Kintex-7 FPGA DSP Kit Reference Design Tutorial

Version 1.5 Kintex™-7

RTL Design Flow for DSP

Revision History

Date	Version	Revision
1/31/2012	1.0	Initial Release
8/1/2012	1.1	Updated for Vivado 2012.2
12/18/2012	1.2	Updated for Vivado 2012.4 and ILA2
03/27/2013	1.3	Updated for Vivado 2013.1 and VIO2
07/2/2013	1.4	Updated for Vivado 2013.2
11/1/2013	1.5	Updated for Vivado 2013.3

Introduction

This design tutorial will teach users a common technique for verifying a DSP design from within an RTL design environment which is to send an impulse into the design and follow the response as it progresses through the DSP datapath. It is difficult to visually validate real world signal processing response of analog data using an RTL environment – for that there are more suitable environments such as MATLAB and Simulink. An impulse input has a predictable response through a FIR filter that can be easily located from within an RTL simulation.

Once behavioral simulation verifies functionally correct behavior the second part of this tutorial will take users through the processes of recompiling the design sources to regenerate the bitstream used in the Getting Started Design Tutorial. This serves to validate that working tools and IP are installed and working correctly on your system. This design uses the Vivado design flow that is now shipping with the Design Suite. Once the bitstream is downloaded the Vivado Analyzer tool will be used to view a sine wave that is generated on the FPGA, sent through the digital up converter (DUC) to the analog interface where the signal is looped back from the DAC to the ADC and back into the FPGA where it is processed by a digital down converter.

The final part of this tutorial will demonstrate how to make a change to the design.

Requirements

The software and hardware requirements for this kit are described in the following section

Software

The software required to use and run the demonstration is:

- Xilinx Vivado Design Suite 2013.3

Hardware

The hardware required to run the tutorial reference design is:

- Xilinx KC705 Evaluation Board
- 4DSP FMC150 dual-channel A/D and dual-channel D/A FMC daughter card
- 12V AC/DC adapter
- USB to micro-USB cable

Hardware Setup

Perform the following steps to setup the hardware running data acquisition and simulation for Simulink

1. Connect DAC Output C to ADC Input A through an MMCX-to-MMCX cable on the FMC150 card. Make sure you use enough force to make a solid connection. You should hear a click.

2. Connect DAC Output D to ADC Input B through an MMCX-to-MMCX cable on the FMC150 card. Make sure you use enough force to make a solid connection. You should hear a click.

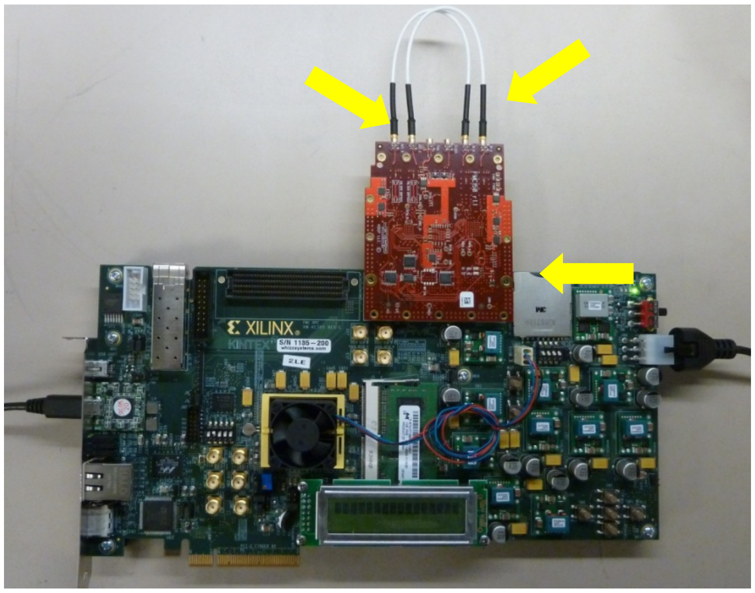


Figure 1 – FMC150 Board Cable Connections

3. Connect the FMC150 to the LPC FMC connector J2 on the KC705.

4. Setup the GPIO DIP switches on SW11 on the KC705 to select a DEMO mode (all OFF).

5. Connect the micro USB cable to the JTAG port,

6. Attach the 12V wall power supply to J49.

7. Power up the board by turning SW15 ON. You should see two blue lights on the FMC150 card that indicate the board and interface are working properly

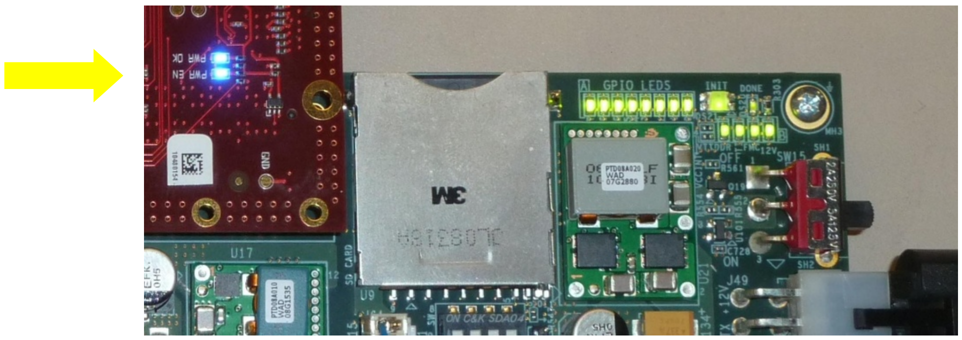


Figure 2 – FMC150 Board LEDs

Downloading and Installing the DSP Kit RTL Tutorial Design

1. Go to the Avnet Kintex-7 FPGA DSP Kit web page: http://www.em.avnet.com/k7dspkit.

2. At the bottom of the page, click “Support files and Downloads”.

3. To download the RTL tutorial design files, click on “Kintex-7 DSP Kit RTL Reference Design Tutorial”. This will download the file “k7dsp_rtl _tutorial_2013.3.zip”.

4. Unzip to your computer’s C: drive using the “Extract to Here” option. The top-level folder name should be “k7dsp_rtl_tut”. If this is not how it was extracted, then change the name as the design contains absolute path names that require this file structure.

Tutorial Lab 1 – Verifying the RTL DUC/DDC to Golden MATLAB Vectors

In this first tutorial you will observe the construction of the design using CoreGen IP blocks including the FIR Compiler. Also as a first step to verification you will simulate the design with an impulse stimulus and compare the resulting impulse response to golden vectors created from MATLAB. Because the impulse input is well defined and repeatable, it allows for an accurate comparison to golden vectors generated from a high level language such as MATLAB.

Verifying the Impulse Response

In this exercise we will drive an impulse into the system, observe the impulse response at various nodes in the DUC / DDC design and compare the values to expected results from our MATLAB reference model. The impulse input will be of magnitude 0.5 and last for exactly one period at the input sample rate, or 96 system-clock cycles. To achieve this in hardware an impulse generator has been incorporated into the code triggered from a pushbutton on the hardware board. This allows for a predictable response from the system that can be captured and compared to the expected MATLAB reference model.

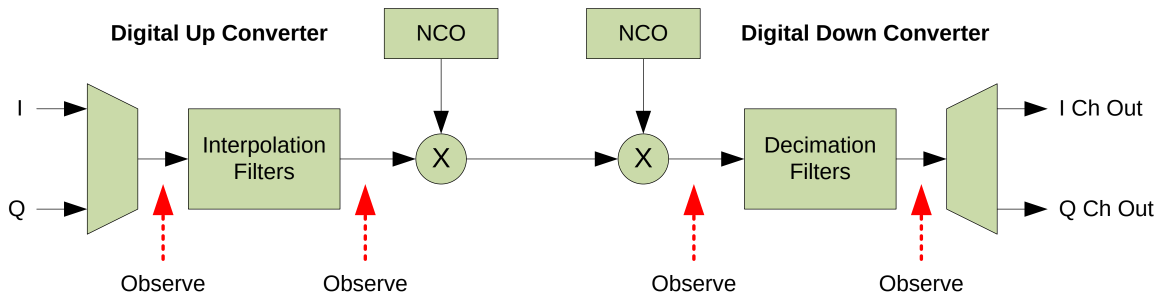
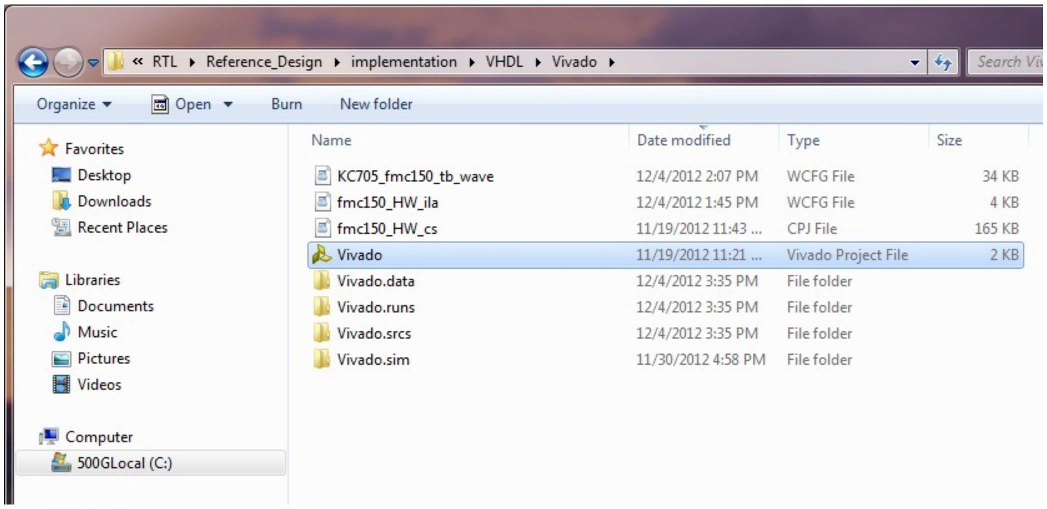


Figure 3 – RTL Simulation Block Diagram

Performing Simulink Simulation with Analog Data Acquisition

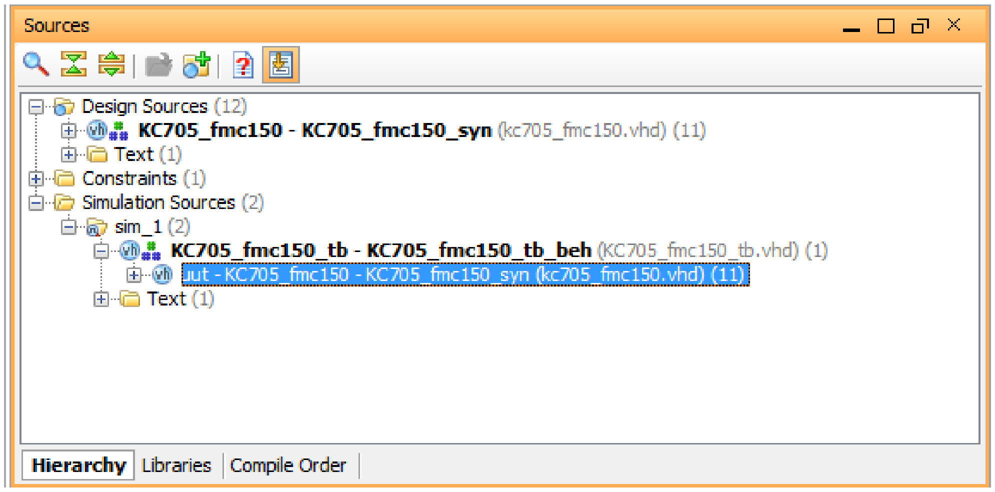
1. Navigate to the Vivado directory and double click on the project file kc705_fmc150.xpr located at c:\k7dsp_rtl_tut\RTL\RefDes\imp\VHDL\Vivado



Name	Date modified	Type	Size
KC705_fmc150_tb_wave	12/4/2012 2:07 PM	WCFG File	34 KB
fmc150_HW_ila	12/4/2012 1:45 PM	WCFG File	4 KB
fmc150_HW_cs	11/19/2012 11:43 ...	CPJ File	165 KB
Vivado	11/19/2012 11:21 ...	Vivado Project File	2 KB
Vivado.data	12/4/2012 3:35 PM	File folder	
Vivado.runs	12/4/2012 3:35 PM	File folder	
Vivado.srcs	12/4/2012 3:35 PM	File folder	
Vivado.sim	11/30/2012 4:58 PM	File folder	

Figure 4 – Vivado Project File

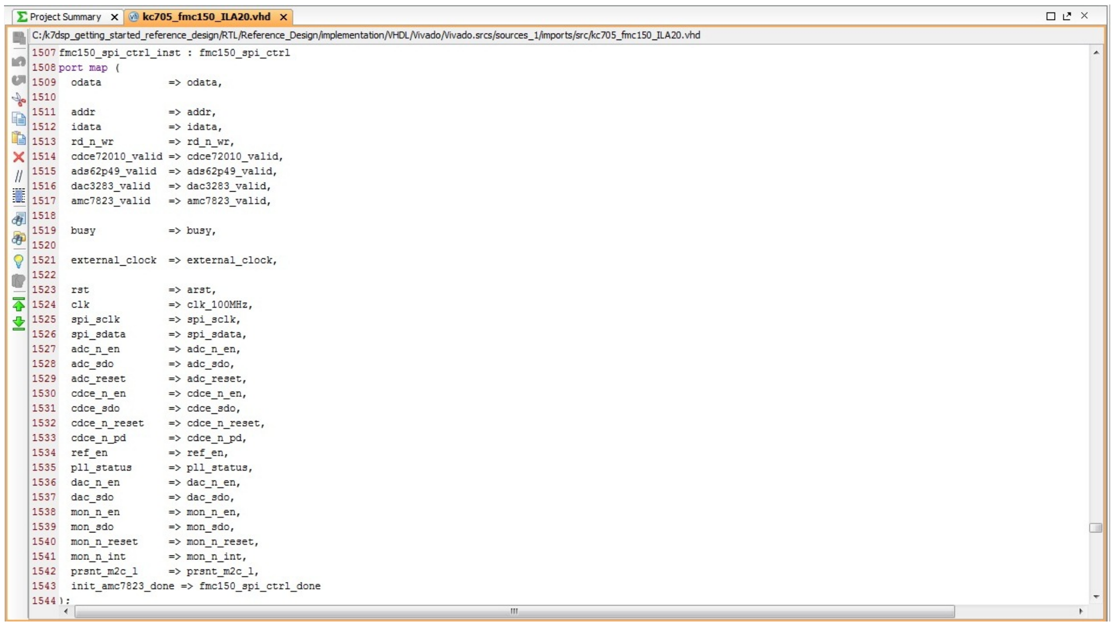
2. From the “Sources Window” double-click on the “uut-kc705_fmc_150” file to bring up the source code



```
Design Sources (12)
  KC705_fmc150 - KC705_fmc150_syn (kc705_fmc150.vhd) (11)
  Text (1)
Constraints (1)
Simulation Sources (2)
  sim_1 (2)
    KC705_fmc150_tb - KC705_fmc150_tb_beh (KC705_fmc150_tb.vhd) (1)
      uut - KC705_fmc150 - KC705_fmc150_syn (kc705_fmc150.vhd) (11)
    Text (1)
```

Figure 5 – Vivado Sources Window

3. Scroll down to lines 1653-1690 and observe the code. This code configures the devices on the fmc150 card for proper operation. You'll see later that this allows you to configure the hardware using Vivado VIO to tune the settings before you build the design with the new values.



C:/k7dsp_getting_started_reference_design/RTL/Reference_Design/implementation/VHDL/Vivado/Vivado.srcs/sources_1/imports/src/kc705_fmc150_ILA20.vhd

```
fmc150_spi_ctrl_inst : fmc150_spi_ctrl
port map (
  odata            => odata,

  addr             => addr,
  idata            => idata,
  rd_n_wr          => rd_n_wr,
  cdce72010_valid  => cdce72010_valid,
  ads62p49_valid   => ads62p49_valid,
  dac3283_valid    => dac3283_valid,
  amc7823_valid    => amc7823_valid,

  busy             => busy,

  external_clock   => external_clock,

  rst              => arst,
  clk              => clk_100MHz,
  spi_sclk         => spi_sclk,
  spi_sdata        => spi_sdata,
  adc_n_en         => adc_n_en,
  adc_sdo          => adc_sdo,
  adc_reset        => adc_reset,
  cdce_n_en        => cdce_n_en,
  cdce_sdo         => cdce_sdo,
  cdce_n_reset     => cdce_n_reset,
  cdce_n_pd        => cdce_n_pd,
  ref_en           => ref_en,
  pll_status       => pll_status,
  dac_n_en         => dac_n_en,
  dac_sdo          => dac_sdo,
  mon_n_en         => mon_n_en,
  mon_sdo          => mon_sdo,
  mon_n_reset      => mon_n_reset,
  mon_n_int        => mon_n_int,
  prsnt_m2c_l      => prsnt_m2c_l,
  init_amc7823_done => fmc150_spi_ctrl_done
);
```

Figure 6 – Top-Level VHDL File

4. From the Project Manager select the "Run Simulation" This will launch a self running behavioral simulation using Vivado Simulator. Allow this to run for a few minutes to complete

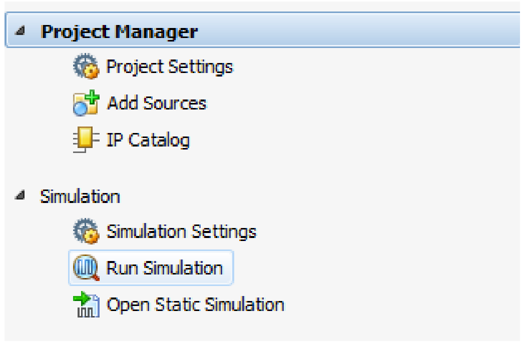
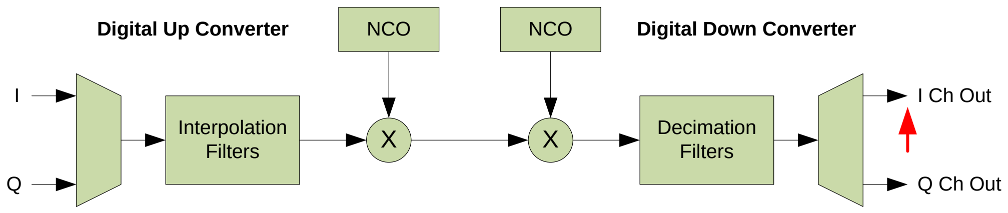


Figure 7 – Vivado Project Manager

5. When the simulation completes use the waveform viewer left hand toolbar to execute the command "Zoom Fit" and use the scroll bar to navigate to the bottom of the waveform viewer. You will see several of the waveforms displayed as an analog impulse signal rather than their digital values. These are the signals that we will use to follow our impulse from the outputs back to the inputs. There is a yellow vertical line at 40.000000 us – this is a curser that can be moved over the peak values of the impulse waveforms.

   We are going to start by observing the impulse at the outputs of the DUC/DDC design. We will just focus on the "I" channel for the purposes of this lab.

6. Scroll down to the bottom of the waveform viewer window using the slider bar on the right of the window. You will see that each group of signals has a heading. The bottom group is called "DUC_DDC_output". Move the yellow curser Observe the peak value of 15343 at 16,649,907ns on the signal "DUC_DDC_BASEBAND_OUT_I".



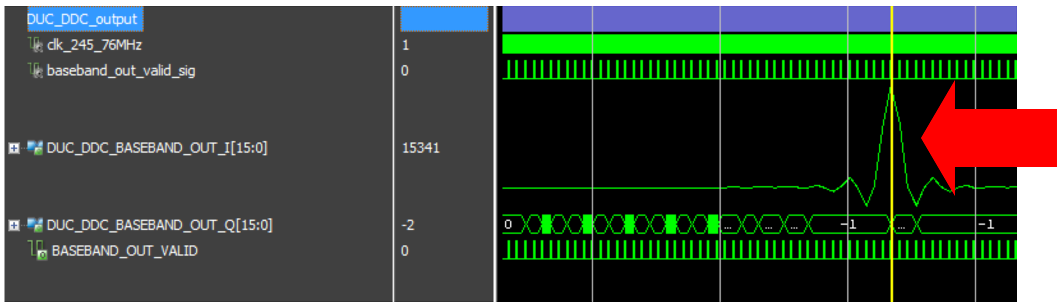
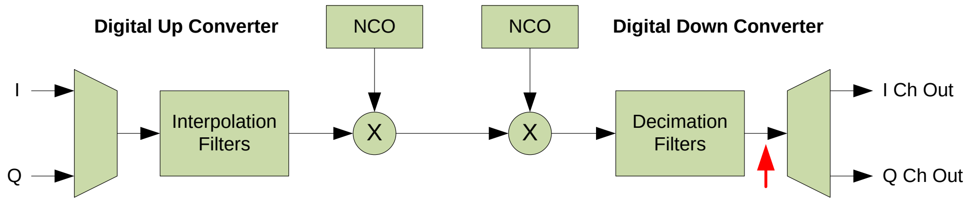


Figure 8 – Vivado Simulator Waveform Window

We are now going to repeat this process for each signal grouping and in doing so we will be following the impulse back through the hardware. For the rest of this lab we will be following the "I" component of the complex signal.

Hint: *If you click the mouse just below the peak of the impulse waveform the curser will automatically move to the peak value of the impulse!*

7. Return to the waveform viewer window toolbar and then scroll up to the signal group called "DDC Ouputs 491.52 MHz Clock Domain" and observe the peak value of 15343 at 16,641,771ns on the bus "ddc_dout_i_r". This is the same value as the previous value because the signal simply passed through a demux block.



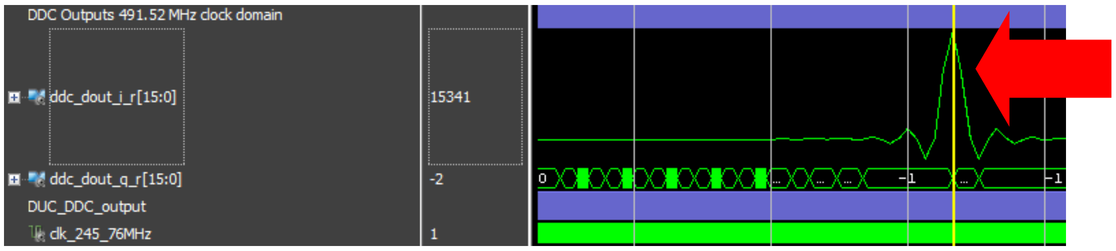
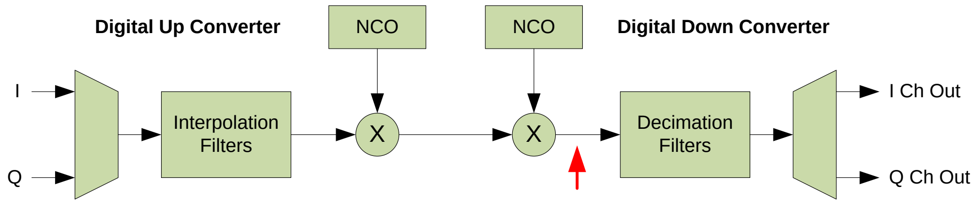


Figure 9 – Vivado Simulator Waveform Window

8. Scroll up to the the "DDC Complex Mixer" signal group and locate the peak value of 34737 on the "dout_i" bus as shown below at time 15,616.635ns



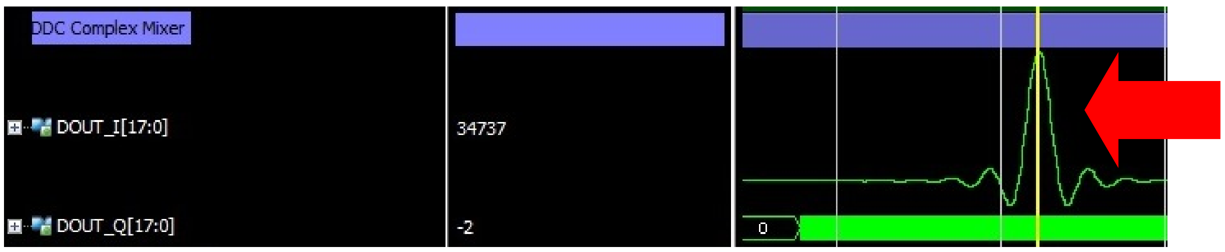
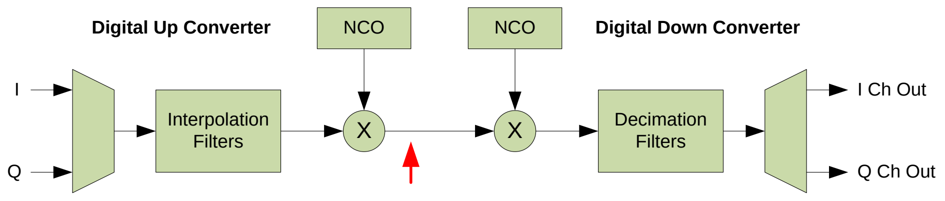


Figure 10 – Vivado Simulator Waveform Window

9. Scroll up to the signal grouping "DUC Ports" and locate the peak value of -8574 at time 15,592.227ns on the bus "dout_i". Note the negative value is due to the phase offset of the DUC complex mixer.



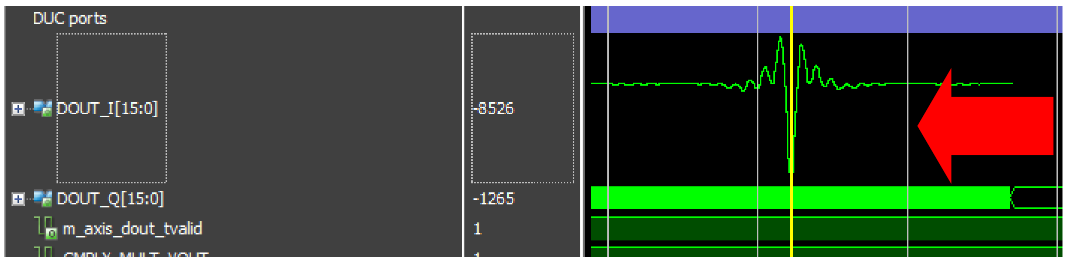
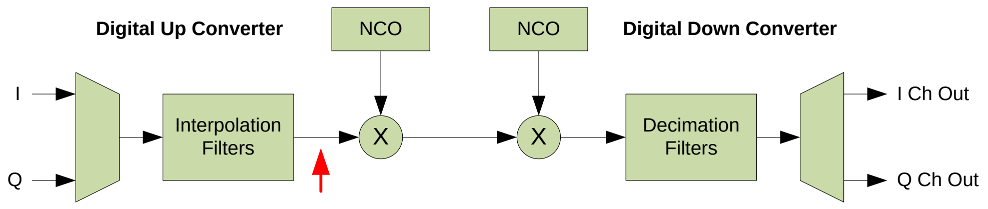


Figure 11 – Vivado Simulator Waveform Window

10. Scroll up to the signal grouping "DUC Complex multiplier" and locate the peak value of 34736 at 15,555.615ns on the bus "din_re"



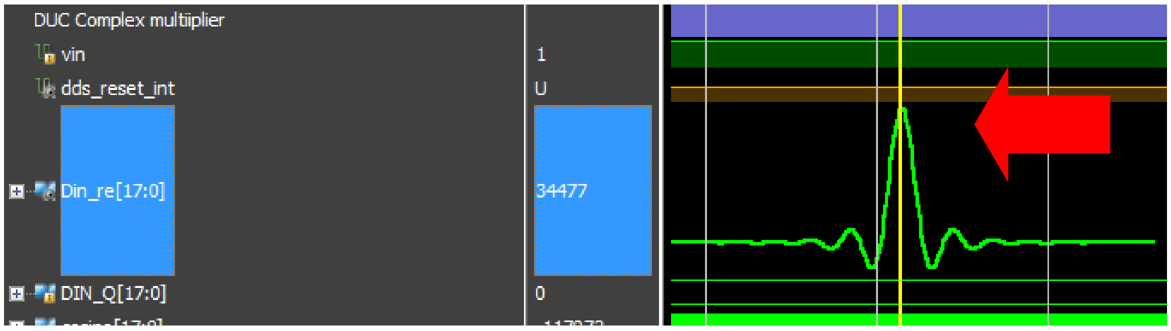


Figure 12 – Vivado Simulator Waveform Window

We've traced the impulse back through the DSP datapath to the inputs of the DUC filters. If you want you can continue to trace the impulse through the filters in the DUC but is not necessary for this tutorial.

11. Once done close the waveform viewer.

Congratulations!

You have followed the impulse response through the system and verified the functional correctness at several stages in the design using the Xilinx ISim simulator. You are now ready to verify your design running on the actual FPGA using Vivado.

Tutorial Lab 2 – Verifying the RTL DUC / DDC in Hardware with the Data Converters Using Vivado Analyzer

In this tutorial we will use 4DSP's FMC150 – Dual DAC / Dual ADC. The RTL DUC drives the Dual DAC. The signals are externally looped back into the Dual ADC which connects to the RTL DDC. The output of the DDC is analyzed with Vivado.

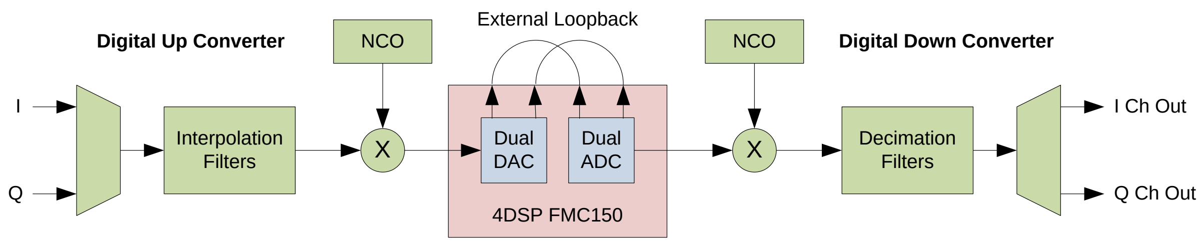


Figure 13 – Block Diagram showing FMC150

Procedure

1. Set the DIP switches on the KC705 to select Analog DEMO mode all OFF (down). This will place the data converters into the DUC to DDC loopback.

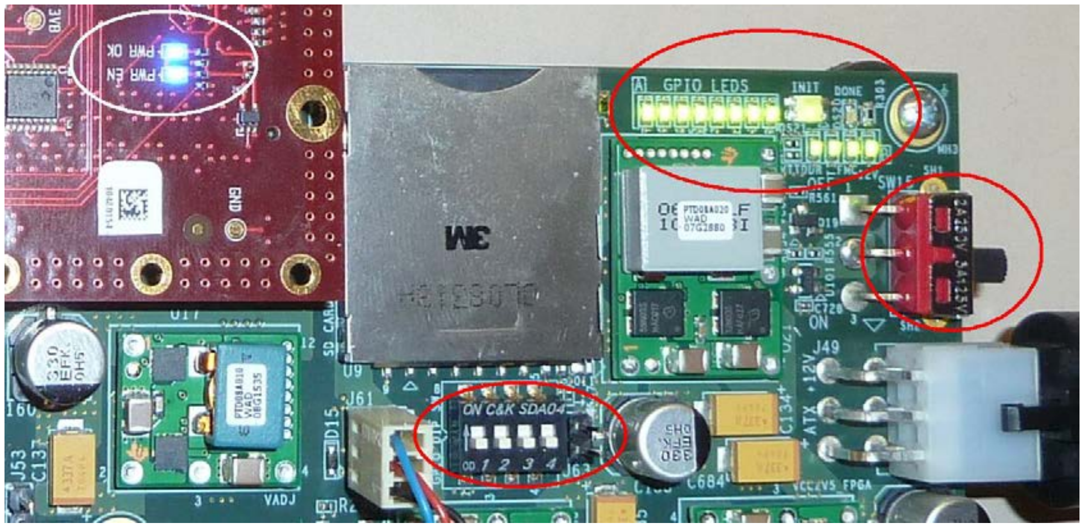
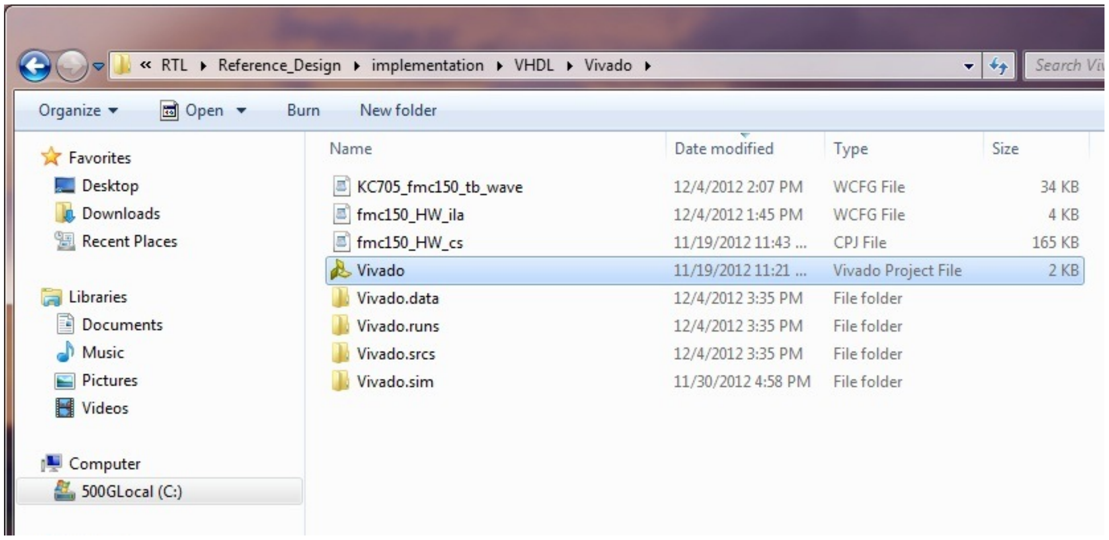


Figure 14 – Board Power Switch, DIP Switch and LEDs

2. Connect a micro USB cable between your PC and the digilent connector of the KC705.

3. Turn on power to the KC705 by turning SW15 ON (Down). The Green GPIO LEDs and power LEDs should all light. The 2 blue LEDs on the FMC module should both be on to show PWR EN and PWR OK.

4. If Vivado is not already open from the first part of the tutorial lab then navigate to the Vivado directory and double click on the project file kc705_fmc150.xpr located at c:\k7dsp_rtl_tut\RTL\RefDes\imp\VHDL\Vivado



Name	Date modified	Type	Size
KC705_fmc150_tb_wave	12/4/2012 2:07 PM	WCFG File	34 KB
fmc150_HW_ila	12/4/2012 1:45 PM	WCFG File	4 KB
fmc150_HW_cs	11/19/2012 11:43 ...	CPJ File	165 KB
Vivado	11/19/2012 11:21 ...	Vivado Project File	2 KB
Vivado.data	12/4/2012 3:35 PM	File folder	
Vivado.runs	12/4/2012 3:35 PM	File folder	
Vivado.srcs	12/4/2012 3:35 PM	File folder	
Vivado.sim	11/30/2012 4:58 PM	File folder	

Figure 15 – Vivado Project File

5. After Vivado is opened select the “Generate Bitstream” step which will launch synthesis, implementation and generate bitstream.

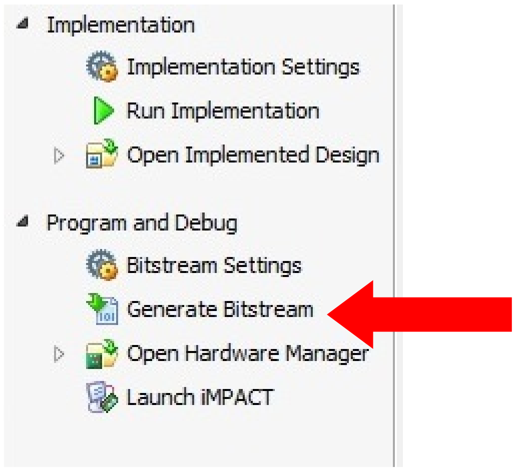


Figure 16 – Vivado Generate Bitstream

6. After the bitstream has been generated, select “Open Hardware Manager” from the flow navigator.

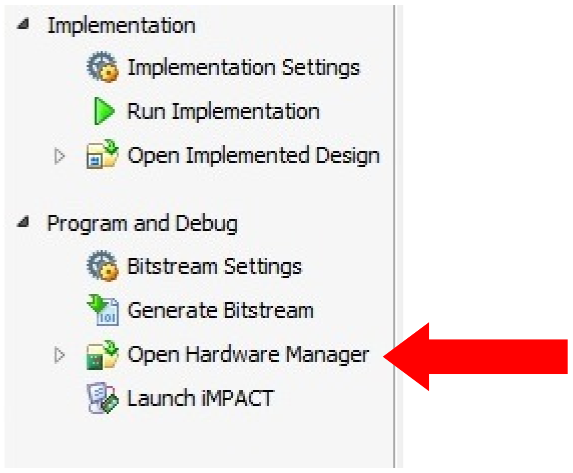


Figure 17 – Open Hardware Manager

7. From the Hardware Manager window, select “Open New Hardware Target.” Click “Next” on the four next windows that open and Finish on the fifth window that opens. This will open the new hardware session on port 60001 on the XC7K325T target.

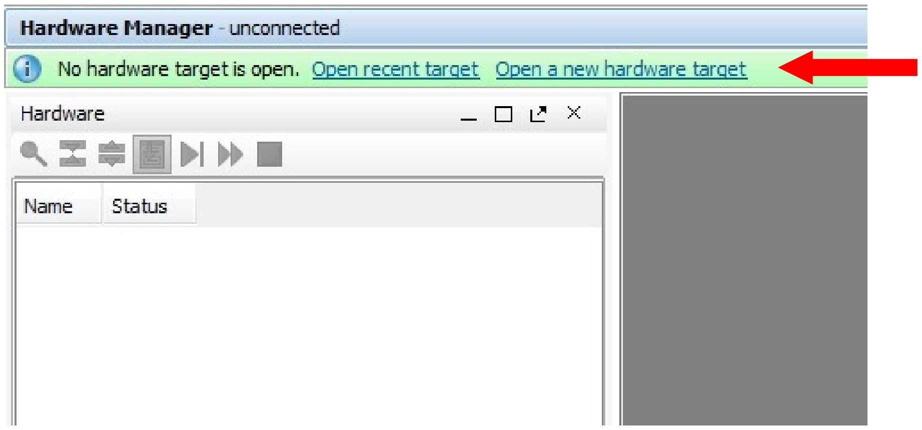
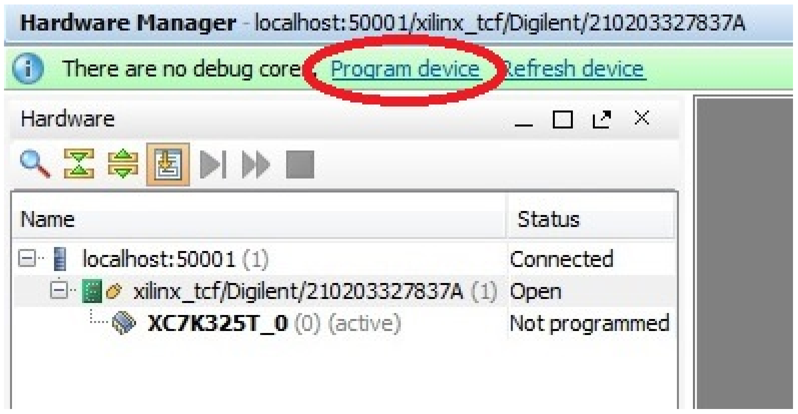

Hardware Manager - unconnected
No hardware target is open. Open recent target Open a new hardware target
Hardware
Name
Status

Figure 18 – Open New Hardware Target

8. In the Hardware Manager window click the “Program Device” and select XC7K325T_0 in the pop up window that appears. A window will appear and select “Program” to download the default bitstream to the FPGA device. This will take a few seconds to complete and a status window will appear to allow you to follow the progress


Hardware Manager - localhost:50001/xilinx_tcf/Digilent/210203327837A

There are no debug core Program device Refresh device

Name	Status
localhost:50001 (1)	Connected
xilinx_tcf/Digilent/210203327837A (1)	Open
XC7K325T_0 (0) (active)	Not programmed

Figure 19 – Program the XC7K325T device

9. When the FPGA configuration is complete the DONE LED should be ON. The GPIO LED 4,5,6,7 are on indicating respectively the PLL lock on the FMC150 and two MMCMs locked in the FGPA and ADC test sequence good.

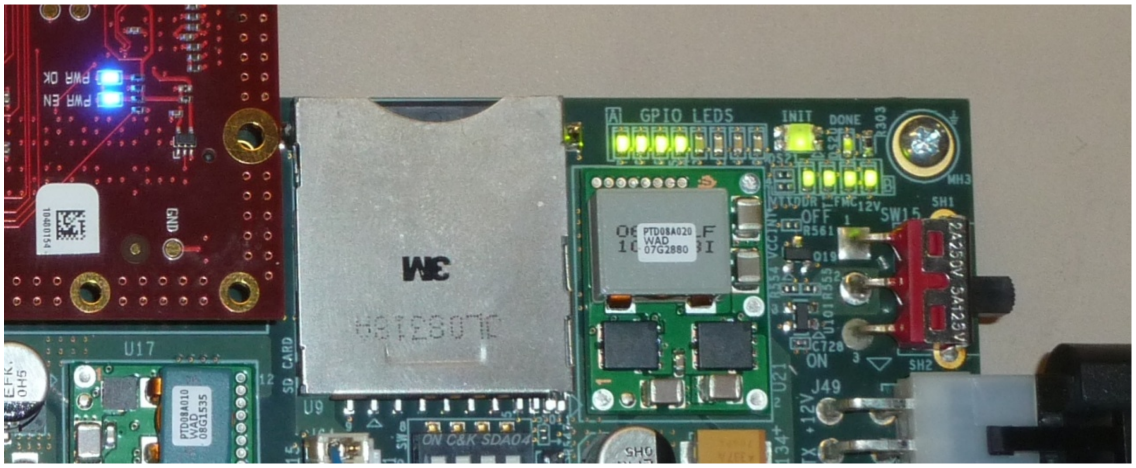
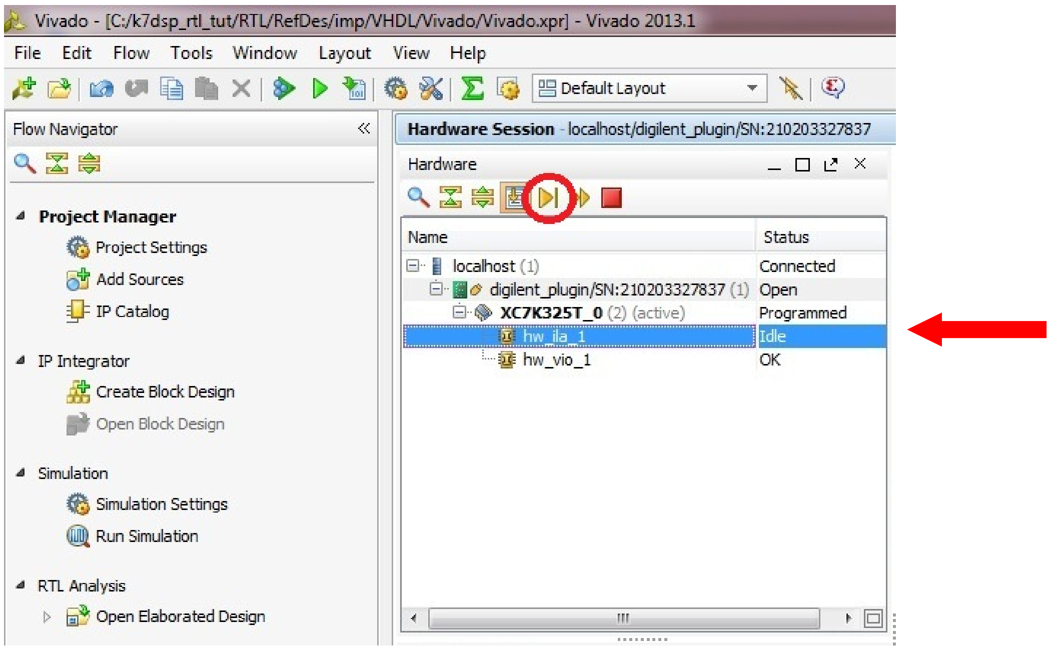


Figure 20 – LEDs that indicate PLL Working Correctly

10. Once the Hardware tile is updated with the hw_ila_1 and the signal list, highlight the hw_ila_1 and then select the Run Trigger icon which is the 5th icon. This will read the probe data from the hardware and display a waveform.



Vivado - [C:/k7dsp_rtl_tut/RTL/RefDes/imp/VHDL/Vivado/Vivado.xpr] - Vivado 2013.1

Hardware Session - localhost/digilent_plugin/SN:210203327837

Name	Status
localhost (1)	Connected
digilent_plugin/SN:210203327837 (1)	Open
XC7K325T_0 (2) (active)	Programmed
hw_ila_1	Idle
hw_vio_1	OK

Figure 21 – Run Trigger in the ILA

11. From the Vivado File tab, select the “Open Waveform Configuration” and highlight the fmc150_HW_ila.wcfg waveform configuration file and select OK. This will apply the wave form configuration settings to view the I and Q data bus as an analog sine wave.

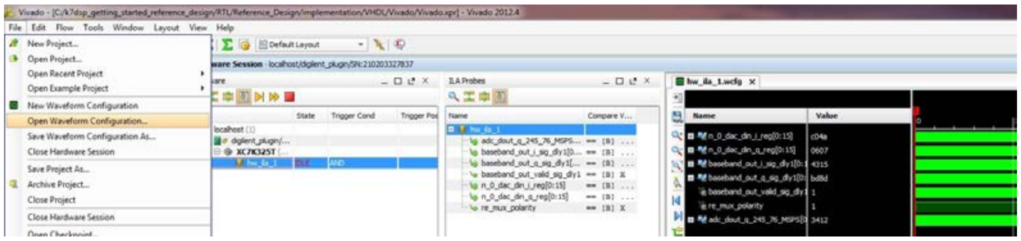


Figure 22 – Open Waveform Configuration File

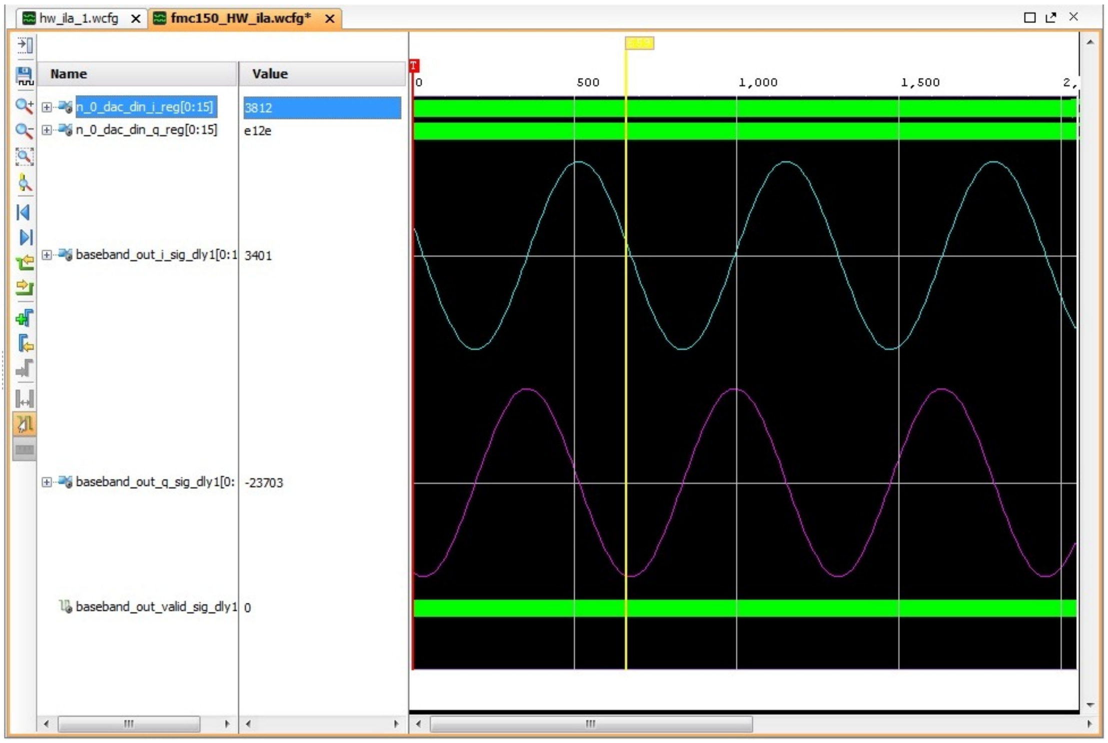
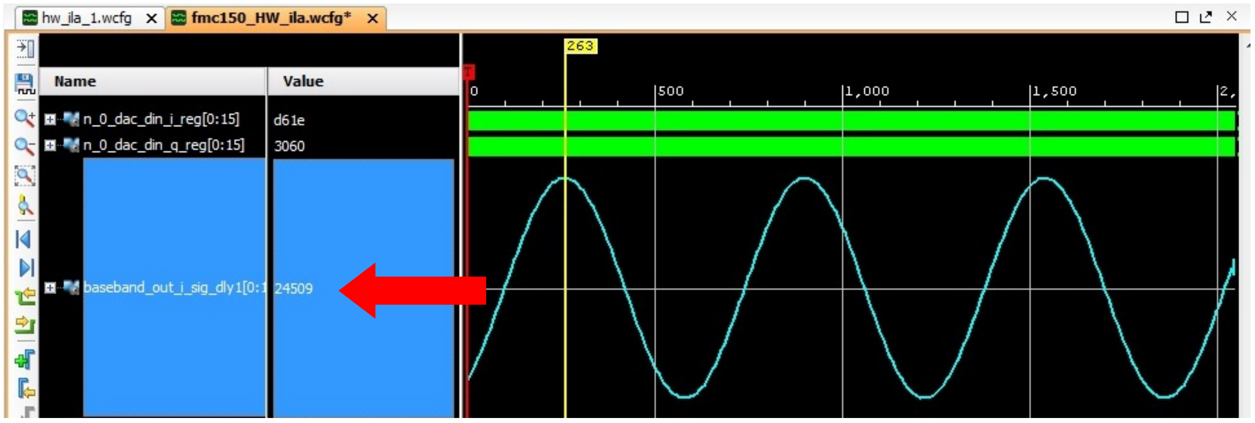


Figure 23 – Correct ILA Waveform

You have now run the Kintex-7 FPGA DSP Development Kit with High-Speed Analog reference design through the FMC150 daughter card and observed data capture in Vivado ILA. Using this fully functional DSP design, you may now develop your DSP application on Xilinx Kintex-7 FPGAs.

12. Select the first Sine wave signal and click on a high point of the wave and note the amplitude value of the captured sine wave. Note that the amplitude is +/-24K. In the next steps of this tutorial we shall change the DAC gain using Vivadoo VIO.

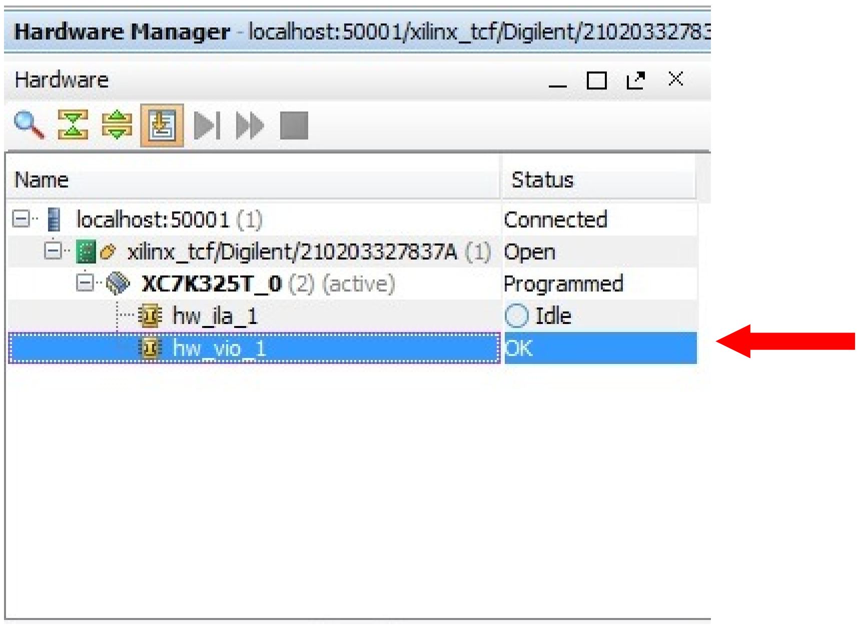


Name	Value
n_0_dac_din_i_reg[0:15]	d61e
n_0_dac_din_q_reg[0:15]	3060
baseband_out_i_sig_dly1[0:1	24509

Figure 24 – Sine Wave Amplitude

13. In the Vivado Hardware Manager window, select the hw_vio_1.

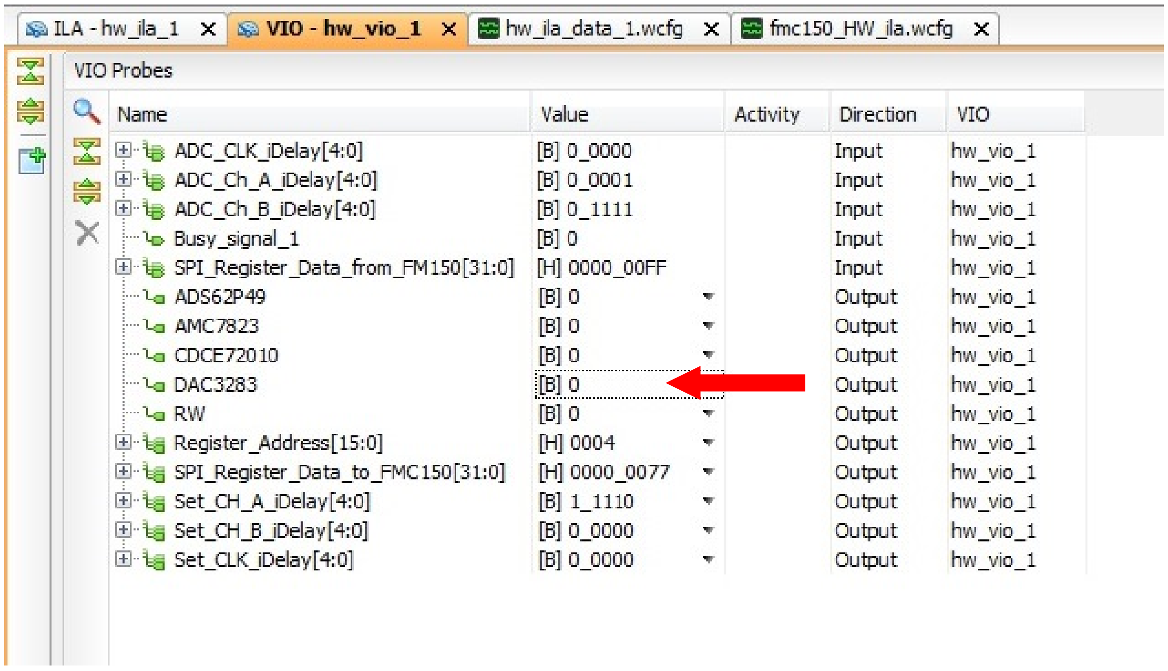
.


Hardware Manager - localhost:50001/xilinx_tcf/Digilent/21020332783
Hardware

Name	Status
localhost:50001 (1)	Connected
xilinx_tcf/Digilent/210203327837A (1)	Open
XC7K325T_0 (2) (active)	Programmed
hw_ila_1	Idle
hw_vio_1	OK

Figure 25 – Selecting the VIO Core

In the VIO Probes window right click to select the DAC3283 Value field and select Toggle mode. In the pop-up box.



Name	Value	Activity	Direction	VIO
ADC_CLK_iDelay[4:0]	[B] 0_0000		Input	hw_vio_1
ADC_Ch_A_iDelay[4:0]	[B] 0_0001		Input	hw_vio_1
ADC_Ch_B_iDelay[4:0]	[B] 0_1111		Input	hw_vio_1
Busy_signal_1	[B] 0		Input	hw_vio_1
SPI_Register_Data_from_FM150[31:0]	[H] 0000_00FF		Input	hw_vio_1
ADS62P49	[B] 0		Output	hw_vio_1
AMC7823	[B] 0		Output	hw_vio_1
CDCE72010	[B] 0		Output	hw_vio_1
DAC3283	[B] 0		Output	hw_vio_1
RW	[B] 0		Output	hw_vio_1
Register_Address[15:0]	[H] 0004		Output	hw_vio_1
SPI_Register_Data_to_FMC150[31:0]	[H] 0000_0077		Output	hw_vio_1
Set_CH_A_iDelay[4:0]	[B] 1_1110		Output	hw_vio_1
Set_CH_B_iDelay[4:0]	[B] 0_0000		Output	hw_vio_1
Set_CLK_iDelay[4:0]	[B] 0_0000		Output	hw_vio_1

Figure 26 –Change the DAC3283 Value to Toggle Button

Note: There may be a slight amplitude mismatch between the 2 channels of 384 kHz sinusoids sampled through the ADC. This is due to variations in the analog signal chain, primarily the frequency response of the RF transformers at the DAC outputs and ADC inputs, which have a low-frequency roll-off below approximately 5 MHz.

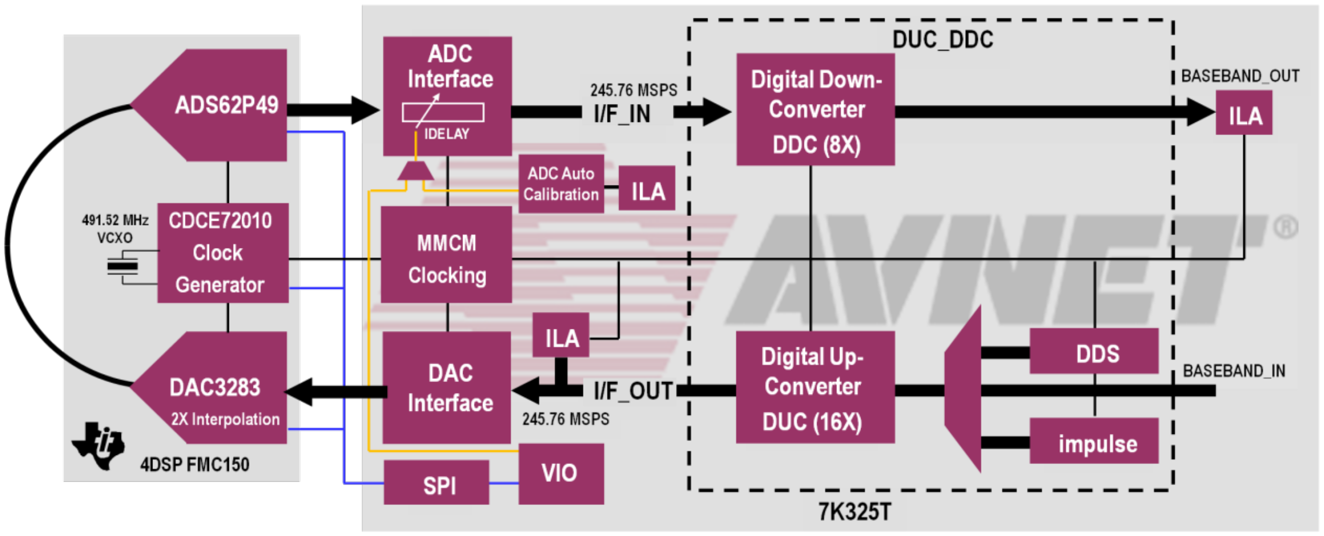


Figure 27 – Block Diagram Showing SPI Control Driven from VIO

14. In order to change the DAC gain we need to write a value to register CONFIG4 in the DAC3283 device. The following steps in the VIO console are required:

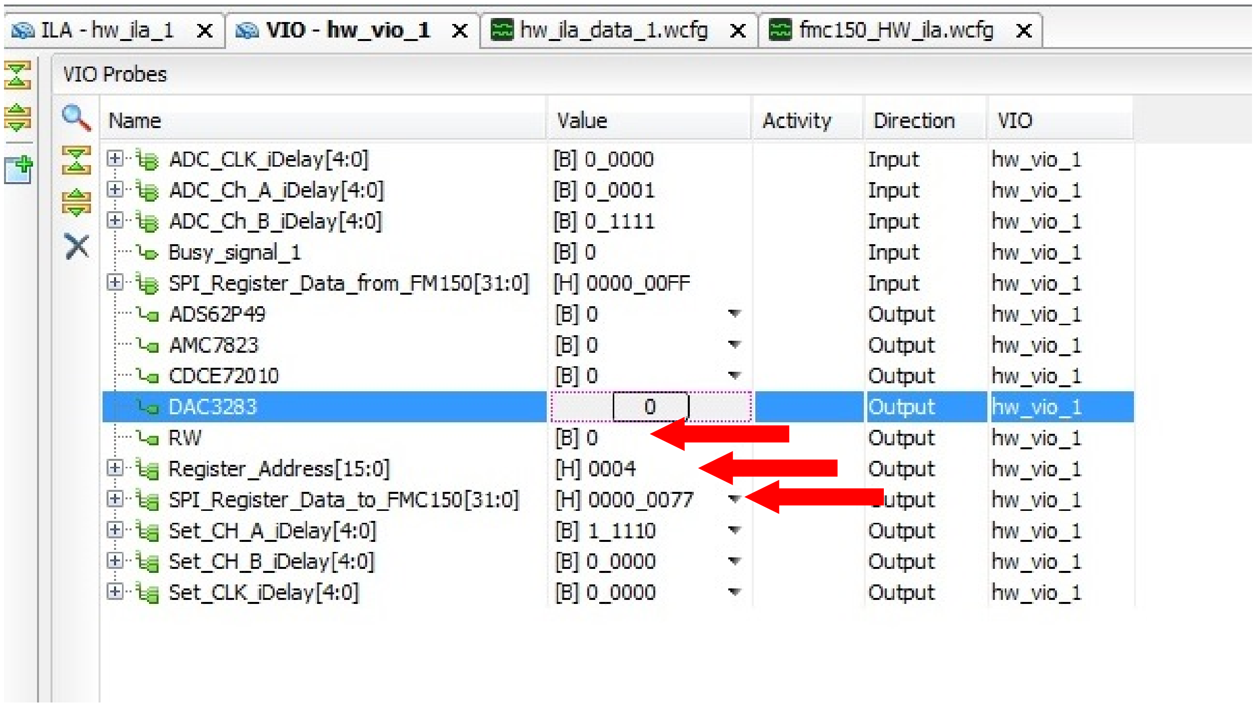
    a. Verify RW is set to 0

    b. Verify Register Address is set to 0004

    c. Verify Register Data In is set to 00000077, this results in half scale on both DAC channels

    d. Click in the toggle box to set DAC3283 to 1

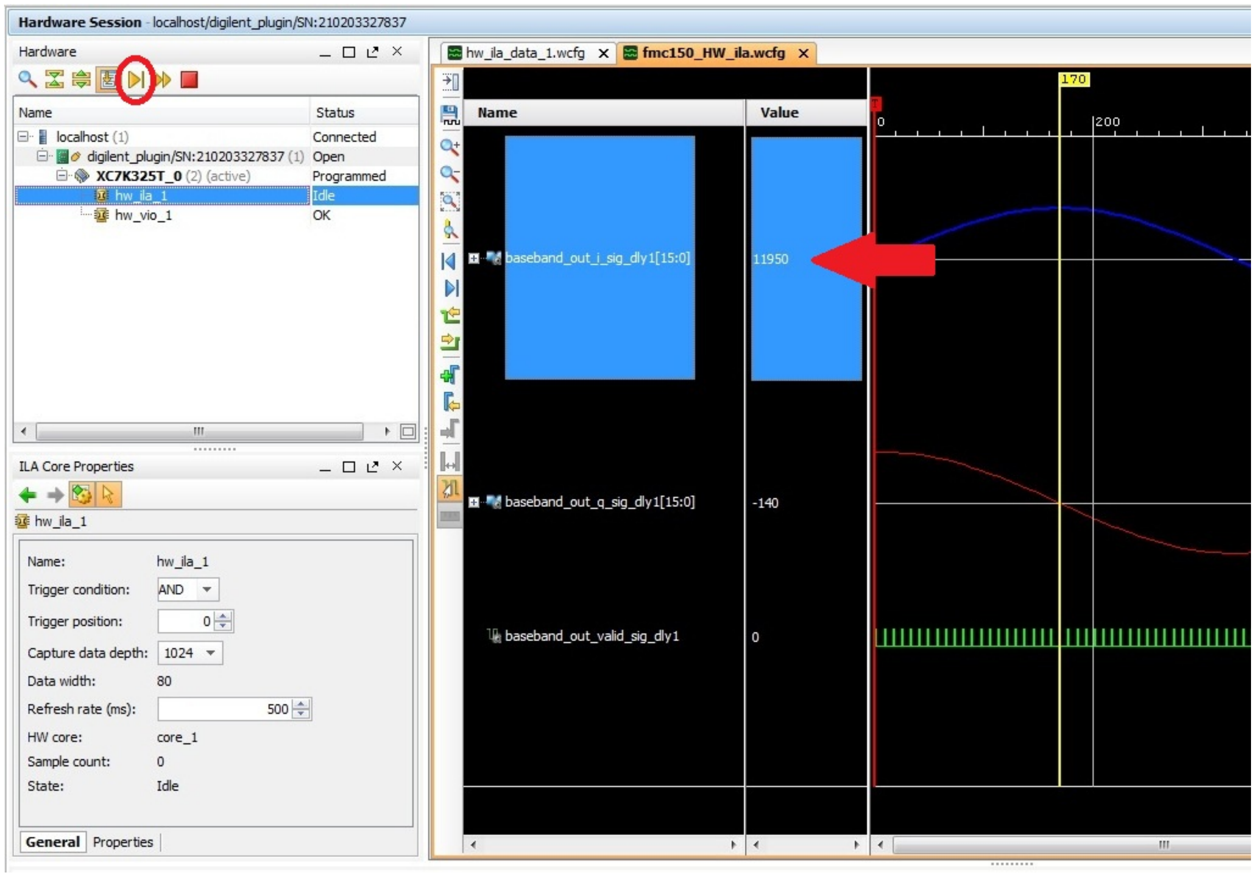
    e. Click in the toggle box to set DAC3283 back to 0



Name	Value	Activity	Direction	VIO
ADC_CLK_iDelay[4:0]	[B] 0_0000		Input	hw_vio_1
ADC_Ch_A_iDelay[4:0]	[B] 0_0001		Input	hw_vio_1
ADC_Ch_B_iDelay[4:0]	[B] 0_1111		Input	hw_vio_1
Busy_signal_1	[B] 0		Input	hw_vio_1
SPI_Register_Data_from_FM150[31:0]	[H] 0000_00FF		Input	hw_vio_1
ADS62P49	[B] 0		Output	hw_vio_1
AMC7823	[B] 0		Output	hw_vio_1
CDCE72010	[B] 0		Output	hw_vio_1
DAC3283	0		Output	hw_vio_1
RW	[B] 0		Output	hw_vio_1
Register_Address[15:0]	[H] 0004		Output	hw_vio_1
SPI_Register_Data_to_FMC150[31:0]	[H] 0000_0077		Output	hw_vio_1
Set_CH_A_iDelay[4:0]	[B] 1_1110		Output	hw_vio_1
Set_CH_B_iDelay[4:0]	[B] 0_0000		Output	hw_vio_1
Set_CLK_iDelay[4:0]	[B] 0_0000		Output	hw_vio_1

Figure 28 –Vivado VIO Dashboard

15. Go back to the Vivado Analyzer retrigger and select the high point of the captured Sine wave of the DDC output. The amplitude on the DDC output is now +/-12.0K



Hardware Session - localhost/digilent_plugin/SN:210203327837

Name	Status
localhost (1)	Connected
digilent_plugin/SN:210203327837 (1)	Open
XC7K325T_0 (2) (active)	Programmed
hw_ila_1	Idle
hw_vio_1	OK

ILA Core Properties — hw_ila_1

Name:	hw_ila_1
Trigger condition:	AND
Trigger position:	0
Capture data depth:	1024
Data width:	80
Refresh rate (ms):	500
HW core:	core_1
Sample count:	0
State:	Idle

hw_ila_data_1.wcfg — fmc150_HW_ila.wcfg

Name	Value
baseband_out_i_sig_dly1[15:0]	11950
baseband_out_q_sig_dly1[15:0]	-140
baseband_out_valid_sig_dly1	0

Figure 29 – Sine Wave Amplitude with reduced Gain

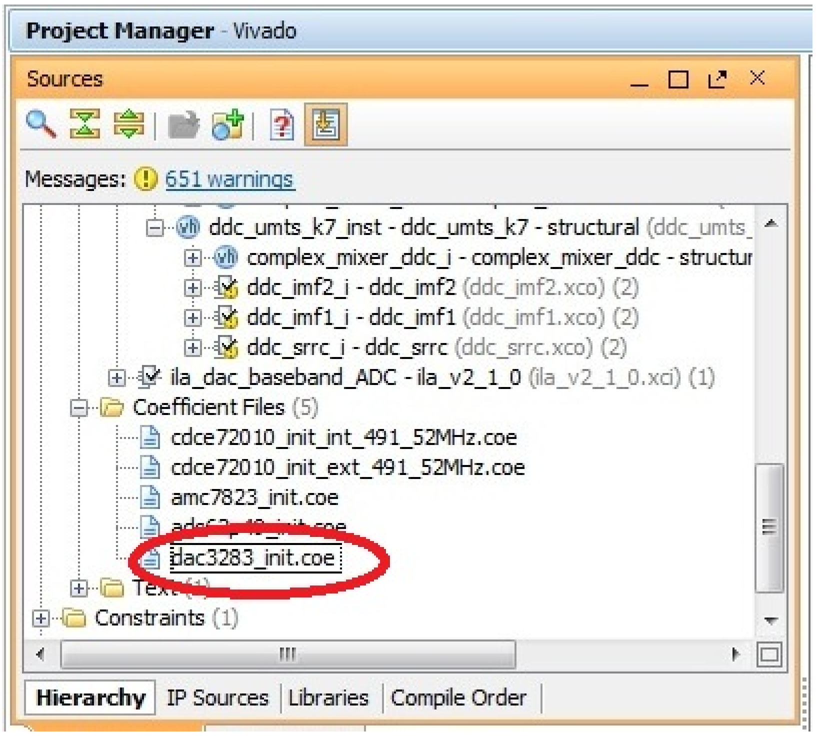
Congratulations!

You have executed the targeted reference design on the hardware using Vivado and using Vivado to modify a register setting on the DAC using the Vivado Virtual IO feature.

Tutorial Lab 3 – Changing the FMC150 Power-Up Reset Configuration

In tutorial Lab 2 we saw how we can quickly change device register settings on the FMC150 using Vivado VIO. This tutorial shows how to change the RTL default power up settings once you have found the proper register settings for your application. As an example the DAC gain is changed.

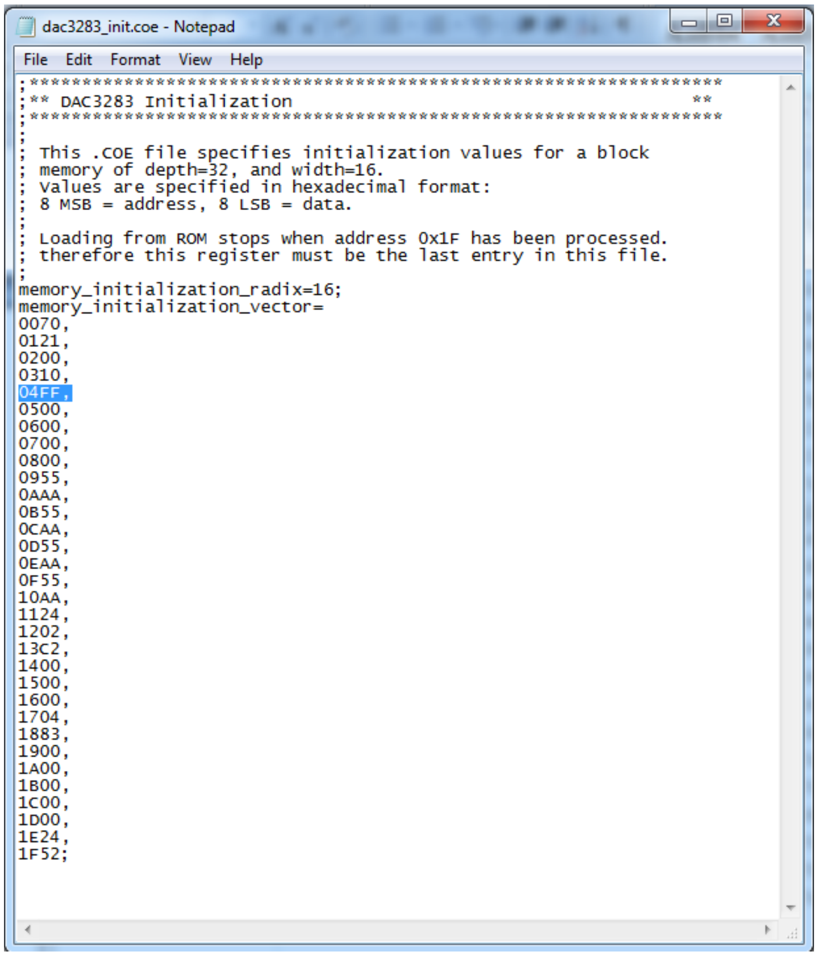
1. In the Progect Manager window in Vivado expaand the Coefficen source t files and select the “dac3283_init.coe” .



```
ddc_umts_k7_inst - ddc_umts_k7 - structural (ddc_umts_
  complex_mixer_ddc_i - complex_mixer_ddc - structur
  ddc_imf2_i - ddc_imf2 (ddc_imf2.xco) (2)
  ddc_imf1_i - ddc_imf1 (ddc_imf1.xco) (2)
  ddc_srrc_i - ddc_srrc (ddc_srrc.xco) (2)
ila_dac_baseband_ADC - ila_v2_1_0 (ila_v2_1_0.xci) (1)
Coefficient Files (5)
  cdce72010_init_int_491_52MHz.coe
  cdce72010_init_ext_491_52MHz.coe
  amc7823_init.coe
  ads62p49_init.coe
  dac3283_init.coe
Text (1)
Constraints (1)
```

Figure 30 – Xilinx .coe File for programming data converter devices

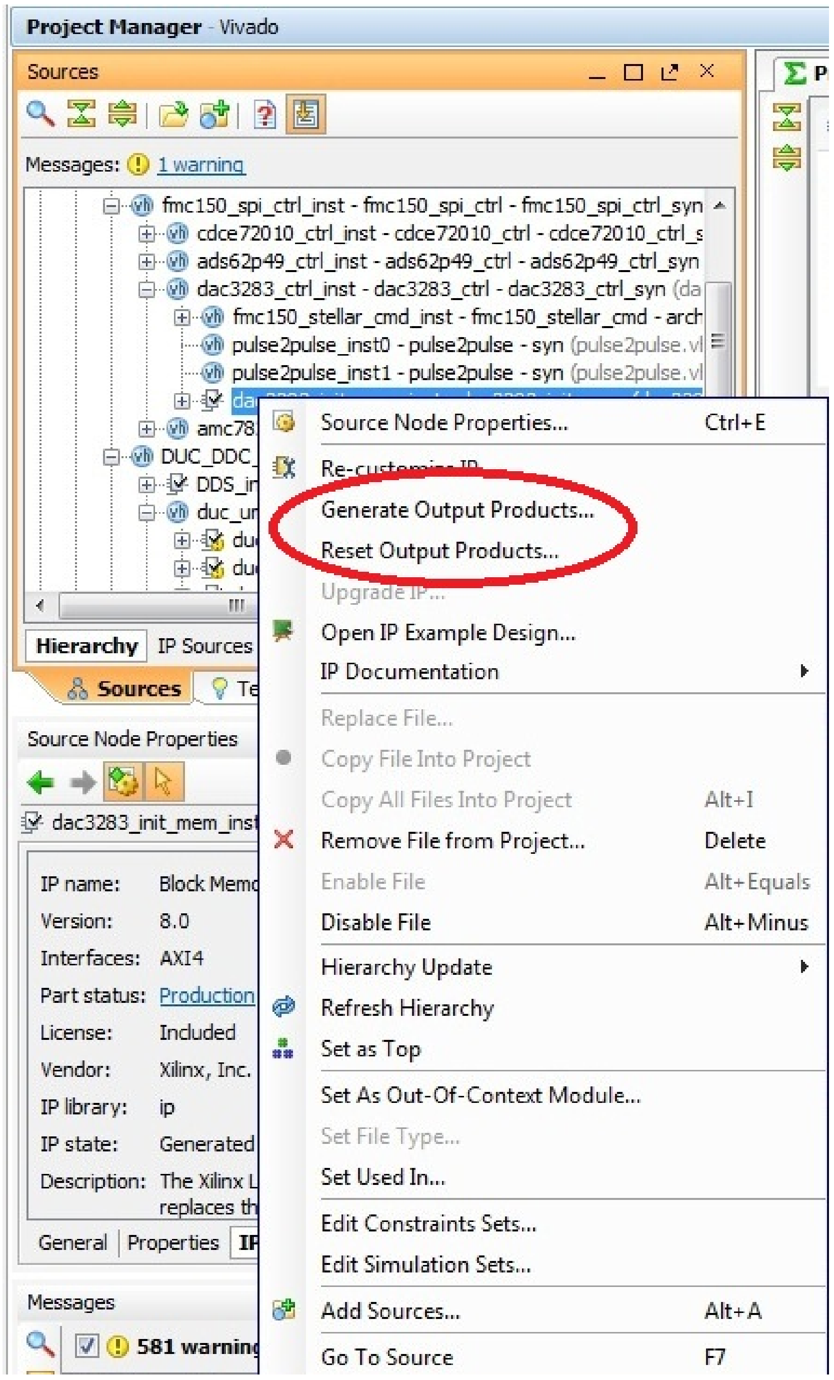
2. This file is the initialization file of the Read-Only-Memory that is used to configure the DAC3283 device after a reset/power up. In order to change the value of register 0x04 from 0xFF into 0x77, we change the line 04FF into 0477 and save the file.



```
;*******************************************************************
;** DAC3283 Initialization                                        **
;*******************************************************************
;
; This .COE file specifies initialization values for a block
; memory of depth=32, and width=16.
; Values are specified in hexadecimal format:
; 8 MSB = address, 8 LSB = data.
;
; Loading from ROM stops when address 0x1F has been processed.
; therefore this register must be the last entry in this file.
;
memory_initialization_radix=16;
memory_initialization_vector=
0070,
0121,
0200,
0310,
04FF,
0500,
0600,
0700,
0800,
0955,
0AAA,
0B55,
0CAA,
0D55,
0EAA,
0F55,
10AA,
1124,
1202,
13C2,
1400,
1500,
1600,
1704,
1883,
1900,
1A00,
1B00,
1C00,
1D00,
1E24,
1F52;
```

Figure 31 – Xilinx .coe File for programming data converter devices

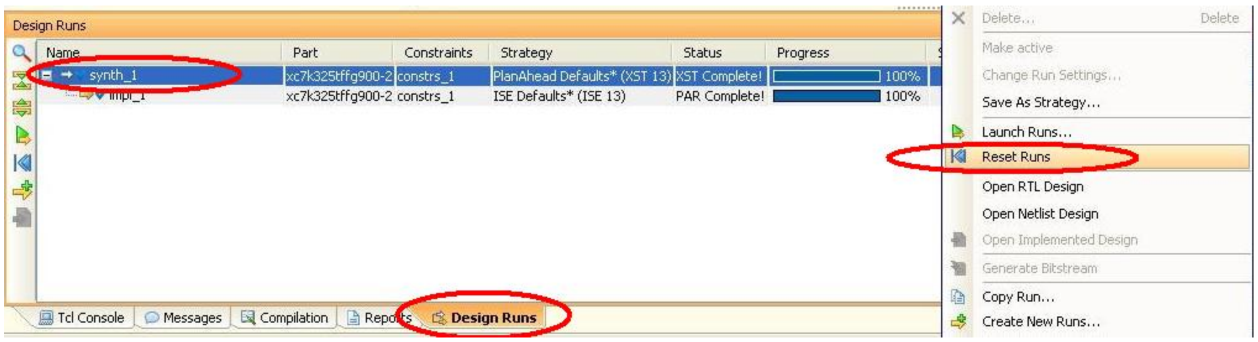
3. From the Vivado project manager select the dac3283_init_mem instance in the hierarchy view and right click to select "Reset Ouput Products". And then select "Generate Output Products" This will generaterate the IP outputs with the design change.


Project Manager - Vivado
Sources
Messages: 1 warning
fmc150_spi_ctrl_inst - fmc150_spi_ctrl - fmc150_spi_ctrl_syn
cdce72010_ctrl_inst - cdce72010_ctrl - cdce72010_ctrl_s
ads62p49_ctrl_inst - ads62p49_ctrl - ads62p49_ctrl_syn
dac3283_ctrl_inst - dac3283_ctrl - dac3283_ctrl_syn (da
fmc150_stellar_cmd_inst - fmc150_stellar_cmd - arch
pulse2pulse_inst0 - pulse2pulse - syn (pulse2pulse.vl
pulse2pulse_inst1 - pulse2pulse - syn (pulse2pulse.vl
Hierarchy
IP Sources
Sources
Source Node Properties
dac3283_init_mem_inst
General
Properties
Messages
581 warning
Source Node Properties... Ctrl+E
Re-customize IP...
Generate Output Products...
Reset Output Products...
Upgrade IP...
Open IP Example Design...
IP Documentation
Replace File...
Copy File Into Project
Copy All Files Into Project Alt+I
Remove File from Project... Delete
Enable File Alt+Equals
Disable File Alt+Minus
Hierarchy Update
Refresh Hierarchy
Set as Top
Set As Out-Of-Context Module...
Set File Type...
Set Used In...
Edit Constraints Sets...
Edit Simulation Sets...
Add Sources... Alt+A
Go To Source F7

IP name: Block Memo
Version: 8.0
Interfaces: AXI4
Part status: Production
License: Included
Vendor: Xilinx, Inc.
IP library: ip
IP state: Generated
Description: The Xilinx L replaces th

Figure 32 – Regenerating IP

4. Now select the Design Runs tab the select synth_1 then right click to select Reset Runs. This will remove previous runs and mark the design for rebuilding.



Name	Part	Constraints	Strategy	Status	Progress
synth_1	xc7k325tffg900-2	constrs_1	PlanAhead Defaults* (XST 13)	XST Complete!	100%
impl_1	xc7k325tffg900-2	constrs_1	ISE Defaults* (ISE 13)	PAR Complete!	100%

Figure 33 – Mark the Design for Rebuilding.

5. After this completes click on Generate Bitstream from the Vivado Project Manager to rerun synthesis and implementation and bit stream creation.

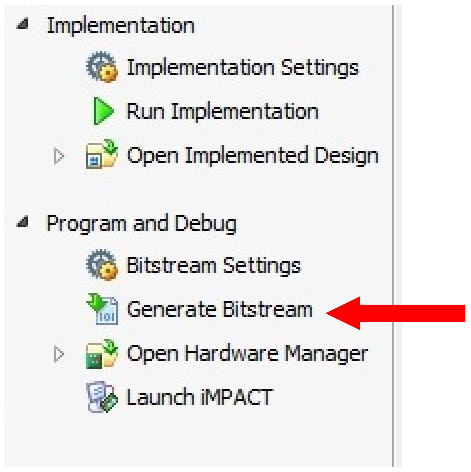


Figure 34 – Regenerating the Bitstream

6. When the design flow process is completed a new programming file (.bit) is available in the Vivado project directory. In the next steps we are going to verify the register content using Vivado VIO. These steps are optional.

7. As shown in the previous labs detailed steps Open Hardware session, attach to the HW , program the device, trigger Vivado Analyzer, open the waveform configuration file and check the amplitude..

8. To chack the changed rester value in HW double click “VIO Console” and take the following steps to read a register:

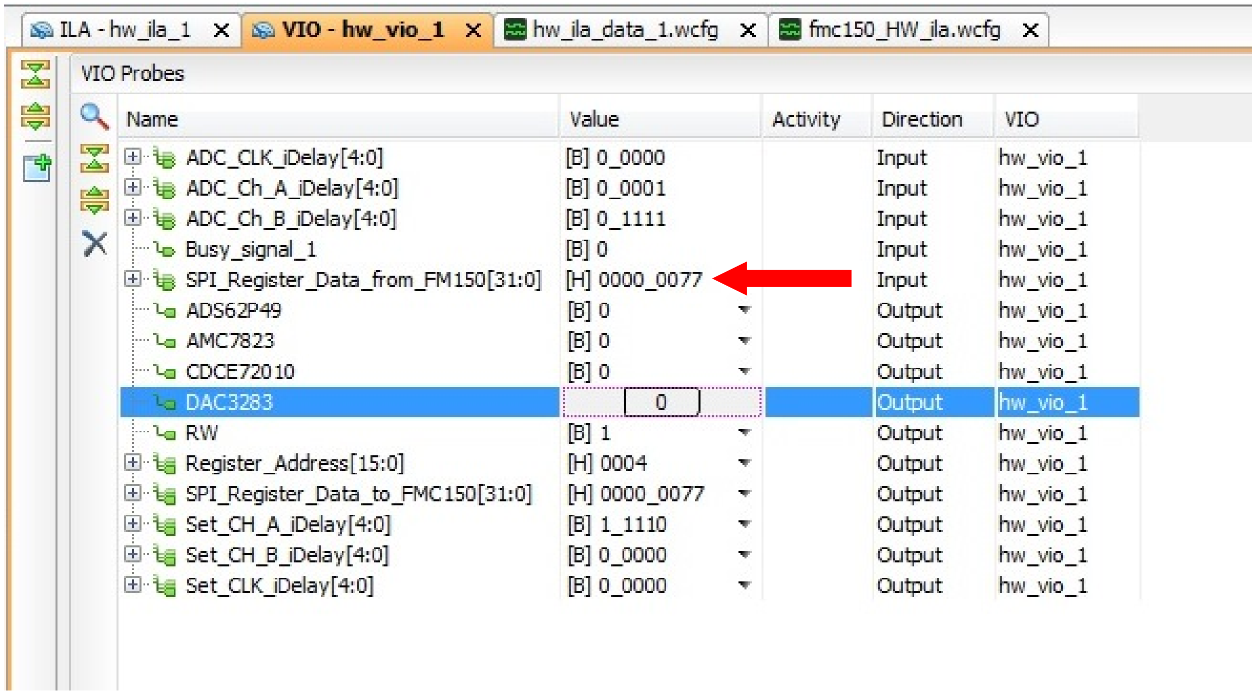
   a. Verify Register Address is set to 04

   b. Change RW to 1

   c. Toggle DAC3283 0 to1.

   d. Toggle DAC3283 1 to 0.

   e. Observe that the “Register Data Out” field reads 00000077.



Name	Value	Activity	Direction	VIO
ADC_CLK_iDelay[4:0]	[B] 0_0000		Input	hw_vio_1
ADC_Ch_A_iDelay[4:0]	[B] 0_0001		Input	hw_vio_1
ADC_Ch_B_iDelay[4:0]	[B] 0_1111		Input	hw_vio_1
Busy_signal_1	[B] 0		Input	hw_vio_1
SPI_Register_Data_from_FM150[31:0]	[H] 0000_0077		Input	hw_vio_1
ADS62P49	[B] 0		Output	hw_vio_1
AMC7823	[B] 0		Output	hw_vio_1
CDCE72010	[B] 0		Output	hw_vio_1
DAC3283	0		Output	hw_vio_1
RW	[B] 1		Output	hw_vio_1
Register_Address[15:0]	[H] 0004		Output	hw_vio_1
SPI_Register_Data_to_FMC150[31:0]	[H] 0000_0077		Output	hw_vio_1
Set_CH_A_iDelay[4:0]	[B] 1_1110		Output	hw_vio_1
Set_CH_B_iDelay[4:0]	[B] 0_0000		Output	hw_vio_1
Set_CLK_iDelay[4:0]	[B] 0_0000		Output	hw_vio_1

Figure 35 – Read of the DAC3283 Register

Congratulations!

You have changed an RTL register value on the FMC150 card; re generated the design outputs and verified the new setting using   the Vivado VIO.

RTL-AES-K7DSP-325T-G-2013.3-v1