Today's Lecture

- Decimation filters for ΣΔ ADCs
  - Digital decimation filters
    - Aliasing in the analog domain
    - Aliasing in the digital domain
    - Coefficient precision and gain scaling
  - Digital arithmetic throughput calculations
    - One-stage decimation
    - Linear phase implications

ΣΔ Analog-to-Digital Converters

- A ΣΔ Analog-to-Digital Converter (ΣΔ ADC) combines
  - An analog ΣΔ modulator which produces an oversampled output stream of 1-bit digital samples
  - A digital decimation filter which takes the 1-bit modulator output as its input and
    - Filters out out-of-band quantization noise
    - Filters out unwanted out-of-band signals present in the modulator's analog input
    - Lowers the sampling frequency to a value closer to 2X the highest frequency of interest

ΣΔ ADCs

- Commercial DSPs aren't designed to handle 1-bit input samples at oversampled data rates
  - A 400Mip DSP only executes 133 instructions per 3MHz sample
  - In 2001, the 32X32b multiply-accumulate cost is 5¢/Mip, independent of the number of active bits/word (ref. 1)
- DSPs are designed to handle 16+ bit wide data words at Nyquist-like sampling frequencies
- ΣΔ decimation filters bridge the speed/resolution gap

Aliasing in the Analog Domain

- We'll continue using the 3MHz, 1-bit ΣΔ modulator and its audio application as the basis for decimation filter analysis
- Sampling action at the modulator input inherently results in aliasing
  - 2980 and 3020kHz alias to 20kHz
  - 2999 and 3001kHz alias to 1kHz
- No digital filter can separate frequency components that have aliased on top of one another

Aliasing in the Analog Domain

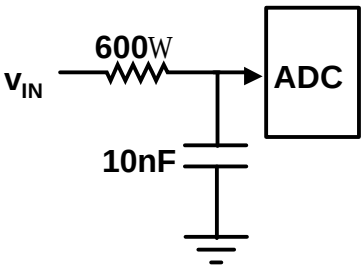
- 1st-order RC filters usually provide adequate antialiasing protection for ΣΔ ADCs
  - A 30kHz LPF provides only 40dB of attenuation at 3MHz,
  - But microphones and other audio transducers produce negligible outputs at 3MHz
- "Transducer loss" is an important factor in all real-world antialiasing filter specifications
  - Talk to veterans about the level of transducer loss you can count on in your application
  - Or measure it

Aliasing in the Analog Domain

- Protecting high-order modulators from instability-provoking square wave inputs provides additional justification for an RC antialiasing filter
- Remember that any RC antialiasing filter adds kT/C noise
  - Almost all of this noise is in the band of interest
  - Let's review a 600Ω, 10nF LPF …

kT/C Noise

- kT/C noise of a 10nF capacitor is 0.64μVrms
- ADC noise from 0-20kHz is 6.68μVrms
- Sum of squares addition yields 6.71μVrms



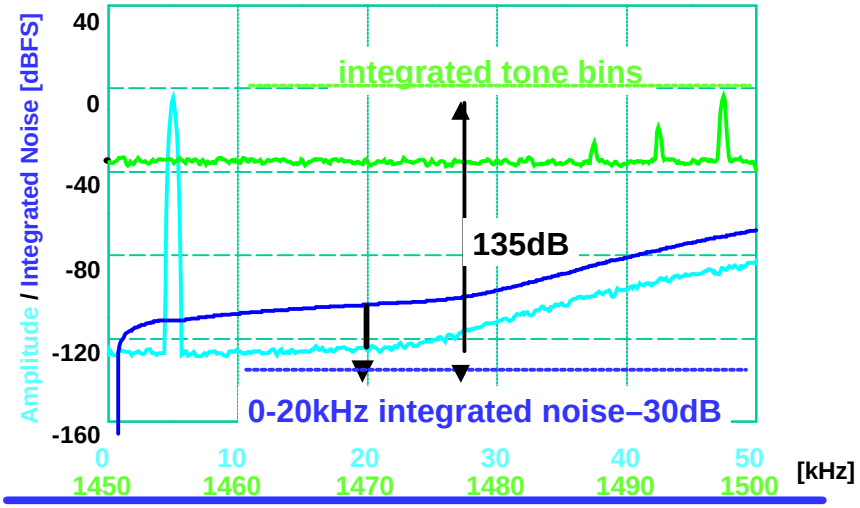
Aliasing in the Digital Domain

- The digital filters we'll develop for audio applications will lower the sampling frequency from 3MHz to 46.875kHz
  - That's called "decimating by 64" or "64X decimation"
- Aliasing can occur in the digital domain whenever sampling frequencies decrease
  - Digital filters which precede the decimation step attenuate signals and noise which would otherwise alias into the 0-20kHz band

Aliasing in the Digital Domain

- Stopband attenuation specifications for ΣΔ decimation filters are based on the need to attenuate ΣΔ tones near $f_S/2$ down to levels 30dB below the 0-20kHz integrated noise
- Let's plot on the same dBFS scale:
  - A full scale 1Vrms, 5kHz input with modulator thermal noise added (plotted from 0-50kHz)
  - Tones for a 5mV dc input (plotted from 1450-1500kHz)

Stopband Attenuation Analysis



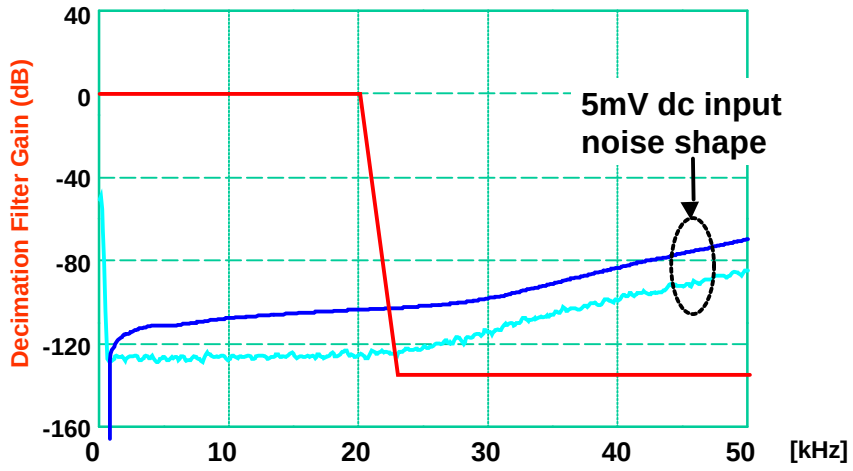
Stopband Attenuation Analysis

- 135dB of stopband attenuation is required for aliased tone suppression
  - ~24b FIR filter coefficients are a likely choice given the 6dB/bit estimate (see lectures on digital filters)
- 135dB of stopband attenuation results in negligible aliased non-tonal quantization noise
- Where should the stopband begin?
  - Given our decimation filter output word rate of 46.875kHz, 23kHz seems a safe choice

Decimation Filter Synthesis

- We'll call our ideal decimation filter "Filter #1"
  - 0.00±0.01dB gain from 0-20kHz
  - 135dB stopband attenuation from 23-2977kHz
  - Linear phase
- The target magnitude response appears on the following slide…
  - Don't try this with an analog filter!

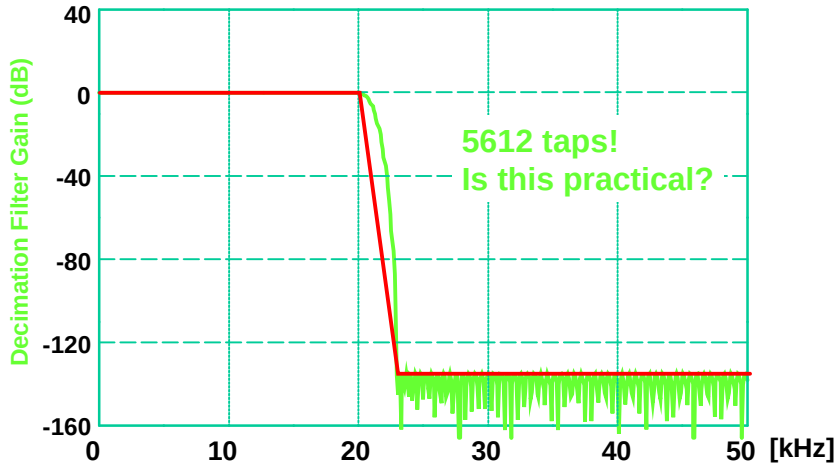
Filter #1 Target Response



Decimation Filter Synthesis

- We'll use MATLAB's implementation of the Parks-McClellan algorithm to synthesize this filter (remez)
- After crunching for a while, MATLAB returns a 5612 tap FIR filter with the following response…

Filter #1 Actual Response



Filter #1

- A classical 5612-tap, f_S=3MHz FIR filter would require a 5612*3MHz=16.8GHz multiply-accumulate rate
- However, in a decimation filter application, we never waste power to compute filter output samples that we immediately decimate away
- The required multiply-accumulate rate is reduced by the decimation ratio to 263MHz

Filter #1

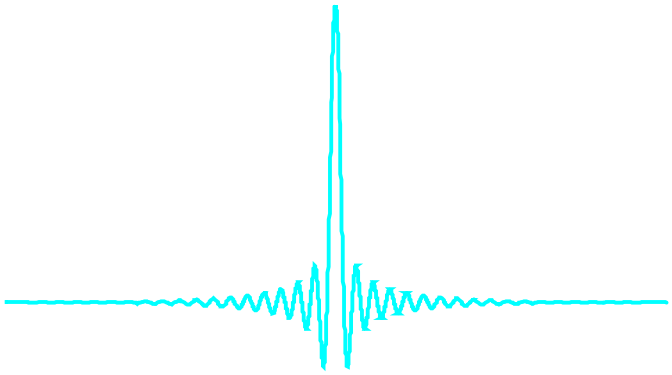
- The second key factor that makes this FIR filter unusual is that it needs no hardware multiplier at all
  - Input data is only 1-bit wide
  - The “multiplier” merely adds or subtracts coefficients from the accumulator
- 263MHz begins to seem reasonable, but we can use another simple trick to reduce power further…

Coefficient Symmetry

- Linear phase filter coefficients are symmetric around the middle of the impulse response
- We’d never waste ROM to store all 5612 coefficients when only 2806 are unique
- A 5612x1b data memory allows us to exploit coefficient symmetry to reduce “multiply”-accumulate rates by another 2X …

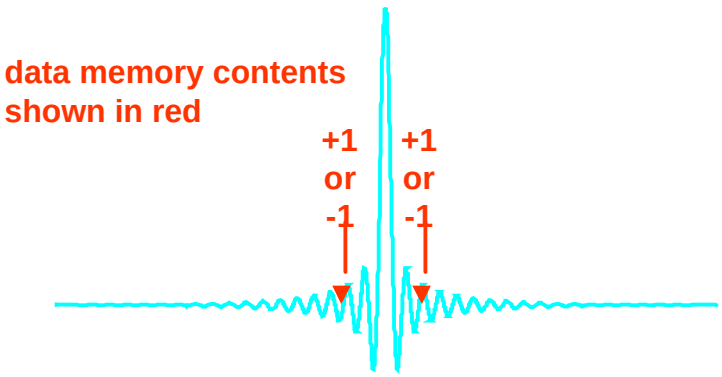
Coefficient Symmetry

- Our 5612 coefficients look like this:



Coefficient Symmetry

- Each time we fetch a coefficient from ROM, we fetch both 1-bit samples that need it from the 5612x1b data memory:



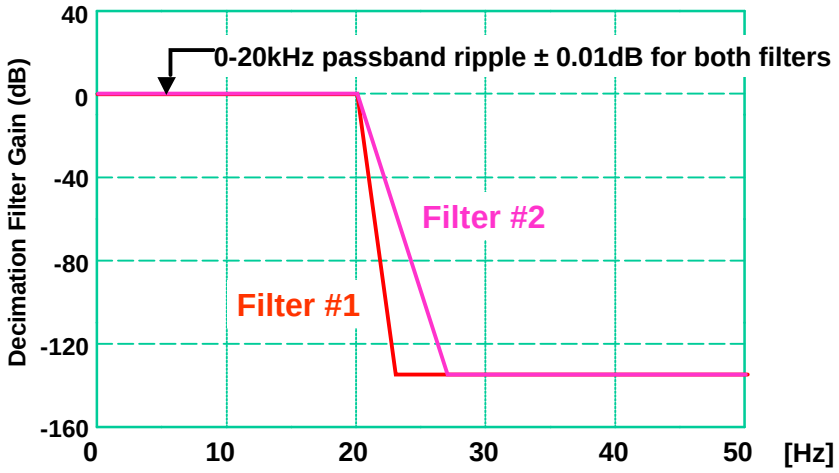
Coefficient Symmetry

- We only have two data states, +1 and –1
- If we add the data before “multiplying”, only 3 results are possible:
  - +2 if both 1b samples are +1
  - -2 if both 1b samples are –1
  - 0 if 1b samples are –1,+1 or +1,-1
  - Our “multiplier” adds, subtracts, or does nothing
- “Multiply-accumulate” in this application requires only an accumulator operating at 132MHz!

Filter #2

- While the throughput requirements of Filter #1 are not outrageous, audio applications economize further
- Modulator input signals that alias into frequencies above 20kHz are inaudible
  - Most people can’t hear 20kHz full-scale sinewaves
  - Who would ever record that anyway?
- So, unless you’re interested in marketing your audio ADC to dogs (dogs can hear up to 30kHz, supposedly), consider Filter #2 …

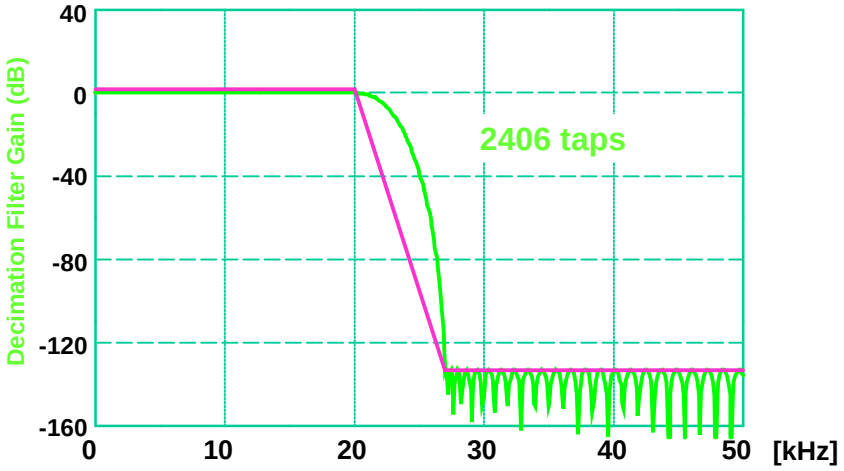
Filter #2 Target Response



Filter #2

- With this filter specification, an input signal at 24kHz will alias to 46.875-24kHz = 22.875kHz without anywhere near 135dB attenuation
  - Neither 24kHz nor 22.875kHz is audible
- Exploiting the audibility of aliased components allows us to widen the transition band…
  - … The most critical factor in determining filter order
  - Let's see what MATLAB cooks up

Filter #2 Actual Response



Filter #2

- Using the same coefficient symmetry trick that helped Filter #1, Filter #2's accumulate rate drops to 2406/5612*132MHz = 57MHz
- Performance compromises are inaudible
- Most companies refuse to pay extra for "aliasing purity", if the extra costs of purity bring no perceptible benefits
  - That's just good engineering

FIR Arithmetic Throughput

- Length-N FIR decimation filters which take input samples at a sampling frequency f_{SIN} and produce output samples at a sampling frequency f_{SOUT}, $f_{SOUT}<f_{SIN}$, require multiply-accumulate rates of

$$f_{MA} = Nf_{SOUT}$$

- Linear phase FIRs which exploit data addition before multiplication reduce this to

$$f_{MA} = \frac{Nf_{SOUT}}{2}$$

FIR Arithmetic Throughput

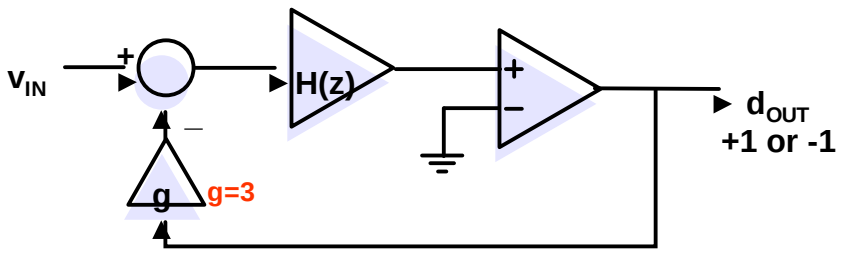
- FIR filters with 1-bit input data don't need traditional hardware multipliers
  - Use add/subtract/do nothing accumulators
- How wide should these accumulators be?
  - What coefficient precision is needed?
  - What output resolution should we use?
  - Let's look at a Filter #2 implementation…

FIR Implementation

- Digital filters usually come with bit-width' that are multiples of 4
- 16-bits results in unwanted digital quantization noise
- So let's try a 20-bit filter for our 16-bit ADC
  - 2^{20}=1048576
  - Each LSB is 1ppm of the ADC input range
- Let's look at the mapping of our 1Vrms full scale sinewave into digital output values
  - Before we set filter gain levels, we need to review modulator outputs

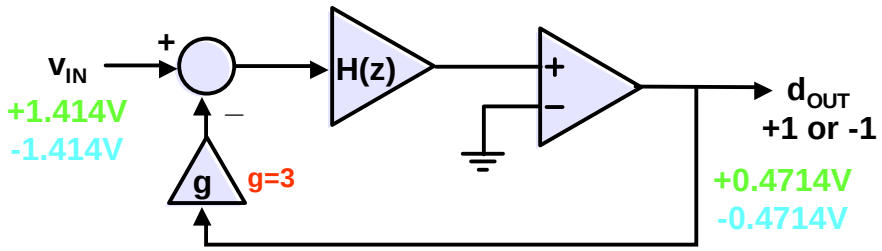
Modulator Outputs

$$\frac{D_{OUT}(z)}{V_{IN}(z)} = \frac{H}{1+gH} \approx \frac{1}{g} = \frac{1}{3}$$



Modulator Outputs

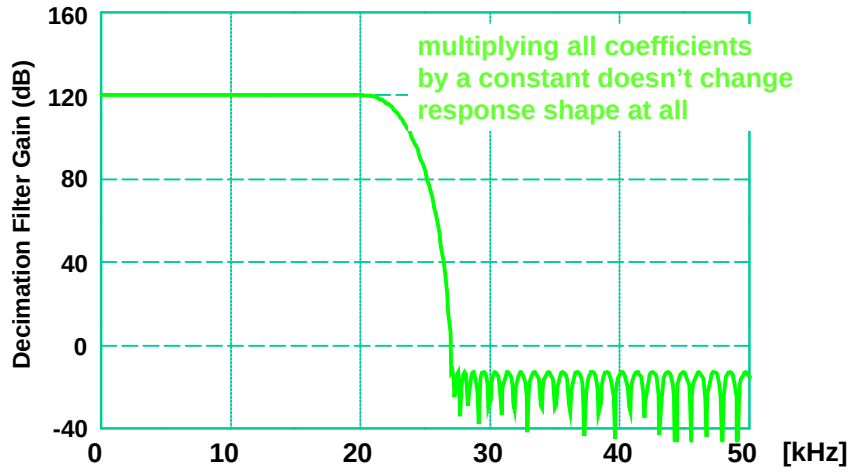
Positive and negative peaks of a 1Vrms full-scale sinewave correspond to levels shown below:



Decimation Filter Gain

- "Gain scaling" in the decimation filter maps the ±0.4714 modulator average output at signal peaks to the 20-bit digital full-scale range of $\pm 2^{19}$
  - Ideal decimation filter dc gain is 1112000=120.9dB
  - To allow for offsets, etc., we'll use a slightly smaller gain of 2^{20}=120.4dB
- An FIR filter's dc gain equals the sum of its coefficients
  - Let's adjust Filter #2's coefficients accordingly …

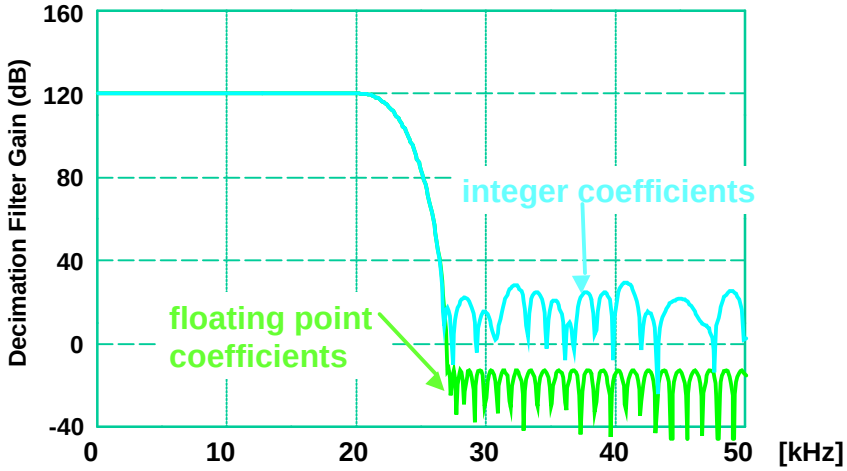
Filter #2 Response



Filter #2 Response

- The gain adjustment is correct, but coefficients are still floating point
- Rounding these coefficients to the nearest integer using MATLAB's round() function yields the following response …

Filter #2 Responses

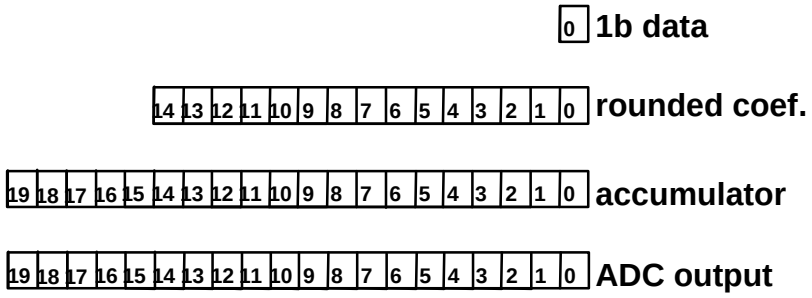


Filter #2 Responses

- The stopband attenuation is horrible, much less than the 135dB requirement, and the problem is obviously coefficient precision
- Check the integer coefficients
  - The biggest one is +15715
  - The smallest one is –3332
  - That's only 14-15b of coefficient precision, and <90dB of worst-case stopband attenuation
- When 2406 coefficients sum to 2^{20}, the biggest coefficient is pretty small

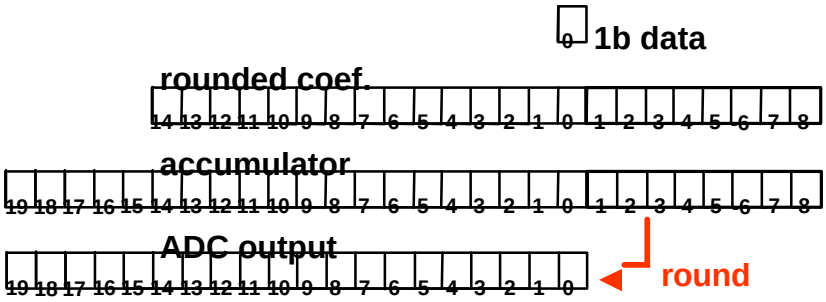
Filter #2 Bit Map

Let's look at the digital scaling in our defective filter :



Filter #2 Bit Map

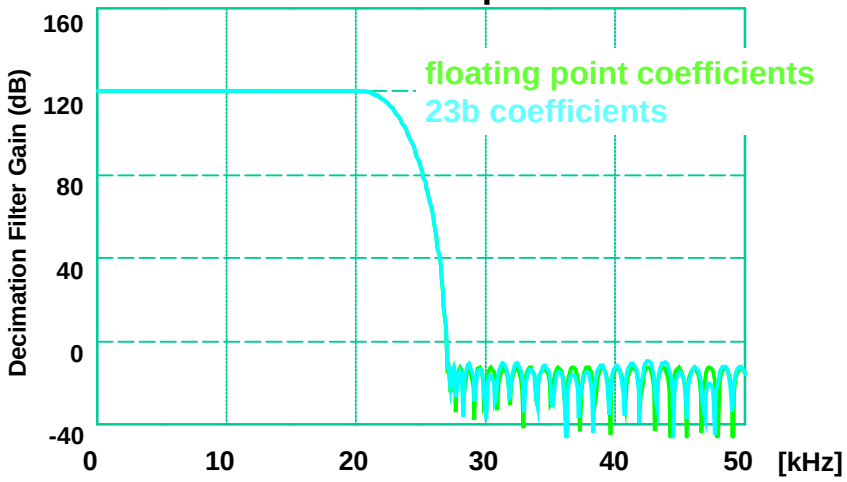
To add coefficient resolution, we'll add 8 coefficient bits below the 2^0 point:



Filter #2 Bit Map

- Higher-precision coefficients are produced with a simple coef=round(256*coef)/256 operation
- The 23b fixed point coefficient magnitude response appears on the following slide …
- Rounding of the 28b accumulator to produce the 20b ADC result adds 20b quantization noise
  - At -122dBFS, that's insignificant for a 103dB dynamic range ADC

Filter #2 Responses

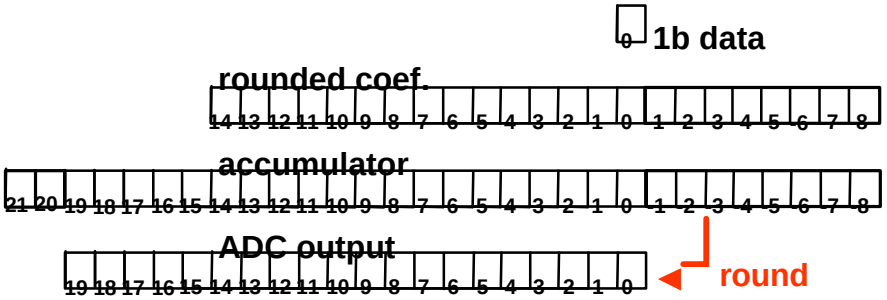


Intermediate Overload

- With our properly scaled coefficients,
  - The sum of coefficients is 1.047e6
  - The sum of coefficient absolute values is 2.040e6
- The accumulator can never reach a value outside the (-2.041e6,+2.041e6) range
  - Two accumulator bits above the ADC output MSB provide intermediate result overload protection …
  - A 30b accumulator for a 20b ADC isn't unusual

Filter #2 Bit Map

The green accumulator bits (20 and 21) provide complete overload protection:



Intermediate Overload

- Given the relatively low cost of this overload protection, it hardly pays to evaluate whether or not accumulator bit 21 can be reached by real-world ADC input signals
- Our first pass decimation filter design is complete
  - We'll add this filter to our stage 2 modulator model next time

References

1. Texas Instruments, C2000 Series DSP datasheets, 2001.
2. R. E. Crochiere and L. R. Rabiner, "Interpolation and Decimation of Digital Signals – A Tutorial Review", Proc. IEEE, 69, pp. 300-331, March 1981.
3. Nav Sooch, "Gain Scaling of Oversampled Analog-to-Digital Converters", U.S. Patent 4851841, 1989.