

# **Assignment: Delta Sigma Simulation**

Arron Zheng, David Armstrong

---

## **1. Abstract**

The Delta-Sigma converter was invented in the 1960s by professor Yasuhiko Yasuda at the University of Tokyo. The name Delta-Sigma comes from the fact that this technique sums (sigma) the differences/change (delta) in the signal. The Delta-Sigma ADC offers many benefits over a traditional ADC including producing higher resolution and better signal-to-noise ratio (SNR). In this lab, we simulate a 2nd order Delta-Sigma modulator and compare our results to Hauser's paper "Principles of Oversampling A/D Conversion".

## **2. Introduction**

The goal of this lab is to create a simulation of the 2nd order Delta Sigma modulator from Hauser's paper. To understand the methods and conclusions presented here, it's necessary to preface with an explanation of the Delta-sigma modulation.

### **Delta-sigma modulation**

Delta-sigma modulation is a method of encoding analog signals to digital signals in an ADC (analog-to-digital converter). This modulation differs from a traditional ADC we previously explored in this class. The first step entails recording the signal's change (referred to as delta) as opposed to recording the value of the signal (as in a traditional ADC). Then the digital output is passed through a 1-bit DAC (digital-to-analog converter) and added to the input signal before modulation, which reduces the error caused by the modulation.

Using a delta-sigma modulator has multiple benefits, namely higher resolution due to oversampling, cost-effectiveness, and higher SNR or dynamic range compared to a traditional ADC. Traditional ADCs use analog lowpass filters which are often expected to have "brick wall" like characteristics, making their design more complex and expensive. Delta-sigma ADCs use sample rates that are large, often 128 or 256 times higher than necessary to satisfy the Nyquist criterion. This provides not only higher resolution, but also better anti-aliasing performance. This oversampling allows for a more gradual "roll-off" for the filter in the frequency domain (as shown in figure 1) which is far less expensive and simpler to build.

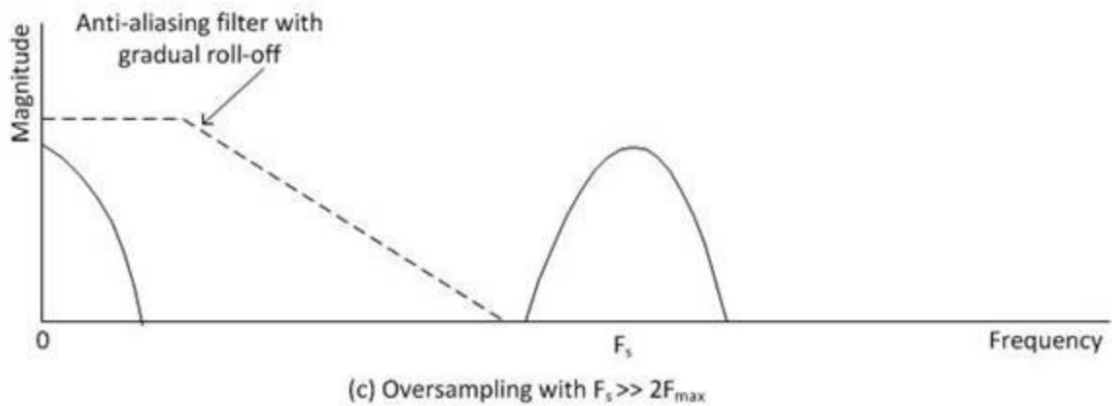


Figure 1: Gradual Roll-Off due to higher oversampling

Secondly, the delta-sigma modulator has a unique behavior referred to as quantization noise shaping. It behaves like a lowpass filter for signals and highpass filter for quantization noise, meaning that it pushes noise to higher frequencies. We are able to take advantage of this phenomenon by using decimation to filter out higher frequencies (where the noise is high) improving SNR.

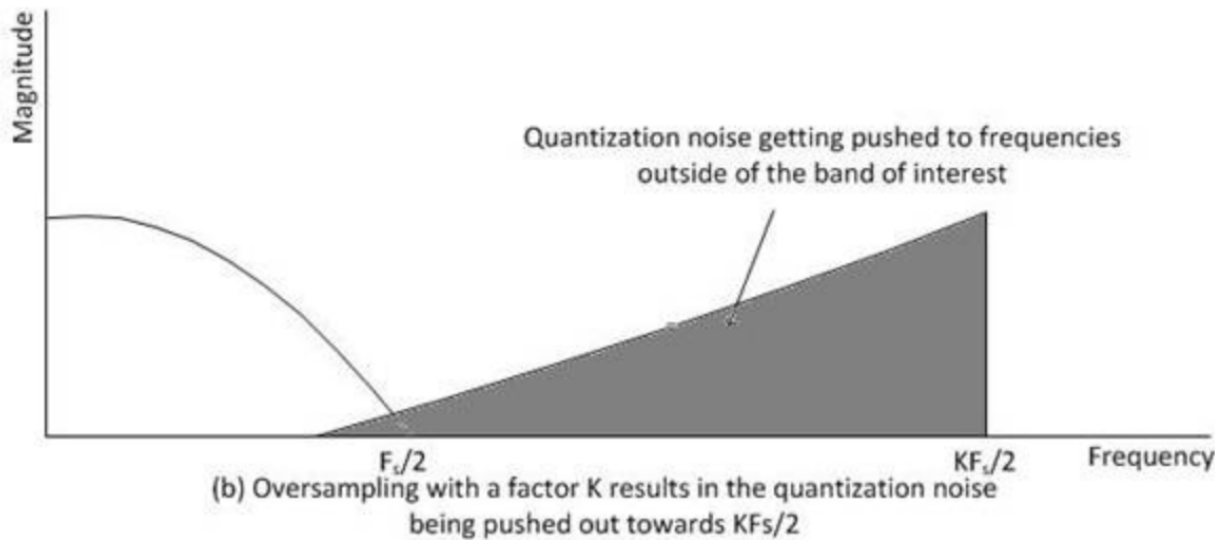


Figure 2: Quantization noise shaping pushing noise to higher frequencies

## Hauser's Paper 2nd order Delta Sigma modulator

Hauser's Paper gives a comprehensive summary of the Oversampling analog-to-digital converters (OSADCs) touching on the 2nd order Delta Sigma modulator shown in figure 3.

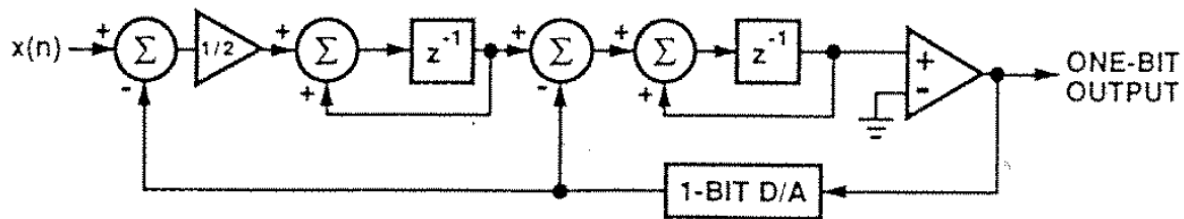


Figure 3: 2nd order Delta Sigma modulator from Hauser's Paper

He concludes many characteristics of this 2nd order Delta Sigma modulator such as the following:

$$SNR_{max}(db) \approx 6.02(N + 2.5L) - 11.14$$

"Equation 29" from Hauser's Principles of Oversampling A/D Conversion. N is the number of bits in the modulator, in our implementation N equals 1, and L is the octaves of oversampling.

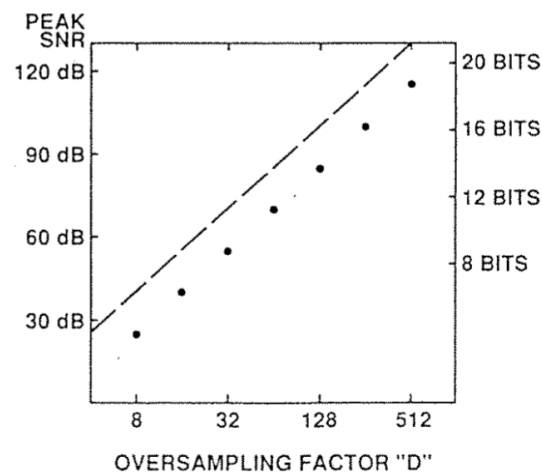


Figure 4: Theoretical "--" and experimental "." values for maximum signal-to-noise ratio (SNR) values after oversampling by a factor D. The experimental values are consistently around 7 dB lower than equation 29 would predict.

In this lab, using the previously created MeasSNR and Decimate subVIs, we created a simulation of this 2nd order Delta Sigma modulator to verify the results in Hauser's paper.

### 3. Design

We used LabVIEW to simulate the modulator and decimation process. The modulator occurs first, and is implemented as seen below. Our signal (amplitude = 100mV, frequency = 5kHz) is unbundled and the discrete values are iterated through to mimic the modulator circuit in figure 3. Two shift registers are used to access previous values of the voltage at the “ $z^{-1}$ ” points and represent the memory characteristic of the system. The signal is transformed into  $\pm 1$  values with the noise shaped into higher frequency levels.

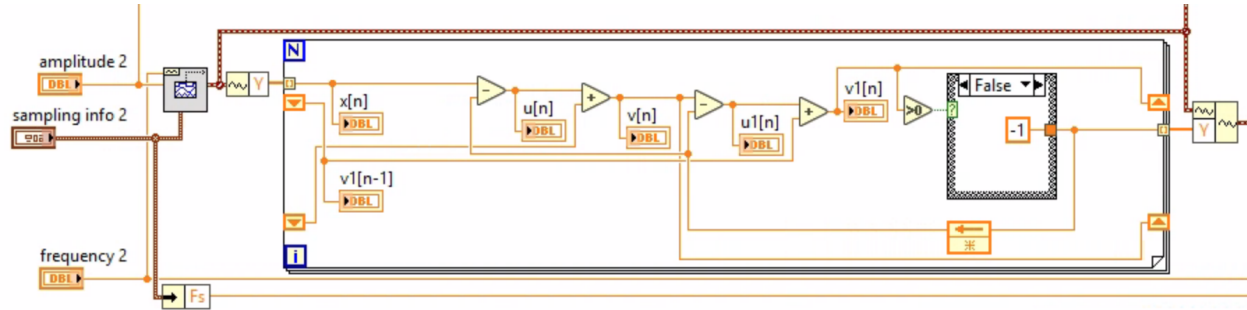


Figure 5: LabVIEW block diagram implementation of second-order delta-sigma modulator. The signal is translated into  $\pm 1$  values as this is a 1-bit modulator.

Next, our modulated signal undergoes decimation of 3 to 8 octaves. At this time, we calculate the expected characteristics based on Equation 29 such as full-scale SNR and number of bits. The experimental and “brick wall” characteristic values are calculated using the “MEAS SNR” subVI that calculates Signal vs Noise RMS values by summing the power spectrum of the input signal. The SNR and bit values are then calculated from the Signal and Noise RMS values and all calculated characteristics are clustered and outputted to the top VI.

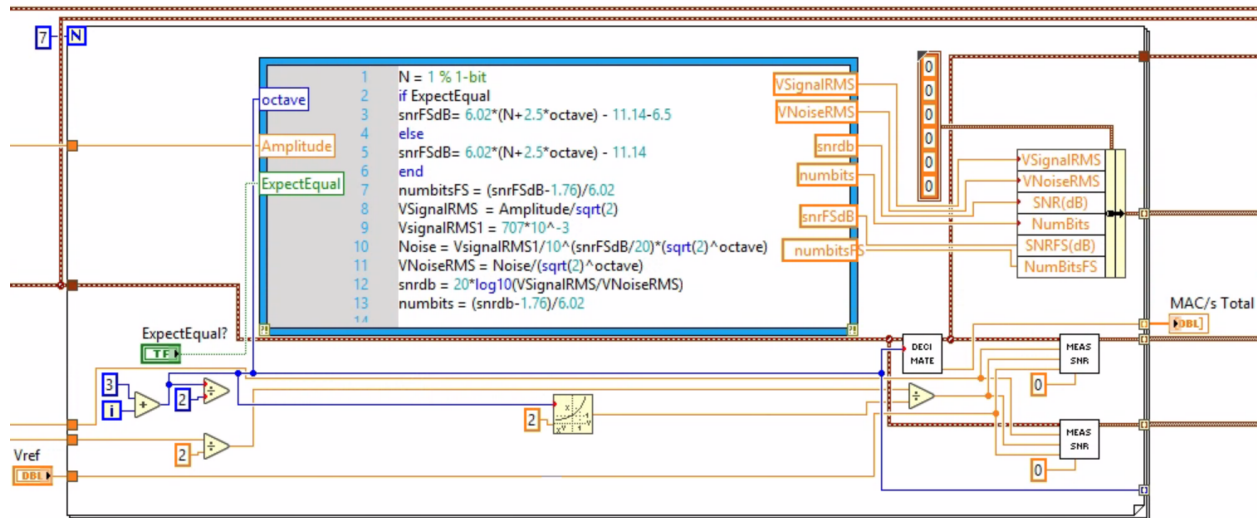


Figure 6: Block Diagram for the decimation factors from 8 to 512. Pre and post-decimation SNR measurements are taken and relative expected values are calculated. Decimation occurs after the noise has been shaped into the higher frequencies with the modulator.

To visualize our findings, we make a graph of the input, modulated, and decimated signals in both the time and frequency domains. We also create the SNR and bit comparison graphs to convey the efficiency gains from decimation as shown in figure 4. Finally, we calculate the bit error between our decimated and modulated signals to ensure our decimation filters properly filter out the noise without significant aliasing occurring.

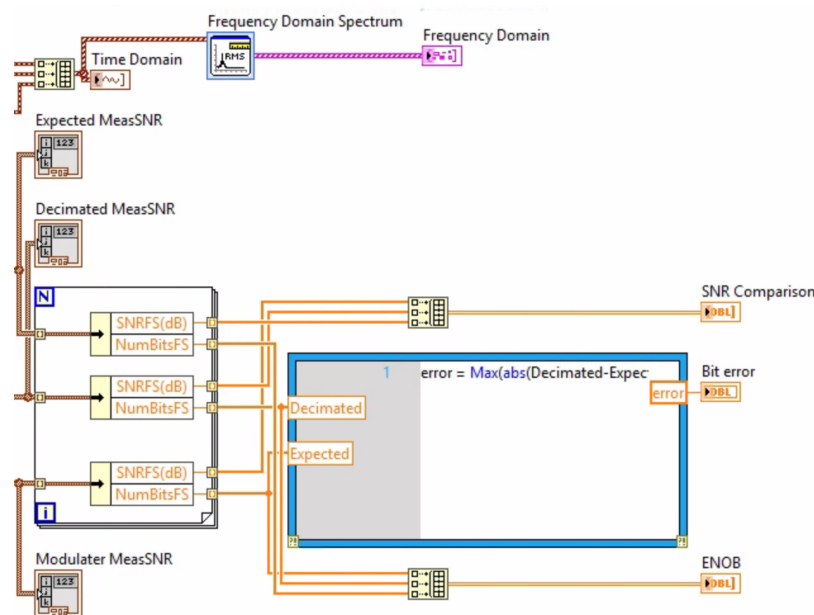


Figure 7: Finally, SNR measurements between Expected, Decimated, and Modulator are compared. Maximum bit error between the Decimated and Modulator terms are to be kept under .5 in this implementation which is achieved by ensuring the two-stage decimation process is strong enough.

Within the DECIMATE subVI, we have a two stage decimation process. The required octaves of decimation are split between the two stages to help minimize the work each filter must do (measured in MAC/s). In order to downsample the signal, a Matlab script is utilized to extract every 2nd, 4th, 8th, etc datapoint, depending on how many octaves of decimation are assigned to that stage. The timestamps of the signal must be maintained by multiplying the time between samples by the decimation factor. This unbundle, downsample, and time-stretch procedure can be seen in figure 8 below.

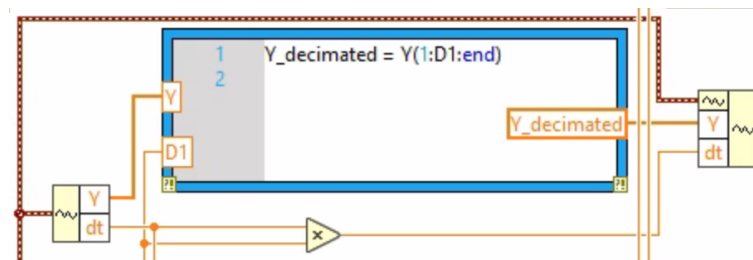
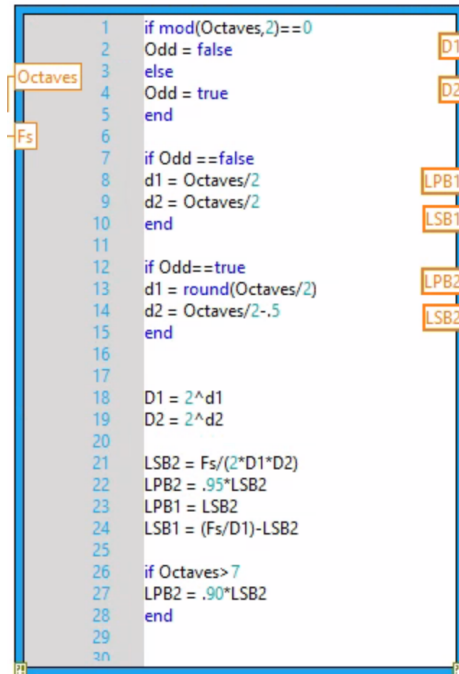


Figure 8: First of two decimation stages. Every “D1th” datapoint is extracted from the original modulated signal and turned back into a waveform with “dt” being updated by multiplying by a factor of D1. This results in a waveform of the same time duration but fewer points by a factor of D1. This also means that “Fs” is being updated by dividing by a factor of D1.

The distribution of decimation octaves and filter cutoff points are calculated using the Matlab script in figure 9 below. The lowercase d’s are the number of octaves per stage while the uppercase D’s represent the factor that dt is scaled by, the signal is sampled at, and Fs is cut down by at each stage. The passband and stopband values were optimized so that the filters have a steep slope at the second stage but as gentle of a slope as possible in the first stage while preventing extreme aliasing.



```

1 if mod(Octaves,2)==0
2   Odd = false
3 else
4   Odd = true
5 end
6
7 if Odd==false
8   d1 = Octaves/2
9   d2 = Octaves/2
10 end
11
12 if Odd==true
13   d1 = round(Octaves/2)
14   d2 = Octaves/2-.5
15 end
16
17
18 D1 = 2^d1
19 D2 = 2^d2
20
21 LSB2 = Fs/(2*D1*D2)
22 LPB2 = .95*LSB2
23 LPB1 = LSB2
24 LSB1 = (Fs/D1)-LSB2
25
26 if Octaves>7
27   LPB2 = .90*LSB2
28 end
29
30

```

Figure 9: Matlab script to distribute octaves of decimation evenly across the two stages, giving the extra octave to the first stage in the case of an odd total number of octaves.

The analysis of the frequency domain graphs at each stage of decimation helped us determine that we must strengthen our decimation filters in order to prevent aliasing and meet the performance requirements.

#### 4. Operation

After hashing out all bugs and conceptual errors, the front panel of our DeltaSigma VI displays the sought after results from this simulation. These include, the frequency and time domain graphs, MeasSNR values, and performance indicators previously mentioned. The following figures display these important results with accompanying analysis.

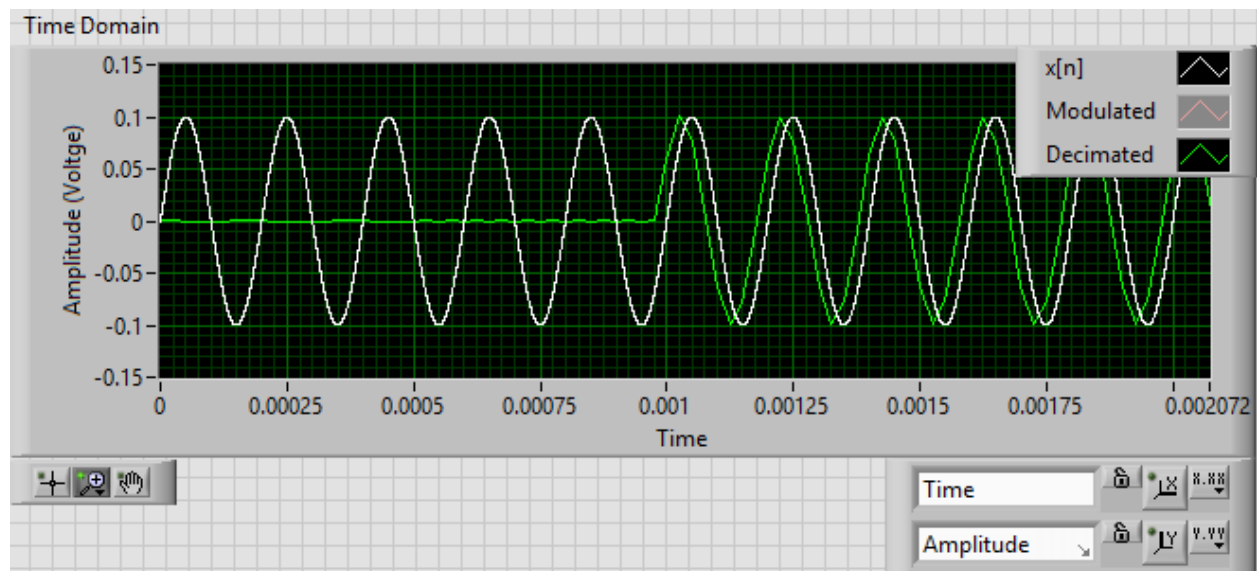


Figure 10: In this time domain graph, we can see the time delay that occurs in the signal after being passed through this modulator/decimator system. The output signal visibly has less data points (it is not as smooth as the original signal) but maintains the proper shape and frequency of the original signal with a much-improved SNR.



Decimated MeasSNR						
VSigalRMS	VSigalRMS	VSigalRMS	VSigalRMS	VSigalRMS	VSigalRMS	VSigalRMS
70.9m	70.6m	70.6m	70.8m	70.9m	70.8m	70.8m
VNoiseRMS	VNoiseRMS	VNoiseRMS	VNoiseRMS	VNoiseRMS	VNoiseRMS	VNoiseRMS
14.9m	2.56m	447u	79.3u	13.3u	2.3u	489n
SNR(dB)	SNR(dB)	SNR(dB)	SNR(dB)	SNR(dB)	SNR(dB)	SNR(dB)
13.6	28.8	44.0	59.0	74.6	89.7	103
NumBits	NumBits	NumBits	NumBits	NumBits	NumBits	NumBits
2.0	4.5	7.0	9.5	12.1	14.6	16.9
SNRFS(dB)	SNRFS(dB)	SNRFS(dB)	SNRFS(dB)	SNRFS(dB)	SNRFS(dB)	SNRFS(dB)
33.5	48.8	64.0	79.0	94.5	110	123
NumBitsFS	NumBitsFS	NumBitsFS	NumBitsFS	NumBitsFS	NumBitsFS	NumBitsFS
5.3	7.8	10.3	12.8	15.4	17.9	20.2

Modulator MeasSNR						
VSigalRMS	VSigalRMS	VSigalRMS	VSigalRMS	VSigalRMS	VSigalRMS	VSigalRMS
70.7m	70.7m	70.7m	70.7m	70.7m	70.7m	70.7m
VNoiseRMS	VNoiseRMS	VNoiseRMS	VNoiseRMS	VNoiseRMS	VNoiseRMS	VNoiseRMS
16.3m	2.78m	488u	86.7u	14.4u	2.73u	492n
SNR(dB)	SNR(dB)	SNR(dB)	SNR(dB)	SNR(dB)	SNR(dB)	SNR(dB)
12.8	28.1	43.2	58.2	73.8	88.3	103
NumBits	NumBits	NumBits	NumBits	NumBits	NumBits	NumBits
1.8	4.4	6.9	9.4	12.0	14.4	16.8
SNRFS(dB)	SNRFS(dB)	SNRFS(dB)	SNRFS(dB)	SNRFS(dB)	SNRFS(dB)	SNRFS(dB)
32.8	48.1	63.2	78.2	93.8	108	123
NumBitsFS	NumBitsFS	NumBitsFS	NumBitsFS	NumBitsFS	NumBitsFS	NumBitsFS
5.2	7.7	10.2	12.7	15.3	17.7	20.2

Expected MeasSNR						
VSigalRMS	VSigalRMS	VSigalRMS	VSigalRMS	VSigalRMS	VSigalRMS	VSigalRMS
70.7m	70.7m	70.7m	70.7m	70.7m	70.7m	70.7m
VNoiseRMS	VNoiseRMS	VNoiseRMS	VNoiseRMS	VNoiseRMS	VNoiseRMS	VNoiseRMS
7.05m	1.25m	220u	38.9u	6.89u	1.22u	215n
SNR(dB)	SNR(dB)	SNR(dB)	SNR(dB)	SNR(dB)	SNR(dB)	SNR(dB)
20.0	35.1	50.1	65.2	80.2	95.3	110
NumBits	NumBits	NumBits	NumBits	NumBits	NumBits	NumBits
3.0	5.5	8.0	10.5	13.0	15.5	18.0
SNRFS(dB)	SNRFS(dB)	SNRFS(dB)	SNRFS(dB)	SNRFS(dB)	SNRFS(dB)	SNRFS(dB)
40.0	55.1	70.1	85.2	100	115	130
NumBitsFS	NumBitsFS	NumBitsFS	NumBitsFS	NumBitsFS	NumBitsFS	NumBitsFS
6.4	8.9	11.4	13.9	16.4	18.9	21.4

Figure 11: Decimated, Modulated, and Expected MeasSNR values. Notably, full-scale SNR increases by approximately 15 dB with each octave of decimation and Decimated NumBits is kept within .5 of Modulator NumBits.

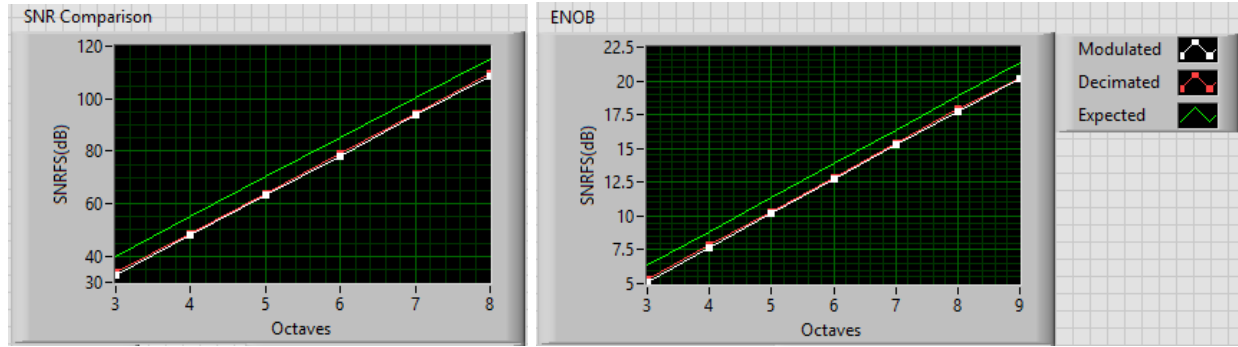


Figure 11: Simulation results mimicking Figure 4 from Hauser. Bits and SNR are linearly correlated so they follow the same improvement trajectory. The results are just under the theoretical values from Equation 29, as expected.

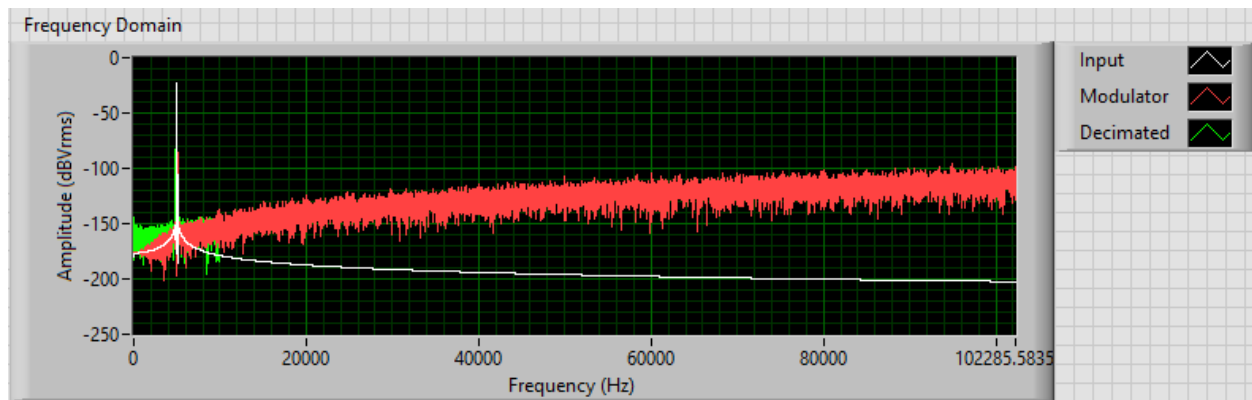


Figure 12: The frequency domain plots display how the modulator shapes the quantization noise into the higher frequency. The decimation process visibly aliases noise into the passband, however it is at acceptable levels given our bit error is within  $\frac{1}{2}$ .

## 5. Discussion/Conclusion

In this lab, we were able to simulate a 2nd order Delta Sigma modulator and replicate Hauser's results in Labview. This simulation yielded results that were consistent with Hauser's expectations. The "brick-wall" filter had values slightly worse than the theoretical values, which is expected. Subsequently, the decimation process did not cause aliasing to an extent that our bit values were off by more than  $\frac{1}{2}$  (bit error = .24 with current parameters). The work done by each filter totals to  $1.8E+9$  MAC/s for 7 octaves of decimation. Since our bit error is only halfway to the  $\frac{1}{2}$  limit, there is room for improvement in terms of weakening the filters in order to be more efficient. A more methodical approach to adjusting filter parameters and octave of decimation distribution could have yielded faster performance. Overall, the implementation was successful, and allowed us to build our continuous time implementation with confidence.