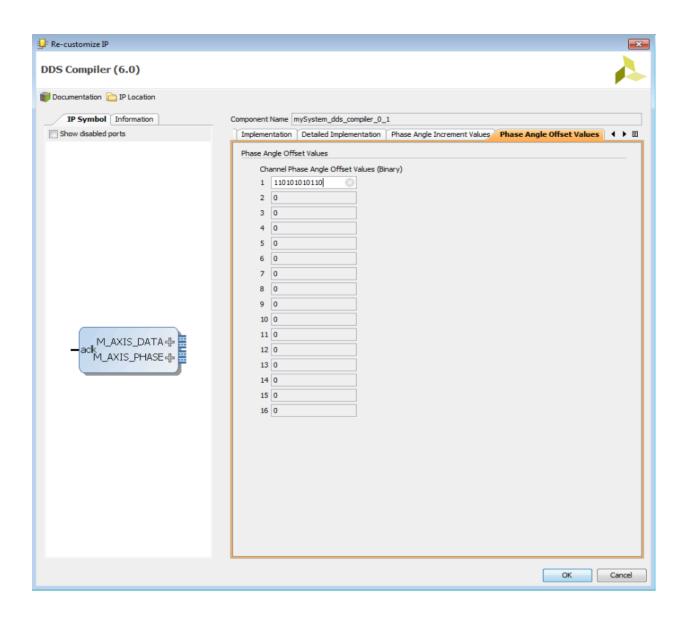
By rounding off to the nearest integer and making it a multiple of phase offset, we get 6828 which is 1AAC in hexadecimal or 1101010101100 in binary.

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)	
W Documentation 📄 IP Location	
IP Symbol Information	Component Name mySystem_dds_compiler_0_3
Show disabled ports	Configuration Implementation Detailed Implementation Phase Angle Increment Values Phase Ani 4 🕨 🗉
	Configuration Options Phase Generator and SIN COS LUT  System Requirements
	System Clock (MHz)         192         [0.01 - 1000.0]           Number of Channels         1         Image: Compared to the second se
	Mode Of Operation Standard -
	Frequency per Channel (Fs) 192.0 MHz
	Parameter Selection Hardware Parameters
	Noise Shaping v
	Hardware Parameters
	Phase Width 16 [3 - 48]
	Output Width 16 🔘 [3 - 26]
aclk M_AXIS_DATA	

🖵 Re-customize IP	
DDS Compiler (6.0)	4
👹 Documentation 🚞 IP Location	
IP Symbol Information	Component Name mySystem_dds_compiler_0_1
Show disabled ports	Configuration Implementation Detailed Implementation Phase Angle Increment Values Phase 4 +
-ack M_AXIS_DATA	Change Increment Values         Phase Angle Increment Values         Channel Phase Angle Increment Values (Binary)         1       110101010100         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