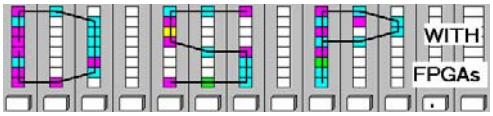
Lastname: _____

5Digit SS:_____

LABORATORY
IIR Filter



LAB IIR: INTRODUCTION TO IIR FILTER
(10 points)

In this lab, you will be introduced to the design of Infinite Impulse Response (IIR) filters. Filters are one of the most important elements in DSP and are typically used to isolate a specific frequency band of a signal. IIR filters are of particular interest because, with just a few coefficients, relatively sharp transition bands can be realized.

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 design a first-order IIR filter and a third order system direct form filter.

Lab Objectives

After completing this lab you should be able to

- Design and simulate an first order IIR filter
- Compare IIR and FIR design parameter
- Determine magnitude, phase and pole zero diagram of IIR filters
- Design a 3. order elliptic low pass filters

Pre-lab (3 points)

1. For a first order IIR filter with a transfer function $F(z)=b/(1+az^{-1})$, determine a and b such that the filter is a halfband filter (i.e., $|F(\omega=0)| = 1$ and $|F(\omega=\pi/2)| = 0.5$).

Feedback gain a = _____

Forward gain b = _____

Hint: The quadratic equation $x^2+px+q=0$ has the solution $x_{1,2}=-p/2+/-\text{sqrt}((p/2)^2-q)$.

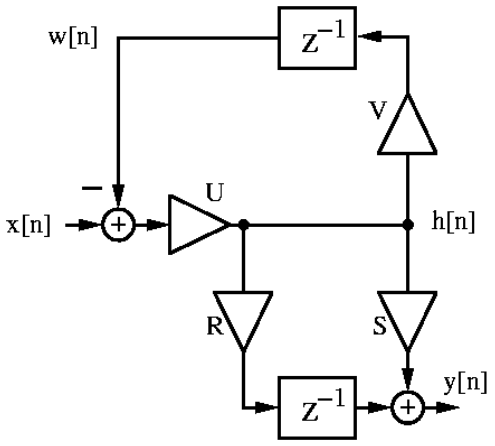
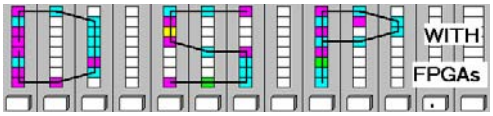


Figure 1

Lastname: _____

5Digit SS:_____

LABORATORY
IIR Filter



2. Determine the transfer function $F(z) = Y(z)/X(z)$ for the system in Figure 1. Compute the intermediate node $H(z)$ and $Y(z)$ first.

   intermediate node: $H(z)$=

   $Y(z)$ =

   $F(z) = Y(z)/X(z)$ =

3. Determine zero(s) and pole(s) of the system in terms of the coefficients U,V,R, and S.

   Zero(s) at = _____ Pole(s) at = _____

4. Find the values of the coefficients (U, V, R, S) so that the transfer function $H(z)$ from part 1 is realized. If possible use U=1.

   U = 1

   V =

   R =

   S =

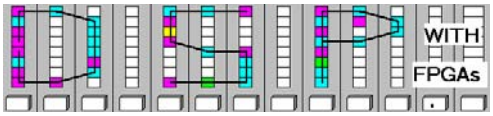
5. Compare the FIR and IIR filters regarding the following properties:

	FIR filter	**IIR filter**
Filter Length		
Filter Linearity		
Coefficient design method		
Pole locations for stable filter		
Coefficient sensitivity to quantization		

Lastname: _____

5Digit SS:_____

LABORATORY
IIR Filter



VHDL Design-lab (7 points)

Follow the directions below to implement a first and third-order IIR filter.

A. Getting Started

If you are in B114 or the digital logic lab:

1. On the desktop double click on **Engineering folder**.
2. Double click on the **MatLab** icon  to start **MatLab**.
3. 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 a New Folder named **DSPwFPGAs** (if you have not so in a previous lab) on your mapped network drive.

B. Develop and Simulate an First-Order IIR Design

1. At the MatLab prompt, use the predefined functions `freqz()` and `zplane()` to plot the frequency spectrum and pole/zero plot, respectively. First, you must define the numerator and denominator polynomial coefficients in vector form.

   - Recall the transfer function you found in the pre-lab, of form
     $F[z] = b[z]/a[z] = (b[0]+b[1]z^{-1}+...) / (a[0]+a[1]z^{-1}+...)$
   - **Use your coefficient values** to define the coefficient vectors. At the MatLab prompt, type the following:

     >> **b=[b[0], b[1], ...]; a=[a[0], a[1], ...];**

   - Now, call the functions to generate the desired plots:

     ```
     >> freqz(b,a)
     >> zplane(b,a)
     ```

2. Complete the following diagrams:

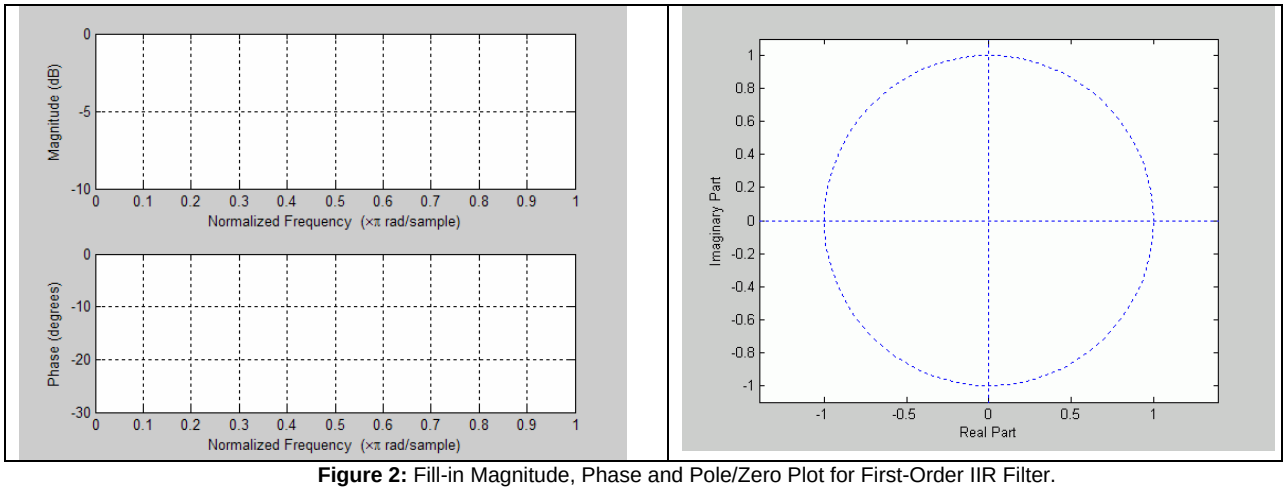
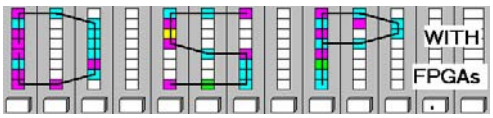


Figure 2: Fill-in Magnitude, Phase and Pole/Zero Plot for First-Order IIR Filter.



3. Download the file `csd3e.exe` from the class webpage or the book CD and put the file in the **DSPwFPGAs** folder.

4. Using the `csd3e.exe` program determine for 2, 3 and 4 operations the CSD coding and completing the following table. Note that the `csd3e.exe` program only works for positive numbers.

Name	Value	Number of Operands	CSD coding	Effective Bits
-a[1]		**2**		
-a[1]		**3**		
-a[1]		**4**		
b[0]		**2**		
b[0]		**3**		
b[0]		**4**		

5. 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.

6. Now develop the VHDL code `iir_order1.vhd` for the first order filter from Fig. 1 in the prelab. Start a new project `iir_order1` and use as data type the signed 16 bit integer type as in lab 5. Use the CSD code with at least 10 bit precision for the constant coefficients. Try to include an intermediate port w as in Fig. 1 and match the following response to an impulse of 1000. Note that y[0] shows b[0] and w[1] displays the feedback coefficient a[1] (multiplied by 1000). Add your e-signature and print this simulation in landscape format.

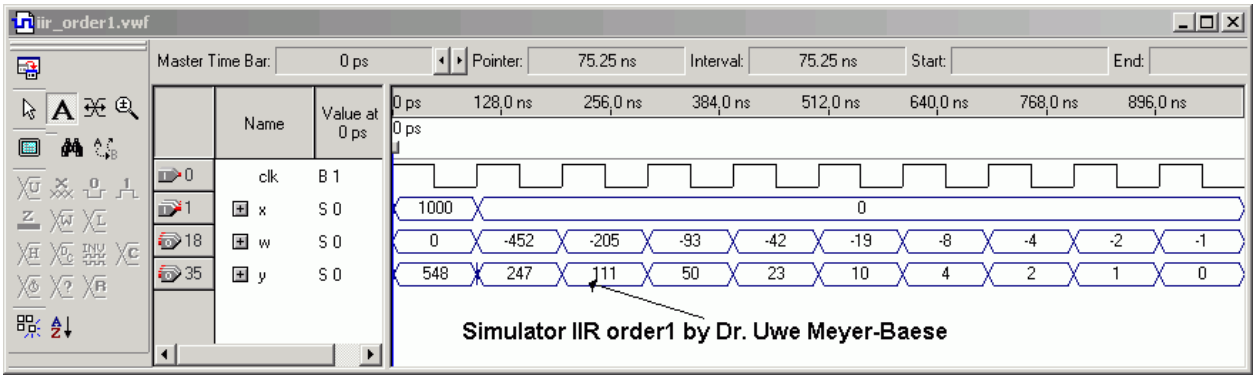


Figure 3: First-Order IIR Filter desired simulation.

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

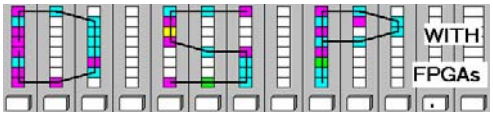
Total logic elements = _____

Embedded Multiplier 9-bit elements = _____

Total memory bits = _____

Registered Performance = _____ MHz

Enter these resource and performance data also in your `iir_order1.vhd` header file.



C. Designing a 3rd Order Direct-Form Filter

1. In this part of the lab a third order IIR elliptic filter is designed as shown in Figure 4. A switch is used in the design process that we can clearly see the forward (via y[n]) and the feedback (via w[n]) coefficients in the simulation.

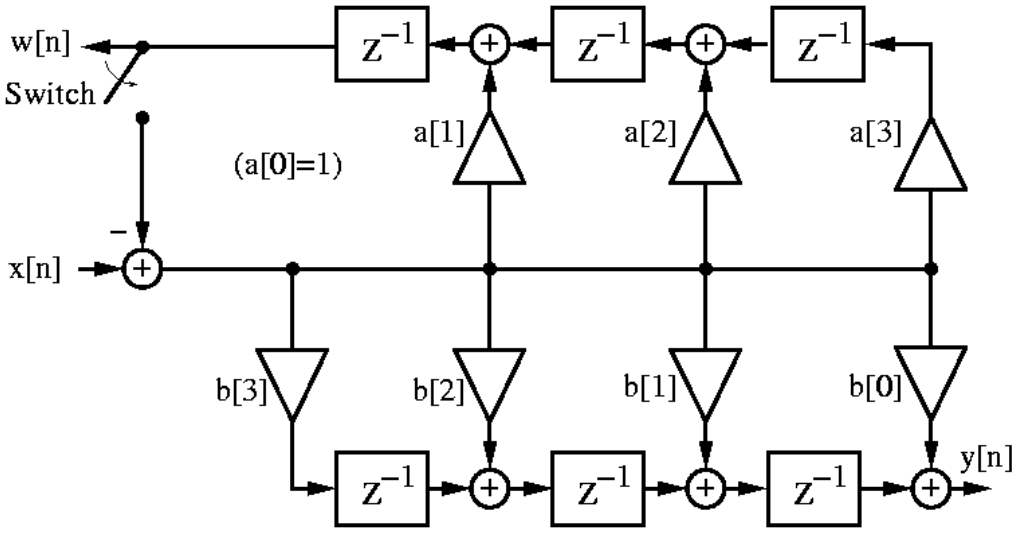


Figure 4: Third-Order IIR Filter.

2. Download `setup_iir3.m` from the class webpage and put the file in the **DSPwFPGAs** folder.

3. Open the file `setup_iir.m` with a text editor. There you will find the filter coefficient, an impulse response computation, and the call to the predefined MatLab function `freqz()` and `zplane()` for the spectrum and pole/zero plot, respectively. In the MatLab prompt, type `setup_iir3` and complete the following diagrams:

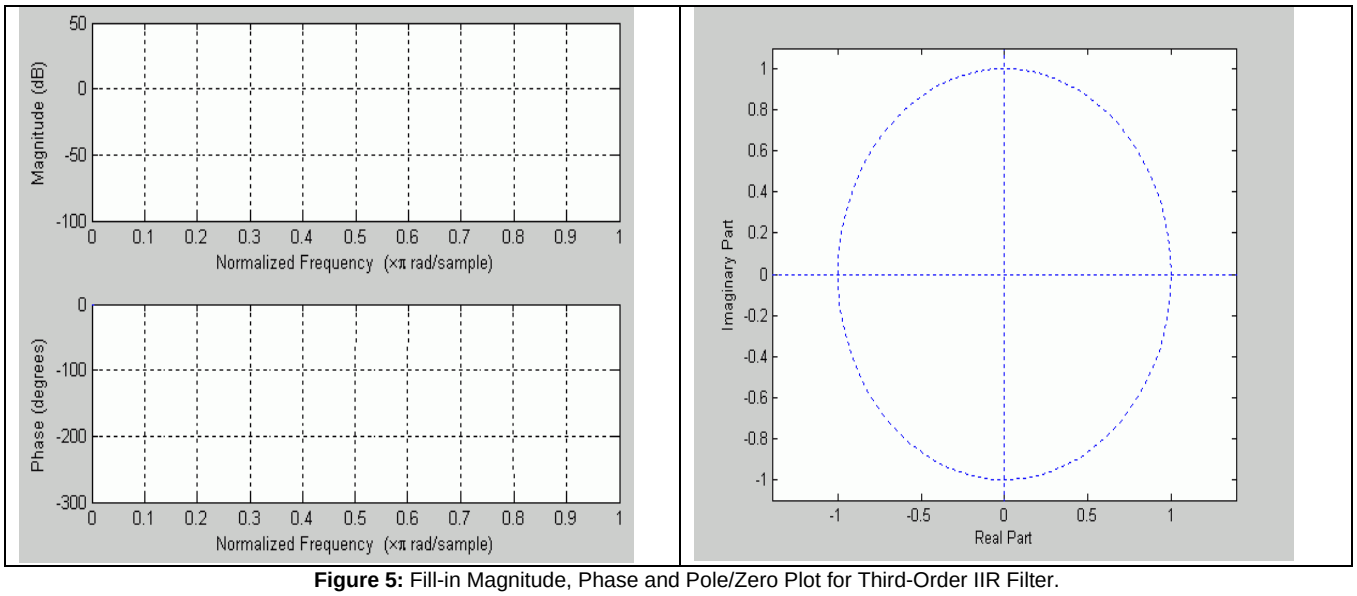
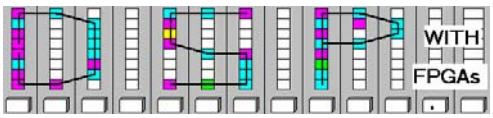


Figure 5: Fill-in Magnitude, Phase and Pole/Zero Plot for Third-Order IIR Filter.



4. Using `csd3e.exe` determine for the CSD coding such that each coefficient has at least **8 bits effective bits** accuracy. Complete the following table with the results.

Name	Value	CSD coding	Number of Operands	Effective Bits
-a[1]	**1.5921**			
a[2]	**1.3872**			
-a[3]	**0.5220**			
b[0]=b[3]	**0.0429**			
b[1]=b[2]	**0.0935**			

5. 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.

6. Develop the `iir_order3.vhd` design. Use a new project `iir_order3`, but the same entity ports and data types as for the first order design. To verify your coefficient coding use the simulation with open switch and match Figure 6. The forward coefficient scaled by 1000 appear at the output y[n] and the feedback coefficients at w[n].

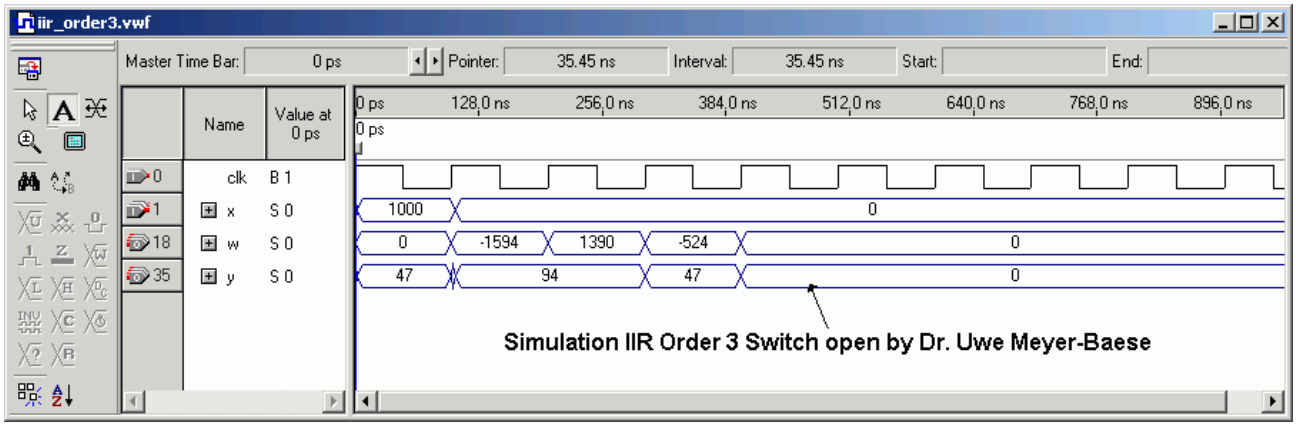


Figure 6: Third-Order IIR Filter simulation with open switch.

7. Now simulated the system with the switch closed and you should get the results from Figure 7 below. Add your e-signature and print this simulation in landscape format.

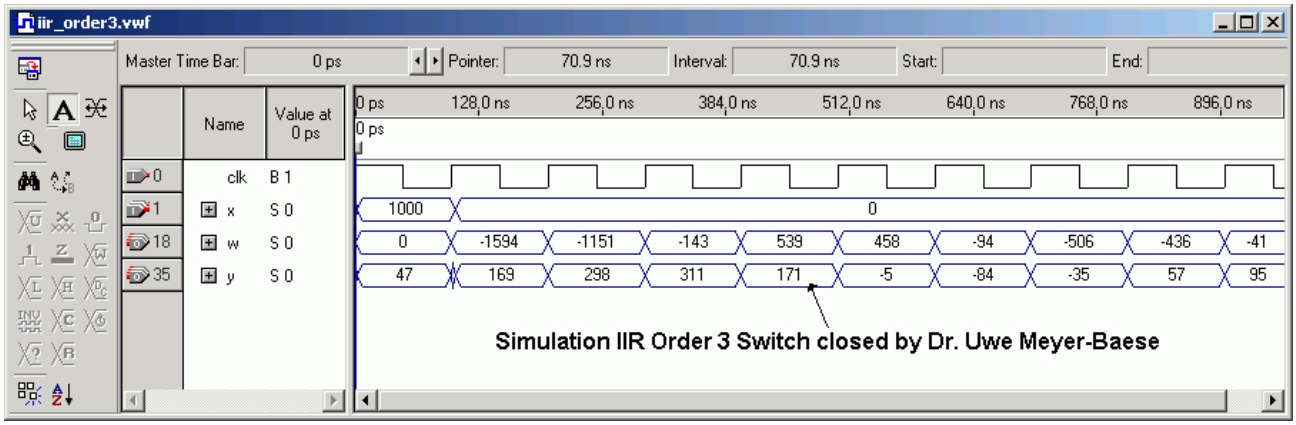


Figure 7: Third-Order IIR Filter simulation with closed switch.

8. 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 `iir_order3.vhd` header file.

D. DE2 Implementation of the IIR Designs

1. Start Quartus II and build a new project called DE2_lab6. 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_lab5 folder into a new DE2_lab6 folder the following files: `DE2_lab5.vhd`, `audio_pll.vhd`, `audio_pll.qip`, `audio_pll.cmp`, `i2c_config.vhd`, `i2c_controller.vhd`, `audio_dac.vhd`, and all `lut*.mif` files. Rename the HDL file `DE2_lab5.vhd` to `DE2_lab6.vhd`.

3. Remove the `fir` component from `DE2_lab6.vhd` and define in the signal section and instantiate `iir_order1` and `iir_order3` as a component into the `DE2_lab6.vhd` as follows:

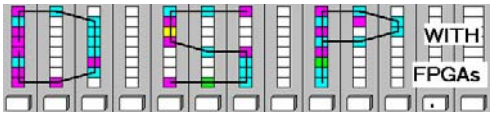
```
C4: iir_order1 PORT MAP (clk=>DACLRCK, x=>x, w=>w1, y=>y1);

C5: iir_order3 PORT MAP (clk=>DACLRCK, x=>x, w=>w3, y=>y3);
```

   Define the required signals in the correct width. For convenience copy the definitions for `S16` data type from the `iir_order1.vhd` or `fir.vhd` file and place it in the signal definition section of `DE2_lab6.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	`y1`	Display first order IIR output
1	0	`y3`	Display third order IIR output

   Depending on how you defined the signals `y1` and `y3` 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 in your `DE2_lab6` VHDL header file.

6. Now download the `DE2_lab6.sop` file to the board and observe with oscilloscope or head set. The 1 kHz signal should always be on, i.e., `SW0=1`). It should be corrupted by harmonic noise (`SW3=1`). Give a grade ("A" works good; "B" works ok; "C" not working), how good the filter works removing the harmonic noise and keeping the gain at one, i.e., sine amplitude does not change. Complete the table below for your results. If not specified, set the switch SW to zero, i.e., "south."

1kHz sine (`SW0=1`) + harmonic noise (`SW3=1`)		**`SW17=0;SW16=1` First order filter**	**`SW17=1;SW16=0` Third order filter**
Gain at one			
Improvement noise			

E. Deliverables:

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

2. Complete this report; print the 3 VHDL files and the 2 simulations for the two IIR filters (7 points).

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

