Lecture 6  FIR Filter Implementations

Objectives:
- Understand basics of FIR filter design using MATLAB
- Understand C67x implementations
- Understand circular buffer usage

FIR Filter Basics
- The transfer function of a $M^{th}$ order FIR filter is:
  \[ H(z) = b_0 + b_1 z^{-1} + \cdots + b_M z^{-M} \]
- The difference equation is:
  \[ y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_M x(n-M) \]
- For linear phase FIR filters, the coefficients are real & symmetrical
- FIR filters are easy to implement in either hardware or DSP software
- FIR filters are inherently stable

FIR Filter Design in MATLAB
- MATLAB provides a large variety of routines for designing FIR filters. We will concentrate on Parks-McClellan’s algorithm.

<table>
<thead>
<tr>
<th>FIR Filter Design</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fir1</td>
<td>Design a window-based finite impulse response filter</td>
</tr>
<tr>
<td>fir2</td>
<td>Design a frequency-sampling-based finite impulse response filter</td>
</tr>
<tr>
<td>fircls</td>
<td>Constrained least square FIR filter design for multiband filters</td>
</tr>
<tr>
<td>firclsdf</td>
<td>Constrained least square filter design for lowpass and highpass linear phase FIR filters</td>
</tr>
<tr>
<td>firclfl</td>
<td>Least square linear-phase FIR filter design</td>
</tr>
<tr>
<td>firacscos</td>
<td>Raised cosine FIR filter design</td>
</tr>
<tr>
<td>firinterpol</td>
<td>Interpolation FIR filter design</td>
</tr>
<tr>
<td>kaiserord</td>
<td>Estimate parameters for an FIR filter design with Kaiser window</td>
</tr>
<tr>
<td>equiripple</td>
<td>Complex and nonlinear-phase equiripple FIR filter design</td>
</tr>
</tbody>
</table>

REMEZORD & REMEZ functions
- \([N,Fo,Mo,W] = \text{REMEZORD}(F,M,DEV,Fs)\)
  - 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:
    \[ B = \text{REMEZ}(N,Fo,Mo,W) \]
  - 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 $Fs$.
  - $M$ 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.
A Design Example

- Lowpass filter spec:
  - 500 Hz passband cutoff frequency, less than 3 dB passband ripple
  - 600 Hz stopband cutoff frequency, with at least 40 dB attenuation
  - Sampling frequency of 2000 Hz

\[
\begin{align*}
\text{rp} &= 3; & \% \text{ Passband ripple} \\
\text{rs} &= 40; & \% \text{ Stopband ripple} \\
\text{fs} &= 2000; & \% \text{ Sampling frequency} \\
f &= [500, 600]; & \% \text{ Cutoff frequencies} \\
a &= [1, 0]; & \% \text{ Desired amplitudes}
\end{align*}
\]

% Compute deviations
\[
\text{dev} = [(10^{\text{rp}/20}-1)/(10^{\text{rp}/20}+1) \quad 10^{-\text{rs}/20}]
\]

\[
[n,fo,ao,w] = \text{remezord}(f,a,\text{dev},fs);
\]

\[
b = \text{remez}(n,fo,ao,w);
\]

freqz(b,1,1024,fs);

\[
\text{title(}'\text{Lowpass Filter Designed to Specifications}'\text{)};
\]

FIR Implementation on C6711

- Must declare buffer arrays to store input samples \( x(n) \), and coefficients \( b(n) \), for \( n = 0 \) to \( M \)

```c
#define M 12
// FIR filter coefficients
float b[] = {1.0, 0.2356, -0.03987, ...

float x[M];
```

- Require to implement delay operator by shifting all previous \( M-1 \) samples and insert current input sample in \( x[0] \):

```c
// Z^-1 delay operator
for (i=M-2; i>=0; i--)  x[i+1] = x[i];
x[0] = input_sample;
```

Circular Buffers

- Shifting data this way is a form of circular buffer - inefficient
- Far better to implement this circular buffer using fixed memory (without moving data) and a moving pointer
- After \( N \) accesses, \( A4 \) returns to the same place

Circular buffer with sliding window
C6x Support for circular buffers

- Program Address Mode Register AMR according to:

  | 31 | 26 | 25 | 21 | 20 | 16 | 15 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
  |----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|
  | BK1| BK0| B7 | B6 | B5 | B4 | A7 | A6 | A5 | A4 |

  - BK0/BK1 = N
  - Block size (bytes) = \(2^{(N+1)}\)

- BK0 and BK1 defines the block size of the buffer as \(2^{(N+1)}\) bytes
- C6x allows two independent circular buffer sizes in powers of 2
- Choose which 8 registers (A4-7, B4-7) as circular buffer pointers

Example: A4 as a pointer for buffer size of 256 bytes

```assembly
asm("MVK.S2 0001h,B2");  // A4 used as pointer
asm("MVKLH.S2 0007h,B2"); // use BK0, size is 256
asm("MVC.S2 B2,AMR");    // setup AMR
```