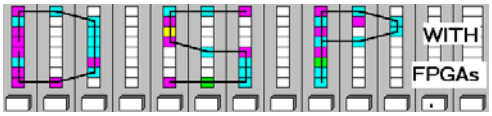
Lastname: _____

5Digit SS:_____

LABORATORY
FIR Filters



LAB 5: Introduction to Finite Impulse Response Filters

(10 points)

In this lab, you will be introduced to finite impulse response (FIR) filter design. Filters are one of the most important elements in DSP and are typically used to isolate a specific frequency band of a signal. FIR filters have low quantization sensitivity, simple structures, and can easily be made to have a linear phase response.

In the **pre-lab,** you will use "pencil-and-paper" to compute the results you expect later in your design implementation. In the **design part,** you will implement a half-band filter using direct coefficient coding and reduced adder graph design. Please bring your own pair of headphones to the lab so you can hear the effects of noise and filtration.

Lab Objectives

After completing this lab you should be able to

- Design and simulate an reduced adder graph, mean, and median filter
- Understand the difference between random, impulse and harmonic distortions
- Apply filtering to different noise types

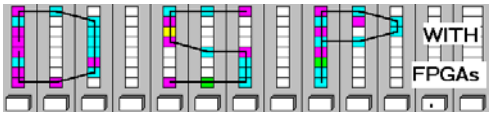
Pre-lab (3 points)

1. For a 4-tap mean ($y[n] = (x[n] + x[n-1] + x[n-2]+ x[n-3])/4$) and a 3-tap median ($y[n]$=median($x[n] + x[n-1] + x[n-2]$) determine the output values $y[n]$ for the input values $x[n]$ given below. Remember that the median filter sorts the data and then forwards the values to the output that is in the middle of the sorted list. Assume zero values ($x[n]$,$y[n]$=0) for n<0 and 10<n.

Filter type	***n***	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	***x*[*n*]**	16	0	-16	0	0	16	16	0	0	0	0
Mean	***y*[*n*]**											
Median	***y*[*n*]**											

1. In Lab 4 a function generator for random, impulse and harmonic noise was developed. Assume that a 1 kHz sine signal is corrupted by noise. What would be your suggestion to remove this noise? Complete the table below for your suggestions. Your choices are: Mean, Median or Halfband filter (i.e., low pass).

Sine signal corrupted by	**Filter used to remove noise**
Random noise	
Impulse noise	
Harmonic noise	



2. Draw the impulse response using "stair functions" for the following F5 length-11 half-band filter. (Coefficient values: $f[0] = f[10] = 3$, $f[2] = f[8] = -25$, $f[4] = f[6] = 150$, $f[5] = 256$)

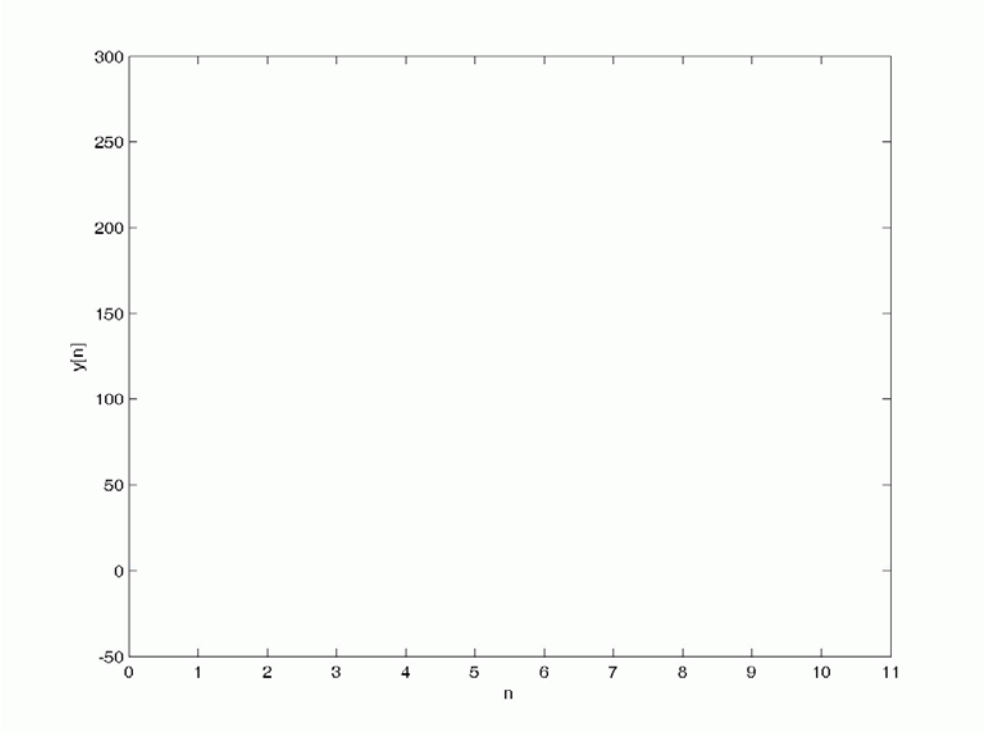


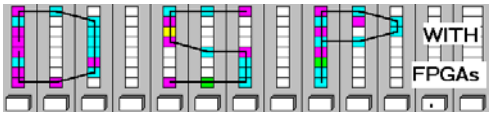
Figure 1: Stair plot of impulse response.

3. Determine the reduced adder graph (RAG) for the F5 length-11 half-band filter with the following coefficients: $f[0] = f[10] = 3$, $f[2] = f[8] = -25=-(3*8+1)$, $f[4] = f[6] =150=25*3*2$, and $f[5] = 256$)

Lastname: _____

5Digit SS:_____

LABORATORY
FIR Filters



VHDL Design-lab

Follow the directions below to implement a moving average (mean), median, and a half-band filter.

A. Getting Started

If you are in B114 or the digital logic lab:

1. On the desktop double click on **Engineering folder** and start version 9.1 of **Quartus II**. Do not use Quartus version 10 since it does not has a build-in simulator.

2. You should not save anything on the local hard disk. You will have to use a USB flash drive, or your "mapped" home directory to save the files. Create (if not done in a previous lab) a new folder named **DSPwFPGAs** on your mapped network drive.

B. Compiling an Existing F5direct Filter Design

1. Within your **DSPwFPGAs** folder make a new folder called **DE2_lab5**. Download the file "**f5direct.vhd**" from the class webpage and put the file in your **DE2_lab5** folder. Create a project `f5direct` and load the file `f5direct.vhd` in the project.

2. Now run a full compilation. From the **Compilation Report** and the **Classic Timing Analyzer Tool** (**Processing** menu) determine

   Total logic elements = _____

   Registered Performance = _____ MHz

   Enter these resource and performance data in your `f5direct.vhd` header file.

3. To run a simulation follow the usual procedure you used in the FPLD Quartus labs to do a simulation. As a reminder you may also consult lab 1 part E. Your simulation should match the waveform in Figure 1below. Finally add your electronic signature before printing the simulation in "landscape" format.

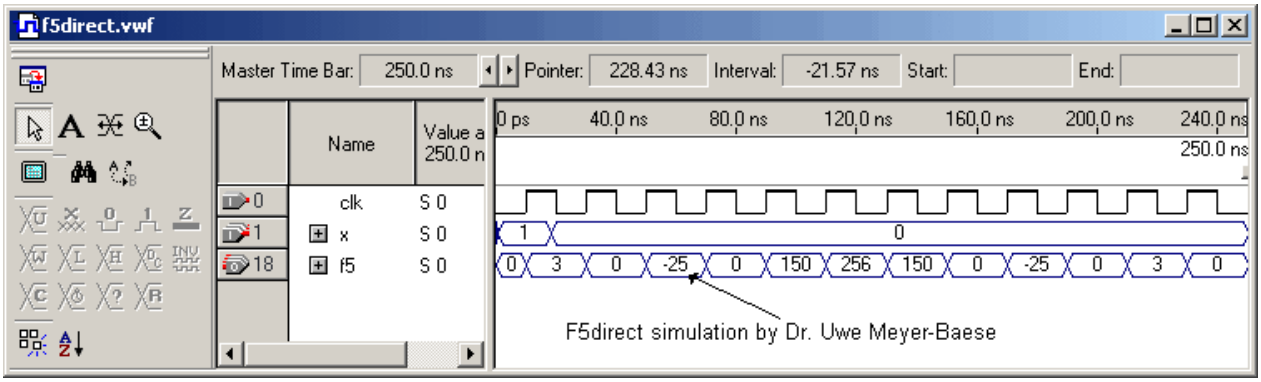
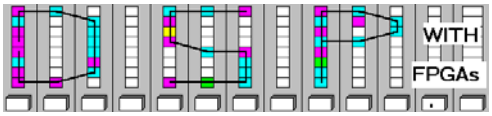


Figure 1: `f5direct.vhd` simulation.



C. Complete the F5 RAG VHDL Design

The direct-form FIR filter can be optimized in several ways. First, we can use a constant coefficient multiplier block (called gain) instead of the general-purpose multiplier. Or, we could use the transposed FIR form to reduce the adder delay. Another way would be to combine the coefficients using the reduced adder graph (RAG) you developed in the pre-lab.

1. Within your "**DE2_lab5**" folder create a new project called **fir**. Copy the filter `f5direct.vhd` to this project folder and rename it to `fir.vhd`.

2. Modify the design (i.e., add your reduced adder graph you had developed in the pre-lab) and simulate. Within **`PROCESS`** P1 use **`VARIABLE`**`s x3, x25, x75, x150, x256`, from the 26 bit signed data type, i.e., S26; to build the reduced adder graph.

3. Simulate the design with the impulse response and try to match with the `f5direct` simulation shown in Figure 1. (Do **not** print at this time.)

4. Now run a full compilation. From the report file and the **Classic Timing Analyzer Tool** (**Processing** menu) determine

   Total logic elements = _____

   Embedded Multiplier 9-bit elements = _____

   Total memory bits = _____

   Registered Performance = _____ MHz

   Enter these resource and performance data also in your `fir.vhd` header file.

5. Compare this design with the direct form design `f5direct.vhd`. Determine the improvements:

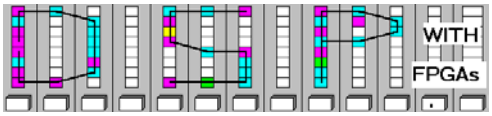
   Total logic elements improvements = _____(LEs)

   Registered Performance improvements = _____(MHz)

Lastname: _____

5Digit SS:_____

LABORATORY
FIR Filters



D. Complete the Mean and Median VHDL Design

In the pre-lab we saw that other filter types may be used to remove noise in the input signal. Two more of these filters are designed next.

1. First a 4-tap moving average FIR should be designed that has the difference equation $y[n] = (x[n] + x[n\text{-}1] + x[n\text{-}2] + x[n\text{-}3])/4$. Call the output of the filter `mean` and use the S16 data type for all signals. You need to design a shift register first, build the summation and then divide the sum by 4.

2. Next we design the 3-tap median filter. Remember that the median filter sorts the data and then forwards the values to the output that is in the middle of the sorted list. You will need a couple of **IF** comparisons to accomplish this function.

3. First perform a simulation with impulse response. Set **Edit->End Time** to 500 ns. Set `clk` period to 20 ns. You should add values ±16 to test the mean and median filters, see below. Single impulses should *not* appear in the median output. Finally add your electronic signature before printing the simulation in "landscape" format**.**

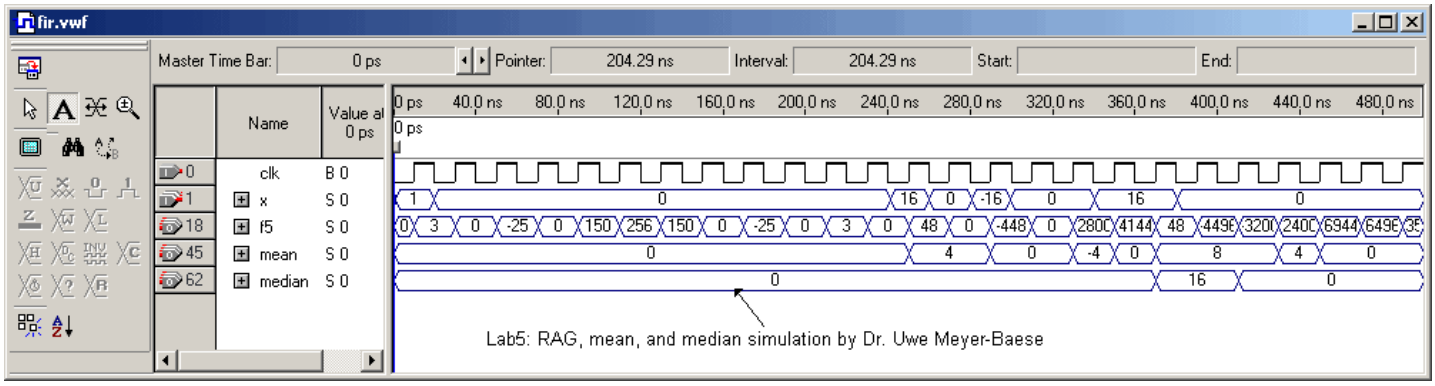


Figure 2: f5, mean and median filter simulation.

E. DE2 Implementation of the FIR Designs

1. Start Quartus II and build a new project called DE2_lab5. As device select EP2C35F672C6 and load the pin assignment file `DE2_pin_small.csv` (from lab4 folder or BlackBoard) into Quartus II using **Assignments-> Import Assignments ...**

2. Copy from your DE2_lab4 folder into the DE2_lab5 folder the following files: `DE2_lab4.vhd`, `audio_pll.vhd`, `audio_pll.qip`, `audio_pll.cmp`, `i2c_config.vhd`, `i2c_controller.vhd`, and `audio_dac.vhd`. Rename the file `DE2_lab4.vhd` to `DE2_lab5.vhd` and change the entity name to `DE2_lab5`.

3. Copy `fir.vhd` to DE2_lab5 and instantiate as a component into the `DE2_lab5.vhd` as follows:

```
C4: fir PORT MAP (clk=>DACLRCK, x=>x, f5=>f5, mean=>mean, median=>median);
```

Define the required signals in the correct width. For convenience copy the definitions for S16 and S26 data type from the `fir.vhd` file and place it in the signal definition section of `DE2_lab5.vhd`.

4. Keep `key(0)` as global reset for the registers and the functionality of `SW0`…`SW3` as in lab4. The signal `dac` should always be a register controlled by `DACLRCK` with `key(0)` as synchronous reset. Replace the `dac` output multiplexer as follows.

SW17	**SW16**	**dac <=**	**Comment**
0	0	`sum`	Display filter input
0	1	`f5/1024`	Display half-band filter output scaled
1	0	`mean`	Display mean output
1	1	`median`	Display median output

Depending on how you defined the signals `f5, mean`, and `median` you may need to use the `CONV_STD_LOGIC_VECTOR(…,16)` function since `dac` is a `STD_LOGIC_VECTOR` type.

5. Now run a full compilation. From the **Compilation Report** and the **Classic Timing Analyzer Tool** (**Processing** menu) for `clock_50` determine

Total logic elements = _____

Embedded Multiplier 9-bit elements = _____

Total memory bits = _____

Registered Performance = _____ MHz

Enter these resource and performance data also in your `DE2_lab5` VHDL header file.

6. Now download the `DE2_lab5.sop` file to the board and observe with oscilloscope or head set. The 1 kHz (always on, i.e., `SW0=1`) should be corrupted by random, impulse, or harmonic noise. Give a grade ("A" works good; "B" some improvement; "C" no improvement), how good the filter works for the noise in question. Complete the table below for your results. If not specified, set the switch SW to zero, i.e., "south."

1kHz sine (SW0=1) +		**SW17=0;SW16=1 Half-band filter**	**SW17=1;SW16=0 Mean filter**	**SW17=1;SW16=1 Median filter**
random noise (SW1=1)				
impulse noise (SW2=1)				
harmonic noise (SW3=1)				

F. Deliverables:

1. Solve the problems of the pre-lab. (3 points).

2. Complete this report, print the VHDL files for `f5direct.vhd, fir.vhd`, and `DE2_lab5.vhd`, and the Quartus simulations for f5direct (see Fig. 1) and the 3 FIR filters as in Fig. 2. (7 points).

Make sure your name and SS is on all pages you turn in!

