ECE4703 B Term 2010 Laboratory Assignment 2

Project Code and Report Due by 3:00pm 10-Nov-2011

The goals of this laboratory assignment are:

- to familiarize you with the digital filter design tools in Matlab,
- to familiarize you with floating point real-time finite impulse response (FIR) filtering on the TMS320C6713,
- to familiarize you with floating point real-time infinite impulse response (IIR) filtering and two different realization structures on the TMS3260C6713, and
- to demonstrate some of the differences between real-time FIR and IIR filtering.

This assignment considers real-time DSP implementation of FIR and IIR filtering as described by input/output relationships

$$y[n] = \sum_{k=0}^{M-1} b[k]x[n-k] \qquad \text{(FIR)}$$

$$y[n] = \sum_{k=0}^{M-1} b[k]x[n-k] - \sum_{\ell=1}^{L-1} a[\ell]y[n-\ell] \qquad \text{(IIR)}$$

where $y[n]$ is the current filter output, $b[0], \ldots, b[M-1]$ are the feedforward filter coefficients, $x[n], \ldots, x[n-M+1]$ is a buffer containing the current and $M-1$ previous inputs, $a[1], \ldots, a[L-1]$ are the feedback filter coefficients, and $y[n-1], \ldots, y[n-L+1]$ is a buffer containing the $L-1$ previous outputs. Part of this assignment is in developing an understanding of the different realization structures of FIR and IIR filters, hence some parts of your code may not perform these calculations exactly as shown above. Please refer to your class notes to make sure that you are performing the calculations appropriately for each specified filter realization structure.

Filter Specifications

While the code you write for this assignment will allow you to realize almost any type of filter, you will be evaluated on your ability to realize a band-pass filter that satisfies the following requirements:

- Passband: 3500Hz – 4500Hz
- Stopband 1: All frequencies below 2500Hz
- Stopband 2: All frequencies below 5500Hz

- Passband maximum ripple of 0.5dB
- Stopband minimum rejection 40dB (both stop bands)

You can pick any sampling rate and any design method you want, e.g. equiripple, least squares, etc., as long as you can meet all of the design requirements and run in real-time. Use the Matlab filter design tools, e.g. `fdatool`, to design filters for each part below that satisfy all of the requirements. Be sure to note your design choices in your report. Use `fdatool` to plot the impulse response, magnitude response, and phase response of your filter. As shown in class, you can export your filter coefficients to C header files from `fdatool`. To do this correctly, please make sure you have selected the correct filter realization structure first, and that you are exporting the filter coefficients as the correct data type. All filter coefficients in this assignment should be generated as *single precision floats*.

In the remainder of this assignment, you will be implementing your filters in real-time on the TMS320C6713 DSK by creating C6713 projects and writing code to compute the filter output according to the filter type (FIR or IIR) and the filter realization structure.

Part 1: Floating Point DF-I FIR filtering

Create a new C6713 project and write source code that satisfies the filter constraints described in the Filter Specifications section using a direct form I FIR filter with single-precision floating point coefficients. Your filter should run in real-time on the DSK at the sampling frequency you have chosen (profile the ISR to be sure). A suggested outline for your program is given below (you are welcome to any approach that makes sense):

1. Declare variables including a single-precision floating point buffer for the input samples. Alternatively, you could declare the input sample buffer as short, and then cast these shorts to single-precision floats each time you multiply the input samples with the filter coefficients. Feel free to see which approach runs faster.
2. Initialize the DSK, codec, set the sampling rate, set up interrupts, etc.
3. Initialize the input buffer array to zero.
4. In the ISR:
   (a) Read in the input sample from the right channel of the AIC23 codec. Because we are reading the channels in stereo, you will also get the left channel too. You do not need to filter the left channel.
   (b) Cast and scale the input sample.
   (c) Put the scaled input sample into the input buffer.
   (d) Compute the filter output using the direct form I calculations with your floating-point filter coefficients and the floating point input buffer.
   (e) Unscale and cast the floating-point final result to a short for output to the DAC. You may want to check for overflow here and perhaps even light up an LED if overflow is detected.
   (f) Write the short filter output to the codec (right channel).

It is recommended that you also write the current input sample to the left channel to the AIC23 so you can easily see the input and output simultaneously on the scope. Buffering the filter output might also be handy for troubleshooting and/or visualization.

All intermediate results in this part of the assignment should be *single-precision floating-point numbers*. For each multiply and each addition, it might be a good idea to have some code to keep track of the largest positive and largest negative valued intermediate results (including the final output) in the filter. While no overflow should occur in the single-precision floating-point intermediate results of your FIR filter (unless you have a bug in your code), you can use this information to determine if overflow occurs at the output of the filter. If you are getting overflow at the filter output, you can scale the input, the filter coefficients, or the final floating-point result prior to casting it as a 16-bit signed integer. This largest positive and largest negative valued intermediate result information will also be very useful to avoid overflow and select proper scaling factors for the intermediate results when developing a fixed-point implementation of the FIR filter in Lab 3.

Convert the final result to a 16-bit signed integer (short datatype) only prior to output by the AIC23 DAC. To generate convincing results that your filter is working correctly, generate a white noise signal in Matlab as discussed in class

```
fs = 44100; % sampling frequency
T = 10; % duration
x = 2*rand(fs*T,1)-1; % zero-mean white noise
x = x*0.95; % reduce the amplitude of x to 95% of full scale
```

and play this white noise signal out of the computer's sound card while recording the filter output of the DSK to a .wav file (preferably on a different computer). Make sure the white-noise signal from the sound card is scaled to almost the entire full range of the AIC23 ADC. You can open this .wav file in Matlab and use the `pwelch` command as described in class

```
[Px,f] = pwelch(leftchannel,1024,512,1024,fs); % estimate spectrum of unfiltered output
[Py,f] = pwelch(rightchannel,1024,512,1024,fs); % estimate spectrum of filtered output
plot(f/1e3,10*log10(Px),f/1e3,10*log10(Py)); % plot spectra in dB
grid on
xlabel('frequency (kHz)');
ylabel('magnitude response (dB)');
```

to plot the magnitude response of your filter. You should type `help pwelch` in Matlab to learn more about what the `pwelch` command does. The absolute dB values of the input and output spectra probably won't make much sense, but the relative values between the input and output should agree with the theoretical predictions from the Matlab filter design tools. Plot the theoretical predicted magnitude response on top of your simulated filter response to see how well they match (you may need to apply an scale factor (or dB offset) to get things to line up). The magnitude response of your real-time FIR filter should be almost indistinguishable from the `fdatool` prediction.

Part 2: Floating-point DF-II Single-Section IIR filtering

Create a new C6713 project and write source code that satisfies the filter constraints described in the Filter Specifications section using a *direct form II single-section IIR filter* with single-precision floating point coefficients and single precision floating point intermediate results. You can use `fdatool` to generate appropriate filter coefficients in the appropriate filter structure and export these coefficients to a header file for use in your C6713 project. Your filter should run in real-time on the DSK (profile to confirm).

As with part 1, you should keep track of the largest positive and largest negative valued intermediate results (including the final output) to ensure there is no overflow at the output. You should also initialize the intermediate result buffer array to zero. Remember that DF-II does not have input/output buffers. You should be only storing and updating the **intermediate buffer** (stored as single precision floats) upon the arrival of each new input sample.

Part 3: Floating-point DF-II Second-Order-Sections IIR filtering

Create a new C6713 project and write source code that satisfies the filter constraints described in the Filter Specifications section using a *direct form II second-order-sections IIR filter* with single-precision floating point coefficients and single precision floating point intermediate results. You can use `fdatool` to generate appropriate filter coefficients in the appropriate filter structure and export these coefficients to a header file for use in your C6713 project. Your filter should run in real-time on the DSK (profile to confirm).

Please write one DF-II SOS function that you can call multiple times from your interrupt service routine. It is up to you how to do this, but the idea here is that your overall filter can be realized by calling the DF-II SOS function several times, each time passing the output from the last DF-II SOS to the input of the next DF-II SOS. All intermediate results should be kept as single precision floats.

The usual difficulty with this part of the assignment is in interpreting the header files that Matlab generates for DF-II SOS filters. Please see the course web site "links and files" page for some information on how to correctly interpret these header files.

In Lab

You will be working in the same teams as in Lab 1. Please let the instructor know if your lab partner has dropped the course.

Specific Items to Discuss in Your Report

In your report, provide theoretical and experimental magnitude response results in the same plot (use different colors or line styles) for each part of the assignment. Your plots should look professional with axis labels, grid lines, and legends as necessary. Your plots should provide immediately convincing evidence to the grader that your filter running on the DSK satisfies the requirements and that that it agrees with the magnitude response predictions from Matlab. Be sure to explain any discrepancies (there may not be any). Discuss any scaling considerations that you used to get your filters working correctly and to avoid overflow in the intermediate results and the output.

Profile the execution time of all of your filters, confirm that they are running in real time, and discuss any differences and/or similarities in your results.