Department of Electrical and Electronic Engineering
Imperial College of Science, Technology and Medicine

EE3T Study Project: Real-Time Digital Signal Processing with TMS320C6000

Laboratory 5 – Real-time Implementation of FIR Filters

Objectives

- Learn to design FIR filter using MATLAB
- Implement the FIR filter using the C6711 system in real-time
- Measure the filter characteristics using a network or spectrum analyser

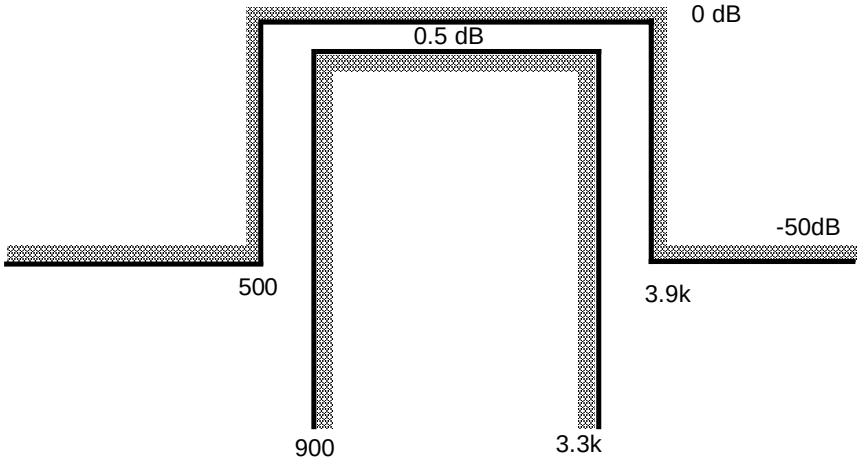
The Problem

The transfer function of an M[th] order FIR filter is given by: $H(z) = b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}$

The corresponding time-domain difference equation is:

$$y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_M x(n-M)$$

Our objective in this laboratory session is to design a FIR filter that satisfies the following specification:-



Show that a passband peak-to-peak ripple is 0.5 dB (i.e. ±0.25dB) is equivalent to a passband deviation of 0.0292, and show that a stopband rejection of -50dB or better translates to a stopband deviation of 0.00316.

Frequency Range	Type	Max Deviation
0 – 500 Hz	stopband	0.00316
0.9 – 3.3 kHz	passband	0.0292
3.9 – 4.0 kHz	stopband	0.00316

Filter Design using MATLAB

Your first step is to design this filter in MATLAB using the Park-McCelland algorithm. Read the description of MATLAB functions `remez` and `remezord` given in Appendix A. You may also find the function `freqz` useful in calculating the frequency response of the filter. Also remember that the sample frequency of the C6711 system is 8 kHz.

You should plot the frequency response of your filter showing clearly that the passband and stopband specifications are met. The resultant coefficients for your filter should be written into a text file called `fir_coef.txt` in a format suitable for inclusion in a C program. For example, this file might contain:

```
// FIR filter coefficients
float  b[ ] = {1.0,  0.2356,  -0.03987,  …….
                       };
```

Note that MATLAB output format is defaulted to only 5 significant decimal digits. Single precision floating point format used by C6711 has around 8 significant decimal digits. Therefore to use the maximum precision available, you must force MATLAB to use long format with the command: `format long e`.

Filter Implementation

You should build this on top of the Lab 4 session on interrupt i/o. For the proper operation of the FIR filter, it is required that the current sample (read using the special function `AD535_HWI_read`) and `M-1` previous samples be processed at the same time, where M is the order of the filter. Therefore your interrupt service routine should perform the following tasks:

- Read one input sample from the codec
- Perform the delay operator z^{-1} to `x(n-1)` to `x(n-M)`
- Perform the convolution function of the filter
- Output `y(n)` to the codec

Since this is interrupt driven, the main program you use should be essentially the same as the interrupt i/o program in lab 4.

Test your implementation with the following compiler optimization options: 1) no optimization; 2) with option –o0; 3) with option –o2. Compare the number of instruction cycles required in each case.

Measurement of Filter Response

Use one of the spectrum analysers in the laboratory to measure the frequency response of your filter working on the C6711 system. Compare the magnitude response of your implemented filter to that predicted by MATLAB.

Optional Extra

If you have time, you may modify your design to implement the FIR filter in integer arithmetic only . Compare the implemented performance to the previous floating-point implementation.

APPENDIX A

REMEZORD FIR order estimator (lowpass, highpass, bandpass, multiband)

[N,Fo,Mo,W] = REMEZORD(F,M,DEV,Fs) finds the approximate order N, normalized frequency band edges Fo, frequency band magnitudes Mo and weights W to be used by the REMEZ function as follows:

B = REMEZ(N,Fo,Mo,W)

The resulting filter will approximately meet the specifications given by the input parameters **F, M**, and **DEV**. **F** is a vector of cut-off frequencies in Hz, in ascending order between 0 and half the sampling frequency **Fs**. If you do not specify **Fs**, it defaults to 2. **M** is a vector specifying the desired function's amplitude on the bands defined by **F**. The length of **F** is twice the length of **M**, minus 2 (it must therefore be even). The first frequency band always starts at zero, and the last always ends at **Fs/2**. **DEV** is a vector of maximum deviations or ripples allowable for each band.

EXAMPLE: Design a lowpass filter with a passband cutoff of 1000Hz, a stopband cutoff of 2000Hz, passband ripple of 0.01, stopband ripple of 0.1, and a sampling frequency of 8000Hz:

```
[n,fo,mo,w] = remezord( [1000 2000], [1 0], [0.1 0.01], 8000 );
b = remez(n,fo,mo,w);
```

CAUTION 1: The order N is often underestimated. If the filter does not meet the original specifications, a higher order such as N+1 or N+2 will.

CAUTION 2: Results are inaccurate if cutoff frequencies are near zero frequency or the Nyquist frequency.

REMEZ Parks-McClellan optimal equiripple FIR filter design.

B=REMEZ(N,F,M) returns a length N+1 linear phase (real, symmetric coefficients) FIR filter which has the best approximation to the desired frequency response described by F and M in the minimax sense. F is a vector of frequency band edges in pairs, in ascending order between 0 and 1. 1 corresponds to the Nyquist frequency or half the sampling frequency. M is a real vector the same size as F which specifies the desired magnitude of the frequency response of the resultant filter B. The desired response is the line connecting the points (F(k),M(k)) and (F(k+1),M(k+1)) for odd k; **REMEZ** treats the bands between F(k+1) and F(k+2) for odd k as "transition bands" or "don't care" regions. Thus the desired magnitude is piecewise linear with transition bands. The maximum error is minimised.

B=REMEZ(N,F,M,W) uses the weights in W to weight the error. W has one entry per band (so it is half the length of F and M) which tells **REMEZ** how much emphasis to put on minimising the error in each band relative to the other bands.

See also FIRLS, FIR1, FIR2, BUTTER, CHEBY1, CHEBY2, ELLIP, FREQZ and FILTER.