Lab 4 – An FPGA Based Digital System Design
ReadMeFirst

Lab Summary

This Lab introduces a number of Matlab functions used to design and test a lowpass IIR filter. As you have seen in the previous lab, Simulink allows you to simulate the digital IIR filter using the difference equation. The same digital filter design can be implemented in the FPGA on the DE2 board. Using the lab bench equipment, you can test the actual filter and compare the performance to the theoretical values obtained in Matlab and Simulink.

Lab Preparation

1) There is a separate document called ***Lab 4 Background.pdf*** that covers some of the terminology and mathematical logistic associated with this lab.

2) Review the material from Difference Equations in Matlab and Simulink from Lab 12.

3) Be familiar with using Quartus II, the DE2 board, and, in particular, the CODEC.

4) You will need a USB thumbdrive (must be FAT32 formatted, not NTFS, in order to work with the lab oscilloscope)

Lab Supplies

DE2 Board x 1
USB Blaster Cable x 1
BNC Coaxial Cable x 4
BNC to RCA Adapter x 3
BNC to Dual RCA/3.5 mm Cable x 2
MSOX-2012A Oscilloscope
DP1022A Function Generator (Arbitrary Waveform Generator)

Part 1: Implementing a Digital Filter in Simulink

There are a number of IIR filter types, each with its own advantage and disadvantage. For the next two labs we will design Butterworth IIR filters. These filters have the characteristic of being maximally flat, or consistent gain, in the passband of the filter. The Matlab command to generate filter coefficients for a Butterworth filter is

```
[b,a] = butter(2,Wn,'low');
```

where b and a are the arrays of the coefficients used in the difference equation of the IIR filter. The 2 in the function argument indicates a 2-pole filter which will return a three element array for b and a. The difference equation for a second order filter takes the form:

$$y[n] = b_0x[n] + b_1*x[n-1] + b_2*x[n-2] + a_1*y[n-1] - a_2*y[n-2]$$

a_0 is always 1. Wn is the normalized cutoff frequency of the filter which can be calculated as:

$Wn = f/ (Fs/2) = 2*f/Fs$

Fs = sample frequency = 46860 Hz (for DE2 board)

f = desired cutoff frequency in Hertz

****

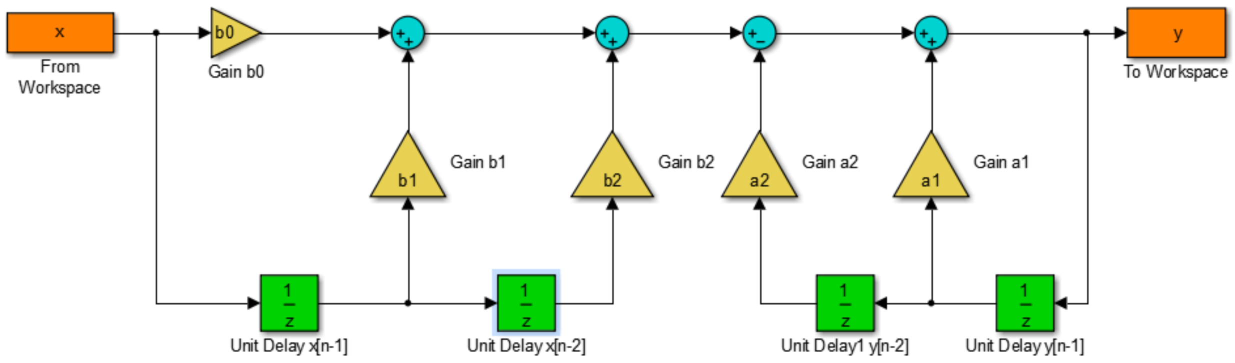
NOTE: The normalized cutoff frequency is usually calculated using the formula Wn = f/Fs. However, the Matlab butter() function uses the formula Wn = f/(Fs/2)

****

Procedure:

1) Use the ***butter()*** command in Matlab to generate the coefficients for a 2 pole lowpass filter with a cutoff frequency of 1000 Hz.

2) Construct the following Simulink model for the difference equation using the coefficients:



Use the variable names in the gain blocks. These will be set in the Matlab file from which simulink is called.

3) Set the Simulation time:

- Start time: 0.0
- Stop time: n_stop (later defined in a MATLAB .m file)

4) Set Configuration Parameters (Solver options)

- Type: Fixed-step
- Solver: discrete (no continuous states)
- Fixed-step size: 1

5) Configure the Gain 2 Sum block (blue)

- Double click, then change |++ to |+-
    - This will change the bottom input to subtraction

6) Configure ***From Workspace*** block

- Variable Name: x

7) Configure ***To Workspace*** block

- Variable Name: y
- ***Save format: Structure With Time***

8) Save your model as ***IIR_Filter_Model.mdl*** (the .mdl is added automatically).

The Matlab script (.m file) initializes the Simulink model parameters and input signal, runs the model, and plots the filter's output.

9) Open the Matlab script ***IIR_Lab4.m*** and add your coefficients on the line prepared for each one.

10) Select the w_hat = 0.05*pi input signal by un-commenting the line of code.

11) Run the M file. The first plot is a comparison of the input signal and filtered output. The output should have a smaller amplitude than the input (and some phase shift). The

gain at this specific normalized frequency is displayed in the command window. The gain is calculated at the input amplitude divided by the output amplitude.

12) Matlab uses the ***freqz()*** function to display to generate data for a linear plot of the filter frequency response. The second figure should show the operating point of your input signal, but in the LINEAR frequency response plot, anything in the passband is usually off the page

13) The third figure shows the same data on a log - log format known as a ***BODE plot***. This is the standard way to display the frequency response of any filter. There is a ***bode()*** function in Matlab, but it was not used here.

14) Re-run the program with the other normalized frequencies and observe how the filter output and operating points change.

Result 1: Qualitatively discuss the effect the filter has for each of the normalized frequencies in terms of signal strength and phase shift

15) Use the data cursor tool to measure the cutoff frequency of the lowpass filter.

On any of the plots, you can use the data cursor tool to measure x,y pairs on the plot. Selecting the tool at the top of the window. Then click on the graph where you want to measure. For more precise measurements, right click and select Selection Style -> Mouse Position.

16) The bandwidth of the filter is the range of frequencies that are not affected by the filter for a lowpass filter the bandwidth is from DC (zero Hertz) to the cutoff frequency. What is this filter's bandwidth?

Result 2: Discuss and record your cutoff frequency & bandwidth measurement in your report.

Checkpoint 1: Show your plots for cutoff frequency and bandwidth measurement to the Lab Monitor.

Part 2: Implementing the Filter in an FPGA

You will be using the DE2 CODEC in combination with the FPGA to design and use the IIR lowpass filter.

Procedure:

1) Download the project ***filter_proj_skeleton.qar*** from the Lab 4 section of the ECE 2050 Lab website onto a new folder on your **Z:\ Drive** and open it.

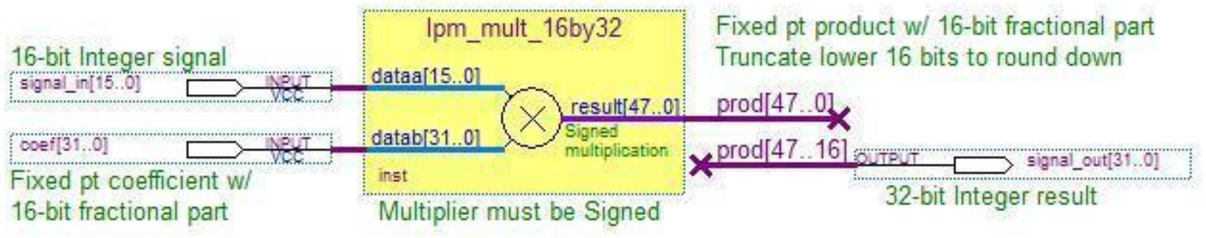
****

NOTE: Do NOT run ANY Quartus projects in "My Documents", the Desktop, or any other My Documents subdirectories.

****

The IIR_Filter block and CODEC interface is already setup for you. Currently IIR_Filter block contains the necessary megafunctions to create the filter but are not connected.

The diagram below shows the fixed-point multiplier you will be using in this project (These are already inside a block for you). The setup assumes that you will be multiplying a 16-bit integer with a 32-bit fixed-point value with 16 fraction bits. The result of the multiplication is then rounded down to a 32-bit integer.



1) Open the block diagram for *IIR_Filter.*

2) Implement the difference equation you created in Matlab with the components in the block diagram. The filter's structure should be very similar to the Simulink model.

****

NOTE: Only use the given components. Do not create or modify any of the lpm blocks or pins.

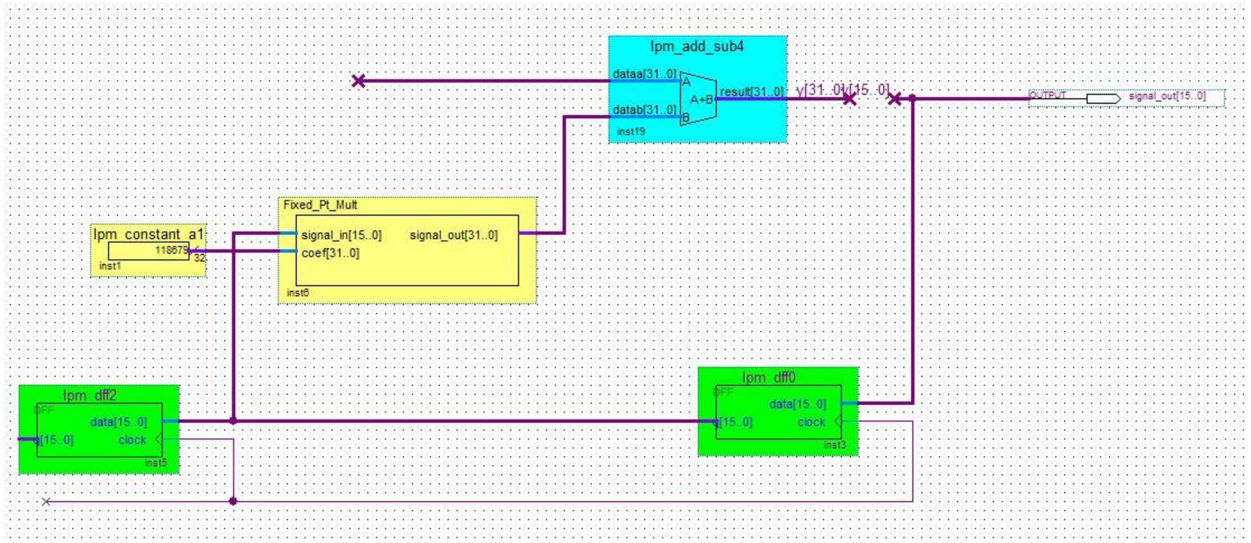
****

3) Use **D flip-flops** to delay a signal by one iteration. You can rotate the lpm_dff using the Right-click menu.

4) Use the **lpm_sub_32bit** module and **lpm_add_sub_32bit** module for the $\boldsymbol{a_2 y}[\boldsymbol{n}-2]$ and $\boldsymbol{a_1 y}[\boldsymbol{n}-1]$ terms respectively.

5) The **lpm_dff's** and **signal_out[15..0]** are 16-bits. You will need to use bus naming convention on the final output in order to shift the data of the 32-bit integer arithmetic megafuctions to match up to the 16-bit integer data signals at the top level of the design.

The first feedback stage on the output should look like this:



8) Use the fixed-point representation for coefficients for the lpm_constant values using the coefficient values you obtained using Matlab's 'butter' function earlier.

9) Compile the project and program the DE2 board.

In your report, you should compare this with the difference equation you implemented in Simulink.

Part 3: Characterizing the Digital Filter

Measure the cutoff frequency of your FPGA IIR filter by following the procedure below:

Procedure 1:

1. Channel 1 of the the MSO-X 2012A oscilloscope, should be connected to the function generator and Line-in of the DE2 Board (BLUE) using a BNC cable, BNC-to-RCA adapter and an RCA to phone plug cable. Use the BNC-T connector on CH1 of the function generator.

2. Channel 2 of the oscilloscope should be connected to Line-out of the DE2 board (GREEN) using the same combination of cables.

3. The Function generator should be set to 100 Hz, 1Vp-p Sine wave.

4. Use ‘Autoscale’ and ‘Meas’ on the Oscilloscope to display the peak-to-peak voltages of both the input and output channels.

5. Measure the Vp-p voltage **of the output signal** at 100 Hz.

6. Divide this value by $\sqrt{2}$.

   This will be the peak-to-peak voltage of the filter output when the signal has been attenuated **-3dB**, the necessary condition that defines the **cutoff frequency**.

7. Increase the frequency on the Function Generator until the output signal is equal to the calculated value. Again, this is when the Vp-p that you found at 100 Hz is reduced to Vp-p(100)/ $\sqrt{2}$.

8. Record the value on the function generator. This is the **cutoff frequency** in Hertz.

9. Calculate the normalized cutoff frequency ($\hat{\omega}$) of your signal in rad/sample and the frequency (ω), in radians/second, and in Hertz. Refer to Part 1 and the Background Slides in the Lab 4 section on the website for help.
   a. Assume that the sampling frequency (Fs) is equal to 46,860 Hz.
   b. $\omega = 2\pi$**f** where ω is in rad/second and **f** is in Hertz.

Result 3: Record the experimental cutoff frequency and bandwidth. Compare your measurements with your Matlab results in your report.

Procedure 2:

Now use your filter to reduce high frequency noise added to a real signal.

1) Set your function generator to 1 Vp-p and 100 Hz.

2) Connect channel 2 of the function generator to the second RCA connector on line-in (blue) of the DE2 board using another BNC cable and adapter.

3) Change the function generator from channel 1 to channel 2 by pressing the Ch1/Ch2 button.

4) For channel 2, press the noise button. Then set the amplitude to 5 Vp-p (it should already be this value by default).

5) Enable the channel 2 output. The DE2 project sums channel 1 and channel 2 signals to provide a noisy signal at the input of your filter.

6) The project uses SW[0] to switch between the filter input and output. Try it. Verify that the filter

reduces the random noise. Recall that the filtered amplitude is greater than the unfiltered

Checkpoint 2: Demonstrate your filter to the lab monitor

7) Use the **Save/Recall button** on the Oscilloscope to save a .png screenshot of both the filter's input and output signal onto your USB Drive.

Result 4: Include screenshots of your input and output in your report.

****

NOTE: You will be using the IIR Filter as a starting point in your next lab. Make sure all members of the group have access to a copy of the project. You can use *Project->Archive Project* to create easy-to-share *Quartus II Archive Files (.qar)*.

****

Checkpoint 3: Clean up and show your workbench to the lab monitor.

Discussion Questions:

Topic 1: What is attenuation? What does a lowpass filter do?

Topic 2: What does Infinite Impulse response mean? What makes a filter IIR?

Topic 3: You need a lowpass filter with a bandwidth of 0.25*π rad/sample and attenuation of 90% or more for frequencies above 0.4*π rad/sample. Listed below are difference equations for lowpass filters. **Which difference equation(s) meet the specifications? For those equations that do not meet the specs, explain why?** Use the ***freqz()*** command in Matlab and record the values $\boldsymbol{a_k}$ & $\boldsymbol{b_k}$ vectors in your answer. Review Part 1 and go over IIR_Lab4.m if you have trouble.

Watch your signs. Notice from Part 1 that some of the $\boldsymbol{a_k}$ coefficients have opposite sign from the corresponding difference equation coefficients.

Lowpass Filter 1):
$y[n] = 0.026x[n] + 0.05x[n-1] + 0.026x[n-2] + 1.5y[n-1] - 0.6y[n-2]$

Lowpass Filter 2):
$y[n] = 0.014x[n] + 0.04x[n-1] + 0.04x[n-2] + 0.014x[n-3] + 2.09y[n-1] - 1.8y[n-2] + 0.6y[n-3]$

Lowpass Filter 3):
$y[n] = 0.24x[n] + 0.08x[n-1] + 0.08x[n-2] + 0.24x[n-3] + 0.96y[n-1] - 0.66y[n-2] + 0.05y[n-3]$

Topic 4: How do you implement a time delay operation in hardware?