# Chapter 6 Supplement Material

## 6 1 *M*-Path Filter

In Section 6.1 we considered a narrow bandwidth 399 tap FIR filter. The filter specifications shown in Figure 6.1 are, two sided 0.05 dB bandwidth of 20 kHz, -80 dB attenuation bandwidth of 40 kHz, and sample rate of 1000 kHz. Recognizing the large ratio of sample rate to bandwidth we reconfigured the filter into a 20-path filter shown in Figure 6.2. This filter reduced the bandwidth and simultaneously reduced the sample rate to fs/20 or 50 kHz. We then formed a dual graph filter which performed 1-to-20 up-sampling, or interpolating, raising the sample rate back to the input rate without altering the frequency response formed by the input filter. The frequency response and the time response of the cascade 20-path filters is shown in Figure 6.3 The workload of the cascade filter is 20 operations (ops) per input sample, and 20 ops per output sample or 40 ops per input. This is a 10-to-1 reduction in workload relative to the direct single filter implementation. The cost we incur to reduce the workload by a factor of 10 is an increase in filter time delay. The delay of the direct FIR filter is 200 samples and the delay of the cascade filter pair is 360 samples. The increase in delay is due to passing the input signal through two filters, one at the put and one at the output.

We can reduce this delay as well as reduce the workload by modifying our approach to the resampling process. Rather than converting the original filter to an *M*-path filter we design an auxiliary filter to lower the sample rate P-to-1 and apply the down sampled signal to a second filter that matches the specifications of the original filter except designed for the reduced sample rate fs/P. The output of the second filter, now called the inner filter is up-sampled then 1-to-P by an up-sampling filter. A modified form of figure 6.2 can be seen in Figure S6.1 where we show that the input and output P-path filters simply perform sample rate changes for a reduced length inner filter which performs the desired filtering task at a reduced input sample rate. In this example the sample rate is reduced 10-to-1 to 100 kHz by an 80-tap 10-path input filter. Its output is processed by a 39-tap inner filter that performs the reduced workload filtering and passes along the bandwidth limited samples to the 80-tap 10-path output filter. This filter performs a 1-to-10 sample rate increase as an interpolation process which forms output samples matched to the original input rate. The inner filter operating at1/10th of the input sample rate with 1/10th of the number of taps as the original filter has a processing workload of 1/100th of the original filter's workload.

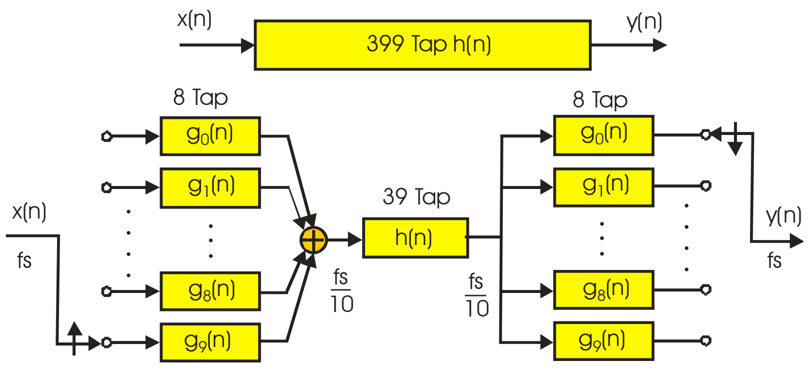

Figure S6.1 Implementing a narrow bandwidth filter as a cascade of an input 10-to-1 10-path filter, an inner filter, and an output 1-to-10 10-path filter

Figure S6.2 shows the time and frequency response of the three cascade filters. The first pleasant surprise is the net reduction in workload to implement the narrowband filter. The workload here is 20 ops as opposed to the original 400 ops and the previous reduction to 40 ops. The time delay has also been reduced. The delay here is 259 samples, an increase from the original 199 but reduced considerably from the 360 of the previous reduced workload. We note the reduced delay even though we have traversed three filters in this cascade.

The lesson to be learned in this example is that when implementing a digital FIR filter, workload reduction exceeding an order of magnitude can be had if we first perform a course bandwidth reduction with its commensurate sample rate reduction with an outer filter and then perform the desired bandwidth reduction at the reduced sample rate. We then use the outer filter interpolator to return to the original input sample rate or to any other desired sample rate.

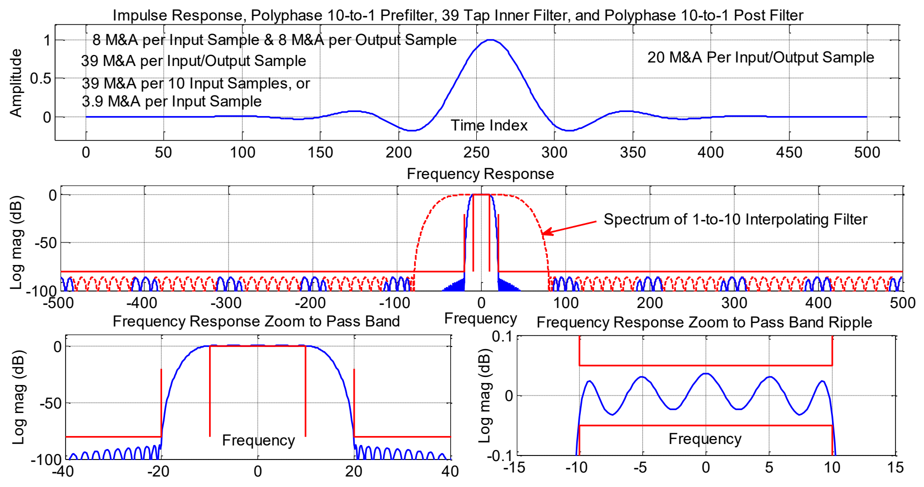

Figure S6.2. Time and frequency response of cascade 10-to-1 down-sampling filter, inner filter, and 1-to-10 up-sampling filters

A requirement to access this significant workload reduction is a sample rate reduction as part of the input bandwidth reduction, a condition assured when there is large ratio of sample rate to bandwidth. It would seem that this option is not available when this condition is not met, such as when the sample rate to bandwidth ratio is small, such 1.5 or 2.2. The surprise is that the option is still valid for this later case. We can use an analysis channelizer to partition the input bandwidth into narrow bandwidth segments for which there is large ratio of sample rate to bandwidth. Thus the computational savings can then be had for wide bandwidth signals partitioned temporarily into narrow bandwidth signals which are then reassembled by a synthesis channelizer.

## S6.2 Phase Aligned Non-Maximally Decimated Filter Bank

This supplement addresses an interesting variant of the maximally decimated filter bank. As we described in section 6.3, the spectral bands centered on multiples of *fs/M* alias to baseband when we perform *M-to-1* down-sampling in our *M*-path filter. The *M-to-1* down sampling occurs at the input commutator where we deliver *M*-input samples to the *M* -path filter and then compute an *M* -point output vector. When we deliver less than *M* input points, say *M/2* input points to the *M* -path filter and then compute the *M* -point output vector we raise the output sample rate from *fs/M* to *fs/(M/2)* or *2fs/M*. This increase in sample rate changes the filter bank from a maximally decimated to a non-maximally decimated filter bank.

With an increased output sample rate the channel centers at *fs/M* no longer alias to baseband. The channels still alias as a result of the down sampling, but they alias to other known offset frequencies. Successive samples from each channel port are spinning due to the aliased frequency offset and we can complete the conversion to baseband by simply de-spinning the successive samples with the conjugate complex phasor.

We have another option which proves the worth of understanding properties of linear systems. We know that time shift in the time domain is responsible for frequency dependent phase shift in the frequency domain. Thus rather than apply a phase rotation to the output ports of each channel output, we can obtain the same phase correction by appropriate end-around rotations of a circular buffer containing the time samples to be processed by the IFFT. The state machine we described earlier will now control the end around rotation of the circular buffer. Use of the circular buffer is performed as an addressing mechanism. In reality, there is no real shifting of data in the buffer. We note that the circular buffer at the input to the IFFT requires no additional processing as opposed to the phase correction terms applied to the output vectors formed at the output of the IFFT.

Figure S6.3 illustrates how the state machine tracks a specific input sample through the serpentine shifts on successive inputs of 4 input samples presented to a six path filter. We deliver the first 4 input samples, declare the sample at the top of array our temporary origin and the data state, state 0. On the next shift of 4 input samples, the tagged sample is moved down to the 4-th row of the buffer. We label this as the next state, state 1, compute the path outputs of the 6-path filter and then shift the output vector down 2-samples to roll the entry with the tagged sample back to the top of the array. When the next 4 samples are input to the array, the serpentine shift of data in the array slides our tagged sample to row 2 in the next column. We label this next state, state 2, compute the path outputs of the 6-path filter and then shift the output vector down 4 samples to roll the entry with the tagged sample back to the top of the array. When the next 4 input samples are input to the array, the serpentine shift slides the tagged sample to the first row in the next column. This where we started. We recognize this as state zero and declare the current sample at the top of the first column as our new temporary origin.

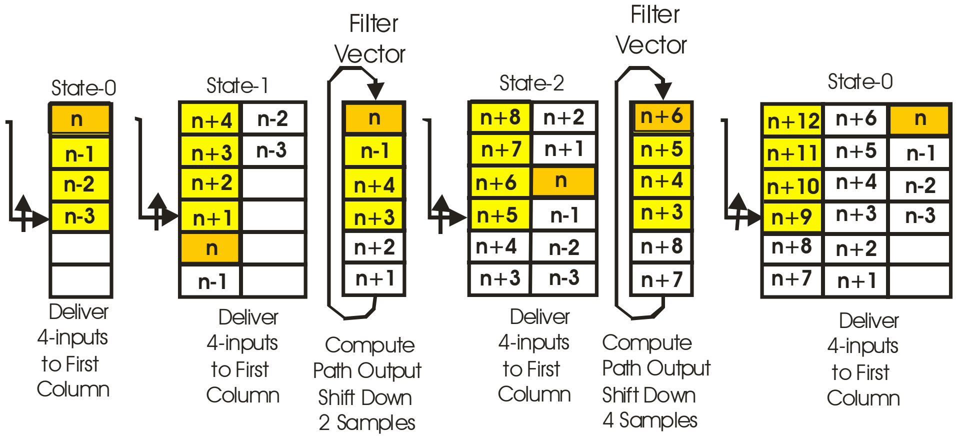
State-0
n
n-1
n-2
n-3
Deliver 4-inputs to First Column
State-1
n+4
n-2
n+3
n-3
n+2
n+1
n
n-1
Deliver 4-inputs to First Column
Filter Vector
n
n-1
n+4
n+3
n+2
n+1
Compute Path Output Shift Down 2 Samples
State-2
n+8
n+2
n+7
n+1
n+6
n
n+5
n-1
n+4
n-2
n+3
n-3
Deliver 4-inputs to First Column
Filter Vector
n+6
n+5
n+4
n+3
n+8
n+7
Compute Path Output Shift Down 4 Samples
State-0
n+12
n+6
n
n+11
n+5
n-1
n+10
n+4
n-2
n+9
n+3
n-3
n+8
n+2
n+7
n+1
Deliver 4-inputs to First Column

Figure S6.3. Deliver 4-input samples to serpentine data array. Monitor origin, top of array on successive shifts. Compute path row outputs, perform circular shift to return origin row to top of array

## S6.3 Nyquist Filter Design

In Section 6.3 we recognized that the cascade $M$ -path channelizers must exhibit a Nyquist filter spectrum to support perfect reconstruction at its output signal spectrum. Communication systems require the Nyquist filter response at the receiver to avoid inter-symbol interference (ISI). They achieve this desired property by performing half of the spectral shaping at the transmitter and half at the receiver with sqrt-Nyquist filters. They do this to minimize the effect of noise added to the signal in the communication channel. We indicated that performing half of the spectral shaping with sqrt-Nyquist filters in each of the filter banks would not lead to satisfactory performance of the cascade filter banks. This is because the sqrt Nyquist filter is a poor filter in terms of its levels of in-band and out-of-band ripple which degrade the perfect reconstruction property we require in our filter banks. To achieve high performance characteristics of reconstruction process we chose to implement a Nyquist filter in one channelizer and implement a wider bandwidth filter in the second channelizer to preserve the spectrum of the Nyquist filter. The spectra of the two filters is shown in Figure 6.8. The second filter is designed with a modified Remez algorithm with care to control the in-band ripple level which degrades the perfect reconstruction process.

We now address the question of how do we design the Nyquist filter? Remarkably simple: we use a windowed sinc function which has the form shown in (S6.1). In the sinc sequence, the distance between zero crossings is the reciprocal bandwidth and the distance between samples is the reciprocal sample rate. Using *Nz* as the number of zero crossings offset from the origin, we can use the argument index vector, *-Nz:1/M:Nz*, as the argument of the sinc function. This argument instructs the sinc script to form samples spanning the interval *–Nz to +Nz* zeros crossings in increment step sizes of *1/M*. Counting the matching left and right samples and the center valued sample the filter length obtained by this indexing is *(2·Nz·M)+1*. We want an odd number of samples but as an implementation consideration, we want it to be one less than *M* times an integer. We obtain this number by removing one sample from each side of the index by starting at *–Nz +1/M* and ending at *+Nz -1/M* and obtain a filter length *(2·Nz·M)-1* where *2·Nz* are integers, i.e. 6, 7, 8, …., which are the number of taps per path of the *M*-path filter. We select an *Nz*, as multiples of 0.5, and set the length of the Kaiser window to the length of the sinc and scale the product for unity passband gain.

```
hh=sinc(-Nz:1/M:+Nz).*kaiser(2*Nz*M+1,beta)';
hh=sinc(-Nz+1/M:1/M:Nz-1/M).*kaiser(2*Nz*M-1,beta)';      (S6.1)
hh=hh/M;
```

The parameter beta in the Kaiser window argument is the window's time bandwidth product. As this parameter increases, the spectral main lobe width increases and the spectral side lobe levels decrease. The spectral main lobe affects the transition bandwidth of the windowed Nyquist filter. After applying the window, we examine the spectrum and adjust the value of beta till the edge of its transition bandwidth meets the stopband spectral mask and examine the level of stopband side lobe. If the side lobe levels are above the required stopband level we have to increase beta. An increased beta increases the window's main lobe width which in turn increases the widowed filter's transition bandwidth. We respond by increasing the filter length by incrementing *Nz.* We stop when the stop band attenuation is below the targeted design level when the transition band edge is touching the stopband spectral mask. This process is illustrated in Figure S6.4. When the filter length and window parameter have been determined, we append a zero to the filter so the total number of weights is an integer multiple of *M* the number of paths.

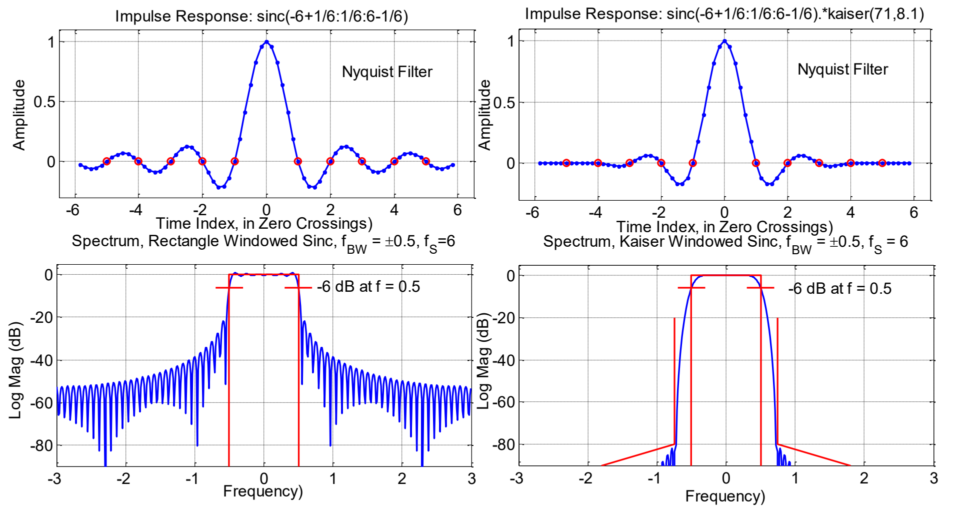

Figure S6.4 Rectangle windowed and Kaiser windowed sinc filter and their spectra with spectral masks

Using normalized frequencies for the passband edge, stopband edge and sample rate of 0.5, 0.75 and 3, we ran through this design process for increasing filter length to obtain the relationship between *Nz*, beta, and attenuation that is shown in table 6.1. If our design criterion for the analysis filter required 90 dB or better side lobe attenuation we would select a 12 taps per path, 71 tap filter with Kaiser window parameter slightly below 9.4

Table S6.1. Stopband attenuation as function of filter length with required beta for that attenuation

| **Ntaps/path** | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|
| **Filter Size** | 35 | 41 | 47 | 53 | 59 | 65 | 71 | 77 |
| **Atten (dB)** | 47.1 | 54.9 | 60.0 | 68.8 | 75.0 | 84.8 | 93.2 | 103.4 |
| **Nz** | 3.0 | 3.5 | 4.0 | 4.5 | 5.0 | 5.5 | 6.0 | 6.5 |
| **Beta** | 4.0 | 5.0 | 5.8 | 6.6 | 7.5 | 8.4 | 9.4 | 10.4 |

The script we used to design the Nyquist filter is shown in (S6.2). The scaling coefficient "6" sets the filter's passband gain to 1 but is removed in the reshape command because we want the path gains to be 1. The reshape command appends a leading 0 to bring the number of coefficients to 72, a multiple of 6, and partitions the prototype impulse response into a 6-path filter with 12 coefficients per path containing coefficients with indices h(r+6n), r = 0, 1, 2, 3, 4, 5 for the row index.

```
hh=sinc(-6+1/6:1/6:6-1/6).*kaiser(71,9.2)';% Nyquist Filter        (S6.2)
hh=hh/6;                                   % Scaling
hh2=reshape([0 6*hh],6,12);                % 6-Path Partition
```

The coefficients of the polyphase partition are shown in Table S6.2 where the rows are labeled 0 through 5. Interesting to note that the 0-th row contains all zeros except for the 1 in column 6. You would expect this in an *M*-path Nyquist filter, we have seen a similar relationship in a true half band filter where a 2-path polyphase partition has the top row all zeros with a single centered value of 0.5 (or 1 if scaled). The remaining rows 1 through 5 are mirror images of their rows 5 through 1 with row 3 being its own mirror image.

Table S6.2. Coefficients of 6-Path Filter with 12 Coefficients per Row

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0.0000 | -0.0000 | 0.0000 | -0.0000 | 0.0000 | 1.0000 | -0.0000 | 0.0000 | -0.0000 | 0.0000 | -0.0000 |
| 1 | -0.0000 | 0.0008 | -0.0050 | 0.0189 | -0.0559 | 0.1747 | 0.9516 | -0.1145 | 0.0395 | -0.0126 | 0.0029 | -0.0003 |
| 2 | -0.0001 | 0.0019 | -0.0111 | 0.0397 | -0.1152 | 0.3906 | 0.8153 | -0.1643 | 0.0573 | -0.0176 | 0.0037 | -0.0003 |
| 3 | -0.0002 | 0.0032 | -0.0162 | 0.0552 | -0.1585 | 0.6166 | 0.6166 | -0.1585 | 0.0552 | -0.0162 | 0.0032 | -0.0002 |
| 4 | -0.0003 | 0.0037 | -0.0176 | 0.0573 | -0.1643 | 0.8153 | 0.3906 | -0.1152 | 0.0397 | -0.0111 | 0.0019 | -0.0001 |
| 5 | -0.0003 | 0.0029 | -0.0126 | 0.0395 | -0.1145 | 0.9516 | 0.1747 | -0.0559 | 0.0189 | -0.0050 | 0.0008 | -0.0000 |

We now address the filter designed for the synthesis channelizer. The prototype for the synthesis filter is designed using a variation of the Remez algorithm. The variation uses a modified weight vector that forms an $L_\infty$, Chebyshev, or equal ripple, passband approximation to unity gain passband and a 1/f or -6dB/octave decay rate for its stopband ripple. The MATLAB script code for the variation is presented in the 8-th script file in this supplement. The MATLAB script for the synthesis filter of a 6-path channelizer with output sample rate of 120 MHz is shown in (S6.3).

```
                                             % Synthesis Filter
gg=remez(70,[0 15 25 60]/60,{'myfrf',[1 1 0 0]},[1 1]);
gg2=[reshape([0 gg],6,12);                   % 6-path partition
```
(S6.3)

We can now examine the spectral response of the Nyquist filter and the reconstruction filter used in the analysis and synthesis filter banks. The top subplot of Figure S6.5 shows the time response of the Nyquist filter with each 6-th sample marked by a red circle to tag the zeros of the Nyquist time series. These zeros effect the polyphase partition by establishing a row consisting of zeros and a single centered 1. The center subplot shows the frequency response of the filter with the spectral masks indicating the -6 dB passband between ±10 MHz and the -90 dB stopband at ±15 MHz. The three bottom subplots show spectral detail important to the filter response. The bottom left most subplot shows the cross-over response at 10 MHz of adjacent channel responses. As expected, the crossover level is -6.02 dB. The bottom center subplot shows the pass band ripple pass in the ±5 MHz frequency span. This span width is the result of the odd symmetric transition band about the -6 dB 10 MHz Frequency. The level of the ripple, 0.0002 dB, is important because it contributes to the reconstruction error level when we merge adjacent channel spectra. The bottom right most subplot shows the transition bandwidth of the Nyquist filter and the spectral stopband mask at 15 MHz.

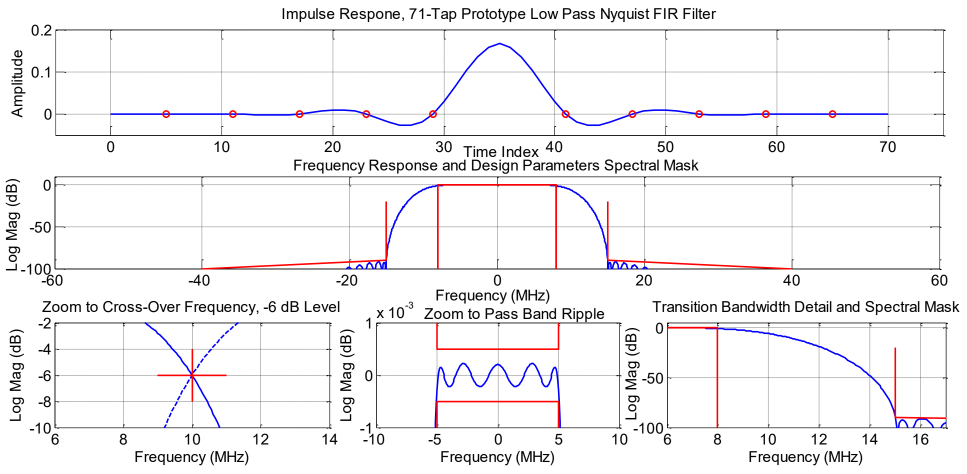

Figure S6.5 Impulse response and frequency response of Nyquist filter of analysis filter bank along with details of crossover levels, in-band ripple levels and transition bandwidth

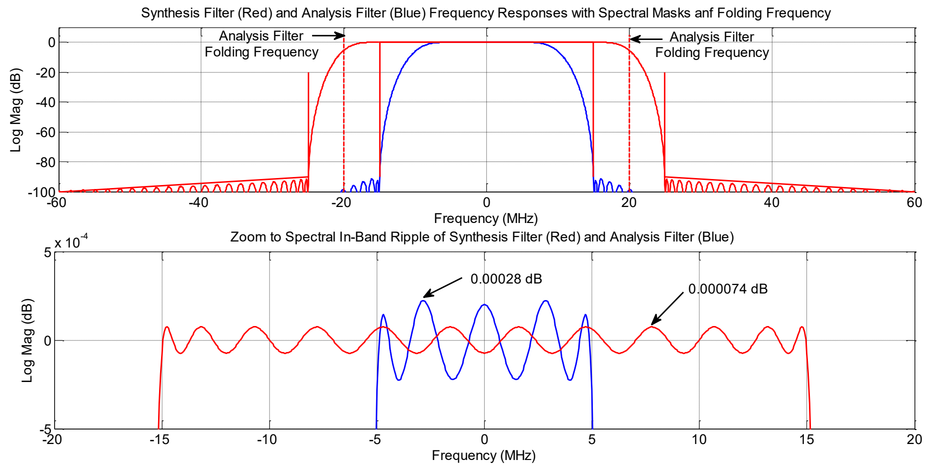

Figure S6.6 Frequency response of reconstruction filter in the synthesis filter bank bracketing frequency response of analysis filter with spectral masks, folding frequencies, and zoom to passband ripple

Figure S6.6 shows the frequency response of the reconstruction filter with the spectral masks of its passband and stopband at 15 and 25 MHz respectively. It is shown as an overlay on the spectral response of the Nyquist filter to illustrate how well the passband fits over the full two-sided bandwidth of the Nyquist filter. The bottom subplot is a zoom to the in-band ripple levels of the reconstruction filter and of the Nyquist filter. Note that both filters have the 10 MHz transition bandwidth and the same 90 dB stopband attenuation level but differ in their passband ripple levels. This is because the Nyquist filter symmetry require the passband ripple and the stop band ripple to have the same values (relative to their target levels of unity and zero). The reconstruction filter response is not so restricted and the penalty weights in the Remez algorithm allow us to set different levels of in band and out of band ripple. Remember the combination of the two ripples become the reconstruction error as the input signal passes through both analysis and synthesis filters.

## S6.4 MATLAB Channelizer Script

1. Maximally Decimated 6-Path Analysis Channelizer
2. Non-Maximally Decimated 6-Path Analysis Channelizer
3. Maximally Decimated 6-Path Synthesis Channelizer
4. Non-Maximally Decimated 6-Path Synthesis Channelizer
5. Non-Maximally Decimated 6-Path Cascade Analysis and Synthesis Channelizers

   Impulse Response, Full Bandwidth Cascade
6. Non-Maximally Decimated 6-Path Cascade Analysis and Synthesis Channelizers

   Impulse Response, Synthesized Three Channel Super-Channel
7. Study of Individual Path Time, Frequency, and Phase Responses and of

   Individual Channel Time, Frequency, and Phase Responses of 6-Path Analysis

   Filter
8. Script File myfrf.m for 1/f Remez filter Stopband Used in Synthesis filter design

```
% 1 %%%% Maximally Decimated 6-path Analysis Channelizer %%%%%%%%

hh=sinc(-6.0+1/6:1/6:6.0-1/6).*kaiser(71,9.2)';   %Prototype
Filter
hh=hh/6;
hh2=reshape(6*[hh 0],6,12);  % Polyphase partition of Filter
Weights

reg=zeros(6,12);             % Input Sample data Array for 6-path
filter
v1=zeros(1,6)';              % 6 Sample Input Vector
v2=zeros(1,6)';              % 6 Path Filter Output Vector
v3=zeros(1,6)';              % Channelizer Output Vector, IFFT of
Filter Vector
v4=zeros(6,100);             % Time Series Output Buffer
x=[0 1 zeros(1,600)];          % Input Sequence to Filter

mm=1;                          % Output Clock Index
for n=1:6:length(x)-6          % For Loop, 6-Input Samples
  v1(1:6)=fliplr(x(n:n+5)).';      % Re-order, start Bottom, go to
top
  reg=[v1 reg(:,1:11)];          % Insert Vector in Filter Delay
Line
    for k=1:6                  % Compute 6 Path Outputs
        v2(k)=reg(k,:)*hh2(k,:)';
    end
  v3=6*ifft(v2);               % Phase Coherent Sum,
Channelizer Output
  v4(:,mm)=fftshift(v3);       % Store Channel Output Samples
  mm=mm+1;                     % Increment Output Clock
```

```
end

figure(1)
subplot(3,1,1)
plot((-0.5:1/500:0.5-
1/500)*6,fftshift(20*log10(abs(fft(hh,500)))),'linewidth',2)
hold on
plot([-0.5 -0.5 +0.5 +0.5],[-100 0 0 -100],'r','linewidth',2)
hold off
grid on
axis([-3 3 -100 10])
title('Spectrum, Prototype Low Pass Filter, 6-Path Maximally
Decimated Filter Bank')
xlabel('Frequency')
ylabel('Log Mag (dB)')

for k=1:6
    subplot(3,3,k+3)
    plot(real(v4(k,:)))
    plot((-0.5:1/500:0.5-
1/500)*1,fftshift(20*log10(abs(fft(v4(k,:),500)))),'linewidth',2)
    grid on
    axis([-0.5 0.5 -2 1])
    title(['Spectrum, Channel(',num2str(k-4),')'])
    xlabel('Frequency')
    ylabel('Log Mag (dB)')
end




% 2 % Non-Maximally Decimated 6-path Analysis Channelizer %%%%%%%%

hh=sinc(-6.0+1/6:1/6:6.0-1/6).*kaiser(71,9.2)';   %Prototype
Filter
hh=hh/6;
hh2=reshape(6*[hh 0],6,12);  % Polyphase partition of Filter
Weights

reg=zeros(6,12);          % Input Sample data Array for 6-path
filter
v1=zeros(1,3)';           % 3 Sample Input Vector
v2=zeros(1,6)';           % 6 Path Filter Output Vector
v3=zeros(1,6)';           % Channelizer Output Vector, IFFT of
Filter Vector
v4=zeros(6,200);          % Time Series Output Buffer
flg=0;                       % Circular Buffer Flag
x=[0 1 zeros(1,100)];            % Input Sequence to Filter

mm=1;                             % Output Clock Index
for n=1:3:length(x)-6             % For Loop, 3-Input Samples
  v1(1:3)=fliplr(x(n:n+2)).';     % Re-order, start center, go to
top
  reg=[reg(4:6,:);reg(1:3,:)];    % Roll Buffer for Serpentine
Shift
  reg(1:3,:)=[v1 reg(1:3,1:11)];  % Insert Vector in Filter Delay
Line
    for k=1:6                     % Compute 6 Path Outputs
        v2(k)=reg(k,:)*hh2(k,:)';
    end
    if flg==0;                          % Test flag for end around shift
        flg=1;                          % no shift, set flag for next
test
    else
        flg=0;                          % set flag for next test
        v2=[v2(4:6);v2(1:3)];           % perform end around shift
    end
  v3=3*ifft(v2);                        % Phase Coherent Sum,
Channelizer Output
  v4(:,mm)=fftshift(v3);                % Store Channel Output Samples
  mm=mm+1;                              % Increment Output Clock
end

figure(2)
subplot(3,1,1)
plot((-0.5:1/500:0.5-
1/500)*6,fftshift(20*log10(abs(fft(hh,500)))),'linewidth',2)
hold on
plot([-0.5 -0.5 +0.5 +0.5],[-100 0 0 -100],'r','linewidth',2)
plot([-1.0 -1.0],[-100 10],'--r','linewidth',2)
plot([+1.0 +1.0],[-100 10],'--r','linewidth',2)
hold off
grid on
axis([-3 3 -100 10])
title('Spectrum, Prototype Low Pass Filter, 6-Path Non-Maximally
Decimated Filter Bank')
xlabel('Frequency')
ylabel('Log Mag (dB)')

for k=1:6
    subplot(3,3,k+3)
    plot(real(v4(k,:)))
    plot((-0.5:1/500:0.5-
1/500)*2,fftshift(20*log10(abs(fft(v4(k,:),500)))),'linewidth',2)
    grid on
    axis([-1 1 -100 10])
    title(['Spectrum, Channel(',num2str(k-4),')'])
    xlabel('Frequency')
    ylabel('Log Mag (dB)')
end



% 3 %% Maximally Decimated 6-path Synthesis Channelizer %%%%%%%%

hh=remez(94,[0 8 12 60]/60,{'myfrf',[1 1 0 0]},[1 200]);
%Prototype Filter
hh2=reshape(6*[0 hh],6,16);       % Polyphase partition of Filter
Weights

v4=zeros(6,100);                 % Input array of 6-pnt vectors
u3=zeros(1,6)';                 % Input vector to IFFT
u2=zeros(1,6)';                 % Output vector from IFFT
reg=zeros(6,16);                % Input Sample data Array for 6-path
filter
u1=zeros(1,6)';                 % 6 output samples from channelizer
y=zeros(1,500);                 % Output Sequence from Channelizer
```

```
w2=kaiser(100,14)'/100; w1=kaiser(39,14)'/4;
v4(5,20)=1;                                   % impulse bin    -
2
v4(1,20)=1;                                   % impulse bin
0
v4(3,1:39)=sinc(-5+1/4:1/4:5-1/4).*w1;        % sinc pulse bin
+2
v4(6,1:39)=sinc(-6-1/3:1/3:6+1/3).*w1;        % sinc pulse bin -
2
v4(2,1:100)=exp(j*2*pi*(0:99)*0.1).*w2;       % cmplx tone bin +1
mm=1;                           % Output Clock Index
for n=1:length(v4)              % For Loop, 1-Input 6-sample
vector
  u3=v4(:,n);                   % Input vector to IFFT
  u2=ifft(u3);                  % Interpolated output from IFFT
reg=[u2 reg(:,1:15)];           % Insert Vector in Filter Delay
Line
    for k=1:6                     % Compute 6 Path Outputs
        u1(k)=reg(k,:)*hh2(k,:)';
    end
  y(mm:mm+5)=u1.';                % Store 6 Channel Output Samples
  mm=mm+6;                        % Increment Output Clock
end

figure(3)
subplot(3,1,1)
plot((-0.5:1/600:0.5-
1/600)*6,fftshift(20*log10(abs(fft(y,600)))),'linewidth',2)
hold on
plot([-0.5 -0.5 +0.5 +0.5],[-100 0 0 -100],'r','linewidth',2)
hold off
grid on
axis([-3 3 -100 10])
title('Spectrum, Prototype Low Pass Filter, 6-Path Non-Maximally
Decimated Filter Bank')
xlabel('Frequency')
ylabel('Log Mag (dB)')

qq=[4 5 6 1 2 3];
aa1=[-20 -20 -20 -20 -50 -20];
aa3=[-0.5 -0.1 -0.05 -0.1 -0.012 -0.05];
aa4=[0.5 1.1 0.3 1.1 0.012 0.3];
for k=1:6
    subplot(3,3,k+3)
    if k==5
    plot(-50:49,real(v4(qq(k),1:100)),'linewidth',2)
    hold on
     plot(-50:49,imag(v4(qq(k),1:100)),'r','linewidth',2)
    hold off
    else
    plot(-19:19,real(v4(qq(k),1:39)),'linewidth',2)
    end
    grid on
    axis([aa1(k) -aa1(k) aa3(k) aa4(k)]);
    title(['Input Time Series, Channel(',num2str(k-4),')'])
    xlabel('Time Index')
    ylabel('Amplitude')
end
```

```
% % 4 %Non-Maximally Decimated 6-path Synthesis Channelizer %%%%%%

hh=remez(94,[0 8 12 60]/60,{'myfrf',[1 1 0 0]},[1 200]);
%Prototype Filter
hh2=reshape(6*[0 hh],6,16);      % Polyphase partition of Filter
Weights

v4=zeros(6,200);           % Input array of 6-pnt vectors
u3=zeros(1,6)';              % Input vector to IFFT
u2=zeros(1,6)';              % Output vector from IFFT
reg=zeros(6,32);             % Input Sample data Array for 6-path
filter
u1=zeros(1,3)';              % 3 output samples from channelizer
y=zeros(1,600);              % Output Sequence from Channelizer

w0=kaiser(63,10)'/12;
w1=kaiser(39,10)'/6;
w2=kaiser(100,10)'/100;
v4(5,20)=1;                                  % impulse bin    -
2
v4(1,20)=1;                                  % impulse bin
0
v4(3,1:63)=sinc(-2.5-1/12:1/12:2.5+1/12).*w0;   % sinc pulse bin
+2
v4(6,1:39)=sinc(-3-1/6:1/6:3+1/6).*w1;          % sinc pulse bin -
2
v4(2,1:100)=exp(j*2*pi*(0:99)*0.05).*w2;        % cmplx tone bin
+1
mm=1;                                        % Output Clock
Index
flg=0;
for n=1:length(v4)               % For Loop, 6-Input Samples
  u3=v4(:,n);                    % Input vector to IFFT
  u2=ifft(u3);                 % Interpolated output from IFFT
  if flg==0;                     % test for circular shift
      flg=1;                     % set flag for next test
  else
      flg=0;                     % set flag for next test
      u2=[u2(4:6);u2(1:3)];      % perform circular shift
  end
reg=[u2 reg(:,1:31)];            % Insert Vector in Filter Delay
Line
    for k=1:3                    % Compute 3 Path Outputs
      p1=reg(k,1:2:32)*hh2(k,:)';
      p2=reg(k+3,2:2:32)*hh2(k+3,:)';
      u1(k)=(p1+p2);
    end
  y(mm:mm+2)=u1.';                 % Store 3 Channel Output Samples
  mm=mm+3;                         % Increment Output Clock
end


figure(4)
subplot(3,1,1)
plot((-0.5:1/600:0.5-
1/600)*6,fftshift(20*log10(abs(fft(y,600)))),'linewidth',2)
hold on
plot([-0.5 -0.5 +0.5 +0.5],[-100 0 0 -100],'r','linewidth',2)
hold off
grid on
```

```
axis([-3 3 -100 10])
title('Spectrum, Prototype Low Pass Filter, 6-Path Non-Maximally
Decimated Filter Bank')
xlabel('Frequency')
ylabel('Log Mag (dB)')

qq=[4 5 6 1 2 3];
aa1=[-20 -20 -20 -20 -50 -34];
aa3=[-0.5 -0.1 -0.05 -0.1 -0.012 -0.02];
aa4=[0.5 1.1 0.2 1.1 0.012 0.1];
for k=1:6
    subplot(3,3,k+3)
    if k==5
    plot(-50:49,real(v4(qq(k),1:100)),'linewidth',2)
    hold on
     plot(-50:49,imag(v4(qq(k),1:100)),'r','linewidth',2)
    hold off
    elseif k==6
    plot(-31:31,real(v4(qq(k),1:63)),'linewidth',2)
    else
    plot(-19:19,real(v4(qq(k),1:39)),'linewidth',2)
    end
    grid on
    axis([aa1(k) -aa1(k) aa3(k) aa4(k)]);
    title(['Input Time Series, Channel(',num2str(k-4),')'])
    xlabel('Time Index')
    ylabel('Amplitude')
end




% 5 % Cascade 3-to-1, and 1-to-3 Non-Maximally Decimated %%%%%%
      6-path Analysis and synthesis Channelizer %%%%%%%%
      Impulse Response, Full Channel Bandwidth %%%%%%%%%

hh=sinc(-6+1/6:1/6:6-1/6).*kaiser(71,9.2)';
   hh=hh/6;
gg=remez(70,[0 15 25 60]/60,{'myfrf',[1 1 0 0]},[3 1]);
   hh2=reshape(6*[0 hh],6,12);
   gg2=reshape([0 gg],6,12);
reg1=zeros(6,12);
reg2=zeros(6,24);
    v1=zeros(1,3)';
    v2=zeros(1,6)';
    v3=zeros(1,6)';
    v4=zeros(6,33);
    u3=zeros(1,6)';
    u2=zeros(1,6)';
flg1=0;
flg2=0;
    x=zeros(1,160);
    y=zeros(1,160);
    x(1)=1;
m1=1;
m2=0;

for n=1:3:160-3
    v1(1:3)=fliplr(x(n:n+2)).';
    reg1=[reg1(4:6,:);reg1(1:3,:)];
    reg1(1:3,:)=[v1 reg1(1:3,1:11)];
        for k=1:6
           v2(k)=reg1(k,:)*hh2(k,:)';
        end
        if flg1==0;
           flg1=1;
        else
           flg1=0;
           v2=[v2(4:6);v2(1:3)];
      end

    v3=3*ifft(v2);
    v4(:,m1)=fftshift(v3);
    m1=m1+1;
    v3=v3.*[1 1 0 0 0 1]';  % binary mask
    u3=v3;
    u2=6*ifft(u3);
%
    if flg2==0;
        flg2=1;
    else
        flg2=0;
        u2=[u2(4:6);u2(1:3)];
    end

   reg2=[u2 reg2(:,1:23)];

    for k=1:3
        p1=reg2(k,1:2:24)*gg2(k,:)';
        p2=reg2(k+3,2:2:24)*gg2(k+3,:)';
        y(m2+k)=p1+p2;
    end

    m2=m2+3;
end


figure(5)
subplot(3,1,1)
plot(0:144,y(1:145),'linewidth',2)
grid on
axis([-5 150 -0.1 1.1])
title('Impulse Response, Cascade 3-to-1 Downsample 6-Path Analysis
Filter Bank and 1-to-3 6-Path Synthesis Filter Bank')
xlabel('Time Index')
ylabel('Amplitude')

subplot(3,1,2)
plot(0:144,y(1:145),'linewidth',2)
grid on
axis([-5 150 -0.00002 .00002])
title('Zoom to Reconstruction Artifacts Levels of Cascade Filter
Bank Impulse Response')
xlabel('Time Index')
ylabel('Amplitude')
text(49,-0.00001,'Artifacts 5 Orders of Magnitude Below Desired
Signal Level','fontsize',14 )

subplot(3,1,3)
```

```
plot((-0.5:1/600:0.5-
1/600)*6,fftshift(20*log10(abs(fft(y,600)))),'r','linewidth',3)
hold on
plot((-0.5:1/600:0.5-
1/600)*6,fftshift(20*log10(abs(fft(hh,600)))),'linewidth',2)
plot((-0.5:1/600:0.5-
1/600)*6+1,fftshift(20*log10(abs(fft(hh,600)))),'linewidth',2)
plot((-0.5:1/600:0.5-1/600)*6-
1,fftshift(20*log10(abs(fft(hh,600)))),'linewidth',2)
plot((-0.5:1/600:0.5-
1/600)*6+2,fftshift(20*log10(abs(fft(hh,600)))),'linewidth',2)
plot((-0.5:1/600:0.5-1/600)*6-
2,fftshift(20*log10(abs(fft(hh,600)))),'linewidth',2)

hold off
grid on
axis([-1.5 1.5 -0.0005 0.0005])
title('Spectrum, Zoom to Spectrum Ripple Levels of Impulse
Response and of Prototype Low Pass Filter Embedded in Analysis
Filter Bank')
xlabel('Frequency')
ylabel('Log Mag (dB)')
text(0.28, 0.00025,' Peak Ripple = 0.00019 dB','fontsize',14)




% 6 %% Cascade 3-to-1, and 1-to-3 Non-Maximally Decimated
       6-path Analysis and synthesis Channelizer %%%%%%%%

hh=sinc(-6+1/6:1/6:6-1/6).*kaiser(71,9.2)';
   hh=hh/6;
gg=remez(70,[0 15 25 60]/60,{'myfrf',[1 1 0 0]},[3 1]);
   hh2=reshape(6*[0 hh],6,12);
   gg2=reshape([0 gg],6,12);
reg1=zeros(6,12);
reg2=zeros(6,24);
    v1=zeros(1,3)';
    v2=zeros(1,6)';
    v3=zeros(1,6)';
    v4=zeros(6,33);
    u3=zeros(1,6)';
    u2=zeros(1,6)';
flg1=0;
flg2=0;
    x=zeros(1,160);
    y=zeros(1,160);
    x(1)=1;
m1=1;
m2=0;

for n=1:3:160-3
    v1(1:3)=fliplr(x(n:n+2)).';
    reg1=[reg1(4:6,:);reg1(1:3,:)];
    reg1(1:3,:)=[v1 reg1(1:3,1:11)];
        for k=1:6
           v2(k)=reg1(k,:)*hh2(k,:)';
        end
        if flg1==0;
           flg1=1;
        else
           flg1=0;
           v2=[v2(4:6);v2(1:3)];
       end

    v3=3*ifft(v2);
    v4(:,m1)=fftshift(v3);
    m1=m1+1;
    %v3=v3.*[1 1 0 0 0 1]';  % binary mask
    u3=v3;
    u2=6*ifft(u3);
%
    if flg2==0;
        flg2=1;
    else
        flg2=0;
        u2=[u2(4:6);u2(1:3)];
    end

   reg2=[u2 reg2(:,1:23)];

    for k=1:3
        p1=reg2(k,1:2:24)*gg2(k,:)';
        p2=reg2(k+3,2:2:24)*gg2(k+3,:)';
        y(m2+k)=p1+p2;
    end

    m2=m2+3;
end


figure(6)
subplot(4,1,1)
plot((-0.5:1/600:0.5-
1/600)*6,fftshift(20*log10(abs(fft(hh,600)))),'linewidth',2)
hold on
plot([-0.5 -0.5 +0.5 +0.5],[-100 0 0 -100],'r','linewidth',2)
hold off
grid on
axis([-3 3 -100 10])
title('Spectrum, Prototype Low Pass Filter, 6-Path Non-Maximally
Decimated Filter Bank')
xlabel('Frequency')
ylabel('Log Mag (dB)')

for k=1:6
    subplot(4,3,k+3)
    plot(-0.5:1/600:0.5-
1/600,fftshift(20*log10(abs(fft(v4(k,:),600)))),'linewidth',2)
    grid on
    axis([-1 1 -100 10]);
    title(['Frequency Response, Channel(',num2str(k-4),')'])
    xlabel('Frequency')
    ylabel('Log mag (dB)')
end

subplot(4,1,4)
plot([-0.5 -0.5 +0.5 +0.5],[-100 0 0 -100],'r','linewidth',2)
hold on
```

```
plot((-0.5:1/600:0.5-
1/600)*2,fftshift(20*log10(abs(fft(v4(1,:),600)))),'r','linewidth'
,2)
plot((-0.5:1/600:0.5-
1/600)*2+1,fftshift(20*log10(abs(fft(v4(1,:),600)))),'r','linewidt
h',2)
plot((-0.5:1/600:0.5-1/600)*2-
1,fftshift(20*log10(abs(fft(v4(1,:),600)))),'r','linewidth',2)
plot((-0.5:1/600:0.5-
1/600)*6,fftshift(20*log10(abs(fft(y,600)))),'linewidth',2)
hold off
grid on
axis([-3 3 -100 10])
title('Spectrum, Super Channel Formed by Merging 3 Channels, of 6-
Path Non-Maximally Decimated Filter Bank')
xlabel('Frequency')
ylabel('Log Mag (dB)')


% 6 Michelle_6_path
% Study of 6-Path Non-Maximally Decimated Analysis Bank
% Fig 1. Time and Freq Resp of 6-Path Analysis filter
% Fig 2. Imp Resp 6-Path Filter Prior to 6-to-1 Downsample
% Fig 3. Phase Resp 6-Path Filter Prior to 6-to-1 Downample
% Fig 4. Imp Resp 6-Channels of Polyphase Filter Bank
% Fig 5. Freq Resp 6-Channels of Polyphase Filter Bank
% Fig 6. Sliding Tone Time Resp 6-Paths Polyphase Filter
%           with Time Resp of Sum, Spectrum of Sum,
%           and Phases of Each Path
%           0-phase difference Nyquist Zone-0,
%           2 pi/6 phase difference Nyquist Zone-1
%           2 pi/3 phase difference Nyquist Zone-2
%           2 pi/2 phase difference Nyquist Zone-3
% Fig 7. Sliding Tone Time Resp of 6-Channels of
%        Polyphase Filter Bank
% Fig 8. Sliding Tone Freq Resp of 6-Channels of
%        Polyphase Filter Bank


% N1=(36/2)*80/22 =65
h0=remez(64,[0 1.5 4.5 18]/18,{'myfrf',[1 1 0 0]},[1 1]);
%h1=remez(64,[0 0.1 3.1 18]/18,{'myfrf',[1 1 0 0]},[1 1]);
f1=1.5;
f2=4.5;
figure(1)
subplot(3,1,1)
plot(-32:32,h0/max(h0),'linewidth',2);
grid on
axis([-35 35 -0.3 1.2])
title(['65-Tap Nyquist Filter, Windowed Sinc, 0.1-dB BW =
\pm',num2str(f1),' kHZ, -80 dB BW =\pm',num2str(f2),' kHz,
f_S = 36 kHz'],'fontsize',14)
xlabel('Time Index','fontsize',14)
ylabel('Amplitude','fontsize',14)
```

```
text(15,0.7,['0.1 dB BW = ',num2str(f1,'%5.1f'),'
kHz'],'fontsize',14)

subplot(3,1,2)
fh0=fftshift(20*log10(abs(fft(h0,2048))));
plot((-0.5:1/2048:0.5-1/2048)*36,fh0,'linewidth',2)
hold on
plot( [-f1 -f1 +f1 +f1],[-90 -0.1 -0.1 -90],'r--
','linewidth',2)
plot( [f2 f2 20],[-20 -80 -80],'r--','linewidth',2)
plot(-[f2 f2 20],[-20 -80 -80],'r--','linewidth',2)
hold off
grid on
axis([-18 18 -100 10])
title(['Spectrum; Passband 0-to-',num2str(f1),' kHz,
Stopband ',num2str(f2),'-to-18 kHz, 80 db
Attenuation'],'fontsize',14)
xlabel('Frequency (kHz)','fontsize',14)
ylabel('Log Mag (dB)','fontsize',14)

subplot(3,2,5)
plot((-0.5:1/2048:0.5-1/2048)*36,fh0,'linewidth',2)
hold on
plot( [-f1 -f1 +f1 +f1],[-0.0015 -0.001 -0.001 -0.0015],'r--
','linewidth',2)
plot( [-f1 -f1 +f1 +f1],[+0.015 +0.001 +0.001 +0.0015],'r--
','linewidth',2)
hold off
grid on
axis([-f1-1 f1+1 -0.0015 0.0015])
title('Zoom to Passband Ripple','fontsize',14)
xlabel('Frequency (kHz)','fontsize',14)
ylabel('Log Mag (dB)','fontsize',14)

subplot(3,2,6)
plot((-0.5:1/2048:0.5-1/2048)*36,fh0,'linewidth',2)
hold on
plot( [f1-1 +f1 +f1],[-0.1 -0.1 -90],'r--','linewidth',2)
plot( [f2 f2 20],[-20 -80 -80],'r--','linewidth',2)
hold off
grid on
axis([f1-1  f2+2 -100 5])
title('Transition BW and Stopband Detail','fontsize',14)
xlabel('Frequency (kHz)','fontsize',14)
ylabel('Log Mag (dB)','fontsize',14)

figure(2)
for k=1:6
    subplot(2,3,k)
    vv=zeros(1,65);
    vv(k:6:65)=6*h0(k:6:65);
    stem(0:64,vv,'linewidth',2)
    hold on
    plot(k-1:6:64,vv(k:6:65),'r*','linewidth',2)
    hold off
    grid on
    axis([0 64 -0.3 1.1])
end


figure(3)
plot(0,0)
hold on
for k=1:6
    vv=zeros(1,65);
    vv(k:6:65)=6*h0(k:6:65);
    ph_vv=unwrap(angle(fftshift(fft(vv,1024))))/(2*pi);
    plot((-0.5:1/1024:0.5-1/1024)*36,ph_vv-
ph_vv(513),'linewidth',2)
end
hold off
    grid on
    axis([-18 18 -18 18])
    set(gca,'xtick',[-18:3:18])
    set(gca,'fontsize',12)

    title('Phase Response of 6-Path Weights h(k:6:64),
k=0,1,...,5, Prior to 6-to-1 Downsample','fontsize',14)
    xlabel('Frequency','fontsize',14)
    ylabel('Phase','fontsize',14)
    text(-17.5,17.8,'\Delta Phase =
180^o','fontsize',14,'rotation',-25)
    text(-14,14.5,'6-Paths, \Delta Phase =
120^o','fontsize',14,'rotation',-25)
    text(-8,+9,'6-Paths, \Delta Phase =
60^o','fontsize',14,'rotation',-25)
    text(-2,3.,'6-Paths, \Delta Phase =
0^o','fontsize',14,'rotation',-25)
    text(+4.0,-1.5,'6-Paths, \Delta Phase =
60^o','fontsize',14,'rotation',-25)
    text(+10.0,-6.5,'6-Paths, \Delta Phase =
120^o','fontsize',14,'rotation',-25)
    text(+15.0,-10.3,'\Delta Phase =
180^o','fontsize',14,'rotation',-25)

%%%%%%%%%%%%%%% 6-to-1 downsampling filter, output sample
rate = 3
%%%%%%%%%%%%%%%%%%%%%%%%% polyphase filter %%%%%%%%%%%%%%
scl=max(h0);
hh0=reshape([h0/scl 0],6,11);

reg=zeros(3,22);
v0=zeros(1,3)';
v1=zeros(1,6)';
v2=zeros(6,25);
x0=zeros(1,75);
```

```
x0(1)=1;

m=1;
flg=0;
for n=1:3:75-3
    v0=fliplr(x0(n:n+2)).';
    reg=[v0 reg(:,1:21)];
        for k=1:3
          v1(k)=reg(k,1:2:22)*hh0(k,:)';
          v1(k+3)=reg(k,2:2:22)*hh0(k+3,:)';
        end
        if flg==0
            flg=1;
        else
            flg=0;
            v1=[v1(4:6);v1(1:3)];
        end
        v2(:,m)=ifft(v1);
        m=m+1;
end

figure(4)
for k=1:6
  subplot(3,2,k)
plot(0:22,6*real(v2(k,1:23)),'linewidth',2)
hold on
plot(0:22,6*imag(v2(k,1:23)),'r','linewidth',2)
hold off
grid on
axis([0 22 -1.1 1.1])
title(['Impulse Response, Chan(',num2str(k-
1),')'],'fontsize',14)
xlabel('Time Index','fontsize',14)
ylabel('Amplitude','fontsize',14)
end

figure(5)
for k=1:6
  subplot(3,2,k)
plot((-0.5:1/128:0.5-
1/128)*12,fftshift(20*log10(abs(3*fft(v2(k,:),128)))),'linew
idth',2)
hold on
plot([-3 -3 +3 +3],[-100 0 0 -100],'--r','linewidth',2)
hold off
grid on
axis([-6 6 -100 10])
title(['Frequency Response, Chan(',num2str(k-
1),')'],'fontsize',14)
xlabel('Frequency','fontsize',14)
ylabel('Amplitude','fontsize',14)
end
```

```
scl=max(h0);
hh0=reshape([h0/scl 0],6,11);
f_h0=fftshift(abs(fft(h0,128)));
w=kaiser(256,5')';

for k=0:1:256
reg=zeros(3,22);
v0=zeros(1,3)';
v1=zeros(1,6)';
v2=zeros(6,34);
%x0=exp(j*2*pi*(0:1200)*1/256)*exp(-j*2*pi*0.365);
x0=exp(j*2*pi*(0:1200)*k/256);

x1=zeros(1,200);

m=1;
flg=0;
for n=1:3:1200-3
    v0(1:3)=fliplr(x0(n:n+2)).';
    reg=[v0 reg(:,1:21)];
        for k=1:3
          v1(k)=reg(k,1:2:22)*hh0(k,:)';
          v1(k+3)=reg(k,2:2:22)*hh0(k+3,:)';
        end
        if flg==0
            flg=1;
        else
            flg=0;
            v1=[v1(4:6);v1(1:3)];
        end
        v2(:,m)=v1;
       x1(m)=sum(v2(:,m));
        m=m+1;
end

figure(6)
subplot(4,1,1)
plot((-0.5:1/512:0.5-
1/512)*36,fftshift(abs(fft(x0(1:512)/512))),'linewidth',2.5)
hold on
for zz=-3:3
plot((-0.5:1/128:0.5-1/128)*36+zz*6,f_h0,'--','linewidth',2)
end
hold off
grid on
axis([-18 18 0 1.1])
title('Channel Spectral Centers and Shifting Tone Frequency
Response','fontsize',14)
ylabel('Log Mag (dB)','fontsize',14)
xlabel('Frequency','fontsize',14)

for k=1:6
subplot(4,3,k+3)
```

```
plot(real(v2(k,:)),'linewidth',2.5)
hold on
plot(imag(v2(k,:)),'r','linewidth',2.5)
hold off
axis([0 100 -1.2 1.2])
grid on
title(['In-Band Tone Time Response, Path (',num2str(k-
1),')'],'fontsize',14)
ylabel('Amplitude','fontsize',14)
xlabel('Time Index','fontsize',14)
end

subplot(4,3,10)
plot(scl*real(x1),'linewidth',2.5)
hold on
plot(scl*imag(x1),'r','linewidth',2.5)
hold off
grid on
axis([0 100 -1.2 1.2])
title('In-Band Tone Response Sum of 6-Paths','fontsize',14)
ylabel('Amplitude','fontsize',14)
xlabel('Time Index','fontsize',14)

subplot(4,3,11)
ww=kaiser(128,0)';
ww=ww/sum(ww);
fx1=fftshift(20*log10(abs(fft(scl*x1(21:148+128)/256,256))))
;
plot((-0.5:1/256:0.5-1/256)*6,fx1,'linewidth',2.5)
grid on
axis([-3 3 -120 10])
title('Frequency Response, Sum of 6-Paths','fontsize',14)
ylabel('Log Mag (dB)','fontsize',14)
xlabel('Frequency (kHz)','fontsize',14)

subplot(4,3,12)
plot(0,0)
hold on
for k=1:6
    plot(fftshift((fft(v2(k,21:148+128)/256))),'-
o','linewidth',2.5);
end
hold off
grid on
axis('square')
axis([-1.5 1.5 -1.5 1.5])
title('Nyquist Plot, Each of 6-Paths','fontsize',14)
ylabel('Real','fontsize',14)
xlabel('Imaginary','fontsize',14)

pause(0.2)
end
```

```
scl=max(h0);
hh0=reshape([h0/scl 0],6,11);
f_h0=fftshift(abs(fft(h0,128)));
w=kaiser(256,5)';

for k=0:1:256
reg=zeros(3,22);
v0=zeros(1,3)';
v1=zeros(1,6)';
v2=zeros(6,86);
x0=exp(j*2*pi*(0:1200)*k/256);

m=1;
flg=0;
for n=1:3:1200-3
    v0(1:3)=fliplr(x0(n:n+2)).';
    reg=[v0 reg(:,1:21)];
        for k=1:3
          v1(k)=reg(k,1:2:22)*hh0(k,:)';
          v1(k+3)=reg(k,2:2:22)*hh0(k+3,:)';
        end
        if flg==0
            flg=1;
        else
            flg=0;
            v1=[v1(4:6);v1(1:3)];
        end
        v2(:,m)=ifft(v1);
        m=m+1;
end

figure(7)
subplot(4,1,1)
plot((-0.5:1/512:0.5-
1/512)*36,fftshift(abs(fft(x0(1:512)/512))),'linewidth',2.5)
hold on
for zz=-3:3
plot((-0.5:1/128:0.5-1/128)*36+zz*6,f_h0,'--','linewidth',2)
end
hold off
grid on
axis([-18 18 0 1.1])
title('Channel Spectral Centers and Shifting Tone Frequency
Response','fontsize',14)
ylabel('Log Mag (dB)','fontsize',14)
xlabel('Frequency','fontsize',14)

for k=1:6
subplot(3,3,k+3)
plot(real(v2(k,:)),'linewidth',2.5)
```

```
hold on
plot(imag(v2(k,:)),'r','linewidth',2.5)
hold off
axis([0 100 -1.2 1.2])
grid on
title(['In-Band Tone Time Response, Channel (',num2str(k-
1),')'],'fontsize',14)
ylabel('Amplitude','fontsize',14)
xlabel('Time Index','fontsize',14)
end

pause(0.2)
end





scl=max(h0);
hh0=reshape([h0/scl 0],6,11);
f_h0=fftshift(abs(fft(h0,128)));
w=kaiser(256,5')';
ww=kaiser(256,10)';
ww=ww/sum(ww)';
for k=0:1:256
reg=zeros(3,22);
v0=zeros(1,3)';
v1=zeros(1,6)';
v2=zeros(6,86);
x0=exp(j*2*pi*(0:1200)*k/256);

m=1;
flg=0;
for n=1:3:1200-3
    v0(1:3)=fliplr(x0(n:n+2)).';
    reg=[v0 reg(:,1:21)];
        for k=1:3
          v1(k)=reg(k,1:2:22)*hh0(k,:)';
          v1(k+3)=reg(k,2:2:22)*hh0(k+3,:)';
        end
        if flg==0
            flg=1;
        else
            flg=0;
            v1=[v1(4:6);v1(1:3)];
        end
        v2(:,m)=ifft(v1);
        m=m+1;
end

figure(8)
subplot(4,1,1)
```

```
plot((-0.5:1/512:0.5-
1/512)*36,fftshift(abs(fft(x0(1:512)/512))),'linewidth',2.5)
hold on
for zz=-3:3
plot((-0.5:1/128:0.5-1/128)*36+zz*6,f_h0,'--','linewidth',2)
end
hold off
grid on
axis([-18 18 0 1.1])
title('Channel Spectral Centers and Shifting Tone Frequency
Response','fontsize',14)
ylabel('Log Mag (dB)','fontsize',14)
xlabel('Frequency','fontsize',14)

for k=1:6
subplot(3,3,k+3)
plot((-0.5:1/256:0.5-
1/256)*12,fftshift(20*log10(abs(fft(v2(k,21:276).*ww)))),'li
newidth',2.5)
hold on
plot((-0.5:1/128:0.5-
1/128)*36,20*log10(f_h0),'r','linewidth',2.5)
hold off
axis([-6 6 -100 10])
grid on
title(['In-Band Tone Frequency Response, Channel
(',num2str(k-1),')'],'fontsize',14)
ylabel('Log Mag (dB)','fontsize',14)
xlabel('Frequency','fontsize',14)
end

pause(0.2)
end


%8
function [DH,DW] = myfrf(N, F, GF, W, A, diff_flag)
%---function [DH,DW] = remezfrf(N, F, GF, W, A, diff_flag)
%REMEZFRF Frequency Response Function for REMEZ.
%  REMEZ(N,F,A, ...) or
%  REMEZ(N,F,{'remezfrf',A}, ...) designs a linear-phase FIR
filter
%  using REMEZ.
%
%  The symmetry option SYM defaults to 'even' if unspecified
in the
%  call to REMEZ.
%
%  See also REMEZ.

%   Copyright 1988-2002 The MathWorks, Inc.
% $Revision: 1.6 $
%  modified by Karl Moerder and fred harris SDSU, copyright
2003

%  [DH,DW]=REMEZFRF(N,F,GF,W,A,diff_flag)
%      N: filter order (length minus one)
%      F: vector of band edges
%     GF: vector of interpolated grid frequencies
%      W: vector of weights, one per band
%      A: vector of amplitudes of desired frequency response
at band edges F
% diff_flag: ==1 for differentiator (1/f) weights, ==0
otherwise
%
%     DH: vector of desired filter response (mag & phase)
%     DW: vector of weights (positive)
%
% NOTE: DH(GF) and DW(GF) are specified as functions of
frequency

% Support query by REMEZ for the default symmetry option:
if nargin==2,
  % Return symmetry default:
  if strcmp(N,'defaults'),
    DH = 'even';   % can be 'even' or 'odd'
    return
  end
end

if nargin < 6
    diff_flag = 0;
else%+++
    error('Differentiator option is not allowed with
myfrf.')%+++
end

% Prevent discontinuities in desired function
for k=2:2:length(F)-2
    if F(k) == F(k+1)
        error('Adjacent bands not allowed.')
    end
end
if length(F) ~= length(A)
    error('Frequency and amplitude vectors must be the same
length.')
end

nbands = length(A)/2;
l = 1;
while (l+1)/2 <= nbands
    sel = find( GF>=F(l) & GF<=F(l+1) );
    % desired magnitude is line connecting A(l) to A(l+1)
    if F(l+1)~=F(l)   %
        slope=(A(l+1)-A(l))/(F(l+1)-F(l));
        DH(sel) = polyval([slope A(l)-slope*F(l)],GF(sel));
    else   % zero bandwidth band
        DH(sel) = (A(l)+A(l+1))/2;
    end
%---DW(sel) = W((l+1)/2) ./ (1 +(diff_flag & A(l+1) >=
.0001)*(GF(sel)/2 - 1));
    if A(l+1) > 0.0001;%+++ check for passband or stopband
        DW(sel) = W((l+1)/2);%+++ passband
    else;%+++
        DW(sel) = W((l+1)/2) * (GF(sel)/GF(sel(1)));%+++
stopband
    end;%+++
    l = l + 2;
end
```