EE 443 Design and Application of Digital Signal Processors
Homework Assignment #3 Spring 2016

Report Due: April 26, 2016

Demo: April 26, 2016

Home Assignment Grading Policy

- Assignments are graded on a group basis.
- Grades are based on both written reports and demos.
- Demos have to be presented to TA on the signed up slot. No late demo

Submission Instructions

- Capture the image from your PC or take a picture of the oscilloscope to show the results
- The report cover page should have: homework number, student name, and student number.

Problem 1:

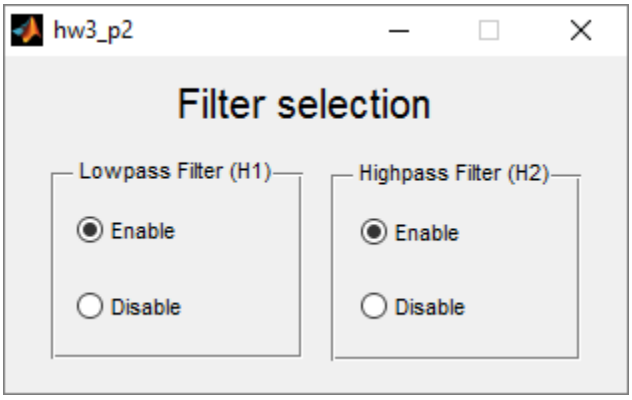
Based on the C program (Lecture Note 4, Pages 11-12) for circular-buffer-based FIR filtering, please convert the convolution routine "convolve(…)" into an assembly routine (please see the "Circular convolution using Assembly code" in our class website) to be called by a C main program, which reads in 8 KHz sampled speech sample-by-sample via interrupt. Upon the arrival of every speech sample, the assembly-based convolution routine is called to perform the lowpass FIR filtering (H1) with specifications: $w_p = 0.2\pi, w_s = 0.35\pi, R_p = 1.2dB,\ A_s = 42dB$ **.** Output the filtered speech sample to the leftChannel. Please first design a lowpass linear phase FIR filter with the equiripple scheme (use the function *sptool* in MATLAB), based on a sampling frequency of 8 KHz.

1. Use C-program to do a convolution and collect 256-filter output data.
2. Use Assembly code to do a convolution and collect 256-filter output data.
3. Send the 256-data to Matlab and do the FFT to see the frequency response for both cases.

Problem 2:

In addition to a lowpass FIR filter (H1) we designed in the Problem 1, please design a highpass FIR filter (H2) with specifications $w_s = 0.7\pi, w_p = 0.85\pi, R_p = 1.2dB, A_s = 42dB$ using *sptool*. Save the created FIR coefficients of both filters, H1 and H2, in a header file (In the filter design tool, select *Targets->Generate C Header*, and export them in a format useful for the DE2i-150 board) and implement a project to either lowpass or highpass your own real-time input stereo music, sampled at 44.1 KHz.

Make a Matlab GUI to control the filtering operations by sending the message to DE2i-150 board



1. Enable/Disable the real-time filtering using H1
2. Enable/Disable the real-time filtering using H2

When both of the real-time filters are enabled, the input stereo music has to be filtered out with both H1 and H2. Output will be a summation of two filter outputs. If both of real-time filters are disabled, the input stereo music will pass to the stereo output without any filtering. Send the 256 left and right channel data to the Matlab and see the frequency response.

Problem 3:

Design an alarm generator based on the IIR sinusoidal generator with sampling frequency **32KHz**. Use the uniform random number generator, within [0, 1], to generate the random number every 1 second. When the random number generator output is **equal or bigger than 0.5**, output a **2.4 KHz** sinusoid, when it is **below 0.5**, output a **1.2 KHz** sinusoid, both generated from the IIR generator.

Problem 4:

Write a C program to implement the tone detector based on Goertzel algorithm with IIR technique. Assign switches to on and off the tone detectors for 2000Hz, 2400Hz, and 2800Hz respectively. Detect the tones from the input signal, generated by the online tone generator sampled at 8000Hz, when switch are on. When the tone detector makes a decision, whether there is a tone or not, send the decision message to Matlab through UART to show which tone is detected in a Matlab command window. Please design your IIR filter for Goertzel algorithm, and the corresponding "N" samples used for detection, also what are the corresponding "k" indexes for the tones under this "N".

Problem 5:

We can capture the 640x480 sized face image of yourself from USB webcam using Atom part of DE2i-150 board. Since our USB camera will capture the color image, we will get 3-colors (RGB) per pixel and each color is represented in 8-bits, i.e. the captured image values of each color channel are between 0 and 255 since they are represented in 8-bit ($N_{initial}$) numbers. Now suppose you are allowed to have only 5 bits ($N_{reduced}$) to represent each color channel value, which can only represent 32 different values, but the value should still be between 0 and 255 for display (5-bit quantization). With the new 5-bit numbers, make a new image and save the resulting lower resolution image.

Tips: Let $s = \frac{2^{N_{initial}}}{2^{N_{reduced}}}$, $\text{NewValue} = floor\left(\frac{x}{s}\right) \times s + \frac{s}{2}$, where x is original value