Project report:
Initial filter Design:
Out of the requirements, the most stringent conditions are the passband ripple, 2nd stop band ripple and group delay variations. Given that we are struggling with these parameters it would be advantageous to commit to a topology that allows the greatest control over these factors while giving the cleanest phase response. I decided to side with an FIR filter to address group delay issues. Among the windowed filters, Blackman fits the bill but would be too big of an order resulting in an order of 223.
Instead we are going with an implementation of the Parks-Mcclellan Algorithm and everage alternation theory to create a filter where I gain control of the magnitude of the variation of both pass and stopband as well as providing minimal and ideally 0 group delay variation.

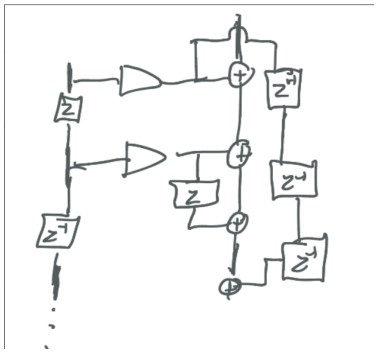
Forming the filter:
For weights we will use the first stopband as the normal weight: 1
The passband magnitude $1 - 10^{.05/20} \approx .005773 \approx .0057$(rounded up to provide margin)
Weight: $\frac{10^{-40/20}}{.0057} \approx 1.7544$
2nd stopband weight: $\frac{10^{-40/20}}{10^{-60/20}} = 10$
Normal freq: [75000, 10000, 125000, 175000]Khz $\div .5MHz$ = [.15, .2, .25, .35]
In order to estimate the order of the filter I naively assumed that there are M + 2 alternations and the order of the filter we derive will be odd or N = 2M+1 where N is the order. Lets naively assume that there is a local minimum located on the edge of the first stopband and a max at the edge of passband, the number of alterations that can occur in this state is 18 from repeating the pattern and subtracting 2 alternations lost in the 2nd transition band
Using this and the previous equations we get N = 37
This value was much too low and did not yield anything close, repeat the process but have 1.5 cycles happen in the first transition band which doubles our alternation hit rate with 7 points lost in transition so M = 40 -6 - 2 = 34
N = 69
Much closer to the desired input but still not close enough to iterate
Move up to 2.5 cycles: M = 60 - 10 - 4 = 46
N = 93
This surpasses our requirements by a fair margin. We can start iterating backwards to the proper order.
Iterating, we arrive at an order of 85. This filter also gives us some margin for design. With a stopbands of -41.2817dB and -61.26151dB and a passband variation of .04303937dB

Further constraints:
SNR
Our SNR is heavily influenced by the order of our filter. No matter how much the hardware improves, there are going to be N+1 elements that are going to be multiplied at some point and added together. We can calculate our baseline Bit rate using our SNR requirements. For this part we are going to assume our max passband variation is .05dB since we don't know the quantization effect on the filter yet.
From Project spec we expect a -6dB or a half amplitude signal in. assuming the worst case in passband we have a -6.05dB signal at the output.
Our SNR needs to be -72dB which is equal to 10log(Signal Power/noise Power)
We use the ideal model of quantization to form our noise where noise power is $\frac{\Delta^2}{12}$ per multiplication. We multiply it with the 86 multiplications needed to perform our analysis.
Signal power is the square of our magnitude
This works out to
20log(Signal Magnitude/sqrt(noise power))
20log(signalMagnitude) = -6.05dB
-78.05 =-20log($\frac{1}{2^k}\sqrt{\frac{86}{12}}$) where k is the bits
k = 14.38
Round up to 15 bits. This is our minimum clearance for bits for all elements.
Just in case I also calculated for a filter of order 86 which also works out to 15bits.
Filter quantizing:
We have a floor of 15 bits
The most sensitive margin out of our filter is the -61.26151dB stopband so let’s design for that one.
The error from quantizing the filter presents itself as 20log($\frac{1}{2^k}\sqrt{\frac{86}{12}}$) when expressed in dB
Currently we design a filter that can perform at -60dB
-61.26151dB + QdB = -60 dB where Q is the error from quantization
Q = -77.3816dB
-77.3816dB = 20log($\frac{1}{2^k}\sqrt{\frac{86}{12}}$)
k = 14.25bits
Round up to 15 bits.
Plugging this value into the filter, I was unfortunately just shy of the -60 dB threshold so I had two choices, to raise the order or commit to hardware changes.
We are improving and innovating assuming our current filter is 16bits at order of 85 which does meet requirements but is non ideal for our uses.

Power comparison and hardware optimization/innovations:

Our filter is symmetric which means that a small number of multipliers can be replaced with delay blocks instead (for example the 43rd multiply and add will operate as normal but the 44th will instead input a delayed value of the 43rd result into the adder.)

In this way we could make small hardware improvements like the diagram attached



This will only save a couple hundred elements though

At 15 bits the multipliers cost 2*15^2 which can be replaced with 5 multipliers with 5, 4, 3, 2, 1 delay for a total save of:

5*2*15*15 - 15*5*(5+4+3+2+1) = 1125 logic gates

At 16bits we can perform similarly with a save of:

5*2*16*16 - 16*5*(5+4+3+2+1) = 1520

For functions with odd symmetry we save need one more delay block thus we can only replace 4 multiply blocks

4*2*15*15 - 15*5*(5+4+3+2) = 750 logic gates

For 16 bit: 5*2*16*16 - 16*5*(5+4+3+2) = 928 logic gates

These hardware improvements are not enough to get us to the difference in power required

The total power of the 85th order filter is calculated by the total multipliers(86), adders(85) and registers(85) minus the logic saving above.

But, we can attempt to cascade the filter. I converted the filter into zeros and poles using zplane on the dfilt.dffir(filter) object, took the angle of each of the zeros to find which value they correspond to and binned them accordingly. 61 zeros have an normalized phase of above .35. The rest of the filter is defined by 24 zeros. So the cascaded filter will be a 61st order and 24th order filter respectively with the 61st order operating with 16 bits and the 24th order filter operating at 15 bits. The 16 bits is just a bit up from the previous iteration since we were so close to ideal stopband behavior.

Calculating out the power output with the assumption that both resulting filters are symmetric(best case)

61*(4*16 + 2*16^2 +5*16) + 2*16^2 -1520 + 24*(4*15 + 2*15^2 +5*15) + 2*15^2
= 52748

Expect cost from just raising the order:

86*(4*15 + 2*15^2 +5*15) + 2*15^2 - 750 = 50010

It is clear that raising the filter order is superior to cascading.

We raise the filter order to 86, plugging 87 into our signal quantization earlier also yields 15 bits again so we have found our implementation.

Our filter will be implemented at an order of 86

Implementation costs 86 Adders, 83 multipliers and 98 registers (after mini optimization above)

Since we implemented an FIR filter we also do not face a risk of overflow.

Matlab implementations:

I first formed my filter using the Parks-Mclellan algorithm from the function firpm

Each element was rounded by first multiplying by 2^bits or 2^15 in our case and rounding, effectively only saving the 15 bits that our implementation accounts for, and dividing by 2^15

So we have only the number that the bits account for

Same was done for the input signal that was formulated off of a sampled sin wave with 100000 samples. Each sample was again rounded with the same method.

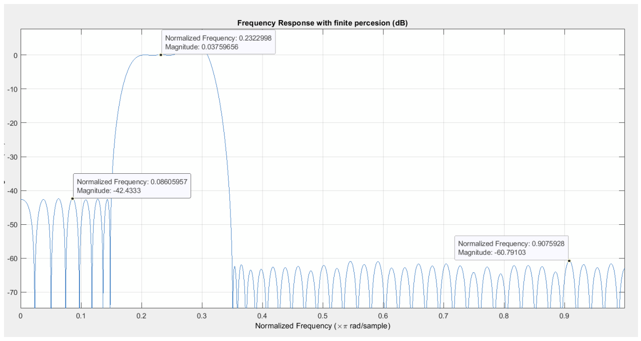
In order to convolve these two I nested two for loops that execute a convolution, I pursued this method because in our filter implementation there is a roundoff happening every multiplication, so a for loop convolution is necessary so I can access every multiplication result to apply rounding.

To test for SNR I swept 10 frequencies from 100000 to 125000 in increments of 2500

Minimum SNR amongst these was 72.4521dB.

The rest of the code generates the graphs or is scrap code to calculate the other factors.

All graphs will be generated by the matlab function and graphs the frequency resp, pole zero plot and impulse both finite and nonfinite filter versions when the code is run. It also generates group delay and worse case SNR for the finite filter.



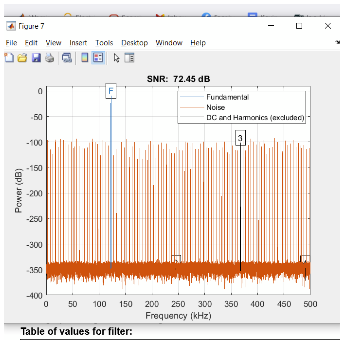


Figure 7
File Edit View Insert Tools Desktop Window Help
SNR: 72.45 dB
Fundamental
Noise
DC and Harmonics (excluded)
Power (dB)
Frequency (kHz)

Table of values for filter:

Table of values for filter from graphical analysis:

Minimum stop band attenuation:	-42.43245dB(lower) -60.78489dB(upper)
Passband ripple	±.036dB
Group Delay variation	0 samples
Min output SNR	72.4521dB
bits(input and filter)	15bits
Power	50010*P