**LAB 7 (OMAP-L138)**

**Finite Impulse Response System (*FIR*)**

**objectives**

* Design a FIR lowpass filter
* Program the OMAP-L138 LCDK processor to implement FIR low-pass filter

**Introduction**

A finite impulse response system is a system with a difference equation that has this form

*y*[*n*] = a0*x*[*n*] + a1*x*[*n*-1] + a2*x*[*n*-2] +…+ aN*x*[*n*-*N*]. 🡪 1

The output of this system depends on the current and previous inputs. It does not depend on previous output such as *y*[*n*-1], *y*[*n*-2], and so forth. This system is of *N*th order. The impulse response of *FIR* filter usually consists of finite length, *N*. The impulse response *h*[*n*] of a *FIR* system can be found from the difference equation by taking the discrete time Fourier transform (DTFT) as follow

*Y*(Ω) = a0 *X*(Ω) + a1 *X*(Ω)e-jΩ + a2 *X*(Ω)e-j2Ω +…+ aN *X*(Ω)e-jNΩ.

*H*(Ω) = *Y*(Ω) / *X*(Ω) = a0 + a1 e-jΩ + a2 e-j2Ω +…+ aN e-jNΩ.

The impulse response *h*[*n*] can be found by taking the inverse DTFT of *H*(Ω) in the above equation.

*h*[*n*] = a0*δ* [*n*] + a1*δ* [*n*-1] + a2*δ* [*n*-2] +…+ aN*δ* [*n*-N].

The impulse response *h*[*n*] can also be found by replacing the input *x*[*n*] in Eq. 1 above by the impulse input *δ* [*n*] and the output *y*[*n*] will be *h*[*n*].

When we program the DSP to be a FIR, the impulse response is defined as one dimensional array *h* = [ a0 a1 a2 … aN] .

The output *y*[*n*] is calculated by the linear convolution of the input *x*[*n*] with the impulse response *h*[*n*], as follow

*y*[*n*] = *h*[0]*x*[*n*] + *h*[1]*x*[*n*-1] + *h*[2]*x*[*n*-2] + … +*h*[*N*]*x*[*n* - *N*]

For a causal input with a 3rd order filter (*N* = 3), a sample calculation of the output will be as follow. Remember causal input means *x*[*n*] = 0 for *n* < 0.

*y*[0] = *h*[0]*x*[0] + *h*[1]*x*[-1] + *h*[2]*x*[-2] +*h*[3]*x*[ - 3]

*y*[0] = *h*[0]*x*[0]

*y*[1] = *h*[0]*x*[1] + *h*[1]*x*[0]

*y*[2] = *h*[0]*x*[2] + *h*[1]*x*[1] + *h*[2]*x*[0]

*y*[3] = *h*[0]*x*[3] + *h*[1]*x*[2] + *h*[2]*x*[1] +*h*[3]*x*[0]

*y*[4] = *h*[0]*x*[4] + *h*[1]*x*[3] + *h*[2]*x*[2] +*h*[3]*x*[1]

*y*[5] = *h*[0]*x*[5] + *h*[1]*x*[4] + *h*[2]*x*[3] +*h*[3]*x*[2]

The last three lines are considered a steady state response because there is full overlap between the input *x*[*n*] and the impulse response *h*[*n*].

**Prelab Exercises**

For the system, whose impulse response *h*[*n*] is shown below in Fig. 1, find the following:

1. The frequency response *H*(Ω) in terms of *A* and *N*.
2. Plot the magnitude and the phase of the frequency response *H*(Ω) for -3π < Ω < 3π. You can set *A* = 1 and *N* = 5.
3. What type of filter is this system? What is the cutoff frequency Ωc? Express the cutoff frequency in terms of *N*. You can consider the cutoff frequency as the first frequency at which | *H*(Ω) | = 0.
4. If the sampling rate was 16 kHz, then what is the cutoff frequency *f*c in Hz?

1

2

3

*N*

*n*

*h*[*n*]

*A*

**Fig. 1**

**lab Exercises**

**Task 1: Matlab Simulation of the FIR Filter**

1. Design a digital FIR lowpass filter with cutoff frequency 4000 Hz and the frequency response for DC is |*H*(0)| = 1 and at the cutoff frequency H(Ωc) = 0. Set the sampling rate of the DSP to be 16000 Hz. For your design you need to figure out first the order of the filter, value of *N*, and the amplitude *A* of the impulse response *h*[*n*] to meet the specified parameters of the lowpass filter.
2. Plot the magnitude |*H*(Ω)| vs Ω for the range -π < Ω < π. What is the cutoff frequency Ωc?
3. To test your design run a simulation in Matlab by using the “filter” function in Matlab.

*Filter\_output = filter (coefficients in numerator, coefficients in denominator, input of the filter)*

You can download the audio materials sentence16k.wav and RingFire16k.wav from the course website. Filter the audio materials using the filter you designed and compare the input and output signal quality.

1. Add to the sentence a tone of frequency 2000. Can you hear the 2000 Hz tone?
2. Repeat step 4 by adding separately to the sentence a tone of different frequencies such as 4000, 6000, and 8000Hz. Which of these frequencies you still can hear and how loud they are at the output of the filter in comparison to the 2000 Hz, and which ones you don’t hear? Explain your answer.

**Task 2: Implement the FIR Filter on the LCDK DSP Board**

1. Program the DSP to be the lowpass filter designed in step 1. The main challenge is to code the convolution operation in *C* language such that the convolution is carried out in real-time.
2. Build and load the code to the LCDK processor.
3. After a successful design, coding, and loading, test your filter by applying audio materials such as music and speech from Matlab or YouTube to the audio input of the LCDK and listen to the audio materials from the audio output of the LCDK. Compare the quality of the original speech and music (the input to the LCDK) to the filtered speech and music (the output of the LCDK).
4. Add to the sentence audio file a tone of frequency 2000 in Matlab and play the sentence with the noise through the LCDK. Do you hear any tone?
5. Repeat step 5 with tone noise of 4000 Hz, 6000 Hz, and 8000 Hz. Which tone do you still hear, and which tone you do not hear any more? Explain your answer.

Call your instructor to verify the output with inputs of different tones: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Hint:** If the filter order is *N* = 3 then define an impulse response variable of length 4 and call it *h*. Define an input array of length 4 and call it *x*. Let us define *x*[0] as the current input, *x*[1] as the input one sampling interval earlier, *x*[2] the input two sampling interval earlier, and so forth. Every time an input comes to the DSP board, the Interrupt Service Routine (ISRs.) will be called and the ISRs function should carry out the following two steps before the next input arrives, see diagram below.

**Step 1**: Assign the current input to *x*[0] and start calculating the current output using convolution as follow *y* = *x*[0]*h*[0] + *x*[1]*h*[1] + *x*[2]*h*[2] + *x*[3]*h*[3]. Remember *x*[3] means an old input that is three sampling interval earlier.

**Step 2**: The next step is to shift all input data one unit to the right so *x*[0] will be available to store the future input when it arrives to the DSP. This is accomplished by first shifting *x*[2] to be stored in *x*[3]. The old *x*[3] will be lost but that is okay because we will not need it to calculate the future output. The next step is to shift *x*[1] to be stored in *x*[2], and the last step is to shift *x*[0] to be stored in *x*[1]. Now *x*[0] will be ready to store the future input.

x[0]

x[1]

x[2]

x[3]

h[0]

h[1]

h[2]

h[3]

3

-2

1

4

6

3

-1

0

0

1

3

2

3

0

0

0

First call

to the ISR

x[0]

x[1]

x[2]

x[3]

0

0

0

0

Initial values of arrays before

input hits the DSP board.

x[0]

x[1]

x[2]

x[3]

3

-2

1

4

6

3

-1

0

-2

3

0

0

Second call to the ISR

x[0]

x[1]

x[2]

x[3]

3

-2

1

4

6

3

-1

0

1

-2

3

0

Third call to the ISR

x[0]

x[1]

x[2]

x[3]

3

-2

1

4

6

3

-1

0

4

1

-2

3

Fourth call to the ISR

Input

x[0]

x[1]

x[2]

x[3]

3

-2

1

4

6

3

-1

0

6

4

1

-2

Fifth call to the ISR

**Lab Report**

Your lab report will consist of the prelab and the CCS code for the FIR filter and your explanation of the code and questions raised in the lab manual.