

Virtex-6 ML605 Development Board with 4-DSP FMC-150 ADC/DAC

Department of Electrical and Computer Engineering Real-Time DSP and FPGA lab

> Colin Fera Matthew Luscher Ashkan Ashrafi



Figure 1 - ML605 and FMC-150

Introduction	2
Project Specific Customizations	3
Requirements Software Requirements Hardware Requirements Limitations and Specifications	5 5
Setup Hardware Connecting the ML605 and FMC-150	7
Connecting the Waveform Function Generator	
Block Diagram	
Software	
Running the Design Verification of Setup Implementation a Digital FIR Filter using Xilinx ISE Process	
Instantiation	
Verification	
Appendix Running the ML605 Reference Design In ISE13.1 Designing a Digital LPF in MATLAB Process	
MATLAB Code	
MATLAB Filter Coefficients	
References	

Introduction

The purpose of this tutorial is to help familiarize readers with the use of the AVNET Xilinx Virtex 6 based ML605 development kit and in particular the accompanying 4-DSP FMC-150 analog to digital converter (ADC) and digital to analog converter (DAC) for real time digital signal processing (DSP) applications. This tutorial assumes familiarity with hardware description languages (HDL's) and basic concepts of digital signal processing.

By following this tutorial and using the accompanying VHDL\Verilog code the reader should be able to:

- Configure the DAC to pass data through from the ADC
- Implement a basic Low pass filter with the aid of MATLAB (*Located in the appendix*)

Portions of this tutorial have been taken from various AVNET and Xilinx tutorials and documentation accompanying the board. Document names as well as links are provided in the references section at the end of this document.

Project Specific Customizations

- Project based on AVNET RTL *Reference Design Tutorial (Available through their website for ISE 13.1).*
- Sampling rate changed to the maximum supported by the internal clock of the FMC-150 at 245.11MHz.
- The Digital Up Converter (DUC), Digital Down Converter (DDC), and Direct Digital Synthesizer (DDS) IP Cores, and supporting code, were removed as they were not needed for the purposes of this tutorial; however, if you wish to follow the aforementioned tutorial of which this project is based on, the DUC, DDC, and DDS are an integral part of their (AVNET) tutorial and must be implemented in the project provided with this tutorial.
- An order 80 digital low-pass filter is implemented in addition to this tutorial as a reference to verify the functional behavior this tutorial. See Appendix
- The project file, originating in ISE 13.1, was moved to ISE 14.4p by the means of instantiating the minimum required IP cores for the demonstration purposes of this tutorial, which suggests not all the IP cores in this project are a necessity for proper functional behavior (i.*e. the aforementioned LPF*).

Requirements

This section reviews the minimum hardware and software requirements needed to successfully complete the tutorial.

Software Requirements

- ISE 14.4p (Latest version available when this documents was created. See Code Customizations section for pseudo-instructions on moving the ISE project, and dependencies, to a different version of ISE.)
- MATLAB (Optional for design of LPF as mentioned in the Appendix. Also, MATLAB *requires the DSP toolbox.*)

(*Note*: Xilinx ISE must be a fully licensed product. The free web pack will not work.)

Hardware Requirements

- AVNET Xilinx ML605 Development Kit
- 4-DSP FMC-150 ADC/DAC
- Waveform Signal Generator
- Oscilloscope (2- 3 channel)
- MMCX to BNC Coax Cable (*Qty 3*)
- BNC to BNC Coax Cable
- BNC Splitter

Limitations and Specifications

A key limitation of the FMX-150 is the ADC/DAC is AC coupled effectively creating a highpass filter and imposes bandwidth limitations on the hardware. In addition, the DAC has an 82MHz 5th order Chebyshev low-pass filter on its output. Listed below summarizes the useful reliable-operational bandwidths, as well as some key limitations, outlined in the FMC-150 user manual.

ADC

- Bandwidth: 400KHz 250MHz
- Input voltage range: $2V_{p-p}$ (note: *the gain is adjustable so achieve a* $1V_{p-p}$ *range*)

DAC

- Bandwidth 3MHz 82MHz
- Output voltage range: 1V_{p-p}

4-DSP, the manufacturer of the FMC-150, offers a service to convert the FMC-150 ADC/DAC to DC coupling. In addition, they can alter the low pass filter on the DAC. The unit would have to be sent in for the modifications at a cost.

Setup

The following instructions will guide the reader through the setup of the aforementioned required hardware and software for this tutorial.

Hardware

Connecting the ML605 and FMC-150

1. Insert the FMC-150 into LPC FMC connector (*J63 on the ML605 board as shown in Figure* 1).



Figure 2 – Inserting the FMC-150 board into the ML605 board. Apply slight pressure to ensure proper connection.

2. Connect ADC input B and DAC outputs C and D to MMCX to BNC cables as shown below. Inserting these cables requires slight force (*you should heard and feel the connection "click" into place. See Figure 2*).



Figure 3 – Connecting the signal cables to the FMC-150. When inserted properly, the reader should feel and hear the cables "click" into place. Inputs A and External Clock are not used in this tutorial.

- 3. Connect a mini-B USB cable to the USB female J22 on the ML605 labeled JTAG (*As shown in Figure 4*). Connect the other end to your computer.
- 4. Set the dipswitches on the lower right corner ML605 board, adjacent to the USB, to all OFF.



Figure 4 – Connect a mini-B USB cable to the ML605 board and connect the other end to the PC with ISE currently running.

Connecting the Waveform Function Generator

- 1. Attach the BNC splitter to output of the function generator. This enables the function generator's output to be observed in parallel with the output of the FMC-150 DAC.
- 2. Connect one of the outputs from the function generator to an input of the oscilloscope *(Channel one of the scope is connected to the output of the function generator as shown in Figure 6).*
- 3. Connect the other output of the function generator to the ADC on the FMC-150 board using the MMCX to BNC Coax Cable on port A (*Or port B, but note which port the function generator is connected to because it will referenced later in this tutorial for specific configuration of the port*).
- 4. Turn on the function generator and oscilloscope and set the function generator to output a $1V_{p-p}$ sine wave at 5MHz.
- 5. Set the oscilloscope to display the waveform to verify proper connection. Using the Auto Set feature on the oscilloscope should provide you with a decent-viewable window of the signal from the function generator (*See Figure 4*).
- 6. Connect DAC channels C and D to the oscilloscope on channels 2 and 3 of the oscilloscope; respectively.
- 7. Ensure the waveform is as expected on the oscilloscope and turn the power to the board on (*As shown in Figure 5*).



Figure 5 – 1) Connection to the oscilloscope with attached BNC splitter on the output of the function generator. 2) Connection of the function generator to the input of the ADC on port A of the FMC-150 board (*connection of the* MMCX to the FMC-150 board not shown here, please see Figure 2). Refer to the documentation provided with the oscilloscope and function generator as needed.

Block Diagram



Diagram 1 – Block diagram of how to setup should look.

Software

Before beginning this section ensure that ISE 14.4 is installed and licensed.

- *1.* Unzip the software provided with this tutorial (fmc150_ISE_14_4.zip) to a suitable location (*Experience has shown that locations with spaces in the path may cause issues*).
- 2. Launch ISE Design Suite 14.4
- 3. From the *File Menu*, select *Open Project* and locate the ISE project file '*fmc150_ISE_14_4*' (*as shown in Figure X*)

	I	SE	Project Navig	ator (P.49d) -	C:\fmc150_ISE_14_4\fm	nc150_IS	E_14_4.xis	e - [ml605_fmc150.vhd]	l	
		File	Edit Vie	w Project	Source Process T	ools W	indow	Layout Help		_ & ×
****	[New Project		で x 回	» 🏓	PB	ø 🄊 🖻 🔪 🔚 🖻	» 🏓	» 🕨 » 💡
D	e		Open Projec		↔□₽	×	112	clk to fpga p	: in	std logic 🔺
			Open Examp	le) 🌆 Simulation	₽ ∃	113	clk_to_fpga_n	: in	std_logic
			Project Brow	ser			114	<pre>ext_trigger_p</pre>	: in	std_logic
8			Copy Project	t		=	115	ext_trigger_n	: in	std_logic
8			Close Project	t		5	116			
	-				MI 605 free 150 mm	(m 📃	117	Serial Periphe		
0	8		New	Ctrl+N	- ML605_fmc150_syn		118	spi_sclk	: out	std_logic
13		2	Open	Ctrl+O		, <u>1</u>	119	spi_sdata	: out	std_logic
- 11	۰.			carro			120			
É	2		Close			1	121	ADC specific	-	and lease
	. [Save	Ctrl+S		1 %	122 123	adc_n_en adc_sdo	: out : in	std_logic std logic
H	×	¥.		Curro		- 24	123	adc reset	: out	std_logic
			Save As			74	124	ado_reset	. 040	Stu_rogit
		þ	Save All				125	CDCE specific	signals	
4					ected.		127	cdce n en	: out	std logic
	ł.		Print Preview	V		6	128	cdce sdo	: in	std logic
		6	Print	Ctrl+P			129	cdce n reset	: out	std logic
4	4				-		130	cdce n pd	: out	std logic
1	-		Recent Files	•			131	ref_en	: out	std_logic
	1		Recent Proje	cts 🕨			132	pll_status	: in	std_logic
	-		Recent roje		-		133			-
			Exit				< ^ ^ ^	D20		+
	-	St	tart 🖳 De	sign 🚺 Fil	les 🚺 Libraries	JE	ADC_au	ito_calibration.vhd 🗵 📄	ml605_fmc1	50.vhd 🛛 🕨
(8	En	rors 🐹 Find	in Files Results	s 📃 Console <u> </u> Wa	arnings				
(Dpe	en a	an existing pr	oject					Lr	150 Col 1 VHDL

Figure 6 – Opening the project from ISE.

Comp	uter	▶ OS (C:) ▶ fmc150_ISE_14_4 ▶		_	✓ Search	n fmc150_ISE_14_4	
Organize 🔻 New fo	lder					II • 🔲	0
Favorites	<u>^</u>	Name	Date modified	Туре	Size		
Desktop		👑 xinx_auto_0_xdb	5/18/2013 1:49 PM	File folder			
Downloads		퉬 xst	5/17/2013 1:39 PM	File folder			
Recent Places	=	ads62p49_init_mem	5/17/2013 1:44 PM	Xilinx ISE Project	6 KB		
Mecent Places		🚾 amc7823_init_mem	5/17/2013 1:44 PM	Xilinx ISE Project	5 KB		
🗎 Libraries		🚾 cdce72010_init_mem_ext	5/17/2013 1:44 PM	Xilinx ISE Project	5 KB		
Documents		🚾 cdce72010_init_mem_int	5/17/2013 1:44 PM	Xilinx ISE Project	5 KB		
Music		🚾 dac3283_init_mem	5/17/2013 1:44 PM	Xilinx ISE Project	5 KB		
Pictures		rss fmc150_ISE_14_4	5/18/2013 1:53 PM	Xilinx ISE Project	44 KB		
Videos		icon	5/17/2013 1:44 PM	Xilinx ISE Project	6 KB		
Videos		ILA_ADC_cali	5/17/2013 1:44 PM	Xilinx ISE Project	5 KB		
Computer	-	📧 ila_baseband_out	5/10/2013 11:04 PM	Xilinx ISE Project	5 KB		
	nam	ne: fmc150 ISE 14 4			▼ ISE Proi	ect Files (*.xise)	•
r ne	: Hall	Incib0_156_14_4			+ ISE FIOJ	ecci lies (.xise)	

Figure 7 – Open the project from the location of the unzipped contents of the project provided with this tutorial.

4. Select the top file *ML605_fmc150* and in the process window (directly below the *Design Hierarchy*), double click *Generate Programming File (As shown in Figure 8)*. This will synthesize the design producing a bit file. This step takes approximately three minutes to complete.



Figure 8 – Select the top module from the Design pane and double click *Generate Programming File* to generate a bit file and synthesize the design.

Running the Design

In this part of the tutorial you will program the FPGA with the bit file created in the *Preparing the Software* section. You will then configure ADC channel(s) for optimal performance. Finally you will pass a sine wave into the ADC from the function generator where it will be output by the DAC and viewed on the oscilloscope.

Verification of Setup

- 1. With the Xilinx project navigator open from the last step of *Preparing the Software*, double click on *Analyze Design Using ChipScope (A window should popup as shown in Figure 9*).
- 2. In the upper left of the ChipScope Pro window, click the 🛱 icon. This icon opens the JTAG search chain and searches for Xilinx cores.
- 3. Two devices should be detected (*displayed in the-most pane of the ChipScope window*), DEV: 0 and DEV: 1.
- 4. Right click *DEV:* 1 and select *Configure* (As shown in Figure 9).



Figure 9 – This window will appear only after a bit file has been generated and the reader clicked on *Analysis Design using Chipscope*. This figure shows what you should expect to see in the left pane after step two from above.

- 5. After clicking on *Configure*, Click *Select New* File and locate 'ml605_fmc150.bit'
- 6. Select *open* then select *OK*. This is the bit file that was generated in *Preparing the Software*. This will program the FPGA using the selected bit file. (*Note, if the board is power cycled, the board must be reprogrammed*). See Figure 10.
- 7. The FPGA is now programmed and you should expect to see the ChipScope Pro leftmost pane populate (*As shown in Figure 11*).



Figure 10 - Select the generated bit file generated in *Preparing the Software*.



Figure 11 – After the FGPA is programmed, the lost-most pane should populate with relevant information regarding the detected-programmed devices.

- 8. Select UNIT: 0 from the left-most pane and double click *VIO Console* (*As shown in Figure 12*).
- 9. Select ADC channel A iDelay from the *VIO console*, type 25 and hit enter (*If you intend to use Chanel B of the ADC then it should set to the same value*).

Note: The connection between the ADC and DAC within the FPGA is not registered making setup and hold times more critical. The setup and hold times cannot be maintained within the FPGA at the sampling frequency used in this tutorial. The iDelay (Incremental Delay) adds a sub-clock period delay to the clock at critical points allowing the setup and hold times to be maintained. Without this you will essentially see noise being output by the DAC.

ChipScope Pro Analyzer [ml605_fmc150]	ATTA A Chapterin	
Eile View JTAG Chain Device VIO Win	idow Help	
😂 🕑 JTAG Scan Rate: 260 ms 💌 S	ii <mark>U i S</mark> X	
Project: ml605_fmc150	👹 VIO Console - DEV:1 MyDevice1 (XC6VLX240T) UNIT:0 MyVIO0 (VIO)	e 3 1
DEV:0 MyDevice0 (System_ACE_CF)	Bus/Signal	Value
DEV:1 MyDevice1 (XC6VLX240T) System Monitor Console	↔ ADC Ch A iDelay at output of iodelaye1	0
UNIT:0 MyVIO0 (VIO) VIO Console	↔ ADC Ch B iDelay at output of iodelaye1	25
- UNIT:1 ILA_DDC_Out (ILA)	► ADC CLK iDelay at output of iodelaye1	00
Trigger Setup Waveform	← Register Data / FPGA RD from FMC150	0000000
- Listing	- SPI bus BUSY	0
Signals: DEV: 1 UNIT: 0	◇ ADC Ch A iDelay	0
Async Input Port	► ADC Ch B iDelay	25
 Sync Input Port ADC Ch A iDelay at output of iodela 	► ADC CLK iDelay	00
- ADC Ch B iDelay at output of iodela	← Register Address	0000
 ADC CLK iDelay at output of iodela Register Data / FPGA RD from FMC 	♀ Register Data / FPGA WR to FNC150	00000000
CH: 47 SPI bus BUSY Async Output Port		
Sync Output Port Register Address		
ADC Ch A IDelay		
Reading file: C:\fmc150_ISE_14_4\ml60	5_fmc150.bit	

Figure 12 - VIO Console within ChipScope Pro window.

10. Check the dipswitches and ensure the four green LED's numbed 5-8 are illuminated (*As shown in Figure 13*). The green LEDs indicate the clocks on the FMC-150 and FPGA are phase-locked; this is critical. The LEDs are indicative of the proper functional behavior of the connection between the ML605 and FMC-150.



Figure 13 – LEDs 5-8 must be illuminated and indicate the clocks on the FMC-150 and ML605 are phase-locked.

11. Activate channels one and two on the oscilloscope, if not done so already, and verify that a sine wave at the previously set frequency of 5MHz is displayed on both channels. *(As shown in Figure 14)*



Figure 14 – The yellow signal shown in this figure is the output of the function generator and the blue signal is the signal from the DAC of the FMC-150 board.

Note: Using the BNC splitter with transmission lines at differing lengths is far from ideal. The sine waves displayed will not be identical but will be similar (As shown in Figure 14). The output from the DAC will be out of phase with respect to the function generator's output and have different amplitudes, which should be within about 30% of the function generator's measured reference signal, but the frequency and form should be very similar.

This completes this portion of the tutorial.

Implementation a Digital FIR Filter using Xilinx ISE

Process

1. First create the COE file; this file initializes the block memory with the filter coefficients. To change the filter coefficients this file will need to be reimported each time.

The format of the file is as follows:

The semicolon character ends every command and can also be used to denote comments. Any characters following the semicolon up until the next line return is ignored. The file should be saved in ASCI format with file extension *.coe*.

The following is the coefficient vector, one coefficient should appear on each line the lines are terminated with a comma until the last line that is terminated with a semicolon. The file can have any number of coefficients between 1 and over 200 depending on which IP core version and options are selected. The number of coefficients dictates the order of the filter.

```
COEFDATA=
fff1,
ffe5,
fff1;
```

2. Right click on the IP Core called lowpass and select remove and then ok. In the next part of this tutorial we will recreate this IP core.

radix=16; ; Denotes that the filter data is hex, other options are 2 or 10 ; for binary or decimal respectively.

	le Edit View Project Source Process											_
	o a X @ @ X 9 6	» / / / / / / / / 🧃	i 🖻 🖻		📼 🎤 🐶 🕨 🔊 🛠 💡							
esigr			⇔⊡₽×	6	Design Overview Summary	^		ML605_	fmc150 Proj	ect Stati	us	
	View: 💿 🎒 Implementation 🔘 🔝 Simulation				IOB Properties		Project File:	duc_ddc_umt	s_virtex6.xise	Parser	Errors:	No
	Hierarchy		^	(C) (C)	- Module Level Utilization							Errors
副		- dac3283_init_mem (dac3283_init_m 3_ctrl - amc7823_ctrl_syn (amc7823_c			Timing Constraints Pinout Report		Module Name:	ML605_fmc15	0	Implen State:	nentation	New
	📄 强 fmc150_stellar_cmd_inst	- fmc150_stellar_cmd - arch_fmc150		ÇĴ	- 🗋 Clock Report		Target	xc6vlx240t-2	f1156		Errors:	
		syn (pulse2pulse.vhd)		MINK.	Errors and Warning		Device:					
-		syn (pulse2pulse.vhd) syn (pulse2pulse.vhd)		30	Parser Messages		Product Version:	ISE 14.4		•	Warnings:	
23	p2p3 - pulse2pulse -	syn (pulse2pulse.vhd)		(4)	Synthesis Messages	=		Balanced			Routing	
2		e2pulse - syn (pulse2pulse.vhd) e2pulse - syn (pulse2pulse.vhd)		_	Translation Messages Map Messages	-	Design Goal:	Daidriceu			Results:	
	amc7823_init_mem_inst	- amc7823_init_mem (amc7823_init	mem.xco)		Place and Route Messages			Xilinx Default	(unlocked)		Timing	
	Sample_pass_inst - sample_pass fir1 - lowpass (lowpass.xco)				Timing Messages Bitgen Messages		Strategy:				Constraints:	
	— 🍕 icon_inst - icon (icon.xco)	If New Source	E		🔄 📓 All Implementation Messages		Environment:	System Settin	<u>qs</u>		Final Timing Score:	
	ila_dac_inst - ila_dac (ila_dac.xc)				Detailed Reports Synthesis Report							
	mib05_tmc150.uct	Add Copy of Source			Translation Report			Data	iled Reports			- I-
	No Processes Running	Open			Map Report Place and Route Report		Report Name				Warnings I	
	-	🚑 Remove		i.	Place and Route Report		Synthesis Report	Status	Generated	LIIOIS	warnings 1	mos
- L	Processes: fir1 - lowpass = 🏂 CORE Generator	Manual Compile Order	- <u>-</u>	l.	Power Report	T	Translation Report		-	-		
11 1	Manage Cores	an Set as Top Module			Design Properties		Map Report					
≎ brari		SmartGuide	++ □ & ×	1	Optional Design Summary Contents		Place and Route		-			
	es Source Libraries	Implement Top Module		1	Show Clock Report		Report					
- E	work		_		Show Failing Constraints		Power Report					
a		File/Path Display			Show Errors		Post-PAR Static Ti Report	ming				
		Expand All					Bitgen Report			-		
		Collapse All					orgenneport					
		Find Ctrl+l	:					-				
		Design Properties		E	Design Summary		×	Secon	dary Report	¢		1 6-
	Errors					_						
. .	21013	Source Properties										

Figure 15 - Removing the original LPF.

- 3. Right click anywhere in the design hierarchy window and select New Source
- 4. Click *IP (CORE Generator and Architecture Wizard),* type the name *lowpass* under the filename field and select *next.*

New Source Wizard	
Select Source Type Select source type, file name and its location.	
BMM File ChipScope Definition and Connection File Implementation Constraints File IP (CORE Generator & Architecture Wizard) MEM File Schematic User Document Verilog Test Fixture VHDL Module VHDL Ibrary VHDL Package VHDL Test Bench Embedded Processor	Elle name: Location: C:\ISE\COMPE475\Final Project\COMPE475_PL_HU\ipcore_dir ↓
More Info	Next > Cancel

Figure 16 - New source -> IP

- 5. When prompted, select *yes* to overwrite the existing core named *lowpass*.
- Expand the view to *Digital Signal Processing*, then *Filters* and highlight *FIR Compiler* (version 5.0, as shown in Figure 19) and click next and then finish.

Citizen Advectator	Desktopi <i>re</i>	oder	Layout Help	t, working's	luc dde umb	, virtex6.x	ise - (Desig	×
New Source Wizard	100	- 52	8 # Qal	a 1 74 - 1		S 10	b T 4	
New Source Wizard Select IP Create Coregen or Architecture Wizard IP Core View by Function View by Name Name Automotive & Industrial AXI Infrastructure AXI Infrastructure BaseIP Basic Elements Communication & Networking Debug & Verification Digital Signal Processing Filters Filters CIC Compiler DUC/DDC Compiler FIR Compiler FIR Compiler FIR Compiler		AXI4	AXI4-Stream AXI4-Stream AXI4-Stream AXI4-Stream	AXI4-Lite	Production Production Production Production Production	License	vendor xilinx.com xilinx.com xilinx.com xilinx.com	Lib ip ip ip ip ip
Math Functions Memories & Storage Elements Standard Bus Interfaces Video & Image Processing			III					4
Search IP Catalog:							Clea	r
All IP versions					Only IP	compatible	e with choser	n part
More Info						<u>N</u> ext	Ca	ncel

Figure 17 – Selecting a IP core

- 7. Click the vector drop down and change this to COE file, then click the browse button and select the COE file created in step 1. The COE file will be imported and the coefficients file field will show a path. If this area is highlighted in red then there may be an issue with the format of this file.
- 8. Fill in the input sampling frequency and clock frequency in this case they are 245.76 and 491.52 respectively. The first page of the fir compiler menu should look as shown below.

V FIR Compiler			X
Documents View			
Freq. Response 🗗 🗙	logi CŽRE	FIR Compiler xilinx.com:ip:fir	r_compiler:5.0
Frequency Response (Magnitude)	Coefficient Vector : 6,	DE File	·
Set to Display : 1 Range: 11 Filter Analysis Passband Stop band Range : 0.0 - 0.5 0.5 - 1.0 Min : -26.226911 dB - - - - - - - - - - 1.0 - - - - - 1.0 - - - - 1.0 - - - - - - - 1.0 - - - - - - 1.0 - - - - - - 1.0 - - - - - - - - - - 1.0 - 1.0 - -	Select format : Input Sampling Frequen Clock Frequency : Input Sample Period :	Frequency Specification icy: 245.76 Range: 0.000001550.0 MHz 491.52 Range: 245.76550.0 MHz 1 Range: 110000000 Clock cycles	
Max : 102.390973 dB 22.907857 dB Ripple : 128.617884 dB IP Symbol IP Freq. Response Implementation Details	Datasheet	Seck Page 1 of 4 Next > Generate Cancel	↓ <u>H</u> elp

Figure 18 – Defining the filter.

- 9. Click next to move on to page 2 of the FIR compiler menu.
- 10. Change the input data width to 14 (*Note the FMC-150 has a 14Bit ADC*)
- *11.* Change output rounding mode to any choice other than full precision, in this case we have chosen *symmetric rounding to zero*.
- 12. Change the output width to 16 all other choices can be left at the default values as shown below.

📢 FIR Compiler	4.4. not, working/uluc, ddc, umts, vintex6.xise - (mi805, fmc150.shd)	
Documents View		
Freq. Response 5 ×	logic FIR Compiler	xilinx.com:ip:fir_compiler:5.0
Frequency Response (Magnitude)	Filter Architecture : Systolic Multiply Accumulate Coefficient Options Use Reloadable Coefficients Coefficient Structure : Inferred Quantization : Integer Coefficients Quantization : Integer Coefficients Coefficient Width : 16 Range: 249 Best Precision Fraction Length Coefficient Fractional Bits : 0 Range: 00 Datapath Options Number of Paths : 1 Range: 116 Input Data Type : Signed • Input Data Fractional Bits : 0 Range: 014 Output Rounding Mode : Symmetric Rounding to Zero Output Width : 16 Range: 131	
Set to Display : 1 Range: 11 Filter Analysis Passband Stop band	Output Fractional Bits : 0	
Range : 0.0 - 0.5 0.5 - 1.0 Min : -26.226911 dB		xt > <u>G</u> enerate <u>C</u> ancel <u>H</u> elp

Figure 19 - FIR Compiler Widow

- 13. At this point, since no other options need to be changed from the default values click generate. This will generate the IP core; the process takes about 1-2 minutes.
- 14. The filter IP core you created should be visible under the Verilog module *sample_pass_inst;* this indicates that the filter is being instantiated under that module as shown below.



This concludes this part of the tutorial.

Instantiation

This code below is the actual *sample_pass_inst* where the filter in this tutorial is being instantiated. Note the multiplexers that control how data is routed between inputs and outputs and also between the inputs and the filter. The dipswitches that control the multiplexers are numbered one through eight, and are shown in Figure 25.

module sample_pass(
input clk_fs,clk_gfs,
input [13:0] din_a,din_b,
output [15:0] dout_a,dout_b,
input [7:0] gpio_dip_sw
);

//wires used in IO Muxes
wire [13:0] dinf;
wire [15:0] doutf;

//IO Muxes select how inputs are passed to outputs.

assign dout_a = (gpio_dip_sw[1:0] == 1) ? doutf : (gpio_dip_sw[1:0] == 2) ? {din_b,2'b00} : {din_a,2'b00}; assign dout_b = (gpio_dip_sw[3:2] == 1) ? doutf : (gpio_dip_sw[3:2] == 2) ? {din_a,2'b00} : {din_b,2'b00};

//combinational assignment of the either input A or B to the filter assign dinf = (gpio_dip_sw[5]) ? din_a : din_b;

//the actual instantiation of the filter.

lowpass fir1 (.din(dinf),.dout(doutf),.clk(clk_gfs));

endmodule



Figure 20 – Dipswitches.

Verification

At this point it is assumed that all previous tutorial sections have been completed

- Setup
 - Preparing the Hardware
 - Preparing the Waveform Signal Generator
 - Preparing the Software
- Design a Simple Digital Filter using MATLAB
- Implement a Digital FIR Filter using ISE
- 1. Ensure your setup identical to Diagram 1 in the *Preparing the Hardware* section.
- 2. Activate channels one, two and three on the oscilloscope and verify that a sine wave at the previously set frequency is displayed on all three channels (*As shown in figure 22. Note, the purple signal is the signal being passed through the filter. Its appearance should look very similar because the frequency of the signal from the function generator has not been tuned to the cutoff frequency of the filter*).



Figure 21 – Expected signals displayed on the oscilloscope.

- 3. Record the peak amplitude of the filtered sine wave to verify the cutoff frequency when you increase the frequency of the signal later in the instructions.
- 4. Increase the frequency of the function generator until the amplitude of the filtered signal drops to $.707*V_{peak}$ (of the reference signal). This is the -3DB point at which signal power has fallen by half its original voltage denoting the start of the filter's cutoff frequency.
- 5. Compare this point to the -3DB frequency shown in the MATLAB plot; they should similar, but not exact.

6. Continue to increase the frequency of the function generator beyond the filter cutoff. The amplitude of the filtered signal should fall gradually to zero as the frequency is increased towards the stop frequency (*As shown in Figure 23 and Figure 24*).



Figure 22 - Function generator output at 7Mhz.



This concludes this part of the tutorial.

Appendix

The appendix of the tutorial provides the reader with additional tutorials and documentation.

Running the ML605 Reference Design In ISE13.1

The purpose of this section is to provide notes regarding the process of running of the ML605 reference design tutorials provided by AVNET.

You can obtain the ML605 reference design tutorial of your choice from:

http://www.xilinx.com/products/boards-and-kits/EK-V6-ML605-G.htm

Notes:

- The reference designs do not work in any version of ISE newer than 13.1.
- Only the tutorial instructions themselves are different. The project code included with this tutorial is a minimization of the actual code from AVNET.
- Even if using ISE 13.1, some IP cores cannot be opened because they are not fully supported.
- You can obtain code versions of the tutorial for ISE back to 12.2 (from the link above). The IP cores in these versions can be opened/modified.
- The tutorial is unusable until the ADC CH A and B iDelay's are set, as demonstrated in the main tutorial. These should be set as soon as the ML605 board has been programed using ChipScope. The recommended value is approximately 25.
- While running the tutorial
 - The DAC outputs a constant ~12MHz sinusoid. This acts as a function generator to drive the ADC which is then viewed in ChipScope.
 - Dipswitch 3 can be switched to bypass the digital up converter and digital down converter such that the ADC input is passed directly out of the DAC output.
- The sampling frequency used by the tutorial is 66.44MHz

Designing a Digital LPF in MATLAB

Process

In this part of the tutorial you will generate a simple digital low pass filter using MATLAB. The filter will have quantized 16bit coefficients, such that it can be readily implemented on the FPGA.

The MATLAB script used to generate this filter is appendix A of this document. The following enumerated steps are provided the reader with a better understanding of the creation

- 1. First you must decide upon the specifications of your filter. For this particular filter we will specify: order, sampling frequency, pass band and stop band frequencies.
 - A high filter order is recommended, here will use order 80. This ensures that the filter cutoff is relatively sharp.
 - Pass band is the frequency at which the signal begins to roll off; here we chose 5MHz.
 - Stop band is the frequency at which the signal will be totally attenuated; here we chose 9MHz.
 - Sampling frequency is the frequency of sampling on the ADC. The code provided with this tutorial sets the sampling frequency at 245.11MHz
- 2. Save the filter specifications as variables in MATLAB.

```
fp = 5e6; %freq at beging of pass band = 5MHz
fst = 9e6; %freq at end of stop band = 9MHz
```

```
n=80; %filter order = 80
fs=245e6; %sampling frequency = 245MHz
```

3. Save the filter specifications in a MATLAB data structure

f=fdesign.lowpass('N,Fp,Fst',80,fp,fst,fs)

```
f =
```

Response: 'Lowpass' Specification: 'N,Fp,Fst' Description: {'Filter Order';'Passband Frequency';'Stopband Frequency'} NormalizedFrequency: false Fs: 245000000 FilterOrder: 80 Fpass: 5000000 Fstop: 9000000

4. Generate the low pass filter.

h = design(f, 'firls', 'Wpass', 1, 'WStop', 100, 'FilterStructure', 'dffir');

5. Change the data format for the filter that was just created to fixed.

%set filter to fixed type set(h,'Arithmetic','fixed');

6. Review the specifications of the filter that was just created.

h =

FilterStructure: 'Direct-Form FIR' Arithmetic: 'fixed' Numerator: [1x81 double] PersistentMemory: false

CoeffWordLength: 16 CoeffAutoScale: true Signed: true

InputWordLength: 16 InputFracLength: 15

FilterInternals: 'FullPrecision'

7. Use the filter visualization tool to view the frequency response of the filter. See Figures 15 and Figure 16.



hfvt = fvtool(h);

Figure 24 – Magnitude Response





8. Export the filter coefficients as 16bit hex such that they can be readily use in a Xilinx *COE* file to initialize block memory in an FIR core.

fcfwrite(h,'simple_lowpass','hex')

In the above command, h is the filter itself, simple_lowpass is the name of the file to be exported, hex is the data format.

The coefficients are listed under numerator:

Numerator: fe6b fd99 fca3

MATLAB Code

%The following script:
%-Sets parameters for a simple digital low pass filter
%-Generates the low pass filter
%-Displays the filters response graphically
%-Retrieves hex coefficients which can be directly used in a COE file
%

fp = 5e6; %freq at beging of pass band = 5MHz
fst = 9e6; %freq at end of stop band = 9MHz

n=80; %filter order = 80 fs=245e6; %sampling frequency = 245MHz

%Instruct MATLAB to design the filter specified above

f=fdesign.lowpass('N,Fp,Fst',80,fp,fst,fs);

% Generate a direct form low pass filter using above specifications

h = design(f, 'firls', 'Wpass', 1, 'WStop', 100, ... 'FilterStructure', 'dffir');

%set filter to fixed type

set(h,'Arithmetic','fixed');

%use the filter visualization tool to

hfvt = fvtool(h);

%retrieve filter coefficients as hex list

fcfwrite(h,'simple_lowpass','hex')

MATLAB Filter Coefficients

% Generated by MATLAB(R) 7.12 and the Signal Processing Toolbox 6.15.
% Generated on: 19-May-2013 16:44:36
% Coefficient Format: Hexadecimal

% Discrete-Time FIR Filter (real) % -----% Filter Structure : Direct-Form FIR % Filter Length : 81 % Stable : Yes % Linear Phase : Yes (Type 1) % Arithmetic : fixed : s16,19 -> [-6.250000e-02 6.250000e-02) % Numerator % Input : s16,15 -> [-1 1) % Filter Internals : Full Precision : s36,34 -> [-2 2) (auto determined) % Output : s31,34 -> [-6.250000e-02 6.250000e-02) (auto determined) % Product % Accumulator : s36,34 -> [-2 2) (auto determined) % Round Mode : No rounding % Overflow Mode : No overflow Numerator: fe6b fd99 fca3 fb8b fa57 f90c f7b4 f659 f507 f3ce f2bc f1e2 f151 f11b f14e f1fc f333 f4fe f766 fa72 fe24 027c 0774 0d02 1319 19a7 2096 27cf 2f34 36a8 3e0a

f1fc	1151	f14e f11b f151	
	f1fc	f766 f4fe	
f1e2 f2bc f3ce		f659 f7b4 f90c	
f1e2 f2bc f3ce f507 f659 f7b4 f90c	f659 f7b4 f90c	fb8b fca3 fd99	
f1e2 f2bc f3ce f507 f659 f7b4 f90c fa57 fb8b fca3	f659 f7b4 f90c fa57 fb8b fca3	fe6b	

References

- [1] No author provided. 2010. Xilinx Vertex-6 DSP Development Kit with High-Speed Analog. Available: https://www.em.AVNET.com/Support%20And%20Downloads/virtex6dsp_rtl_referen ce_design_tutorial_13_1.zip
- [2] No author provided. 2010. FMC150 User Manual. Available: https://www.em.AVNET.com/Support%20And%20Downloads/FMC150_user_manual. pdf