

Design of a Second-Order Delta-Sigma Modulator for
Use in Biomedical Signal Acquisition

by Taraka Neelakant Yerra, Bachelor of Science

A Thesis Submitted in Partial
Fulfillment of the Requirements
for the Master of Science Degree

Department of Electrical and Computer Engineering
in the Graduate School
Southern Illinois University Edwardsville
Edwardsville, Illinois

December 2009

ABSTRACT

DESIGN OF A SECOND-ORDER Delta-Sigma MODULATOR FOR USE IN BIOMEDICAL SIGNAL ACQUISITION

by

Yerra Taraka Neelakan

Advisor: Dr. George L Engel

This thesis presents the design and simulation of a small, low-power, second-order, Δ - Σ (delta-sigma) modulator intended for use in multi-channel, portable systems where there is a need for digitizing biomedical signals such as Electro-Cardiogram (ECG), Electroencephalogram (EEG), or Electrocorticography (ECoG) waveforms. In particular, the modulator may, in the future, be fabricated in a 0.5 μm CMOS process and used with a Brain-Controlled Interface (BCI) which uses an ECoG recording modality.

The Δ - Σ would ultimately be combined with a sinc^3 filter along with appropriate amplification and filter circuits, preceding the modulator, to implement a single channel. Eight to sixteen fully-differential channels are expected to be integrated along with a custom DSP core which will produce the 16-bit data streams at a per channel output rate of 250 Samples/sec.

The second-order modulator described herein employs switched-capacitor (SC) technology, operates at a frequency of 256 kHz, uses a 1.2 Volt reference, is implemented as a fully-differential circuit with SC common-mode feedback, and

utilizes correlated double sampling (CDS) in the first integrator stage to manage $1/f$ noise.

A MATLAB® based simulation/analysis tool was written to investigate modulator performance. A strobed comparator and folded-cascode amplifier required by the modulator were also successfully designed. Both VerilogA behavioral and transistor-level simulations were performed using Cadence's Spectre® electrical simulator. All relevant (thermal and $1/f$ noise) time-domain noise sources were included in the MATLAB® simulations as well as in the electrical simulations.

The modulator (in a 125 Hz signal bandwidth) shows a 112.3 dB signal-to-noise and distortion ratio (SNDR) in the presence of a wide variety of circuit non-ideal effects including noise. This corresponds to an ENOB (Effective Number of Bits) of 18.3 bits. The modulator is expected to occupy an area of 0.19 mm² and is suitable for use in a wide variety of low-power, low-cost biomedical instrumentation design applications.

ACKNOWLEDGEMENTS

I would like to thank my mentor Dr. George Engel for his constant support, encouragement, and guidance throughout this project. I would also like to acknowledge the support offered to me by my fellow researchers: Nagachaitanya Yelchuri, Hu Wang, Nam Nguyen, and Kader Hadjsaid.

In addition, I would like to thank Dr. Brad Noble, Dr. Scott Smith and other ECE faculty members at SIUE who taught me many things that helped me with this project. I would be remiss if I did not also acknowledge the help given to me by Mr. Steve Muren who was instrumental in providing the necessary equipment for the lab. Finally, I would like to extend my appreciation to my parents, my brother, cousins and friends whom have supported me all my life.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGEMENTS	iv
LIST OF FIGURES	vii
LIST OF TABLES	ix
Chapter	
1. INTRODUCTION	1
Background.....	1
BCI Recording Technology.....	3
Description of Custom ASIC.....	5
Object and Scope of Thesis.....	11
2. ADC ARCHITECTURE	13
Comparison of ADC Architectures.....	13
Over-sampled ADCs.....	15
Delta-Sigma (Δ - Σ) ADCs.....	16
Delta-Sigma (Δ - Σ) Modulator.....	17
Delta-Sigma (Δ - Σ) ADC for BCI Application.....	20
Second-Order Delta-Sigma (Δ - Σ) Modulator.....	23
MATLAB® Code Description.....	24
3. DESIGN OF SUB-SYSTEMS.....	29
Delta-Sigma (Δ - Σ) Modulator Design.....	29
Integrator.....	32
First Integrator.....	33
Second Integrator.....	34
Comparator	38
Reference Generator	42
Clock Signal Generator	42
4. RESULTS	44
MATLAB® Simulation Results.....	44
Electrical Simulation Results	46
5. SUMMARY/FUTURE WORK	49
Summary	49

Future Work.....	50
REFERENCES	51
APPENDICES	53
A. MATLAB® Code.....	53
B. VerilogA Code.....	69

LIST OF FIGURES

Figure	Page
1.1 ECoG array (64-electrodes) being used with subject.....	4
1.2 Block diagram for single representative channel	6
1.3 CHS pre-amplifier with gain controlled by capacitor ratio.....	8
1.4 Anti-aliasing filter	9
2.1 Frequency spectrum of Nyquist-rate ADC and over-sampling ADC	14
2.2 Block diagram of Delta-Sigma (Δ - Σ) ADC	16
2.3 Block diagram of second-order Δ - Σ modulator with Decimator	17
2.4 Magnitude response of Noise Transfer Function.....	18
2.5 Block diagram of the Δ - Σ ADC for the BCI application.....	23
2.6 Modified architecture of second order Δ - Σ modulator	24
2.7 FFTs at output of each stage in second order Δ - Σ ADC	26
2.8 Time domain waveforms at output of each stage in Δ - Σ ADC	27
3.1 Second order Δ - Σ modulator implementation	30
3.2 Clock diagram for second order Δ - Σ modulator	31
3.3 CDS integrator (first integrator)	34
3.4 Second integrator.....	35
3.5 OTA schematic.....	37
3.6 OTA frequency response.....	37
3.7 Comparator schematic	39
3.8 Latch schematic.....	41
3.9 OTA in unity gain configuration.....	42

3.10	Clock generator schematic	43
4.1	Time domain waveforms	45
4.2	Frequency domain waveforms	45

LIST OF TABLES

Table	Page
2.1 Classification of ADC architectures	14
3.1 Gain and phase margin of OTA for multiple process corners.....	36
3.2 Transistor sizes for OTA schematic	38
3.3 Transistor sizes for comparator schematic.....	40
3.4 Transistor sizes for dynamic latch schematic	41
4.1 Results obtained with VerilogA models	47
4.2 Results obtained with Transistor models	48

CHAPTER 1

INTRODUCTION

Background

The merger of systems neurophysiology and engineering has recently resulted in revolutionary new approaches which link human brain activity and man-made devices in an effort to replace lost sensory and motor function. A Brain-Controlled Interface (BCI) is a device that is capable of capturing brain activity connected to a paralyzed individual's intention to perform an act and then is able to restore communication and/or movement to that immobilized person [Sch:06]. This restoration of communication or movement is accomplished by stimulating actuators that in turn carry out the intended act.

The BCI devices which are currently being developed make use of recordings of electrical activity associated with the scalp, the surface of the brain, or even from within the cerebral cortex. These signals are then translated into command signals that can drive prosthetic limbs, computer displays, *etc.* In short, every BCI has four broad components: (1) recording of neural activity, (2) extraction of the intended action (3) generation of desired action using prosthetic effectors, and (4) feedback either through some intact sensation like vision or produced and applied by the prosthetic device.

As one might imagine, there is a lot of excitement in the field of BCI these days because of the distinct possibility that now exists for actually helping a wide range of patients with neural disorders. Moreover, BCI technology has great potential to make possible additional scientific insight into the way large populations of neurons interact in order to produce specific behaviors. Much

research is currently being performed in the BCI field in both clinical and research settings. The addition of new technology for engineering tissue-electrode interfaces, *the development of highly integrated low-power electronics which can be used for recording neural signals*, and the development of improved extraction algorithms to transform the recorded signals to movement will help transform, what to date, have been exciting laboratory demonstrations into patient practice in the near future!

Dr. Daniel Moran, Ph.D. a Washington University in Saint Louis (WUSTL) assistant professor of biomedical engineering in the WUSTL School of Engineering and Applied Science is a nationally known researcher working in the area of BCIs [Fit:04]. Dr. Robert E. Morley, associate professor of electrical and systems engineering in the WUSTL School of Engineering and Applied Science and Dr. George L. Engel, professor of electrical and computer engineering in the Southern Illinois University Edwardsville (SIUE) School of Engineering are considering entering into a collaboration with Dr. Moran in an attempt to advance BCI recording technology. This Thesis work is an early attempt to determine what types of advancements might be feasible. The hope is that in the future the work can serve as a springboard for a viable proposal to the National Institute of Health (NIH) to support continued work.

Currently, discrete electronics, which are both bulky and power hungry, are being used to record electrical signals associated with brain activity. The collaboration looks to develop a very small-area, low-power, inexpensive, multi-channel custom integrated circuit (IC) that can be used for recoding the signals from a large array of electrodes. This Application Specific Integrated Circuit (ASIC) would be used to convert the analog signals associated with brain activity into a digital format. The custom IC would then be paired with a commercially available

chip capable of wireless transmission of the digitized signals to a host computer where they can be further processed.

In the next two sections of this thesis, we will (1) review the current recording technology in use today in an attempt to determine appropriate specifications for the proposed ASIC and (2) describe a custom ASIC that could be used for biomedical signal acquisition in BCI or similar applications. In the final section of this chapter we will define the object and scope of material to be addressed in the remainder of the thesis; namely, the design of a critical circuit (a second-order delta sigma modulator) needed for digitization of the electrical signals associated with brain activity.

BCI Recording Technology

The first step in the BCI process is to capture signals containing information about the subject's intended movement. Currently, the four primary recording modalities are (1) electro-encephalography (EEG), (2) electro-corticography (ECoG), (3) local field potentials (LFPs), and (4) single-neuron action potential recordings (Single Units) [Sch:06]. All four of these techniques require recording micro-volt level extra-cellular potentials generated by neurons in the brain's cortical layers. The modalities given above are listed in order of least invasive to most invasive.

Because EEG is non-invasive, it has dominated BCI research but EEG recordings can only resolve low spatial frequencies. Higher spatial as well as temporal resolution results in more useful data and better control. Generally the more invasive techniques yield both higher spatial and temporal resolution but are considered less safe. An electrical bandwidth of approximately 30-50 Hz is sufficient for recording EEG signals.

The advantage of ECoG-based systems is that recording electrodes are approximated on the cortical surface, yielding a much finer spatial resolution (on the order of mm). An electrical bandwidth of 100 – 200 Hz is sufficient for ECoG recordings. While ECoG systems are invasive, it is believed that because they are on the brain surface that they are more robust than penetrating electrodes (Single Units or LFPs). The electrical bandwidth required by Single Units or LFPs is on the order of 5 kHz.

Research conducted by Dr. Moran primarily makes use of ECoG-based systems. A 64-element electrode array (Figure 1.1a) is shown attached to the surface of the brain in Figure 1.1 (C). The pictures in Figure 1.1 were taken from Dr. Moran's website [Mor:09].

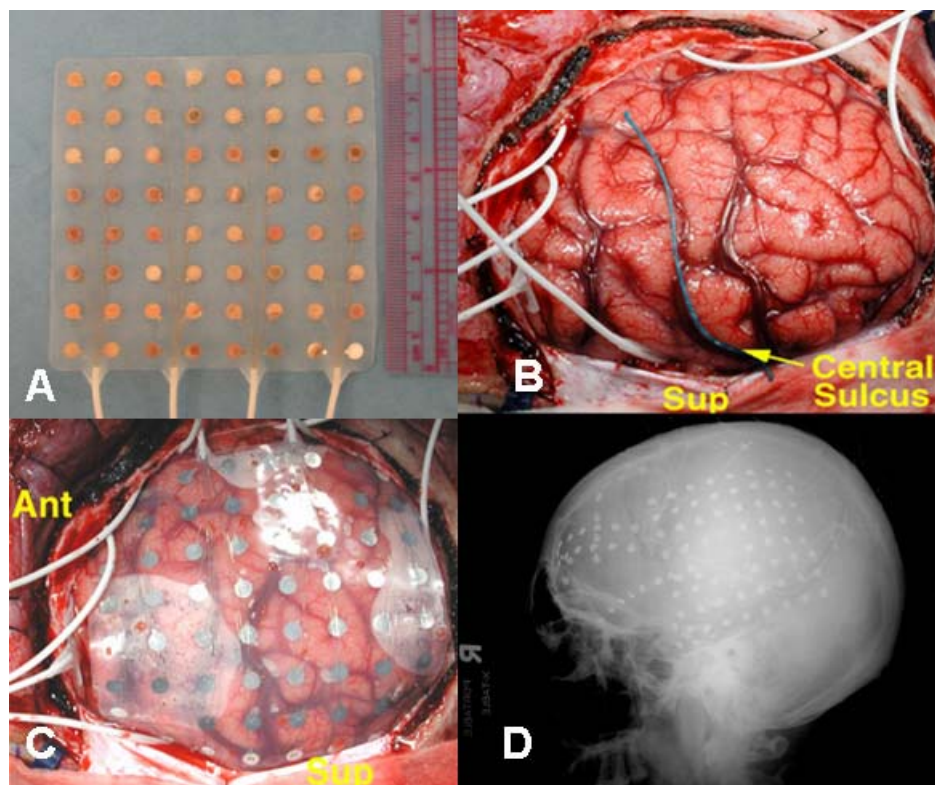


Figure 1.1: ECoG array (64-electrodes) being used with subject

In short, the ECoG modality provides a nice compromise between safety and performance. While the use of LPFs and Single Units may offer superior performance because of higher spatial and temporal resolution, the use of these modalities is less common and is generally considered less safe for the subject. It is for these reasons that proposed ASIC to be described in the following section of this thesis is targeted at BCIs that employ either EEG or ECoG recordings [Sch:06].

A decision was made to support high resolution recordings with an electrical bandwidth of 125 Hz. With a desired bandwidth of 125 Hz it is necessary for the analog waveforms to be digitized at a rate of at least 250samples/sec in order to satisfy the Nyquist Sampling Theorem. The lower corner frequency can be set at around 1 Hz. Because of this relatively small bandwidth requirement, the proposed ASIC can be very low-power and small in size. We feel that a 64-channel ASIC capable of servicing the 64-element electrode array pictured in Figure 1.1 is feasible. In the following section we describe the design of such a custom ASIC.

Description of Custom ASIC

While the target technology assumed throughout this thesis is the AMIS (American MicroSystems Inc. Semiconductor) 0.5 μm NWELL process (C5N), any similar CMOS process should yield similar results. The key features of the process that make it a good choice for the proposed application include the ability to realize high-density resistors and capacitors. The AMIS C5N process supports both double poly capacitors (1 fF/ μm^2) and compact resistors implemented using a high-resistance (1 k Ω /square doped second-level polysilicon layer). The process also affords the designer 3 metal layers.

CMOS technology was chosen over a bipolar process because CMOS technologies allow dense digital circuits to be integrated on the same die as the

desired analog circuitry. It is also much less expensive to fabricate CMOS circuits. Moreover, high resolution analog-to-digital conversion is best achieved using CMOS. Unfortunately, relative to bipolar devices, MOS devices exhibit excessive $1/f$ (*i.e.* flicker) noise characteristics. This is unfortunate given the low-frequency signal bandwidth of the proposed design. As we will demonstrate, by using appropriate circuit techniques any inherent limitations resulting from the choice of a CMOS technology can be effectively overcome.

Figure 1.2 presents a block diagram for a single representative channel able to process signals associated with brain activity. The proposed ASIC would consist of many (8 or 16) identical signal processing channels of the type illustrated in Figure 1.2 along with various other support circuits (bias circuits, a single digital signal processor (DSP) which would support all channels, and a switch matrix). The switch matrix allows pairs (8 or 16 pairs depending on the number of implemented channels) of electrodes to be selected from the 64-element electrode array.

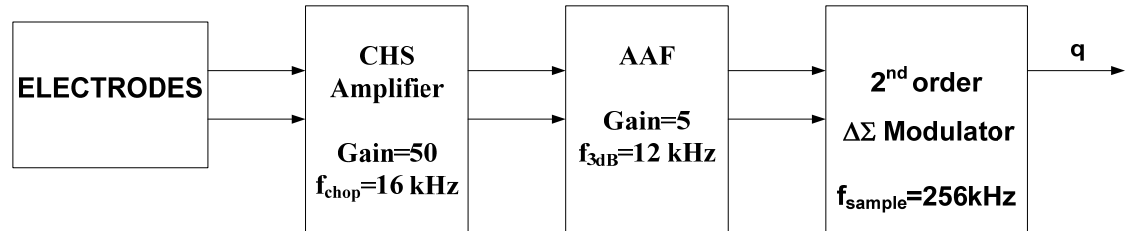


Figure 1.2: Block diagram for single representative channel

In Figure 1.2, the differential signal associated with two adjacent electrodes is first amplified by a chopper-stabilized (CHS) pre-amplifier. The purpose of the pre-amplifier is to provide gain without introducing significant noise. Chopper stabilization is used to remove $1/f$ noise associated with the MOS input devices of

the pre-amplifier that might deteriorate the signal-to-noise ratio (SNR). After a thorough study, we concluded that a chopping frequency of 16 kHz was optimal.

As explained in the introduction, the very poor $1/f$ (flicker) noise characteristics of the FETs in a CMOS process (especially the NFETs when an NWELL process is used) make the design of the pre-amplifier challenging. The technique most effective in suppressing low-frequency noise in this application is chopper stabilization (CHS) [Tem:98]. The CHS technique applies modulation to transpose the signal to a higher frequency where there is less $1/f$ noise and then demodulates it back to baseband after amplification.

For simplicity, the modulation is performed using a square-wave carrier. After modulation, the signal is transposed to odd harmonic frequencies of the modulation signal. After amplification, the signal is demodulated using the same square-wave carrier that was used for the initial modulation. While the input signal is modulated, amplified, and then demodulated, the input-referred noise and offset voltages associated with the amplifier are modulated only once.

The gain of the CHS pre-amplifier is chosen to be approximately 50 although the effective gain is 35 (or 31 dB). This reduction in gain is due to the chopping process and the finite bandwidth of the core amplifier used in the pre-amplifier. The bandwidth of the core amplifier was selected to be twice the chopper frequency in order to minimize clock feed-through/injection effects associated with the chopper switches and so that the effective gain in the passband would differ by 3 dB or less from the desired gain due to finite bandwidth effects.

After a thorough literature search, an appropriate circuit topology for the CHS pre-amplifier was decided upon. The circuit tentatively chosen for integration is illustrated in Figure 1.3. Each MOD block is a simple set of 4 CMOS switches which alternate the polarity each half cycle of the 16 kHz square-wave chopper

clock. We plan to realize the pseudo-resistor [Har:03] by using two small PFETs in series in separate wells with the source of each PFET connected to its local NWELL. For small signals, the dynamic resistance is expected to be near $100\text{ G}\Omega$. The use of the proposed pseudo-resistor not only allows us to implement very large dynamic resistances, but if the voltage would change suddenly, the resulting transient would be relatively short-lived (*i.e.* much shorter than one would expect given a lower corner frequency of 1 Hz).

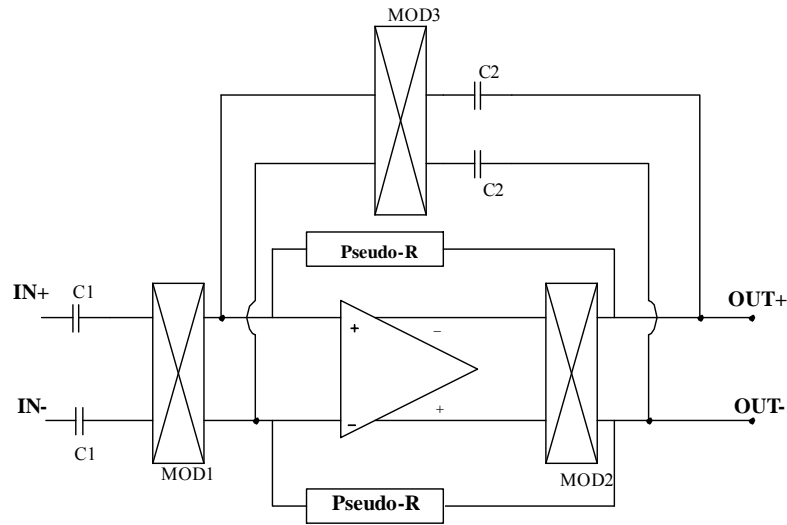


Figure 1.3: CHS pre-amplifier with gain controlled by capacitor ratio

The reason for this is that the dynamic resistance of the structure falls rapidly as the input level rises because, depending on the polarity, each PFET acts either as a diode-connected bipolar junction (lateral PNP parasitic) transistor (BJT) or a field effect transistor (FET). In either case, the device will allow current to flow easily once the magnitude of the voltage approaches several hundred mV. When the voltage across the structure is small, both devices are essentially in the off-state (except for leakage and sub-threshold currents) yielding an extremely large dynamic resistance.

The CHS amplifier is followed by a second-order, low-pass, anti-aliasing filter (AAF) with a corner frequency at perhaps 12 or 24 kHz. The second-order low-pass filter, illustrated in Figure 1.4, provides anti-aliasing. The circuit was selected over a Sallen-Key structure because it places fewer demands on the input common-mode range of the core amplifier and because the passband gain is determined by the ratio of two capacitors as was the case for the pre-amplifier.

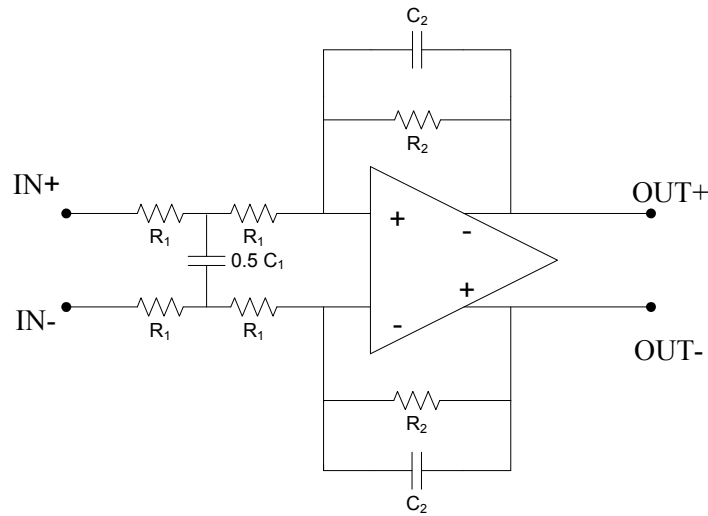


Figure 1.4: Anti-aliasing filter

The AAF is needed specifically to attenuate the spectral artifacts created as a result of the chopper stabilization used in the pre-amplifier design prior to digitization in order to avoid aliasing. In order to increase the signal level, a gain of 5 (or 14 dB) is embedded in the AAF filter. The overall gain is approximately 175 (or 45 dB).

The differential output of the AAF is then applied to a 2nd order delta-sigma modulator operating at a sampling rate of 256 kHz. The modulator will employ a 1.2 V reference. *It is the design of the 2nd order delta-sigma modulator which is the*

focus of this thesis. The 1-bit output from the modulator is labeled, q , in Figure 1.2.

The 2nd order modulator is needed because the most appropriate type of analog-to-digital converter (ADC) for use in applications requiring low output sample rates but very high resolution are delta-sigma over-sampling converters. This style of converter trades off accuracy in time for accuracy in amplitude and makes use of noise shaping [All:03]. The one-bit data stream, q , can be digitally processed to yield a much higher resolution digital data stream but at a significantly lower data rate. The guiding design equation for a delta-sigma converter is given below

$$\text{DR} := \sqrt{\frac{3}{2} \cdot \frac{\text{OSR}^{2 \cdot L + 1}}{\pi^{2 \cdot L}} \cdot (2 \cdot L + 1)} \quad \text{Eqn. 1.1}$$

Where DR is the achievable dynamic range of the converter, OSR is the oversampling ratio, and L is the order of the modulator. In the proposed design L is 2 (second-order) and the OSR is the ratio of 256 Ksamples/sec to 250 Samples/sec *i.e.* 1024. Once the achievable dynamic range is computed, then the effective number of bits (ENOB) for resolution purposes may be calculated using the equation

$$\text{ENOB} := \frac{\log\left(\frac{\text{DR}}{\sqrt{3}}\right)}{\log(2)} + 0.5 \quad \text{Eqn. 1.2}$$

The one-bit data stream from each of the channels could be serviced by a single digital signal processor (DSP). A single processor is capable of handling all

of the digital traffic associated with the multiple signal processing channels because of the low sampling rate of 256 K-Samples/sec employed in this design. Moreover, recall that the eventual output rate (per channel) from the DSP will be 250samples/sec. This output sampling rate will allow us to support a bandwidth of 125 Hz. Use of Eqn. 1.1 and Eqn. 1.2 suggest that > 20 bits is theoretically achievable but as we shall see in later chapters of this thesis, the analog noise sources and other circuit non-idealities will allow us to realize only 17.5 bits.

Our goal was to ensure a signal-to-noise ratio of 15 – 20 dB for the lowest amplitude signals (1 μ V peak-peak). Using our simple model for the brain activity signal (a 20 Hz sinewave), this peak-peak value corresponds to an RMS level of 0.35 μ V. In later chapters of this thesis we will demonstrate that this goal can be met with a 17-bit converter and the gains selected for the CHS pre-amplifier and AAF.

Object and Scope of Thesis

The object of this thesis is to describe the design and simulation (both at the behavioral and at the electrical level) of a small, low-power second-order delta-sigma modulator intended for use in multi-channel, portable systems where there is a need for digitizing biomedical signals such as electro-cardiogram (ECG), electro-encephalogram (EEG), or electro-corticography (ECoG) waveforms. In particular, the design described herein may someday be used in a Brain-Controlled Interface (BCI) which employs an ECoG modality.

There are five chapters in this thesis. A discussion of the rising importance of BCIs, a description of various recording modalities, and a description of a proposed custom multi-channel ASIC which can be used to acquire EEG and ECoG waveforms is presented here in Chapter 1. In Chapter 2, the operating principles of

delta sigma ADCs are presented and system specifications are developed. The electrical level design of the modulator is described in Chapter 3. Both results from MATLAB® and the results obtained using Cadence's Spectre® program are presented in Chapter 4. The results demonstrate that the modulator performance meets all of the specifications established in Chapter 2. Finally, conclusions are drawn and items requiring additional work are identified in Chapter 5.

CHAPTER 2

ADC ARCHITECTURE

Comparison of ADC Architectures

Based on the sampling rate of the input analog signal, Analog-to-Digital Converters (ADC) can be classified into two main categories: 1) Nyquist-rate analog-to-digital converters 2) Over-sampling analog-to-digital converters. According to the Nyquist sampling theorem, the sampling frequency f_s must be at least twice the highest frequency of the input signal, f_B , in order to recover the signal from its samples. Nyquist-rate ADCs operate with an input signal frequency close to half the sampling frequency and require a very sharp cutoff for the preamplifier or anti-aliasing filter, making it difficult to implement [All:03].

Over-sampling ADCs operate with an input signal frequency much less than half the sampling frequency. Since the sampling frequency is much higher than the Nyquist rate, the anti-aliasing filter requirements are less stringent than those of Nyquist-rate converters as can be seen by looking at the comparison presented in Figure 2.1 [All:03].

Within the two major classes of converters, there are many sub-classifications. Table 2.1 (taken from [All:03]) presents the classification of various types of ADCs along with their resolution and conversion rates. Though Nyquist ADC's are very fast, their resolution is limited by component matching and circuit non-idealities.

Over-sampling ADCs are based on trading off resolution in time for increased resolution in amplitude [Bos:88]. Over-sampling converters are very popular because they do not require stringent matching and small offsets. Their sampling rate usually needs to be several orders of magnitude higher than the

Nyquist rate, so over-sampling method are best suited for relatively low frequency signals; for example, EEG and ECoG signals!

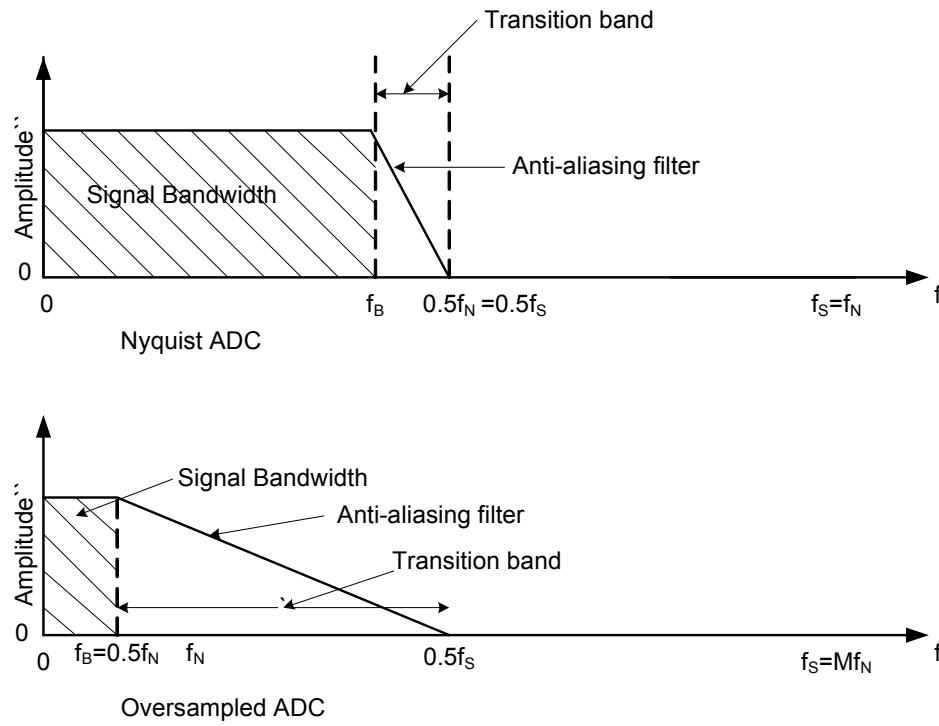


Figure 2.1: Frequency spectrum of Nyquist-rate ADC and over-sampling ADC

Conversion rate	Nyquist-rate ADCs	Over-sampled ADCs
Slow (10-100 Samples/sec)	Serial (ramp, dual-ramp). High resolution possible. (< 14 bits possible)	Very high resolution possible (< 24 bits possible)
Medium (1-100 KSamples/sec)	Successive approximation Algorithmic (< 16 bits possible)	Moderate resolution (< 18 bits possible)
Fast (1 – 100 MS/sec)	Flash Multiple-bit pipeline Folding and interpolating (< 14 bits possible)	Low resolution (< 6 bits possible)

Table 2.1: Classification of ADC architectures

Over-sampled ADCs

Over-sampled ADCs are sometimes referred to as “noise-shaping” ADCs but not all over-sampled ADCs actually employ noise-shaping. The key techniques employed in these ADCs are oversampling and noise shaping. In over-sampled ADCs, the sampling is done at much higher rate than the Nyquist rate ($2 * f_B$). The ratio of sampling rate in over-sampled ADCs to the Nyquist rate is called over-sampling ratio, M . Over-sampling converters take advantage of today’s VLSI technology tailored toward providing high-speed/high-density digital circuits rather than accurate analog circuits by performing most of the conversion process in the digital domain [Can:97]. The requirements for a Nyquist rate ADC anti-aliasing filter is much more stringent when compared to an over-sampled AD, but an over-sampling ADC requires a relatively fast and complex signal processing module.

Over-sampled ADCs are classified into three major categories. They are straight oversampling ADCs, predictive (delta modulation) and noise shaping (delta-sigma modulation) ADCs. Straight over-sampling does not use the concept of noise shaping where as predictive and noise shaping ADCs do. In straight over-sampling ADCs it is assumed that the quantization noise is equally distributed white noise from DC to one half the sampling rate. The out-of-band noise is removed by using a digital filter. In straight over-sampling ADCs, doubling the over-sampling ratio increases the resolution by 0.5 bits.

In predictive over-sampling ADCs, a loop filter is placed in the feedback path which shapes both quantization noise and signal, where as in noise shaping ADCs the loop filter is placed in the feed-forward path which shapes (high-pass) the quantization noise only, preserving the signal. A noise-shaping ADC employing a coarse quantizer is called Delta-Sigma (Δ - Σ) ADC. Among all ADCs discussed above

the Δ - Σ ADC is the most widely used over-sampling ADC because it is much less sensitive to circuit imperfections.

Delta-Sigma (Δ - Σ) ADCs

A delta-sigma ADC consists of an analog block and a digital block as shown in the Figure 2.2.

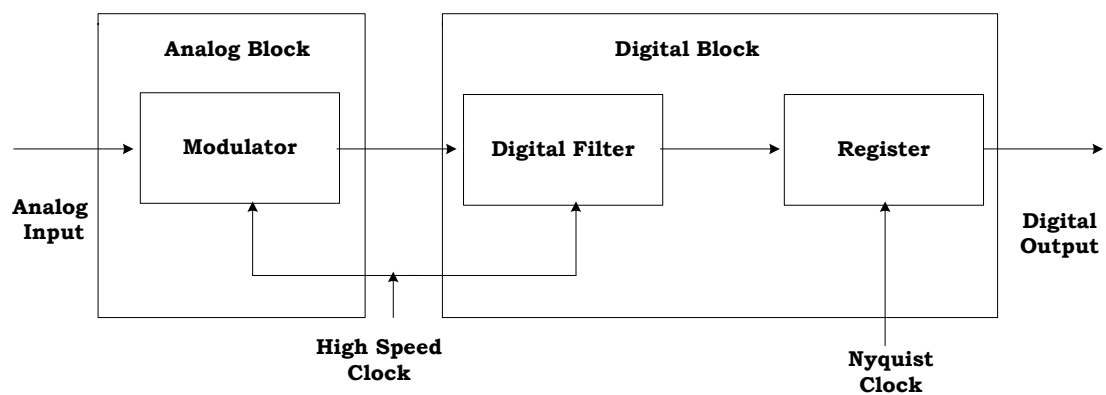


Figure 2.2: Block diagram of Delta-Sigma ADC

The analog block consists of a modulator, and the digital block consists of digital decimation filters which in many cases occupies most of the die area and consumes more power than the analog part. The Δ - Σ modulator consists of an analog filter followed by a coarse quantizer (typically a simple analog comparator) in a feedback loop. Together with the filter, the feedback loop acts to attenuate the quantization noise at low frequencies while emphasizing the high frequency noise.

The block diagram of delta-sigma modulator is shown in Figure 2.3 [All:03]. The decimation filters consists of a low-pass filter and a down sampler. The decimation filters following the analog modulator smoothes the output of the modulator, attenuating noise, interference, and high frequency components of the

signal before they can alias into the signal band when the output is sampled at the Nyquist rate. These filters decimate the output of the modulator by the oversampling ratio (M) to produce digital samples at the Nyquist rate. As a part of this thesis, we designed the second order Δ - Σ modulator which is the analog portion of the Δ - Σ ADC.

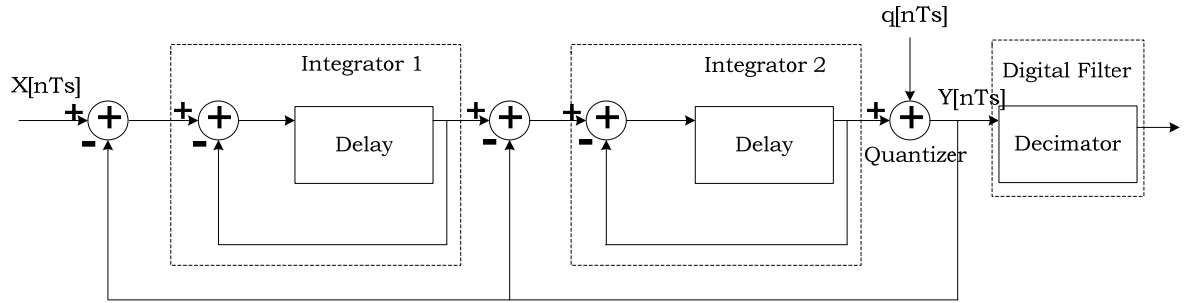


Figure 2.3: Block diagram of second-order Δ - Σ modulator with decimator

Delta-Sigma (Δ - Σ) Modulator

A Δ - Σ Modulator is the only analog circuit in a delta-sigma ADC. A classical Δ - Σ modulator consists of integrators (analog filters) followed by a comparator (coarse quantizer). The order of the Δ - Σ modulator is given by the number of integrators in the forward path of the modulator. An L^{th} order Δ - Σ modulator will have L integrators in the forward path and its output is given by Eqn 2.1.

$$Y(z) = X(z) \cdot z^{-1} + Q(z) \cdot (1 - z^{-1})^L \quad \text{Eqn. 2.1}$$

where $X(z)$, $Y(z)$ and $Q(z)$ are the z-transforms of the modulator input, output and the quantization error respectively. The multiplication factor of $X(z)$ is called the signal transfer function (STF), whereas of $Q(z)$ is called the noise transfer function

(NTF). In the above expression z^{-1} represents a unit delay and $(1-z^{-1})$ represents a high pass characteristic allowing noise suppression at low frequencies [All:03]. Therefore, the output of a Δ - Σ modulator is always the sum of the input signal (delayed by one clock period) and the high-passed quantization noise. The high frequency quantization noise is removed by using the digital filters with a “brick-wall” low-pass response.

For a Δ - Σ modulator of L^{th} order, the NTF will be an L^{th} order high-pass filter which results in L zeros at DC resulting in L^{th} -order noise-shaping at low frequencies. The key point in the operation of Δ - Σ modulators is that most of the quantization noise is pushed out of the band of interest to the user. In Figure 2.4, the magnitude of the NTF is plotted as a function of frequency for different modulator orders along with the ideal magnitude response of the digital decimation filter. From the Figure 2.4, it is clear that as the modulator order increases, the portion of quantization error that falls into the signal band decreases.

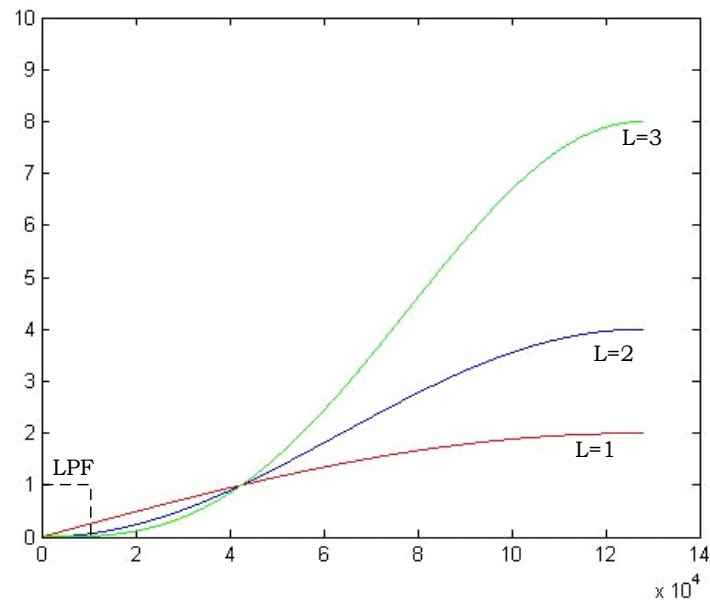


Figure 2.4: Magnitude response of Noise Transfer Function

Unlike Nyquist rate converters whose performance metrics are integral non-linearity and differential non-linearity, the performance measures for Δ - Σ ADCs are Quantizing Signal-to-Noise Ratio (QSNR) and Dynamic Range (DR). QSNR is defined as the ratio of the signal power to baseband noise power, and DR is defined as the ratio of the power of a full-scale sinusoidal input to the power of the sinusoidal input for which SNR is one [All:03]. The expression for DR in terms of oversampling ratio (M), number of bits of quantizer (B,) and order of the modulator (L) is given by Eqn. 2.2.

$$DR = \frac{3}{2} \cdot \frac{2 \cdot L + 1}{\pi^{2L}} \cdot M^{2L+1} \cdot (2^B - 1) \quad \text{Eqn. 2.2}$$

From the above equation, the over-sampling ratio, M, is given by

$$M = \left(\frac{2}{3} \cdot \frac{DR^2}{2L + 1} \cdot \frac{\pi^{2L}}{2^B - 1} \right)^{\frac{1}{(2L+1)}} \quad \text{Eqn. 2.3}$$

The DR for an ADC in terms of effective number of bits (N) of ADC is given by Eqn. 2.4 and Eqn. 2.5.

$$DR = \sqrt{3 \cdot 2^{2N-1}} \quad \text{Eqn. 2.4}$$

$$DR_{dB} = 6.02 N - 1.76 \quad \text{Eqn. 2.5}$$

The effective number of bits for a Δ - Σ modulator is given by

$$\text{ENOB} = \frac{\text{DR}_{\text{dB}} - 1.76}{6.02} \quad \text{Eqn. 2.6}$$

With a Δ - Σ ADC, by doubling of the sampling rate we can achieve an increase in dynamic range of $3(2L+1)$ dB, providing $L+0.5$ extra bits [Can:00]. Therefore, for a second-order modulator (like the one proposed in this thesis for use in a BCI application) an increase of 2.5 bits per octave of over-sampling is achievable.

Delta-Sigma (Δ - Σ) ADC for BCI Application

As discussed in Chapter 1, the signal bandwidth for the intended applications does not exceed 125Hz. Moreover, our design goal was to ensure at least a 20 dB signal-to-noise ratio (SNR) for the lowest amplitude signals (1 μ V peak-peak). Using our simple model for the brain activity signal (a 20 Hz sinewave), this peak-peak value corresponds to an RMS level of 0.35 μ V. To meet our goal of achieving a 15 - 20 dB SNR, the total integrated (input-referred) noise at the input to the channel must be less than 35 nV.

Recall from Chapter 1, that the electrode signal is first amplified by a chopper stabilized amplifier (CHS) with an effective gain of 35 (31 dB) followed by an anti-aliasing filter (AAF) with a gain of 5 (14 dB). The overall gain is approximately 175 (or 45 dB). Therefore, the integrated noise at the input of the ADC would be approximately 6 μ V. Consequently, we required that the quantization noise of the ADC should be less than 6 μ V. It is well-known that the total integrated noise of a linear converter, ε , with a step-size, δ , is

$$\varepsilon = \delta / \sqrt{12} \quad \text{Eqn. 2.7}$$

We conclude that the ADC must have a quantization step, δ , which is less than 20 μV . If one chooses the ADC's reference voltage, V_R , as 1.2 Volts (a bandgap voltage), then one can compute, δ , by using Eqn. 2.8

$$\delta = 2 V_R / 2^N \quad \text{Eqn. 2.8}$$

Where we have assumed that the range of the converter is analog signal ground plus or minus V_R , and N is the effective number of bits (ENOB) of the Δ - Σ ADC. For N equal to 17 bits, the converter step-size, δ , is 18 μV , satisfying our desire for it to be less than 20 μV . Therefore, we must design a Δ - Σ ADC with an ENOB of at least 17 bits. As we have already stated, 3 – 4 bits are needed to satisfy our design for a SNR of 15 – 20 dB even with the smallest input signals applied to the input of the ADC. Another 3 – 4 bits account for variability in signal level. This leaves 9 – 11 bits of additional dynamic range.

While fewer bits would be necessary if the gain of the CHS and/or the AAF were significantly increased, we decided against this option. Recall from our discussion in Chapter 1 that the gain of both of these blocks is determined by the ratio of capacitors. As it stands, these ratios are already relatively large (50 for the CHS and 5 for the AAF). Implementing large capacitor ratios entails large die area which is contrary to our need for a small, inexpensive die and our need to realize many (8 or 16) channels.

Furthermore, especially in the case of EEG recordings, the aforementioned 9 – 11 bits of dynamic range are extremely useful. When a subject blinks, a step or impulse input is superimposed with the EEG waveform. Given the lower corner frequency of 1 Hz for the CHS circuit, it can take a relatively long time for the

signal to return to baseline. Along with the CHS's dynamic DC baseline restorer circuit (discussed in Chapter 1), the 9 – 11 bits of dynamic range (which the 17-bit ADC affords us), makes certain that the signals associated with brain activity will never drop out.

In order to achieve 17 bits of resolution and a signal bandwidth of 125 Hz, we selected a sampling frequency of 256 kHz. This gives an over-sampling ratio of $256 \text{ kHz} / 250 \text{ Hz}$ or 1024. A factor of 1024 represents 10 octaves. In the previous section, we showed that for a second-order modulator one obtains 2.5 bits/octave. Hence, one might expect 22.5 bits. One might conclude that a sampling frequency of 64 kHz would suffice, but the analysis used to predict 22.5 bits does not account for circuit non-idealities (for example, analog noise sources). We will demonstrate later in this thesis that selecting the 256 kHz sampling frequency made it easier to achieve an ENOB of 17 bits in the presence of analog noise and made it easier to obtain a small modulator (reduced die area)s.

A block diagram of the proposed 17 bit Δ - Σ ADC is presented in Figure 2.5. The first stage of the decimation filter is a sinc^3 filter which can be implemented very efficiently using a cascade of comb filters. The optimal order of the sinc filter for a Δ - Σ ADC which includes an L^{th} order modulator is $L+1$. Therefore a sinc filter of order 3 is needed in implementing this ADC. The output of the sinc filter is decimated by a factor of 256. This output is filtered by two decimation filters. The use of a sinc filter followed by a pair of decimation filters (each decimating by a factor of 2) has been proven to be the most efficient method of producing the final digital output [Can:00]. Most of the Δ - Σ ADC applications demand decimation filters with linear phase characteristics. Therefore, we used symmetric FIR filters.

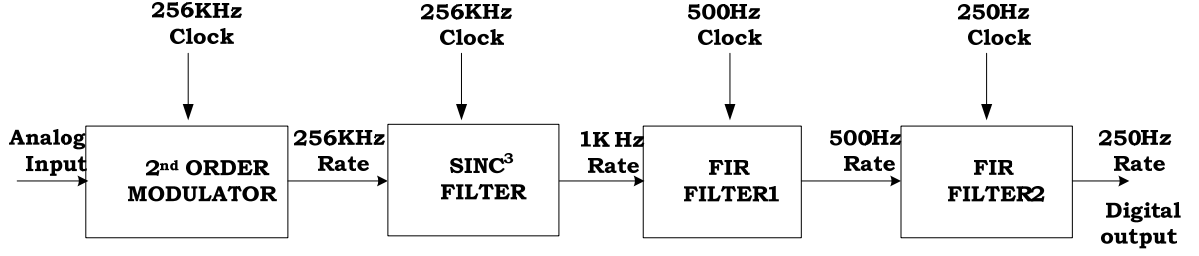


Figure 2.5: Block diagram of the Δ - Σ ADC for the BCI application

Second-Order Delta-Sigma (Δ - Σ) modulator

The focus of this thesis is the design of a second-order Δ - Σ modulator for the Δ - Σ ADC. In order to validate the findings from the previous section, system level simulations were performed in MATLAB®. The next section in this thesis describes the functions that were written in MATLAB®, the simulation results, and the specifications that were arrived at based on those results.

A block diagram for a second-order Δ - Σ modulator is presented in Figure 2.6 [Bos:88]. The important building block of a Δ - Σ modulator is an integrator. The integrator was implemented using a switched-capacitor (SC) block utilizing an Operational Transconductance Amplifiers (OTA). The specifications for these amplifiers had to be determined in order for us to be able to design the amplifier and hence the integrators. The 1-bit quantizer is implemented using a strobed comparator. The ADC is designed with an input range of -1.2 Volts to 1.2 Volts relative to the analog signal ground, which we call AGND. The AGND voltage in our design is 2.4 Volts. Therefore, the effective input range of the ADC is from 1.2 Volts to 3.6 Volts.

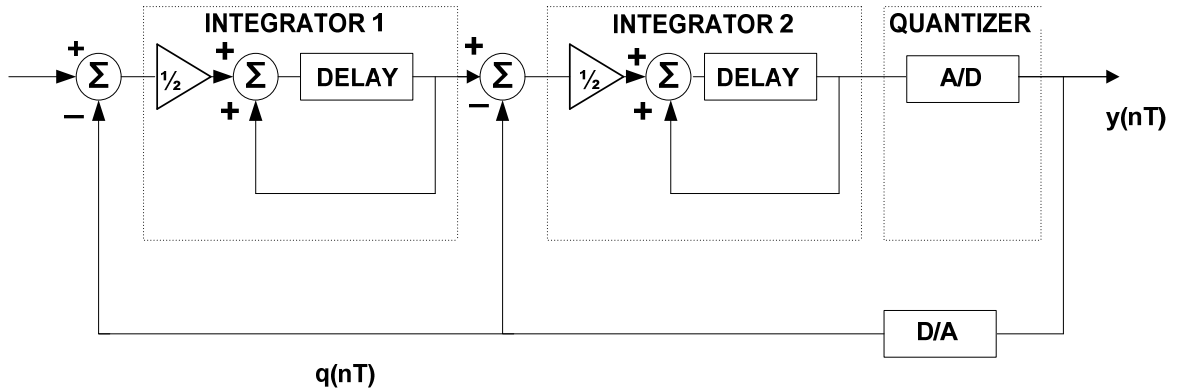


Figure 2.6: Modified architecture of second order Δ - Σ modulator

MATLAB® Code Description

A second-order delta-sigma analysis tool was developed using MATLAB®. A listing of all of the MATLAB® code which was developed as part of this thesis work is included in Appendix A. The function “DeltaSigmaTool” is a tool used to analyze the performance of second-order Δ - Σ modulators. It can be used to determine the ENOB values for a modulator and can calculate the theoretically achievable value. In addition to being used for the analysis of a 1-bit modulator output stream, the tool contains a model of a second-order delta-sigma model with many important non-ideal effects included.

The frequency domain and time domain responses at the model’s modulator output, sinc filter output, and decimation filter outputs can be viewed. The function “DeltaSigmaTool” calls many other functions: SineWaveGenerator (generates a sinewave with user specified frequency and peak amplitude), SecondOrderModulator (model with many non-ideal effects), SincCubedFilter (implements the sinc^3 filter and first decimation), DecimationFilter (designs and simulates Filter 1 or Filter 2 illustrated in Figure 2.5).

“DeltaSigmaTool” has the capability of reading the 1-bit output bit stream obtained from Spectre® (or some other electrical simulator) and then can process

the 1-bit data stream using the sinc and decimation filters. The results at the various steps in this process are presented in both frequency domain and time domain plots. This function takes the middle group of samples at each stage output to eliminate the transients in the waveforms.

The function “SineWaveGenerator” takes as arguments the number of samples, amplitude of the sine wave, sampling frequency and frequency of the sine wave required. It generates the number of samples required in the form an array which is passed to the function implementing the second order modulator.

The function “SecondOrderModulator” is a behavioral description of a second order Δ - Σ modulator. This function takes as arguments the reference voltage, the integrator capacitance, and the number of samples of sine wave. The function simulates the effect of the integrated noise associated with switched capacitors and noise on the reference. The function takes the samples from the sine wave generator and generates the output as a series of 1’s and 0’s which represent the Pulse Density Modulated (PDM) output of a real Δ - Σ modulator. An example frequency spectral plot and time domain response at the output of the modulator is shown in Figures 2.7 and 2.8 respectively. The noise shaping is clearly evident.

The function “SincCubedFilter” implements the sinc^3 filter. This is the first of three digital stages. It takes as arguments the bit stream produced by the modulator and the decimation factor. A sinc^3 filter consists of three integrators, followed by the decimator, followed by three differentiators. The 1-bit data stream from the modulator is converted into a binary value (either +K or -K) which is ‘n’ bits wide, depending on whether the input is high or low. The value of ‘n’ is given by the Eqn 2.9 and is the required width of the registers used to implement the integrators and differentiators that make up the sinc^3 filter.

$$n = \frac{A \cdot \log(D)}{\log(2)} + 1 \quad \text{Eqn. 2.9}$$

D is the decimation factor for the sinc filter and A is order of the sinc filter (3 in this application).

The function “DecimationFilter” implements a FIR filter by using the function “HalfbandFilter” which uses the in-built MATLAB® function “remez” for generating the FIR filter coefficients. The decimation filter decimates the input by a factor of 2. Two decimation filters in cascade are used to implement the final stage of digital filtering.

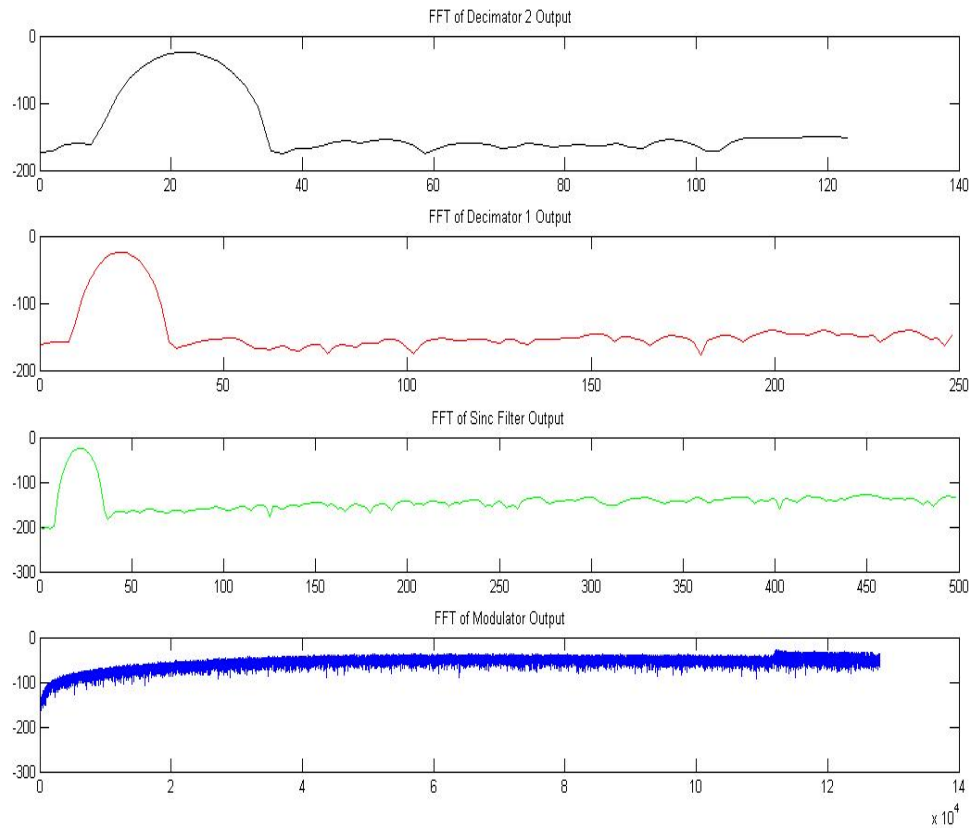


Figure 2.7: FFTs at output of each stage in Δ - Σ ADC.

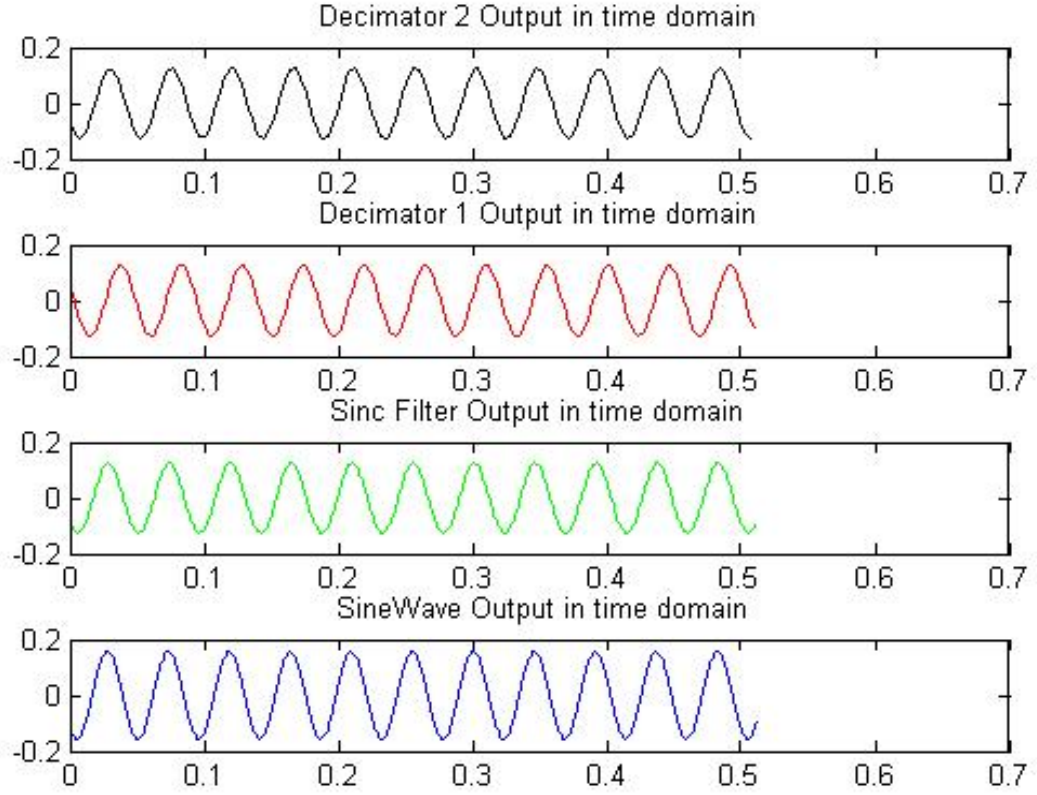


Figure 2.8: Time domain waveforms at output of each stage in Δ - Σ ADC

For calculating the FFT of the time domain outputs generated by the above functions we used “Hodiewindow” which windows the signal by generating the coefficients required for performing the FFT operation. This prevents spectral blurring.

The function “randflicker” implements the flicker noise of the input FETS of the OTA used in the analog integrators. This generates the random flicker noise sequence which is used in implementing the time domain noise in performing the time domain simulations on the second order modulator in Spectre®. The flicker noise coefficient K_f is passed as the argument to the function. The value of K_f is

obtained by performing the noise analysis on the OTA using Cadence's electrical simulator Spectre®.

Other functions like "ComputeSNR", which computes the signal to noise ratio, and "ComputeENOB", which computes the effective number of bits were written to perform the calculations needed to estimate the performance of the second-order Δ - Σ modulator.

CHAPTER 3

DESIGN OF SUB-SYSTEMS

The second order Δ - Σ modulator described thus far is comprised of two switched-capacitor (SC) integrator stages and a strobed comparator. The target technology is the AMIS 0.5 micron, NWELL process (C5N). The process supports three metal layers, double-poly capacitors, and a high resistance poly layer. The C5N process is a 5 Volt process.

The circuits described in this chapter were designed assuming the (nominal) process parameters shown below:

V_{TN} is threshold voltage of NFET = 0.75 Volts

V_{TP} is threshold voltage of PFET = -1 Volts

K_{PN} is transconductance parameter of NFET = 100 $\mu\text{A}/\text{V}^2$

K_{PP} is transconductance parameter of PFET = 32 $\mu\text{A}/\text{V}^2$

K_{aN} is 1/f noise parameter for NFET = 6.3×10^{-26} A F [Lee:02], [OCo:99]

K_{aP} is 1/f noise parameter for PFET = 3.8×10^{-30} A F

Delta-Sigma (Δ - Σ) Modulator design

The Δ - Σ modulator is a sampled data system so they are easily implemented with switched capacitor (SC) circuits. A fully-differential configuration has been adopted to ensure high power supply rejection, reduced clock feed through / switch charge injection errors, improved linearity, and increased dynamic range.

The Δ - Σ modulator consists of two SC integrators followed by a strobed comparator. In SC integrator circuits, the main sources of noise are thermal noise from the amplifier, kT/C noise resulting from the integrator switches, and flicker

noise from the amplifier. The complete schematic of the second order Δ - Σ modulator is shown in the following Figure 3.1.

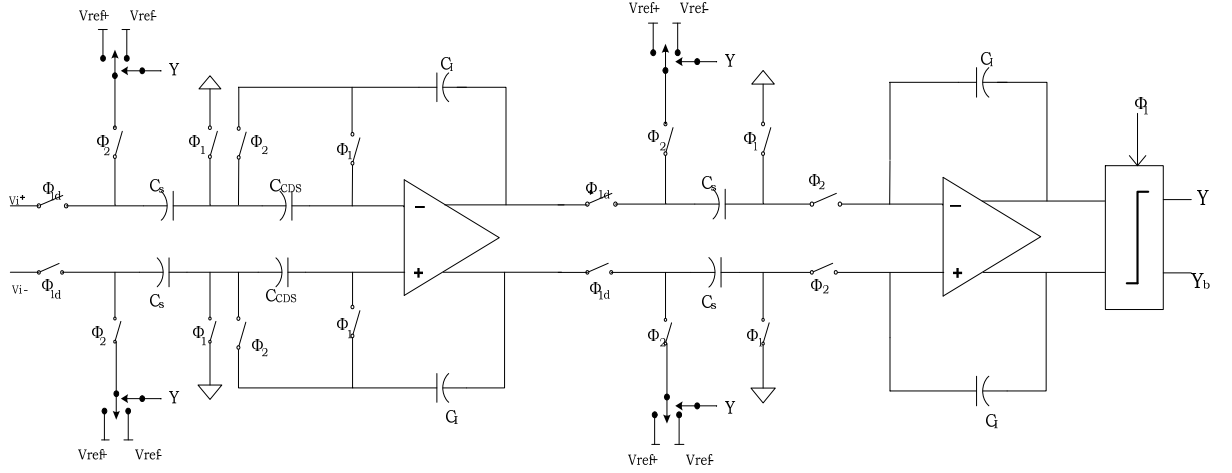


Figure 3.1: Second-order Δ - Σ modulator implementation

The operation of a second order Δ - Σ modulator is controlled by a non-overlapping two-phase clock. During phase 1, all of the switches labeled Φ_1 and Φ_{1d} are closed and all the switches labeled Φ_2 are open, and the input to each integrator is sampled onto capacitors C_s . During phase 2, all the switches labeled Φ_2 are closed and all the switches labeled Φ_1 and Φ_{1d} are open, and the charge stored on C_s is transferred to integrating capacitor C_i .

The strobed comparator will make a decision during Φ_1 , and the comparator will reset during Φ_2 . Therefore, during phase 2 the output of the two-level D/A network is subtracted from the input to each integrator [Ber:88]. With this clocking arrangement, the time available for the integration and the time for comparison are both one cycle. Both integrators have a gain of 0.5. It is evident from the clock

phases, that the modulator is fully pipelined by realizing both integrators with one sample delay. The two phase non overlapping scheme used for the implementation of modulator is shown in Figure 3.2.

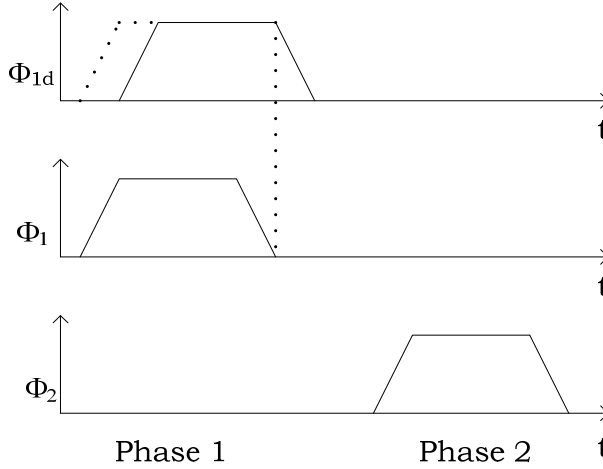


Figure 3.2: Clock diagram for second order Δ - Σ modulator

The effect of charge injection is suppressed by closing the switch Φ_1 slightly before Φ_{1d} . Since Φ_1 is connected to virtual ground node, they do not exhibit charge injection. Once Φ_1 has closed, and before the other has opened, C_S is floating; thus the subsequent closing of Φ_{1d} during the interval when both Φ_1 and Φ_2 are close will not inject charge onto C_S . Figure 3.2 shows the timing diagram for all the switches in the modulator. The clocks are non-overlapping in order to prevent charge sharing. The switches are closed when the controlling signals are high. The clock signal switch Φ_{1d} is generated by delaying the clock Φ_1 . The control clock signals are generated by the clock generator circuit which is discussed later in this section.

We used a two-level quantization because it avoids the need for matching level spacing's. A misplaced level introduces a change of quantization range and a dc offset, neither of which need be critical [Can:00]. For a two-level quantization process, the threshold level need not be accurately positioned because it is

preceded by the high DC gain of the integrator. Thus, two-level modulators can have very robust circuits. All the subsystems of the second order Δ - Σ modulator yet to be designed were initially modeled using VerilogA. The VerilogA descriptions can be found in Appendix B.

Integrator

A second-order Δ - Σ modulator consists of two integrators and one comparator. The two integrators in the forward path accumulate large quantization errors that result from the use of a two-level quantizer which forces their average to zero. For an ideal integrator, the output $y(nT)$, is the sum of the previous output $y((n-1)T)$, and the previous input, $x((n-1)T)$.

$$y(nT) = A \cdot x[(n-1)T] + y[(n-1)T] \quad \text{Eqn. 3.1}$$

The constant A represents the gain preceding the input to the integrator from Figure 2.6. Taking the z-transform for the Eqn 3.1 results in Eqn 3.2 which is the transfer function for an ideal integrator.

$$H(z) = \frac{Az^{-1}}{1 - z^{-1}} \quad \text{Eqn. 3.2}$$

Circuit implementation of the integrators results in many errors because of finite gain and bandwidth as well as those due to circuit nonlinearities.

First Integrator

In the design of Δ - Σ modulator the linearity, noise, and settling behavior associated with the first integrator are the most important circuit parameters. The rest of the components in the modulator loop and their non-idealities have less influence in the band of interest. Because of the low-frequency nature of the class of applications targeted in this design, careful attention needed to be paid to the $1/f$ noise behavior of the circuits employed.

As the name suggests $1/f$ noise has a power spectral density that decreases as frequency increases. MOSFETS in general display very poor $1/f$ noise characteristics (especially NFETS). One way to improve $1/f$ noise performance is to use very large devices but this is costly in terms of die area. Another approach is called Correlated Double Sampling (CDS) [Mil:00]. We adopted CDS to reduce flicker noise associated with the first integrator. The schematic of the first integrator with CDS is shown in Figure 3.3.

In an integrator with CDS, a non overlapping two-phase clock is used. Switches Φ_1, Φ_{1d} conduct during phase 1, and switches Φ_2 conducts during phase 2. Switch Φ_{1d} is opened slightly ahead of switch Φ_1 to reduce signal-dependant charge injection onto sampling capacitor C_S . During phase 1 the input V_{in} is sampled across C_S and amplifier offset is sampled on C_{CDS} . In the phase 2 charge difference proportional to the input voltage V_{in} and V_{ref} is transferred from C_S to C_I , while dc offset and flicker noise of the amplifier are cancelled by the voltage stored on C_{CDS} during phase 1. For proper operation we made C_{CDS} equal to C_S [Mil:00].

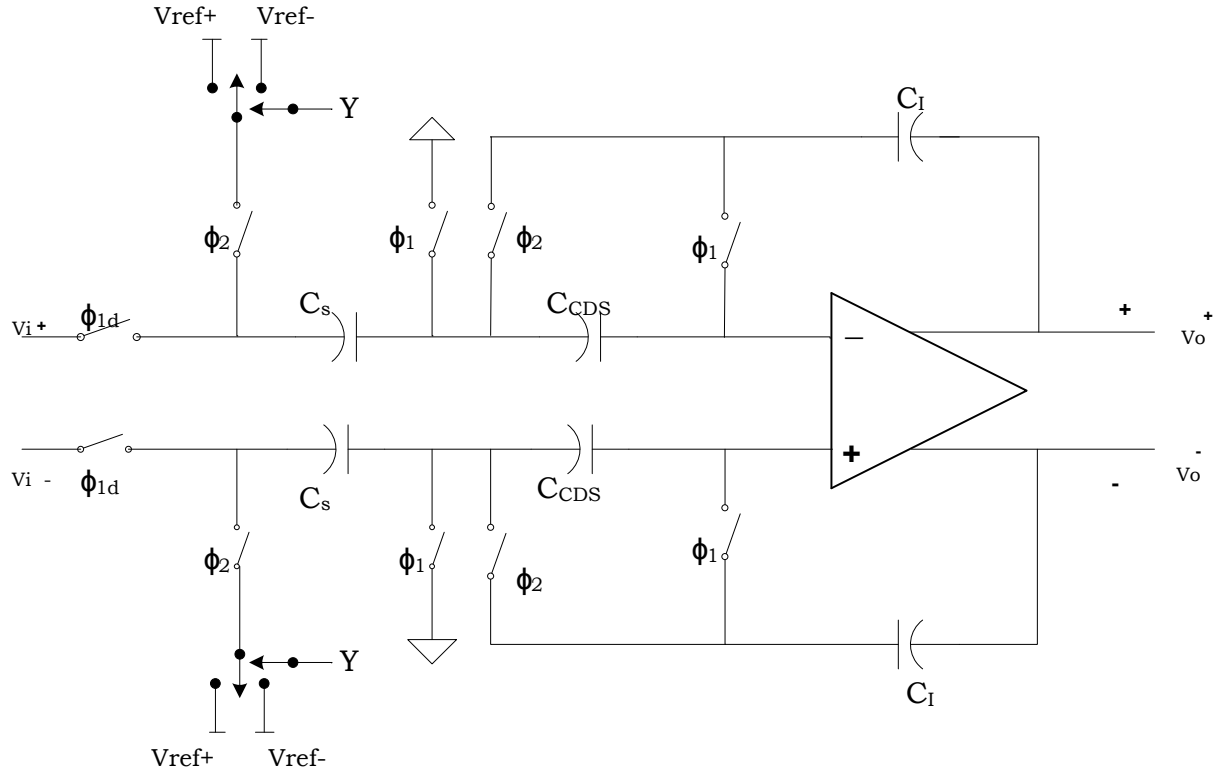


Figure 3.3: CDS integrator (first integrator)

Second Integrator

The MATLAB® results presented in Chapter 4 make evident that the linearity, noise, and settling behavior associated with the second integrator does not play a vital role in determining the system performance. Hence, a standard integrator design employing a parasitic insensitive resistor is used in this modulator. The schematic of the second integrator is shown in the Figure 3.4.

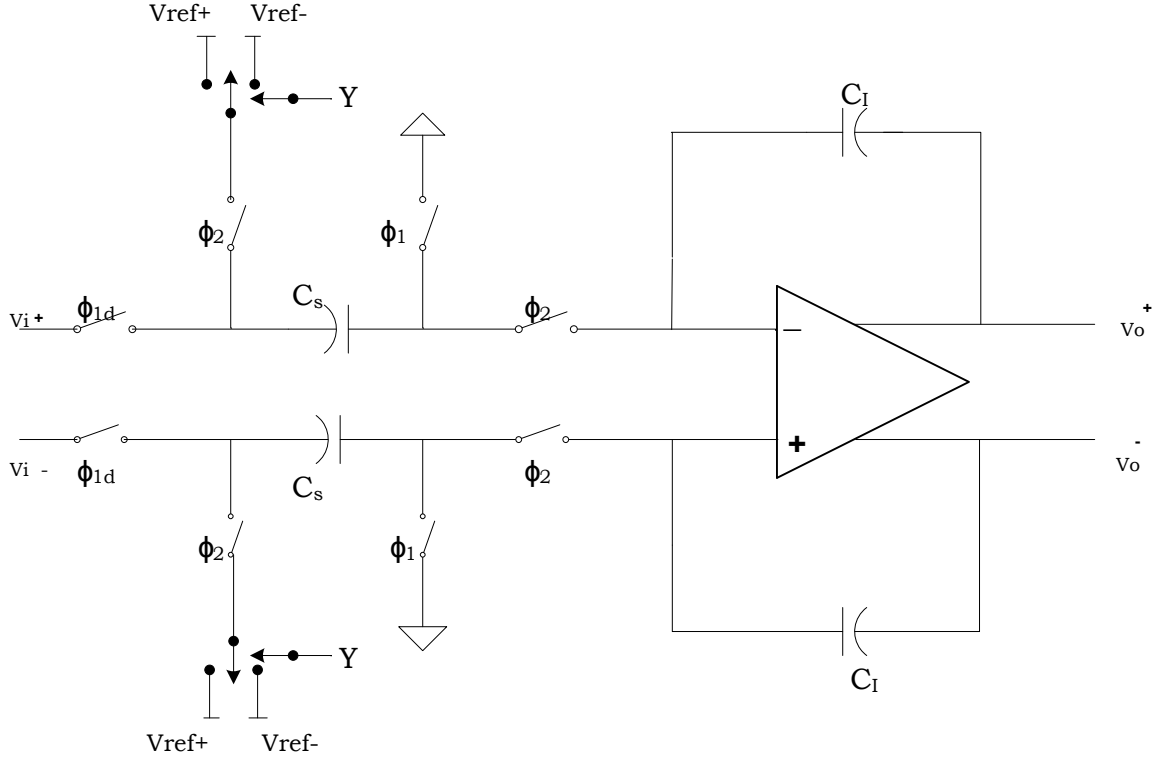


Figure 3.4: Second integrator

The key to success in realizing an integrator lies in the design of a differential amplifier. An amplifier that is not slew-limited is essential in order to avoid slewing distortion. The slew-rate requirement is more stringent for the implementation of integrator because the integration is accomplished only during Φ_2 and thus must be completed in one-half the clock cycle. The OTA is initially implemented using VerilogA code which includes the specifications needed. The electrical level simulations were initially performed using this VerilogA model. The VerilogA code can be found in Appendix B.

The OTA used in both the integrators is a folded cascode OTA with switched-capacitor common-mode feedback. We chose 1pF capacitors to use in the common-mode feedback network. The OTA schematic is illustrated in Figure 3.5. The bias

current flowing in device M_{11} is 20 μA . Since the frequency of the clock is 256 kHz, the period of the clock is 3.9 μsec and a half-period is 1.95 μs .

The OTA must meet the following specifications:

Input common mode range of 1.25 V to 3.75 V

Operate from a supply voltage of 4.75 V to 5.25 V

Slewing time of 1 μs and a linear settling time of 1 μs

Be able to drive load capacitances between 0.5 pF and 5 pF

Phase Margin of 60 degrees across all corners

Low frequency open loop gain of 80 dB across all corners

Total input referred integrated noise less than 100 μV

Input offset voltage less than 20 mV

Common Mode Rejection Ratio of 60 dB

Output voltage swing of +/- 1.2 V

The OTA frequency response using the typical process corner is as shown in Figure 3.6. As seen in figure, the low-frequency open-loop gain is 71 dB. The OTA has a GBW of 30 MHz with a phase margin of 60 degrees when driving a 5 pF load. Table 3.2 gives the Gain and phase margin for multiple process corners and loads.

Load = 1 pF	Typical	Worst case speed	Worst case power
Phase margin	54.5	52.9	53.9
Gain	84 dB	85 dB	84 dB
Bandwidth	9.2 MHz	9.4 MHz	10.4 MHz
Load = 5 pF			
Phase Margin	62.7	60.4	60.4
Gain	88 dB	88 dB	87 dB
Bandwidth	3.7 MHz	3.4 MHz	3.5 MHz

Table 3.1: Gain and phase margin of OTA for multiple process corners

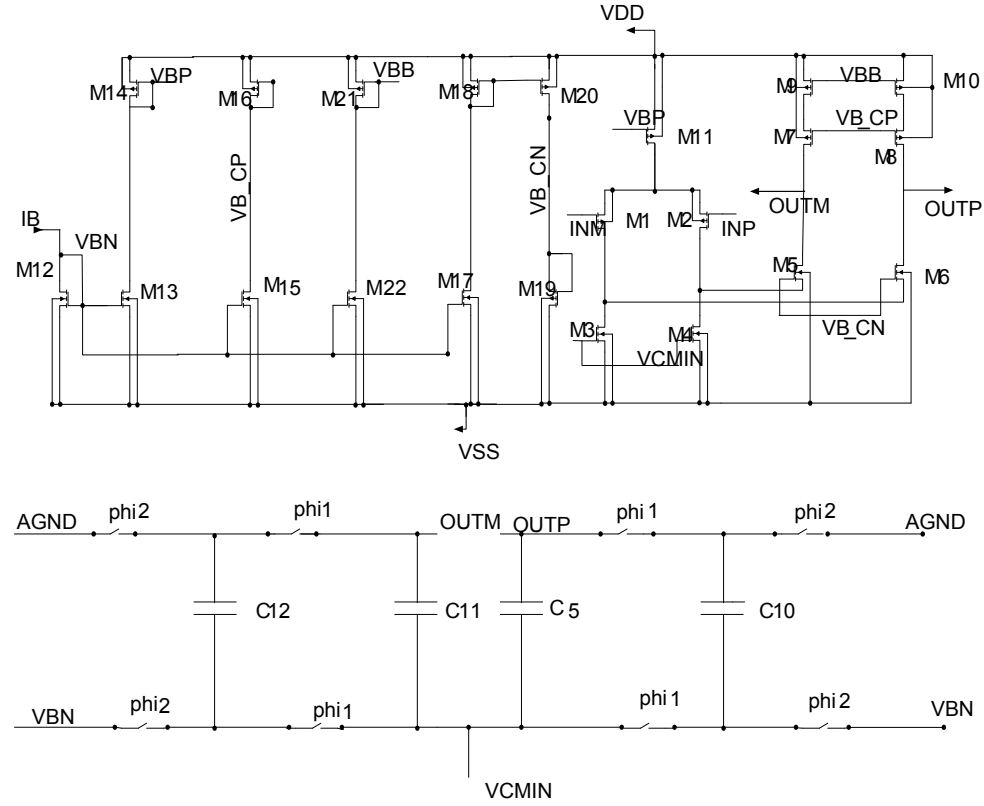


Figure 3.5: OTA schematic

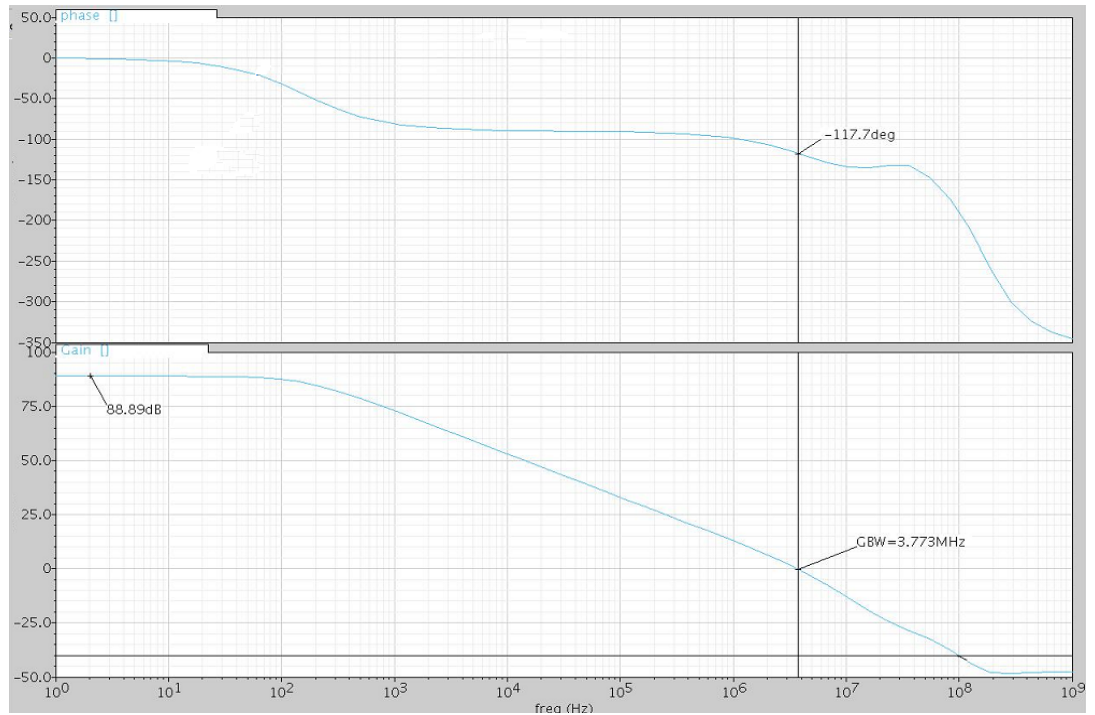


Figure 3.6: OTA frequency response

Transistor	Type	Width(μm)	Length(μm)	Multiplier
M1	p	59.4	2.15	2
M2	p	59.4	2.15	2
M3	n	2	6.6	3
M4	n	2	6.6	3
M5	n	50.9	4	1
M6	n	50.9	4	1
M7	p	87.5	2	1
M8	p	87.5	2	1
M9	p	31.1	15	1
M10	p	31.1	15	1
M11	p	24.9	8	1
M12	N	2	6.6	2
M13	n	2	6.6	2
M14	p	24.9	8	1
M15	n	2	6.6	2
M16	p	13.9	10	1
M17	n	2	6.6	2
M18	p	10	5	1
M19	n	8.12	20	1
M20	P	10	5	1
M21	p	31.1	15	1
M22	n	2	6.6	2

Table 3.2: Transistor sizes for OTA schematic

Comparator

The 1-bit quantizer in the forward path of a Δ - Σ modulator is implemented by using a comparator. The principle design parameters for the comparator are speed, input-referred noise, input offset and dead zone. The offset and noise are suppressed by the feedback loop of the modulator. The speed of the comparator must be adequate to achieve the desired sampling rate.

From the Verilog A simulation results it is evident that neither sensitivity nor offset considerations present stringent design considerations for the



Transistor	Type	Width(μm)	Length(μm)	Multiplier
M1	n	5	2	1
M2	n	5	2	1
M3	n	5	2	1
M4	n	5	2	1
M5	n	5	2	1
M6	n	5	2	1
M7	p	15	2	1
M8	p	15	2	1
M9	p	15	2	1
M10	p	15	2	1

Table 3.3: Transistor sizes for comparator schematic.

One implementation of a dynamic comparator is shown in Figure 3.7 and the transistor sizes are shown in Table 3.2. The lower set of NMOS devices (M_1 , M_2) operate in triode region and they are connected to the differential input. The upper cross-coupled inverter-latch regenerates when the latch clock goes high, the drain currents of the active switching NMOS devices are steered to obtain a final state determined by the mismatch of the total resistance. The outputs of the comparators are stored in dynamic latches, the outputs of which are buffered and then used to drive the switches in the feedback network. The important constraint in designing the latch is input voltage offset because it will limit the resolution of the comparator. The schematic of the latch is shown in the following Figure 3.8.

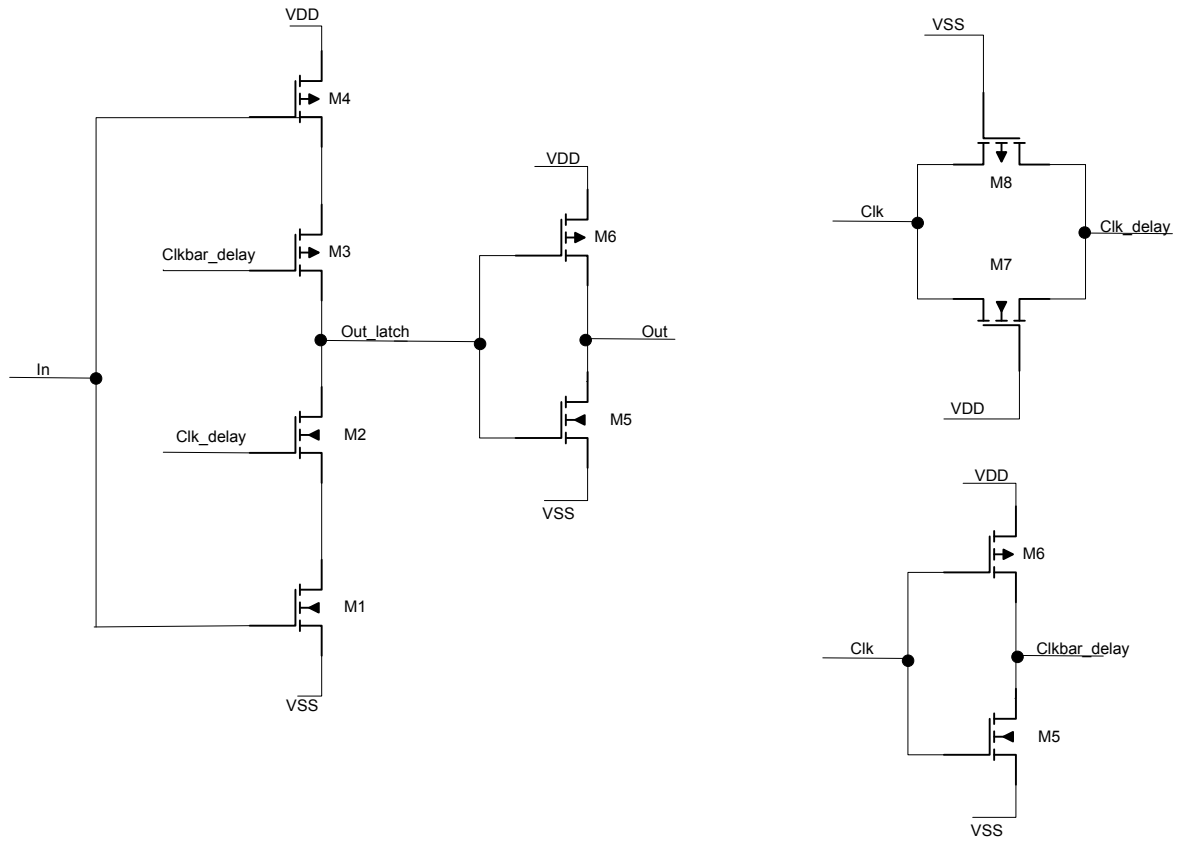


Figure 3.8: Latch schematic

Transistor	Type	Width(μm)	Length(μm)	Multiplier
M1	n	0.9	0.6	1
M2	n	0.9	0.6	1
M3	p	2.7	0.6	1
M4	p	2.7	0.6	1
M5	n	0.9	0.6	1
M6	p	0.9	0.6	1
M7	n	0.9	0.6	1
M8	p	0.9	0.6	1
M9	p	8.1	0.6	1
M10	n	2.7	0.6	1

Table 3.3: Transistor sizes for Dynamic latch schematic

Reference Generator

The Reference voltages required for the modulator are generated by using a reference generator circuit which is to be designed in future. The voltages generated are buffered by using two single ended OTA connected in unity gain configuration. The OTA used is a single ended OTA which is discussed in previous section. The OTA is connected in unity gain configuration as shown in Fig 3.9.

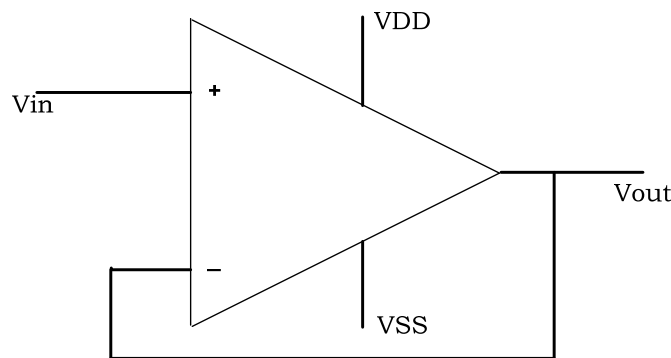


Figure 3.9 OTA in unity gain configuration

Clock Signal Generator

The clock signal Φ_1 , Φ_{1d} and Φ_2 are generated by clock generator circuit. The circuit consists of a cross coupled nor gates with a series of inverters as shown in the Figure 3.10. All the gates used to design the clock generator are designed using full complementary logic design. In this clock generator, the delay needed to avoid the overlap Φ_1 and Φ_2 is realized by the gate delays. The required delay between Φ_1 and Φ_{1d} is obtained by properly sizing the inverters. Since delay through the inverter is directly proportional to the length of the FETS used, the length of the FETS used in inverters is made long enough to obtain the required delay. The circuit is normally fed only by the digital supply lines; however it is better

arrangement to use the analog power lines for biasing the last four inverters (as shown in the Figure 3.10) which actually generates the required clock signals [Rou:86]. This arrangement will reduce the digital noise in the signal Φ_1 , Φ_{ld} and Φ_2 .

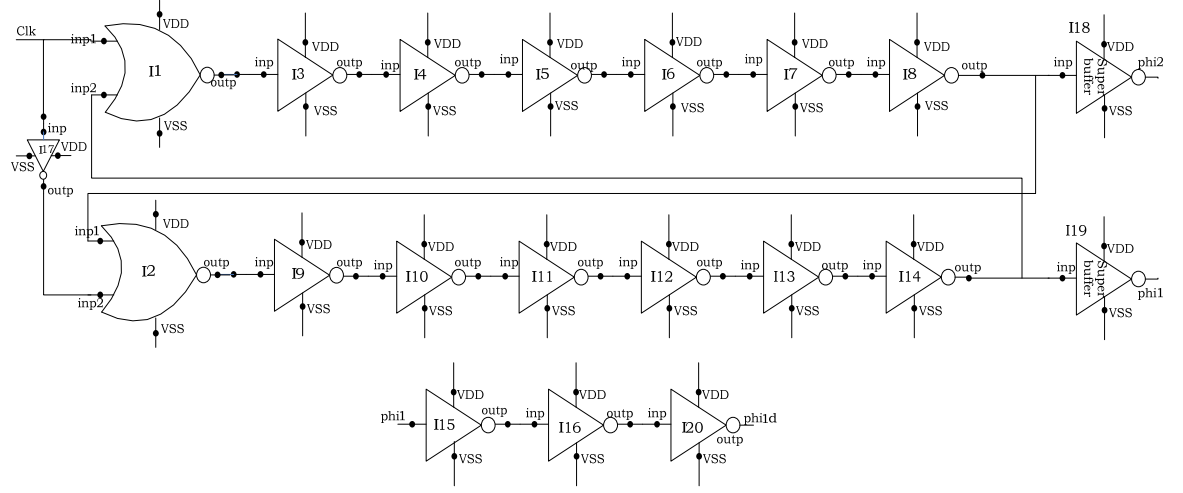


Figure 3.10: Clock generator schematic

CHAPTER 4

RESULTS

Chapter 4 presents the results of simulations performed on the second order Δ - Σ modulator at both behavioral level simulations and electrical level simulations. The effect of random offsets on the second order Δ - Σ modulator performance are accounted both in behavioral simulations performed in MATLAB® and in electrical simulations performed using Cadence's Spectre® program. The results of these analyses are also presented.

MATLAB® Simulation Results

The complete second order Δ - Σ modulator is implemented behaviorally using MATLAB® as explained in chapter 2. The MATLAB® code includes both integrated noise associated with the capacitor and reference noise.

The specifications used for MATLAB® simulation are listed below:

- Input sine wave frequency (f_0) is 22Hz.
- Sampling frequency (f_s) is 256 KHz.
- Frequency resolution of the samples (Resolution) is 2.
- Bandwidth (BW) is 125Hz.
- Number of samples of sine wave (N) is $f_s/\text{Resolution}$.
- Reference Voltage (V_{ref}) is 1.25.
- Amplitude of input sine wave is $V_{\text{ref}}/8$.
- Integrating capacitance C_{int} is 3pf.

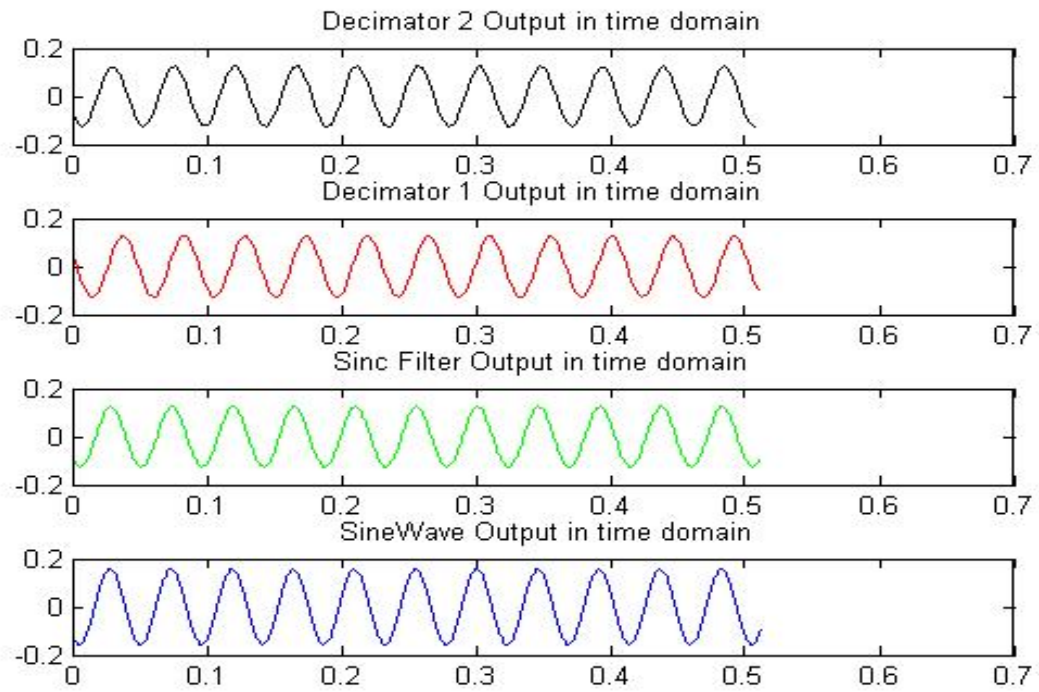


Figure 4.1: Time domain waveforms

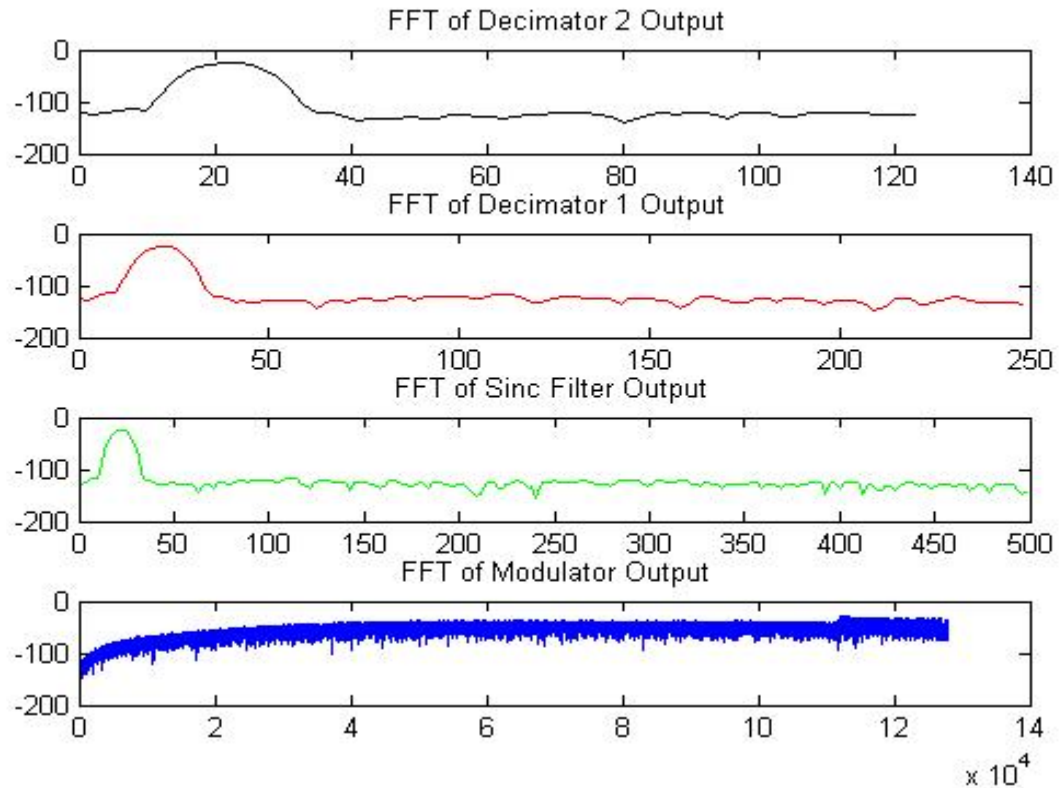


Figure 4.2: Frequency domain waveforms

The theoretically calculated effective number of bits by using the Eqn 2.6 is 22.86, and the obtained effective number of bits by considering all the noise parameters (integrated noise and reference noise) is 16.84. The time domain and frequency domain waveforms generated by the “DeltaSigmaTool” are shown in Figure 4.1 and Figure 4.2 respectively.

Electrical Simulation Results

To get and estimate of the ideal performance of second order Δ - Σ modulator VerilogA models of the components are used and electrical simulations are performed on it using cadence AMS simulator. Later each component is replaced by its transistor level design and simulations are done.

The VerilogA models initially used are ideal models of the components and the obtained ENOB is 21.73 as shown in the table 4.1. Later the effects due to every component are modeled one after the other to clearly observe which has the maximum effect on the performance of the modulator. The simulation results are tabulated in the following table 4.1. The important thing to be observed in the table is adding the thermal noise to components lost 3 bits and adding correlated double sampling to the first integrator gained us a bit. From the table it is also clear that the effects related to comparator such as offset and dead zone does not play a vital role in degrading the performance of the modulator.

S.No	Effects modeled	ENOB
1	Noise free switch and OTA (Ideal!)	21.73
2	Added thermal noise of switch and thermal noise of OTA	18.47
3	Added 1/f noise (no CDS)	17.78
4	Added thermal and 1/f noise to reference	17.21
5	Added very large comparator offset (100mV)	17.3
6	Added CDS to 1 st integrator	18.4
7	Added very large comparator dead zone (20mV)	18.05
8	Added Phi1d	18.22

Table 4.1 Results obtained with VerilogA models

After getting an estimate of the performance of a second order Δ - Σ modulator, VerilogA models of all the components are replaced by their corresponding transistor level designs. In cadence the noise associated by the components (OTA) cannot be simulated in time domain analysis, so a MATLAB® script “NoiseGenerator” is written which is used to generate the time domain noise sequence for performing the time domain simulations on the second order modulator in Spectre®. The results obtained after performing the time domain simulations with transistor models are tabulated below in table 4.2.

S.No	Effects modeled	ENOB
1	Noise free transistor switch and VerilogA OTA (no delayed clocks), VerilogA comparator.	19.54
2	Added thermal and 1/f noise of reference	18.75
3	Added thermal and 1/f noise of integrator OTAs (CDS used in 1 st integrator)	18.97
4	Added a noise free comparator	17.54 (due to harmonics)
5	Thermal noise of switches	18.3 (Dithering suppressed harmonics)

Table 4.2 Results obtained with Transistor models

The important observation in table 4.2 is adding thermal noise to the switches increased the ENOB by 1 bit instead of degrading the ENOB. This is because of the dithering phenomenon. The problem in Δ - Σ modulators is quantization noise becomes periodic in response to a low level DC inputs. Injecting a noise like signal at the input randomizes the quantization noise. This process of injecting the noise at the input is called dithering. If the circuit thermal noise is large then it will act as dither to the input signal thereby suppressing the harmonics.

CHAPTER 5

SUMMARY/FUTURE WORK

Summary

This thesis presented the design and simulation of a small, low-power, second-order, Δ - Σ (delta-sigma) modulator intended for use in multi-channel, portable systems where there is a need for digitizing biomedical signals such as Electro-Cardiogram (ECG), Electroencephalogram (EEG), or Electrocorticography (ECoG). In particular, the modulator may someday be fabricated in a 5-Volt AMIS 0.5 μm , double-poly with high-resistance layer, tri-metal CMOS process (C5N) and used with a Brain-Controlled Interface (BCI) which employs ECoG.

The second order modulator was implemented in a switched-capacitor technology with a fully differential architecture. To reduce the $1/f$ noise because of first integrator correlated double sampling technique was employed. The modulator operates at a frequency of 256 kHz, uses a 1.2 Volt reference. The modulator shows 112.3dB signal-to-noise and distortion ratio (SNDR) in a 125 Hz signal bandwidth. This corresponds to an ENOB (Effective Number of Bits) of 18.3 bits. The modulator is expected to occupy an area of 0.19 mm^2 .

A strobed comparator, a fully differential folded-cascode amplifier required for implementing the integrators and a single ended folded-cascode amplifier for voltage reference were successfully designed at transistor level. Both VerilogA-based and transistor-level simulations were performed using Cadence's Spectre® electrical simulator.

A MATLAB-based simulation/analysis tool was developed. It implements the complete second order Δ - Σ ADC in MATLAB® and can be used to simulate the second order modulator (with some non idealities) and can also be used to analyze the performance of second order Delta-Sigma modulator.

Future Work

The modulator simulates correctly at the typical process corner, simulations at the “worst-case-power” and the “worst-case-speed” process corners are yet to be performed. The electrical simulations presented in the thesis were performed with ideal input and reference voltages and not from reference voltage generator. The reference voltage generator and the current source are to be developed which are needed to implement the modulator. The modulator has to be tested in order to account the effect of capacitor mismatches. The electrical simulations are performed by using a VerilogA clock generator. The modulator is to be tested by replacing the VerilogA model by the transistor level design of the clock generator. The modulator is to be tested by adding the time domain noise to the comparator.

The digital sinc³ filter circuit preceding the modulator is to be developed. The custom DSP core which implements the FIR filters is to be developed. The modulator must be physically laid out, verified against schematic, and design rule verified.

REFERENCES

- [All:03] Phillip E. Allen and Douglas Holberg, "CMOS Analog Circuit Design: Second Edition", Oxford university Press, 2003.
- [Bos:88] Bernhard E. Boser, Bruce A. Wooley, "The Design of Sigma-Delta Modulation Analog-to-Digital Converters", IEEE Journal of Solid-State Circuits, Vol. 23, No. 6, December 1988.
- [Can:00] James C. Candy and Gabor C. Temes, "Oversampling Methods for A/D and D/A Conversion".
- [Den:07] Tim Denison, Kelly Consoer, Wesley Santa, Al-Thaddeus Avestruz, John Cooley, and Andy Kelly, "A 2W 100 nV/sqrt(Hz) Chopper-Stabilized Instrumentation Amplifier for Chronic Measurement of Neural Field Potentials", IEEE Journal of Solid-State Circuits, Vol. 42, No. 12, December 2007.
- [Fit:04] Tony Fitzpatrick, "Thought Control: Human Subjects Play Real Mind Games", Washington University in St. Louis Record, Vol. 28, No. 35, June 2004.
- [Har:03] Reid R. Harrison and Cameron Charles, "A Low-Power Low-Noise CMOS Amplifier for Neural Recording Applications", IEEE Journal of Solid-State Circuits, Vol. 38, No. 6, June 2003.
- [Han:07] Grani A. Hanasusanto, Yuanjin Zheng, "A Chopper Stabilized Pre-amplifier for Biomedical Signal Acquisition", 2007 IEEE International Symposium on Integrated Circuits (ISIC-2007).
- [Mil:00] Milovanovic D, Savic M, Nikolic M "A Third Order Sigma-Delta Modulator".
- [Mor:09] <http://labs.seas.wustl.edu/bme/dmoran/>
- [Sch:06] Andrew Schwartzm X. Tracy Cui, Douglas J. Weber, and Daniel Moran, "Brain-Controlled Interfaces: Movement Restoration with Neural Prosthetics", Neuron 52, October 5, 2006 (205-220).
- [Tem:96] Christian Enz and Gabor Temes, "Circuit Techniques for Reducing the Effects of Op-Amp Imperfections: Autozeroing, Correlated Double Sampling, and Chopper Stabilization", Proceedings of the IEEE, Vol. 84, No.11, November 1996.
- [Ter:04] Stephen C. Terry, Benjamin J. Blalock, James M. Rochell, M. Nance Ericson, and Sam D. Caylor, "Time-Domain Noise Analysis of Linear Time-Invariant and Linear Time-Variant Systems Using MATLAB® and HSPICE",

- [Woo:97] Bruce Wooley and Shahriar Rabii, "A 1.8-V Digital-Audio Sigma-Delta Modulator in 0.8- μm CMOS", IEEE Journal of Solid-State Circuits, Vol. 32, No. 6, June 1997.

APPENDIX A

MATLAB® Code

```

function[] = DeltaSigmaTool()
%
% This is a tool to analyze the performance of second order delta sigma modulators.
%
TRUE = 1 ;
FALSE = 0 ;

ReadFromFile = TRUE; %FALSE ; % Set equal to TRUE if you want to import modulator vector
LoadQ = FALSE ;      % Set to TRUE if you want to load previous generated q vector

Resolution = 2 ;      % Frequency resolution in Hertz
Fs = 256e3 ;          % Modulator sampling frequency
Fo = 22 ;             % Frequency of input sine wave
BW = 125 ;            % Desired ADC bandwidth in Hertz
NumTaps_1 = 59 ;      % Number of taps to use in first decimation filter
NumTaps_2 = 59 ;      % Number of taps to use in second decimation filter

Vref = 1.25 ;         % Reference voltage
A = Vref / 8.0 ;      % Input sine wave amplitude
Cint = 3e-12 ;        % Value of integrating capacitor in modulator
NOISE = TRUE ;        % When true noise is added in modulator

D = (Fs/(2 * BW)) / 4 ; % Intermediate decimation factor used by sinc filter
D = round(D) ;
STDOUT = 1 ;          % File identifier for the screen
fid = STDOUT ;

N = Fs / Resolution ; % Length of input time sequence (2 to some power)
N = 2 ^ (round(log2(N))) ;
%
% Save screen output to diary file
%
diary on ;
fprintf(fid, '\nSampling frequency is %g Hz.\n', Fs) ;
fprintf(fid, 'Desired output bandwidth is %g Hz.\n', BW) ;
fprintf(fid, 'Signal frequency is %g Hz.\n', Fo) ;
fprintf(fid, 'Desired frequency resolution is %g Hz.\n', Resolution) ;
fprintf(fid, 'Number of taps in first decimation filter is %g.\n', NumTaps_1) ;
fprintf(fid, 'Number of taps in second decimation filter is %g.\n', NumTaps_2) ;
fprintf(fid, 'Integrating capacitor is %g F and referece is %g Volts.\n', Cint, Vref) ;
if (NOISE == TRUE)
    fprintf(fid, 'Noise generation is turned on.\n') ;
end
if (ReadFromFile == TRUE)
    fprintf(fid, 'Reading from file is turned on. Ignore above.\n') ;
end
if (LoadQ == TRUE)
    fprintf(fid, 'Loading previously save modulator output. Ignore above.\n') ;
end
fprintf(fid, '\n') ;

%
% Let us do a few quick error checks before continuing

```

```

%
    ErrorCheck(Fs, N, Fo) ;
%
% Apply sinewave input to second order delta-sigma ADC.
% DSP block consists of a sinc cubed filter followed by two decimation filters
% Sinc cubed filter decimates by factor of D.
% Each decimation filter decimates by a factor of two.
% We want the time sequences we take the FFT of to be length N but since
% there is delay associated with the filters, let's make original sequence 2N long.
% We will take the middle N samples which should be free of any transients.
%
    SineWave = SineWaveGenerator(2*N, A, Fs, Fo) ;
    if (ReadFromFile == FALSE)
        ModulatorOutput = SecondOrderModulator(SineWave, Vref, Cint, NOISE) ;
        save ModulatorOutput ;
    elseif (LoadQ == TRUE)
        load ModulatorOutput ;
    else
        rfid = fopen('SC_AMS_allreals_realClockGen.txt') ;
        ModulatorOutput = fscanf(rfid, '%g');
        L = length(ModulatorOutput) ;
        if (L < 2*N)
            error('Modulator time sequence is too short') ;
        else
            ModulatorOutput = ModulatorOutput(1 : 2*N)' ;
        end
    end
    SincOutput = SincCubedFilter(ModulatorOutput, D) ;
    Decimator_1_Output = DecimationFilter(NumTaps_1, SincOutput) ;
    Decimator_2_Output = DecimationFilter(NumTaps_2, Decimator_1_Output) ;
%
% Lets use the middle group of samples in each array so that we don't have transients
% to contend with. For the decimators we need to make sure that length (L) of sequence is
% 2 raised to some power
%
    L = length(SineWave) ;
    SineWave = SineWave(L/4+1 : 3*L/4) ;
    L = length(ModulatorOutput) ;
    ModulatorOutput = ModulatorOutput(L/4+1 : 3*L/4) ;
    L = length(SincOutput) ;
    SincOutput = SincOutput(L/4+1 : 3*L/4) ;
    L = length(Decimator_1_Output) ;
    L = 2^(round(log2(L)-0.5)) ;
    Decimator_1_Output = Decimator_1_Output(L/4+1 : 3*L/4) ;
    L = length(Decimator_2_Output) ;
    L = 2^(round(log2(L)-0.5)) ;
    Decimator_2_Output = Decimator_2_Output(L/4+1 : 3*L/4) ;
%
% Plot time domain waveforms in one figure but in different strips so we can compare waveforms
%
    figure(1) ;
    hold on ;
    TimeDomainPlot(SineWave, Vref, Fs, 'b', 4, 4) ;
    TimeDomainPlot(SincOutput, Vref, Fs/D, 'g', 3, 4) ;
    TimeDomainPlot(Decimator_1_Output, Vref, Fs/(2*D), 'r', 2, 4) ;
    TimeDomainPlot(Decimator_2_Output, Vref, Fs/(4*D), 'k', 1, 4) ;
    hold off ;

```

```

%
% Take FFTs
%
ModulatorOutFFT = FFTmagnitude(ModulatorOutput) ;
SincOutFFT = FFTmagnitude(SincOutput) ;
Decimator_1_FFT = FFTmagnitude(Decimator_1_Output) ;
Decimator_2_FFT = FFTmagnitude(Decimator_2_Output) ;
%
% Plot frequency domain waveforms in one figure but in different strips for easy comparison
%
figure(2) ;
hold on ;
FreqDomainPlot(ModulatorOutFFT, Vref, Fs, 'b', 4, 4) ;
FreqDomainPlot(SincOutFFT, Vref, Fs/D, 'g', 3, 4) ;
FreqDomainPlot(Decimator_1_FFT, Vref, Fs/(2*D), 'r', 2, 4) ;
FreqDomainPlot(Decimator_2_FFT, Vref, Fs/(4*D), 'k', 1, 4) ;
hold off ;
%
% Compute SNRs and ENOBs for the various waveforms
%
ModulatorSNR = ComputeSNR(ModulatorOutFFT, Fs, Fo, BW);
ModSimENOB = ComputeENOB(Vref, A, ModulatorSNR) ;
fprintf(fid, 'At modulator output with perfect filter, SNR is %g dB and ENOB is %g bits.\n', ModulatorSNR,
ModSimENOB) ;
SNRtheory = TheoreticalSNR(Fs, BW) ;
ModTheoryENOB = ComputeENOB(1,1, SNRtheory) ;
fprintf(fid, 'Theoretically predicted ENOB is %g bits.\n', ModTheoryENOB) ;

SincSNR = ComputeSNR(SincOutFFT, Fs/D, Fo, 4*BW);
SincSimENOB = ComputeENOB(Vref, A, SincSNR) ;
fprintf(fid, 'At sinc output, SNR is %g dB and ENOB is %g bits.\n', SincSNR, SincSimENOB) ;

Decimator_1_SNR = ComputeSNR(Decimator_1_FFT, Fs/(2*D), Fo, 2*BW);
Dec_1_SimENOB = ComputeENOB(Vref, A, Decimator_1_SNR) ;
fprintf(fid, 'At first decimation filter output, SNR is %g dB and ENOB is %g bits.\n', Decimator_1_SNR,
Dec_1_SimENOB) ;

Decimator_2_SNR = ComputeSNR(Decimator_2_FFT, Fs/(4*D), Fo, BW);
Dec_2_SimENOB = ComputeENOB(Vref, A, Decimator_2_SNR) ;
fprintf(fid, 'At second decimation filter output, SNR is %g dB and ENOB is %g bits.\n', Decimator_2_SNR,
Dec_2_SimENOB) ;

diary off ;
end

```



```

function[out] = SecondOrderModulator(x, Vref, Cint, NOISE)
%
% Simulates the linear portions of a 2nd-order sigma-delta converter
%
% x is the time sequence input waveform
%
%
% Noise parameters of the modulator
%
alpha_int = 2.5 ;           % Noise fudge factor for integrator
alpha_ref = 2.5 ;           % Noise fudge factor for reference generator
Ut = 26e-3;                 % Thermal voltage
q = 1.602e-19;              % Charge of electron
sigma_int = alpha_int*sqrt(Ut*q/Cint); % Standard deviation of integrator Noise
sigma_ref = alpha_ref*sqrt(Ut*q/Cint); % Standard deviation of ref Noise

%
% Description of second order modulator
% Modulator is composed of two integrators: the "w" integrator and the "y" integrator
% The 1-bit modulator output is the output of the comparator, q.

w(1) = 0.0;
y(1) = 0.0;
q(1) = -1;

%
% Use the length function to determine how long input time sequence is
%
L = length(x);
%
% Need to generate white noise to add into integrator outputs and to reference.
%
wn = sigma_int * randn(1,L) ;
yn = sigma_int * randn(1,L) ;
rn = sigma_ref * randn(1,L) ;
%
% Only add noise when user asks for it
%
if (NOISE == 1)
    for k = 2:L
        VrefPlusNoise = Vref + rn(k) ;
        w(k) = w(k-1) + 0.5 * ( x(k) - (VrefPlusNoise * q(k-1)) ) + wn(k) ;
        y(k) = y(k-1) + 0.5 * ( w(k-1) - (VrefPlusNoise * q(k-1)) ) + yn(k) ;
        q(k) = sign(y(k));
    end
else
    for k = 2:L

        w(k) = w(k-1) + 0.5 * ( x(k) - (Vref * q(k-1)) ) ;
        y(k) = y(k-1) + 0.5 * ( w(k-1) - (Vref * q(k-1)) ) ;
        q(k) = sign(y(k));
    end
end
out = q ;
end

```

```

function[out] = SincCubedFilter(q, decifactor)

% SincCubedFilter
%   Simulates a Sinc cube filter and returns a column vector
%   of the filtered digital data. Was designed to be used
%   with the second order Sigma-Delta modulator

K = 3; % Order of sinc filter = order of SigmaDeltaMod + 1
bits = K*log2(decifactor) + 1; % The number of bits of the accumulators in the sinc cube filter.

max = 2^bits ;
maxnum = max - 1 ;
posmax = max/2 - 1 ; % Fixes the maximum possible positive # in accumulator
negmax = max/2; % Fixes the maximum possible negative # in accumulator
value = 1 ;

index=1;
a=0;
b=0;
c=0;
d=0;
e=0;
f=0;
sincout=0;
%
% Start the loop for repeated operations to convert the single bit data
% to the required no. of bits having the desired value.
%

for i=1:length(q)
    if q(i)==1
        in = value;
    else
        in = max - value;
    end

%
% Integrator stages are defined below
%

    asum = in + a;
    if (asum > maxnum)
        asum = mod(asum,max);
    end

    bsum=asum+b;
    if (bsum > maxnum)
        bsum = mod(bsum,max);
    end

    csum=bsum+c;
    if ( csum > maxnum)
        csum = mod(csum,max);
    end
end

```

```

a=asum;
b=bsum;
c=csum;
integ_out = csum;

%
% Decimation and differentiation of the integrator output is defined below
%

if (rem(i,decifactor)==0)
    dsum=integ_out+(max-d);
    if (dsum > maxnum)
        dsum=mod(dsum,max);
    end
    esum=dsum+(max-e);
    if (esum > maxnum)
        esum = mod(esum,max);
    end
    fsum=esum+(max-f);
    if (fsum > maxnum)
        fsum = mod(fsum, max);
    end
    d=integ_out;
    e=dsum;
    f=esum;
    sincout=fsum;
    if (sincout > posmax)
        out(index) =((-1*(max-sincout))/negmax);
    else
        out(index) =(sincout/negmax) ;
    end % end if

    index=index+1;
end % end for loop

end

```

```
function[OutputSignal] = DecimationFilter(NumTaps, InputSignal)
%
% This script implements a halfband filter and decimates by a factor of 2.
% The number of taps to use is specified as an input parameter.

%
% Generate the filter coefficients
%
    FilterCoefs = HalfbandFilter(NumTaps);
%
% The convolution function performs filtering in the time domain
%
    out_temp = conv(InputSignal, FilterCoefs);
%
% Decimate output vector by factor of 2
%
    OutputSignal = out_temp(1:2:length(out_temp));

end
```

```

function[out] = HalfbandFilter(taps)

% HALFBAND  Function 'halfband' generates a column vector of halfband
%           filter coefficients. The filter is a low pass filter, with
%           a pass band of 1/4 th of the sampling frequency.

%           where TAPS is the no. of taps of the full band filter. TAPS
%           must be an ODD number for this function to work. The returned
%           value COEF is the coefficients of the halfband filter. The size
%           of COEF is 2*TAPS+1.
%
%           This function uses the Parks-McLellan optimisation algorithm
%           for FIR filters.

f=[0.0 0.9 1.0 1.0];
m=[1.0 1.0 0.0 0.0];
%w=[8 2];
w=[50 1];
tmp=remez(taps,f,m,w);

ftaps=taps*2+1;
g(1:ftaps)=0.0;
g((ftaps+1)/2)=1.0;
g(1:2:(ftaps+1)/2-1)=tmp(1:(taps+1)/2);
g((ftaps+1)/2+1:2:ftaps)=tmp((taps+1)/2+1:(taps+1));

g=1/2*g; % scaling of magnitude.

%[h,w]=freqz(g,1,256,500);
out=g;
%semilogy(abs(w),abs(h));
%title('Frequency response of second filter');
end

```

```

function [out] = HodieWindow(N)
%
% Script to compute a N-term floating point Hodie window
%
% Copyright 2001 Eric Swanson
%
% N is the number of filter coefficients
%

    window = zeros(N,1);
    bottom = zeros(N,1);
    top = 2.5 * ones(N,1);
    index = [0:N-1]';
    v = 2 * pi / N;

%
% Hodie window cosine coefficients
%
    a0=0.61640321314050;
    a1=0.98537119272586;
    a2=0.49603771622007;
    a3=0.14992232793243;
    a4=0.02458719103474;
    a5=0.00176604651487;
    a6=0.00003158118857;
%
% Coefficients sum to N
%
    for m=1:N;
        n1=m-.5;
        window(m,1)=a0-a1*cos(v*n1)+a2*cos(v*2*n1)-a3*cos(v*3*n1)+a4*cos(v*4*n1)-
a5*cos(v*5*n1)+a6*cos(v*6*n1);
    end;
    out = window;
%
%plot
%
%plot(index,window,'-g',index,bottom,'-b',index,top,'-b');
%axis([0 1000 0 2.5]);
%axis off;

```

```

function[SineWave] = SineWaveGenerator(N, A, Fs, Fo)
%
% Generates a sinusoidal time sequency
%
% N is the number of samples to in the time sequence
% A is the peak amplitude of the sine wave
% Fs is the sampling frequency
% Fo is the frequency of the sine wave in Hertz

%
% Fhat is the normalized frequency i.e. F0/Fs
%
Fhat = Fo / Fs ;
K = 2 * pi * Fhat ;
index = linspace(0, N-1, N) ;
SineWave = A * sin(K * index);

% Ts = 1 / Fs ;
% time = Ts * index ;
% plot(time, SineWave)
%
end

```

```

function[] = NoiseGenerator()

%
% Script to generate noise vector containing both white and flicker noise
% components.
%
% Paramters
%
TRUE  = 1 ;
FALSE = 0 ;

Ut    = 26e-3;           % Thermal voltage
qe    = 1.602e-19;       % Charge of electron
Rn    = 1e4 ;            % Effective noise resistance in Ohms
NBW   = 1e6 ;            % Desired noise bandwidth
Kf    = 7e-11 ;          % Flicker noise coefficient
time  = 1.3 ;            % Length of sequence in seconds

WRITE_FILE = TRUE ;

%
% Thermal noise standard deviation
%
sigma = sqrt(4 * Ut * qe * Rn * NBW)
%
% Determine the length of vector in samples
% NBW = 1/2 * SampleFrequency
%
SampleFreq = 2 * NBW ;
T = 1 / SampleFreq ;
%
% Compute time vector
%
N = round(time / T) ;
N = 2 ^ round(log2(N)+0.5)
t = T * linspace(0, N-1, N) ;
%
% Compute white noise vector
%
wn = sigma * randn(1, N) ;
%
% Compute flicker noise vector
%
fn = randflicker(1, N, Kf) ;
%
% Add the two together
%
noise = wn + fn ;
%
% Plot in time domain so we can see of waveform looks reasonable
%
figure(1) ;
plot(t, noise) ;
%
% And also plot in frequency domain
%
mag = abs(fft(noise)) ;

```



```

DeltaFreq = SampleFreq / N
freq = DeltaFreq * linspace(0, N/2-1, N/2) ;
mag = 1/N * mag(1 : N/2) ;
figure(2) ;
loglog(freq, mag) ;

%
% Open a ASCII text file and write results to file
% We need to create a PWL file for Spectre
%
if (WRITE_FILE == TRUE)
    fid = fopen('noise.pwl','w') ;
end

deltT = T - T/100 ;
deltTinMS = 1e3 * deltT ;
deltTinUS = 1e6 * deltT ;
for k = 1 : N
    if (WRITE_FILE == TRUE)
        t(k) = 1e3 * t(k) ;
        fprintf(fid, '%15.8fm\t%g\n', t(k), noise(k)) ;
        fprintf(fid, '%15.8fm\t%g\n', t(k)+ deltTinMS, noise(k)) ;
    end
end % end for
if (WRITE_FILE == TRUE)
    status = fclose(fid) ;
end

end

```

```

function yk = randflicker(M,N,kf)
%RANDFLICKER    Time-domain flicker noise generation.
%    RANDFLICKER(M,N,kf) is an M-by-N matrix with random entries chosen
%    from
%    a time-domain simulation of flicker (1/f) noise.  M is the number of
%    trials of flicker noise to create.  N is the time-dependent number of
%    samples in the flicker noise sequence.

% Input parameters
alpha = 1;
wk_sigma = 2.5*sqrt(kf);

% Generate white noise for the sequence wk.
wk = wk_sigma * randn(M,N);

% Generate the coefficients hk.
hk = zeros(1,N);
hk(1+ 0) = 1;
for k = 1:N-1
    hk(1+ k) = (k - 1 + alpha/2) .* (hk(1+ k-1) / k);
end

% Take the fast fourier transform of hk and wk and compute the power
% spectral densities and spectral densities.
hk_fft = fft(hk); clear hk;
wk_fft = fft(wk,[],2); clear wk;
yk_fft = wk_fft .* repmat(hk_fft, [M,1]); clear hk_fft wk_fft;

yk = real(ifft(yk_fft,[],2));

```

```

function [SNR] = ComputeSNR(FFT, Fs, Fo, BW)

%
% Script to compute signal-noise-ratio
%

%
% Compute signal bin
%
    L = length(FFT) ;
    BinSize = Fs / L ;
    SignalBin= round(Fo/BinSize) ;
%
% Hodie Window confines spectrum to Signal Bin +/- 8 bins
%
    LowerSignalBin = SignalBin - 8 ;
    UpperSignalBin = SignalBin + 8 ;
%
% Compute end of BW bin
%
    BWbin= round(BW/BinSize) ;
%

    SumSignalSquared = 0.0 ;
    SumNoiseSquared = 0.0 ;
%
% Calculate noise power preceding signal bins
%
    for k = 2 : 1: (LowerSignalBin - 1)
        SumNoiseSquared = SumNoiseSquared + FFT(k) * FFT(k);
    end
%
% Estimate the noise power in the signal bins
%
    EstNoisePwr = FFT(LowerSignalBin - 1) * FFT(LowerSignalBin - 1) ;
    EstNoisePwr = EstNoisePwr + FFT(UpperSignalBin + 1) *
FFT(UpperSignalBin + 1) ;
    EstNoisePwr = EstNoisePwr / 2.0 ;
%
% Calculate signal power
%
    for k = LowerSignalBin: 1 : UpperSignalBin
        SumSignalSquared = SumSignalSquared + FFT(k) * FFT(k) - EstNoisePwr
    ;
        SumNoiseSquared = SumNoiseSquared + EstNoisePwr ;
    end
%
% Calculate noise power after signal bins
%
    for k = UpperSignalBin + 1 : 1 : BWbin
        SumNoiseSquared = SumNoiseSquared + FFT(k) * FFT(k) ;
    end

    SNR = 10 * log10(SumSignalSquared/SumNoiseSquared);
end

```

```

function [ENOB] = ComputeENOB(Vref, A, SNR)
%
% Script to convert peak SNR to ENOB
%
    PeakSNR = SNR + 20*log10(Vref / A) ;
    ENOB = (PeakSNR - 1.76)/ 6.02 ;

end

function[] = ErrorCheck(Fs, N, Fo)
%
% This script will do some error checking
% In order to do SNR calculations we need to be sure that the
% signal bin is large enough
%
    BinWidth = round(Fs / N) ;
    SignalBin = Fo / BinWidth ;
    if (SignalBin <= 10)
        error('Increase input frequency or improve resolution. Aborting.')
    ;
    end

end

function[out] = FFTmagnitude(Waveform)

%
% Using Hodie window to minimize spectral blurring
%
    L = length(Waveform) ;
    coefficients = HodieWindow(L) ;
    WindowedWaveform = Waveform .* coefficients' ;
%
% Return magnitude part of FFT
%
    out = abs(fft(WindowedWaveform)) ;
end

function[] = FreqDomainPlot(FFTwaveform, Vref, Fs, color, locn, total)
%
% This script will plot FFTs associated with delta sigma ADC
%
    L = length(FFTwaveform) ;
    DBFS = 20*log10((2*FFTwaveform)/(L * Vref)) ;
    index = linspace(0, L/2-1, L/2) ;
    deltaF = Fs / L ;
    frequency = deltaF * index ;
    subplot(total, 1, locn) ;
    plot(frequency, DBFS(1 : L/2), color) ;

end

```

```

function [SNR] = TheoreticalSNR(Fs, BW)
%
% Script to compute theoretically predicted signal-noise-ratio
%
    L = 2 ;                                % Modulator order

    OSR = Fs / (2 * BW) ;                  % Compute oversampling ratio

    DR = OSR^(2*L + 1) ;                   % Compute dynamic range
    DR = DR * 1.5 * (2*L + 1) ;
    DR = DR / pi^(2*L) ;
    DR = sqrt(DR) ;

%
% This would be peak SNR (neglecting overload)

    SNR = 20*log10(DR) ;                   % Express in decibels

end

function[] = TimeDomainPlot(Wave, Vref, Fs, color, locn, total)
%
% This script will plot waveforms associated with delta sigma ADC
%
    L = length(Wave) ;
    index = linspace(0, L-1, L) ;
    Ts = 1 / Fs ;
    time = Ts * index ;
    subplot(total, 1, locn) ;
    plot(time, Wave, color) ;

end

```

APPENDIX B

VerilogA Code

```
// VerilogA for thesis, diff_OTA_CMFB, veriloga
`include "constants.vams"
`include "disciplines.vams"

//
// VerilogA behavioral model of an OTA
//
// We model the following:
//
// dominant and parasitic poles
// low-frequency open-loop gain
// OUTPut resistance
// INPut capacitance
// OUTPut capacitance
// OUTPut voltage swing
// slew rate
//

module diff_OTA_CMFB(INM, INP, Ib, OUTM, OUTP, VDD, VSS, ph1, ph2, AGND);

    inout    INP, INM ;    // Differential INPut nodes
    inout    Ib ;          // Bias current node
    inout    OUTP, OUTM ;   // OUTPut node
    inout    VDD, VSS, AGND ; // Supply nodes
    inout    ph1, ph2;

    electrical ph1, ph2;
    electrical INP, INM ;
    electrical Ib ;
    electrical OUTP, OUTM ;
    electrical VDD, VSS, AGND ;

    electrical x ;          // INPut parasitic pole node
    electrical VOH, VOL ;   // OUTPut voltage limits (high and low)

//
// Vos    DC offset of OTA
// gmi    INPut transconductance
// Cin    INPut capacitance
// Cout   OUTPut capacitance
// A0     Low frequency open-loop gain
// Ro     OUTPut resistance
// wp     Parasitic pole frequency (rad / sec)
// Vsat   Saturation voltage of OUTPut devices
// Rb     Used to define the voltage on bias pin
// Is     Diode reverse leakage current
// n      Diode non-ideality factor

    parameter Vos = 0.0 ;
    parameter gmi = 142u ;
    parameter Cin = 0.5p ;
    parameter Cout = 0.05p ;
    parameter A0 = 30e3 ;
    parameter Ro = A0 / gmi ;
```

```

parameter  wp = 6.28 * 10e6 ;
parameter  Rb = 1000 ;
parameter  Vsat= 1.0 ;
parameter  Is = 1e-15 ;
parameter  n = 0.25;

//
// Variable declarations
//
// Iss      Small signal current from voltage controlled current source
// Ibias     Bias current
//
real Iss ;
real Ibias ;

//
// Begin behavioral description of OTA
//
analog begin

    V(VOL) <+ V(VSS) + Vsat ;
    V(VOH) <+ V(VDD) - Vsat ;
//
// Define a voltage on bias pin.
// Determine value of bias current.
//
    Ibias = I(Ib) ;
    V(Ib) <+ Rb * Ibias ;
//
// Model INPut stage and parasitic pole
// Cin is INPut capacitance
//
    I(INP, INM) <+ Cin * ddt(V(INP, INM)) ;
    V(x) <+ laplace_nd(V(INP, INM) - Vos, {1}, {1 , 1/wp}) ;
//
// Model OUTPut stage
// Limit OUTPut current (implements slewing)
//
    Iss = gmi * V(x)/2 ;
    if (Iss > Ibias) Iss = Ibias ;
    if (Iss < -Ibias) Iss = -Ibias ;
//
// Also limit OUTPut voltage. Use diodes.
// We use diode to avoid discontinuities that slow or stop simulator.
// Model OUTPut resistance and OUTPut capacitance.
// Viewed from outside of symbol, current into the pin is considered positive
//
    I(OUTP) <+ -Iss ;
    I(OUTP) <+ V(OUTP) / Ro ;
    I(OUTP) <+ Cout * ddt(V(OUTP)) ;
    I(OUTP) <+ Is * (exp( (V(OUTP) - V(VOH)) / (n * $vt)) - 1) ;
    I(OUTP) <+ -Is * (exp( (V(VOL) - V(OUTP)) / (n * $vt)) - 1) ;

    I(OUTM) <+ Iss ;
    I(OUTM) <+ V(OUTM) / Ro ;
    I(OUTM) <+ Cout * ddt(V(OUTM)) ;

```

```
I(OUTM) <+ Is * (exp( (V(OUTM) - V(VOH)) / (n * $vt)) - 1) ;  
I(OUTM) <+ -Is * (exp( (V(VOL) - V(OUTM)) / (n * $vt) ) - 1) ;  
end  
  
endmodule
```



```

// VerilogA for thesis, Noisy_switch, veriloga

`include "constants.vams"
`include "disciplines.vams"

module Noisy_switch (outp, inp, s, VDD, VSS) ;

    inout    outp, inp ;    // Switch termoutpls
    inout    s ;           // Control voltage
    inout    VDD, VSS ;    // Supply nodes

    electrical outp, inp ;    // Switch termoutpls
    electrical s ;           // Control voltage
    electrical VDD, VSS ;    // Supply nodes

    electrical x ;           // Internal node

//
// Ron is the on resistance of our switch
// Roff is the off resistance
// tpd is the delay associated with switch changing its resistance
// tr is rise time
// tf is fall time
// BW is the bandwidth of our white noise generator
//

    parameter Ron = 10K ;
    parameter Roff = 10G ;
    parameter tpd = 0p ;
    parameter tr = 100p ;
    parameter tf = 100p ;
    parameter BW = 100e6 ;

//
// Rch is the current channel resistance of the switch
// sigma is integrated noise voltage in bandwidth BW
// BW is noise bandwidth of our noise source
// BW should be much larger than other BWs in your circuit
// Don't make BW too large, however, because it slows the simulator
// noise is white noise sample
// sample_period is the period white noise sample is held for
// seed is a variable used by the random number generator
//

    real    Rch ;
    real    Vth ;
    real    sample_period ;
    integer seed ;
    real    sigma ;
    real    noise ;

```

```

//
// Begin behavioral description of "noisy" switch
//
analog begin
  @(initial_step) begin
    Vth = (V(VDD) + V(VSS)) / 2.0 ;
    Rch = Ron;           // Rch = Roff ;
    if (V(s) < Vth) Rch = Roff ;
    seed = $random ;
    sample_period = 1 / (2 * BW) ;
    sigma = sqrt(4 * $vt * 1.602e-19 * Ron * BW) ;
  end

  @(cross(V(s)-Vth,+1,100p)) Rch = Ron ;
  @(cross(V(s)-Vth,-1,100p)) Rch = Roff ;

  @(timer(0, sample_period))begin
    noise = sigma * $dist_normal(seed, 0, 1) ;
  end

  V(x, inp) <+ noise ;
  I(outp, x) <+ V(outp, x) / transition(Rch, tpd, tr, tf) ;

end
endmodule

```

```

// VerilogA for thesis, comparator_SC, veriloga

`include "constants.vams"
`include "disciplines.vams"

module comparator_SC(in_plus, in_minus, clk, outp, outpbar, VDD, VSS);

    inout    in_plus, in_minus, clk, outp, outpbar, VDD, VSS;
    electrical in_plus, in_minus, clk, outp, outpbar, VDD, VSS;

    parameter real output_high = 5;
    parameter real output_low  = 0;
    parameter real Vth = 2.5;
    parameter real slope = 1e20 from (0:inf);
    parameter real trise = 10n from [0:inf);
    parameter real tfall = 10n from [0:inf);
    parameter real tdel = 0 from [0:inf);

    real high_minus_low;
    real high_plus_low;
    real hold, holdbar, Vdiff;
    integer rand_value;

    analog begin
        if(analysis("static")) begin
            @(initial_step) begin
                high_minus_low = 0.5*(output_high - output_low);
                high_plus_low = 0.5*(output_high + output_low);
            end
        end
        @(cross(V(clk)-Vth, +1)) begin

            hold = high_minus_low * tanh(slope*V(in_plus, in_minus)) + high_plus_low;
            holdbar = high_minus_low * tanh(slope*V(in_minus, in_plus)) + high_plus_low;

        end
        V(outp) <+ transition(hold, tdel, trise, tfall);
        V(outpbar) <+ transition(holdbar, tdel, trise, tfall);
    end
endmodule

```

```

// VerilogA for thesis, clock_gen, veriloga

`include "constants.vams"
`include "disciplines.vams"

module clock_gen( phi1, phi2, phi1d,clk,VDD,VSS );

    input      clk,VDD,VSS;
    inout      phi1,phi2,phi1d;
    electrical  phi1,phi2,phi1d;
    electrical  clk,VDD,VSS;

    parameter real Fs = 256e3 from (0:inf);
    parameter real tr = 10n from (0:inf);
    parameter real tf = 10n from (0:inf);
    parameter real td = 10n from (0:inf);

    parameter real high = 5;
    parameter real low = 0;

    parameter real Ts = 1/Fs;
    parameter real phi12 = 10n;
    parameter real phi21 = 10n;
    parameter real phi1delay = 5n;

    integer phi1_val;
    integer phi1d_val;
    integer phi2_val;

    analog begin

        @(initial_step) begin
            phi1_val = 0;
            phi2_val = 0;
        end

        @(timer(0,Ts))          // creating phi1 logic signal
            phi1_val = 1;
        @(timer((Ts/2)-phi12,Ts))
            phi1_val = 0;

        @(timer(0,Ts))          // creating phi2 logic signal
            phi2_val = 0;
        @(timer(Ts/2+phi21,Ts))
            phi2_val = 1;
        @(timer(Ts-phi21-phi12,Ts))
            phi2_val = 0;

        @(timer(0,Ts))          // creating phi1d logic signal
            phi1d_val = 0;
        @(timer(phi1delay,Ts))
            phi1d_val = 1;
    end

```

```
@(timer((Ts/2)-phi12+phi1delay,Ts))
  phi1d_val = 0;

V(phi1) <+ transition ((phi1_val ? high:low),0,tr,tf);
V(phi1d) <+ transition ((phi1d_val ? high:low),0,tr,tf);
V(phi2) <+ transition ((phi2_val ? high:low),0,tr,tf);
end

endmodule
```

```

// VerilogA for thesis, latch, veriloga

`include "constants.vams"
`include "disciplines.vams"

module latch(VDD, VSS, clk, in, out);

    inout VDD;
    electrical VDD;
    inout VSS;
    electrical VSS;
    inout clk;
    electrical clk;
    inout in;
    electrical in;
    inout out;
    electrical out;

    parameter real Vth = 2.5;
    parameter real tr = 10n from (0:inf);
    parameter real tf = 10n from (0:inf);
    parameter real td = 10n from (0:inf);

    real vout;

    analog begin
        @(initial_step) begin
            vout = 2.5;
        end

        if( V(clk) > Vth) begin
            if( V(in) > Vth) vout = 0;
            else vout = 5;
        end
        else vout = vout;

        V(out) <+ transition (vout , 0, tr ,tf);
    end
endmodule

```

```

// VerilogA for thesis, data_recorder, veriloga

`include "constants.vams"
`include "disciplines.vams"

module data_recorder (Vin,clk);

    input      Vin,clk;
    electrical Vin,clk;
    real       Vth;
    real       bit;
    real       MAX ;
    real       MIN ;
    real       clock_time;

    integer     mcd1;

    analog begin
        @(initial_step) begin
            mcd1 = $fopen("/home/tyerra/Desktop/neel/SC_AMS_allreals_realClockGen.txt") ;
            Vth = 2.5;
            MIN = 137e-3;    //Start time
            MAX = 1.26;      //Stop time
            clock_time=0;
        end

        @(final_step) begin
            $fclose(mcd1);
        end

        @(cross(V(clk)-Vth,+1)) begin
            clock_time = $abstime;
            if(V(Vin)>Vth) bit = +1;
            else        bit = -1;

            if((clock_time > MIN)&&(clock_time < MAX))
                $fstrobe(mcd1, "%g", bit);
        end

    end
endmodule

```