David Castro

Fall 2022

212A Project

Initial Design Based off Requirements:

Parameter	Value
Sampling Rate (F_s)	1MHz
Pass-Band Edges (f_{p1} , f_{p2})	75KHz, 175KHz
Stop-Band Edges (f_{s1} , f_{s2})	95KHz, 125KHz
Minimum Stop-Band Attenuation	40dB or (0.01 V/V in non-dB)
Maximum pass-band ripple	0.05dB (0.0058 V/V in non-dB)
Pass-band group delay variation	< +/- 5 samples
Minimum output SNR (for -6dB full-scale sinusoidal input)	72 dB

$$\text{Digital frequency: } \omega = \frac{2 * \pi * f}{f_s}$$

Digital Parameter	Digital Value
w_{p1}	0.15π (satisfied in filter)
w_{p2}	0.35π (satisfied in filter)
w_{s1}	0.19π (satisfied in filter)
w_{s2}	0.25π (0.30π chosen for filter instead but still satisfies the condition)

FIR Minimax Optimal Filter Design Reasoning:

A FIR Minimax Optimal Filter was chosen. An IIR or FIR filter can be used. I chose the FIR filter because a linear phase response was desired. While this can be done with an IIR filter, I chose FIR for the simplicity. While FIR filters can be designed many ways, I had the ability to do either windowing or Minimax Optimal. Based off the requirements of the filter, a Hamming window would be needed in order to satisfy the requirements. The length of the Hamming window would be:

$$\frac{6.22\pi}{N} = width\ of\ Transition\ Band = 0.04\pi \rightarrow N = 166$$

While Kaiser window would reduce the order of the windowing method, the Hamming can be seen as approximation for the order that would be needed. The Minimax Optimal filter on the other hand only needed an order of 120. Thus, the Minimax Optimal Filter was chosen.

While designing the Minimax Optimum filter, w_{s2} was made a little more stringent in order to fix the non-monotone shape seen. When the transition band is large for optimum filters it can result in some non-monotone behavior in the transition band. In order to fix this, w_{s2} was set to 0.30π instead of 0.25π. This still satisfies the condition of the filter.

Since the coefficients are quantized, this means there needed to be some space left for ripple error. Quantization will cause for the ripple to grow. If the filter barely meets the specifications, a high number of bits would be needed for the coefficients. Instead, it is much easier to meet the conditions when the filter is a bit more stringent - ripple wise. An order of 120 for the Minimax Optimum filter was chosen, and it produced a frequency response that was already more stringent, meaning the order did not need to be increased. (Quantization section will go into more detail about the bits required for the coefficients in order to meet specifications).

The Layout of the Filter:

The layout of the filter is based off scaling used for keeping the input bounded. Because of how L-inf normalization works, aggressive scaling can be seen for certain inputs when the filter being scaled is seen as a high-pass or low-pass filter. For example, while the whole filter may be a band-stop, some of the stages combined can make a high-pass filter. In order to fix this problem, I tried to group the zeros in order to make the scaled stages all-pass like. I first created biquads based off complex conjugates. Then group those biquads with opposing zeros on the opposite side of the unit circle. From here I had filter stages consisting of 4 zeros. I grouped these one more time into 8 zero stages that would have the zeros somewhat spread out in order to get a somewhat all-pass filter. Since my filter is of 120 order, and I grouped 8 zeros together, this means I have 15 stages.

The gain factor 'k' given when separating the zeros through the function, "tf2zpk()", was applied equally to each stage. I believe this gain factor can be applied in an optimized matter in order to fix some of the large coefficients that cause the necessity for a longer word length as will be seen in the next section.

The ordering of the stages follows from least magnitude to largest magnitude in order to not degrade SNR. This is explained in the "scaling and round-off error" section below.

Quantization Reasoning:

- Coefficient Quantization:

The first quantization to be discussed will be FIR coefficient quantization. Based off the amount of word length given to the coefficients, the ripple of the filter in the bands may exceed the specifications. We are able to estimate how many bits will be needed for this filter:

$$B > log_2 \frac{\sqrt{(M+1)/12}}{d_{min}} \quad , \qquad M = order$$

Our current filter has a d_{min} of 1.0024 where we are allowed 1.0029 in the passband

$$B > log_2 \frac{\sqrt{(120+1)/12}}{1.0029 - 1.0024}$$

$$B > 8.7563$$

The bits (B) requirement is based off the quantization step – fraction bits. This means at least 9 bits must be given to the fractional part of the word length in order to meet the requirements.

The above calculation requires that each of the coefficients are independent of each other and is less conservative. The next calculation is a much more conservative bound:

$$B > log_2 \frac{M+1}{2d_{min}} \quad , \qquad M = order$$

$$B > log_2 \frac{120+1}{2(1.0029 - 1.0024)} = 11.7035$$

The conservative bound shows at least 12 bits must be given the fractional word length. When the coefficients are quantized in MATLAB, it is seen that at least 14 bits for the fractional word length is required because a small portion of the filter exceeds the bounds. 12 bits nearly satisfies the requirement as the ripple is pretty consistent. 13 bits is also a slightly outside the bound.

Most of the coefficients are within [-1,1] ,but a few do go over the amount. The highest coefficient is around 6.6351, meaning there would need to be at least 3 bits in order to represent this number. Since there is also a sign bit, that means an extra bit is needed. This makes 4 bits. With there being 4 bits going towards the integer side, and 14 bits for the fractional side. **The total word length will need to be 18 bits long.**

- Scaling and Round-Off Error:

We had the goal of scaling the intermediate nodes in each stage to be within a bound of [-1, 1]. Scaling can cause for significant degradation of the SNR, since the round-off error, q[n], sees the gain of the scaling. The ordering of the stages is important here. Since the scaling method L-∞ utilizes the infinity norm, this requires for the stages to be ordered from smallest magnitude to largest magnitude. This will help improve the SNR. My self-calculated SNR showed it required 23 fraction bits to achieve the desired SNR of 72 dB, assuming the round off noise is white, independent, and uniformly distributed. When tested in MATLAB and checking with the 'snr()' function it took 23 fraction length bits with 1 sign bit in order to meet the 72 dB requirement. While there is some natural error of the calculations, since noise/dither is added to the input signal in order to remove harmonics, the experimental and theoretical were very close. For some input signals, 2 decimal bits (1 of them sign) were needed in order to have the filter working properly. This makes sense because our coefficients are not bounded [-1,1] therefore after some multiplication or addition, some of the nodes within the stage may exceed our range of [-1,1]. Because of this, 1 extra bit will be used for the integer portion of the word length.

$$Power\ of\ Noise\ at\ Output = \frac{1}{2\pi}\int_{-\pi}^{\pi} \sigma_{q_n}^2 * |G_n(w)|^2$$

$$Power\ of\ Signal = \frac{A^2}{2} \rightarrow Given: -6\text{dB fullscale} \rightarrow \text{A} = 0.5012$$

$\sigma_{q_n}^2 = N_{Taps} * \frac{2^{-b}}{12}$, $G_n(w) = filter\ seen\ by\ noise$, Taps = multiplication= Roundoff-Error

Bits (Word-Length)	SNR (dB) Experimental	SNR(dB) Theoretical
2 Sign + 17 Fraction Bits	40.121 dB	38.645 dB
2 Sign + 19 Fraction Bits	54.9253 dB	50.65 dB
2 Sign + 21 Fraction Bits	64.4691 dB	62.73 dB
2 Sign + 22 Fraction Bits	67.6164 dB	68.75 dB
2 Sign + 23 Fraction Bits	76.5414 dB	74.77 dB

Word-Length ended up being chosen = 2 Integer (1 sign) + 23 fraction bits = 25 bits total

Hardware Requirements:

15 stages of 9 Coefficients (scaling after each stage is applied to each stage). This means there are 9 multipliers per stage. After each multiplier there is one adder except for one coefficient, therefore there are 8 adders per stage. There are 8 delays per stage, meaning 8 registers per stage.

Currently, all rounding is limited to the Word-Length of 25 bits (2 Integer (1 sign) + 23 fraction bits), excluding coefficient bits.

Hardware Estimation off N-bits	Calculations	Logic Gates
n-bit Adder - 4n	(15stages)(4*25bits)(8adders)	12,000
n-bit Multiplier - $2n^2$	(15stages)(2*25^2)(9multi.)	168,750
n-bit Register - 5n	(15stages)(5*25)(8registers)	15,000
	Total Logic Gates =	195,750 Logic Gates

How to Improve:

I was unable to implement these changes, but I believe these things may be done in order to improve the filter:

SNR causes for a requirement of a certain word-length. The way scaling applies for the stages still degrades the SNR heavily I believe. I think two ways to fix this is by combining the zeros in an even more optimized way and applying the ‘k’ gain factor from “tf2zpk()” heavily towards stages that have a large amount of gain. If more of the stages are combined together, it may help with SNR degradation. The more zeros combined, the closer you can get towards an all pass like filter at that stage. Right now, the large word-length comes from SNR which relates to the scaling being quite aggressive. The problem with combining more stages though is that coefficient quantization can have a greater affect. Slightly increasing the order of the filter can help solve this problem. These are some changes that can implemented and tested quite easily.

A more difficult or large change that can help with the amount of logic gates being used is interpolation of the filter. Since the filter is a band-stop, removing the extra images from interpolation is not straightforward. One way I think of being able to implement interpolation is by having a low-pass and high-pass, applying the interpolation process, and then combining them. The way they would be implemented is by just having an input feed into the high-pass and low-pass then summing their result (parallel implementation). I tested around with this and found that a high order would be needed in order to satisfy the ripple requirement in the passband. While I was not able to fully implement it, in order to compare, if this does meet the requirements, it is a possibility that a significant amount of logic gates may be reduced.