The University of Queensland
School of Information Technology and Electrical Engineering

ELEC3004/7312: Signals, Systems and Controls

EXPERIMENT 3: ECHO FILTERS ON THE NEXYS 2

Aims

In this laboratory session you will:

1. Use VHDL for implementing a simple “echo” filter via a difference equation.
2. Compare IIR and FIR filters
3. Study comb filters
4. Have some fun with echo and reverb!

Introduction

Two common digital filter types are IIR and FIR, each of which has advantages and disadvantages. Filters are available in low pass, high pass, bandpass, notch etc, and can have different responses such as Chebyshev, Elliptic, etc. Another type of filter is the comb filter which works by combining an input signal with a related signal which forms constructive and destructive interference. The two options we will explore involve combining the input signal with a delayed version of either the input (FIR), or the output (IIR). When constructive interference occurs, the output signal reaches a maximum. Conversely, destructive interference causes the output signal to reach a minimum, sometimes reaching zero. The depth of the notch depends on the relationship between maximum and minimum output values. The spacing between the notches depends on the sampling frequency and the delay.

Preparation

Note: preparation will be checked at the start of each laboratory class.

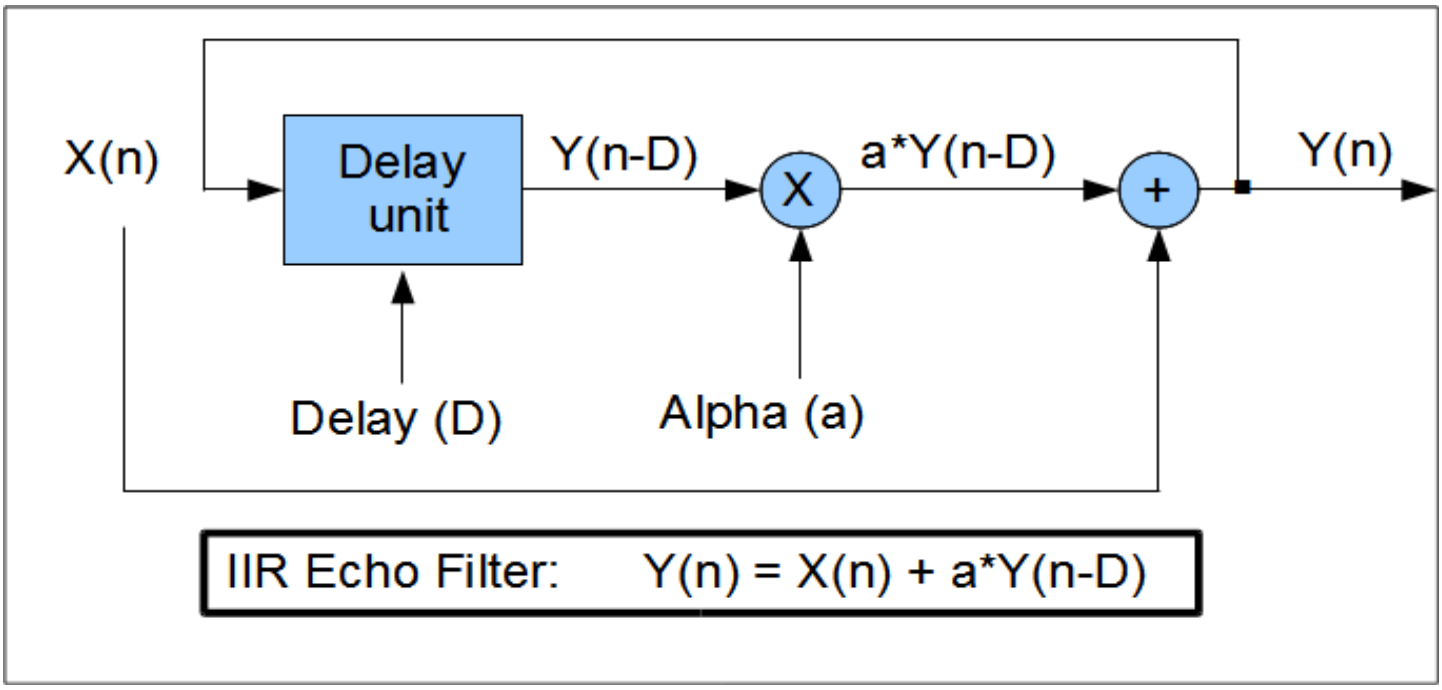


Figure 1: IIR Echo Filter block diagram

Figure 1 shows a block diagram of an IIR Echo Filter consisting of a memory (Delay Unit), a multiplier, adder and feedback path.

Alpha (α) is a value between 0.0 and 1.0, D (delta) is the delay value in samples.

Answer the following questions:

1. A simple echo filter can be defined by the difference equation: $y[n] = x[n] + \alpha\, x[n - \Delta]$, where $y[n]$ is the output, $x[n]$ is the input of the filter, α is the amplitude of the echo and Δ is the echo delay in samples, n. Clearly, this is an FIR filter (the equation does not have any feedback from the output.)
    a) Implement this difference equation in Matlab and confirm its operation with Handel's hallelujah chorus (see *help sound*). Note, to hear an echo try setting Δ to equal the number of samples in ~100ms, i.e., the sampling frequency divided by 10. You can use a *for* loop to implement this.
    b) Using *zplane* command, sketch the pole and zero locations in the *z*-plane for $\alpha = 0.8$ and $\Delta = 6$;
    c) Using *freqz,* sketch $|H(\omega)|$, i.e., the magnitude response of this filter from zero to half the sampling frequency. Alternatively, you could also use the *fvtool* command
    (We will be using this command in lab 4 so please familiarise yourself if you can)
    d) What effects do α and Δ have on the shape and number of peaks in the magnitude response?
    e) Suggest how a "fading" echo might be implemented by changing the original FIR filter to be IIR.
    What role do the poles and zeros now play? Confirm your design by implementing this filter in Matlab. *(Hint: IIR filters also have feedback from the output)*

2. Redraw Figure 1 as follows. Delete the feedback connection from Y(n) to the delay unit. Now connect X(n) to the input of the delay unit as well as the adder. Replace the Y(n-D) text with X(n-D). Write down the difference equation. What type of filter is this?

3. Now redraw Figure 1 as follows: Place a 2-position switch at the input of the delay unit so the centre connection goes to the delay unit. Now connect the other two lines, one of which is a connection to Y(n), and the other joins to X(n) as well as X(n)'s connection to the adder. Place the text Z(n) at the input to the delay unit, and replace the Y(n-D) text with Z(n-D). Write down the difference equation. You will use this setup in Part 3.

4. With reference to http://en.wikipedia.org/wiki/Comb_filter write the two equations for Continuous-time comb filters

Equipment

1. PC with Xilinx ISE, Digilent Adept & Matlab;
2. PMOD AD1 and DA2 boards plus 2 PMOD CON4 boards
3. ADC712 and DAC712 Filter boards (fit between ADC/DAC and PMOD CON4)
4. Nexys 2 + JTAG interface cable/s;
5. Oscilloscope (preferably TDS1002);
6. 2 x cable: mono RCA male to mono BNC male, 0.5 - 1 metre long
7. Mono or stereo 3.5mm male to mono or stereo RCA male
8. Mono or stereo Y-adaptor, 3.5mm Male to 2 x 3.5mm Female
9. Signal Generator;
10. External speakers + audio jack cable + power pack;
11. 1 BNC T-adaptor M to 2F (F-M-F);
12. 1 x cable BNC Male to BNC Male, 0.5 - 1 metre long.

Procedure

Part 1: IIR Echo filter

With reference to EXPERIMENT 1: INTRODUCTION TO THE Nexys2, carry out the following:

1. Start Xilinx ISE and Digilent Adept
2. Connect PMOD DA2 to JB, and connect PMOD AD1 to JC, both on the top row, then connect a PMOD CON4 to each**. If you have the filter boards, insert them between the PMODCON4 boards and each converter PMOD.**
3. Using an appropriate cable and T-piece combination, connect the output of the signal generator to PMOD AD1 and to channel 1 of the oscilloscope.
4. Connect the output of PMOD DA2 to channel 2 of the oscilloscope.
5. Set CH1 and CH2 of oscilloscope to DC Coupling from the CH1/CH2 Menu buttons.
6. Pull the Offset button out and set the Amplitude control counter-clockwise and select SINE wave.
7. Open the FPGA Project: Prac3_Part1_2012.xise (download from ELEC3004 website)
8. Move all slideswitches to the off position (edge of the board).

9. You will need to experiment with offset voltage and amplitude on the signal generator, so that you don't get a distorted output. Suitable starting values would be approx 2.5 volts DC for **Offset**, then **Amplitude** of about 2.5 Vp-p.

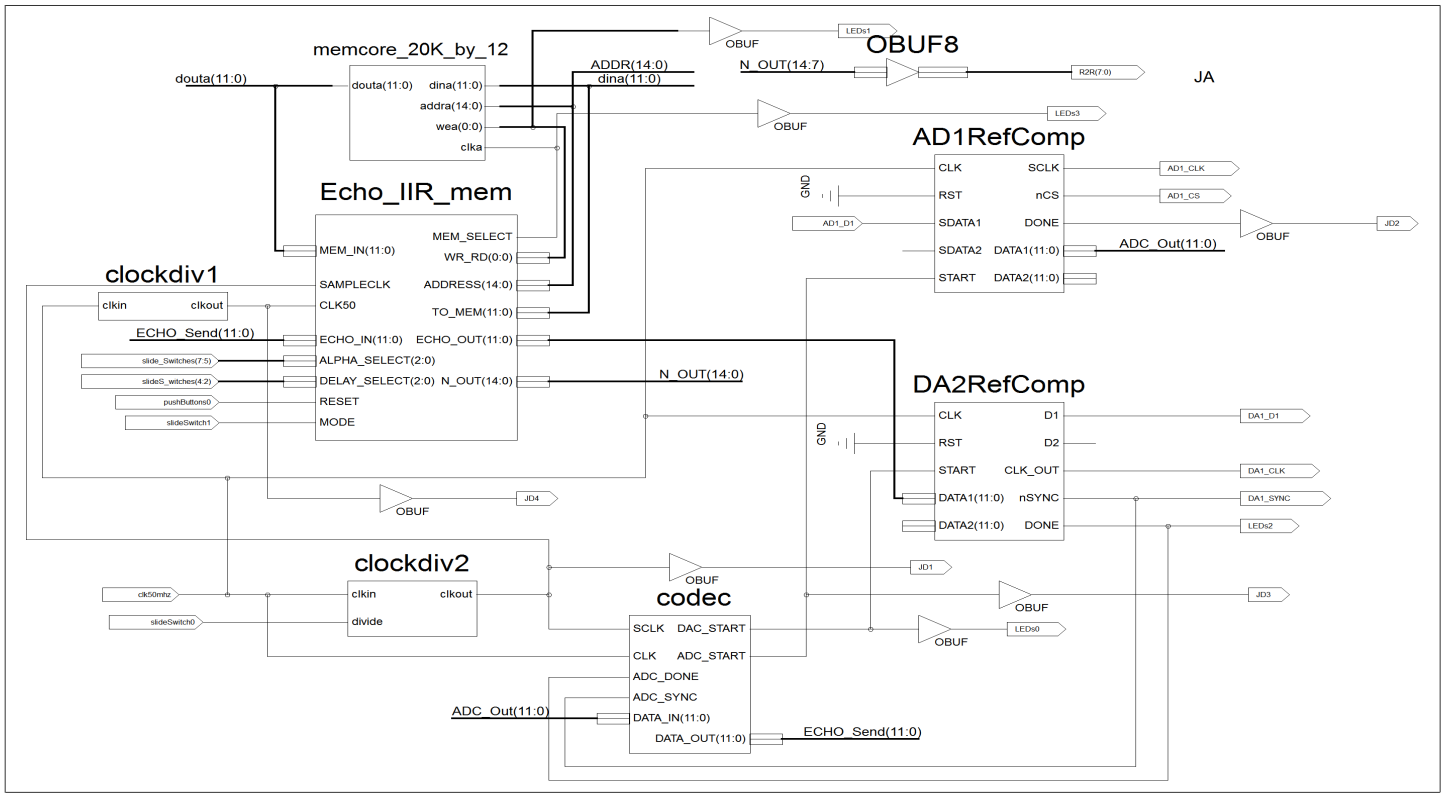
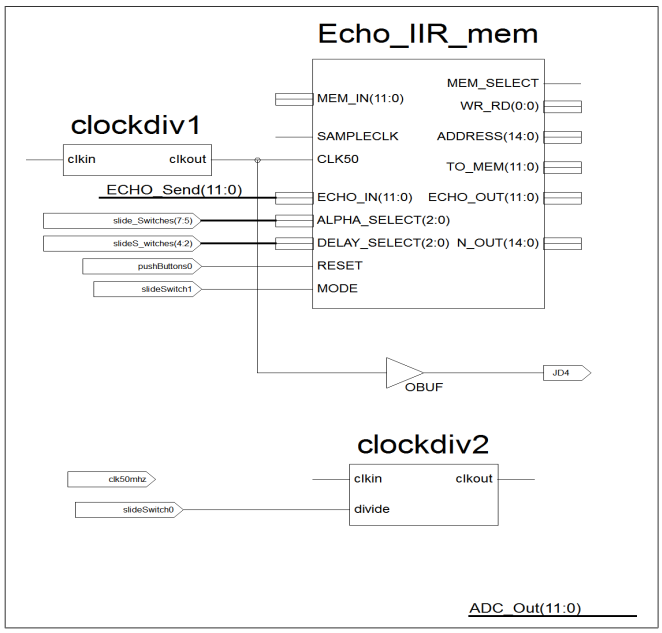
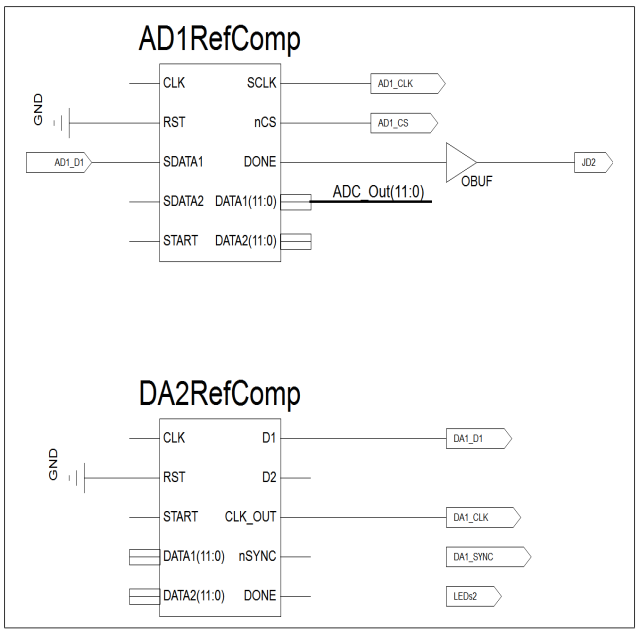


Figure 2. IIR Filter schematic

If you get a message saying that a project file is missing, it could be because the path for the Project file is different to where you are using your files currently. If so, make sure all the required files are in your local project directory, then **Add Source** from the **Project** menu.





The IIR filter module **The ADC and DAC modules**

In **Echo_IIR_mem.vhd** you will see some text that looks like this:

constant MaxDelay : integer range 0 to 19999:=19999; --20000 12-bit Words is limit of FPGA
signal N : integer range 0 to MaxDelay; --Counts current sample
signal D : integer range 0 to MaxDelay; -- Requested delay time in samples
signal Delta : integer range 0 to MaxDelay;

MaxDelay sets the amount of memory reserved in the Nexys2 FPGA. The memory is arranged as a circular file so that the last sample location exists just before the first sample location. This means that sample location 19 999 is just prior to sample location 0. **N** is the counter that controls where the current sample is stored.

D is the number of samples used for the echo delay and is related to the echo time by the sampling rate F_s. For example, if F_s = 25000 samples/second, then a delay of 1 second would require a D = 25000. As there are limited memory resources on the FPGA, this is limited to 20000 samples, or (20000/25000 = **0.8**) seconds of delay. It is then a simple process to calculate the number of samples required for a delay of 0.2 seconds, for instance.

This value is then used for the value of D, which cannot exceed the value of MaxDelay. If you want a longer delay, you will need to reduce sampling frequency **Fs**, which can be done by increasing the value of Q2 that is set in **clock_div2.vhd** and/or **using slideSwitch0.**

Of course reducing F_s will reduce your Nyquist frequency F_c, and introduce more aliasing errors. Before digital delay units became affordable, analogue systems used Bucket Brigade Devices like the SAD1024. See http://en.wikipedia.org/wiki/Bucket-brigade_device for an example, where the sampling frequency (and the maximum input frequency) was controlled by a high speed clock and the data was stored in capacitors.

The IIR echo filter equation can be expressed as ***Y(n) = X(n) + alpha * Y(n-D)*** where *Y(n)* is the current output, *X(n)* is the current input sample, *Y(n-d)* is the delayed sample stored in memory, and **alpha** is a value between 0.0 and 1.0. If alpha = 0, then there is no echo and the output is the same as the input, ie ***Y(n) = X(n)***. Increasing alpha towards 1.0 increases the amplitude of the echo, ***Y(n-D),*** in the output signal.
Here **alpha** is a floating-point number, but we can only use **fixed-point** values. This is accomplished using a rational fraction like a/b. If alpha is *too* big, you can get distortion in the output signal due to overflow errors.

You will see some code that looks like this:

```
ALPHA1 <= to_INTEGER(unsigned(ALPHA_SELECT));
C_INT <= (ALPHA1 *B_INT)/ 8;  --alpha = eg 0.5 = (4 *B_INT)/ 8
```

Our code allows the use of values 0/8 to 7/8 for alpha, and MaxDelay * 0/8 to 7/8.

For this part of the experiment, you can adjust the value of *alpha* and *D* by changing the settings of **slideSwitches** 7-5 for Alpha, and **slideSwitches** 4-2 for Delay.

NOTE: Do not connect cables from the PC to PMOD AD1/CON4 if you don't have the filter board. Firstly, some Nexys2 boards have been destroyed by dodgy connections. Secondly, you will only hear the positive-going part of the waveform, which will then sound distorted.

1. Set the function generator to the 10 kHz range and repeatedly sweep from about 100 Hz to about 30 kHz, whilst experimenting with various values of alpha and D.
2. Repeat this using *TheForce.wav* continuously whilst experimenting

Note: you can use the Windows Media Player to continuously play *TheForce.wav* after connecting the headphone jack on the PC to the PMOD AD1/ADC712 Filter/ CON4 combination using the 3.5mm male to RCA male cable. The output can be heard on the speakers by using a PMOD_AMP and small speaker, or PMOD_CON4 and RCA to 3.5mm adaptor with powered speaker. The tutors will help with any connection issues.

Part 2: FIR Echo Filter

The FIR echo filter equation can be expressed as *Y(n) = X(n) + alpha * X(n-D).*

- *X(n-D)* is the delayed sample stored in memory.

Find the line of text in Echo_IIR_mem.vhd which looks like

"---------- Store values to memory here ----------"

1. Modify Echo_IIR.VHD so that X, rather than Y, is stored to memory,
2. Set values for D and alpha as for the IIR filter;
3. **Generate programming File** and download as before
4. Play TheForce.wav continuously as before;
5. Alternatively, try using the function generator to sweep the frequency band by hand – attempt this both slowly and quickly, using the 1kHz, 10kHz and 100kHz ranges in turn ;
6. What differences do you observe between the IIR and the FIR filters?

Part 3: Selectable FIR/IIR mode
Refer to your answer to question 3 in the preparation

1. Modify **Echo_IIR_mem.vhd** so that you can quickly switch between IIR and FIR filter mode, using **slideSwitch1.** ***Hint: Look for code using signal Z and function Z_Val.***

2. Play around with various frequencies on the function generator. Try various combinations of Delay and Alpha values and compare the different outputs that you hear, as you switch between IIR and FIR mode. Using alpha = 6/8, how many echoes can you hear for each of the FIR and IIR modes?

3. Try changing between sample rates (25 kHz and 10 kHz) using **slideSwitch0**. You will notice a change in echo times, but what other things are happening to the output signal?

Part 4: Comb Filters

From your work on the preparation questions, you will have looked at comb filters. In this part, you will slowly change the frequency on the function generator, over a limited range, in order to see the shape of a comb filter. The shape will depend on the sample rate, the feedback amount (alpha) and the delay value chosen. A small alpha will give shallow troughs, whereas a large alpha will cause deep troughs.

Set the function generator on the **OSCILLOSCOPE** to **Frequency Fine**, then very slowly sweep the frequency from about 30 Hz up to 100 Hz. Just use enough of the range for you to establish the distance, in Hz, between peaks or troughs, as well as the maximum and minimum values at those frequencies.

Using your modified code from part 3 will allow you to quickly switch between IIR and FIR mode, as well as changing the sample rate from 25 kHz to 10 kHz.

1. **With all slideswitches set to off (zero) (FIR mode)**, sweep the function generator from 30 Hz to about 300 Hz and confirm that the output amplitude is constant over the range.

2. **Set alpha value to 1/8 and delay to 1/8 (this is 2500 samples).** This will give the minimum feedback amount and shortest delay. You will see a variation in amplitude at certain frequencies due to constructive and destructive interference. Now sweep the function generator from about 30 Hz to about 100 Hz, or enough for you take enough measurements to establish the following:

   The frequencies at which maxima and minima occur, as well as the amplitude at those frequencies, for example, 47Hz = 2.5Vpp, 55Hz = 1.8Vpp, 63Hz = 2.5Vpp etc

   Estimate the spacing between peaks (or troughs)

3. **Leave alpha set to 1/8** and repeat 2. for Delay values of 2/8, 3/8 and 4/8

4. **Set alpha to 7/8** and repeat 3.This will show the maximum trough depth.

5. Referring to the comb filter diagrams at http://en.wikipedia.org/wiki/Comb_filter , write equations to show the frequency spacing relationship due to the sample rate and the delay value (no. of samples), as well as the relationship between alpha value and trough depth.

6. **Move slideswitch1 to 1 (IIR mode), set alpha to 7/8, delay at 1/8** then sweep from 30 Hz to 300Hz and observe whether the IIR filter settles as quickly as the FIR filter. No measurements have to be recorded

7. Now repeat 5., but with **delay set to 6/8**.

END of Experiment