

525.446

DSP Hardware Laboratory

Assignment #5

Electronic Project Archive due Oct 14 before 7:20 p.m. demonstration in class Oct 14

FIR, Lab2

Introduction

In “FIR, Lab1”, a simple FIR filter was implemented entirely in unoptimized C. This laboratory assignment, “FIR, Lab2”, is intended to be a good example of the performance gains that can be realized with careful attention. In this lab, we will design a new, longer filter in MATLAB to filter a sample input signal. This will be attained by optimization of the convolution sum using:

- 1) Optimizing features of the C compiler
- 2) TMS320C6713 DSPLIB – hand tuned assembly language functions written by TI.

Goal

To make the longest FIR filter possible, and quantify some of the performance gains to be had with various levels of effort.

Reading

Reference Chapter 3 (Architecture and Instruction Set of the C6x Processor) and Chapter 4 (Finite Impulse Response Filters) of Chassaing – both will be useful for this assignment.

References

SPRA666 : *Hand-Tuning Loops and Control Code on the TMS320C6000*

SPRAA14 : *Introduction to Compiler Consultant*

SPRU657B : *TMS320C67X DSP Library Programmer's Reference Guide*

DSP Implementation

The basic outline for evaluation of these techniques will use the framework from “FIR Lab1”. For all pieces of this lab, keep the functionality of the first FIR lab. Similarly, the data type should be at least “float” to make comparison fair.

Part I)

Use a function from the C6713 DSPLIB to aid in making a fast filter. Many functions exist to help here, and all are hand-tuned assembly language. Note : Though there are lots of functions tailored for filtering, they may not be the easiest to use in our simple “sample at a time” framework. DSP_dp_dotprod and DSP_sp_dotprod are math functions which can be used in our framework rather easily. When using functions, remember to note the **restrictions** that are placed on the using of them. As discussed in class, certain assumptions with respect to alignment of data, number of iterations...etc. were made when writing this code. These restrictions may dictate slight modifications of your main program.

Use compiler comments, LEDs, or timer code to estimate what you think is the maximal length filter that you can implement using this method, and design that filter (e.g. in FDAtool.)

Implement on the board and verify correct performance. Graph the real frequency response and theoretical (the chirp can be used for this)

Part II)

Now, using only C code, and the techniques discussed in lecture and the references, do the same thing as in part I. Namely, make a filter implementation, calculate the maximum length that you can attain with everything at your disposal, design a filter for that length (or slightly less ☺) and implement. Create plots of measured and theoretical frequency response.

Downloading DSPLIB

The Code Composer installation that came with the DSK6713 did not include the correct DSPLIB. You will need to download correct version of DSPLIB for the C6713, available at this URL: <http://focus.ti.com/docs/toolsw/folders/print/spr121.html>.

Note, there are no hard specifications for the filter design itself, use your judgement. Reasonable implementation can make a very good filter with optimizing techniques, signals which move outside the passband should be very quickly attenuated to the point where they can't be heard.

525.446
DSP Hardware Laboratory
Assignment #5, Grade Sheet

FIR, Lab2 (____/100)

- (____/5) **SW1 chooses if the FIR filter is in the loopback path. LED1 is lit as an indicator.**
- (____/5) **SW2 chooses if the DDS is injected in the loopback path prior to the filter (it should be added to the input samples). LED2 is lit as an indicator.**
- (____/5) **LED3 indicates how long the processing is taking.**
- (____/25) **fir filter generated and correctly implemented using a function from the C67x DSPLIB for filtering.**
>= 1024 taps = 25, >= 800 taps = 20, >= 512 taps = 15, <512 taps = 12
Sample rate not preserved : 0
- (____/25) **fir filter generated and correctly implemented using pure C code with tuning techniques**
>= 1024 taps = 25, >= 800 taps = 20, >= 512 taps = 15, <512 taps = 12
Sample rate not preserved : 0
- (____/20) **(document) Performance described and documented.**
Overall architecture described
Steps taken to achieve performance gains described
Documentation of how you decided the maximum filter length that you could implement.
- (____/10) **(document) Performance Plots**
Theoretical freq response of C filter
Theoretical freq response of DSPLIB filter
Measured freq response of C filter
Measured freq response of DSPLIB filter
- (____/5) **Well Documented Code**