**EE 445S Real-Time Digital Signal Processing Laboratory – Prof. Brian L. Evans**

**Lab 3 Instructions – Part 2**

**1. Frame-based talk-through**

Create a new project as before. After adding the files in “common code” folder, add “frames.h”, “ISRs.c” and “main.c” in “code\chapter\_06\ccs\Frame”. Finally add the target configuration files as usual.

Run the program; connect the board with the signal generator and the oscilloscope. Check the result of talk-through. Tune the frequency in the signal generator and check the wave form in the oscilloscope; what do you find?

**2. FIR filter design (same as last week)**

Run fdatool in Matlab to design a bandpass FIR filter:

* Design method: Equiripple
* Order: 30
* Sampling frequency: 48000Hz
* Passpand: 5000Hz to 15000Hz (Fstop1 and Fstop2 can be 4000Hz and 16000Hz)

Export the coefficients into the workspace with variable name “Num”.

Set “C:\CD\code\appendix\_e\MatlabExports” as the current folder of Matlab.

In the command window run

>>fir\_dump2c(‘coeff’,’B’,Num,length(Num));

You will find “coeff.c” and “coeff.h” in the current folder. Open “coeff.h”, change “define B\_SIZE 31” to “define N 30”. Open “coeff.c”, change “B\_SIZE” to “N+1”. You will need these two files later.

**3. Theoretical magnitude response measurement (same as last week)**

Go back to fdatool. Measure the magnitude response (dB) for the FIR filter that you just designed. You need to measure the magnitude response for 1 kHz, 2 kHz … 24 kHz. Save these values in an Excel file. You will need to compare them with the experimental values later.

**4. DSK implementation of frame-based FIR filter**

Create a new project as before. After adding the files in “common code” folder (instead of adding “vectors.asm”, please add “vectors\_EDMA.asm”); add “frames.h”, “ISRs.c” and “main.c” in “code\chapter\_07\ccs\FiltFrm\_6748”. Add the “coeff.h” and “coeff.c” in “code\appendix\_e\MatlabExports” that you just generated. Finally add the target configuration files as usual.

Run the program; connect the board with the signal generator and the oscilloscope. Tune the frequency in the signal generator from 1 kHz to 24 kHz and record the experimental magnitude response. Compare this with the theoretical magnitude response. (You may draw two lines in the same graph)

**5. Clock cycle measurement**

Compare the number of clock cycles as follows:

(1) Frame-based FIR vs. Sample-based FIR

 Measure the number of clock cycles cost by the “ProcessBuffer()” function; divide the number by 1024 (length of the frame). This is the number of clock cycles for generating one filter output sample using Frame-based FIR. (Note that we don’t need to consider the clock cycles for data transfer since it’s handled by the EDMA.)

 Measure the number of clock cycles cost by the codes between “ReadCodecData()” and “WriteCodecData()” in the “FIRmono\_ISRs.c” file that you used last week for the sample-based FIR filter using circular buffering.

 Compare the two numbers.

(2) Code optimization

 For circular buffering scheme, measure the number of clock cycles cost by the codes between “ReadCodecData()” and “WriteCodecData()” in the “FIRmono\_ISRs.c” while using different optimization levels from 0 to 3.

To change the optimization level you can right click on the project 🡪 build properties 🡪 basic options 🡪 choose the optimization level.

 Now, we would like to compare the performance of the C FIR filter routine using circular buffer with one coded in assembly language. Please download the assembly language file convol1.sa from <http://users.ece.utexas.edu/~bevans/courses/realtime/lectures/laboratory/c6748winDSK/lab3/index.html> and add it to your project. Also, instead of the FIRmono\_ISRs.c routine used till now, please replace it with the one in the webpage mentioned above. Insert breakpoints between lines 70 and 72 of FIRmono\_ISRs.c to profile the assembly code. Please go through the assembly code. In this case, the ‘convolve’ function accepts as input the size of the circular buffer in ‘blocks’. Circular buffer size (in bytes) is 2^(Nblock+1). The data has been aligned in memory using the #pragma directive in FIRmono\_ISRs.c. Try different optimization levels from 0 to 3.