ECE4703 B Term 2008 Laboratory Assignment 3

Project Code and Report Due by 3:00pm 20-Nov-2008

The goal of this laboratory assignment is:

- to explore the effects of filter coefficient quantization, fixed-point processing, and filter realization structures for real-time infinite impulse response (IIR) filtering on the TMS320C6713.

This assignment considers real-time DSP implementation of IIR filtering as described by the input/output operation

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

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.

Like the previous assignment, the code you write for this assignment will allow you to realize any IIR filter within real-time capabilities of the C6713. To provide a common basis for evaluation, you should realize a filter that satisfies the following requirements:

- Bandpass filter
- IIR elliptic design method.
- Minimum filter order
- Options = "match exactly both"
- Sampling rate 32kHz.
- Frequency edges: fstop1 = 4kHz, fpass1 = 4.5kHz, fpass2 = 5.5kHz, fstop2 = 6kHz
- Magnitude specifications: Astop1 = 50dB, Apass = 1dB, Astop2 = 50dB

Use the Matlab filter design tools, e.g. `fdatool`, to design an IIR filter that satisfies the requirements. Plot the impulse response, magnitude response, and phase response of your filter. Note the filter structure chosen by `fdatool`. You may want to convert this structure to a different structure by going to the "Edit" menu in `fdatool` and exploring the options available to you for converting the filter structure and also for converting between a single section and a cascade of second order sections (SOS). For the reasons described in lecture, we will be focusing primarily on the Direct Form II structure with second order sections in this assignment (DFII-SOS).

1 Part 1: IIR filtering in Matlab

As in the previous assignment, we will first simulate the IIR filter in Matlab to ensure that it is working correctly and to also measure certain internal characteristics of the filter that will be useful when we realize the filter in fixed-point and on the DSK.

1.1 Part 1a: Floating Point Implementation of an IIR filter

For the floating-point implementation of an IIR filter, we will assume that all filter coefficients and intermediate calculations are represented as 64-bit double-precision floating numbers. The inputs to the filter are assumed to be 16-bit signed integer (short datatype) samples from the AIC23 ADC. The outputs of the filter must also be cast to 16-bit signed integers for proper interpretation by the AIC23 DAC.

You can use the same input/output quantization code as in Lab assignment 2. The main difficulty here will be in interpreting second order sections filter coefficients generated by Matlab. Please see the "links and files" link on the course web page for some hints on how to interpret the C header file generated by `fdatool`. While you are not generating C header files in this part of the assignment, this should still give some hints on how the matrix of second order section filter coefficients should be interpreted.

Some important details:

- As in the previous assignment, the goal here is to simulate operation on the C6713, so please do not use the Matlab commands `filter` or `conv` to compute the output of your filter. Instead, use `for` loops (or `while` loops if you prefer).
- Please write a separate DFII-SOS function in Matlab that you can call repeatedly in the computation of your filter output. It is recommended that you make the SOS filter coefficients and each section's internal memory arrays global variables. In this case, the SOS function will only require you to pass in the SOS section's index and its current input. Your SOS function can access the global variables to retrieve the appropriate filter coefficients and internal memory and then use these quantities to compute the SOS output. The SOS output should be passed back by your SOS function. Your SOS function should also update the DFII-SOS internal memory (global) appropriately.
- As with FIR filtering, you should keep track of the largest positive and largest negative valued intermediate results (including the final output) in each stage of the filter.
- You may want to initialize the input and output buffer samples to zero.

Quantize the final calculation in $Q15$ format (checking for overflow and handling it appropriately) and store the result in a vector of IIR filter outputs. Test that your filter is working correctly by following the procedure described in the previous assignment. Try to 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 offset to get things to line up).

Please make sure your floating-point IIR filter implementation is working correctly and that you are able to determine the largest positive and largest negative intermediate results before proceeding to the fixed-point IIR filter implementation.

1.2 Part 1b: Fixed Point Implementation of an IIR filter

For the fixed-point implementation of your IIR filter, all filter coefficients (both feedforward and feedback) should be quantized as short datatypes (16-bit signed integers). You should determine the best Q-representation that will allow you to represent your filter coefficients with minimum quantization error while also avoiding overflow. All intermediate calculations should be stored as 32-bit signed integer variables with an appropriate number of fractional bits to avoid overflow and minimize truncation error. You should also use the same Q-representation for the feedforward and feedback filter coefficients since the inputs and outputs are both interpreted as $Q15$ and you can only add variables that have the same number of fractional bits.

You should check for overflow manually after every fixed-point calculation. You can use the same overflow function that you used in the previous lab assignment. How should you scale your final 32-bit intermediate result to produce the 16-bit output? Can you use any of the "largest positive" and "largest negative" results from your floating-point IIR filter to determine appropriate scaling factors? If you are careful to avoid overflow and minimize truncation error, you should be able to achieve a fixed-point IIR filter response almost identical to the response you obtained in the floating-point case.

2 Part 2: IIR filtering on the DSK

In the remainder of this assignment, you will be implementing your IIR filter in real-time on the TMS320C6713 DSK by writing code in C to compute the filter output. You should be using the same filter coefficients as in Part 1, but you may find it more convenient to export these coefficients to a C header file for inclusion in your CCS projects.

2.1 Part 2a: Floating-Point Implementation of an IIR filter on the DSK

Convert your Matlab code from Part 1a to a C6713 project that runs in real-time on the DSK at a sampling frequency of 32kHz. It is recommended that you write an SOS function that you can call repeatedly from your ISR to compute the filter output. All intermediate results in this part of the assignment should be *double-precision floating-point numbers*. Convert the final result to a 16-bit signed integer (short datatype) only prior to output by the AIC23 DAC.

Test that your filter is working correctly by following the procedure described in Lab assignment 2. The magnitude response of your real-time IIR filter should be indistinguishable from the Matlab simulation in Part 1a. Profile your code to determine the number of cycles needed to complete the ISR (including calls to your SOS function). Does your code run in real-time?

2.2 Part 2b: Fixed-Point Implementation of an IIR filter on the DSK

Convert your Matlab code from Part 1b to a C6713 project that runs in real-time on the DSK at a sampling frequency of 32kHz. *You are not permitted to use any floating-point variables in this part of the assignment.* Use the same steps as Part 2a to determine if your filter is working correctly and that it is running in real-time.

3 In Lab

You will be working in the same teams as in Lab 1.

4 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. Be sure to explain any discrepancies. Discuss all 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 your final real-time double-precision floating-point IIR filter and your final real-time fixed-point IIR filter on the DSK, including all calls to external functions. Discuss any insight obtained from the profiling results.

In addition to meeting the requirements of the assignment, there are several opportunities for additional exploration that you should discuss in your report:

1. What is the minimum number of bits needed to represent the coefficients of the bandpass filter so that the filter is stable after quantization? Discuss how the filter structure (DFII versus DFII-SOS) changes your answer.

2. Discuss how your DFII-SOS fixed-point representations and scaling strategy would change if your intermediate results had to be stored in 16-bit signed integers. Can you confirm that your strategy would work by testing it in Matlab and/or on the DSK?

3. In the fixed-point implementation of the FIR filter on the DSK, purposely change the design of your filter to cause occasional overflow. How does the effect of overflow differ between FIR and IIR filters?