

# Design of an 18-bit, 20kHz Audio Delta-Sigma Analog to Digital Converter

Jon Guerber *ECE 627, Spring 2009*

**Abstract**—The design and simulation of an 18-bit 20kHz analog to digital converter is presented with an oversampling rate of 128 using a cascade of integrators feedback form. The full delta sigma design includes Matlab, Simulink and Cadence simulation of the DS modulator and decimation filter. Non-ideal effects have been investigated and their impact quantified.

## I. INTRODUCTION

The demands on modern analog to digital converters have been increasing rapidly in the past few decades. Cutting edge applications today require higher resolution in technology nodes with ever growing mismatch. The delta sigma modulator offers a solution to achieving high resolution data conversion from low resolution devices with only additional digital overhead and signal oversampling.

This report will detail the design of such a delta sigma converter for audio applications operating at 20 kHz with 18 bits of resolution. The document will begin by covering the basic requirements and design choices for the DS modulator, followed by architectural characteristics and structure, non-ideal effects of the modulator, decimation filtering, and finally circuit level simulations.

## II. BASIC REQUIREMENTS AND DESIGN CHOICES

The design requirements for the proposed delta sigma modulator are located in Table I and a simplified block diagram of the analog to digital converter is shown in Figure 1. A full block diagram can be found later in the paper. Based on the bandwidth, clock frequency, and resolution requirements, the architectural choices shown in the second half of Table I were determined. The oversampling ratio of 128 is the maximum possible value given the clock frequency and the order and resolution were chosen to meet resolution requirements. The value of  $H_{inf}$  was also determined based on stability and resolution tradeoffs.

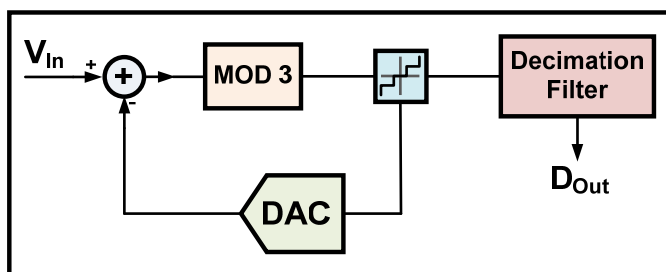


Figure 1: Simplified DS A/D Converter Block Diagram

TABLE I  
DELTA SIGMA DESIGN REQUIREMENTS AND ARCHITECTURE

Parameter	Specified Value
Signal Bandwidth	20 kHz
Clock Frequency	Less than 5.2 MHz
Resolution	18 bits
Oversampling Ratio	128
Order	3 <sup>rd</sup>
Quantized Resolution	2 bits
Architecture	CIFF
$\ H\ $	3

While there are many architectural choices for delta sigma modulators, the cascade of integrators feedback form (CIFF) was chosen due mainly to the reduced  $kT/C$  noise requirements of the first integrator leading to lower unit capacitor values and less die area than other architectures. This architecture also has the benefit of allowing for the removal of the signal from the modulation path, reducing harmonic distortion.

Even with its many advantages, the CIFF modulator does have some downsides. First, the feed forward of integrator outputs requires an accurate active adder before the quantizer which takes up area and power while adding noise. In this design the choice was also made not to send the signal directly to the adder to reduce complexity, power, adder swing, and chip area with a slight degradation in the signal transfer function.

## III. MODULATOR CHARACTERISTICS AND RESPONSE

The response of the 3rd order modulator is shown in Figure 2 with an STF rise described previously. While not typically not desirable, the STF shaping penalty was minimal for the provided benefits. The pole zero diagram is shown in Figure 4 and the zeros have been optimized for the lowest signal band noise. It is important to note that one zero was kept at DC in order to provide DC suppression and prevent the possibility of low frequency tones [1].

The details of modulator NTF and STF can be seen in Figure 3 and Figure 5. The zero optimization can be seen along with the simulated third order noise shaping characteristic. From these diagrams, it's clear that the simulated modulator noise floor is well below the 108 dB limit in the signal band with a simulated ENOB of 22.89 bits. Finally, Figure 6 shows the SQNR achievable with varying input amplitudes. Note that different frequency inputs can have different SQNR responses as is shown.

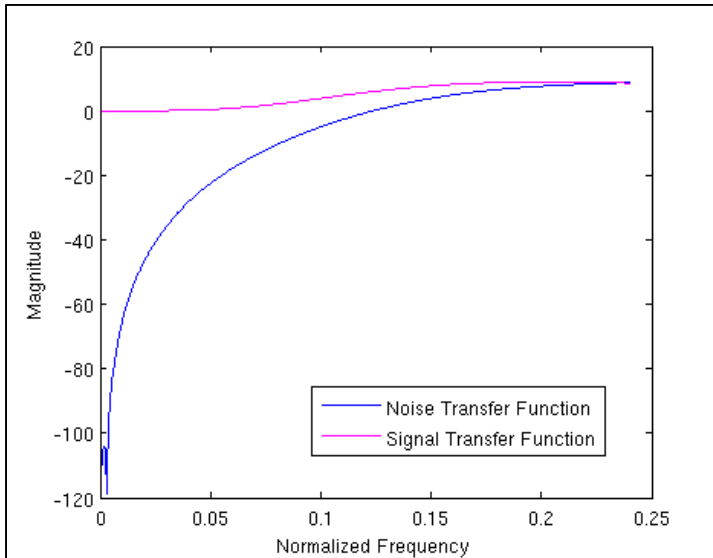


Figure 2: Ideal Modulator NTF and STF Transfer Functions

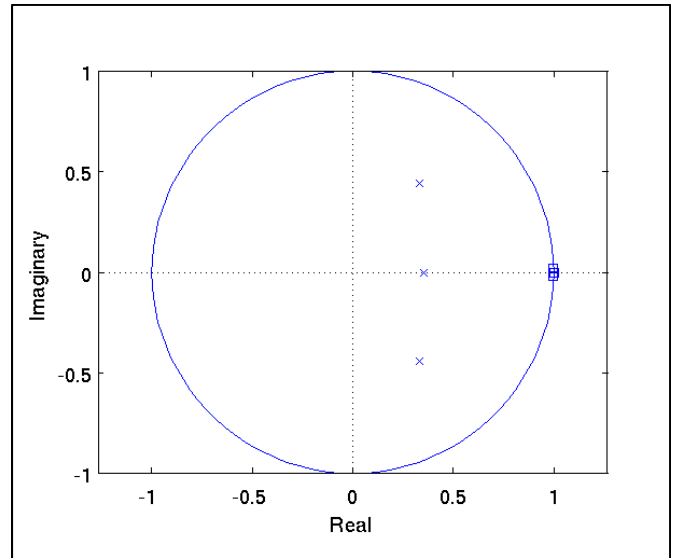


Figure 4: Modulator Pole Zero Diagram

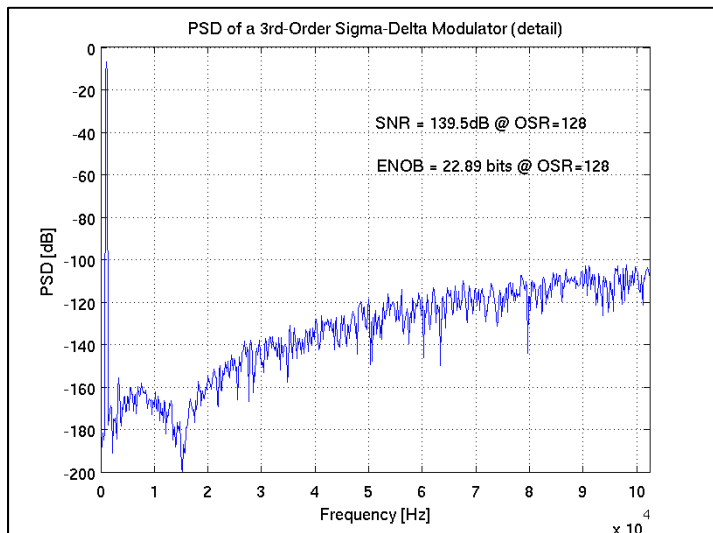


Figure 3: Signal Band Noise of 3rd Order Modulator

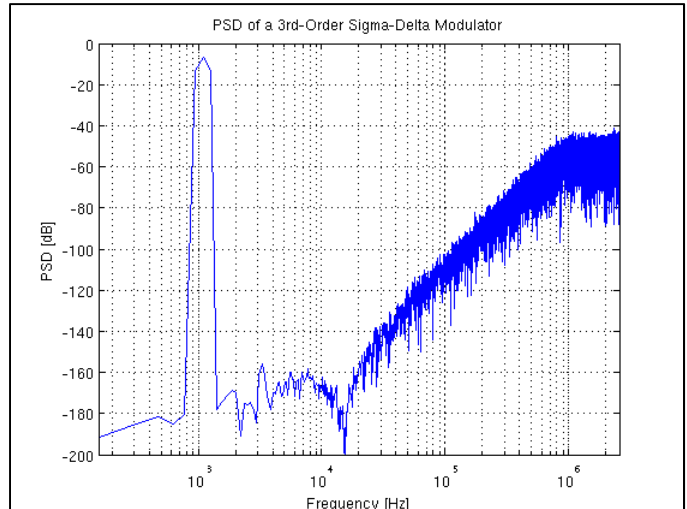


Figure 5: Predicted NTF with 3rd Order Noise Shaping

#### A. Coefficient Selection and Dynamic Range Scaling

Due to the number of coefficients in the typical delta sigma modulator, there are multiple degrees of freedom with which to customize parameters. Using the “Delta Sigma Toolbox” for Matlab [2] it is possible to generate a generic calculated set of these coefficients given a specified architecture and the coefficients for this design are shown in Table II. Using a few of the degrees of freedom, it is possible to adjust the coefficients to limit the swing of each integrator in the modulator to a reasonable value. This is known as dynamic range scaling. Figure 8 shows the three integrator outputs before and after dynamic range scaling. Note that before scaling, an integrator would saturate beyond the desired range of operation at a constant input of only 20% of the max level.

Further coefficient flexibility can be utilized by quantizing the dynamically scaled coefficients to rational quantities. This allows for unit elements to be used in realizing the capacitor ratios, leading to less mismatch and distortion.

TABLE II  
MODULATOR COEFFICIENTS

Parameter	Calculated Value	Dynamically Scaled Value	Quantized Value
A1	1.982126	2.1035432	21/100
A2	1.504027	2.101099	21/100
A3	0.413137	2.3013556	23/100
B1	1	0.9422797	9/100
G1	3.61392e-4	.00144106	.0009
C1	1	0.9422797	9/100
C2	1	0.75967756	8/100
C3	1	0.25078159	3/100

#### IV. MODULATOR NON-IDEALITIES

Like any analog system, there are inherent non-idealities. Here the effects of common non-idealities will be examined in the context of the delta sigma modulator.

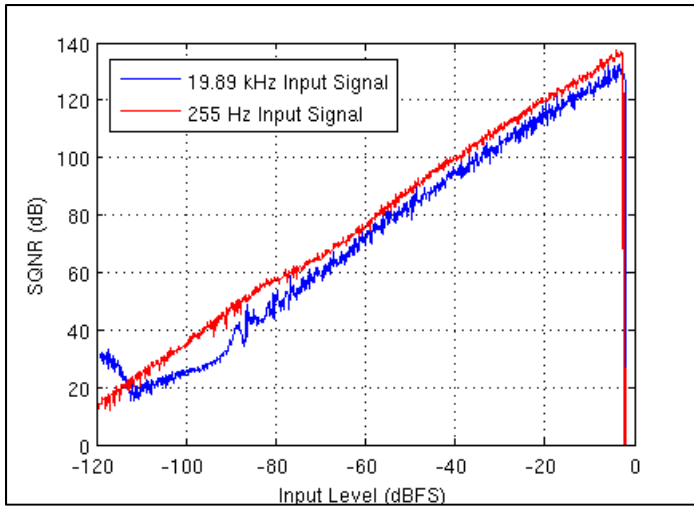


Figure 6: SQNR vs. Input Amplitude

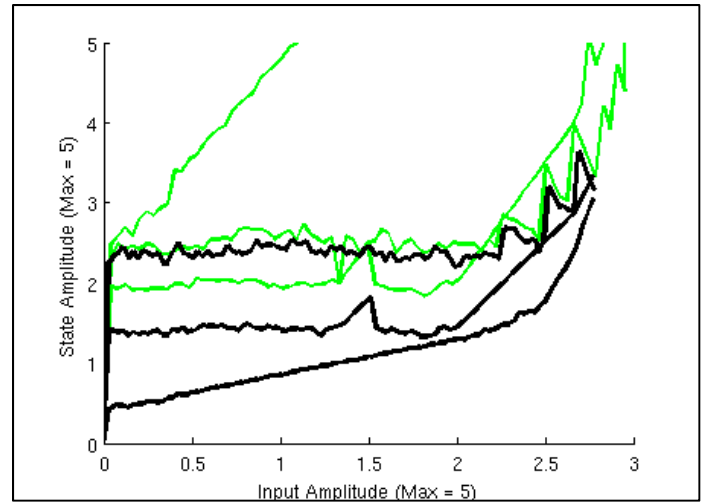


Figure 8: Modulator Integrator Output Amplitudes Before (Green) and After (Black) Dynamic Range Scaling

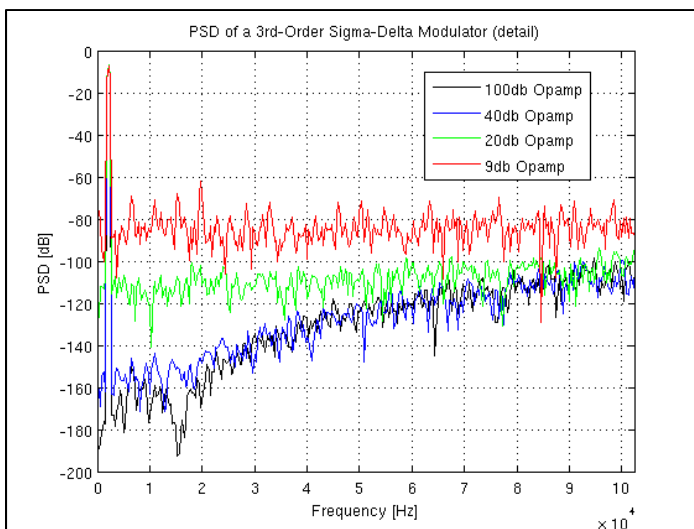


Figure 7: Effect of Finite Opamp Gain on Signal Band

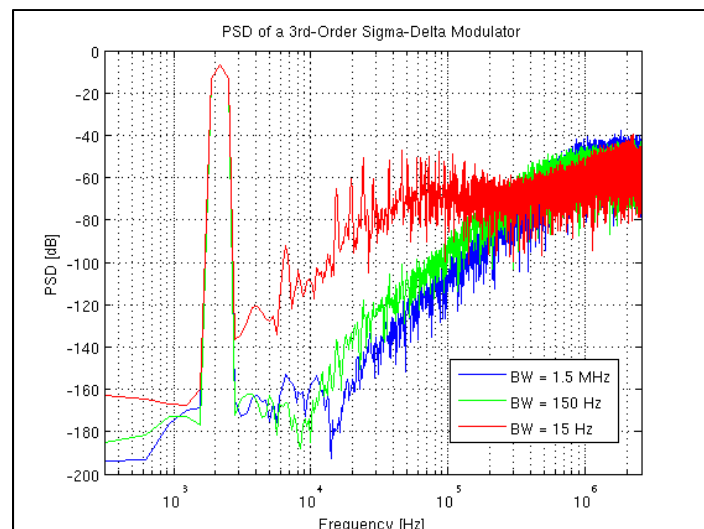


Figure 9: Effect of Opamp Bandwidth on Modulator Spectrum

### A. Finite Opamp Gain

In order to realize effective integrators, the use of opamps is required. However, opamps can be power hungry devices and much of this power dissipation is related to the gain of the device. It can be shown that loop gain error given by  $A/(1+A)$  will shift the modulator poles from their desired locations [3]. This in turn will affect the NTF zeros, moving the signal band noise higher as can be seen in Figure 7.

Gain error in a switched capacitor delta sigma modulator can often be seen as DC static settling error during each phase, thus distorting the lower frequency band. This makes many other non-idealities (such as slew rate) look like gain error to a first degree. The amount of opamp gain for a given delta sigma modulator should be determined by the amount of tolerable signal band distortion.

### B. Finite Opamp Bandwidth

The bandwidth of integrator opamps will determine the time constant for the settling after each clock phase. With limited bandwidth, there will be some settling error determined by the size of the input voltage step, dominant pole time constant, and feedback factor (beta) as shown in Equation 1.

$$\varepsilon_{Set} = V(1 - e^{-T\beta/\tau})$$

Equation 1: Settling Error from Finite Bandwidth

From intuition about the nature of settling time, it is clear that higher frequency tones will have a larger voltage range to travel and thus the potential for more bandwidth induced settling error. This is shown in Figure 9 where much of the distortion occurs in frequencies above the signal band. The finite bandwidth limitation appears much like gain error, only affecting more of the modulator spectrum.

### C. Slew Rate Limitations

One of the more dynamic non-idealities of a delta sigma modulator is slew rate. Lowering the slew rate slightly won't affect the modulator performance as long as the integrator outputs can settle by the end of each phase. However, once this condition is not met, there is a drastic increase of inband noise as can be seen in Figure 10.

It is often useful to know the necessary slew rate requirements for opamps in a given delta sigma modulator. For this design, assuming the opamps must slew for 20% of

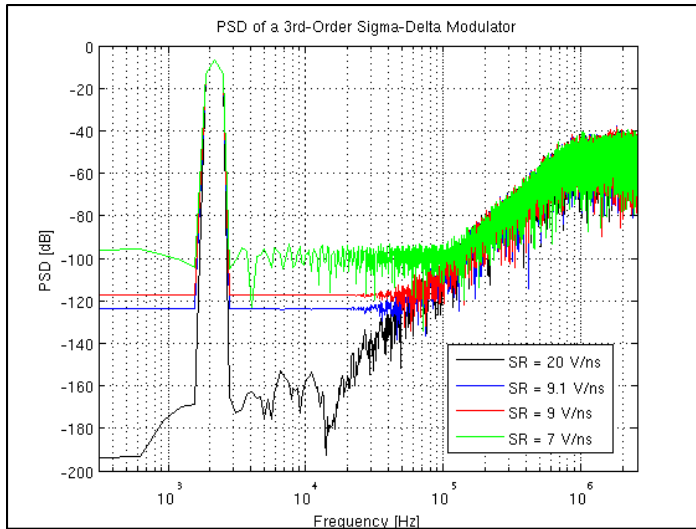


Figure 10: Impact of Slew Rate on Modulator Performance

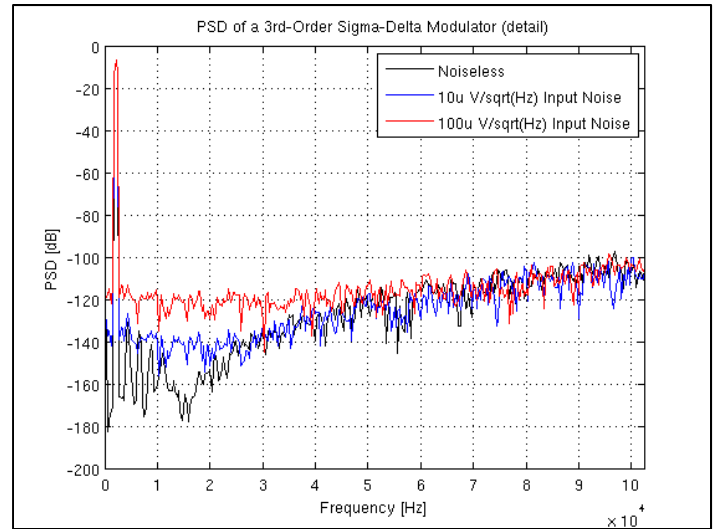


Figure 12: Impact of Analog Noise on Modulator Signal Band

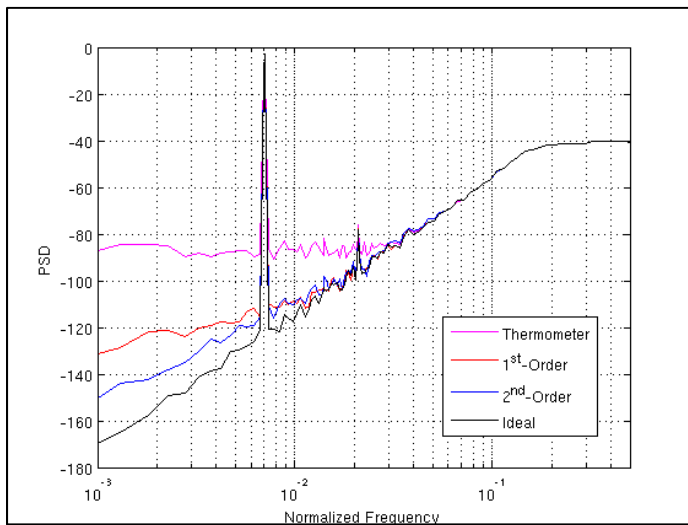


Figure 11: 1% Capacitor Mismatch with Error Shaping and Ideal Devices

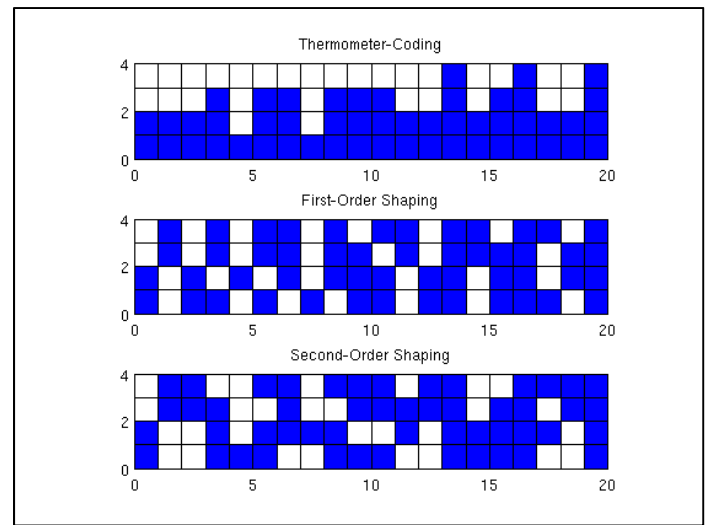


Figure 13: Noise Shaping Element Selection Algorithms

each period and must reach 60% of the fullscale integrator output (assuming a couple  $\tau$  of settling), the opamp slew rate can be approximated by the following equation (Normalized to a 1V step):

$$SR \approx \frac{V_{Step} f_{clk} (2)}{.2} = 30.769 \frac{V}{\mu s}$$

By following the above slew rate formula, it is clear that slew rate induced distortion can be avoided.

#### D. Analog Noise

While analog noise is an expected phenomenon in delta sigma data converters and is typically shaped to give the desired converter resolution, too much can degrade performance. Particularly harmful is flicker noise, which will naturally inhabit the signal band of audio delta sigma modulators. Figure 12 shows the impact of too much white noise degrading the signal band, even with noise shaping.

#### A. Capacitor Mismatch

One of the early factors limiting delta sigma designers to single bit quantizers was the problem of capacitor mismatch. As shown in Figure 11, a typical thermometer coded capacitive DAC can increase the noise floor in the signal band by nearly 30dB. This effect was combated by techniques such as dynamic element matching which can provide capacitive error shaping to make higher order quantizers realizable. First order error shaping can be realized with a relatively simple data weighted average scheme, where as higher orders require more complex algorithms. First and second order shaping techniques for this modulator are shown in Figure 13.

#### B. Digital Truncation Errors

Since delta sigma modulators leverage a lot of digital hardware to perform their data conversion, it is expected that they would suffer from truncation errors. Figure 16 shows the designed modulator's response to decimation filtering given 4 bit, 11 bit, and full accuracy digital processing. Clearly for 18-bit delta sigma modulation, high digital accuracy is required in filtering. However, digital circuits are becoming increasingly cheaper and power efficient.

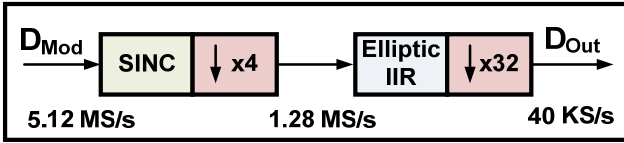


Figure 14: Digital Decimation Filter Block Diagram

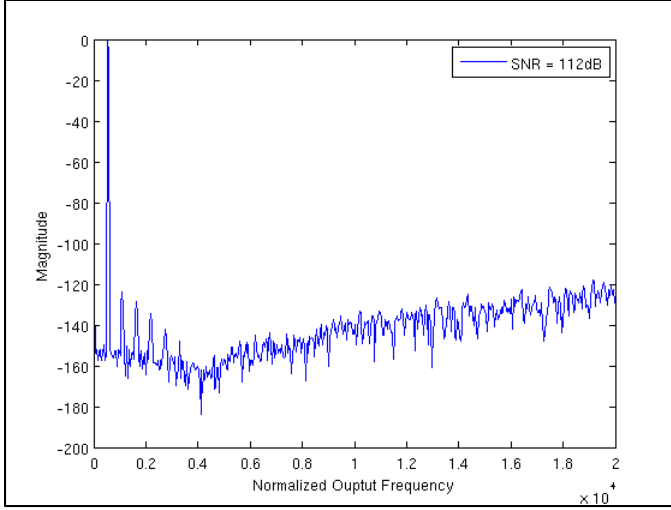


Figure 15: Modulator Spectrum After Decimation

### C. Other Delta Sigma Error Sources

Some other common delta sigma error sources include clock jitter and switch non-linearities. Both of these have the potential to cause settling and accuracy issues and require attention in full modulator chip design.

## V. DECIMATION FILTERING

In order to remove the shaped noise and recover the signal from an over sampled delta sigma modulator, both decimation and filtering are required. As shown in Figure 14, this modulator has a 4th order sinc filter followed by a 12th order elliptic IIR filter.

The sinc filter was implemented as a Hogeauier structure with four cascaded integrators followed by decimation, then followed by four delay additions [4]. This filter allows for the more computationally intensive filtering to be completed at a lower sampling rate. One drawback however is that the sinc filter injects droop at higher frequencies and cannot be used as the only filter.

Following the sinc filter is a 12th order elliptic filter chosen based on SNR requirements. The elliptic structure offers the sharpest cutoff frequency for the lowest order, with some penalty in signal phase and group delay. This filter showed no problems in reducing the noise beyond 18 bits (< 108 dB). Following the elliptic filter is a decimation by 32. Filter specifications can be found in Table III while the post-filtering modulator spectrum is shown in Figure 15.

## VI. CIRCUIT LEVEL SIMULATIONS

The final step in the design of the delta sigma modulator was circuit level simulations in Cadence. To begin the design, a unit capacitor value had to be determined from  $kT/C$  noise requirements shown in the following equation:

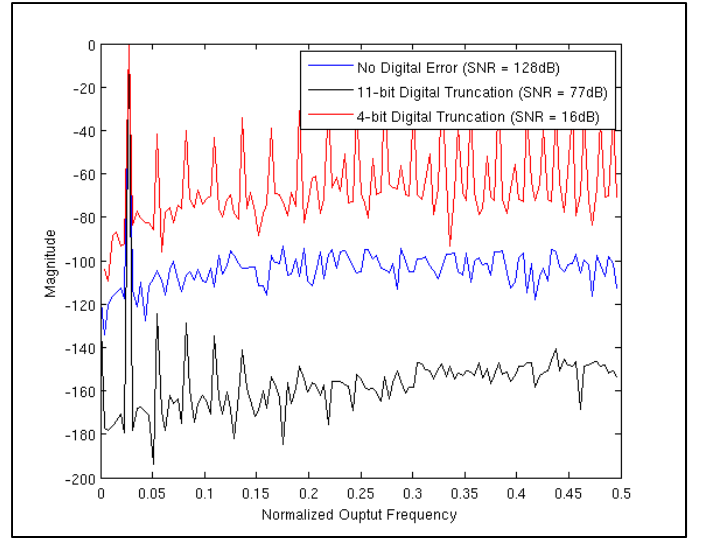


Figure 16: Digital Truncation Error Effects on Noise after Decimation (Black = No Truncation, Blue = 11 bit, Red = 4bit)

TABLE III  
DECIMATION FILTER SPECIFICATIONS

Parameter	Specified Value
Sinc Order	4
Sinc Decimation	x4
Elliptic IIR Order	12
Elliptic Passband Ripple	.1 dB
Elliptic Stopband Attenuation	135 dB
Elliptic Decimation	x32

$$C_c = \frac{2kT(10^{115/10})}{V_{Amp} * OSR \left( \frac{1}{b_1} + \frac{1}{c_1} \right)} \approx 45pF$$

Equation 2: First Stage Unit Capacitor Value

While the unit capacitor needed for noise requirements is large, it can be replaced with a smaller capacitor in subsequent stages as the noise requirements drop. The approximate total capacitance load of the simulated circuit was 137 pF.

The cadence simulated block diagram is shown in Figure 17. Some of the Cadence time domain output is shown in Figure 19 which is not as high of resolution as expected. A full switched capacitor implementation can be found in Figure 18.

## VII. CONCLUSION

This paper has presented a 18-db audio delta sigma digital to analog converter and discussed major design choices, simulations, and non-idealities. With the scaling of technology and the efficiency of digital systems, delta sigma converters will become even more prevalent and it is imperative that we take the time to correctly understand the design challenges and potential uses of the structure.

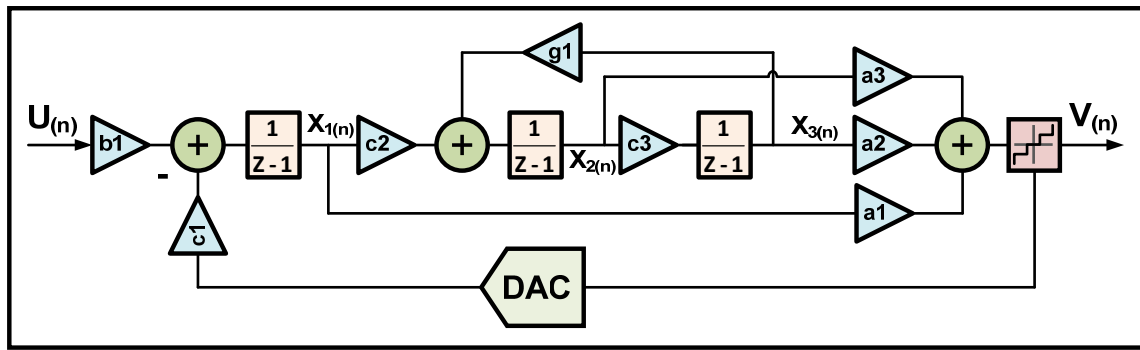


Figure 17: Block Diagram of the Implemented CIFF Modulator Structure

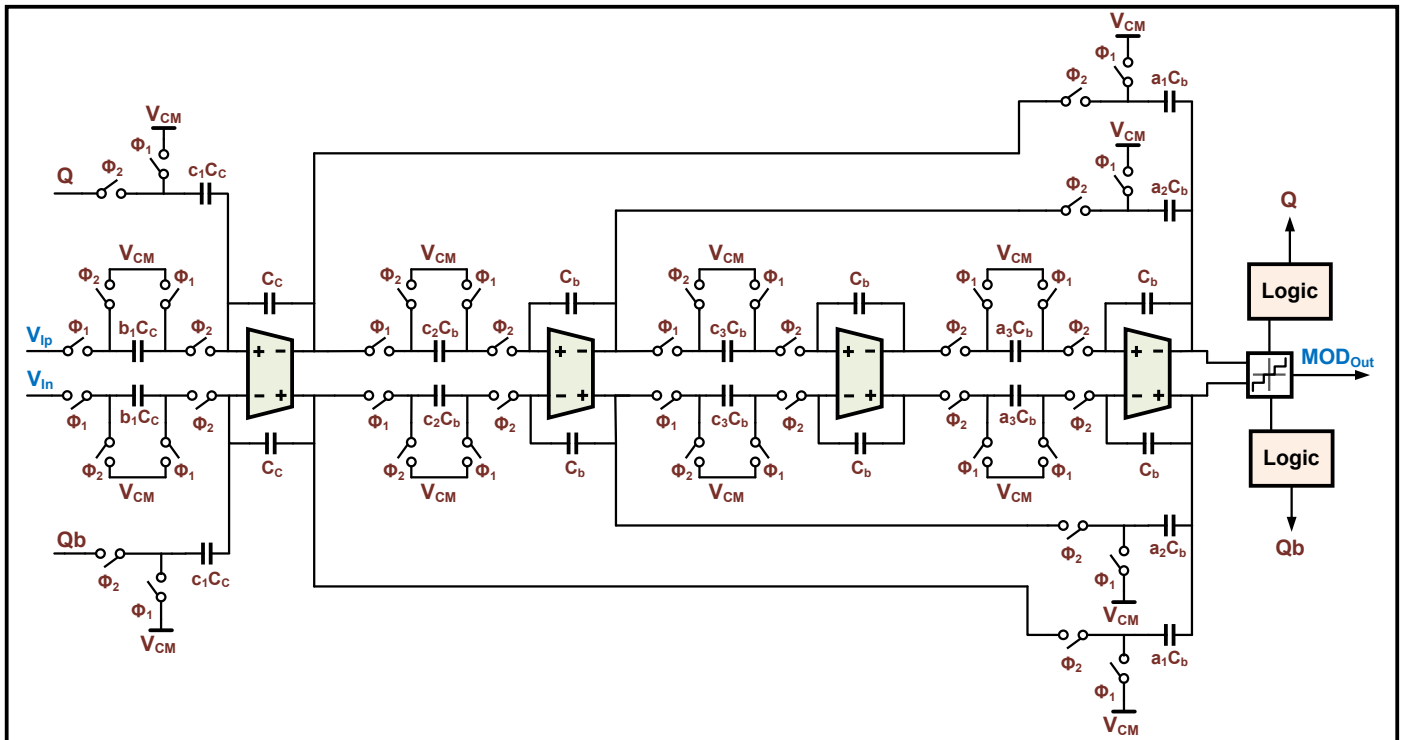


Figure 18: Switched Capacitor Modulator Implementation

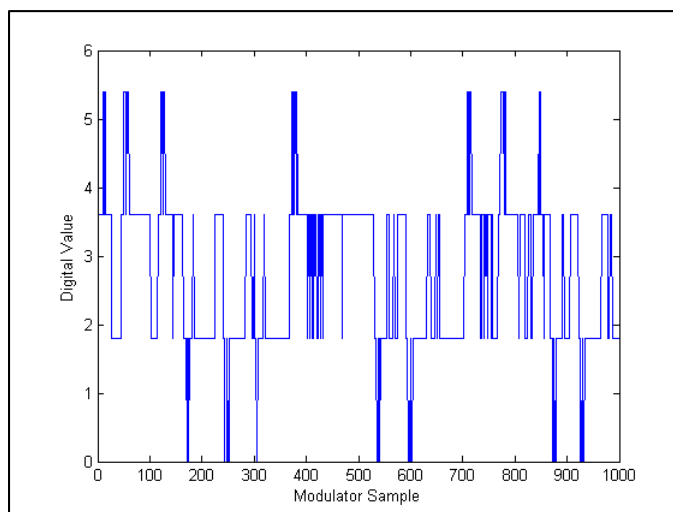


Figure 19: Simulated Modulator Time Domain Output

TABLE IV  
PERFORMANCE SUMMARY

Parameter	Value
Resolution	18-Bit
Signal Bandwidth	20 kHz
Oversampling Rate	128
SNR Pre-filtering	139 dB
SNR Post-Filtering	112 dB
Filtering	4 <sup>th</sup> order sinc / 12 <sup>th</sup> order Elliptic
Total Capacitance	≈ 137 pF
Operating Voltage	1.8 V
Max Voltage Level	1.08 V
Modulator Architecture	CIFF

## VIII. REFERENCES

- [1] R. Schreier, G. Temes, "Delta Sigma Data Converters," Wiley Publishing, 2005.
- [2] R. Schrier, "The Delta Sigma Toolbox for Matlab," Jan, 2000, <http://www.mathworks.com/matlabcentral/fileexchange/19>.
- [3] F. Maloberti, "Data Converters," University of Pavia, 2007.
- [4] E. Hogenauer, "An Economical Class of Digital Filters for Decimation and Interpolation," IEEE Transactions on Acoustics, Speech, and Signal Processing, April 1981.

## IX. APPENDIX A: MATLAB SCRIPTS

## A. Modulator Simulation Script

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% The Delta Sigma Project By Jon Guerber
%
% May 30, 2009      ECE 627
%
% This Script will determine Delta Sigma Parameters,
Plots, and Coefficients
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mex('delsig/simulatedDSM.c')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Part 1 -- Initialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Add path for the delta Sigma Toolbox
addpath('/nfs/spectre/u9/guerberj/Classes/ECE 627 --
Data Converters (Temes)/Final Project/delsig')

% Adding the Specified Parameters
OSR = 128;
N = 3;
opt = 2;
Nlev = 5;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Part 2 -- Finding Ideal STF and NTF
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

H = synthesizENTF(N,OSR,opt,3)

plotPZ(H)
xlabel('Real');
ylabel('Imaginary')

f = [linspace(0,0.75/OSR,100)
linspace(0.75/OSR,0.5,100)];
z = exp(2i*pi*f);
magH = dbv(evalTF(H,z));
subplot(221);
plot(f,magH);
figureMagic([0 0.1],0.01,2, [-100 10],10,2 );
xlabel('Normalized frequency (1\rightarrow f_s)');
ylabel('dB')
title('NTF Magnitude Response')

fstart = 0.01;
f = linspace(fstart,1.2,200)/(2*OSR); z =
exp(2i*2*pi*f);
magH = dbv(evalTF(H,z));
subplot(223);
semilogx(f*2*OSR,magH);
axis([fstart 1.2 -100 -30]);
grid on
sigma_H = dbv(rmsGain(H,0,0.5/OSR));
hold on;
semilogx([fstart 1], sigma_H*[1 1]);
plot([fstart 1], sigma_H*[1 1],'o');
text( 0.15, sigma_H+5, sprintf('rms gain =
%5.0fdB',sigma_H));
xlabel('Normalized frequency (1\rightarrow f_B)');
ylabel('dB')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Part 3 -- Simulate the DSM Response
(FFT) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Nfft = 2^13;
tone_bin = 57; % Sine Wave Input
Frequency
t = [0:Nfft-1];
%u = .05*(nLev-1)*sin(2*pi*tone_bin/Nfft*t);
fB = ceil(Nfft/(2*OSR)); ftest=floor(2/3*fB)

u = 0.5*sin(2*pi*ftest/Nfft*[0:Nfft-1]); % half-
scale sine-wave input
v = simulatedDSM(u,H,Nlev);
n = 1:500;
subplot(224)
stairs(t(n),u(n),'g'); % Plot the Time-Domain
Output
hold on
stairs(t(n),v(n),'b'); % Plot it with the
Signal

f = linspace(0,0.5,Nfft/2+1);
echo on
spec = fft(v.*ds_hann(Nfft))/(Nfft/4); % Get the
FFT Spectrum
echo off;
figure; clf;
subplot(222)
plot(f, dbv(spec(1:Nfft/2+1)), 'b')
figureMagic([0 0.5], 0.05, 2, [-120 0], 20,
1, [], 'Output Spectrum');
xlabel('Normalized Frequency')
ylabel('dBFS')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Part 4 -- Calculate and Plot SNR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

amp = linspace(-120,0,1000);
%amp = [-120, -100, -80, -60, -40, -20,-10,-6,-4,-
3,-2.5,-2,-1.5,-1,0]
[snr, amp] = simulateSNR(H,OSR,amp,0,Nlev,.0039,13);
%text_handle = text(0.05,-10, sprintf('SNR = %4.1fdB
@ OSR = %d',max(snr),OSR),'vert','middle');

plot(amp,snr,'b');
hold on
[snr, amp] =
simulateSNR(H,OSR,amp,0,Nlev,.00005,13);
%text_handle = text(0.05,-10, sprintf('SNR = %4.1fdB
@ OSR = %d',max(snr),OSR),'vert','middle');
plot(amp,snr,'r');
legend('19.89 kHz Input Signal','255 Hz Input
Signal')

%figureMagic([-120 1], 10, 1, [0 140], 10,
1, [], 'SQNR');
xlabel('Input Level (dBFS)');
ylabel('SQNR (dB)');
axis([-120 0 0 140])
grid on
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Part 5 -- Find Modulator Coefficients
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

form = 'CIFF';
[a,g,b,c] = realizeNTF(H,form)
b(2:end) = 0;

% Plot the response with chosen coefficients
ABCD = stuffABCD(a,g,b,c,form);
[ntf, stf] = calculateTF(ABCD);
f = linspace(0,0.24,1000);
z = exp(2i*pi*f);
magNTF = dbv(evalTF(ntf,z));
magSTF = dbv(evalTF(stf,z));
subplot(224)
figure
plot(f,magNTF,'b',f,magSTF,'m','LineWidth',1)
xlabel('Normalized Frequency');
ylabel('Magnitude');
legend('Noise Transfer Function','Signal Transfer
Function')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Part 6 -- Preform Dynamic Range
Scaling %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

u = linspace(0,(Nlev-1),100); % Input
Signal (Tested one value at a time)

```

```

Np = 1e4; % Number
of same input points to test at time %
T = ones(1,Np); %
Normalized input signal = 5 (Nlev) %
maxima = zeros(N,length(u)); %
for i = 1:length(u) %
    ui = u(i); %
    [v,xn,xmax] = simulateDSM( ui(T), ABCD, Nlev ); %
% Get Values for the max of each output node %
maxima(:,i) = xmax(:); %
if any(xmax>1e2) % Why %
100 ??? %
    umax = ui % Max %
Stable input (Only stable to 2.88) %
u = u(1:i); %
maxima = maxima(:,1:i); %
    break; %
end %
figure; clf %
for i = 1:N %
    %plot(u,maxima(i,:), 'o'); %
    if i==1 %
        hold on; %
    end %
    plot(u,maxima(i,:), 'g-', 'linewidth', 2); %
end %
axis([0 3 0 10]) %
xlabel('Input Amplitude (Max = 5)') %
ylabel('State Amplitude (Max = 5)') %

x = Nlev.*([.64 .638 .60]) %
[ABCDs,umax] = scaleABCD(ABCD,Nlev,0,x,Nlev+5, [],Np) %
[as,gs,bs,cs] = mapABCD(ABCDs,form) %

u = linspace(0,umax,100); %
for i = 1:length(u) %
    ui = u(i); %
    [v,xn,xmax] = simulateDSM( ui(T), ABCDs,Nlev ); %
    maxima(:,i) = xmax(:); %
    if any(xmax>1e2) %
        umax = ui; %
        u = u(1:i); %
        maxima = maxima(:,1:i); %
        break; %
    end %
end %
%figure; clf %
for i = 1:N %
    %plot(u,maxima(i,:), 'ro'); %
    if i==1 %
        hold on; %
    end %
    plot(u,maxima(i,:), 'k-', 'linewidth', 3); %
    axis([0 3 0 5]) %
    xlabel('Input Amplitude (Max = 5)') %
    ylabel('State Amplitude (Max = 5)') %
    %legend('Pre-Dynamic Range Scaling',' ',' ','Post- %
Dynamic Range Scaling') %
end %

% Part 7 - "Quantize" The Cap Values %
% %
R = 100; % Rounding Factor %
asn = round([as].*R)./R %
gsn = gs; %
%gsn = round([gs].*R)./R %
bsn = round([bs].*R)./R %
csn = round([cs].*R)./R %

[a1,g1,s1,c1] = mapABCD(ABCD,form) %

```

## B. Non-Ideal Effects Matlab Script

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% The Delta Sigma Non-Ideal Effects Simulink Script
%
% 6-11-09
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic
addpath('/nfs/spectre/u9/guerberj/Classes/ECE 627 --
Data Converters (Temes)/Final Project/SDtoolbox')
t0 = clock;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part 1 - Input Signal Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
bw = 20e3; % Input Signal Bandwidth
OSR = 128; % Oversampling Ratio
Fs = OSR*2*bw; % Oversampling
Frequency
Ts = 1/Fs; % Sampling Period
N = 2^14; %17 for decimation %
FFT Sample Number
Sig_Bin = 7; % Signal Bin (Prime
Number)

Ampl = .8-pi/128; % Sine Wave
Amplitude
fin = Sig_Bin*Fs/N; % Input Frequency (in
Hz)
Fin = fin;
finrad = fin*2*pi; % Input Frequency in
Radians
Ntransient = 0; % ?? FFT parameter

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part 2 - Coefficients Input
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a1 = asn(1); a2 = asn(2); a3 = asn(3)
b1 = bsn(1); b4 = bsn(4)
c1 = csn(1); c2 = csn(2); c3 = csn(3);
g1 = gsn(1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part 3 - Integrator and Quantizer
Parameters %
alfa = (1e5-1)/1e5; % Opamp Finite Gain
(Loop Gain...)
Amax = 135; % Saturation Limit
sr = 2000000e6; % Slew Rate
GBW = 150000e6; % Gain Bandwidth Product
noise1 = 0*100e-6; % First
integrator output noise sigma [V/sqrt(Hz)]
delta = 0; % Random Sampling Jitter
Standard Deviation [s]
NCOMPARATORI = 5; % 2-bit Quantizer
match = 9e-10; % Matching Accuracy
ref = Amax;

k = 1.38e-23; % Boltzman's Constant
Temp = 300; % Kelvin Temperature
Cs = 40e-12; % First Integrator
Integrating Cap

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part 4 - Simulation Code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
open_system('./Delta_Sigma_CIFF') % Opens the
simulink model if not already present
options=simset('InitialState', zeros(1,3), 'RelTol',
1e-3, 'MaxStep', 1/(Fs));
sim('./Delta_Sigma_CIFF', (2*N+Ntransient)/Fs, options
); % Simulates the model

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part 5 - Calculate SNR and PSD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
w = hann_pv(N); % Hannin window to
ensure an interger # of input sines

```



```

f = Fin/Fs; % Normalized Input Frequency
fB = N*(bw/Fs); % Baseband frequency limit
yy1 = zeros(1,N); % Preload array
yy1 = yout(2+Ntransient:1+N+Ntransient); % Don't know what this does but it works

ptot = zeros(1,N); % Preload array
[snr,ptot] = calcSNR(yy1(1:N),f,1,fB,w,N) % Find the SNR
Rbit = (snr-1.76)/6.02; % Resolution in bits

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Part 6 - Plot the output
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1);
clf;
plot(linspace(0,Fs/2,N/2), ptot(1:N/2), 'b');
grid on;
title('PSD of a 3rd-Order Sigma-Delta Modulator')
xlabel('Frequency [Hz]')
ylabel('PSD [dB]')
axis([0 Fs/2 -200 0]);

figure(2);
clf;
semilogx(linspace(0,Fs/2,N/2), ptot(1:N/2), 'k');
grid on;
title('PSD of a 3rd-Order Sigma-Delta Modulator')
xlabel('Frequency [Hz]')
ylabel('PSD [dB]')
%legend('BW = 1.5 MHz','BW = 150 Hz', 'BW = 15 Hz', 'BW = 1.5 Hz')
%legend('SR = 20 V/ns','SR = 9.1 V/ns', 'SR = 9 V/ns','SR = 7 V/ns')
axis([0 Fs/2 -200 0]);
hold on

figure(3);
clf;
plot(linspace(0,Fs/2,N/2), ptot(1:N/2), 'r');
hold on;
title('PSD of a 3rd-Order Sigma-Delta Modulator (detail)')
xlabel('Frequency [Hz]')
ylabel('PSD [dB]')
axis([0 2*(Fs/R) -200 0]);
grid on;
%legend('100db Opamp','40db Opamp','20db Opamp','9db Opamp')
legend('Noiseless','10u V/sqrt(Hz) Input Noise','100u V/sqrt(Hz) Input Noise')

% hold off;
% text_handle = text(floor(Fs/R),-40, sprintf('SNR = %4.1fdB @ OSR=%d\n',snr,OSR));
% text_handle = text(floor(Fs/R),-60, sprintf('ENOB = %2.2f bits @ OSR=%d\n',Rbit,OSR));
% hold on

% s1=sprintf(' SNR (dB)=%1.3f',snr);
% s2=sprintf(' Simulation time =%1.3f min',etime(clock,t0)/60);
disp(s1)
disp(s2)
toc

C. Decimation Filter Script
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Decimation Filter %%%%%%%%%
%
% This Script will plot the signal response to both a fourth order sinc
% and 12th order Elliptic Filter
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic
mex('delsig/simulateDSM.c')
addpath('/nfs/spectre/u9/guerberj/Classes/ECE 627 -- Data Converters (Temes)/Final Project/delsig')

% plot(f,dbv(frespHBF(f,f1,f2)))
OSR = 4;
%OSR2 = 128/OSR;
N = 18;
%% Brute Force Decimation (Hogenauer)
%youtt = yout(2^14:end);

p = 1000;
y1 = zeros(length(yout),1);
y2 = zeros(length(yout),1);
y3 = zeros(length(yout),1);
y4 = zeros(length(yout),1);

size(y1(2:end))
size(yout(2:end))
size(yout(1:(end-1)))

y1(1:end) = cumsum(yout(1:end)); y1 = round(y1.*p)/p; p1 = max(y1); % y1 = round(y1.*(p1*p))/(p*p1);
y2(1:end) = cumsum(y1(1:end)); y2 = round(y2.*p)/p; p2 = max(y2); % y2 = round(y2.*(p2*p))/(p*p2);
y3(1:end) = cumsum(y2(1:end)); y3 = round(y3.*p)/p; p3 = max(y3); % y3 = round(y3.*(p3*p))/(p*p3);
y4(1:end) = cumsum(y3(1:end)); y4 = round(y4.*p)/p; p4 = max(y4); % y4 = round(y4.*(p4*p))/(p*p4);

y4d = zeros((length(y4)/OSR),1);
y5 = zeros(length(y4d),1);
y6 = zeros(length(y5),1);
y7 = zeros(length(y6),1);
y8 = zeros(length(y7),1);

size(y4)

for i=1:1:(length(y4)/OSR)
    y4d(i) = y4(i*OSR);
end

y5(1) = y4d(1);
y5(2:end) = y4d(2:end) - y4d(1:(end-1)); y5 = round(y5.*p)/p; p5 = max(y5); %y5 = round(y5.*(p5*p))/(p*p5);

y6(1) = y5(1);
y6(2:end) = y5(2:end) - y5(1:(end-1)); y6 = round(y6.*p)/p; p6 = max(y6); %y6 = round(y6.*(p6*p))/(p*p6);

y7(1) = y6(1);
y7(2:end) = y6(2:end) - y6(1:(end-1)); y7 = round(y7.*p)/p; p7 = max(y7); %y7 = round(y7.*(p7*p))/(p*p7);

y8(1) = y7(1);
y8(2:end) = y7(2:end) - y7(1:(end-1)); y8 = round(y8.*p)/p; p8 = max(y8); %y8 = round(y8.*(p8*p))/(p*p8);

%plot(yout)
%figure
y9 = y8(257:end)./(OSR^4);
length(y9)
g = linspace(0,.00005,256);
%plot(g,y9)
xlabel('Time (s)')
ylabel('Normalized Voltage')

%figure

Np = (2^N)/OSR;
w1 = Hann(length(y8));

```

```

f=(0:1:(Np)-1)*(1);
fftouts = abs(fft(y8.*w1'));
m = max(fftouts);
fftoutdB = 20*log10(fftouts./m);
fb = size(fftoutdB)
sf = size(f)
%plot(fftoutdB(1:(2^B)/2));
plot(f(1:(Np)/4),fftoutdB(1:(Np)/4),'r');
xlabel('Normalized Ouptut Frequency')
ylabel('Magnitude')
legend('No Digital Error (SNR = 128dB)', '11-bit
Digital Truncation (SNR = 77dB)', '4-bit Digital
Truncation (SNR = 16dB)')

figure

Np2 = 2^N
f=(0:1:(Np2)-1)*(1);
fftout = abs(fft(yout));
m = max(fftout);
fftoutdB = 20*log10((fftout)./m);
size(fftoutdB)
size(f)
%plot(fftoutdB(1:(2^B)/2));
%plot(f(1:(Np2)/4),fftoutdB(1:(Np2)/4));
toc

[z,p,k] = ellip(12,.1,135,.015625);
[sos,g]=zp2sos(z,p,k);
h2=dfilt.df2sos(sos,g);

y10 = filter(h2,y8);

for i=1:1:(length(y10)/32)
    y11(i) = y10(i*32);
end
e = size(y11)

y12 = y11(end-2047:end);
w = Hann(length(y12));
size(w)

y13 = y12.*w;
plot(y12)
figure

Np = (2^N)/128;

f=(0:1:(Np)-1)*(1);
fftouta = abs(fft(y13));
m = max(fftouta);
fftoutdB = 20*log10(fftouta./m);
fb = size(fftoutdB)
sf = size(f)
%plot(fftoutdB(1:(2^B)/2));
plot(f(1:(Np)/4).*39.0625,fftoutdB(1:(Np)/4),'b');
xlabel('Normalized Ouptut Frequency')
ylabel('Magnitude')
legend('SNR = 112dB')
% legend('No Digital Error (SNR = 128dB)', '11-bit
Digital Truncation (SNR = 77dB)', '4-bit Digital
Truncation (SNR = 16dB)')
%plot(y12)

bandf = fftouta(1:1024);
xsort = sort(bandf.*bandf,'descend');
xsig=xsort(1);
noise=sum(xsort(100:end));
SNR = 10*log10(xsig/noise)

D. Capacitor Error Shaping Script

%%% Capacitor Error Shaping %%%
%
% This script will show the effect of capacitor
errors and demonstarte
% higher order error shaping

% Specify the modulator NTF, the mismatch-shaping
TF, and the number of
% elements
%ntf = synthesizenTF(3,[],[],4);
M = 4;
sigma_d = 0.01; % 1% mismatch
mtf1 = zpk(1,0,1,1); %First-order shaping
echo off;
A = 1/sqrt(2); % Test tone amplitude, relative
to full-scale.
f = 0.007 ; % Test tone frequency, relative
to fB.
% (Will be adjusted to be an fft
bin)

N = 2^14;
fin = round(f*N);
w = (2*pi/N)*fin;
echo on;
u = M*A*sin(w*[0:N-1]);
v = simulateDSM(u,ntf,M+1); % M unit elements
requires an M+1-level quant.
v = (v+M)/2; % scale v to [0,M]
sv1 = simulateESL(v,mtf1,M);
echo off

figure(1); clf
T = 20;
subplot(211);
plotUsage(thermometer(v(1:T),M));
set(gcf,'NumberTitle','off');
set(gcf,'Name','Element Usage');
title('Thermometer-Coding')
subplot(212);
plotUsage(sv1(:,1:T));
title('First-Order Shaping');
if LiveDemo
    set(1,'position',[9 204 330 525]);
    changeFig(18,.5,1);
    pause
end

ideal = v;

% DAC element values
e_d = randn(M,1);
e_d = e_d - mean(e_d);
e_d = sigma_d * e_d/std(e_d);
ue = 1 + e_d;

% Convert v to analog form, assuming no shaping
thermom = zeros(M+1,1);
for i=1:M
    thermom(i+1) = thermom(i) + ue(i);
end
conventional = thermom(v+1)';

% Convert sv to analog form
dv1 = ue' * sv1;

window = ds_hann(N);
spec = fft(ideal.*window)/(M*N/8);
spec0 = fft(conventional.*window)/(M*N/8);
spec1 = fft(dv1.*window)/(M*N/8);

figure(2); clf
plotSpectrum(spec0,fin,'r');
hold on;
plotSpectrum(spec1,fin,'b');
plotSpectrum(spec,fin,'g');
axis([1e-3 0.5 -200 0]);
x1 = 2e-3; x2=1e-2; y0=-180; dy=dbv(x2/x1);
y3=y0+3*dy;

hold off;
grid;
ylabel('PSD');

```

```

xlabel('Normalized Frequency');
legend('thermometer','rotation','ideal DAC');
fprintf(1, 'Paused.\n');
pause

%Now repeat the above for second-order shaping
mtf2 = zpkl([ 1 1 ], [ 0 0 ], 1, 1); %Second-order
shaping
sv2 = simulateESL(v,mtf2,M);

figure(1); clf
T = 20;
subplot(311);

plotUsage(thermometer(v(1:T),M));
set(gcf, 'NumberTitle', 'off');
set(gcf, 'Name', 'Element Usage');
title('Thermometer-Coding')
subplot(312);
plotUsage(sv1(:,1:T));
title('First-Order Shaping');
subplot(313);
plotUsage(sv2(:,1:T));
title('Second-Order Shaping');
if LiveDemo
    set(1, 'position', [9 204 330 525]);
    changeFig(18, .5, 1);
    pause
end

dv2 = ue' * sv2;
spec2 = fft(dv2.*window)/(M*N/8);

figure(2); clf
plotSpectrum(spec0,fin, 'm');
hold on;

plotSpectrum(spec1,fin, 'r');

plotSpectrum(spec2,fin, 'b');
plotSpectrum(spec,fin, 'k');
axis([1e-3 0.5 -180 0]);
x1 = 2e-3; x2=1e-2; y0=-180; dy=dbv(x2/x1);
y3=y0+3*dy;
%plot([x1 x2 x2 x1],[y0 y0 y3 y0], 'k')
%text(x2, (y0+y3)/2, ' 60 dB/decade')
legend('Thermometer', '1^{st}-Order', '2^{nd}-
Order', 'Ideal');
hold off;
grid;
xlabel('Normalized Frequency');
ylabel('PSD');

```

## X. APPENDIX B: CADENCE SIMULATION NETLIST

```

// Generated for: spectre
// Generated on: Jun 11 10:27:30 2009
// Design library name: Switch_Cap
// Design cell name: Delta_Sigma
// Design view name: schematic
simulator lang=spectre
global 0
include
"/nfs/guille/a2/rh80apps/cadence/IC5141/tools/dfII/s
amples/artist/ahdlLib/quantity.spectre"
parameters vdd=1.8 fin=10k cc=100f

// Library name: Switch_Cap
// Cell name: Opamp_Ideal_true
// View name: schematic
subckt Opamp_Ideal_true vcmo vin vip von vop
R0 (net21 net017) resistor r=1K
C0 (net017 0) capacitor c=1p
E0 (net21 0 vip vin) vcvs gain=5000
E1 (vop vcmo net017 0) vcvs gain=.5
E2 (vcmo von net017 0) vcvs gain=.5

```

```

ends Opamp_Ideal_true
// End of subcircuit definition.

// Library name: Switch_Cap
// Cell name: Switch
// View name: schematic
subckt Switch clk1 clk2 in out
C0 (net49 in) capacitor c=100a
W0 (in out clk1 net49) relay vt1=0.499*vdd
vt2=0.501*vdd ropen=1G \
rclosed=1
V0 (net49 0) vsource dc=0 type=dc
ends Switch
// End of subcircuit definition.

// Library name: Switch_Cap
// Cell name: Delta_Sigma
// View name: schematic
I206 (q4 net0143) amp gain=-1
I207 (q2 net0137) amp gain=-1
I208 (q1 net0134) amp gain=-1
I205 (q3 net0140) amp gain=-1
I199 (net0143 net0140 net0137 net0134 fb_b) adder_4
gain1=1 gain2=1 \
gain3=1 gain4=1
I198 (q4 q3 q2 q1 fb) adder_4 gain1=1 gain2=1
gain3=1 gain4=1
I192 (net086 r3 q3) comparator sigout_high=.45
sigout_low=0 \
comp_slope=10000
I190 (net086 r1 q1) comparator sigout_high=.45
sigout_low=0 \
comp_slope=10000
I191 (net086 r2 q2) comparator sigout_high=.45
sigout_low=0 \
comp_slope=10000
I193 (net086 r4 q4) comparator sigout_high=.45
sigout_low=0 \
comp_slope=10000
I177 (vcm net0113 net0305 x2b x2) Opamp_Ideal_true
I178 (vcm net098 net0217 net0347 net0353)
Opamp_Ideal_true
I176 (vcm net293 net249 x1b x1) Opamp_Ideal_true
I179 (vcm net0108 net0237 net0367 net0361)
Opamp_Ideal_true
I106 (net0169 net0166 net086) subtractor
I143 (p1 p2 net0129 vcm) Switch
I144 (p2 net0135 fb_b net0129) Switch
I108 (p2 net0159 fb net0153) Switch
I107 (p1 p1 net0153 vcm) Switch
I104 (p2 p1 net0169 net0367) Switch
I103 (p2 p1 net0361 net0166) Switch
I99 (p1 p2 net0177 vcm) Switch
I98 (p2 p1 x2b net0177) Switch
I97 (p2 p1 x1b net0181) Switch
I96 (p1 p2 net0181 vcm) Switch
I93 (p2 p1 x1 net0189) Switch
I92 (p1 p2 net0189 vcm) Switch
I73 (p1 p2 net0198 x2b) Switch
I74 (p2 p1 net0221 vcm) Switch
I75 (p1 p2 vcm net0349) Switch
I76 (p2 p1 net098 net0349) Switch
I77 (p1 p2 net0351 vcm) Switch
I78 (p2 p1 net0351 net0217) Switch
I79 (p1 p2 x2 net0221) Switch
I80 (p2 p1 vcm net0198) Switch
I82 (p1 p2 vcm net0258) Switch
I83 (p2 p1 net0353 net0233) Switch
I84 (p2 p1 net0363 net0237) Switch
I85 (p1 p2 net0363 vcm) Switch
I86 (p2 p1 net0108 net0365) Switch
I87 (p1 p2 vcm net0365) Switch
I88 (p1 p2 net0233 vcm) Switch
I89 (p2 p1 net0258 net0347) Switch
I90 (p2 p1 x2 net0261) Switch
I21 (p2 p1 net222 vin_1) Switch
I14 (p1 p2 net245 vcm) Switch
I91 (p1 p2 net0261 vcm) Switch
I13 (p1 p2 vcm net265) Switch
I12 (p2 p1 net293 net265) Switch

```

```

I64 (p1 p2 net0286 x1b) Switch
I65 (p2 p1 net0309 vcm) Switch
I66 (p1 p2 vcm net0357) Switch
I67 (p2 p1 net0113 net0357) Switch
I68 (p1 p2 net0359 vcm) Switch
I69 (p2 p1 net0359 net0305) Switch
I70 (p1 p2 x1 net0309) Switch
I71 (p2 p1 vcm net0286) Switch
I19 (p1 p2 net263 vcm) Switch
I17 (p2 p1 net263 net249) Switch
I15 (p2 p1 vip_1 net245) Switch
I20 (p1 p2 vcm net222) Switch
C34 (net0129 net263) capacitor c=.94*cc
C29 (net0153 net265) capacitor c=.94*cc
C27 (net0177 net0363) capacitor c=2.09*cc
C26 (net0181 net0363) capacitor c=2.1*cc
C24 (net0189 net0365) capacitor c=2.09*cc
C13 (net0217 net0347) capacitor c=cc
C14 (net0221 net0349) capacitor c=.25*cc
C15 (net0198 net0351) capacitor c=.25*cc
C16 (net098 net0353) capacitor c=cc
C10 (net0305 x2b) capacitor c=cc
C11 (net0309 net0357) capacitor c=.76*cc
C12 (net0286 net0359) capacitor c=.76*cc
C17 (net0108 net0361) capacitor c=cc
C19 (net0258 net0363) capacitor c=2.3*cc
C21 (net0233 net0365) capacitor c=2.3*cc
C22 (net0237 net0367) capacitor c=cc
C23 (net0261 net0365) capacitor c=2.1*cc
C0 (net293 x1) capacitor c=cc
C18 (net249 x1b) capacitor c=cc
C6 (net245 net265) capacitor c=.94*cc
C20 (net222 net263) capacitor c=.94*cc
C9 (net0113 x2) capacitor c=cc
V8 (r4 0) vsource dc=-675m type=dc
V7 (vrefn 0) vsource dc=0 type=dc
V6 (vrefp 0) vsource dc=1.8 type=dc
V9 (r3 0) vsource dc=-225m type=dc
V10 (r2 0) vsource dc=225m type=dc
V11 (r1 0) vsource dc=675m type=dc
V1 (vcm 0) vsource dc=900.0m type=dc
V2 (p2 0) vsource type=pulse val0=0 vall=1.8
period=195.3125n \
    delay=97.656n rise=1n fall=1n width=80n
V4 (p1 0) vsource type=pulse val0=0.0 vall=1.8
period=195.3125n rise=1n \
    fall=1n width=80n
V0 (net277 net278) vsource dc=0 type=sine val0=0
vall=1.8 rise=1p fall=1p \
    width=20p freq=fin ampl=600m sinephase=0
sinedc=0 pacmag=1 \
    pacphase=0 mag=1 phase=0 xfmag=1
fundname="Jon"
E1 (vip_1 vcm net277 net278) vcvs gain=.5
E0 (vin_1 vcm net277 net278) vcvs gain=-.5
simulatorOptions options reltol=1e-3 vabstol=1e-6
iabstol=1e-12 temp=27 \
    tnom=27 scalem=1.0 scale=1.0 gmin=1e-12 rforce=1
maxnotes=5 maxwarns=5 \
    digits=5 cols=80 pivrel=1e-3
sensfile="../psf/sens.output" \
    checklimitdest=psf
tran tran stop=200u errpreset=conservative cmin=100a
write="spectre.ic" \
    writefinal="spectre.fc" annotate=status
maxiters=5
finalTimeOP info what=oppoint where=rawfile
modelParameter info what=models where=rawfile
element info what=inst where=rawfile
outputParameter info what=output where=rawfile
designParamVals info what=parameters where=rawfile
primitives info what=primitives where=rawfile
subckts info what=subckts where=rawfile
saveOptions options save=allpub
ahdl_include
"/nfs/guille/a2/rh80apps/cadence/IC5141/tools/dfII/s
amples/artist/ahdlLib/amp/veriloga/veriloga.va"
ahdl_include
"/nfs/guille/a2/rh80apps/cadence/IC5141/tools/dfII/s
amples/artist/ahdlLib/adder_4/veriloga/veriloga.va"

```

```

ahdl_include
"/nfs/guille/a2/rh80apps/cadence/IC5141/tools/dfII/s
amples/artist/ahdlLib/comparator/veriloga/veriloga.v
a"
ahdl_include
"/nfs/guille/a2/rh80apps/cadence/IC5141/tools/dfII/s
amples/artist/ahdlLib/subtractor/veriloga/veriloga.v
a"

```

## XI. AUTHOR



**Jon Guerber** (S'05) received the B.S. degree in Electrical Engineering from Oregon State University in 2008 and is currently working towards a Masters in Electrical Engineering from Oregon State University.

During the Summer of 2008 he was with Teradyne Corp developing high frequency signal tracking and active power management solutions for semiconductor test devices. During the 2007 he was with Intel Corp. investigating high performance, small form factor motherboard architectures to support future PC microprocessor requirements. He is currently a research member of the Analog and Mixed Signal group at Oregon State University in Corvallis, Oregon with a focus in the areas of deep-submicron, low-voltage Analog to Digital Conversion and continuous time filters.

Mr. Guerber is a life member of the Eta Kappa Nu Electrical Engineering Society and an Active Wikipedia Electronics Contributor