EE 451 – LAB 5

FIR Filter Design

In this laboratory you will design FIR filters with different window techniques, and will learn more about the capabilities of the C6713.

Introduction

Filtering is one of the most useful signal processing operations. DSPs are now available to implement digital filters in real time. A FIR filter operates on discrete-time signals and can be implemented on a DSP such as the TMS320C6x. This process involves the use of an ADC to acquire an external input signal, processing of the samples, and sending the result through a DAC. Filter characteristics such as center frequency, bandwidth, and filter type can be readily implemented and modified.

The Prelab

1. Find the impulse response, $h[n]$, of the following lowpass filter,

$$H(e^{jw}) = \begin{cases} 1 & |\omega| \leq \omega_c \\ 0 & \omega_c < |\omega| \leq \pi \end{cases}$$

where $\omega_c = \pi/4$.

2. $h[n]$ has infinite number of terms. In order to implement the filter you will need to limit the number of terms used. Write a MATLAB code to generate 101 terms of $h[n]$ for n = -50,…,50, and plot of the truncated impulse response.

3. Store the 101 terms in a text file with the following format:

```
//=========== fir.cof =========
// This file is used in the FIR lab
// Created by Hector Erives 8/2008

 #define N 101

float h[N]={
       0.0063661977,
       0.0045934506,
             . . .
};
```

This will be a header file which you will include in your program and compile in the main program.

4. Plot the magnitude of $H(e^{jw})$ vs. f

5. Change the number of taps and comment on the change in the magnitude of the response of $H(e^{jw})$.

6. Up to this point you have used a rectangular. Now, instead of using a rectangular window use a Hamming window. Use a stem plot to plot the new coefficients. Figure 1 shows filter responses with the two windows, and a sampling frequency of F_T= 8 kHz.

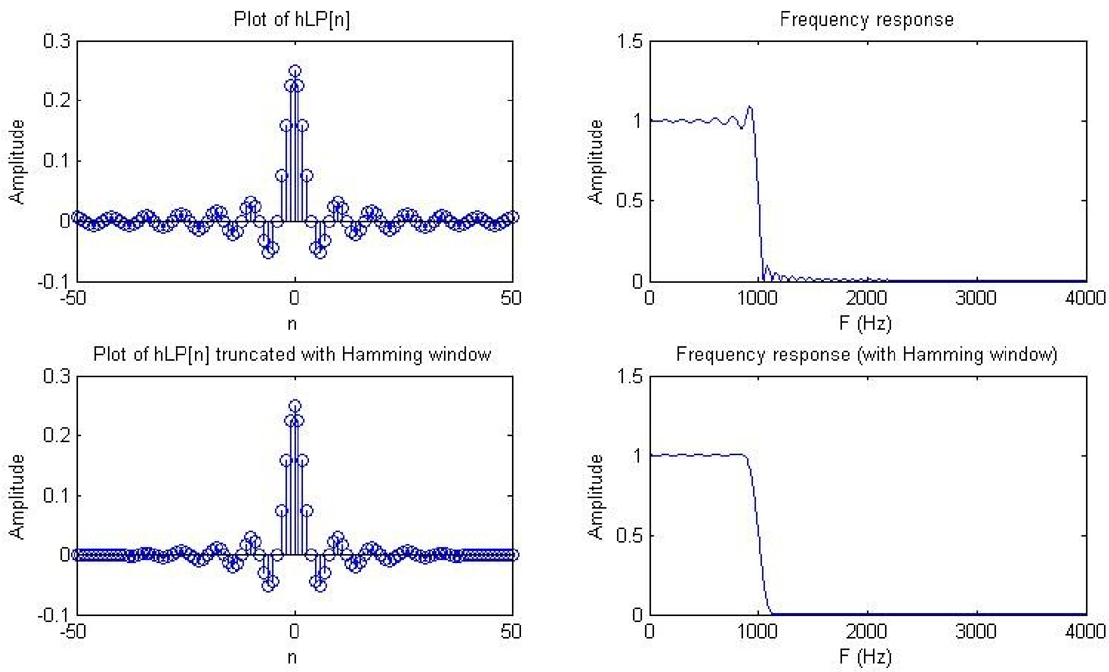

Figure 1. Filter responses with two different windows.

The Lab

Create a program that implements and FIR filter using the N coefficients *h(0)*, *h(1)*, … , *h(N-1)* you have generated in the prelab using a rectangular window, and N input samples *x(n)*, *x(n-1)*, … , *x(n-(N-1))* uses an ISR to read a signal from LINE IN and output it through the LINE OUT in real-time. Connect the function generator to the LINE IN input and verify that the program can reproduce the signal at the LINE OUT.

1 Start CCS and begin a new project. Create and add a configuration file to the project. Select File → New → DSP/BIOS Configuration. We will use a HWI to (i) read a new input sample from the codec, (ii) calculate the filter output, (iii) shift delay line contents, and (iv) output to the codec, as outlined in Figure 1.

2 Write some code to implement the following equation in the ISR

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

where *h(k)* are the coefficients you have generated using a rectangular window, and *x(n)* are the data read from the codec. Note: use a circular buffer to store *x(n)*

3 Output the original signal from the left channel on the left channel and the output *y(n)* on the right channel.

4 Connect the function generator to the LINE IN and an oscilloscope to LINE OUT. Observe the two channels on the oscilloscope as you vary the frequency on the function generator.

5 Measure the frequency response of the filter and compare it to the MATLAB results.

6 Now use the coefficients generated using the Hamming window. Measure the frequency response of the filter and compare it to the MATLAB results.

7 What is the limitation on the order of the FIR filter?

```
//========= Lab4.c =========
//  This program implement a FIR filter
//

#include "dsk6713.h"
#include "dsk6713_aic23.h"                          // codec support
#include "dsk6713config.h"

#include "fir.cof"
float x[N];

Uint32 fs = DSK6713_AIC23_FREQ_8KHZ;                // set sampling rate
#define DSK6713_AIC23_INPUT_MIC 0x0015
#define DSK6713_AIC23_INPUT_LINE 0x0011
Uint16 inputsource=DSK6713_AIC23_INPUT_LINE;        // select LINE IN input

void filter(void)
{

  // code to read LINE IN, compute the output of the FIR, and output it to LINE OUT
  return;

}

void main()
{

  // Set up needed to for interrupts
  IRQ_globalDisable();                              //disable interrupts
  DSK6713_init();                                   // call BSL to init DSK-EMIF,PLL)
  hAIC23_handle=DSK6713_AIC23_openCodec(0, &config);// handle(pointer) to codec
  DSK6713_AIC23_setFreq(hAIC23_handle, fs);         // set sample rate
  DSK6713_AIC23_rset(hAIC23_handle, 0x0004, inputsource);  // choose mic or line in
  MCBSP_config(DSK6713_AIC23_DATAHANDLE,&AIC23CfgData);// interface 32 bits to AIC23
  MCBSP_start(DSK6713_AIC23_DATAHANDLE, MCBSP_XMIT_START | MCBSP_RCV_START |
  MCBSP_SRGR_START | MCBSP_SRGR_FRAMESYNC, 220);    // start data channel
  CODECEventId=MCBSP_getXmtEventId(DSK6713_AIC23_codecdatahandle);//McBSP1 Xmit
  IRQ_map(CODECEventId, 11);                        //map McBSP1 Xmit to INT11
  IRQ_reset(CODECEventId);                          //reset codec INT 11
  IRQ_globalEnable();                               //globally enable interrupts
  IRQ_nmiEnable();                                  //enable NMI interrupt
  IRQ_enable(CODECEventId);                         //enable CODEC eventXmit INT11
  MCBSP_write(DSK6713_AIC23_DATAHANDLE,0);          //start McBSP interrupt outputting a sample

}
```

Figure 1. Program template for Lab4.