Department of Electrical, Computer & Biomedical Engineering
Faculty of Engineering & Architectural Science

Programs: Electrical & Computer Engineering

Course Number	**ELE792**
Course Title	**Digital Signal Processing**
Semester/Year	Fall 2023
Instructor	Dr Anwar Mirza

Lab Report No.	**2**

Report Title	**Implementation of Finite Impulse Responses on the DSK**

Section No.	2
Submission Date	October 16th 2023
Due Date	October 16th 2023

Name	Student ID	Signature*
Kisoban Rajendran	501021927	K.R

(Note: remove the first 4 digits from your student ID)

**By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work <u>will result in an investigation of Academic Misconduct and may result in a "0" on the work,</u> an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at:*

https://www.torontomu.ca/content/dam/senate/policies/pol60.pdf

Introduction

In this lab, different FIR filters were implemented through different ways in C code. This was then taken a step further to study the responses of an input sine wave through a scrambler (as seen below) which implemented multiple different filters.

Design Methodology

In part 1, a lowpass filter was implemented to have a passband of 10kHz and a stopband of 15 Khz, with stopband attenuation of 60dB and maximum passband attenuation of 1dB. This was implemented as an 18th order lowpass filter in a coefficient file which was multiplied by the input signal x[n] (which was a sinusoid) in the main code. Changing one of these parameters can affect the number of coefficients needed for the filter. For example if the passband was increased, then the number of coefficients needed would also increase, but if the stopband was increased while still keeping the same passband frequency then the number of coefficients would decrease since the cutoff wouldn't be very sharp and have a longer and wider gap in the transition phase from pass to stop band.

This implementation was done in two different methods. Method 'A', multiplied the system response (low pass filter) denoted h[n] with the input (denoted 'dly[0]') then it shifted the delay samples one unit to the right since the first element (dly[0]) was reserved for the 'input_sample' function which took samples from the signal generator and outputted them to the DSP and to the oscilloscope. Method B also implemented this same system but with the use of a circular buffer. This is done through the use of pointers. Once the delay pointer reaches the last output sample (from the coefficient file) it would then point to the 'start' pointer which was set to point to the first sample of the dly (dly[0]) and this essentially continues to output the samples from the DSP since dly[0] was set to the input sample function. By measuring the number of clock cycles each method takes (by specifically placing breakpoints before and after the interrupt function), it can be seen that method B was faster than method A.

In Part 3, a notch filter was implemented to attempt to single out a single frequency. In z - domain, the frequency to be canceled out is a zero in the transfer function. Two poles are also added (one for each zero) at z = 0 so that the notch filter (in frequency domain) would have a sharper cutoff and to ensure causality (transfer function with same number of poles and zeros).

$$H(z) = \frac{(z-e^{j\omega})(z+e^{j\omega})}{z^2}$$

$$H(z) = \frac{z^2-ze^{j\omega}-z^{j\omega}+e^0}{z^2}$$

$$H(z) = \frac{z^2-2zcos(\omega)+1}{z^2}$$

$$H(z) = 1 - 2z^{-1}cos(\omega) + z^{-2} \qquad \text{Eq. (1)}$$

$$\frac{Y(z)}{X(z)} = 1 - 2z^{-1}cos(\omega) + z^{-2}$$

$$Y(z) = X(z)[1 - 2z^{-1}cos(\omega) + z^{-2}]$$

$$Y(z) = X(z) - 2X(z)z^{-1}cos(\omega) + z^{-2}X(z)$$

$$y[n] = x[n] - 2cos(\omega)x[n-1] + x[n-2] \qquad \text{Eq. (2)}$$

Eq. (2) illustrates the difference equation for the notch filter. The coefficients were implemented as a 3 element array of the coefficients in the difference equation (1, -2cos(w), 1). It also had to be noted that ω was in radians/second and the sampling frequency was set to 48Khz. These had to be taken to account for to convert ω into frequency. To convert the desired notch frequency into radians to be used as a coefficient, the conversion used was:

$$\omega = f_0 \frac{2\pi}{fs} \qquad \text{Eq. (3)}$$

Where,

f_0, is the desired notch frequency

f_s, is the sampling frequency (set at 48000Hz)

Essentially to display a notch from a $(0, \pi)$ range the notch frequency ranges would have to be from a $(0, 24Khz)$ range.

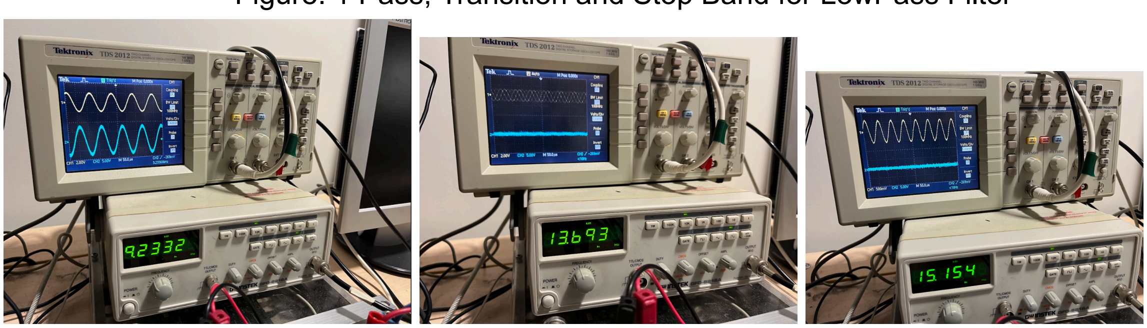
In Part 4, the implementation of the scrambler was studied. Below is the block diagram of the scrambler. The scrambler is attempting to output the lower sideband of the signal. First the input signal is put through a low pass filter to cancel all frequencies above 3 Khz or below -3Khz. This is then multiplied by another sinusoid (implemented through a coefficient file). In the frequency domain, this would shift the spectrum by the carrier frequency of 3.3Khz, essentially adding all frequencies by 3.3Khz. The maximum frequency possible after this would be 6.3Khz (upper sideband) while the minimum frequency possible is 300Hz (lower sideband). This is put through the same low pass filter once again to only keep the lower sideband of the original input spectrum. However, for this to work, the input signal's minimum frequency must be 300Hz, otherwise if the frequency was lower than 300 Hz then the lower sideband cannot be outputted since the mixer's frequency is set to 3.3Khz and therefore nothing would be outputted.

$$INPUT \rightarrow [LPF\ 3Khz] \rightarrow \otimes \rightarrow [LPF\ 3Khz] \ \rightarrow OUTPUT$$
$$\uparrow$$
$$\sim 3.3Khz$$

Results

Part 1

Figure. 1 Pass, Transition and Stop Band for LowPass Filter



Part 2

Number of Clock Cycles

Trials	1	2	3	Average
Method A	1937	1865	1873	1891.67
Method B	1679	1785	1785	1749.67

It can be seen that Method B is faster since it has lower clock cycles than Method A.

Part 3

Figure. 2 Notch Frequency at 5000 Hz

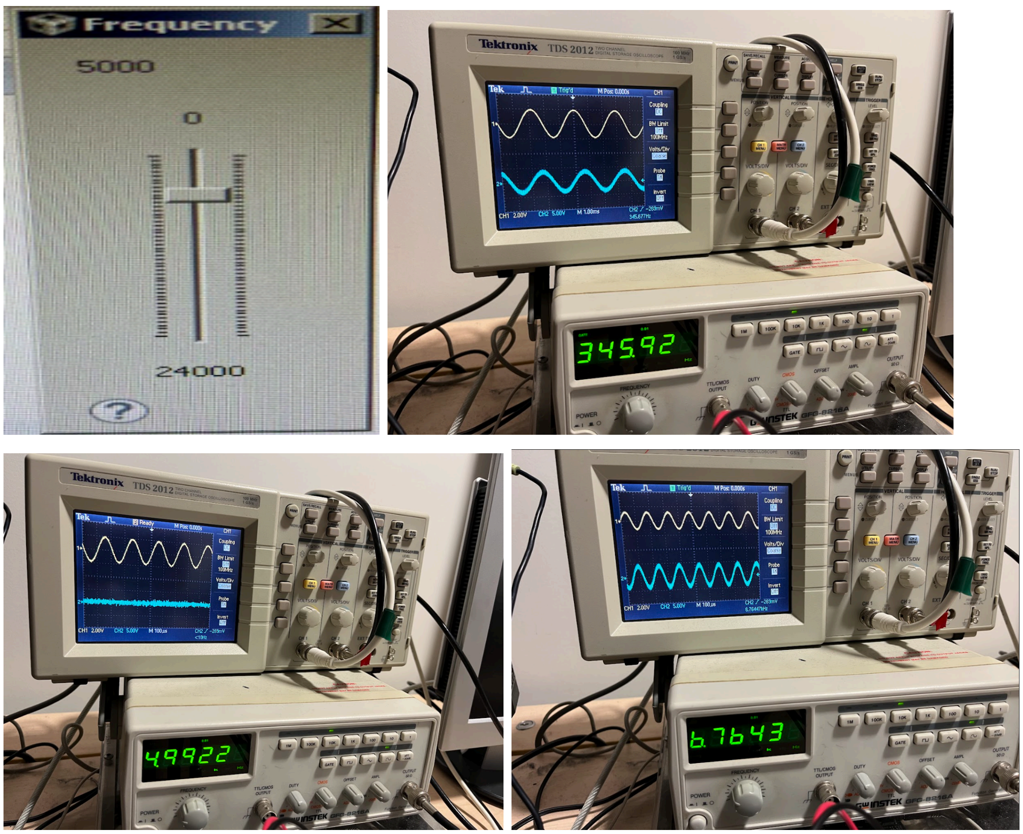
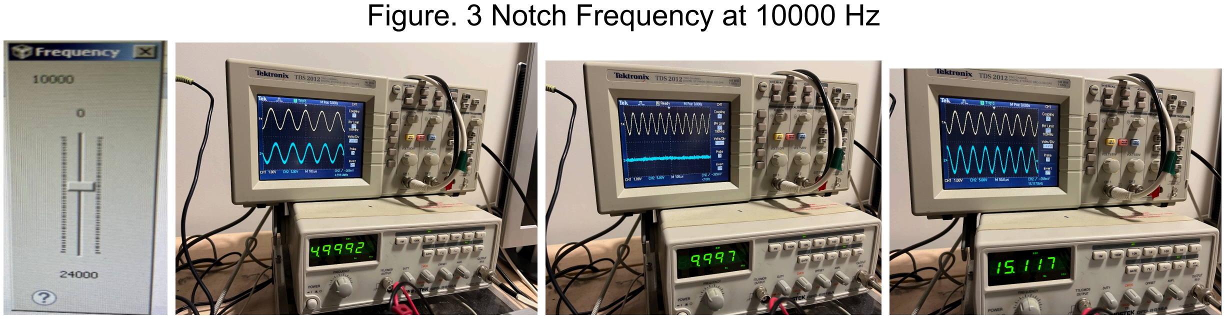


Figure. 3 Notch Frequency at 10000 Hz



Conclusion

Overall, the implementations of these filters in C code can be used in many different applications such as the scrambler code. Some challenges faced were the increasing amplitude of the output signal (in the passband of the low pass filter) which had to be mitigated with a scaling factor so that it replicated the input signal when the frequency was set to the passband frequency. Things that were learned were programming different types of filters so that it can be used in different applications such as the scrambler code.