DESIGN OF ROBUST AND FLEXIBLE ON-CHIP ANALOG-TO-DIGITAL CONVERSION ARCHITECTURE

A Dissertation
Presented to
The Academic Faculty

by

Daeik D. Kim

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
School of Electrical and Computer Engineering

Georgia Institute of Technology
July 2004

DESIGN OF ROBUST AND FLEXIBLE ON-CHIP ANALOG-TO-DIGITAL CONVERSION ARCHITECTURE

Approved by:

Dr. Martin A. Brooke, Committee Chair

Dr. Paul E. Hasler

Dr. David E. Schimmel

Dr. John D. Cressler

Dr. Paul A. Kohl
(School of Chemical and
Biomolecular Enginerering)

30 July 2004

To the greatest engineer

God,

who designed heart and mind.

ACKNOWLEDGEMENT

My sincerest appreciation is for my advisor Dr. Martin A. Brooke, the institute's best advisor award winner. Not only has he shown the passion for research and instinctive sense for electronic phenomena, but also he has been a good teacher and mentor to me. Also I would like to thank to Dr. Nan M. Jokerst for her support with optoelectronic integration and testing.

I gratefully acknowledge Dr. David E. Schimmel, Dr. Paul E. Hasler, Dr. John D. Cressler, and Dr. Paul A. Kohl for their service on my committee. Their knowledge and guidance were essential to complete my research. Throughout my graduate study, I have been lucky to get a scholarship from Korean government, and I want to give thanks for the Korean people's support.

In addition, I am grateful for my parents and parents-in-law for their love, prayers, and support. Also I would like to thank to my lovely future babies for their generosity to wait to be born. My final thoughts and thanks go to my beautiful and precious wife Minchi, whose love and kindness always cheered and comforted me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This dissertation presents a comprehensive design and analysis framework for system-on-a-chip analog-to-digital conversion design. The design encompasses a broad class of systems, which take advantage of system-on-a-chip complexity. This class is exemplified by an interferometric photodetector array based bio-optoelectronic sensor that is built and tested as part of the reported work.

While there have been many discussions of the technical details of individual analog-to-digital converter (ADC) schemes in the literature, the importance of the analog front-end as a pre-processor for a data converter and the generalized analysis including converter encoding and decoding functions have not previously been investigated thoroughly, and these are key elements in the choice of converter designs for low-noise systems such as bio-optoelectronic sensors.

Frequency domain analog front-end models of ADCs are developed to enable the architectural modeling of ADCs. The proposed models can be used for ADC statistically worst-case performance estimation, with stationary random process assumptions on input signals. These models prove able to reveal the architectural advantages of a specific analog-to-digital converter schemes quantitatively, allowing meaningful comparisons between converter designs.

The modeling of analog-to-digital converters as communication channels and the ADC functional analysis as encoders and decoders are developed. This work shows that analog-to-digital converters can be categorized as either a decoder-centered design or an encoder-centered design. This perspective helps to show the advantages of nonlinear decoding schemes for oversampling noise-shaping data converters, and a new nonlinear decoding algorithm is suggested to explore the optimum solution of the decoding problem.

A case study of decoder-centered and encoder-centered data converter designs is presented by applying the proposed theoretical framework. The robustness and flexibility of the resulting analog-to-digital converters are demonstrated and compared. The electrical and optical sensitivity measurements of a fabricated oversampling noise shaping analog-to-digital converter circuit are provided, and a sensor system-on-a-chip using these ADCs with integrated interferometric waveguides for bio-optoelectronic sensing is demonstrated.

INTRODUCTION

This dissertation describes a comprehensive design and analysis framework for system-on-a-chip analog-to-digital conversion design. The design encompasses a broad class of systems that take advantage of system-on-a-chip complexity. This class is exemplified by an interferometric photodetector array based bio-optoelectronic sensor that is built and tested as part of the reported work.

It is assumed that this class of sensors could use quite large arrays of sensors to sense inputs signals; for example, linear or two-dimensional photodetector arrays or hundreds of parallel sensors ganged to improve selectivity. Furthermore, these types of sensors could require very sensitive front-ends in the presence of possibly wide-band noise from on-board digital signal processing (DSP) or data communication interfaces; for example, weak photodetector signals of a few nano amperes (nA) might be important. In addition, most of these sensors would have relatively low input bandwidths related to physical world time scales, and may have large amounts of signal processing performed on the data after it is recorded. For example, many chemical sensors take seconds to minutes to operate, and there is plenty of time for even slow signal processors to perform computations on the raw measured data. Finally, because of the slow speed of some of the sensor input signals, the sensor would need to complete the sensing task expeditiously, not wasting sensor signal time. This will lead to a need for some parallel hardware at

each sensor interface, which in turn leads to a constraint on the size of the direct sensor interface circuitry.

This last constraint, of expeditious sensor operation, will be discussed in Chapter 1. It will be shown that for the case where there are many parallel weak sensor input signals placing some of the sensor analog-to-digital conversion (ADC) interface in parallel at each sensor location is very desirable, from sensing time standpoint, and from a sensitivity standpoint.

Two aspects of ADC modeling, linear frequency domain analysis and nonlinear time domain analysis, will be developed to provide theoretical analysis tools for comparison purposes. In Chapter 2, frequency domain modeling of ADC analog front-ends will be examined and it will be used to produce an architectural comparison of ADCs. The analysis will emphasize the importance of data converter pre-processing as a limiting factor in SoC ADC design.

Chapter 3 is an inquiry into the time domain behavior of the ADC process that enables a performance comparison using a newly discovered $\Delta\Sigma$ decoding algorithm. The trade-off between performance and complexity explored in this chapter is useful to determine what would be an optimal combination of ADC encoding and decoding schemes.

In this chapter, it will be shown that there are two separately developed approaches to design data converters, an encoder-centered method and a decoder-centered method. The encoder-centered method has its emphasis on effective ways to

transform inputs to preserve important characteristics and to enhance conversion efficiency. The intermediate results from the encoder-centered approach are not simple to understand as they are not in formats resembling the desired output, and, in fact, this approach requires complex algorithms to produce the desired output. The decoder-centered method takes a top-down approach to solve the data conversion problem. This method concentrates on the final result of conversion and intermediate processing outcomes are therefore similar to the final form of the data.

A good example of the class of encoding-centered converters is the well-known delta-sigma (ΔΣ) encoder and decoder (CODEC), whereas traditional Nyquist sampling conversion schemes, such as slope-integrating ADCs, iterative algorithmic ADCs, and flash ADCs, are regarded as decoder-centered methods. The comparison of two types of ADCs will lead to several important modeling and analysis techniques, which will be useful as design criteria.

In Chapter 4, ΔΣ ADCs and single-slope ADCs are taken as examples of decoder- and encoder-centered data converters for a comparison study of sensor readout system ADCs. Designs of these two ADCs are proposed and compared through the provided theories and practical considerations. ADC robustness and flexibility are examined with model variations, temperature dependency, clock jitter, etc. A fabricated SoC sensor ADC is demonstrated and its performance is measured.

The entire thesis provides a qualitative and quantitative analysis that connects theory and application. It is hoped that this work is not only valuable as an ADC comparison and evaluation methodology but also as an ADC design framework.

Chapter 1

BACKGROUND

In this chapter it is determined that oversampling ΔΣ ADC and integrating ADC are the best candidates for system-on-a-chip (SoC) sensor ADC applications. These two types of converters are then compared in both the frequency and time domain (which is equivalent to a comparison of Nyquist sampling ADCs and ΔΣ oversampling ADCs). It is shown that under most circumstances the ΔΣ oversampling ADCs provide a sensitivity advantage of more than 20 dB and do so with much less complex front-end hardware.

This chapter begins with a description of a bio-optoelectronic sensor system application (BOSS) and from this we derive general sensor SoC ADC requirements (sections 1.1 and 1.2). An argument that a few ADC designs are more appropriate than others for sensor system ADCs designs is given in section 1.3. The fact that many textbooks present useful and detailed design and analysis tools for a few selected SoC ADCs, but fail to provide a comprehensive comparison between the two important classes of ADCs considered here, Nyquist ADCs and ΔΣ ADCs, is stated in section 1.4. Also several published ADC comparison studies are summarized in the same section.

1.1 Bio-Optoelectronic Sensor System

The bio-optoelectronic sensor system (BOSS) is a multi-investigator project that provided the motivation for this study. The BOSS implements an array of real-time SoC sensors, which detect chemical and biological agents in various forms [1]. The basic system concept is shown in Figure 1. A product prototype of this system has been fabricated by other researchers by assembling interferometric waveguides and other commercial components, such as diode lasers, a photo diode detector array, and an external signal processing system. The entire prototype package has size of 2.5 by 3.0 by 6.5 inches. The system can detect chemicals such as benzene, chloroform, methylene chloride, toluene, etc. Also it can detect bio-molecules such as IgG/anti-IgG and salmonella. The sensitivity of the system is demonstrated to be up to 10 parts-per-billion (ppb), dependent on the target agents [2-6].

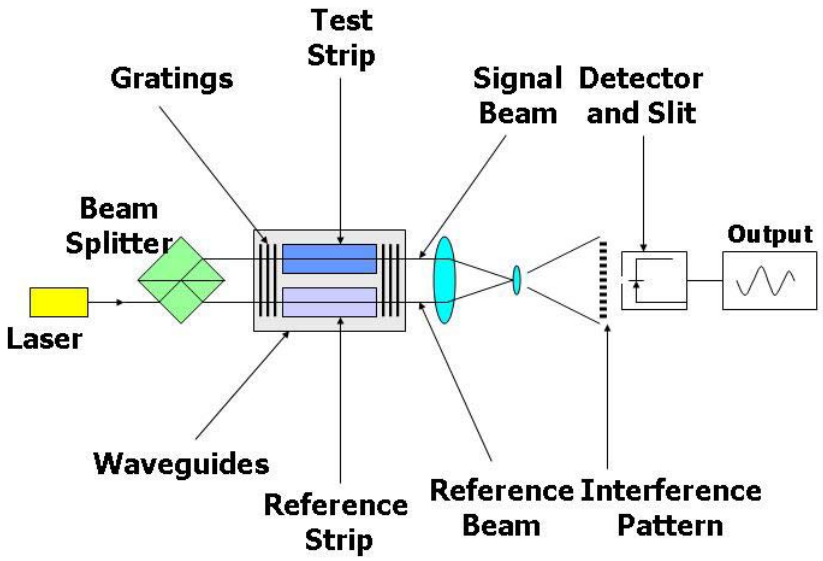

Figure 1: Bio-optoelectronic sensor system block diagram

The goal of the BOSS project is to integrate all optical and electrical components on a stand-alone silicon complementary metal-oxide-semiconductor (Si-CMOS) chip, which produces an output equivalent to the prototype. This thesis concentrates on the design of the mixed-signal read-out component, or analog-to-digital conversion architecture of the BOSS SoC.

1.2 ADC Requirements

In this section we will show that for sensor system-on-a-chip applications, small, reliable, low-power, and sensitive ADCs are required. The requirements on the size of the

system are strict since the integration of optical components takes a large chip area. Also it is undesirable to have the uneven geometry of the Si-CMOS circuitry beneath the optical systems. Though there are several cases of successful implementation of optical components over a Si-CMOS IC [7], the optical components in the BOSS sensor are interferometric waveguides, which are very sensitive to fluctuations in the refraction index of material around the waveguide and the smoothness of the waveguide surface. The integration sites for waveguide, interferometer, and laser will occupy the principal area of the chip, as will be seen later in the integrated prototype in Chapter 4. The ADC and the post-signal processing system area will be limited to a small portion of the chip due to these optical component restrictions and the available low-cost chip sizes provided by the MOSIS service [8]. Also, the ADC should be located as near as possible to the sensing components such as photo detectors to avoid signal degradation.

In using the results of the analysis in Chapter 2 it can be shown that in sensor system-on-a-chip applications, where there is considerable background digital noise or environmental noise, directly connected oversampling ADCs have a distinct advantage over Nyquist rate ADCs that integrate the input signal between samples. This is a new result, as previously (particularly in imaging applications) it has been believed by many that integration of signal between samples (for example, on a CCD) was the lowest noise approach.

The specific numbers from the analysis are quite dramatic on this point. To obtain these numbers, two input signal types were considered, first a band-limited but spectrally white input was considered, and then the same input but with first order band-

pass filtered inputs above the Nyquist rate added. The first input type is not very severe, mimicking typical complex real world inputs, the second input type is much more severe, in that the unwanted signals above the Nyquist rate will cause aliasing problems in many converter types; however, in system-on-a-chip applications, it is quite likely that large signals due to the digital processing, power supply, or other signal processing circuitry on the system will appear on the input. This model also represents the case where there is uncertainty regarding the spectral content of the input, which often occurs in the real world when systems operate autonomously in uncertain environments. In this case the choice of converter conversion rate (the Nyquist rate) could be made too low, leaving some signal power above the Nyquist rate.

In all of the simulations in Chapter 2, a background circuit noise level was chosen so that the in-band noise of a single stage was 16-bits (98.1 dB) below the signal power. This is actually more noise than we have measured in our CMOS chips described in Chapter 4, and provides a measure of the effectiveness of each design considered, in that the closer the designs signal to noise is to 98.1 dB, the better the design is performing.

Table 1 presents the results, with the details being presented in Chapter 2. For the less severe band-limited case, the oversampled converter provides a 20 dB advantage over the integrating Nyquist converter (approximately 3-bits more resolution) due entirely to out-of-band circuit noise, which is integrated by the Nyquest converter and removed by the oversampling converter. However, in the case of the more severe input with unwanted pink signal spectrum above the Nyquist rate (the system-on-a-chip type

input), the oversampling converter has an 87.7 dB (more than 14-bits) advantage, due to the severe aliasing that occurs in the Nyquist rate converter.

Table 1: Comparison of the SNR of integrating Nyquist sampling and oversampling front-end models

Front-end model / Input signal model	Nyquist sampling	Oversampling	Oversampling advantage
Input signal is white up to Nyquist	73.5 dB	93.5 dB	20.0 dB
Input signal is white up to Nyquist and first-order cutoff (pink) above Nyquist	6.5 dB	94.2 dB	87.7 dB

The overall conclusion of this work is that for the class of applications we are considering, directly connected oversampling converters are very advantageous. However, oversampled converters must operate directly connected to the input during the entire sensing time, to be able to measure the out-of-band signals so they can be filtered out. Thus, if many sensors are used and sensitivity is the primary concern, parallel oversampled converters are required to ensure the longest possible sensor connection time to minimize the in-band noise. If a single oversampled converter were used, then for a given sensor output rate the converter would only connect to a given sensor for a small portion of the overall sensor sample time. This would raise the Nyquist rate of the

converter and increase the amount of out-of-band noise sampled by the number of sensors used. For example, if there are 1000 parallel sensors used to make a measurement, then using a single oversampled converter to make a measurement will produce 1000 times more noise than using 1000 parallel oversampled converters, because each sensor will be sampled 1000 times slower in the parallel case. With integrating converters, it is also routine to perform at least the integration in parallel (e.g., CCD imagers). Thus, it will be assumed that for the class of applications to be considered here, only parallel oversampling converters make sense. This may not be true for applications where sensitivity is not an issue.

This places a very severe size constraint on the digitizing portion of the oversampled converters. Fortunately, there are very compact first-order oversampling digitizers that use ΔΣ encoding to shape quantization noise and thus achieve high numbers of bits of resolution with essentially a one-bit architecture. These first-order ΔΣ converters and the very similar single-slope Nyquist ADCs will be chosen eventually as the primary converters to be compared in section 1.3. However, in the later parts of this section, we will consider many other types of converters and show that they are not appropriate for a sensor of the same class as the BOSS sensor.

The sensitivity of the BOSS sensors is dominated by the resolution of ADC and a minimum of 12-bits resolution has been found to be necessary to get a meaningful output when the sensor is implemented from discrete components [9, 10]. It is probable that compromises in integrating the sensor will raise the sensitivity requirement even higher

to 14-bits or more, so we will assume the BOSS sensor class requires very sensitive converters.

The speed of the integrated circuit is not an overriding issue with this application since the reaction of chemical and biological agents occurs slowly compared to available low-cost CMOS signal processing speeds. This slow conversion speed also means that we assume that considerable off-chip digital signal processing is quite feasible on the digital chip output. We expect this would be performed just prior to when the sensor data is used in an application. The lower-bound of the sampling rate is in the order of 0.1-1 Hz [3, 4].

1.3 ADC Family Selection for Sensor SoC Application

The BOSS sensor can now be categorized as one of a class of ADC applications that restrict the digitizer size and require high sensitivity, but do not focus on output data rate or power consumption issues. These attributes, small size of digitizer and high sensitivity at the expense of higher output data rates and higher power, will restrict the ADC families we need to investigate to find the best converter designs. We will consider flash, algorithmic, slope-integrating, and ΔΣ ADCs in this comparison.

Flash type ADCs consist of a very large bank of thermometer coded comparators and a decoder [11-17]. Most flash ADCs concentrate on high-speed conversion, sacrificing power consumption, sensitivity, and circuit area for speed [18, 19]. High

sensitivity flash type converters are rare, and the available resolution is below 10-bits [20]. They provide very fast, simple, and straightforward operation, but they are not suitable for BOSS type sensor SoC integration since they take up too much circuit area.

Algorithmic ADCs refer to a class of ADCs that perform a binary search to find the conversion result [11-17]. There are many variations on this architecture utilizing pipelined stages of differing resolution; however, most algorithmic ADCs have 12-bits or lower resolution, which make them unattractive for sensor applications. In addition the circuit size becomes large when passive components and the number of pipelined stages are optimized for high sensitivity (see Table 9 in Appendix A). Thus, although some very small algorithmic converters have been built, their accuracy is poor, making this family unsuitable for BOSS-style sensor SoC applications.

Single-slope serial, dual-slope serial, and ΔΣ ADCs are all capable of very compact digitizers that achieve high resolutions [11-17]. Usually the first two schemes are implemented as Nyquist sampling converters, while the ΔΣ converters use oversampling since ΔΣ noise shaping provides dramatic enhancement of accuracy when combined with oversampling (1-bit ΔΣ sampling can achieve 18 – 24 bits of accuracy with oversampling [21]). Another major difference is that ΔΣ converters require a digital decoder or filter to retrieve the measured digital result, whereas slope-integrating converters produce a final digital result with minimal digital signal processing. For the BOSS type of application where data rate off chip is not considered an issue and abundant digital signal processing is assumed to be available off chip, it is not possible to discriminate between these two ADC families on the basis of the external digital

processing required. Thus we are forced to compare these two fundamentally different ADC technologies to determine which can achieve the best sensitivity.

General descriptions and applications of single-slope ADCs, dual-slope ADCs, and iterative ADCs are presented in Appendix A. Appendix B provides discussions on ΔΣ oversampling ADCs as encoders and decoders.

1.4 SoC ADC Comparison Studies

Many specialized technical areas, such as signal processing, communication, control, information theory, radio frequency circuit design, analog circuit design, digital circuit design, and VLSI system design, are involved in ΔΣ modulator design and analysis. This means that the study of the converter itself requires a wide range of skills. As a result, ΔΣ converters have not been fully understood in many aspects, even though there are numerous successful industrial and academic implementations. New architectures, variations, and implementations are reported frequently in the literature.

The concept of the ΔΣ modulator first appeared as a patent in 1960, filed by Cutler in 1954 [22]. Although the concept was attractive, it was not practical for many years until high performance digital signal processing hardware technology for decoder design became possible with the development of fine-line VLSI Si-CMOS processes in the 1980s. In the interim, decoder-centered ADCs that do not need complex decoding hardware became popular choices for ADC implementations. Today oversampling noise

shaping ADCs are widely used for slower speed applications such as audio, and high-speed ADC designs employ Nyquist rate decoder-centered schemes.

There are many publications in books and journals dealing with ΔΣ ADCs. Table 2 shows recent books on data converter design and analysis.

Table 2: Recent books contain data converter topics

Ref.	Title	Author	Year
[15]	CMOS Integrated Analog-to-Digital and Digital-to-Analog Converters	Plassche	2003
[11]	CMOS Analog Circuit Design	Allen	2002
[13]	CMOS Mixed-Signal Circuit Design	Baker	2002
[23]	Mixed-Signal Systems	Handkiewicz	2002
[16]	An Introduction to Mixed-Signal IC Test and Measurement	Burns	2001
[24]	Design of Analog CMOS Integrated Circuits	Razavi	2001
[25]	CMOS Data Converters for Communications	Gustavsson	2000
[12]	CMOS Circuit Design, Layout, and Simulation	Baker	1998
[17]	Analog Integrated Circuit Design	Johns	1997
[26]	Delta-Sigma Data Converters	Norsworthy	1997
[27]	Principles of Data Conversion System Design	Razavi	1995
[28]	Sigma Delta Modulators	Hein	1993
[29]	Oversampling Delta-Sigma Data Converters	Candy	1992
[14]	High-Speed Analog-to-Digital Conversion	Demler	1991
[30]	Analog MOS Integrated Circuits for Signal Processing	Gregorian	1986

While ΔΣ modulators can be considered as an extension of classic converters functionally [11-17], they are regarded as a totally different system when a detailed nonlinear analysis is required [26, 28, 29]. The design of ΔΣ ADCs is mostly conducted using a linear approximation in the frequency domain. The converter architecture yields to linear methods when the quantizer in the loop is assumed to approximate to a white noise source [11-17, 23-25, 27, 30]. This approximation is adequate, provided that the effective resolution of the converter is high and the input is random. This linearization approach enables complex higher-order design of ΔΣ ADCs; however, it does not match experiments and measurements in several aspects as discussed in Appendix B.

With the linearized quantization noise model, the main advantage of ΔΣ oversampling ADCs over Nyquist ADCs is a simplified analog front-end. Linear ΔΣ analysis articles and books listed in Table 2 do not provide a complete analysis of the effects of analog front-ends. A quantitative analysis on how much the oversampling ADC analog front-end is better or worse than Nyquist ADC is necessary for converter choice and design, particularly in the BOSS-like system-on-a-chip sensor case, where the small size of the converter is an issue. Despite its wide use, linear ΔΣ ADC analysis tends to compute worst-case ΔΣ ADC performance, and it is shown that full nonlinear modeling is essential for a fair comparison.

To understand both desirable and undesirable nonlinear phenomena in ΔΣ ADCs, a nonlinear modeling technique is essential [26, 28, 29]. With nonlinear models, known

initial states of a ΔΣ ADC and a deterministic signal input are necessary to obtain an accurate analysis. General analysis tools for unknown internal states and unknown input signals are still being developed [28, 31]. Those techniques involve complex linear algebra, projection, and state-space modeling as discussed in Appendix B. However, it has been shown that better ADC performance can be obtained for a given ΔΣ ADC design by adopting nonlinear modeling of ΔΣ for decoder design [32]. In general optimum solutions to ΔΣ oversampled ADC decoder design have yet to be found. Most of the nonlinear algorithms currently proposed are limited to special cases, such as a specific noise shaping filter order and architecture. To date, the only general solutions are based on linear analysis.

There are more than 2,000 ΔΣ modulator related articles currently in the published literature [33]. Most of the articles and texts concentrate on details of specific conversion schemes or describe ΔΣ ADCs with simple linear models. While it is well known that oversampling noise shaping ADCs have achieved better performance for certain applications than Nyquist ADCs, the reasons why are not well elucidated in any published articles or texts. Furthermore, ADC analysis and comparison for sensor SoC integration applications are not found in any references. There are some articles that discuss comparisons of Nyquist rate ADCs and ΔΣ ADCs, and Table 3 summarizes these articles.

Table 3: Articles on ΔΣ data converter comparison

Ref.	Title	Auhtor	Year
[34]	Comparison between PWM and sigma-delta modulation in a power factor correction system	Dallago	2002
[35]	A sigma-delta modulator as an A/D converter	Plassche	1978
[36]	Comparison of vector sigma-delta modulation and space-vector PWM	Nieznanski	2000

In [34], the performance of sigma-delta modulator (ΣΔM) and pulse-width modulator (PWM) in single-phase AC / DC boost power factor correction (PFC) system is compared. Sigma-delta modulators have irregular switching frequency, and they can avoid concentration of emissions at discrete frequencies, which is a common problem in pulse-width modulation. Also sigma-delta modulators provide high noise rejection and high linearity between the modulation signal and duty cycle through internal integrator when compared to PWM schemes. With a measured line current spectrum, a sigma-delta modulator provided much less distortion than a pulse-width modulator. Also sigma-delta modulator showed lower conducted emission at the average switching frequency than pulse-width modulation. Only the encoder or modulator part of the sigma-delta modulation is considered in the study.

A first-order ΔΣ ADC is demonstrated in [35]. It uses bipolar technology for the analog circuits and MOS technology for the digital circuits. Bipolar technology was used for the sigma-delta modulator, since the technology has advantages for input and reference source circuits over MOS technology. The design includes an auto-zero circuit, voltage-to-current converter, multi-input data acquisition system, and digital-to-analog converter. The measured resolution is more than 16-bits with 1.8×2.9 mm^2 and 2×3.2 mm^2 chip and 27 mW power consumption. This paper shows that low-speed but high-accuracy ADCs can be implemented with sigma-delta modulator that outperforms integrating ADCs such as the dual-slope ADCs.

The third article in Table 3 shows that a modified vector sigma-delta modulator (VΣΔM) can replace a space-vector pulse-width modulation in the inverter control applications. Vector sigma-delta modulators are regarded as a variation of pulse-density modulation (PDM), and the original idea is modified to fit within power electronics applications by eliminating direct polarity reversal and by reducing inverter switch loss. Vector sigma-delta modulator performance is comparable or better than space-vector pulse-width modulation in aggregate quality, total harmonic distortion, switching loss factor, and switching frequency characteristics. Vector sigma-delta modulators are shown to provide smooth fine-tuning by parameter adjustment, freedom from minimum pulse-width problems, and relaxed hardware requirements since a timer is not required. The article concentrates on the vector ΔΣ encoder since decoding is not required with inverter applications [36].

These studies present specific design comparisons for particular (and different) applications rather than a general comparison.

Chapter 2

ARCHITECTURAL COMPARISON OF SYSTEM-ON-A-CHIP ADC WITH STATISTICALLY DEFINED INPUT

This chapter provides an architectural performance comparison between Nyquist rate sampling ADCs and oversampling ADCs. The comparison is performed by making a linear approximation to the quantization stage of the converters and then using the frequency domain analysis to compute the average statistical performance of the ADCs to band-limited random inputs and additive broadband noise.

All the function blocks that the input signal goes through until it gets to the actual data converter are defined as the analog front-end of ADC system, as shown in Figure 2. Many ADC system implementations only include the ADC itself, ignoring auxiliary but critical front-end functional blocks [37-45]. The analog front-end plays an important role in conditioning the input signal and transferring it to an ADC. It can be a limiting factor for ADC noise and bandwidth performance. Models and analysis for analog front-ends are discussed in Appendix C.

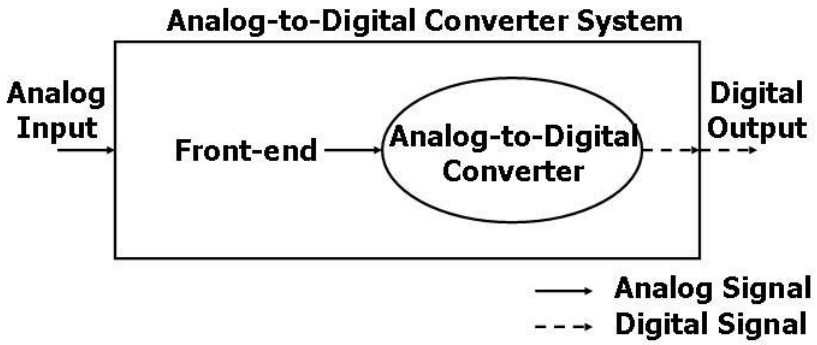

Figure 2: Definition of front-end and ADC in ADC system

We are assuming a relatively wide-band input signal that is typically more severe than most real-world signals. This is because real-world signals tend to only have signal content well below the Nyquist rate. However, in the system-on-a-chip application we are considering, input frequency content at or above the Nyquist rate is very likely due to interference from the integrated digital and other components on the system. Thus although stand-alone converters might perform better than the predicted signal-to-noise ratio (SNR) in this analysis, system-on-a-chip converters will probably perform quite close to the simulated results, and would not be adequately modeled with only low-frequency inputs.

Another issue with frequency domain analysis is that the ΔΣ ADC has nonlinear internal states. However, the nonlinearity of the ΔΣ ADC produces effects mainly in the quantization noise band that are filtered out, rather than in the low-frequency signal band. Thus the effect of neglecting nonlinearity in this analysis is small.

2.1 Architectural Comparison

In the following subsections, an architectural comparison of ADCs will be presented by utilizing the analog front-end models and input signal models provided in Appendix C. All the analog front-end components are modeled in frequency domain and cascaded according to the front-end models shown in Figure 93 and Figure 94 in Appendix C. There are four possible front-end configurations to be compared as summarized in Table 4. Most of these set-ups are applied to both Nyquist sampling and oversampling ADCs.

Table 4: Analog front-end configurations; o = use this element × = not use the element

Configuration	Buffer	LPF	SH	ADC
Traditional ADC	o	o	o	o
Direct sample-and-hold ADC	o	×	o	o
Direct ADC	o	×	×	o
Converter constraint ADC	×	×	×	o

The input signal power P_0 is normalized to 1 and a flat spectrum noise power N_0 of 1×10^{-10} per front-end stage is chosen. The noise power is selected to produce 16-bits equivalent output when only one input stage is used and nothing else affects the SNR. Also the selected noise power turns out to be a good estimation of a real circuits' input noise, as demonstrated in Chapter 4. Each stage in the converter is assumed to add the same amount of noise power $N_t = N_0$ to the signal. This is analogous to each component using similar circuits in the signal path, each with similarly-sized transistors having similar current flow. The underlying assumption here is that ADC architectural differences will not allow a significant difference in the amount of noise reduction achieved through design optimization. The sample-and-hold capacitor is assumed to be C = 0.1 nF, which will generate kT/C noise equal to about a half of N_t at T = 300 K.

The output signal and noise spectrum are obtained with numerical analysis to produce SNR for each configuration. The SNR is obtained as a ratio of the signal power in the desired band to all the noise power in dB. For oversampling, it is assumed that a first-order anti-alias LPF provides enough attenuation to avoid signal aliasing when the OSR is high enough (usually the OSR = 256 - 1024), and that a post digital signal processing algorithm removes all remaining out-of-band noise through multirate signal processing. The simulations are performed in frequency domain with frequency step 2×10^{-3} and frequency range $[0, 2\times10^{2}]$. The signal and noise powers are obtained through the extended and closed trapezoidal rule integration [46].

2.2 Analytic Comparison for Traditional Front-end

2.2.1 Band-limited Input Signal Model A

Using the strictly band-limited signal model A ($X_{in,A}(f)$ given in the equation (34) in Appendix C), the relationship of signal bandwidth to SNR can be explored for the traditional ADC with buffer, anti-aliasing low-pass filter, and sample-and-hold, as modeled in Figure 93 (Appendix C). A desired input signal bandwidth f_M is normalized 1 for all the following simulations. All filter cut-off frequencies are set to the normalized frequency 1. The resulting SNR is plotted in Figure 3. As long as the input signal is band-limited to the designed signal band $f_M = 1$, the order of the filter does not have much effect on the signal-to-noise ratio. Slight differences with different filter order come from

the different attenuation of input signal power. The picked filter cut-off frequency 1 provides the almost optimal signal-to-noise ratio with the band-limited input, whose bandwidth is less than 1. When the input signal bandwidth is larger than the designed signal band f_M, the filter order can make significant improvement in signal-to-noise ratio by changing filter cut-off frequency. A three-dimensional plot of SNR as a function of input signal bandwidth and filter cut-off frequency for sixth-order anti-alias LPF, shown in Figure 4, demonstrates that less change in SNR occurs as signal bandwidth changes when an optimal cut-off frequency is used rather than the normalized frequency 1.

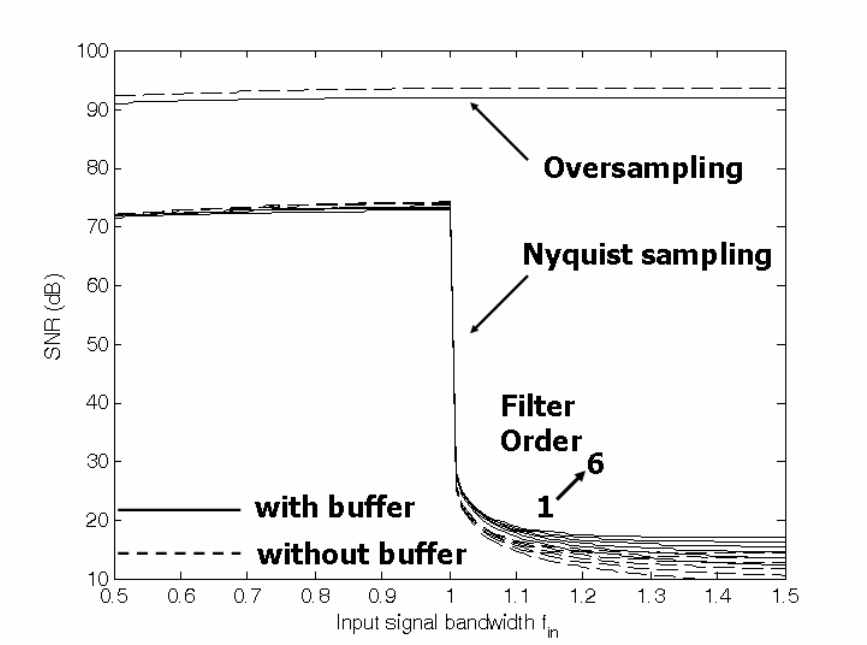

Figure 3: Traditional ADC front-end SNR performance with input signal model A versus input signal bandwidth

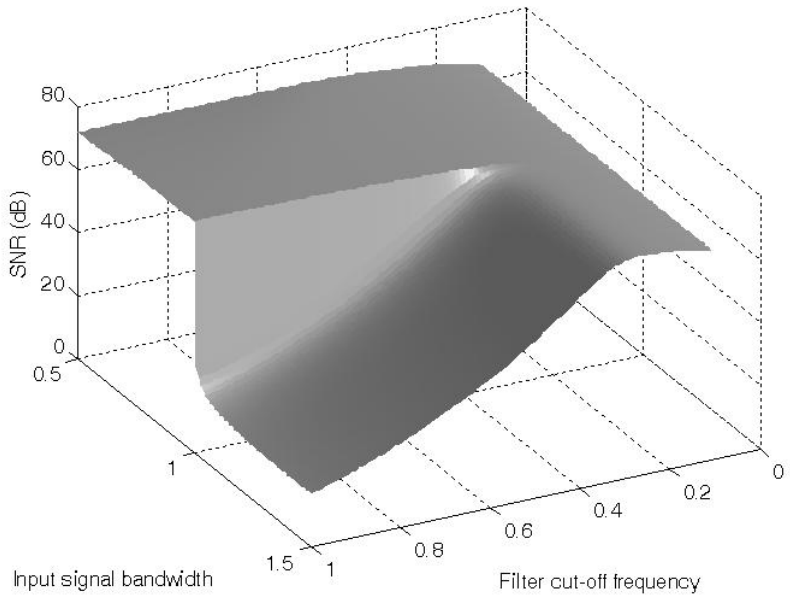

Figure 4: SNR plot as a function of signal bandwidth and filter cut-off frequency for the 6th order LPF case (Notice that the SNR is essentially unchanged with signal bandwidth if the optimal filter cutoff for wide-band input is used.)

The results in Figure 3 show that although band-limiting the input improves the SNR of the Nyquist converters, the improvement is insufficient to overcome the performance advantage of the oversampling converters (which also improve with signal band limiting).

2.2.2 Flat Input Signal Model B

With the flat spectrum input signal model B ($X_{in,B}(f)$ given in the equation (35) in Appendix C), the traditional analog front-end given in Figure 93 in Appendix C produces

the SNR shown in Figure 5. The Nyquist sampling case shows that the presence of a first-order buffer only increases the SNR slightly. An optimal cut-off frequency that gives the maximum SNR for each LPF can be obtained from the figure. This optimum frequency is quite low compared to the sampling rate. There is an apparent reduction in the improvement of SNR with increased order of LPF, so that filter orders greater than second-order add little to the converters' performance.

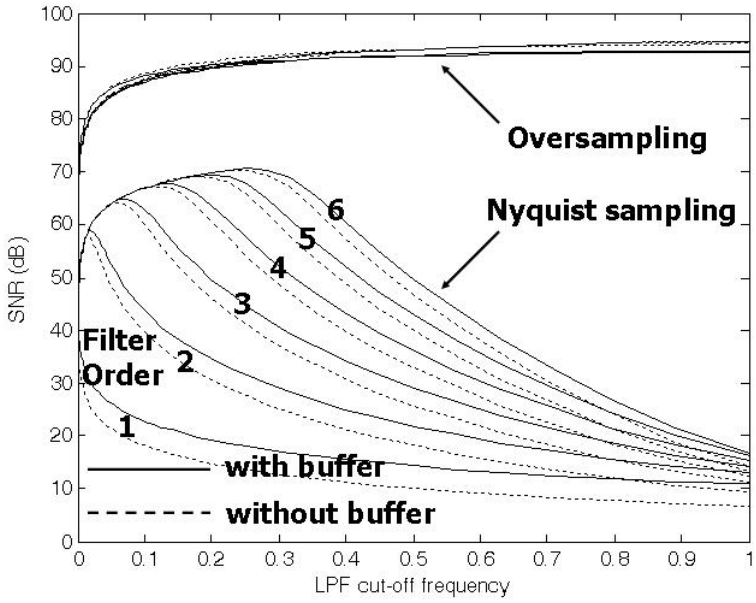

Figure 5: Traditional ADC front-end SNR performance with input signal model B

The oversampling case shows that the order of the LPF does not affect the performance; this suggests that no filter is needed with an oversampling converter. In fact if a filter is used with an oversampling converter, then the cut-off frequency of the LPF

should be outside of the signal band to maximize the SNR. Therefore, a buffer as a low-order LPF will be enough for oversampling to avoid additional noise that might come from high-order filters.

It is clear from Figure 5 that the oversampling converter outperforms the best Nyquist converter by approximately 20 dB, and outperforms the Nyquist converter with second-order filter by 30 dB. Coupled with the elimination of the need for an anti-aliasing filter, this performance advantage makes the oversampling converter a clear favorite for applications where sensitivity is a priority.

In summary the oversampling converters maintain their 20 dB advantage over Nyquist converters for most signal bandwidths, and actually achieve nearly 40 dB improvement over the second order LPF Nyquist case when wide-band input signals are applied.

2.3 Analytic Comparison for Analog Front-end with Direct Sample-and-Hold

The analog front-end configuration with direct sample-and-hold consists of a buffer, sample-and-hold, and ADC only, as shown in Figure 94 (Appendix C). As discussed in the classic configuration case in the previous section, Nyquist sampling with a higher-order anti-alias LPF cannot achieve anywhere near the performance of the oversampling case. In the direct sample-and-hold front-end, the absence of an anti-alias

LPF causes more signal aliasing, so Nyquist converters suffer even more. One solution to this is a cascade of similar buffer stages with cut-off frequency around f_M.

2.3.1 Band-limited Input Signal Model A

The band-limited input signal $X_{in,A}(f)$ is applied to the configuration and the output SNR is plotted in Figure 6. Since the buffer is not designed for the rejection of out-of-band signal, SNR degrades quickly as the input signal bandwidth goes beyond 1. In summary, the band-limited input helps the Nyquist converters considerably when signal content above f_M is eliminated; however, the Nyquist converters still under perform the oversampled converters by 20 dB even when the signal is band limited.

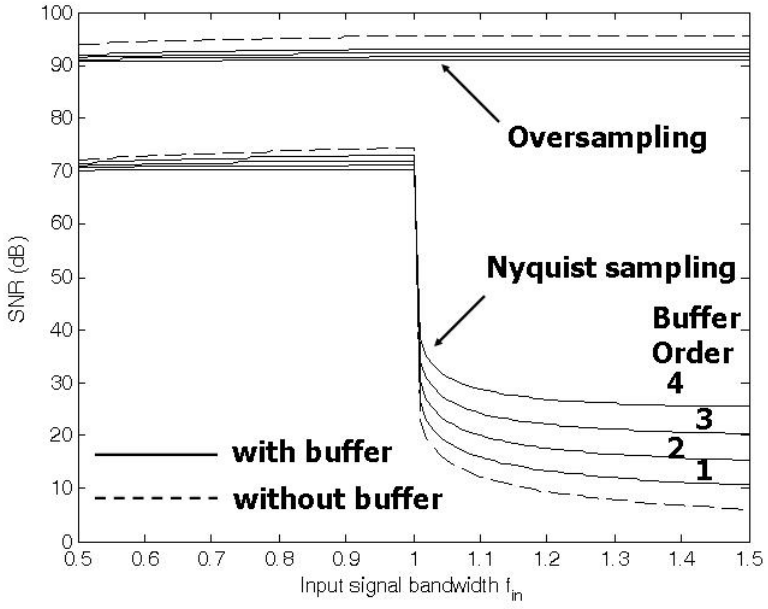

Figure 6: Direct sample-and-hold ADC front-end SNR performance with input signal model A versus input signal bandwidth

2.3.2 Flat Input Signal Model B

Figure 7 shows the resulting SNR with the wideband input signal model B, using a buffer cut-off frequency fixed at $0.7f_M$. Clearly a higher-order buffer can simulate the effect of anti-alias LPF when it is designed as a filter with low cut-off frequency. Usually a buffer is designed to provide a gain-bandwidth around $0.6f_M$ - f_M to ensure high-speed operation [47]. In the oversampling case, the order of buffer does not contribute much to the performance, and the buffer itself is not necessary since sample-and-hold works as a LPF with a proper capacitance value. In summary, the oversampled converter outperforms the Nyquist converters by around 30 dB, even with the buffer-based filters.

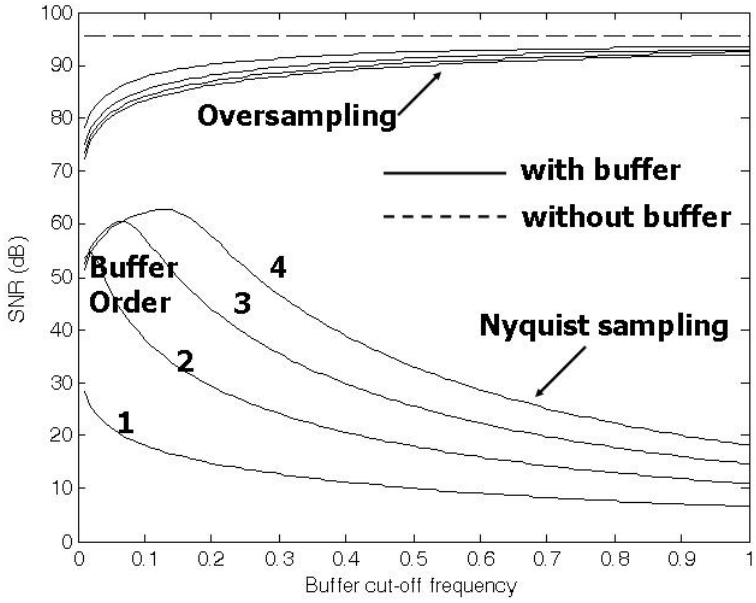

Figure 7: Direct sample-and-hold ADC front-end SNR performance with input signal model B versus buffer cut-off frequency

2.3.3 Analog Front-end with Direct Conversion

Many contemporary ADCs do not have explicit anti-alias LPF and sample-and-hold. This direct conversion configuration consists of a buffer and an ADC as shown in Figure 94 in Appendix C.

The band-limited signal model A, $X_{in,A}(f)$ is applied to input with a fixed buffer cut-off frequency $0.7f_M$ and the output SNR is given in Figure 8. The figure argues that there is no need for any FE function block to get the maximum available SNR as long as the input signal is strictly band-limited to f_M. The strict band-limitedness assumption on input signal is rather naive, and a filtering function is unavoidable in practice. Any non-

ideal filter attenuates some portion of in-band signal power and adds noise to the signal. The available SNR performance of Nyquist sampling is better with direct conversion than with direct sample-and-hold conversion for the input signal band less than f_M, and it is worse for the signal band greater than f_M. As a matter of fact, the absence of sample-and-hold would cause the Nyquist ADC output to become unreliable since a varying signal applied directly to an ADC causes the conversion algorithm to fail in most cases (see Appendix C.6 for an example). The actual ADC output error without a sample-and-hold depends on the conversion algorithm. The oversampling performance is better than in the direct sample-and-hold case since there are fewer blocks that add noise than direct sample-and-hold.

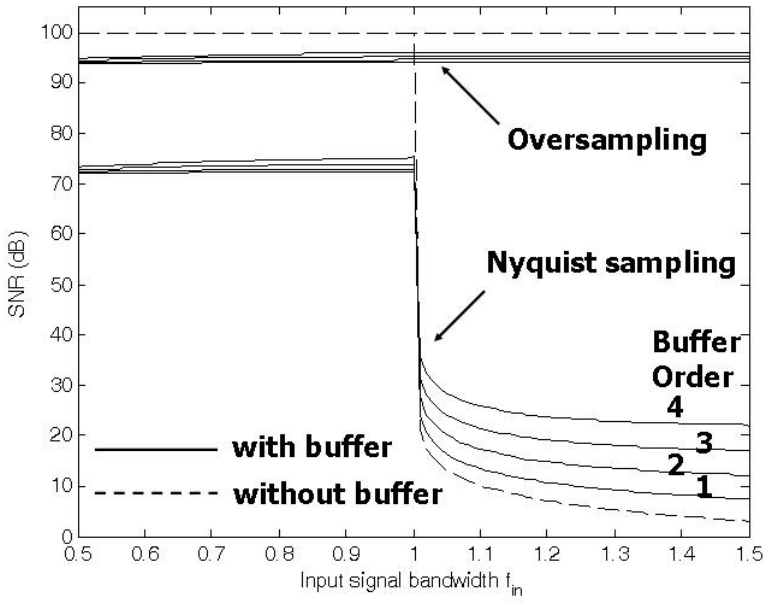

Figure 8: Direct sample-and-hold ADC front-end SNR performance with input signal model A versus input signal bandwidth

When the input signal model is $X_{in,B}(f)$, the buffer is essential since there is no way to attenuate out-of-band signal power. Without a filter function block to attenuate out-of-band signal, there is a total signal aliasing, and the output SNR will be $-\infty$. The output SNR is given in Figure 9. Nyquist sampling performance is worse than the direct sample-and-hold case due to the absence of sample-and-hold as a low-pass filter. Oversampling configuration performance is better since less noise power is added in the analog front-end.

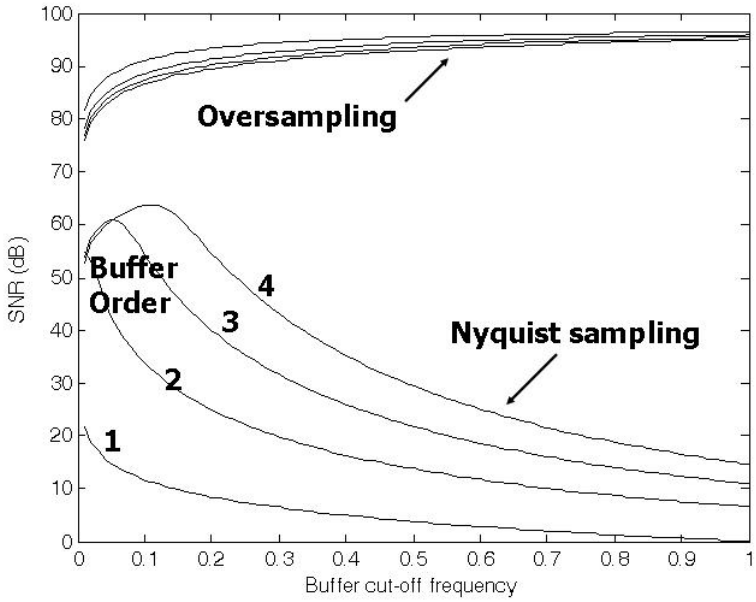

Figure 9: Direct sample-and-hold ADC front-end SNR performance with input signal model B versus buffer cut-off frequency

2.3.4 Summary

Of all the architectures explored so far, the oversampling converter with direct connection to the signal is by far the best alternative for high sensitivity applications. When the input can be assumed band-limited, Nyquist converters with just a sample-and-hold input will perform to within 20 dB of the oversampling converter and, as such, might be desirable if there are not the resources necessary for the digital signal processing necessary for operation of the oversampling converters.

2.4 Converter-constrained ΔΣ Conversion

It is discussed in Appendix C that a first-order ΔΣ ADC has an extended filter transfer characteristic which is similar to a LPF (shown in Figure 103 in Appendix C). A second-order ΔΣ can be proven to have the same characteristics, except that the asymptotic filter order is second order through the same analysis. The cut-off frequency of the asymptotic ΔΣ LPF is $f_{s,os}/2\pi$ and the attenuation at $f_{s,os}$ is about -10 dB. Thus when the oversampling ratio is high enough, the signal power that folds back, due to aliasing, into $[-f_M, f_M]$ will be 6 dB more attenuated by this filter. In this case, the oversampling assumption does not hold since the interested frequency range is the sampling frequency, not the signal bandwidth. The higher the order of ΔΣ ADC, the more attenuation is available for the in-band and folded-back signal power. This means that ΔΣ ADCs can provide a better SNR performance of the "no buffer" case in Figure 8 even when the

input is not band-limited and the buffer is not present. A simulation shows that fourth-order noise shaping ΔΣ ADC can have 64 dB SNR without a buffer, taking advantage of the inherent filtering characteristics.

2.5 Summary of Analog Front-end Performance Comparison

This section provides a summary of available SNR with various analog front-end configurations. We will assume that the filter order of the traditional ADCs is always fourth order and that the buffer for direct sample-and-hold and direct ADC cases is also fourth order.

When the band-limited input signal model A, $X_{in,A}(f)$ is applied, the output SNR plot is given in Figure 10. We see that the band-limited signal boosts the SNR of the traditional ADCs without LPFs by around 1 – 3 dB; however, they are still 20 – 30 dB worse than any oversampling converter. We also see that the oversampling ΔΣ ADC is still the best performer by slightly less than 30 dB.

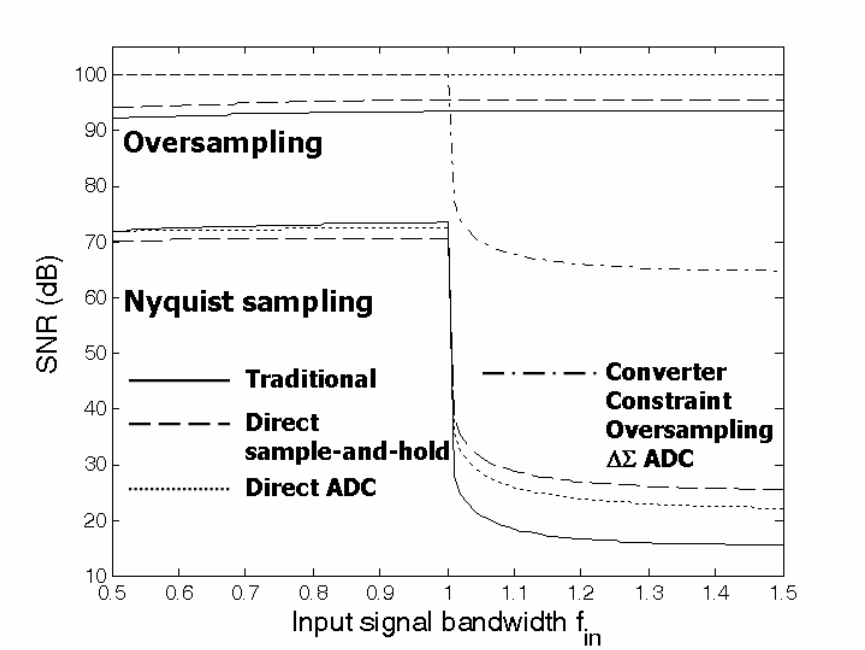

Figure 10: All ADC analog front-end SNR performance with input signal model A versus input signal bandwidth

With the wide-band input signal model B, $X_{in,B}(f)$, the SNR plot of every configuration is shown in Figure 11. Clearly any oversampled converter with a wide-band input stage provides almost the same performance. It is also clear that even a well-designed Nyquist converter will suffer approximately 30 dB sensitivity penalties. Furthermore, the filters and buffers will need to have bandwidths of 1/10th of the sampling frequency.

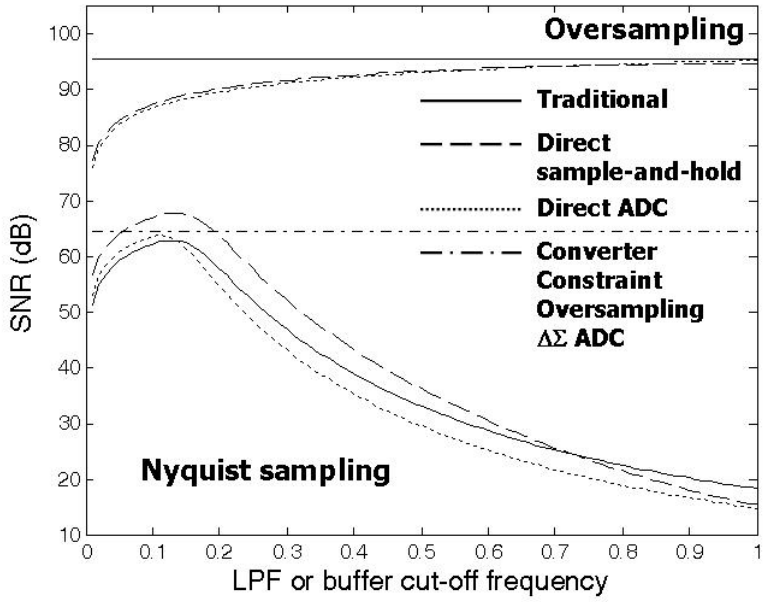

Figure 11: All ADC analog front-end SNR performance with input signal model B versus filter or buffer cut-off frequency

According to the simulations, though oversampling ADCs need minimal signal conditioning, the minimum first-order filtering is enough and more filtering actually degrades signal quality. Much more rigorous filtering is necessary for Nyquist sampling, and there is a trade-off between the filter noise and filter specification. Consequently, oversampling ADCs with the simplest front-ends can have better signal-to-noise ratio than Nyquist sampling ADCs with necessary analog front-ends in the perspective of a statistically-defined input signal in frequency domain.

The relative SNR performances compared with the maximum available performance of all cases when input signal model B with bandwidth f_M is applied are summarized in the Table 5.

Table 5: Relative SNR comparison of all cases with input signal model A, signal bandwidth f_M

Configuration	SNR (dB)
Buffer, 4th-order LPF, sample-and-hold Nyquist sampling	-26.7
Buffer, direct sample-and-hold Nyquist sampling	-29.5
Buffer, direct converter Nyquist sampling	-27.7
4th-order LPF, sample-and-hold oversampling	-6.6
direct sample-and-hold oversampling	-4.5
direct oversampling	0.00

Chapter 3

CODEC MODELING OF ADC AND DECODING PERFORMANCE ANALYSIS

This chapter takes a time-domain analysis approach to model an ADC as an encoder and decoder, or as a CODEC. While the discussion provided in the previous chapter puts emphasis on the available performance of ADC by characterizing the analog front-end, there is a limitation in the modeling of ΔΣ modulators as linear since they are actually nonlinear. Though the linear approximation is good enough for general-purpose analysis and tends to give a lower-bound pessimistic performance, the linearization of quantization error forces ΔΣ modulators to be underestimated in their performance.

A perspective that an ADC can be modeled as a communications channel and CODEC is presented in sections 3.1 and 3.2. A new proposed ΔΣ decoding scheme is presented in section 3.2. To explore the maximum available performance of ΔΣ CODEC, several decoding schemes, including the suggested algorithm, are compared in sections 3.3 - 3.5.

3.1 ADC Modeling as a Communication Channel

An ADC can be modeled as a general communication channel, and the channel capacity C of it can be derived using information theory as discussed in Appendix D. For an ADC input signal $X \subset [0, 1]$ with mean $\mu_X=1/2$ and variance σ_X^2, the output of ADC Y_k is given as (1). The quantization noise N is assumed to be Gaussian noise with a small enough quantization step Δ, whose mean is N = 0 and variance is $\sigma_N^2=\Delta^2/12$. Though Y_k and N_k take discrete values, let us ignore it for the time being. The input X_k takes an analog value. Let us assume an ideal anti-alias LPF and sample-and-hold for ADCs. The information capacity of a channel is given in (2), as discussed in the Appendix D. The quantization step size Δ and the number of quantization steps S_Δ are in inverse relation.

(1): $$Y_k = X_k + N_k, k = 1,2,\cdots,K$$

(2): $$C = \frac{1}{2}\log_2\left(1+\frac{P}{\Delta^2/12}\right)$$

It is notable that the information capacity of a channel is a function of signal power and noise variance, which is equivalent to noise power. For both a flash type ADC and a $\Delta\Sigma$ ADC, maximum signal power P is given as (3). With flash ADC, quantization noise power $\sigma_N^2=\Delta^2/12$ will be distributed over sampled spectrum as a white noise, and the capacity of the channel will be given as (4).

(3): $$P = \left(\frac{1}{2\sqrt{2}}\right)^2$$

(4): $$C = \frac{1}{2}\log_2\left(1 + \frac{P}{\Delta^2/12}\right)$$

With first-order ΔΣ ADC, the noise power can be obtained, as shown in (5), for a large oversampling ratio M, through an approximation of a sinusoidal frequency spectrum of the noise-shaping filter [11]. An ideal brick wall LPF is assumed for the downsampling filter. The channel capacity of ΔΣ ADC then is given in (6). The channel capacity is directly related to the number of bits available in the ADC as a communication channel. The results are compatible with linear analysis of ΔΣ analysis, as shown in Figure 12, where 9 dB per twice oversampling is available for the first-order filter (see Figure 83 in Appendix B).

(5): $$\sigma_{\Delta\Sigma}^2 = \int_{-1/2M}^{1/2M} \sigma_N^2 \,|2\sin(2\pi f)|^2 = \frac{\pi^2\sigma_N}{3M^3}$$

(6): $$C_{\Delta\Sigma} = \frac{1}{2}\log_2\left(1 + \frac{P}{\sigma_{\Delta\Sigma}^2}\right) = \frac{1}{2}\log_2\left(1 + \frac{36PM^3}{\pi^2\Delta^2}\right)$$

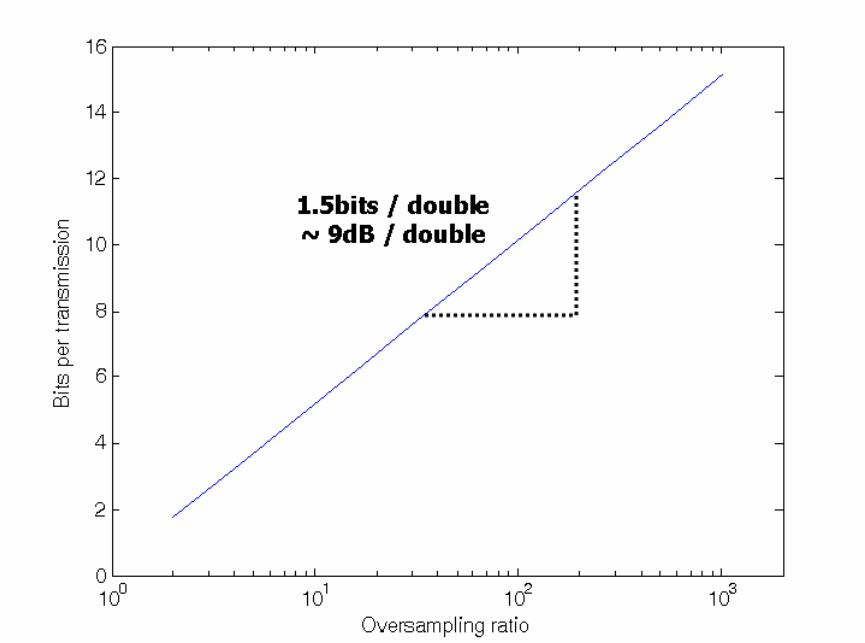

Figure 12: ΔΣ channel capacity as a function of oversampling ratio

3.2 ADC Encoding and Decoding

In most cases, an ADC input is an amplitude-modulated signal with a carrier. The carrier can be just a DC level or a sinusoid with a carrier frequency for a lock-in mode operation. Only DC amplitude input signal is treated in this research for simplicity. By the nature of analog-to-digital conversion, this analog input is turned into a digital representation, such as thermometer code, gray code, circular code, signed magnitude code, one's complement code, two's complement code, offset binary code, etc. [15]. One interesting thing to mention is that there are ADC internal digital representations that are different from the final digital representation. The flash ADC internal representation is the thermometer code, while the single-slope ADC uses a time-domain thermometer code.

A flash ADC internal representation goes through a decoder to generate a binary code, which can be in any code representation, and a counter converts single-slope ADC internal representation to the desired representation. An iterative algorithmic ADC has rather a natural binary code oriented design since its internal and final representations are same [11-13, 15, 17]. Usually, the final desired output is natural binary code, while it will depend on the number system that the following digital signal process system has. The entire process of A/D conversion can be modeled as encoding and decoding as shown in Figure 13. If an ADC has an intermediate representation similar to the final output representation, it can be thought of as a decoder-centered or top-down approach design.

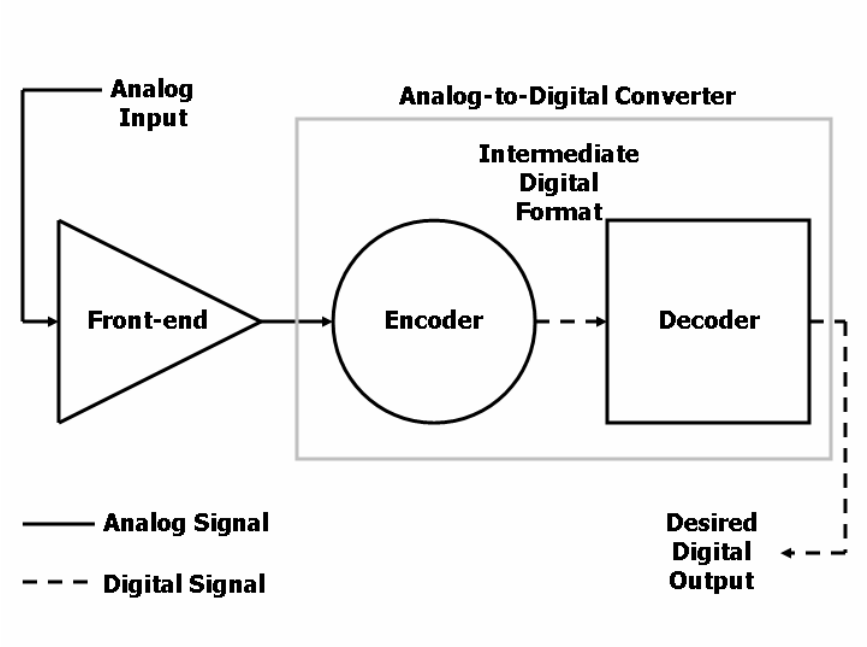

Figure 13: Encoding and decoding of analog-to-digital conversion process

The first conversion from analog signal to internal digital representation is a lossy conversion since infinitely accurate information in analog signal, excluding noise, is lost and unrecoverable quantization error is introduced. In practice, there is a thermal noise floor as a lower bound of noise at room temperature, and a conversion would be effectively lossless if the quantization noise power added is less than the noise floor. The second conversion from internal digital representation to final digital representation is lossless with most Nyquist sampling ADCs, in the sense that the acquired information is not lost during conversion. In practice, many of the second conversion processes have error correction capacity to enhance conversion speed and linearity [11-13, 15].

A ΔΣ ADC has a digital stream output as an intermediate output format, which is usually a 1-bit stream, and there are various downsampling filters and decoding algorithms to convert the internal representation to the desired final format. Unlike Nyquist sampling ADCs, the second conversion process is not necessarily lossless, though the process occurs in the digital domain. Traditionally, downsampling filters have been used as a decoder, and several nonlinear decoding algorithms have been devised to attempt to find an optimal solution to the given ΔΣ output stream [31, 48-54].

Without noise in the input and noise added by the circuits, the output stream of a ΔΣ encoder is deterministic for a given input. Though the analog input is a real number, it can be approximated and replaced with a rational number since digitization will discard unconverted information, resulting in an integer output. With L-bits output stream, there are 2^L degree of freedom, or the number of code words, while only a portion of this freedom is taken advantage of by the ΔΣ encoder [29]. Let us assume that the input is

distributed over [0, 1] and a reference value for 1-bit analog-to-digital decision is 1/2. Using the nonlinear discrete ΔΣ system model discussed in Appendix B, the decoding problem for the first-order noise shaping encoder with constant input x and oversampling ratio L can be formulated as (7), where u is unknown internal state and y is observed digital output stream. At time n=k-1, input x can be estimated by adding up equations from n=0 to n=k-1 as given in (8). The bounds given in the equation are rather pessimistic since the internal state u is not known at all. Using all the bounds throughout the conversion period, a tighter bound for the estimated x can be obtained in (9). As a good guess, the middle point of the bound can be suggested as the solution to the problem as given in (10).

(7): $$u[n+1] = u[n] + x - y[n], n = 0,1,2,\cdots L-1$$

(8): $$\frac{-1+\sum_{n=0}^{k-1} y[n]}{k} \le x \le \frac{+1+\sum_{n=0}^{k-1} y[n]}{k}$$

(9): $$\max_{k=[1,L]}\left\{\frac{-1+\sum_{n=0}^{k-1} y[n]}{k}\right\} \le x \le \min_{k=[1,L]}\left\{\frac{1+\sum_{n=0}^{k-1} y[n]}{k}\right\}$$

(10): $$x_{est} = \left[\max_{k=[1,L]}\left\{\frac{-1+\sum_{n=0}^{k-1} y[n]}{k}\right\} + \min_{k=[1,L]}\left\{\frac{1+\sum_{n=0}^{k-1} y[n]}{k}\right\}\right] / 2$$

More rigorous bounds can be found by incorporating comparator output of each time step. When an output bit is high, the internal state is greater than the reference value; therefore, the lower bound estimation can be given as in (11). In the same way, an upper bound is found in (12) when the output bit is low. The estimation of input signal using both bounds is given in (13).

$$\textbf{(11):}\quad x \geq \frac{\sum_{n=0}^{k-1} y[n] + 1/2 - u[0]}{k} \geq \frac{\sum_{n=0}^{k-1} y[n] - 1/2}{k}$$

$$\textbf{(12):}\quad x \leq \frac{\sum_{n=0}^{k-1} y[n] + 1/2 - u[0]}{k} \leq \frac{\sum_{n=0}^{k-1} y[n] + 1/2}{k}$$

$$\textbf{(13):}\quad x_{est} = \left[\max_{k=[1,L]} \left\{ \frac{-1/2 + \sum_{n=0}^{k-1} y[n]}{k} \right\} + \min_{k=[1,L]} \left\{ \frac{1/2 + \sum_{n=0}^{k-1} y[n]}{k} \right\} \right] / 2$$

The decoding performance of the proposed algorithm will be compared in the following sections. The solution is similar to that of the "zoomer" algorithm, in the sense that it uses iterative bounds, but it does not assume any initial conditions on converter internal state during derivation [28, 49]. Also it uses two-way bounds rather than one-way bounds to get tighter bounds. The actual performance of the proposed algorithm is dependent on the internal states. Figure 14 shows an example of the solution tracking behavior of the lower and upper bound in the proposed algorithm. The input value to ΔΣ

ADC is $\pi/7 \approx 0.4488$ as a random number, and the absolute error of each bound and the final estimation from the input value are plotted. The operation of the nonlinear maximum and minimum functions can be seen in the long periods of constant error. During these periods, the bounds at individual iterations are worse than that for earlier iterations.

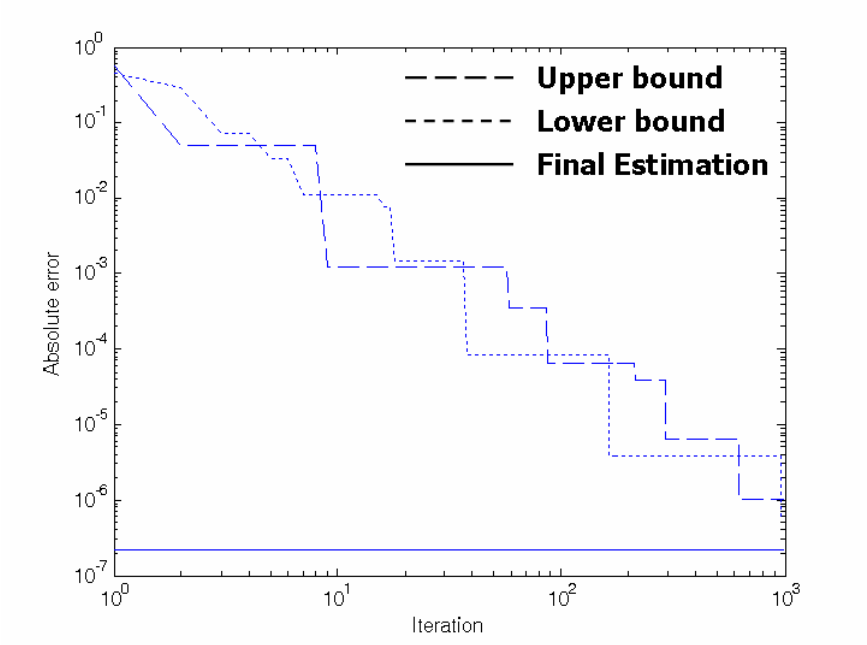

Figure 14: The solution tracking behavior of the proposed algorithm

3.3 ΔΣ Nonlinear Decoding Algorithm Performance in Ideal Circuits

A first-order ΔΣ modulator with non-ideality models shown in Figure 15 is used to evaluate the performance of various decoding algorithms. The input range of the ADC is [0, 1] and the ideal decision point for the 1-bit quantizer is 1/2 as defined in (14). All the non-ideal factors are turned off for ideal circuit decoding performance measurement.

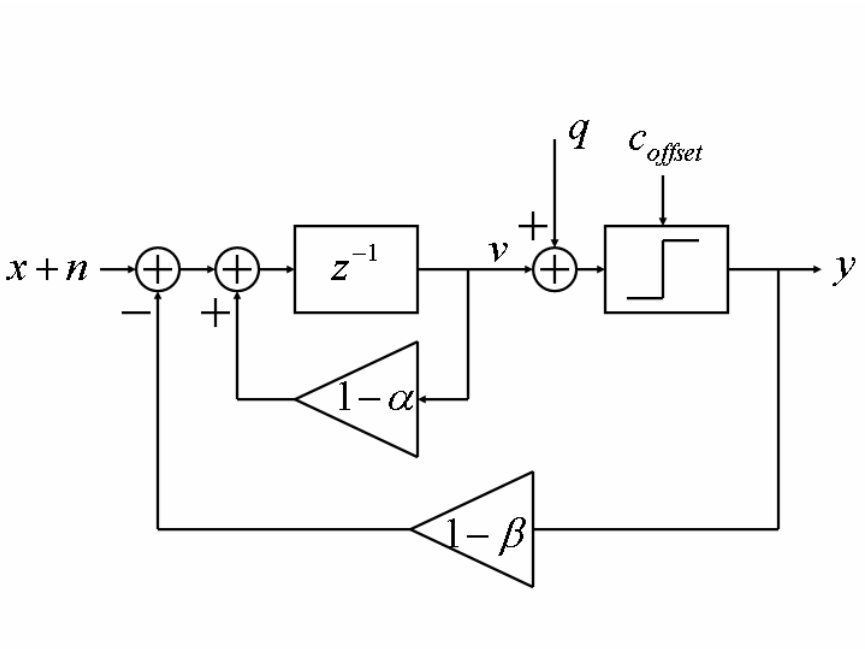

Figure 15: Error model of 1st-order 1-bit ΔΣ ADC

(14): $$q(x) = \begin{cases} 1 & x > 0.5 \\ 0 & x \le 0.5 \end{cases}$$

A sub-optimal filter [50] is selected to show the relative performance of linear filters. Three nonlinear decoding algorithms for DC level acquisition - zoomer algorithm [49], recursive [48], and the proposed scheme - are compared with assumptions that input signal is stationary and noiseless, and that circuit is also noiseless and perfect. There is an issue in zoomer algorithm that the unknown initial state degrades search results and makes it worse than the simplest linear filter. As long as it is a decoder, rather than a filter, it is possible to initialize internal state for every decoding. All nonlinear algorithms are given initial conditions for the best performances. Not only the signal-to-noise ratio (SNR) measurement, but also the decoding time could be of interest. Though the running time of each algorithm is highly dependent on the specific hardware realization, each decoding time is recorded and averaged for on-hand comparison [48, 49, 53, 54].

The zoomer, recursive, and proposed algorithm show better SNR than sub-optimal linear filter by about 5.3 dB, 2.6 dB, and 10.2 dB, as shown in Figure 16. Every decoding scheme has more than 9 dB gain per twice OSR. Also the average decoding time of each algorithm is shown in Figure 17.

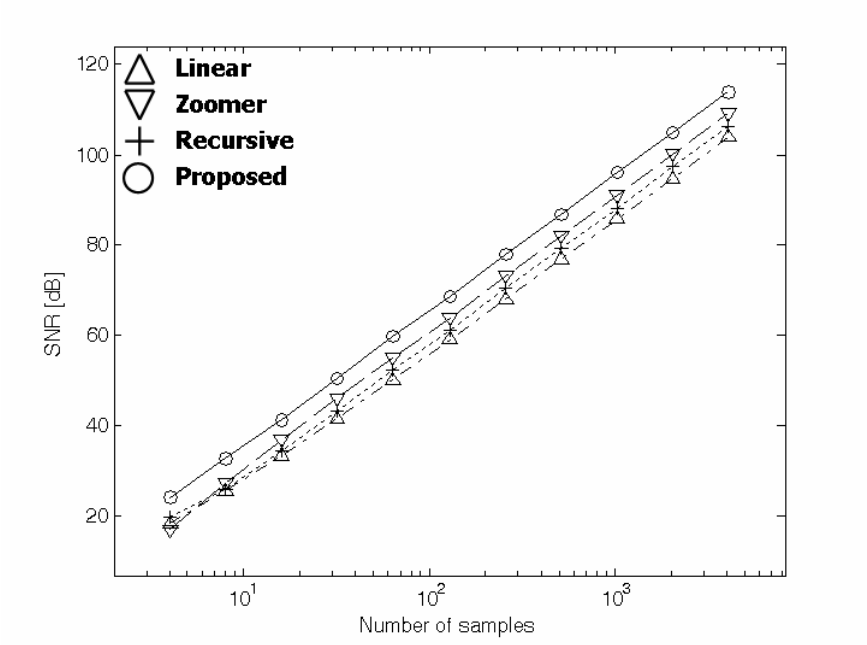

Figure 16: Signal-to-noise ratio performance of ideal ΔΣ decoding schemes

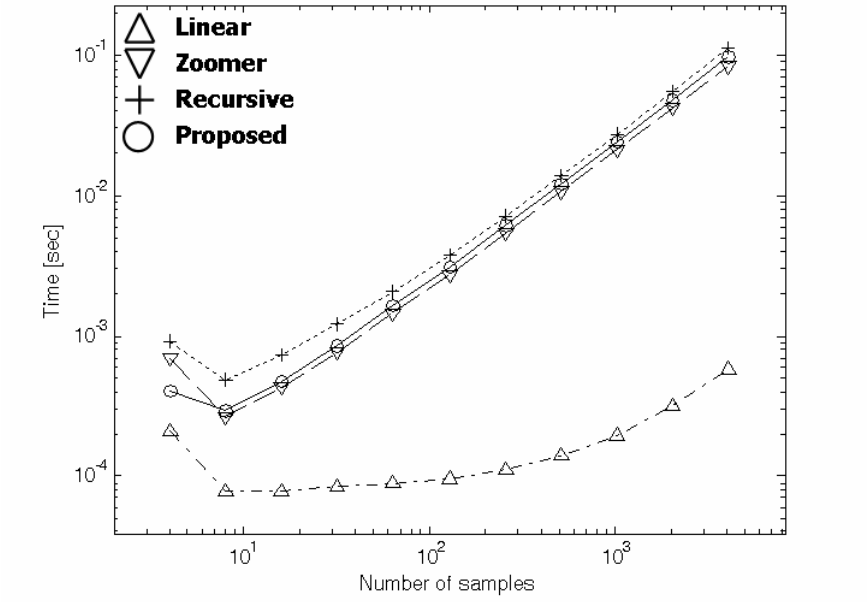

Figure 17: Average decoding time of each decoding scheme

3.4 ΔΣ Nonlinear Decoding Performance with Nonidealities

The ideal circuit is rather imaginary and provides no better than a mathematical verification. There are inevitable circuit imperfections and unexpected disturbances, as modeled in Figure 15. Stationary circuit imperfections, such as quantizer offset, leakage in integration, and feedback gain, are considered first. Noise in signal and integrator are equivalent in the first-order ΔΣ ADC. The comparator for a 1-bit ADC is prone to decision errors. These random influences are also considered. The oversampling ratio (OSR), the number of samples taken for filtering or decoding, is fixed to 256 for all of the following simulations. The definition of OSR in this context is different from that used in the signal processing.

3.4.1 Stationary Circuit Nonlinearity

The offset of quantizer can be modeled as (15), where c_{offset} is the amount of offset in comparator. The leaky integration is described by a difference equation (16), where the degree of leakage is α. Also feedback error β caused by non-unity feedback gain is modeled in the same equation.

(15): $$q(v) = \begin{cases} 1 & v > 0.5 + c_{offset} \\ 0 & x \le 0.5 + c_{offset} \end{cases}$$

(16): $v[n+1] = (1-\alpha)v[n] + x[n] - (1-\beta)y[n]$

Figure 18 is the simulation result of DC offset sweep of [-0.05, 0.05]. The performance of the zoomer and the proposed algorithms degrade noticeably as DC offset is increased while others remain same or fluctuate. As mentioned in [28, 49], DC offset in quantizer is equivalent to offset in initial condition for zoomer, and it cannot be compensated as long as the offset is not known. For the proposed algorithm, the obtained upper and lower bounds become inaccurate as the offset increases, which results in a rapid downturn of SNR.

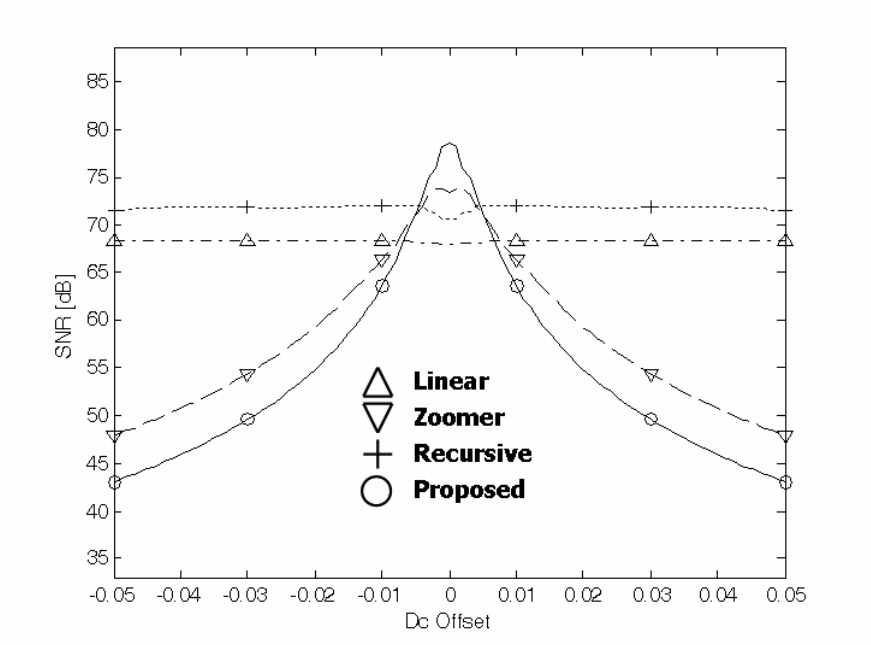

Figure 18: Signal-to-noise ratio with DC offset c_{offset} in comparator

The integration leakage α is swept in range of $[0, 2\times10^{-3}]$ and the resulting SNR is plotted in Figure 19. The performance degrades rapidly with all algorithms. It is interesting to see that the linear filter and nonlinear algorithms are following similar trajectories. Feedback error β causes almost the same effect on the decoding performance for the same sweep range. Figure 20 shows SNR of decoding algorithms with feedback error β sweep over $[-1\times10^{-3}, 1\times10^{-3}]$.

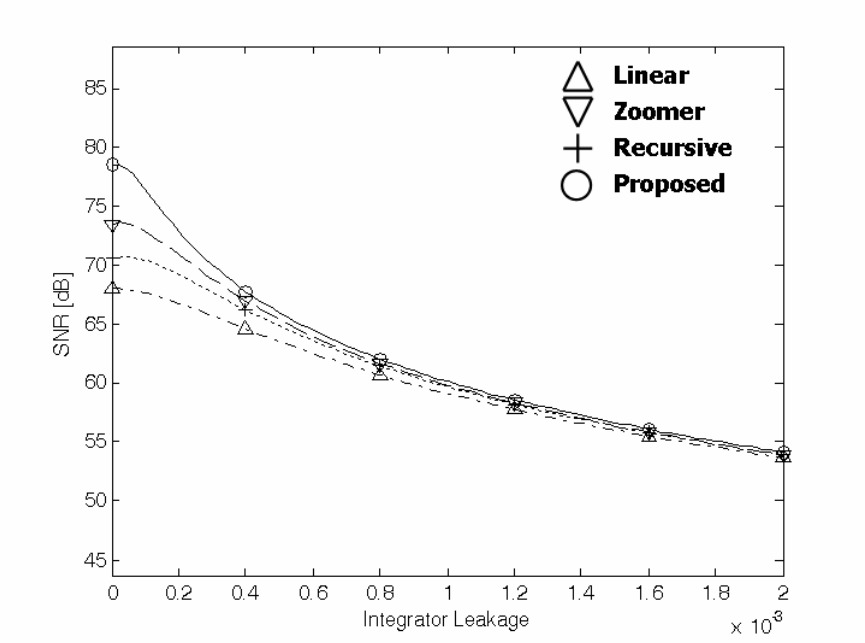

Figure 19: Signal-to-noise ratio with integrator leakage α

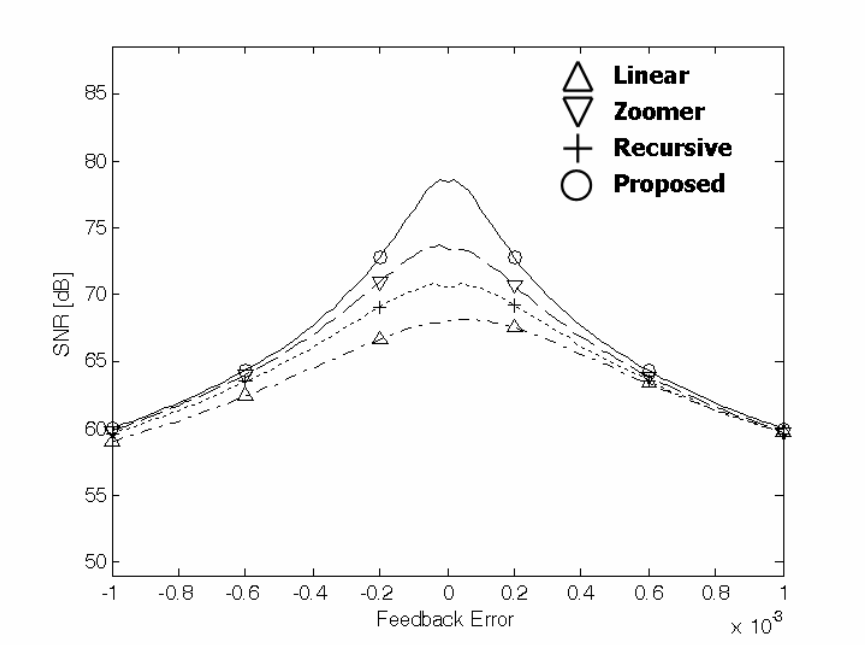

Figure 20: Signal-to-noise ratio with feedback error β

3.4.2 Performance with Noise Sources

Two separate Gaussian random noise sequences were generated with standard deviation σ_n and σ_q, and added to the converter input and the quantizer input. Unequal weights that nonlinear algorithms put on the output streams of ΔΣ would make them vulnerable to burst noise. Also robustness against time-varying signal would be interesting to see if the algorithms are useful for the reconstruction of band-limited signal, though it is also ignored here to concentrate on the DC-level acquisition application, assuming that OSR is high enough and input is almost DC for sensor SoC application.

All algorithms are less sensitive to quantizer noise than signal noise as shown in Figure 21 and Figure 22, which is natural since the ΔΣ decoder should have a low-pass

filtering characteristic, and quantization noise shaping reduces noise power in lower frequency band. Also nonlinear algorithms fail faster than linear filters with extreme noise in quantizer. The propose algorithm fails faster than other algorithms, and it is reasonable that the bounds are calculated based on the ideal and fixed quantizer decision point.

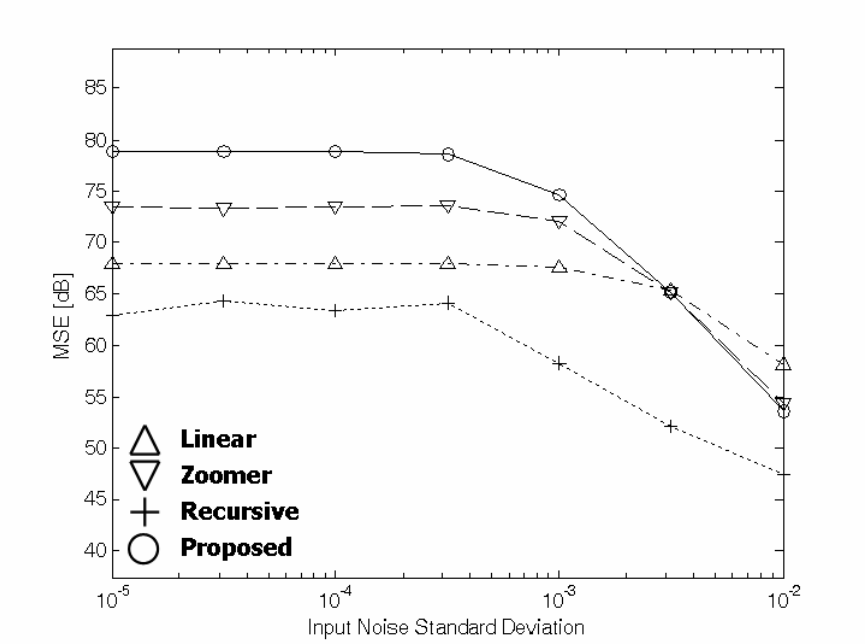

Figure 21: Signal-to-noise ratio with input signal noise

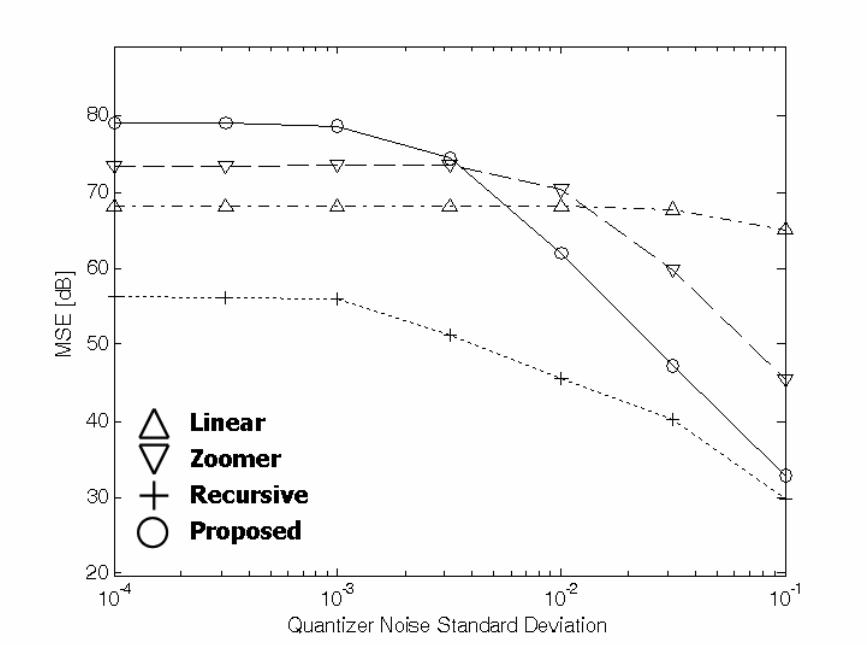

Figure 22: Signal-to-noise ratio with quantizer noise

3.5 Summary with Comprehensive Evaluation of ΔΣ Decoding Algorithms

All non-ideal factors are taken into consideration to provide an estimate of real circuit performance. Let us assume $\alpha = 1.0\times10^{-3}$, $\beta = 1.0\times10^{-3}$, $c_{offset} = 2.0\times10^{-3}$, $\sigma_n = 5.0\times10^{-6}$, and $\sigma_q = 5.0\times10^{-6}$. When OSR is $2^8 = 256$, the estimated performance is summarized in dB in Table 6. The loss of the linear filter experience is smaller than that of nonlinear decoding schemes. Though the proposed algorithm suffers the worst loss of sensitivity, it still shows the best decoding performance, followed by zoomer algorithm.

Table 6: Decoding performance with all non-ideal effects

Decoding algorithm	Estimated SNR
Optimal	68.32
Zoomer	72.97
Recursive	71.04
Proposed	74.55

Chapter 4

CASE STUDY ON SENSOR ADC READOUT SYSTEM

This chapter provides a case study for robust and flexible ADC designs. The application is an integrated system-on-a-chip (SoC) sensor system with embedded photo detector array input. The simplified system diagram is given in Figure 23.

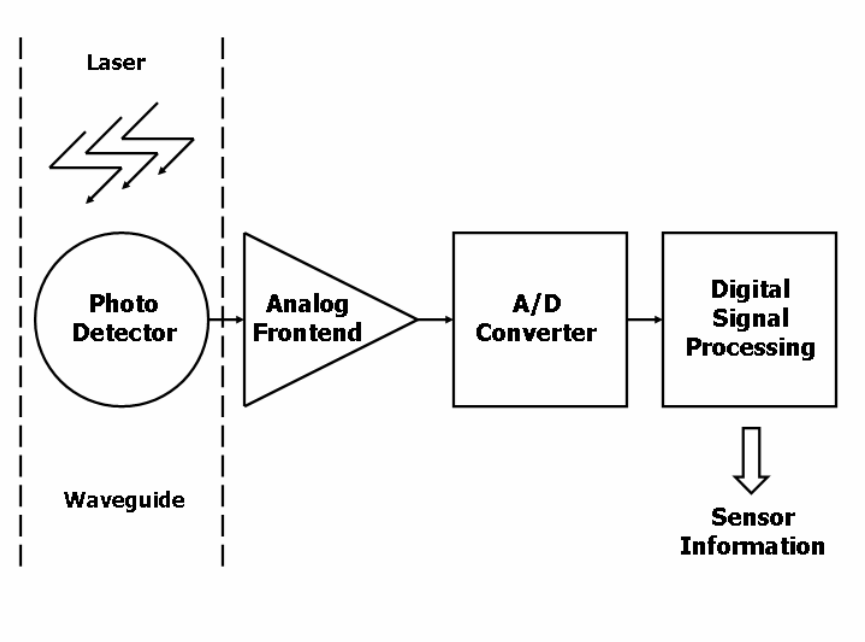

Figure 23: Simplified sensor system diagram

Several considerations for integrated SoC sensor application are discussed in section 4.1. Performance requirements of analog front-end for photo detector input is analyzed in section 4.2. Two popular sensor readout ADCs with high sensitivity and compactness are designed for comparison in sections 4.3 and 4.4. Theoretical developments discussed in Chapter 2 and Chapter 3 are put to use for the comparison of designed ADCs in section 4.5. Implemented ΔΣ ADC circuit testing results with electrical, optical, and sensor system measurements will be given in sections 4.6 - 4.8.

All the designs presented in the chapter are based on AMI Semiconductor 1.5 μm Si-CMOS process provided through MOSIS [8].

4.1 Embedded Photo Detector and Array

There are many configurations available for photo detectors with standard Si-CMOS process, including BJT and PiN type detectors as shown in Figure 24 and Figure 25 [55].

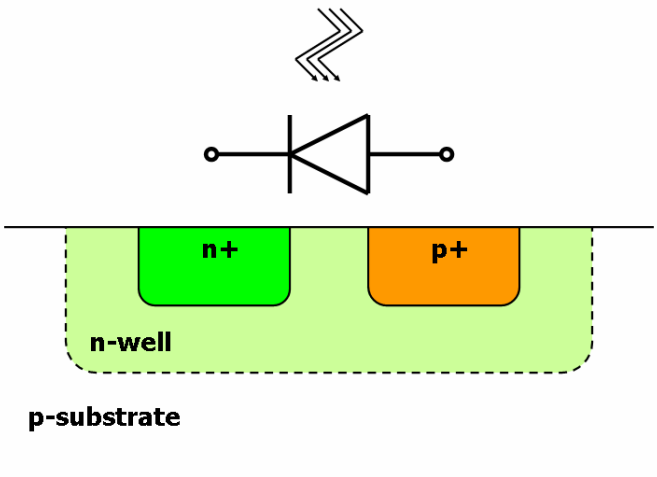

Figure 24: Embedded PiN photo detector

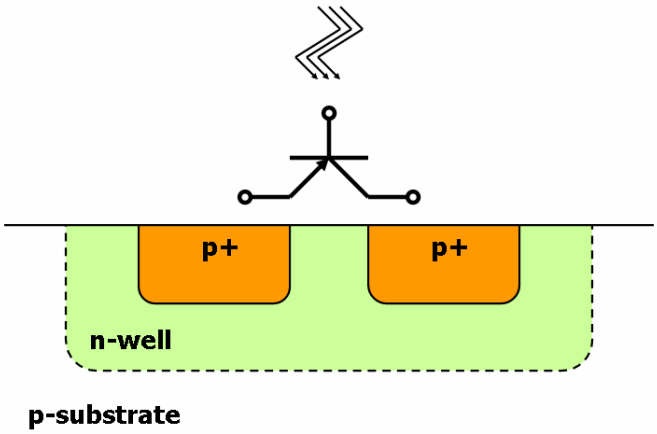

Figure 25: Embedded BJT photo detector

The size, spacing, and position of the photo detector should be optimized for the maximum sensitivity of embedded waveguide application. The sensitivity of a photo diode pixel is the function of detector size and noise. For waveguide input signal, one detector makes one pixel in an array, and an array of detectors makes a one-dimensional image scanner to perceive an optical pattern in the waveguide, assuming coupling of laser

into photo detectors. The detector size should be large enough to collect sufficient light compared with the dark noise of the detector and it should be small enough to generate meaningful pattern information in the waveguide.

Figure 26 shows the photo and dark current measurement of embedded Si-CMOS detector with evanescent field coupling, where the measured responsivity is 0.12 A/W at 632.8 nm [7]. “Rule of thumb” sensitivity available with the embedded detector is about 100 dB, which is equivalent to about 16-bits resolution.

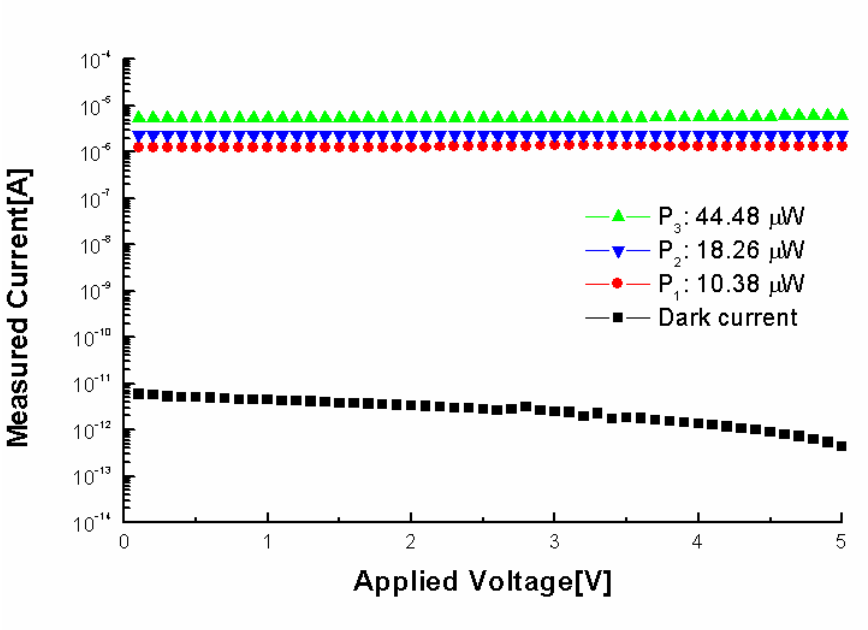

Figure 26: Photo and dark current measurement with evanescent coupling to Si-CMOS detector

The plots in Figure 27 and Figure 28 are time-domain responses of an embedded Si-CMOS detector with 85 Hz and 1 kHz chopped HeNe laser. The photo detector voltage biasing was 2.2 V, and the oscilloscope was terminated with 1 MΩ in the first plot and 50 Ω load in the second plot [7]. The bandwidth of the detector is above 1 kHz according to the plots.

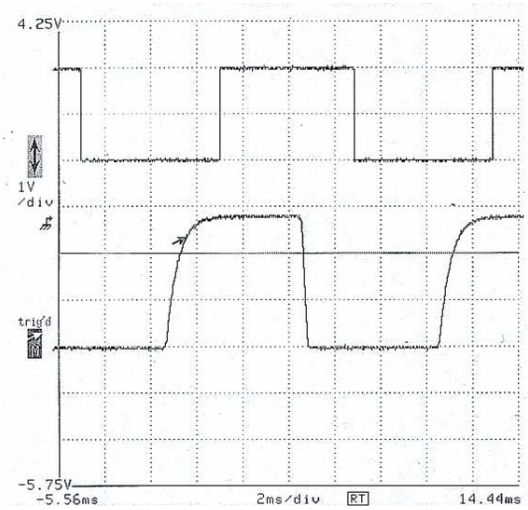

Figure 27: Time domain response to 85 Hz-chopped signal

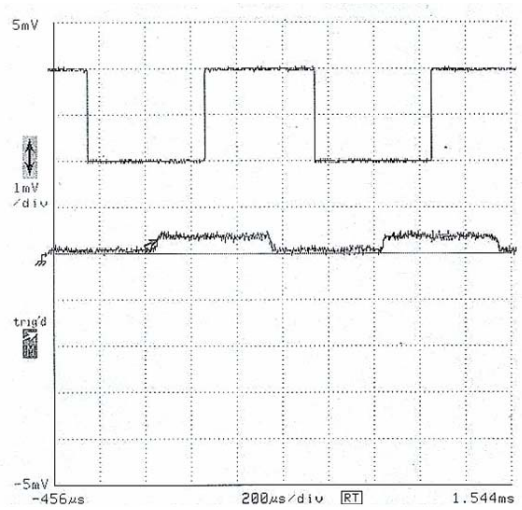

Figure 28: Time domain response to 1 kHz-chopped signal

With embedded optical waveguide and integrated Si-CMOS photo detector array, the coupling of laser into the detectors will be butt coupling or evanescent field coupling. Figure 29 and Figure 30 shows two side views of Si-CMOS circuits. Figure 29 has a common ground made of metal contacts to the substrate. Figure 30 has diffusion area used as an electrical common ground connection. The diffusion-based ground contact produced a very different geometry for the waveguide. The actual coupling mechanism will be highly dependent on the properties of integrated waveguide on the surface of the Si-CMOS circuit. Important factors are the thickness and termination of waveguide, the geometry of the Si-CMOS circuit itself, the uneven Si-CMOS chip surface, and the position of metal and contacts, which will cause scattering of the laser.

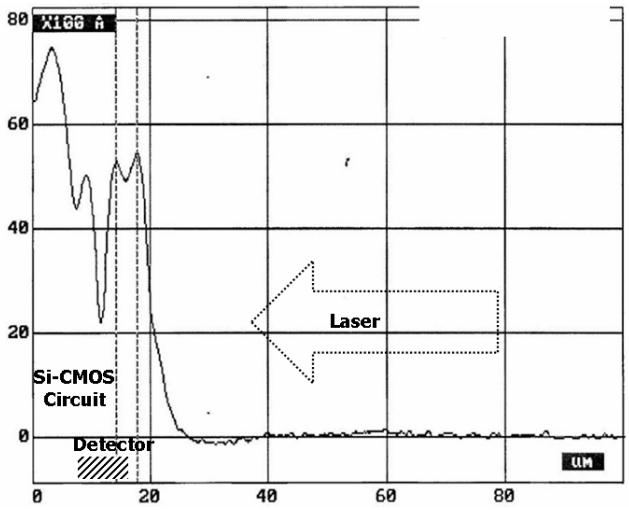

Figure 29: Side view of Si-CMOS circuit A

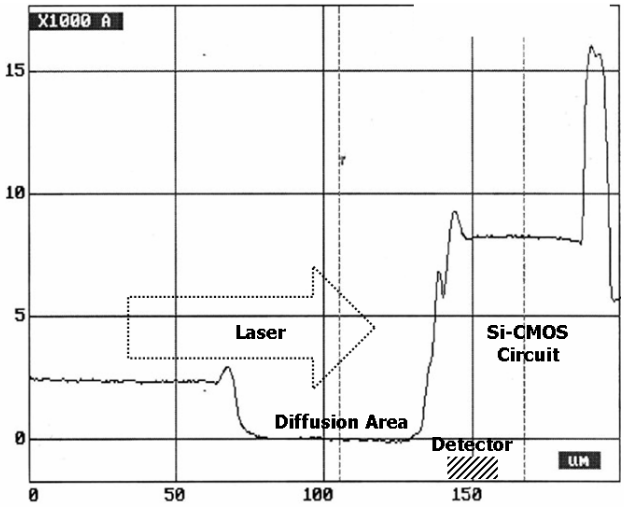

Figure 30: Side view of Si-CMOS circuit B

4.2 Analog Front-end

The analog front-end designed for the embedded photo detector collects photocurrent as input with voltage biasing to photo detector. The output of the analog

front-end should be current to be compatible with a designed analog-to-digital converter, which will be discussed in the following sections.

As a buffer, an analog front-end should provide a constant biasing independent of photo current input. A designed analog front-end buffer schematic diagram is given in Figure 31. Figure 32 shows HSPICE [56] simulated biasing voltage to photo detector when the input photo current is linearly increased. Also output current of the buffer is plotted along the input current sweep in Figure 33.

All the simulations were performed with the recent 34 AMI Semiconductor 1.5 μm Si-CMOS transistor models obtained through MOSIS [8], and the MOSIS run numbers are provided in Appendix E.

The buffer biasing monotonically decreases while it remains relatively constant in the range of 0 - 1 μA. The change in biasing voltage can be ignored as long as the supplied bias voltage is large enough for the photo detector to be turned on, according to the photo detector characteristics given in the previous section.

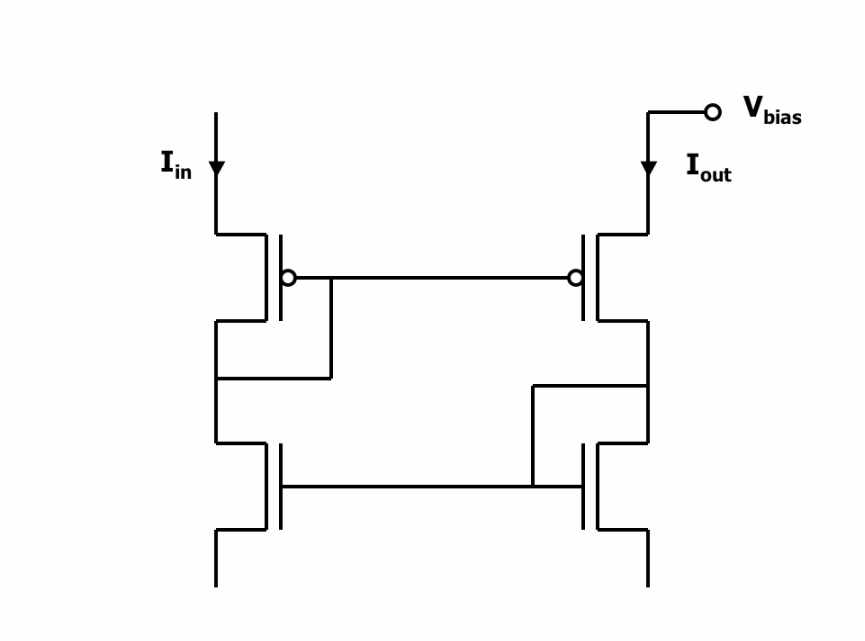

Figure 31: Analog front-end buffer schematic diagram

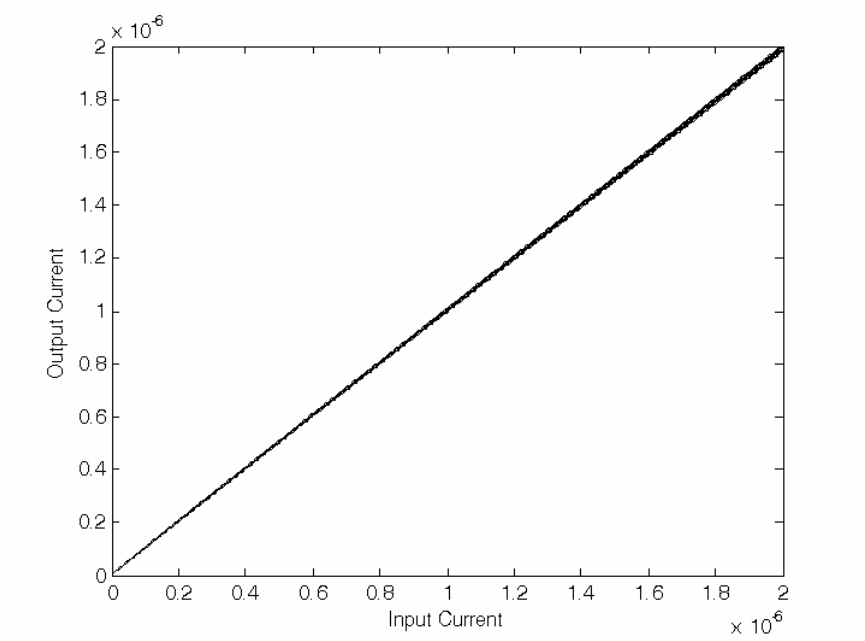

Figure 32: Output current of analog front-end with DC sweep

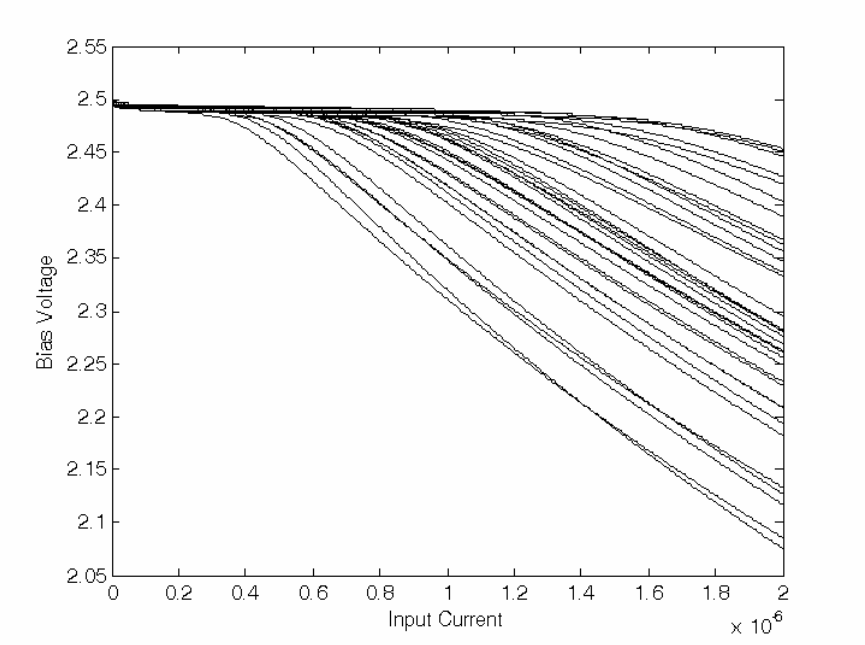

Figure 33: Bias voltage output to detector with input DC sweep

Also the bandwidth and noise of the buffer are important characteristics, as shown in Figure 34 and Figure 35. The buffer is biased through the photo detector, and the frequency and noise responses are dependent on the input current level. Assuming input photo current range is 0 – 1 μA, the circuit biasing point is set to 0.5 μA.

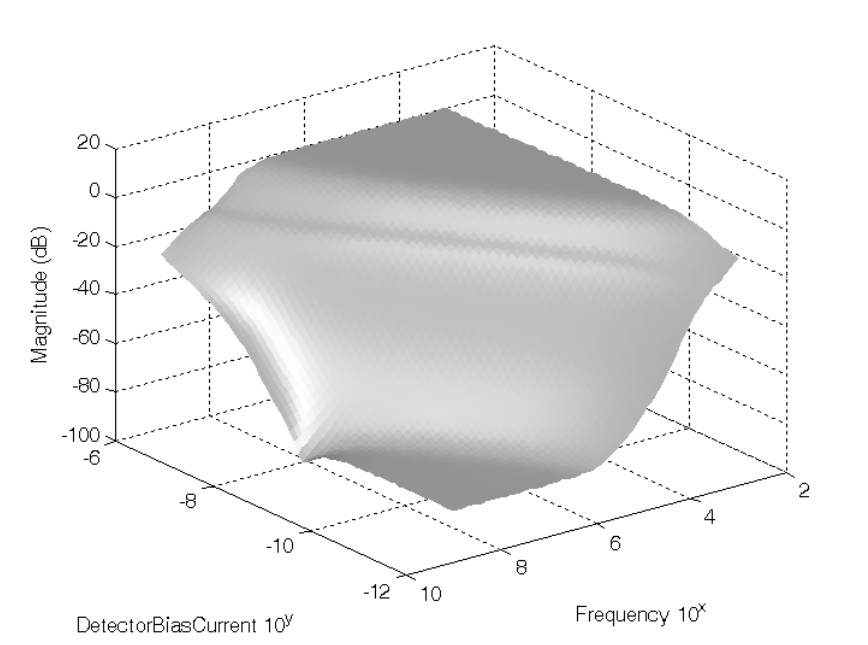

Figure 34: Analog front-end frequency response with varying input current

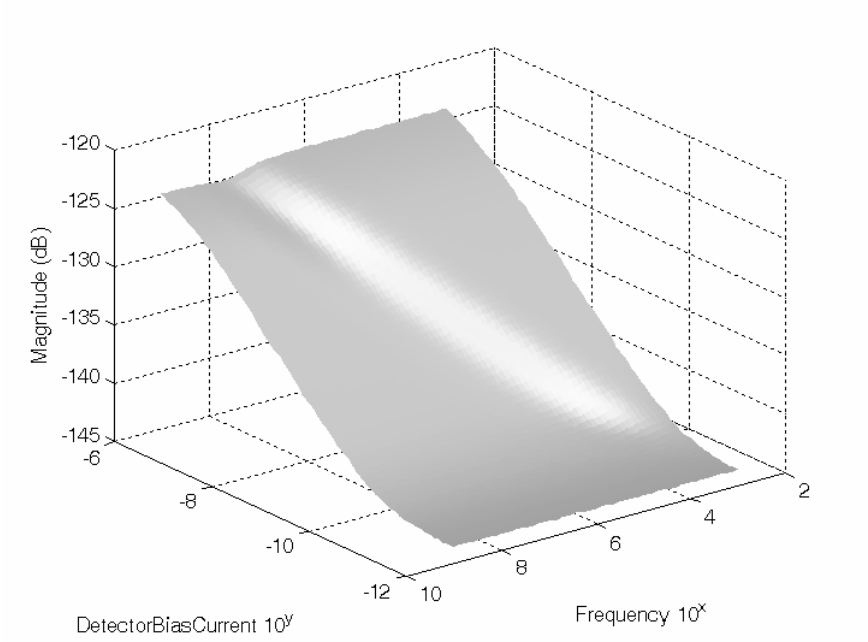

Figure 35: Analog front-end output equivalent noise with varying input current

The buffer bandwidth goes up to the order of 10 MHz at the biasing point, as illustrated in Figure 36, which is high enough for sensor applications. Input equivalent flat spectrum noise is also obtained, which is about 4×10^{-13} V/√Hz across the bandwidth at the biasing, as shown in Figure 37. The equivalent voltage was calculated from the output of an ideal trans-impedance amplifier attached to the buffer output.

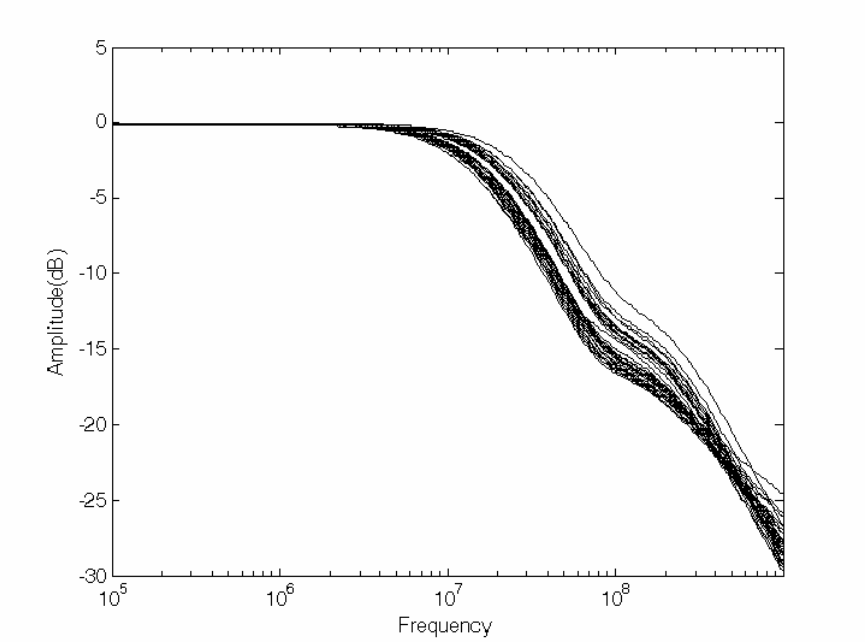

Figure 36: Analog front-end frequency response

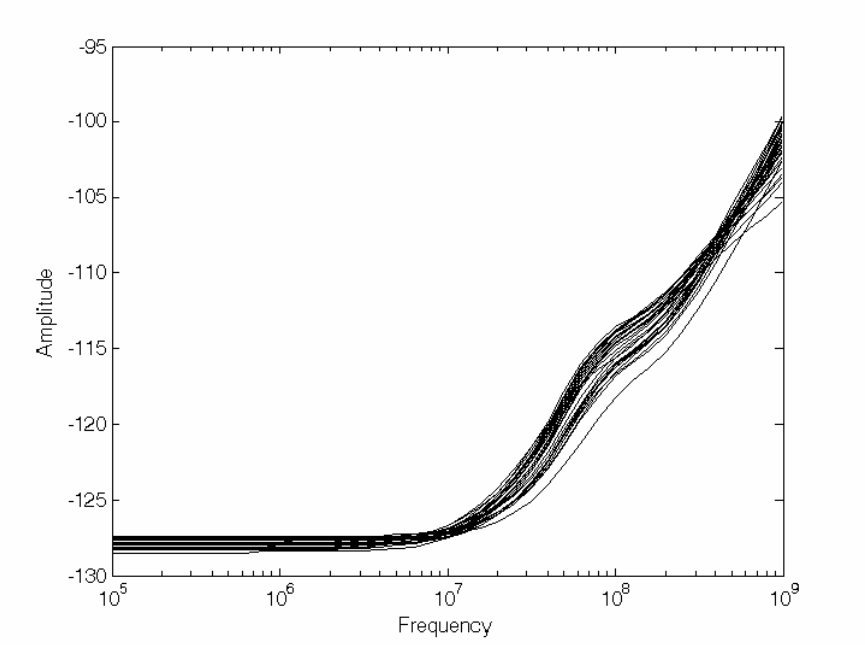

Figure 37: Analog front-end input equivalent noise

The simulated noise power is a noise floor for the buffer, where the circuit is not sensitive any more to the signal power below it. The simulated analog front-end allows more than 127 dB dynamic range with respect to the desired signal input range. The dynamic range is equivalent to about 20.8-bits resolution with digital representation, as shown in Figure 37.

4.3 ΔΣ ADC Design

The ΔΣ ADC has been an alternative way to implement ADC for instrumentation. In this section a first-order continuous-time ΔΣ ADC design and simulation outputs will be presented. Figure 38 shows ΔΣ converter function block diagram.

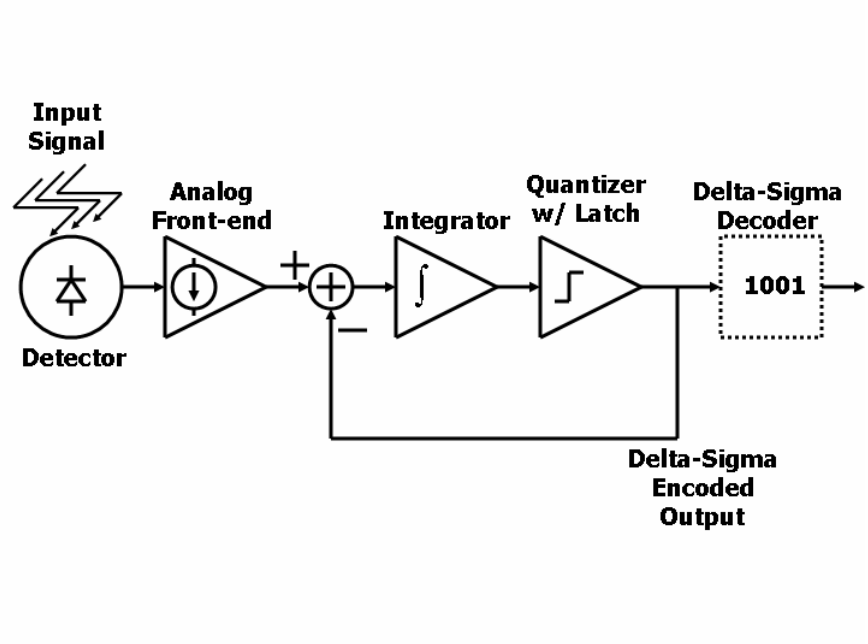

Figure 38: First-order ΔΣ ADC function block diagram

Detector and analog front-end specifications and characterization were provided in the previous sections. A continuous-time integrator with a capacitor is used for the first-order filtering in the feed-forward path. A clocked latch works as a quantizer and unit-time delay. To maintain flexibility in the selection of decoding scheme, decoding function block is provided by external post-processing function block. The circuit schematic and layout of the design are provided in Figure 39 and Figure 40, and the active circuit area is $144 \times 105\ \mu m^2$, excluding capacitor area.

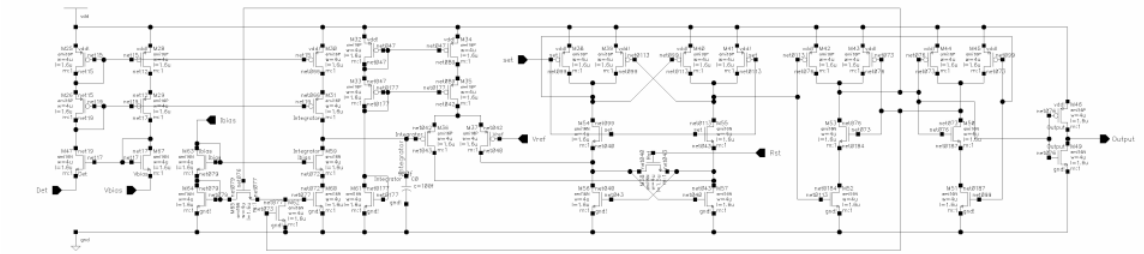

Figure 39: First-order ΔΣ ADC circuit schematics

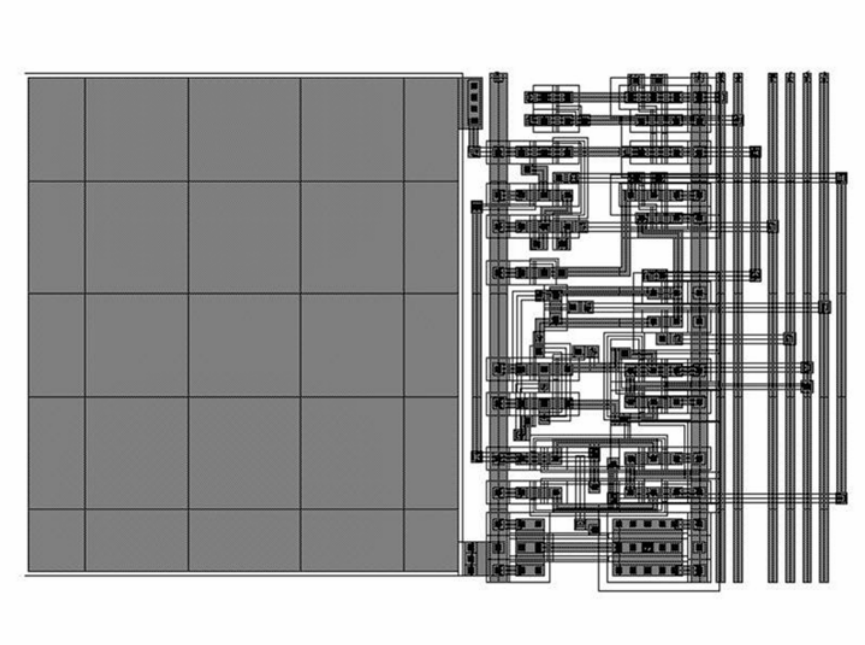

Figure 40: First-order ΔΣ ADC circuit layout

The design was extracted for spice simulation with parasitic capacitance. Figure 41 is an example ΔΣ ADC HSPICE simulation output with sinusoidal current input. A plot in Figure 42 is ΔΣ oversampling converter output level acquisition HSPICE simulation of input current sweep from 0 to 1 μA, using MOSIS Si-CMOS model run number T0CU (see Appendix E). The decoding filter used in the plot is a first-order comb filter.

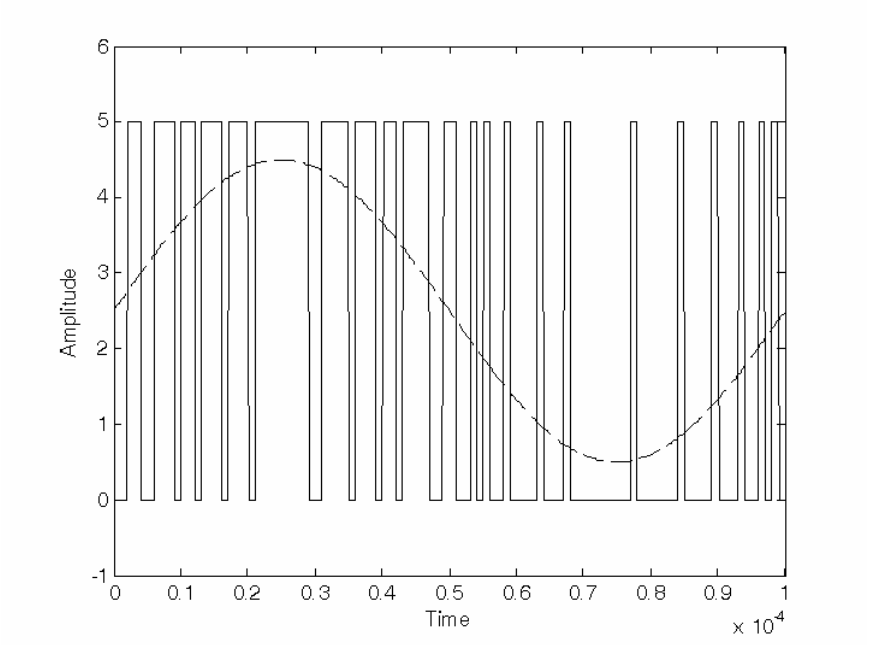

Figure 41: First-order ΔΣ ADC extracted circuit simulation output

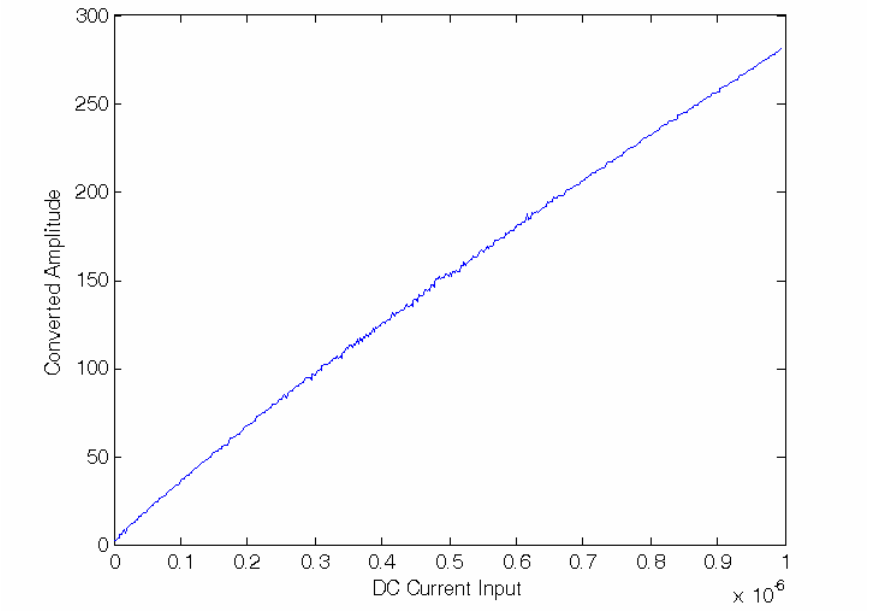

Figure 42: First-order ΔΣ ADC DC sweep output acquisition with comb filter

4.4 Single-slope ADC

Single-slope ADCs have been used for instrumentation system-on-a-chip designs traditionally. This section provides a single-slope ADC design and simulation outputs. Figure 43 shows a single-slop ADC function block diagram. To simplify the design, the dotted function blocks are supported externally.

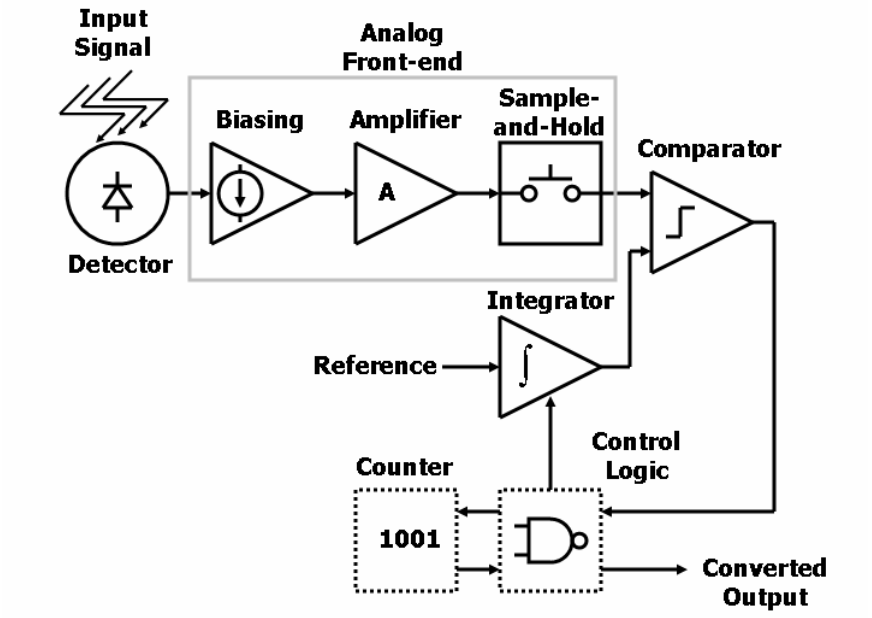

Figure 43: Single-slope ADC function block diagram

The designed single-slope ADC uses a trans-impedance amplifier to convert current input to voltage. The required gain is 5 V / 1 μA = 5×10^6 V/A. Using the same buffer given in earlier section, biasing circuit, amplifier stages, and sample-and-hold

compose analog front-end of the ADC as shown in the figure. The single-slope ADC analog front-end AC characteristics are shown in Figure 44 and Figure 45.

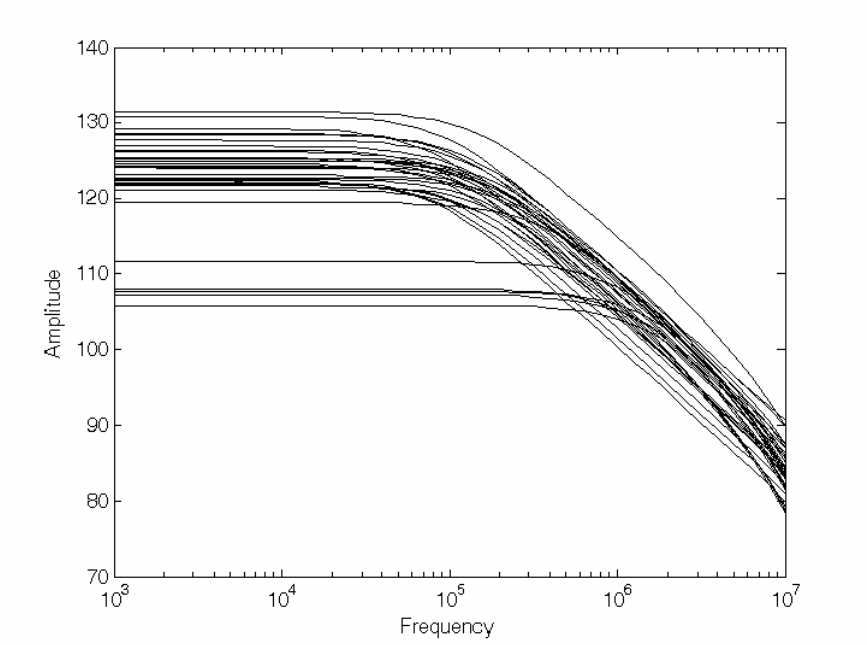

Figure 44: Frequency response of single-slope ADC analog front-end

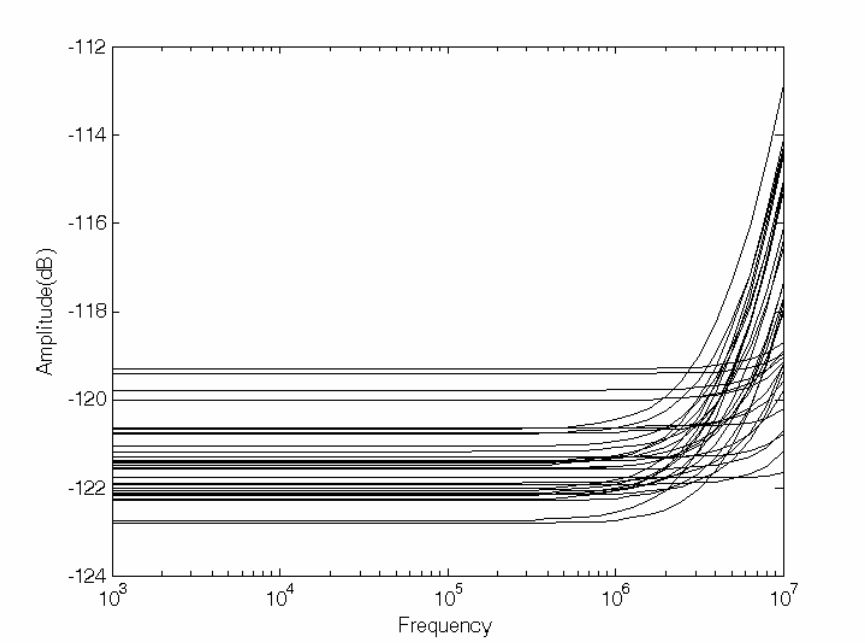

Figure 45: Single-slope ADC analog front-end input equivalent noise

According to Figure 44, the nominal gain of the front-end is about 125 dB with an approximate signal bandwidth 10 kHz. As shown in Figure 45, the worst-case input equivalent noise with respect to input signal level is about –119 dB, which is equivalent to 19.5-bits resolution in digital representation. There is 8 dB more circuit noise involved in the single-slope ADC analog front-end compared with the ΔΣ oversampling ADC analog front-end.

The single-slope ADC design schematic and layout are shown in Figure 46 and Figure 47. The active circuit takes $200 \times 115\ \mu m^2$, excluding capacitors. Analog front-end, reference integrator, and comparator are included in the circuit, assuming counter and control logic are provided externally.

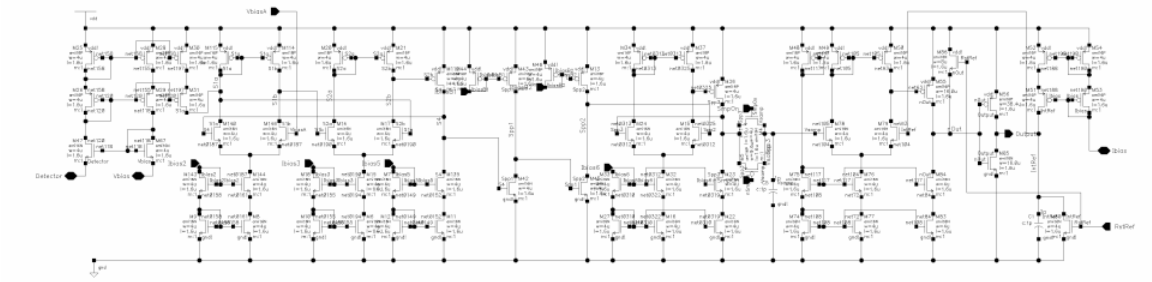

Figure 46: Single-slope ADC circuit schematics

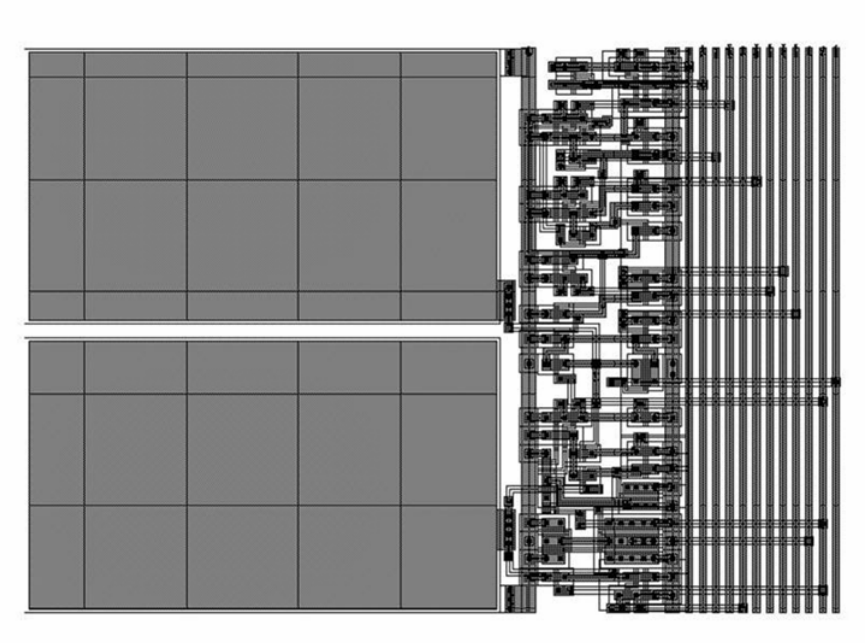

Figure 47: Single-slope ADC circuit layout

The layout was extracted with parasitic capacitance, and Figure 48 shows circuit HSPICE simulation output with sinusoidal input current. A DC-level acquisition HSPICE simulation of input current sweep from 0 to 1 μA is provided in Figure 49. The DC-sweep acquisition plot shows variable gain error while the monotonicity of the output is maintained. Transistor model MOSIS run number T0CU (see Appendix E) is used for simulations.

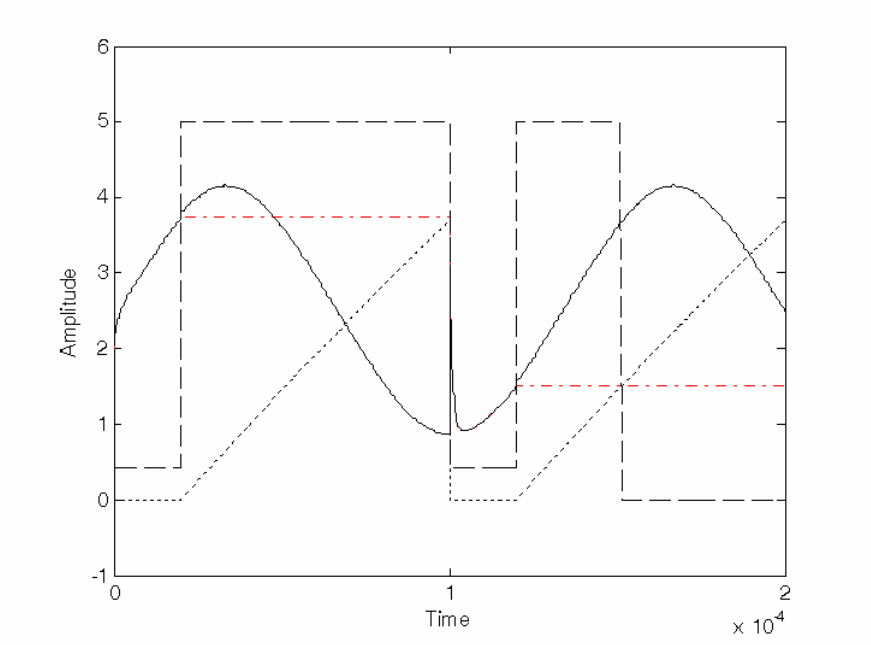

Figure 48: Single-slope ADC extracted circuit HSPICE simulation output

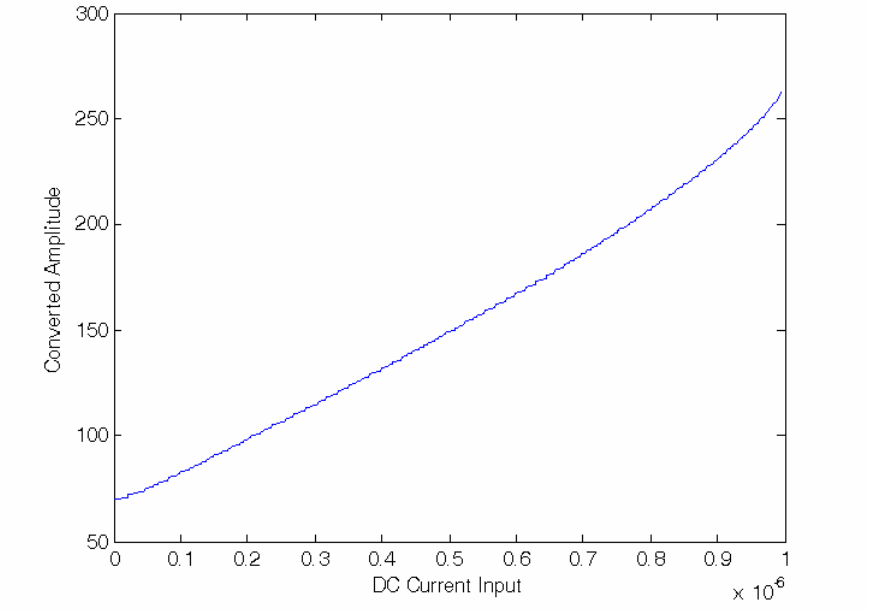

Figure 49: Single-slope ADC DC sweep output

4.5 Conversion Performance Comparison

The theoretical results obtained in the previous chapters are used in this section to provide comparison arguments between designed ADCs. After the architectural comparison of ΔΣ ADC and single-slope ADC is discussed, the algorithmic decoding performances of two ADCs are also compared. Several simulations with varying circuit environments will be presented to demonstrate robustness of ADCs.

4.5.1 Architectural Comparison of ADCs

Let us assume that the input signal bandwidth is 10 Hz. Among several input signal models proposed, the strictly band-limited signal model will be useful to demonstrate the effect of signal band-limitedness. The analog front-ends discussed in Appendix C are used for analog front-end models. All frequencies are normalized to the unit frequency (see Appendix B.1.2 for frequency normalization) to simplify the numbers. A first-order LPF with cut-off frequency 1 is used for ΔΣ ADC anti-alias LPF, and a sixth-order Butterworth LPF with cut-off frequency 0.4 is used for single-slope ADC anti-alias LPF. For the simplicity of comparison, both anti-alias LPFs are assumed to be noiseless, which would favor single-slope ADC. Figure 50 and Figure 51 show available SNR plots of ΔΣ ADC and single-slope ADC. Variables for the plots are input noise power and input signal cut-off frequency of signal model A defined in Appendix C.

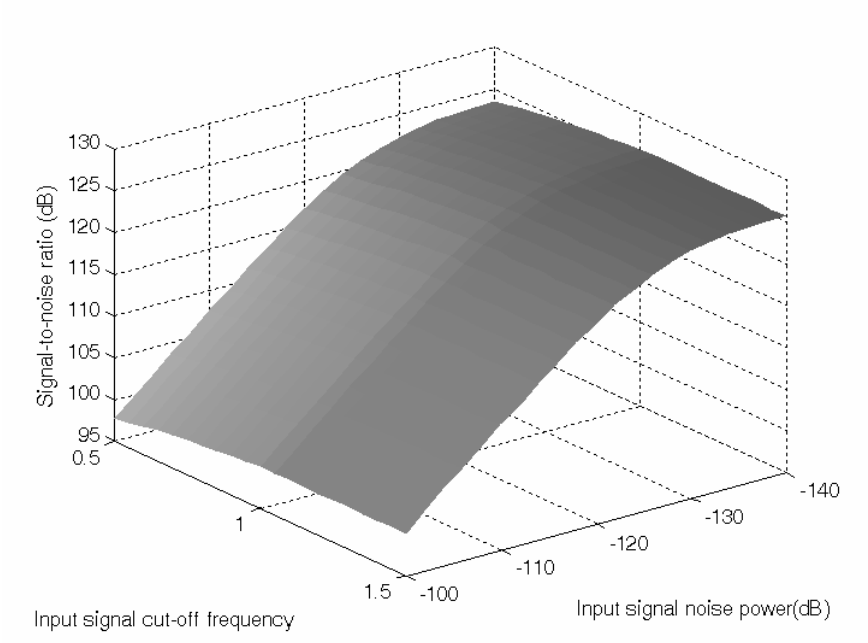

Figure 50: ΔΣ ADC available SNR

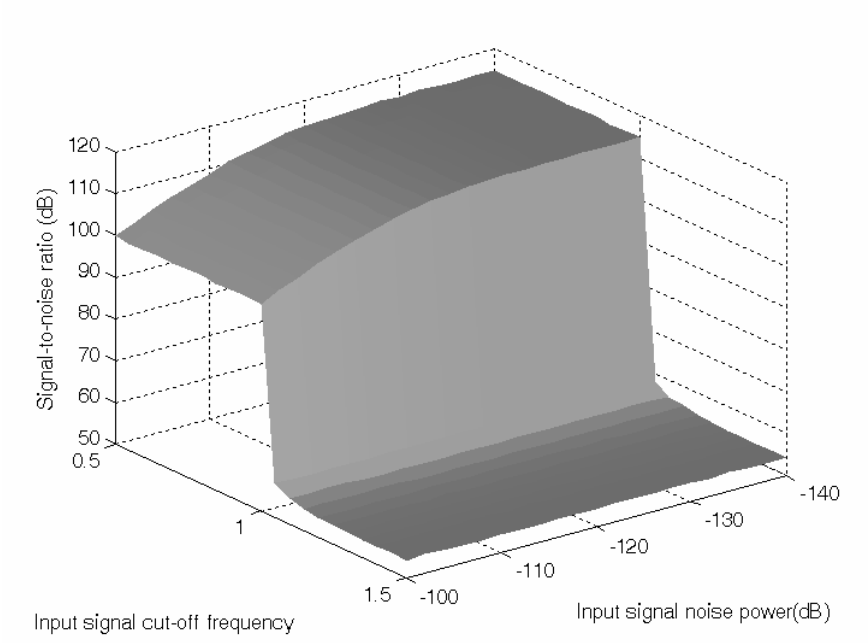

Figure 51: Single-slope ADC available SNR

The plot shows that single-slope ADC suffers extensive loss of SNR due to signal aliasing, even with the sixth-order anti-alias LPF. The amplifier used in the analog front-end of single-slope ADC adds more noise than that of ΔΣ ADC, and the available SNR with optimistic input signal cut-off frequency is worse than ΔΣ ADC. ΔΣ ADC takes advantage of full signal dynamic range by employing simpler analog front-end. Oversampling of input signal enables effective signal aliasing rejection with the simple first-order anti-alias LPF.

4.5.2 Algorithmic Comparison of ADCs

An oversampling ratio for a clocking frequency is equivalent to a given time period of a Nyquist A/D conversion. Both ΔΣ ADC and single-slope ADC have maximum available clocking frequency. The maximum operating clock frequency of single-slope ADC is determined by the operation speed of counter, control logic, and flip-flops, while that of ΔΣ ADC is mainly constrained by comparator and flip-flop in the given designs. The single-slope ADC maximum frequency can be increased by adopting higher-speed adder, sacrificing circuit area, design complexity, and power consumption. For a given oversampling ratio, mean-squared error of data conversion output is a measure of how much sensitivity an ADC has. Figure 52 shows a comparison between ideal single-slope ADC sensitivity and ideal ΔΣ ADC sensitivity.

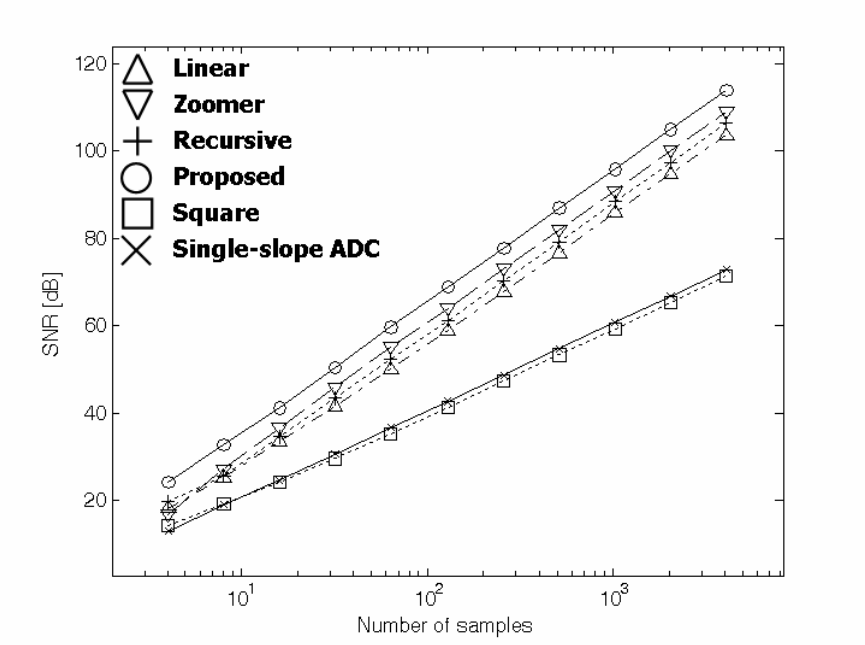

Figure 52: Decoding algorithmic comparison of ADCs

As discussed in the Appendix B, the performance of ΔΣ ADC is highly dependent on the decoding algorithm. A square filter, which is equivalent to first-order comb filter, or averaging filter, is basically a counter, and it performs worse than single-slope ADC by 3dB. The comb filter and single-slope ADC performances are increased by 3dB per twice oversampling. The advantage of ΔΣ ADC is that it has several alternative decoding schemes with various algorithmic complexities. As shown in the plot, the optimal filter [50] and the proposed algorithm (see Chapter 3) outperform single-slope ADC algorithms. The gain of these two schemes increases by more than 9 dB per twice oversampling.

4.5.3 ADC Robustness Comparison with Simulations

The comparisons provided so far used performance measures such as SNR to see how much sensitivity an ADC would produce. The robustness test of an ADC can be demonstrated by changing circuit environments to observe how much the performance and the designed characteristics maintained. Also circuit performance dependency on wafer device model variations should be minimized to enhance yield per wafer.

Figure 53 shows the DC-level acquisition sweep of ΔΣ ADC with temperature sweep from 0 º to 100 ºC. The plot shows highly linear analog-to-digital converted outputs with temperature sweep. It is obvious that ΔΣ ADC is robust against temperature variations.

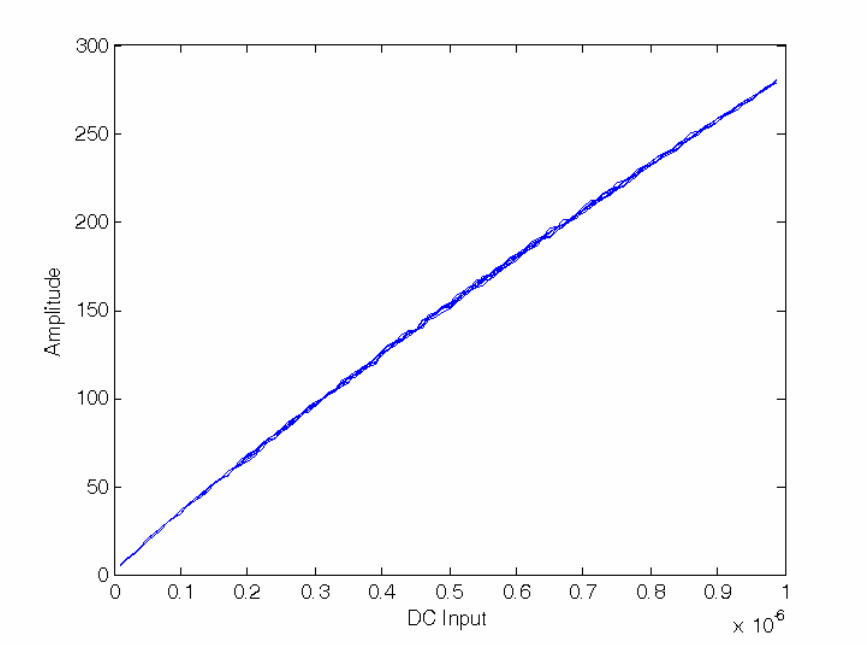

Figure 53: ΔΣ ADC DC sweep with temperature sweep 0-100ºC

Figure 54 shows DC-level acquisition sweep of designed single-slope ADC with temperature sweep of 20 – 30 ºC. The converted output shows a huge drift as temperature increases, and it suggests that the given single-slope ADC design is highly temperature dependent. To enhance temperature independency in single-slope ADC design, a temperature compensation technique should be incorporated, though achieving high-gain, low-noise, stable, and temperature stable circuits would give rise to a complicated optimization problem.

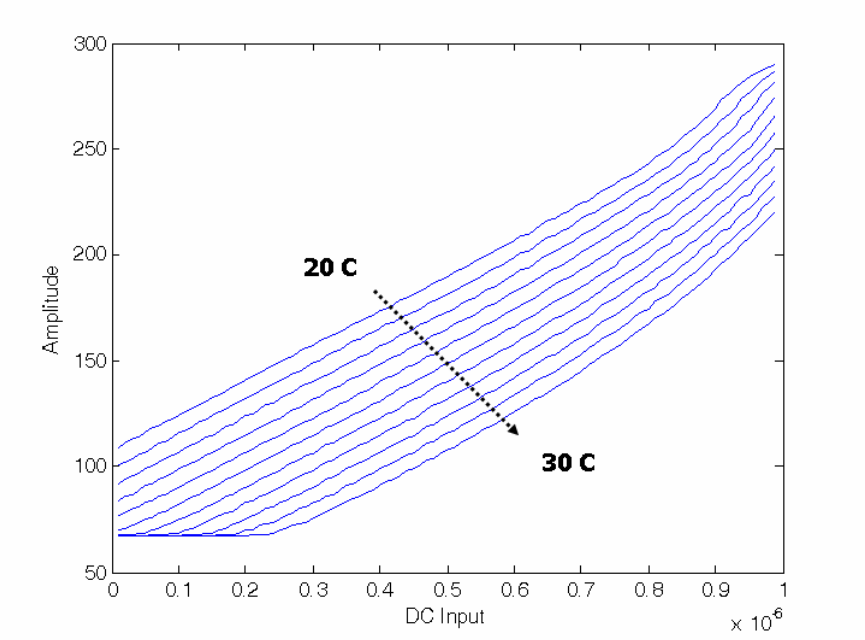

Figure 54: Single-slope ADC DC sweep with temperature sweep 20 – 30 ºC

One device model is not enough to ensure that a circuit design would operate properly with circuit fabrication. Wafer to wafer variations should be taken into account

for a design to be robust and reliable. A set of recent AMI Semiconductor 1.5 μm Si-CMOS process transistor models were obtained through MOSIS [8], and the given ADC designs were simulated with each model. Figure 55 and Figure 56 show DC-level acquisition sweep of ΔΣ ADC and single-slope ADC with 34 recent Si-CMOS transistor models. While ΔΣ ADC shows model invariant DC-level acquisition performance, single-slope ADC performance depends on transistor model extensively. Both temperature and model dependency of single-slope ADC comes from trans-impedance amplifier characteristics. Simulation result suggests that the amplifier should be designed carefully and the biasing of amplifiers should be intelligent.

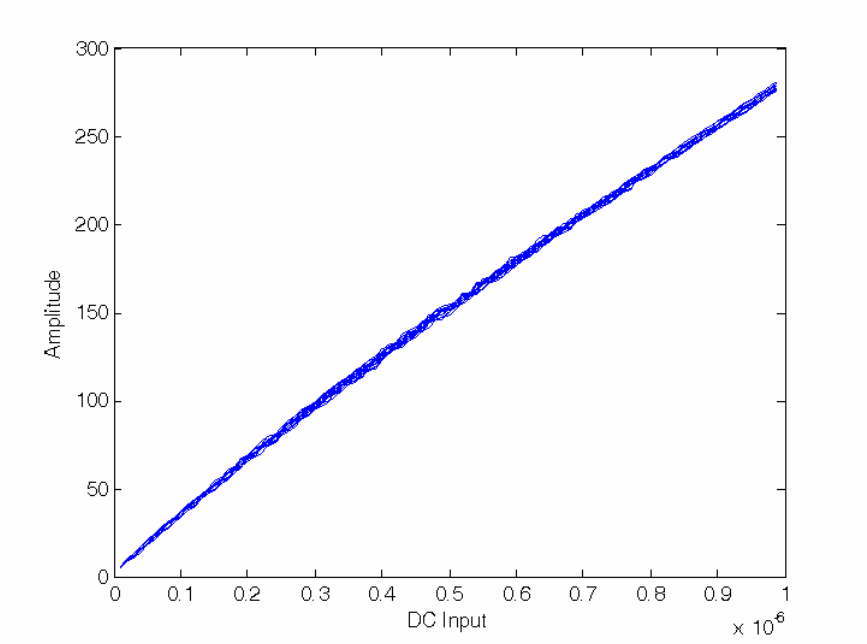

Figure 55: ΔΣ ADC DC-level acquisition sweep with 34 CMOS transistor models

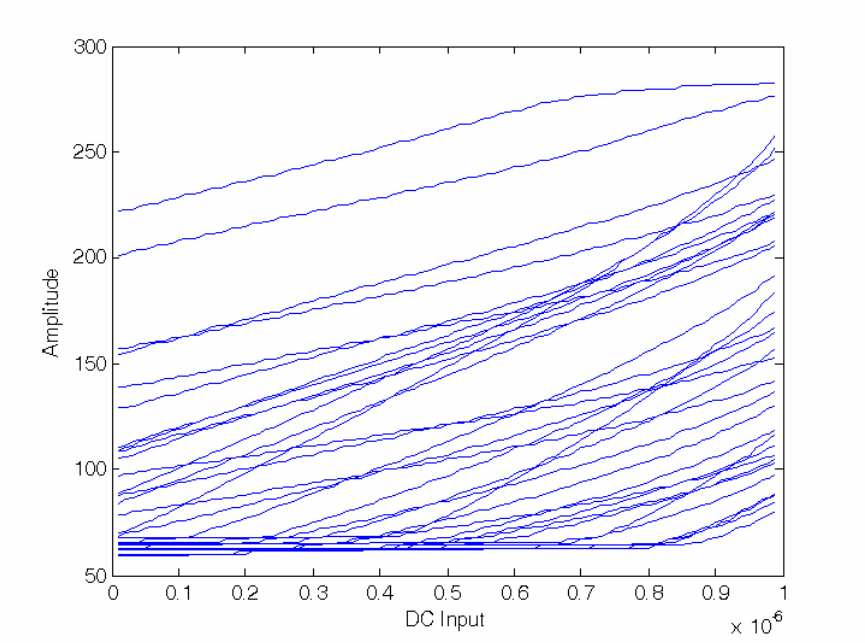

Figure 56: Single-slope ADC DC-level acquisition sweep with 34 CMOS transistor models

Another important issue in a real A/D conversion circuit is clock jitter. With a varying signal, the maximum allowed clock jitter is given as (17) [15]. Using the equation, the reduction in SNR due to the clock jitter can be obtained as (18) [15]. The reduction in Nyquist ADC SNR with clock jitter is plotted in Figure 57.

$$\textbf{(17):}\ \Delta t_{\max} = \frac{2^{-n}}{\pi f_{in}}$$

$$\textbf{(18):}\ SNR_{reduction} = 6.02 \times \frac{\log(1+[2^{n}\pi\sqrt{6}\Delta/T])}{2\log 2}$$

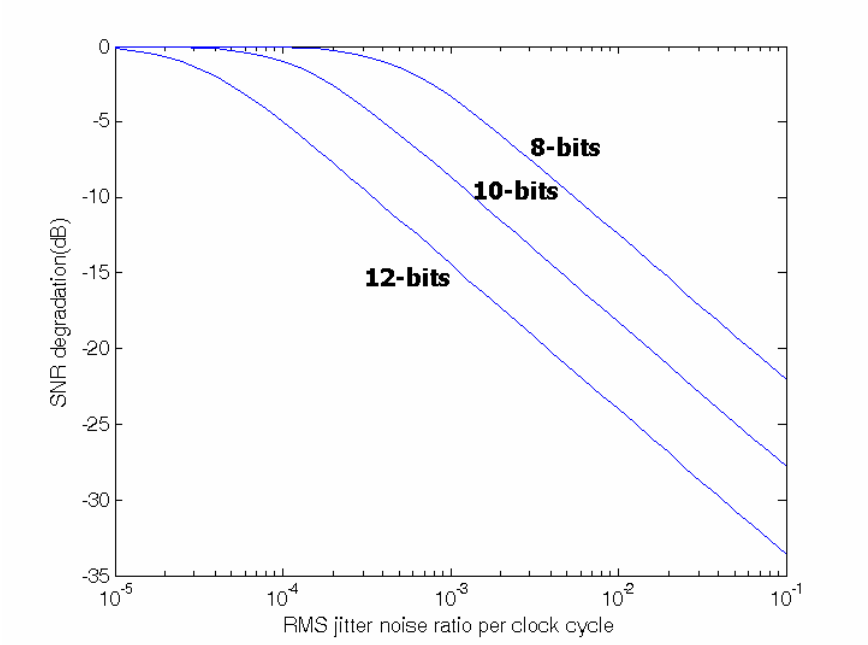

Figure 57: Nyquist ADC SNR degradation with clock jitter

The effects of clock jitter in ΔΣ ADC have been analyzed in several studies, and the solution depends on the definition of jitter and the form of ΔΣ converter [57-59]. To get statistical analysis, a randomized clock jitter is generated and given to a simulation clock input, whose duty cycle is 0.25. An example of randomly generated clock jitter eye diagram is shown in Figure 58. In the eye diagram, the random numbers have Gaussian distribution with zero mean and standard deviation $\sigma = 2\times10^{-2}$.

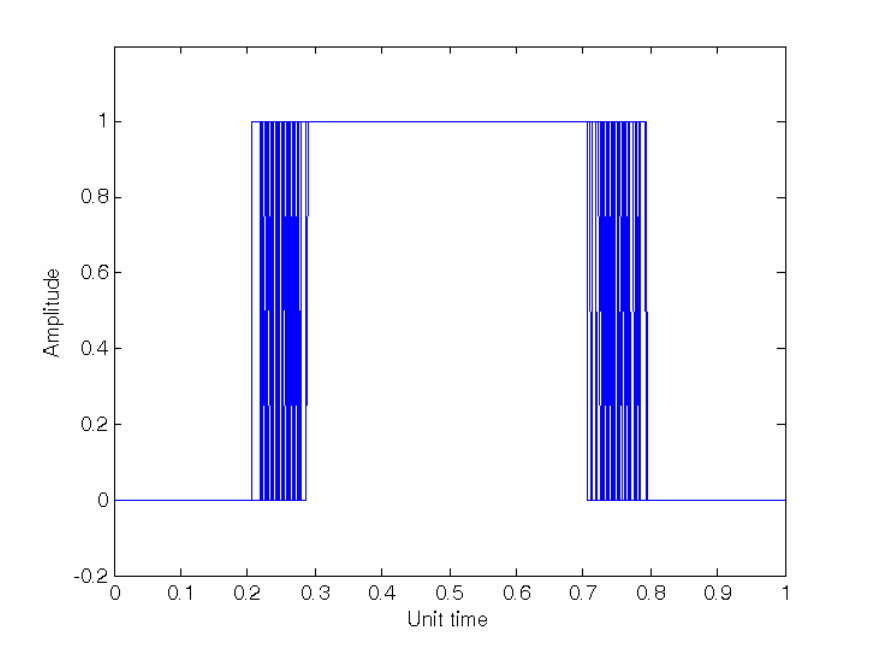

Figure 58: Randomly generated clock jitter example

For a generated Gaussian random number with zero mean and standard deviation σ, the amount of jitter was obtained by multiplying the generated random number by clock period. For example, a random number $x = 2\times10^{-2}$ is generated with given σ and the clock period is $T = 1\times10^{-6}$, which is f = 1 MHz in frequency. Then the amount of jitter given to simulation is $xT = 2\times10^{-8}$. Every fall and rise edge of each clock is jittered randomly and independently in this way. For each given standard deviation σ, 20 sets of randomly jittered clocks were supplied to DC-level acquisition sweep simulations to get statistical data. The simulation clock period was 0.5 μsec and each simulation was 280 clock cycles long. Each output was filtered with a sinc filter to estimate input value. For each σ, mean-squared error was obtained and Figure 59 shows the simulation output. The

range of output is [0, 2^8-1=255] for the plots. The 8-bits resolution Nyquist sampling ADC mean-squared error (MSE) plot with clock jitter is also given in the figure for comparison. The figure shows that ΔΣ ADC is less affected by jitter, and the converter resolution is not degraded in the given RMS jitter noise range since MSE remains below 0 dB. There will be a rapid performance degradation with ΔΣ ADC when a greater jitter noise is applied [57, 58].

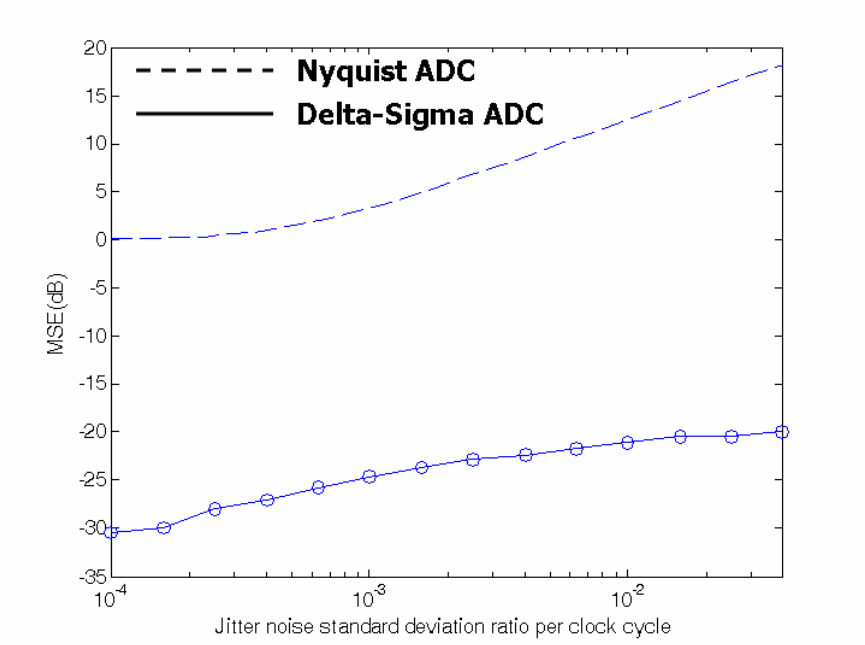

Figure 59: ΔΣ ADC available SNR with jitter noise

4.6 Sensor System-on-a-Chip Fabrication and Integration

A SoC with embedded Si-CMOS photo detector arrays and integrated mixed signal processing system was fabricated with AMI Semiconductor 1.5 μm standard Si-CMOS process through MOSIS, as shown in Figure 60.

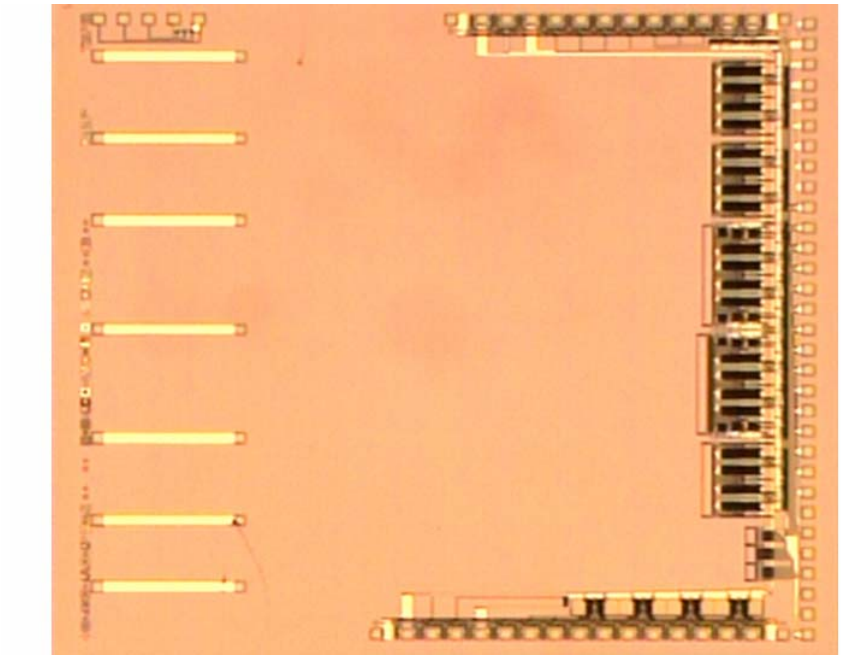

Figure 60: Fabricated chip photo

The chip size is 4.6×4.7 mm^2 and it has several variations of detector arrays such as PiN and BJT photo detectors. Data converters included in the chip are 8 parallel ΔΣ ADCs. Several configurations for photo detectors were designed on the chip, and the photo detector pixel size is 8×8 μm^2. Each array has 24 or 48 photo detectors and the

width of the array is about 192 μm. Each photo detector has its own analog front-end with a switch to turn it on and off. The addressing of the photo detector can be described as shown in Figure 61, where a group of 8 photo detectors are connected to 8 parallel analog-to-digital converters with 8 parallel electrical switches.

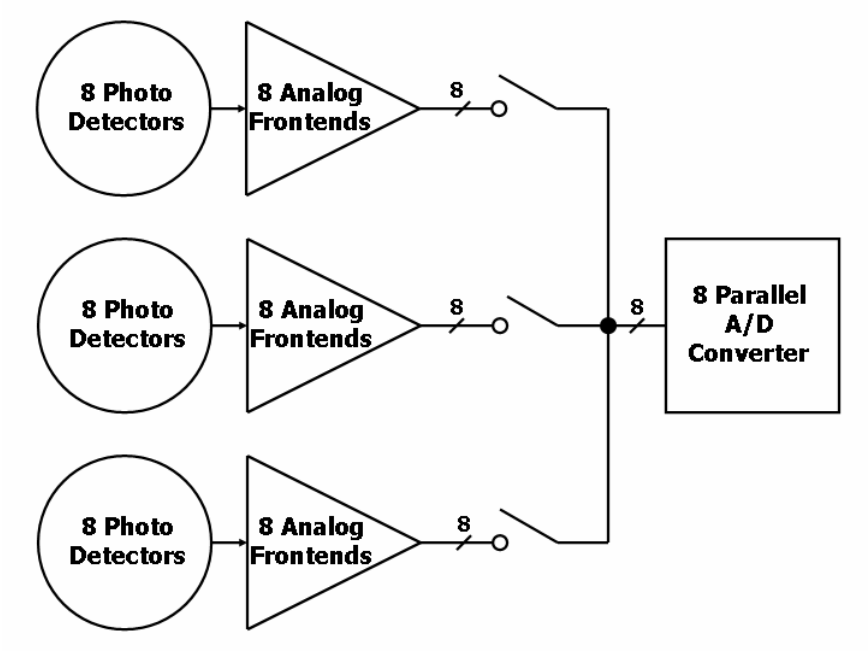

Figure 61: Photo detector array addressing diagram

Optical interferometric waveguides were integrated using $SiO_2/Si_3N_4/SiO_2$ with PECVD on the Si-CMOS circuit, as shown in Figure 62. Figure 63 shows the coupling of external laser into the waveguide. Sensing arm of waveguide is patterned with Hexafluoroisopropanol (HFIP). With several selected materials and knowledge-based

pattern recognition algorithm, this interferometric waveguides and the circuit can be a sensor system for aqueous and gaseous materials.

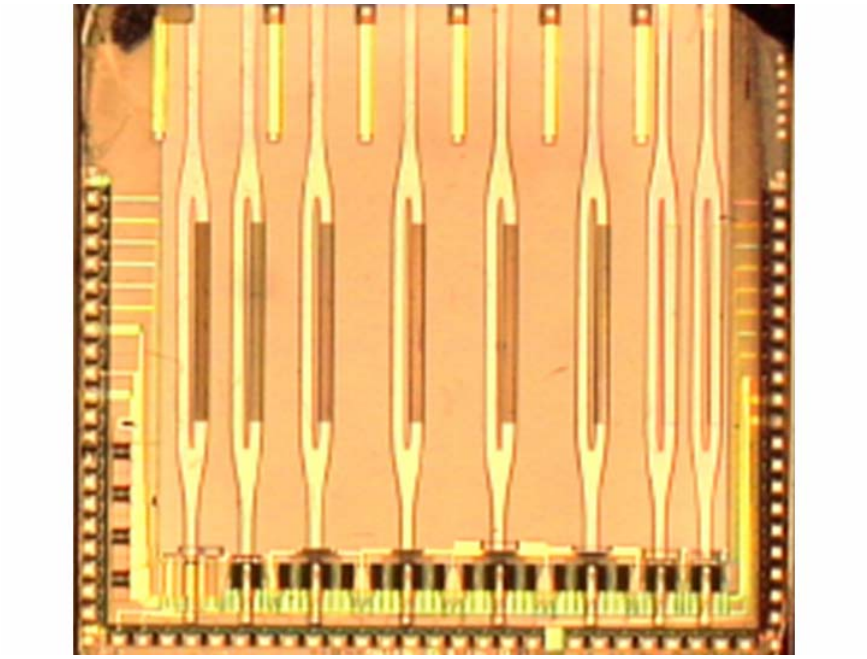

Figure 62: Integrated Mech Zender interferometric waveguides

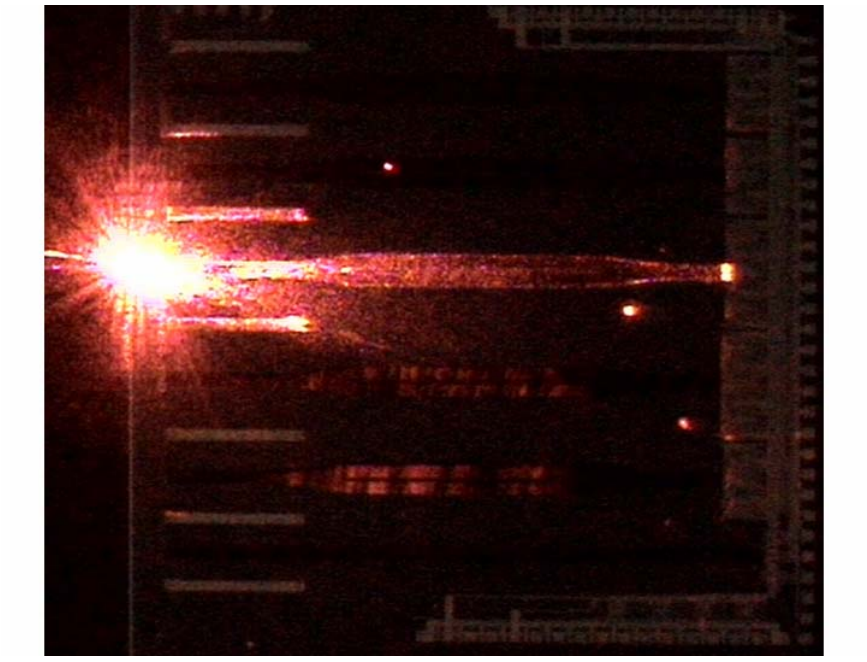

Figure 63: Laser coupling into integrated waveguide

4.7 ΔΣ ADC Sensitivity Measurement

The fabricated chip can take an electrical current input instead of detector current input through the selection switches. A clock signal is generated with Tektronix data pattern generator DG2020A [60], and the output data is collected by National Instrument PCI-DIO-32HS data acquisition unit [61]. All the circuit-biasing voltages and currents are supplied by Keithley 236 source-measure unit [62]. Also electrical current input is controlled by two separate source-measure units. Beginning at 0.1 μA, differential currents are added to input. The collected output through first-order comb filtering is shown in Figure 64.

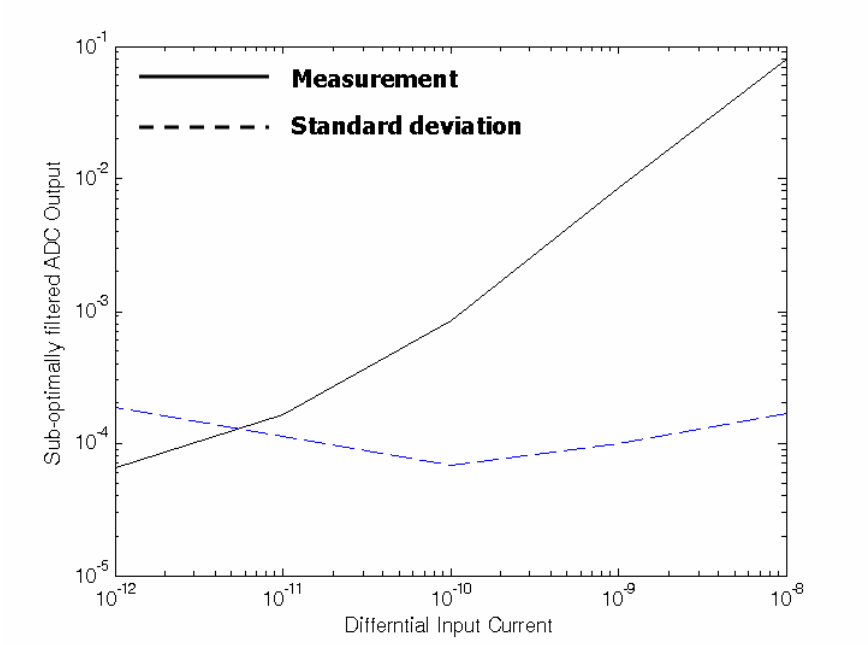

Figure 64: ΔΣ ADC sensitivity measurement with electrical input

The measurements are taken 10 times per input level, and the averaged input and standard deviation are plotted in the figure. According to plot, the fabricated ΔΣ ADC circuit is able to sense current down to the order of 10 pA against noise. The noise power is inferred through standard deviation, assuming the noise is Gaussian. The noise is not necessarily circuit noise since the accuracy of source-measure unit is ±(0.21 %+20 pA) at 100 nA, which is equivalent to ±230 pA. The measured sensitivity is better than reported current input sensor array applications [63].

Since the actual signal reaches the analog front-end through the photo detector, optical sensitivity measurement would provide the actual sensor system sensitivity. Laser attenuated through optical attenuator was coupled into single mode optical fiber, and then focused onto a photo detector. The optical power came out of the single mode fiber was

measured with Newport 1835-C calibrated optical power meter [64]. The collected data was filtered through first-order comb filter and the result is plotted in Figure 65.

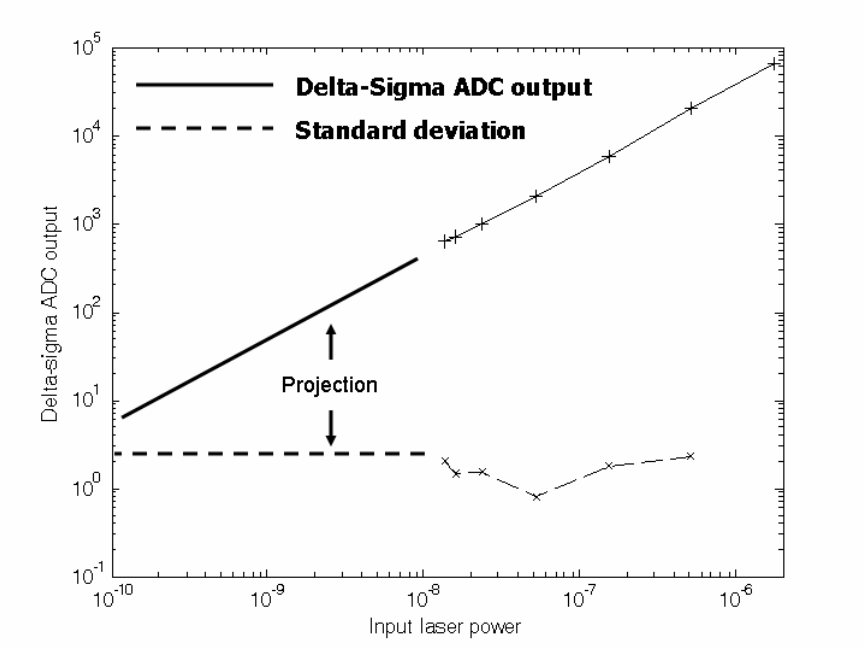

Figure 65: ΔΣ ADC sensitivity measurement with optical input

The plot shows that the fabricated data converter circuit can detect less than 10 nW laser input to detector. One thing to be mentioned is that not all optical power is actually coupled into photo detector from optical fiber since it is coupled through air. According to the derived standard deviation, the plot suggests that ΔΣ ADC sensor system would be able to sense down to 100 pW laser power.

4.8 ΔΣ ADC Sensor System Measurement

A complete SoC sensor system requires the integration of various sophisticated optical components such as edge-emitting laser, interferometric waveguide, etc. Not only does each component require cutting-edge technology, but also the components integration itself requires challenging processes. A demonstration of sensor SoC was performed with the fabricated chip and the integrated interferometric waveguide. The sensor system testing set up diagram is shown in Figure 66.

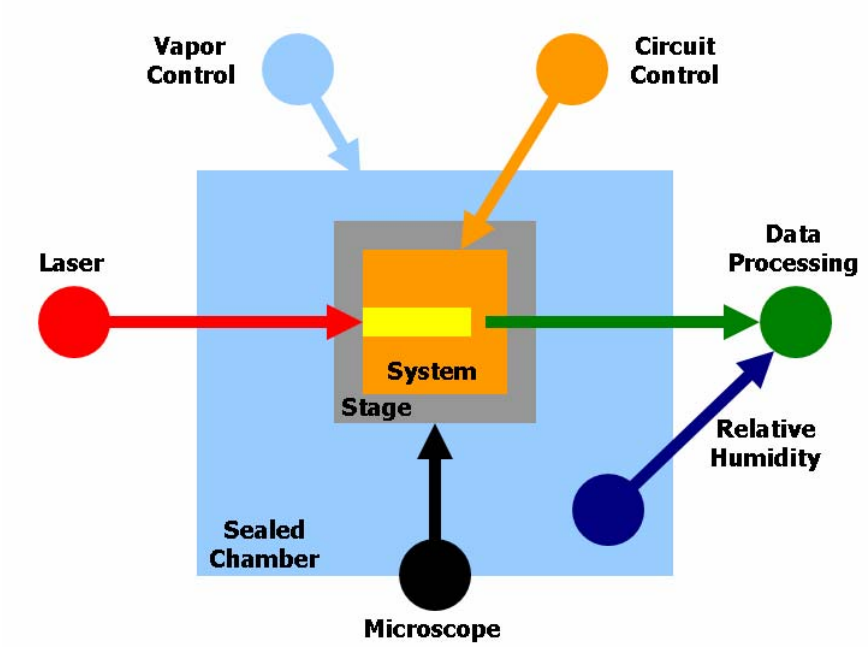

Figure 66: Vapor sensor system block diagram

The waveguide integrated chip was placed in a sealed chamber, and HeNe laser was coupled with lens into the waveguide. The optical alignment was achieved through microscope and 3-dimensional stage. Precise concentrations and flow rates of volatile analyte chemicals were delivered to the test chamber with nitrogen gas carrier stream, which is vapor in this case. The circuit output and relative humidity was collected by a data processing unit. With the given set up, vapor was turned on and off at 20 minutes and 75 minutes from the beginning of experiment, and the processed data is plotted in Figure 67. In the plot, the explicit changes of output are observed with slow varying noise, which can come from laser drift, thermal expansion of metal stage, epoxy and FR4 board responses to vapor, and low-frequency circuit noise, etc..

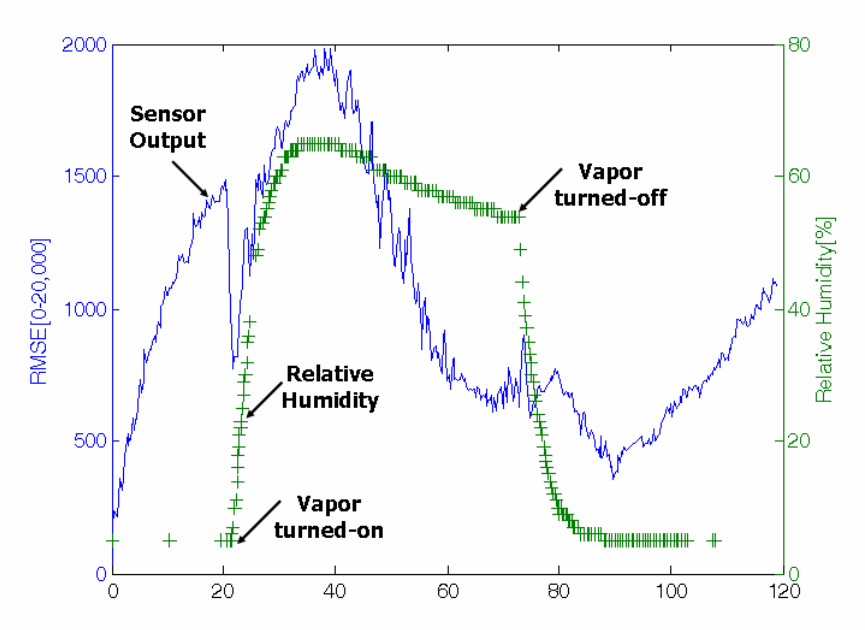

Figure 67: Vapor sensor system measurement

4.9 Summary of Case Study

The sensor SoC integration requires optical components, such as waveguide, laser, and photo detector. Integration of these components requires specialized know-how and considerations to be compatible with each other. The comparison of designed ADCs shows that there are more benefits with ΔΣ ADC than with single-slope ADC in circuit area, robustness, and sensitivity. The fabricated ΔΣ ADC was tested for sensitivity, and it proved a vapor sensor SoC successfully with integrated interferometric waveguide.

Chapter 5

CONCLUSION

5.1 Future Research

As shown in the sensor system demonstration in Chapter 4, a fully integrated sensor SoC could be attained if a laser, such as an edge-emitting laser, were integrated on the chip. It will remove all mechanical noises involved in the testing set up, while the noise of integrated laser would remain an issue to go over. A laser driver that biases the laser also contributes noise to the laser. The laser and driver noise modeling will reveal whether 1/f noise of light source can be ignored, or whether flat spectrum noise of light source is low enough to provide enough sensitivity [65]. In case 1/f noise is a major noise source for the measurement and the flat noise spectrum is low enough that a better sensitivity is available at higher sampling frequency, the lock-in operation of signal source and ADC can be useful to achieve better sensitivity [66, 67].

There are various ways to implement the lock-in amplifier. The oscillator can be implemented with analog or digital circuitry. The receiver demodulation of received signal also can be performed in analog or digital domain. As a SoC, with light source driver and receiver demodulator on the same system, full phase information is available.

Therefore, the phase-locking of transmitter and receiver can be established easily. While the spectral purity of oscillation can be a problem, digital sources are useful to produce almost infinite Q-factor oscillation as long as a good filtering is provided [68].

ΔΣ modulation concept provides a wide variety of choices for lock-in mode sensor operation. Band-pass ΔΣ [69, 70] ADC and high-pass ΔΣ [71, 72] ADCs enable a very simple demodulation in digital domain. Also the signal transmitter section can be implemented with high-fidelity ΔΣ DAC [73]. To achieve higher-frequency operation, since the switched-capacitor technology has a speed limit, continuous-time filters should be used for ΔΣ converters. Though radio frequency continuous-time filter experiences more circuit noise than switched-capacitor technology, several breakthroughs, such as on-chip integrated commercial level wireless transceivers and band-pass ΔΣ converters, have been reported [68, 74].

5.2 Conclusion

This dissertation presented the frequency domain modeling of the analog front-end of ADC and the time domain modeling of ADC as an encoder and decoder. The analog front-end modeling of ADC provided the architectural comparison of various ADC configurations, and it also proved the statistically worst-case available ADC performance with stationary process assumption on input signal. The modeling produced

the quantitative analysis of analog front-ends, and it showed the architectural advantage of oversampling data converters.

The perspectives that an ADC can be modeled as a communication channel or a CODEC provided a general ADC model for both Nyquist sampling ADC and oversampling noise-shaping ADC. Also a nonlinear decoding algorithm for ΔΣ ADC was proposed, and the algorithm performance was demonstrated. The proposed method outperformed known linear and nonlinear decoding schemes.

A decoder-centered ADC and an encoder-centered ADC were designed for SoC application, and both were analyzed with the proposed theoretic models and tools. Also the robustness and flexibility of the designs were examined. The ΔΣ ADC design was fabricated with embedded photo detector arrays through the Si-CMOS process. The electrical and optical sensitivities of the circuit were measured. Also a sensor SoC was demonstrated with integrated interferometric waveguide.

Appendix A

DECODER-CENTERED ADC

A.1 Single-Slope Conversion

The single-slope serial ADC and dual-slope serial ADC can be selected as comparison standards since they have the simplest concepts and implementations. In SoC sensor applications, this type of converter might be used because of its small size and low power consumption.

Single-slope ADCs search linearly from the lowest value to the highest value, or vice versa, to determine which digital code would represent the input analog value best. The single-slope ADCs have many different implementations and a basic functional block diagram is shown in Figure 68. The integrator and counter are both set to zero output as initial conditions. When an input value is sampled and held, the counter starts and the integrator collects a small reference value until the integration output exceeds the input value as shown in Figure 69. The counter output is the A/D converted value.

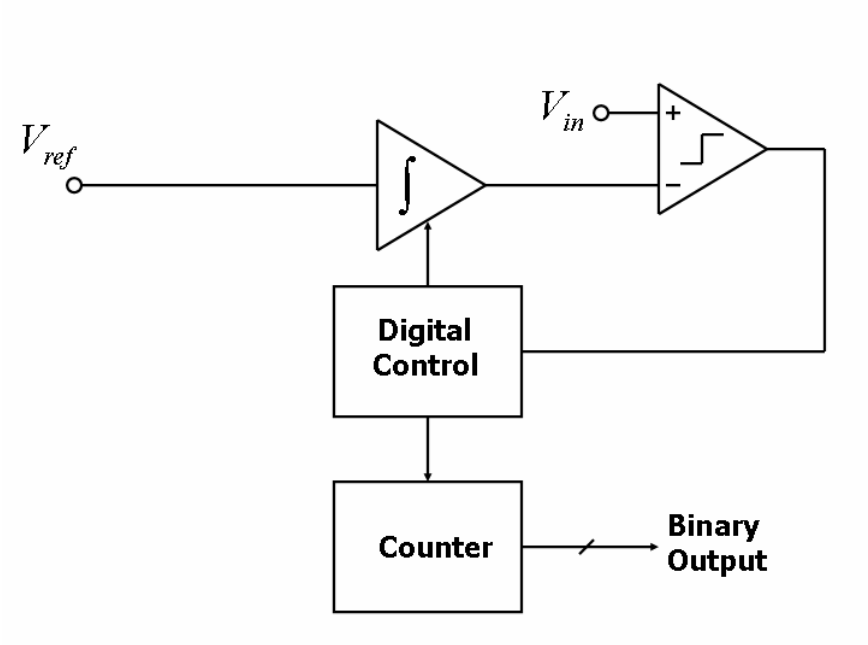

Figure 68: Single-slope ADC function block diagram

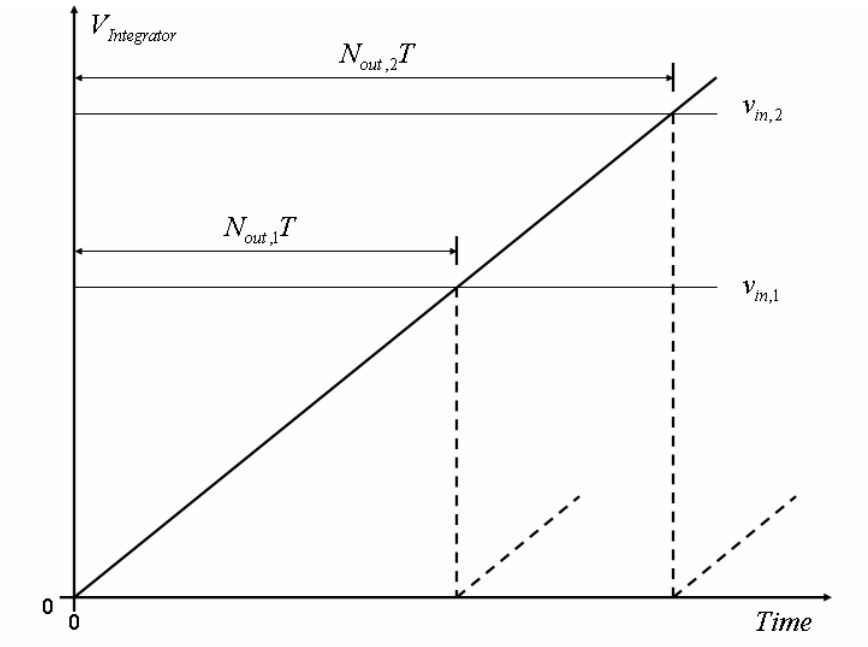

Figure 69: Single-slope ADC integrator output waveform

The single-slope converter is the simplest and the most intuitive ADC. The major components are sample-and-hold, integrator, comparator, counter, and control logic. The conversion is slow since it is a linear search process. The precision required of the integrator, reference value, and comparator will dominate conversion speed and sensitivity. The integration of reference would band-limit noise to reduce the effective noise. The worst case conversion time is LT, where $L=V_{supply}/V_{ref}$ is the number of countable levels and T is the clock period [11, 12, 15, 17].

There are many examples of use of this design in the literature due to its simplicity and compactness in spite of the limited resolution and slow conversion time. Implementations are found in CCD or CMOS image sensors and multichannel parallel converters. The reported resolutions are 8 - 12 bits with 10 - 128 μsec conversion time [9, 37, 41]. Table 7 summarizes single-slope ADC implementation examples.

Table 7: Single-slope ADC implementation examples

Resolution (bits)	Conversion Time (μsec)	Application
12	10	16 Channel Parallel ADC [9]
8	128	32 by 24 pixels CCD [41]
8	16.3	128 by 128 pixels CMOS [37]

A.2 Dual-Slope Conversion

The concept of dual-slope conversion is identical to single-slope conversion, except that it uses upward and downward slope at the same time to reduce the effect of nonlinearity and inaccuracy of the slope. Figure 70 shows the functional block diagram of dual-slope converter and Figure 71 shows the integrator output waveforms of dual-slope converter.

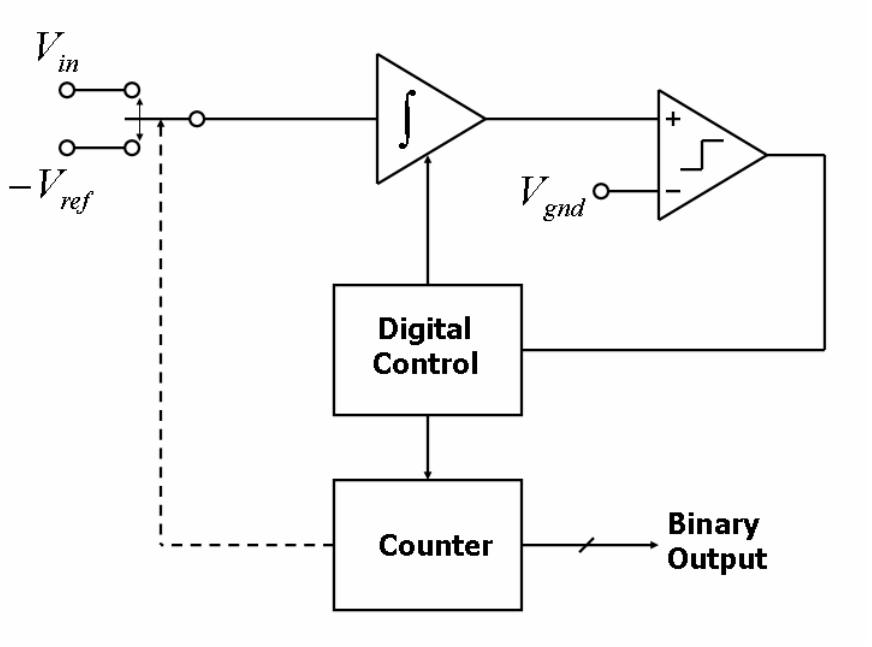

Figure 70: Dual-slope ADC function block diagram

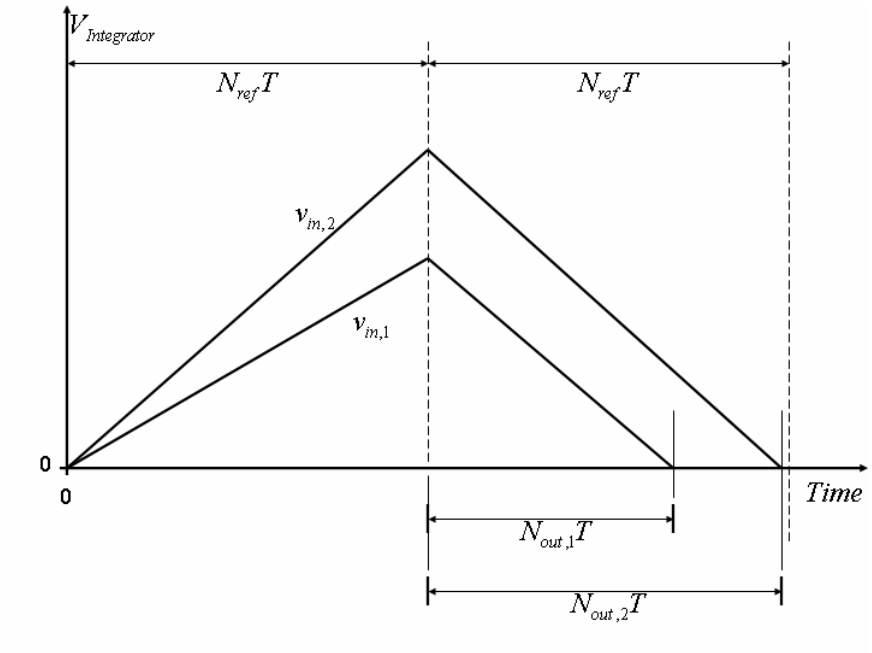

Figure 71: Dual-slope ADC integrator output waveform

After the input value is sampled and held, the integrator output is set to V_{gnd} to initialize it. Then V_{in} is integrated for N_{ref} clock cycles, after which, the counter is reset to 0, and $-V_{ref}$ is applied to integrator input instead of V_{in}. The integration of $-V_{ref}$ continues until integrator output reaches V_{gnd}, while the counter is counting clock cycles. The input V_{in} estimation can be found from the counter output N_{out}, as in (19) [11-13, 15, 17].

(19): $$V_{in} = \frac{N_{out}}{N_{ref}} V_{ref}$$

It has been shown that the analog integration inherent in the slope integration conversion process reduces the effect of noise. The major source of conversion error in dual-slope ADC is the offset error in the integrator, and there are several algorithms to reduce or cancel the offset error [75, 76].

Dual-slope conversion is widely adopted for the precise instrumentation applications. The literature contains examples demonstrating 12 - 16 bits resolution and a variety of conversion times depending mainly on the IC fabrication process chosen [10, 42, 77, 78]. A summary of implementation examples is shown in Table 8.

Table 8: Dual-slope ADC implementation examples

Resolution (bits)	Conversion Time	Application
12	50 μsec	Medical Instrumentation [10]
12	0.41 μsec	Micro-instrumentation [42]
14	15 μsec	A/D and D/A Calibration [77]
16	2.5 μsec	Time-to-digital Converter [78]

A.3 Iterative Algorithmic Conversion

The iterative algorithmic ADC uses a binary search process to determine a digital code that best represents analog input value. It has been used to implement ADC since the 1960's, and partial and full integrations were completed in 1970's. It is the most time-efficient algorithm after the flash type converters [79-82].

A functional block diagram of an iterative algorithmic ADC is shown in Figure 72. Initially, the input voltage V_{in} is sampled and held for the comparator, which will generate a comparison result with V_{ref}. The error between the sampled value and the comparator output is doubled and sampled for the next iterative operation cycle.

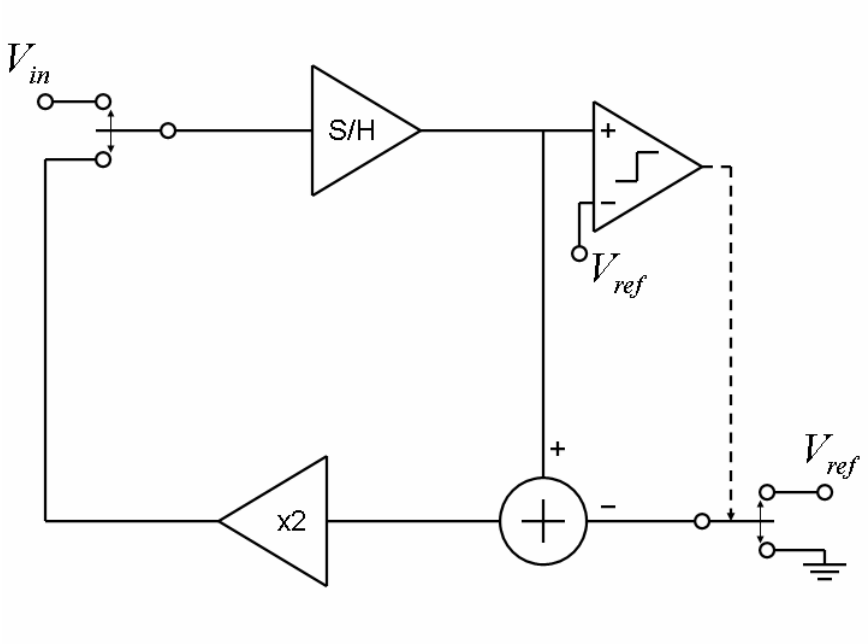

Figure 72: Iterative algorithmic ADC function block diagram

With this scheme, a small error in the earlier cycle propagates with multiplication through the cycles and causes error consequently. Error is caused by the mismatch of passive components such as capacitors, the offset of comparator and multiplier, and also by the inaccuracy of multiplication, subtraction, and sample-and-hold circuit.

The iterative algorithm conversion shares the same concept with pipelined algorithmic ADCs and successive approximation ADCs. The pipelined ADC is an unfolded version of iterative algorithm conversion in time. It is appropriate for a higher-speed application. The successive approximation ADC does not need multiplication nor subtraction, and it requires longer conversion time and an accurate DAC [11, 12, 15, 17]. Because of the large area of the pipelined and successive approximation converters these are not considered for the current small area SoC sensor application.

The implementations of iterative algorithm conversion in the literature show 9 – 14-bits resolutions with various sampling rates depending on the process feature size. In many cases, major concern was to take care of the mismatch problems to get higher resolution [38-40, 83]. There has been intensive research to enhance the resolution of iterative algorithmic ADC through self calibration and dynamic element matching [84, 85]. Generally speaking, the conversion speed of this type of converters is faster than the linear algorithms such as single and dual-slope converters, and the resolution is lower that that of dual-slope converter. A summary of implementation examples is shown in Table 9.

Table 9: Iterative algorithm ADC implementation examples

Ref.	Resolution (bits)	Sampling Rate	Area	Power	Process
[83]	12	8 kHz	1.55 mm^2	17 mW	5 μm CMOS
[39]	10	40 MHz	3.99 mm^2	85 mW	0.8 μm CMOS
[40]	9	5 MHz	5.48 mm^2	180 mW	3 μm CMOS
[38]	14	10 MHz	19 mm^2	219 mW	0.8 μm CMOS

Appendix B

ΔΣ ENCODER-CENTERED ADC

B.1 ΔΣ Encoder

The ΔΣ CODEC is used because it has a quantization noise shaping property that enables highly accurate low-speed converters in conjunction with oversampling and digital signal processing. Noise shaping is a technique to relocate quantization noise power to a disposable (high) frequency band to increase converted SNR in the signal band. Oversampling is a concept that uses a higher sampling rate than the necessary Nyquist rate to enable a digital filter to enhance the SNR of an ADC by removing high frequency out of signal band noise such as relocated noise due to noise shaping [13, 15, 17, 26, 29].

The origin of the ΔΣ modulation concept goes back to 1950's and the development of the ΔΣ has gone through several innovations [22, 26, 86-89]. One of the earlier forms of ΔΣ conversion can be found with delta modulation, an encoding scheme for communication channel that uses oversampling and differential encoding by a single

quantization step as shown in Figure 73. It has been shown that this method suffers from slope overloading and granular noise effects as shown in Figure 74 [90].

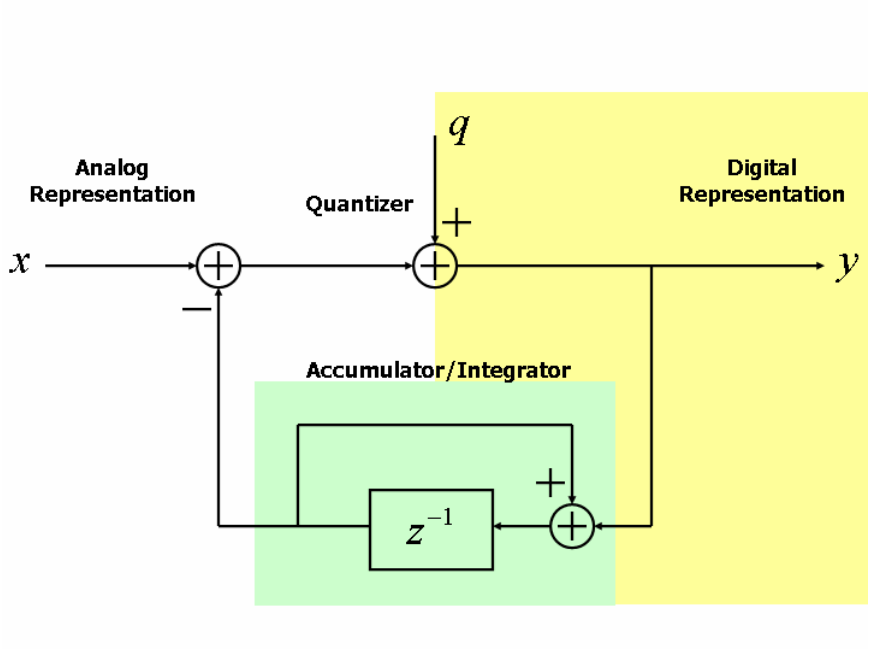

Figure 73: Delta modulation signal flow diagram

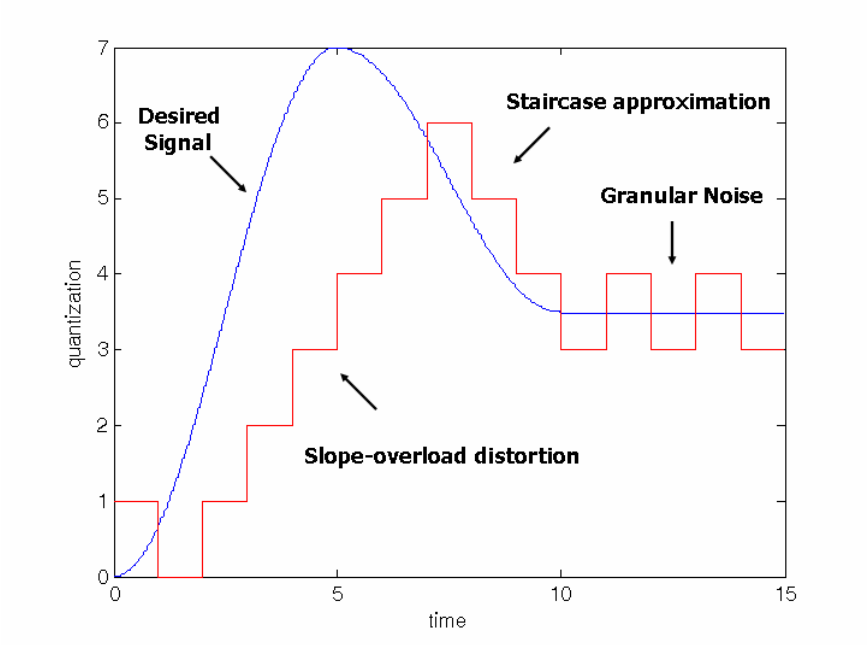

Figure 74: Delta modulation output waveform

B.1.1 Quantization Noise Models

The quantization noise spectrum can be shown to be approximately flat for high-resolution converters. The root-mean-squared (RMS) value of quantization noise is $q_{rms}=\Delta/\sqrt{12}$, where Δ is a quantization step size of ADC. All of the quantization noise power will be mapped on the entire spectrum $[-f_s/2, f_s/2]$ when a signal is sampled with the sampling frequency f_s. The power spectral density of the sampled quantization noise is given in (20).

$$\textbf{(20):}\ Q(f)=\frac{q_{rms}^2}{f_s}$$

There are several assumptions inherent in (20). It assumes the signal is random and uniformly distributed, that there are enough levels of quantization to prevent saturation, and that the quantization step size Δ is small. Examples of quantization error of sinusoidal wave with different number of quantization levels in time and frequency domain are given in Figure 75 and Figure 76. A rigorous mathematical analysis without any assumptions on noise also verifies that the approximation is good enough for many cases [26, 91-93].

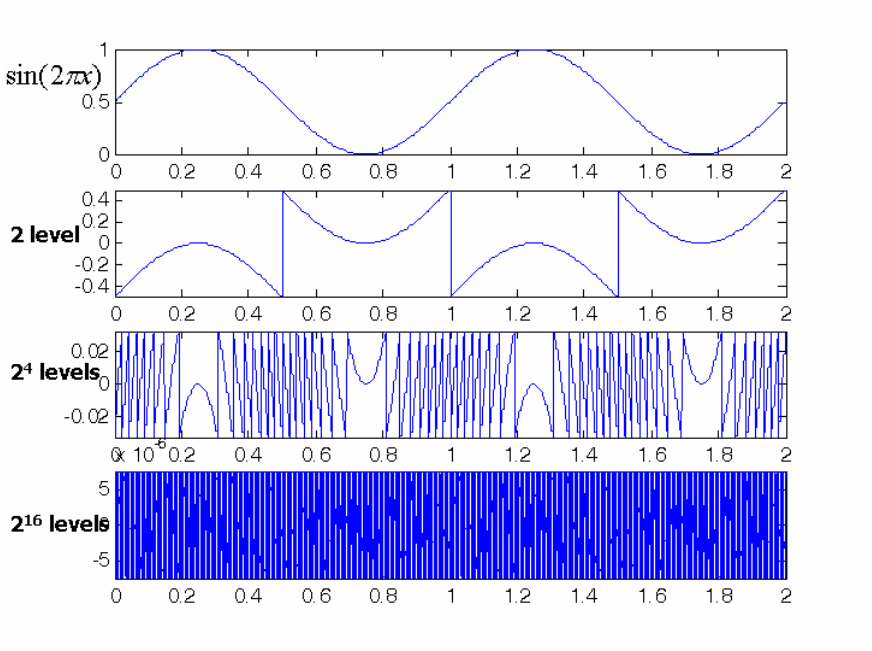

Figure 75: Quantization error in time domain with different quantization levels

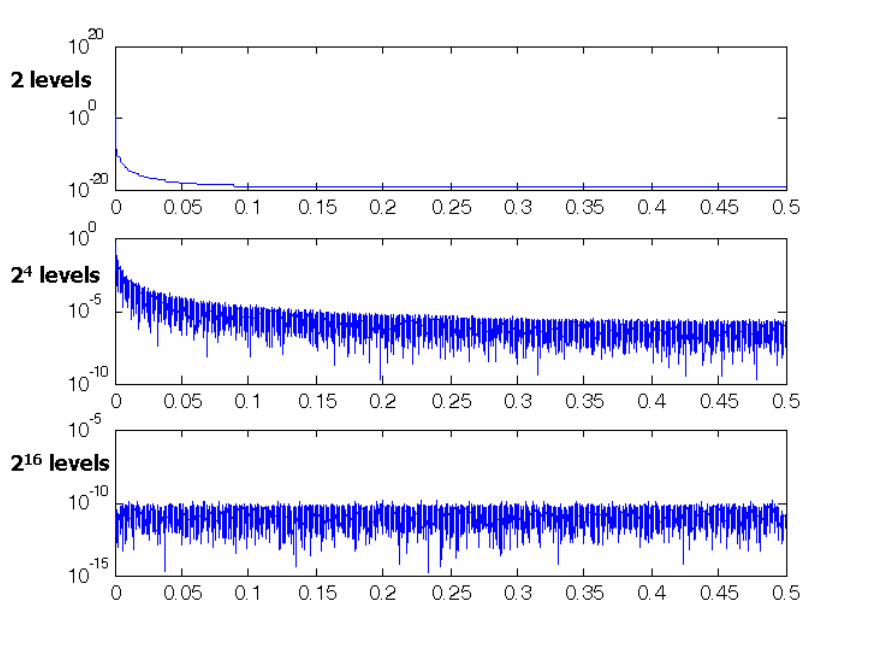

Figure 76: Quantization error in frequency domain with different quantization levels

B.1.2 Oversampling

A frequency response of a system or a power spectral density of a signal can be normalized to an arbitrary frequency for a fair comparison between the systems or the signals. The normalization can be done to a frequency f_N or to a unit frequency 1. The unit frequency will be more intuitive than the f_N in many cases since it reduces a variable in the expressions and plots, as shown in Figure 77. For a continuous-time signal band-limited to f_M, the sampling frequency f_s must satisfy $f_s \geq 2f_M$ for a lossless reconstruction of original signal according to the Nyquist theorem. As far as the sampled data keeps sampling period information, it remains in sampled frequency domain. If sampling time information is removed, sampled sequence can be aligned to unit time, therefore the

frequency of signal is normalized to discrete-time frequency as shown in Figure 78 [94, 95].

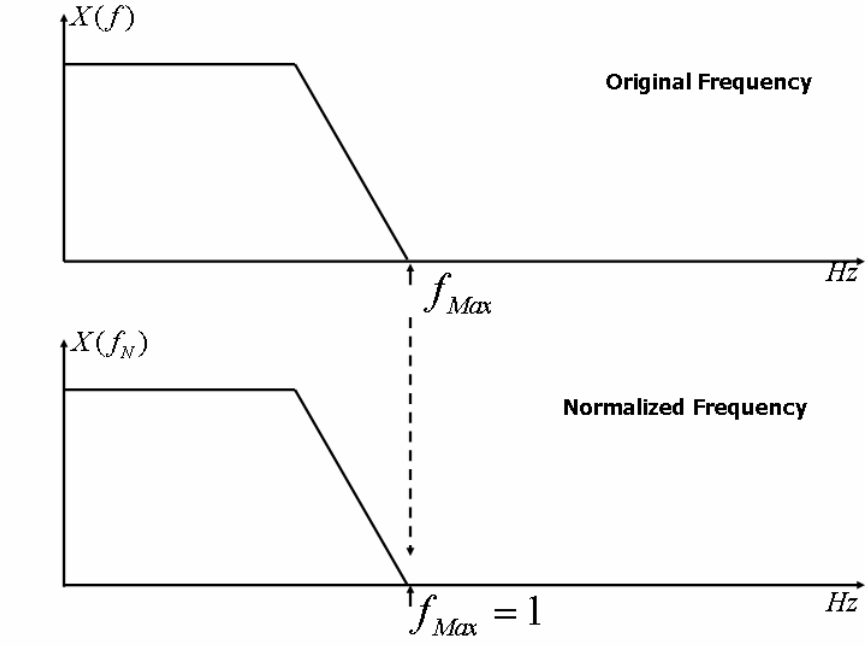

Figure 77: Normalized frequency concept diagram

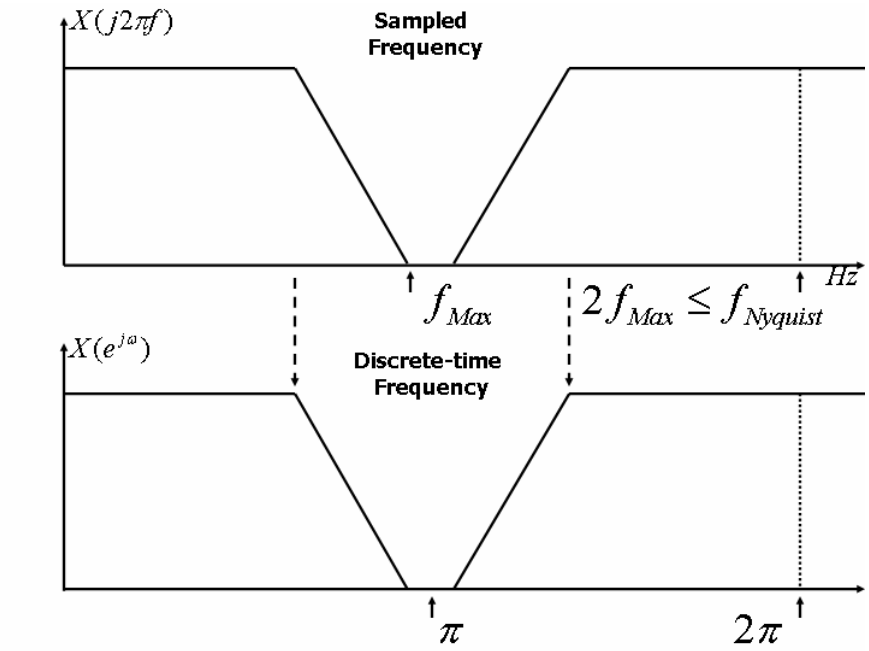

Figure 78: Sampled frequency concept diagram

Let's assume an ideal ADC that has a fixed number of quantization levels and does not have a conversion speed limit. Further, assume the input signal of interest is strictly band-limited to f_M and Nyquist sampling frequency is given as $f_N=2f_M$. If the signal is sampled at the Nyquist rate then the noise will be as in (20), however if a higher sampling rate is used then, by (20), the quantization noise power spectral density will be reduced by 1/2 (3dB) every time the sampling frequency is doubled, as a result the in-band signal-to-quantization noise ratio is increased by 3dB every time the sampling frequency is doubled, as long as the out-of-band noise power is later removed by digital filtering, as shown in Figure 79.

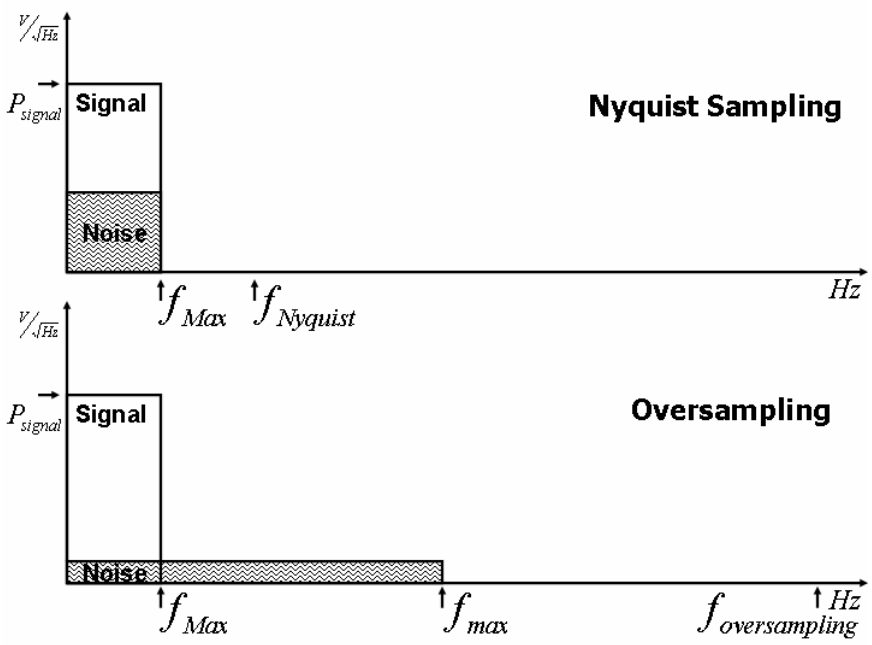

Figure 79: Oversampling concept diagram

B.1.3 ΔΣ Encoding Analysis

A block diagram of the first-order ΔΣ noise shaping converter is shown in Figure 80. The 'Δ' comes from the negative feedback of the output and the 'Σ' comes from the inner integrator. Both ΣΔ and ΔΣ are commonly used at the same time in the literature [26, 29]. It differs from the delta modulator mainly in the placement of the integrator. The location of the integrator in the analog domain before the quantizer makes for a simple encoder implementation.

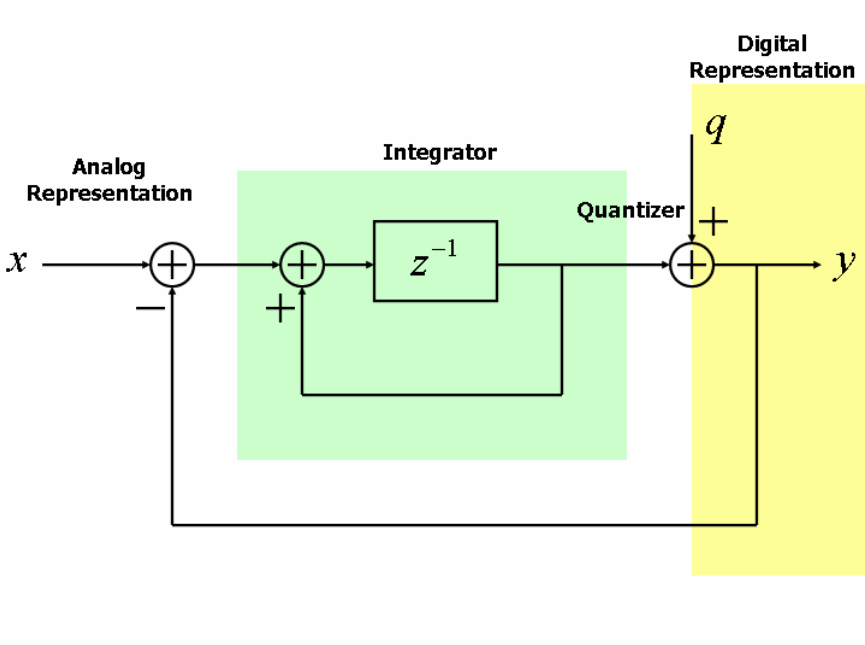

Figure 80: Block diagram of the first order ΔΣ ADC

The output y[n] can be solved as (21) in discrete-time domain and as (22) in z-domain. The quantization noise shaping appears in the difference term q[n]-q[n-1] in (22), which is a first-order high-pass filter in the frequency domain. If the quantization noise spectrum is flat, (as is true for random input and high resolution) then the output quantization noise will have the same spectrum as the filter frequency response. If more integrators are used, an N-th order noise shaping filter will result as in (23). The quantization noise shaping spectrum plots with the filter order N=1, 2, 3 are shown normalized in Figure 81.

(21): $y[n] = x[n-1] + q[n] - q[n-1]$

(22): $Y(z) = z^{-1}X(z) + (1 - z^{-1})Q(z)$

(23): $Y(z) = z^{-1}X(z) + (1 - z^{-1})^{N}Q(z)$

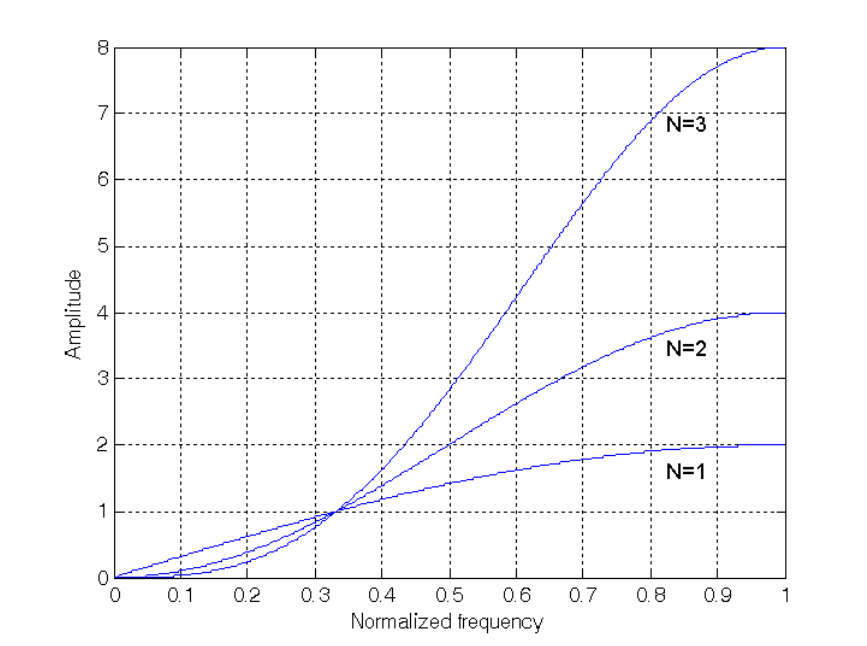

Figure 81: Shaped quantization noise spectra with noise shaping filter order N=1,2,3

The actual quantization noise spectrum and shaped quantization noise spectrum are highly dependent on the input signal characteristics. A simulation of a random input signal to the first-order ΔΣ modulator generates the shaped noise spectrum as shown in Figure 82. This is somewhat like the theoretically estimated spectrum, although the limited sample size and random nature of the signal causes fluctuations of the spectrum.

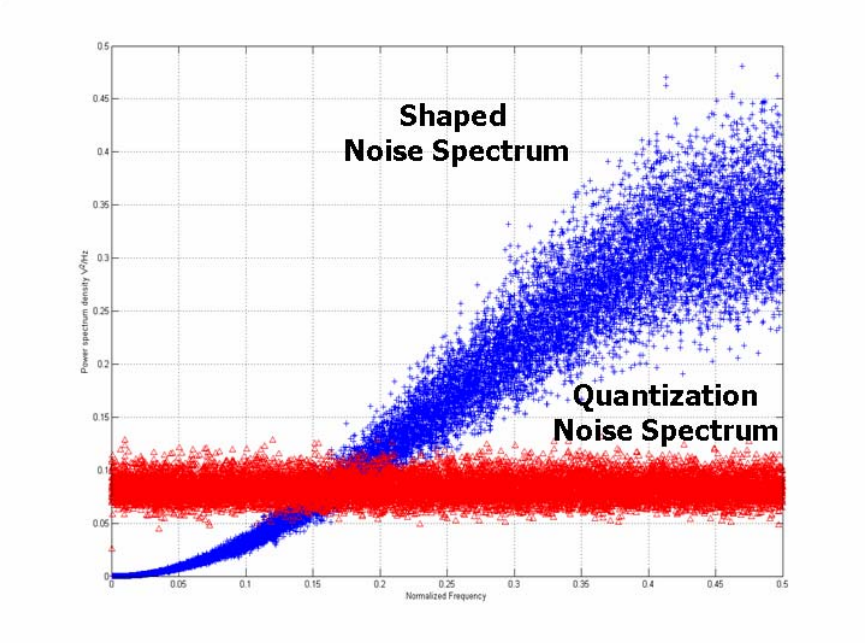

Figure 82: Shaped quantization noise spectrum with first-order ΔΣ modulator and random input

The noise power after ideal filtering of all signal above the Nyquist frequency and downsampling can be estimated as in (24), where N is the order of noise shaping and M is the oversampling ratio (OSR). Then we can now calculate the dynamic range of the ADC which is defined as the maximum signal power over the total noise power. The result is given in (25). The estimated SNR's of 1-bit quantization noise-shaping filter of order N=1, 2, 3 are plotted (using (25) to calculate SNR=DR) in Figure 83. The effective number of bits can be obtained by using the dynamic range of a Nyquist ADC (26), and solving for the number of bits B as shown in (27) [11].

(24): $$P_{signal} \cong \left(\frac{\pi^{2N}}{2N+1}\right)\left(\frac{1}{M^{2N+1}}\right)\frac{\Delta^2}{12}$$

(25): $$[DR_{\Delta\Sigma}]^2 = \frac{3}{2}\frac{2N+1}{\pi^{2N}}M^{2N+1}$$

(26): $$[DR_{Nyquist}]^2 = 3\times 2^{2B-1}$$

(27): $$B = \frac{[DR]_{dB} - 1.76}{6.02}$$

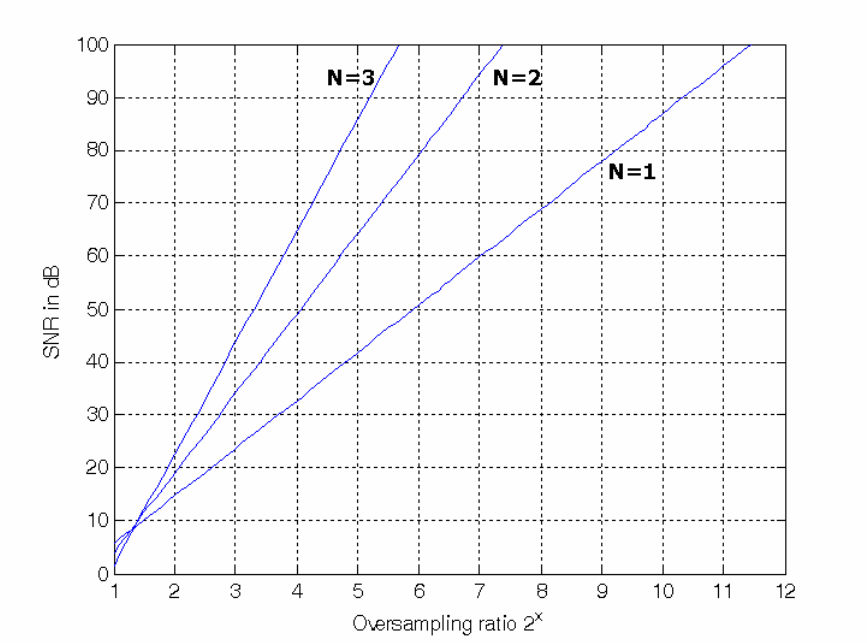

Figure 83: SNR versus oversampling ratio with noise sampling filter order N=1,2,3

To see how effective noise shaping is, consider a comparison between a 1-bit quantizer and a 1-bit noise-shaping converter. A 85-taps FIR Hanning window [94] is used to remove frequency content above Nyquist frequency for both converters. The results are shown in Figure 84 and Figure 85. The noise-shaping converter output, once filtered, is a much better reconstruction of the original signal, whereas the 1-bit quantizer output shows severe distortion due to quantization noise.

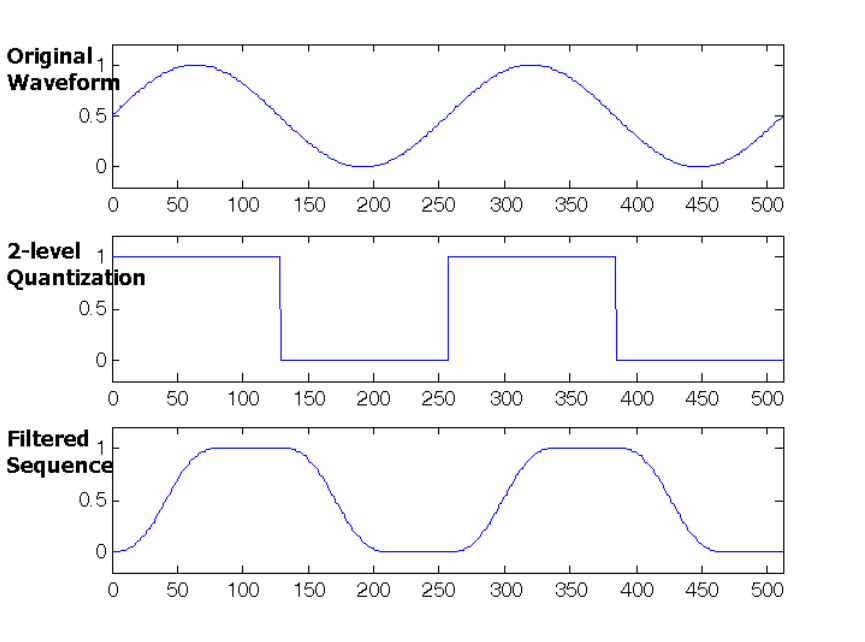

Figure 84: Signal reconstruction with 1-bit quantizer

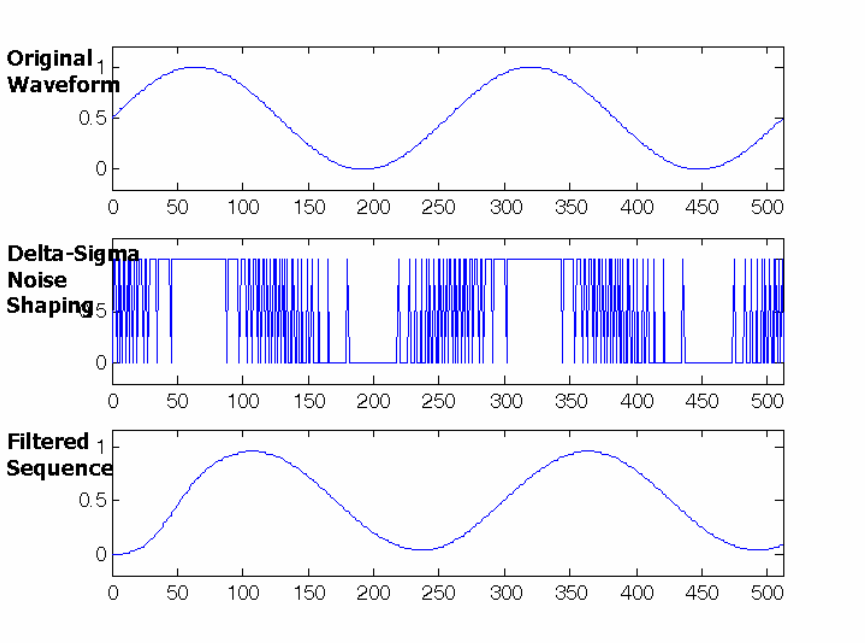

Figure 85: Signal reconstruction with 1-bit noise shaping quantizer

B.1.4 Implementation Examples

The ΔΣ encoder implementations have high degree of freedom in the order of noise-shaping filter, the resolution of quantizer, and the decimation filter. Some implementations have specific target applications such as integrated service digital network (ISDN) transceiver, hearing aid, and compact disk (CD) quality voice signal processing, while others implement the stand-alone ΔΣ ADCs. Various ΔΣ converters reported in the literature have 14 - 18 bits resolutions with 24 kHz - 12.5 MHz output rates as given in Table 10 [96-101].

Table 10: ΔΣ ADC implementation examples

Ref.	Bit Res.	Output Rate	Order	Quantization	Process
[96]	13	80 kHz	3	1	1.5 μm CMOS
[97]	18	48 kHz	4	4	2.4 μm BiCMOS
[98]	16	24 kHz	3	1	2 μm CMOS
[99]	14	80 kHz	2	1	1.75 μm CMOS
[100]	12	12.5 MHz	3	4	0.65 μm CMOS
[101]	12	1.92 MHz	2	6	0.18 μm CMOS

B.1.5 ΔΣ Encoding Issues

In Figure 80, there are both a quantizer (ADC) and a digital-to-analog converter (DAC). These can be 1-bit or multi-bit converters and the ΔΣ ADC will still operate. One of the advantages of a 1-bit based ΔΣ converter is that it does not suffer from ADC or DAC nonlinearity issues since 1-bit ADC and DAC cannot exhibit nonlinear behavior.

Another choice in the converter architecture is the order of the integration. First-order ΔΣ modulators are easy to fabricate since they are less susceptible to the passive component matching errors and nonlinearity issues.

ΔΣ modulators are seen to suffer from sub-signal frequency tone creation as shown in Figure 86 and DC input dependent conversion error as shown in Figure 87. Full nonlinear modeling of the modulator is necessary to explain these phenomena.

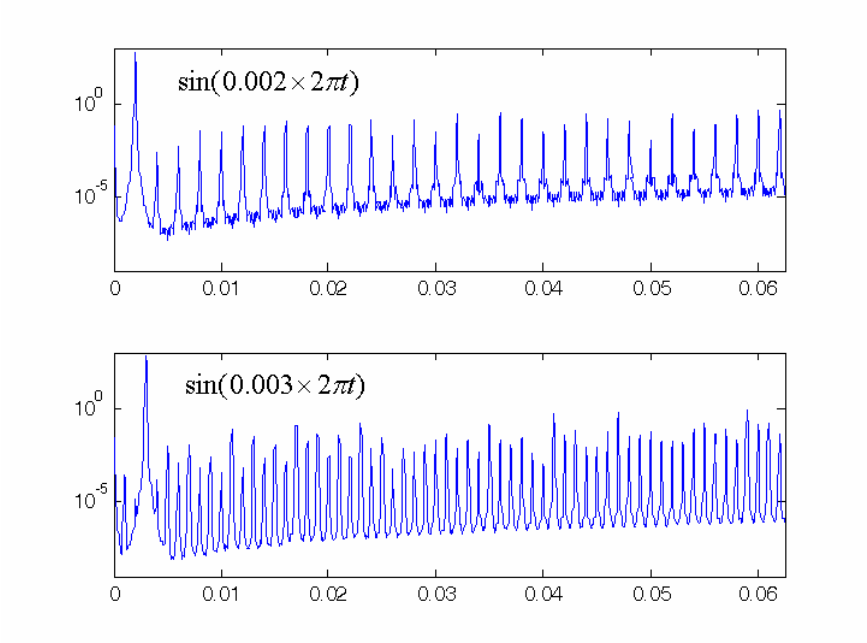

Figure 86: Tones excited with first order 1-bit noise shaping ADC

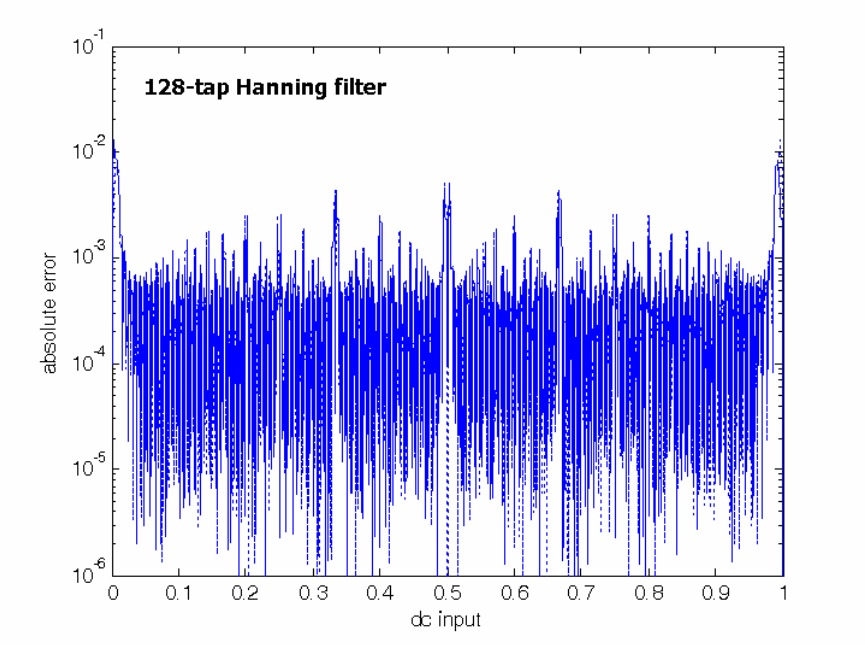

Figure 87: DC error pattern of first order ΔΣ ADC

As noted earlier, the dynamic range of ΔΣ ADC is determined in part by the OSR and the downsampling filter. This can allow flexible converter designs that can change their run-time operation mode, by changing OSR and downsampling scheme. The SNR of the converter can be adaptively updated to control the conversion accuracy and circuit power consumption. It adds more flexibility for implementations [11, 26, 102, 103].

By exchanging low-pass signal band and disposable high-pass noise band, a band-pass ΔΣ converter can be designed with an appropriate noise-shaping filter, and it is attractive to communications systems [26, 104-106]. Also a high-pass architecture is beginning to be explored for direct conversion communication applications [71, 72, 107-109].

A high-order ΔΣ architecture converter might suffer from the instability even with a specific range of dc input. A stabilized high-order noise-shaping architecture can be obtained with several design techniques [86, 110-112]. A complete stability analysis of ΔΣ conversion architecture is under development. There are several approximate and heuristic stability analysis techniques such as linear models, root locus, and describing function method [26, 29, 113, 114].

The concept of the shaping of quantization noise, the feedback of the quantization error, and the modulation of the input signal have been extended to the variety of signal processing schemes that mix analog and digital signal representations. Various steps such as poly-phase ΔΣ ADC, time-interleaved ΔΣ ADC, and space-time ΔΣ ADC have been taken in this direction [115-121].

B.2 ΔΣ Decoding

The performance of ΔΣ conversion is highly dependent on the decimation filter or decoding scheme. Linear filters have been proposed for high-speed downsampling. These filters are based on a linear approximation of the modulator model and the resulting quantization noise. Most of these filters have been designed in the frequency domain. Considering the nonlinear nature of the ΔΣ converter, several nonlinear decoding algorithms have been devised by using time-domain observation of the modulator. The

time-domain approach uses the nonlinear state space system modeling and estimation of encoder internal states.

B.2.1 Linear Decimation Filters

The decimation of the modulated stream has been a speed bottleneck for high-speed ΔΣ ADCs. The comb filter has been a frequently used approach for the first stages of decimation filtering since it can be implemented only with addition or subtraction making it very fast, and it can be simplified to yield a compact system using (28) as shown in Figure 88 [51, 122].

$$\textbf{(28):}\ D(z)=\left[\frac{1}{N}\sum_{i=0}^{N-1}z^{-i}\right]^{k}=\frac{1}{N^{k}}\times\frac{1}{(1-z^{-1})^{k}}\times(1-z^{-N})^{k}$$

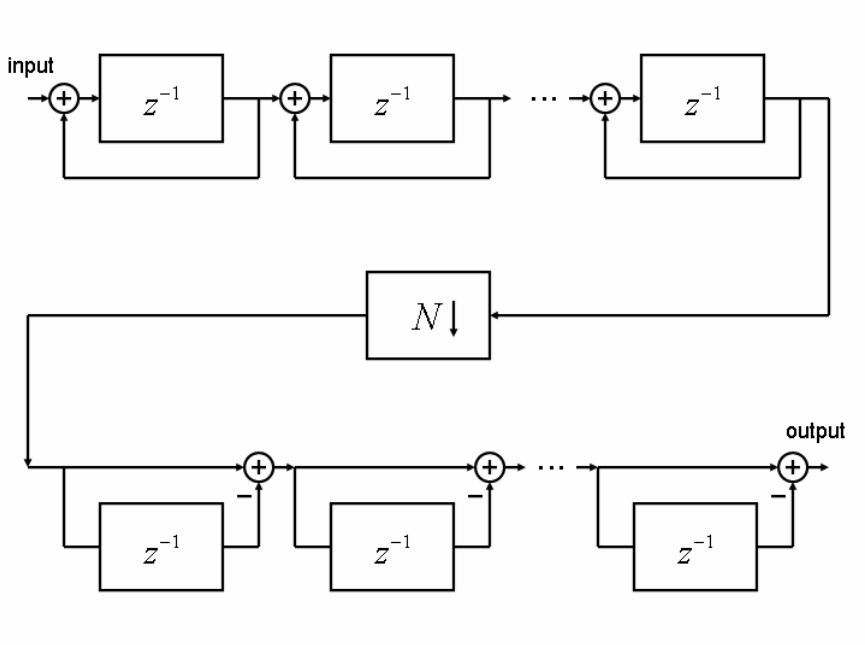

Figure 88: Simplified decimation filter block diagram

The difference blocks in the diagram runs have N times slower since it comes after N-bit downsampling. The accumulation block must be as fast as the sampling frequency. A group of carry save adders and a 2-bits per 1-bit representation have been proposed for the high-speed decimation of ΔΣ output [123, 124]. The accumulation block could overflow due to the nature of the operation and the limited word length of accumulator. It has been shown that the filter will be operational in spite of the overflow, as long as the arithmetic is modular [52].

In the way of finding optimum filters for ΔΣ modulator, a sub-optimal filter based on the spectral analysis of quantization noise was proposed as given in (29) [50]. This filter has been used as a comparison standard for many nonlinear decoding algorithms

[49, 53]. It is interesting that the sub-optimal filter impulse response is similar to triangular filter, which is the sinc^2 filter, and the triangular filter is used as a comparison standard in several studies [48]. The filtering performance of the sub-optimal filter and sinc filters are compared in Figure 89 and Figure 90.

$$\textbf{(29):}\ h_{N,opt} = \frac{6(n+1)(N-n)}{N(N+1)(N+2)}, 0 \le n \le N-1$$

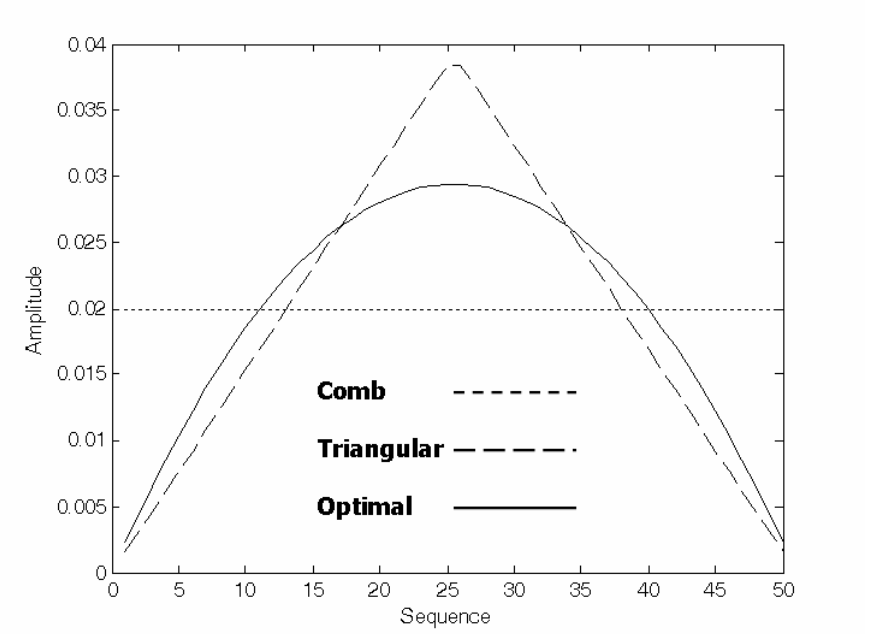

Figure 89: Impulse response of linear filters

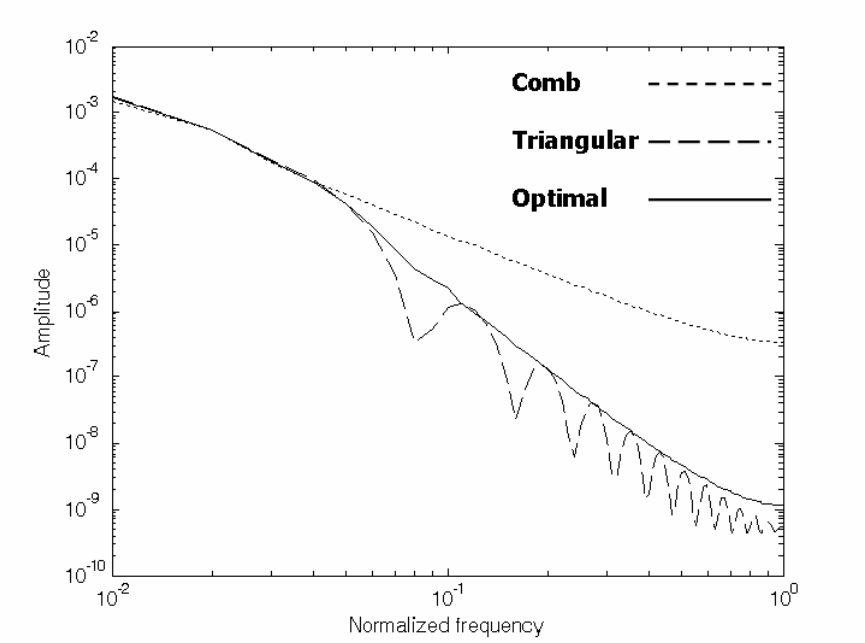

Figure 90: Frequency response of linear filters

B.2.2 Nonlinear Modeling and Decoding of ΔΣ Converter

While switched-capacitor technology is used for the majority of ΔΣ converters, continuous-time filters are useful for the implementation of ΔΣ converter variations such as band-pass ΔΣ converters [69]. In most cases, continuous-time converter analysis is conducted in the discrete-time domain with the approximate discrete transformations replacing continuous-time transformations since this method benefits from the many established discrete-time signal processing techniques. Discrete-time modeling of ΔΣ encoders can adopt a general class of first-order linear discrete-time system models given in (30) and (31). In the equation, u is state space variable, x is input, and y is output. Let's

assume that A, B, C, and D are time-invariant, which means that they are fixed and do not change over time.

(30): $\vec{u}[n+1] = A\vec{u}[n] + B\vec{x}[n]$

(31): $\vec{y}[n] = C\vec{u}[n] + D\vec{x}[n]$

Considering scalar analog input and scalar digital output of an N-th order interpolative ΔΣ modulator, the general first-order linear system equation can be turned into a nonlinear ΔΣ converter model given in (32). The linear system output y[n] is replaced with a nonlinear quantization function Q(c·u[n]) with the quantization error defined in (33). The transition matrix A has integrator connectivity information and the vector b has input signal distribution connectivity. The d maintains feedback connectivity from the quantized output to internal states.

(32): $\vec{u}[n+1] = A\vec{u}[n] + b\vec{x}[n] - \vec{d}Q(\vec{c} \cdot \vec{u}[n])$

(33): $e[n] = \vec{c} \cdot \vec{u}[n] - Q(\vec{c} \cdot \vec{u}[n])$

This model is useful for the modeling of general class ΔΣ converters. A large number of high-order converters can be transformed to a diagonal form with this method. This transformation enables identification and stability analysis of the system [125]. The modeling method can be directly applied to several useful analyses on ΔΣ encoders. With the second-order encoder given in Figure 91, internal states of the converter can be plotted in state space for given input signals as shown in Figure 92. This geometric view of converter is useful for stability analysis and design verification [126].

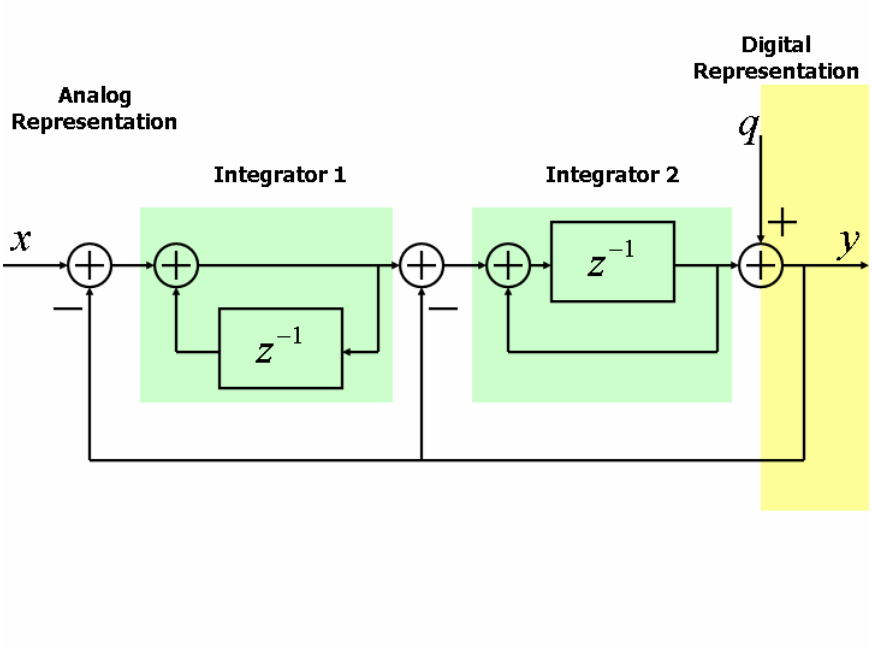

Figure 91: Second-order ΔΣ modulator signal flow diagram

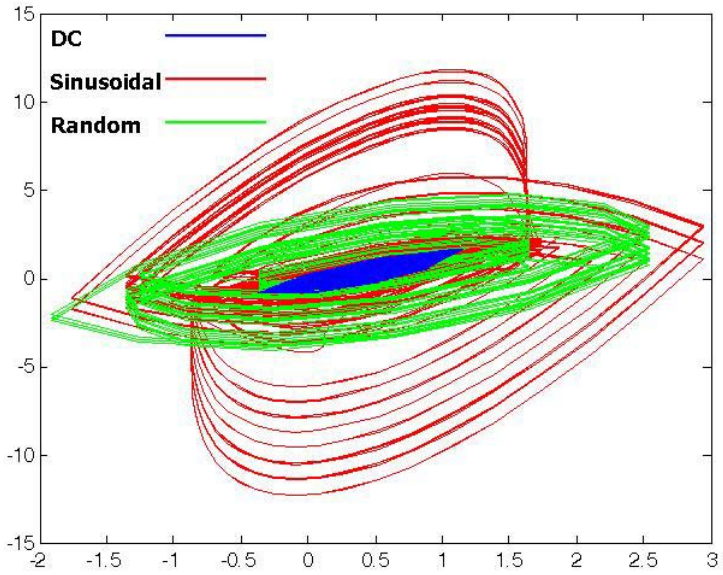

Figure 92: Converter state space trajectories

To overcome linearly approximated decoder performance, several nonlinear modeling techniques for ΔΣ converter have been proposed, such as rational cycle model [53], auto-correlative state estimation [54], recursive stream analysis [48], Viterbi decoding [127], and optimal internal decision estimation [49]. It is interesting to find a study that proves that a democratic representation, which every bit output of ΔΣ encoder has equivalent weight, i.e. linear finite impulse response filter, cannot exceed the accuracy of non-democratic, or nonlinear algorithms [32, 127].

Appendix C

ANALOG FRONT-END MODELING

C.1 Analog Front-end Models

Figure 93 and Figure 94 illustrates several possible analog front-end models for ADC use. The traditional Nyquist sampling and oversampling front-ends have anti-alias low-pass filters and sample-and-hold before the ADC, as shown in Figure 93. The cut-off frequency f_c of the LPF should be less than the desired highest signal frequency f_M to avoid aliasing. For Nyquist sampling, the sampling frequency is $f_s=2f_M$ and the sampling frequency for oversampling is $f_{s,os}=2Nf_M$, where N is the oversampling ratio. There are several published implementation examples for Nyquist sampling whose analog front-end have only sample-and-hold [37-40], or do not include both anti-alias LPF and sample-and-hold [41, 42]. Also the majority of published noise-shaping oversampling converters are not equipped with either the LPF or sample-and-hold [43-45]. These cases are modeled in Figure 94.

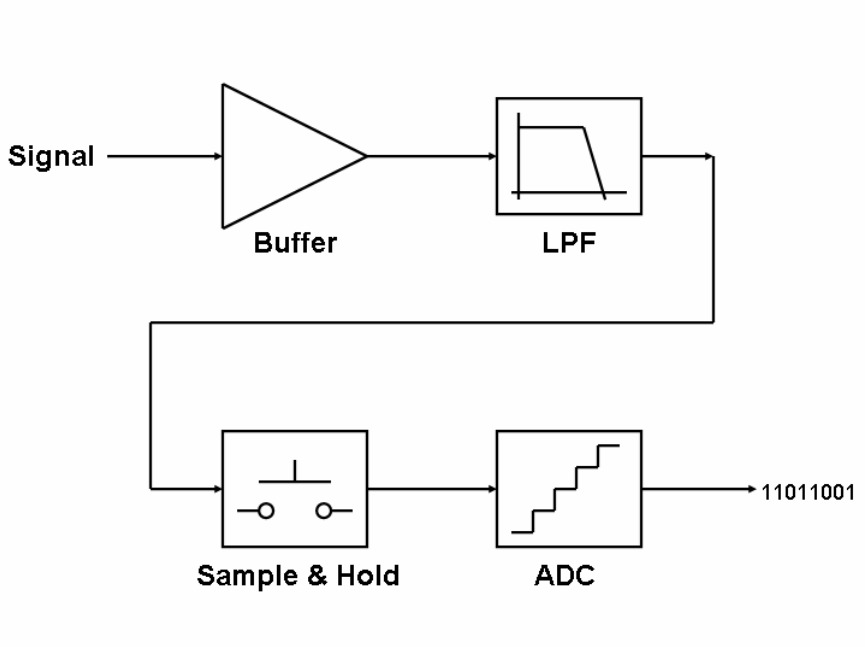

Figure 93: Nyquist and oversampling with anti-alias LPF and sample-and-hold

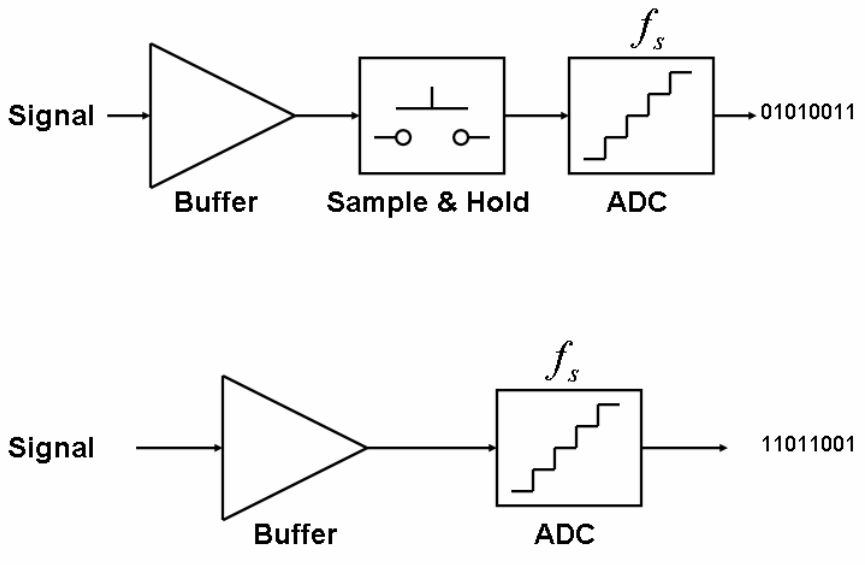

Figure 94: Direct sample-and-hold and direct converter

C.2 Input Signal Models

The input signal characteristics can be defined in several ways. A band-limited signal is a good model as a realistic input. Let's define the signal model A, $X_{in,A}(f)$ with a flat spectrum density $P_{in,A}$ and a strict band-limit $f_{in,A}$ as expressed in (34). When the bandwidth of the model is below Nyquist sampling frequency, it will show the effect of signal and noise propagation without signal aliasing. Also this model can be used to demonstrate the effect of signal aliasing on the performance of an ADC configuration, when the cut-off frequency $f_{in,A}$ is greater than Nyquist sampling frequency. The abrupt change of power density at $f_{in,A}$ makes the model rather ideal.

The other approach to model input signal is a signal with flat spectrum over the infinite bandwidth. A signal model B, $X_{in,B}(f)$ with a constant power density $P_{in,B}$ is given in (35). The model is useful to model any undesired higher-frequency signal than Nyquist frequency that the designer didn't take into account. Since the signal is not band-limited, it can be regarded as the statistically worst case or a white noise signal.

$$\textbf{(34):}\quad X_{in,A} = \begin{cases} P_{in,A} & f \le f_{in,A} \\ 0 & f > f_{in,A} \end{cases}$$

$$\textbf{(35):}\quad X_{in,B}(f) = P_{in,B}$$

A pure random signal such as noise usually shows a flat spectrum and, as such, it will have constant power density over infinite frequency range, and thus deliver infinite power, which is totally unrealistic. A widely used way to imitate real signals is to filter white noise. Thus the third approach to model input signals is a low-pass white noise. This model produces a band-limited signal with the spectral continuity. Let's assume a first-order LPF with a cut-off frequency $f_{in,C}$. Then the definition of input signal spectrum $X_{in,C}(f)$ is given as in (36) with the spectrum density $P_{in,C}$ at f=0 and the cut-off frequency $f_{in,C}$. Power spectral density plots of input signal models are provided in Figure 95.

(36): $$X_{in,C}(f) = \frac{P_{in,C}}{f^2 / f_{in,C}^2 + 1}$$

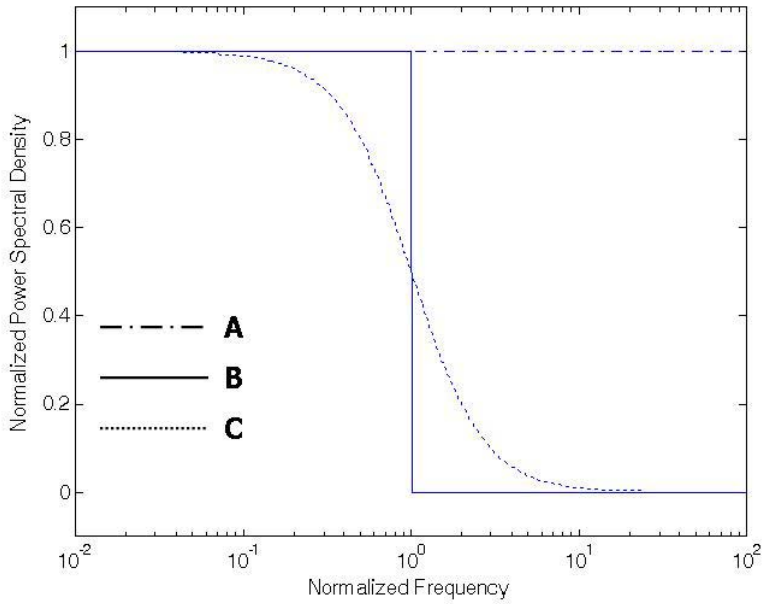

Figure 95: Input signal models. A=abruptly band-limited white noise, B=white noise, and C=first-order low-pass filtered white noise

C.3 Input Buffer

An input buffer can be an amplifier with gain or, often, a unity-gain buffer that separates signal source and front-end. The input buffer will have a gain and a bandwidth, and a first-order model low pass filter is an adequate model in most cases. When the bandwidth is higher than the signal bandwidth f_M, the buffer works as an all-pass filter for signal, though it will filter noise in the higher-frequency region. If the gain-bandwidth is lower than f_M, or it is almost same as f_M, the buffer will filter out high-frequency in-band signal. Let's assume that the buffer transfer function model $H_{bf,k}(f)$ has unity gain and bandwidth $f_{bf}=f_M$ as shown in (37), where k is the order of the buffer block to represent

the effect of cascaded buffers. The frequency responses of Butterworth filters and cascaded first order buffers are compared in Figure 96. We see that cascaded buffers can produce significant in band attenuation of signal.

$$\textbf{(37):}\ H_{bf,k}(f) = \left(\frac{1}{jf / f_{bf} + 1}\right)^k$$

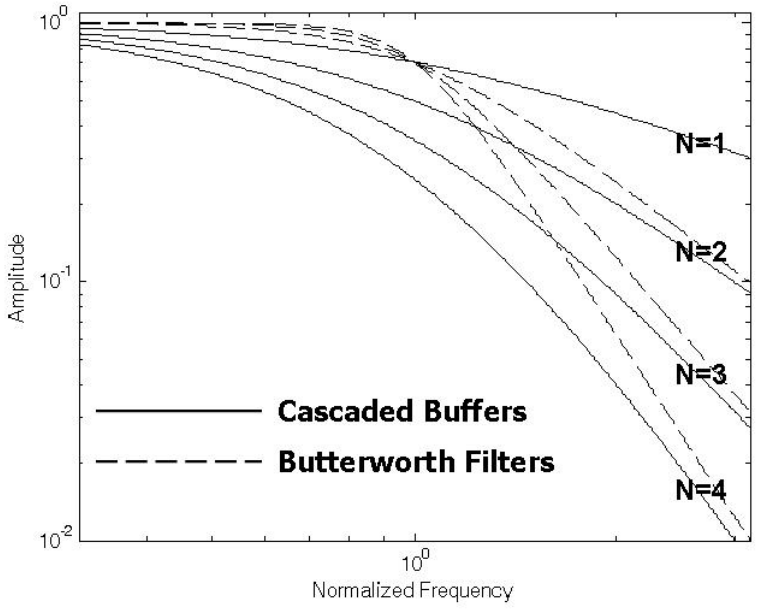

Figure 96: Input buffer frequency responses

C.4 Models for Anti-alias LPF

Higher the order of anti-alias LPF used, the more attenuation is available at f_M and less signal power is attenuated in the signal band. The filters can be implemented with switched capacitors, or with active resistor and capacitor circuits. The implementation of high-order LPF is quite costly since it takes large area for parasitic components. Also the precision of filter poles is subject to the statistical variation of fabrication process. The noise induced by cascaded filter or switching of capacitors will also be increased with high-order filters. For this study, we assume k-th order Butterworth filters used as anti-alias LPFs. The magnitude of the frequency response $H_{lp,k}(f)$ for such filters is as appears in (38) with cut-off frequency $f_{lp,k}$.

$$\textbf{(38):}\ |H_{lp,k}(f)| = \frac{1}{\sqrt{(f/f_{lp,k})^{2k}+1}}$$

As an example let us examine the effect of filtering on the signal-to-noise ratio of a first-order filtered white noise input (previously called input model C). We assume the initial SNR before filtering is 100 dB due to 10^{-10} DC noise power density with input model C, and that the filter adds the same amount of noise power (assumed to be 1.0×10^{-10}) regardless of the filter order. The noise power densities were selected to demonstrate a 16-bits resolution sensitivity application and to show the SNR degradation from the maximum available 100 dB SNR from the input signal. The optimal cut-off frequency for

maximum SNR can be obtained numerically as shown in Figure 97. For each filter order, the effect of a single stage buffer is also displayed. From the plot we see that a filter order of 2 greatly enhances SNR, and a buffer has little impact on the SNR.

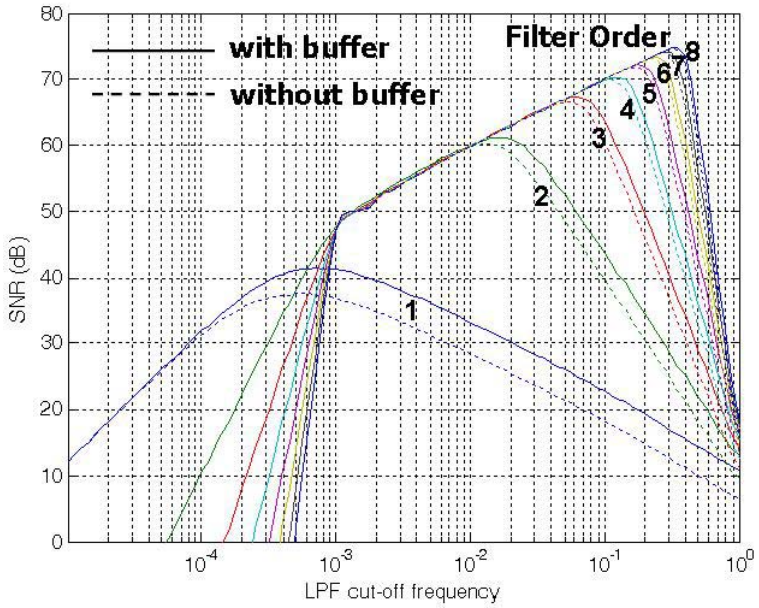

Figure 97: SNR improvement of Butterworth filters for band limited input C

C.5 Modeling of Sample-and-Hold

A simple voltage sampling and holding circuit is modeled as shown in Figure 98. The switch in the figure is turned on to track and sample the input voltage. After sampling, it is turned off to hold the sampled voltage for analog-to-digital conversion. With the switch turned on, the frequency domain representation of the current that goes

through the switch is given in (39), and the holding voltage $V_{sh}(f)$ is obtained as given in (40) [11-13].

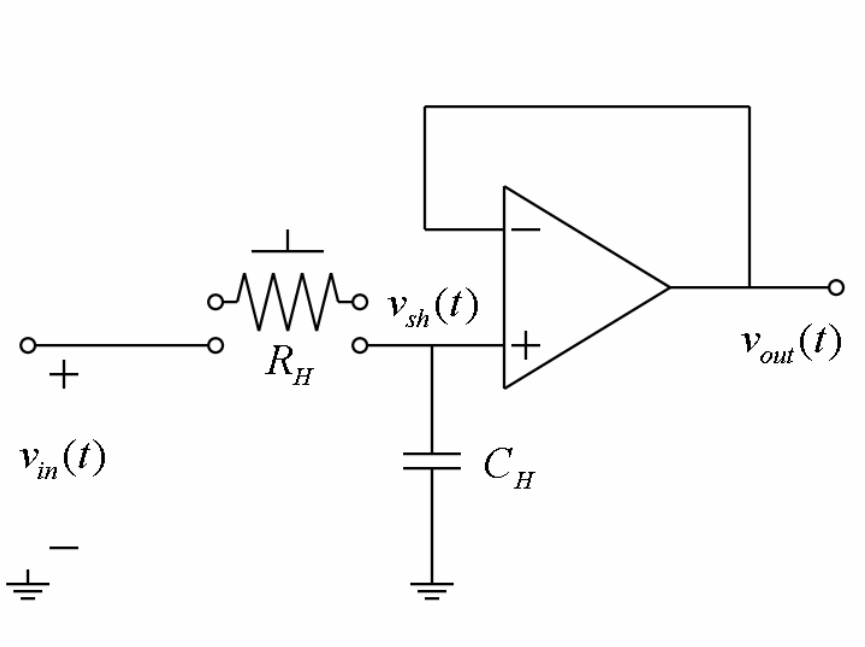

Figure 98: Voltage sample-and-hold model

(39): $$I_{in} = \frac{V_{in}(f)}{R_H + 1/(j2\pi f C_H)}$$

(40): $$\begin{aligned} V_{sh}(f) &= \frac{1}{j2\pi f C_H} I_{in}(f) \\ &= \frac{1}{j2\pi f C_H R_H + 1} V_{in}(f) \end{aligned}$$

The holding capacitance C_H must be large enough to avoid current leakage during holding cycle and it should be small enough to track the input voltage during sampling cycle. There is a trade-off between the tracking accuracy and the holding accuracy with the capacitor size. As shown in (40), the sample-and-hold circuit is a low-pass filter when the switch is on. The time constant of the filter is $C_H R_H$. To allow fast tracking, it should be small, and it should be small enough to avoid transient error, which should be smaller than the desired ADC quantization step. As mentioned, a smaller holding capacitor suffers more from capacitor leakage current. Also the smaller capacitance involves larger kT/C noise [65]. The transient characteristics of the buffer between v_{sh} and v_{out} should be taken into account. It must be fast enough to follow the sampled voltage, while the high-order overshooting should be avoided for accurate analog-to-digital conversion [11-13].

A current integrating sample-and-hold front-end can be modeled in a similar way. For input current i(t), integration time T, and capacitance C, the sampled voltage v(t) is obtained as (41) in time domain and (42) in frequency domain. When Nyquist sampling frequency 2 is taken for the normalized signal band 1, the integrating time T is less than 0.5. Figure 99 shows the frequency response V(f) with a first-order low-pass filter whose cut-off frequency is 0.5, as an asymptote. Capacitance C is 0.5 in the figure to adjust DC gain to 1. The figure shows that the current integrating sample-and-hold type circuits are low-pass filters in its operation.

(41): $[u(t) - u(t-T)] * i(t) = Cv(t)$

$$\textbf{(42):}\ V(j2\pi f) = \frac{1 - e^{-j2\pi fT}}{j2\pi fC} I(j2\pi f)$$

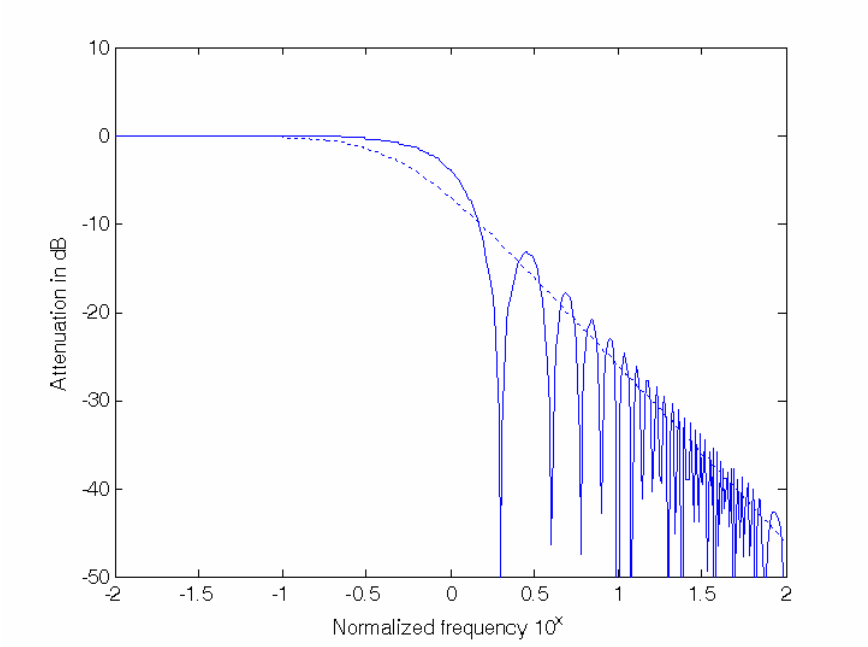

Figure 99: Current integrating sample-and-hold frequency response

C.6 Available ADC SNR without Sample-and-Hold

A sample-and-hold takes a critical role in Nyquist sampling ADCs. The importance of sampling timing and clock jitter already have been examined and analyzed [15, 25]. The absence of sample-and-hold can be treated as a worst case of sampling timing miss or clock jitter noise.

Let's assume A/D conversion of a sinusoid with oscillation frequency f_0 and peak-to-peak amplitude 1. Intuitively, higher the oscillation frequency, as high as 1/2 of Nyquist frequency f_{Nyq}, and greater the oscillation amplitude, ADC conversion error will be larger. The largest conversion error will occur when sinusoid crosses mean value during analog-to-digital conversion process. The actual amount of error introduced by an ADC will be highly dependent on the specific conversion algorithms. For example, single-slope ADC is likely to experience less error than algorithmic ADC since algorithmic ADC encodes exponentially while single-slope ADC encodes linearly. A single-slope ADC conversion error example is provided in Figure 100.

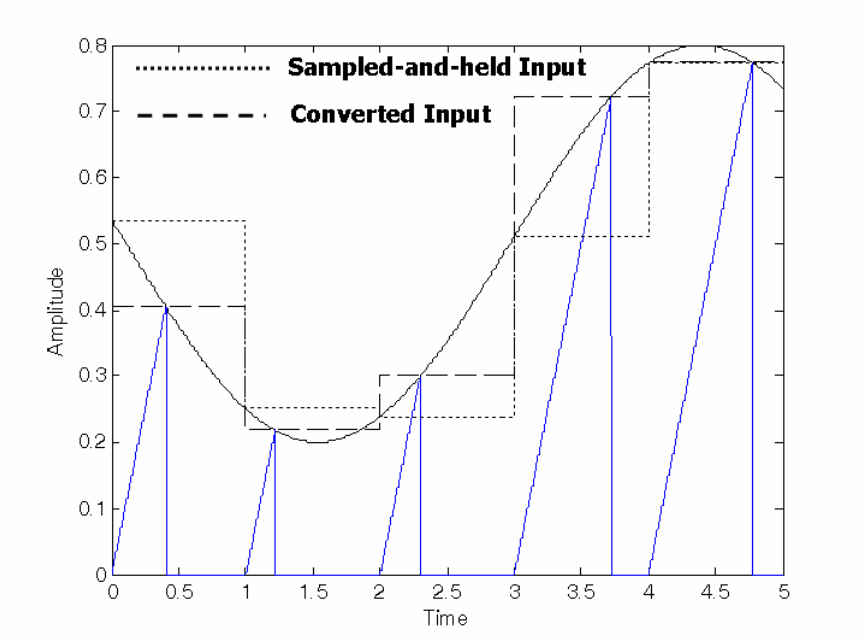

Figure 100: Single-slope ADC conversion error example without sample-and-hold

Let's assume that input signal has bandwidth 1, and Nyquist frequency is $f_{Nyq} = 2$. A pseudo ideal ADC with infinite precision is assumed that its conversion takes whole sampling period and an input value is sampled and held during conversion. The conversion error of the pseudo ADC is difference between the converted output and the input value at the end of sampling period. Also a single-slope ADC with $2^{10} = 1024$ counting levels is assumed for comparison. Mean-square errors of both ADCs are plotted over sinusoidal input frequency as shown in Figure 101. The plot claims that input signal should be band-limited to $f_0 = \Delta$ to avoid conversion error. Higher the precision of an ADC, lower the allowed sinusoid frequency for full resolution. Also conversion error is increase by 6dB when sinusoid frequency is increased by 2 times. An ADC with different conversion algorithm will show different mean-squared error.

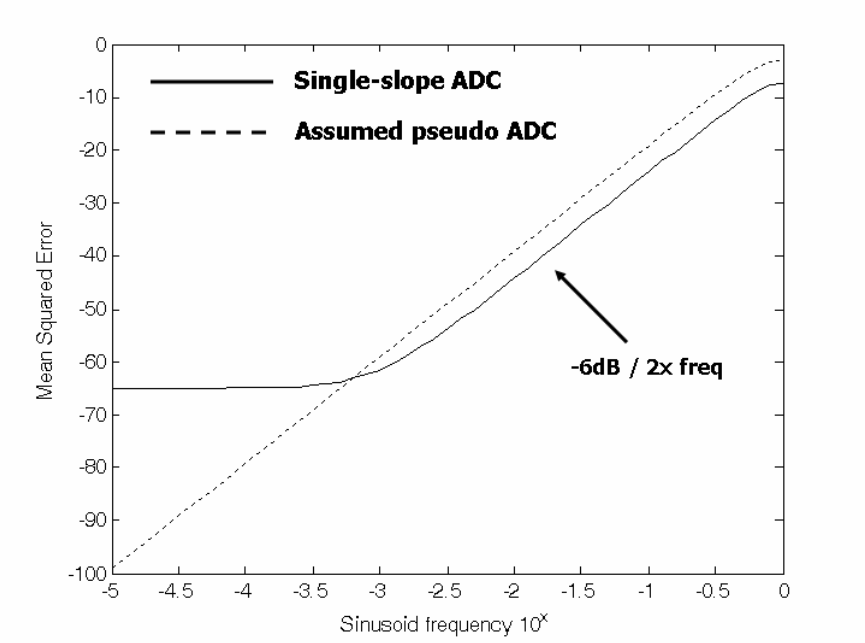

Figure 101: Single-slope and pseudo ideal ADC conversion error

C.7 Continuous-time ΔΣ ADC Model

The property of first-order ΔΣ ADC can be described with an analog model as shown in Figure 102. In the diagram, A and B represents gain of input buffer and feedback path, and C is the capacitance value of integrator. The transfer function of the integrator and a delay with sampling period T is e^{sT}/sC in the s-domain. The model transfer function is given as (43). The model is can be replaced with the discrete-time ΔΣ ADC model (44), when the approximate transformation $z \approx e^{sT}$ and $sT \approx 1-z^{-1}$ are performed and when A=C/T and A=B. The discrete-time model approximation $z \approx e^{sT}$ and $sT \approx 1-z^{-1}$ are valid only when the signal frequency is low compared to the sampling frequency [11, 128].

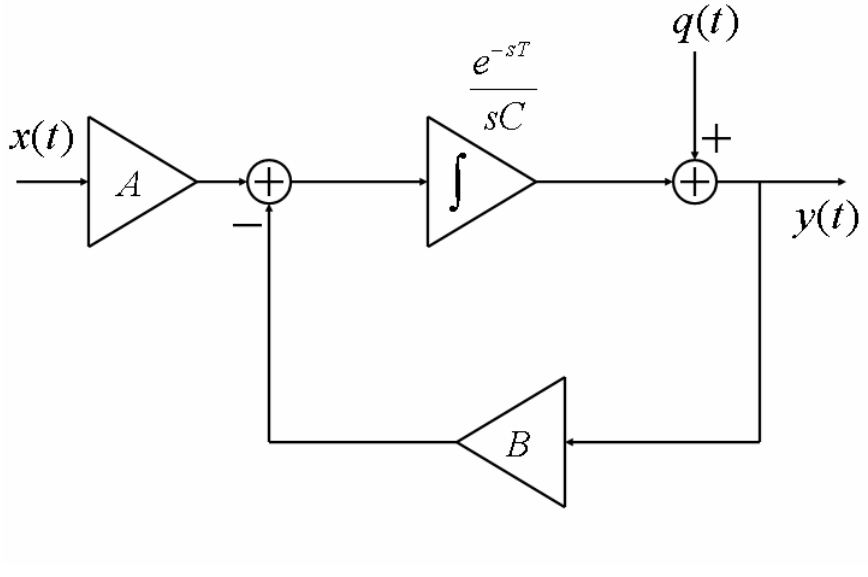

Figure 102: Continuous-time ΔΣ ADC model

$$
\begin{aligned}
Y(s) &= G_x(s)X(s) + G_q(s)Q(s) \\
&= \frac{Be^{-sT}}{sC + Ae^{-sT}}X(s) + \frac{sC}{sC + Ae^{-sT}}Q(s) \\
&= \frac{e^{-sT}}{sT + e^{-sT}}X(s) + \frac{sT}{sT + e^{-sT}}Q(s)
\end{aligned}
\qquad \textbf{(43):}
$$

$$
\textbf{(44):}\ Y(z) = z^{-1}X(z) + (1 - z^{-1})Q(z)
$$

The frequency response of the signal transfer function $G_x(s)$ is plotted to see the effect of the ΔΣ ADC as a signal filter as shown in Figure 103. Frequency is normalized with respect to the oversampling frequency $f_{s,os}/2=Nf_M$, not to the maximum desired signal frequency f_M as other diagrams are. A first-order LPF (dotted line in Figure 103 is an asymptote of $G_x(f)$. When the frequency responses are normalized to the signal band, it becomes clear how higher oversampling ratio allows a wider signal bandwidth through as shown in Figure 104. This characteristic will become important as a means of band limiting the input signal. This allows ΔΣ ADCs to operate very well with no anti-alias filter.

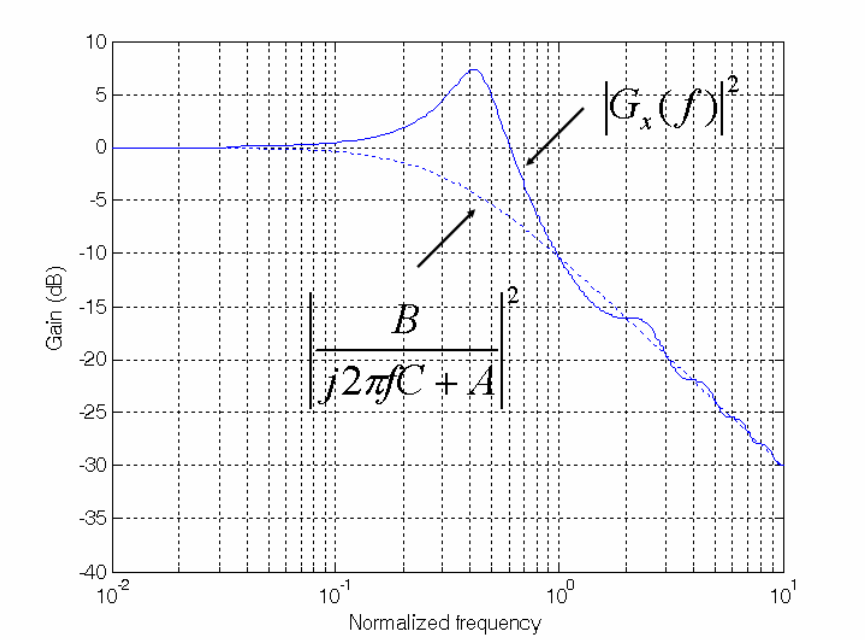

Figure 103: Frequency response of continuous-time ΔΣ ADC model

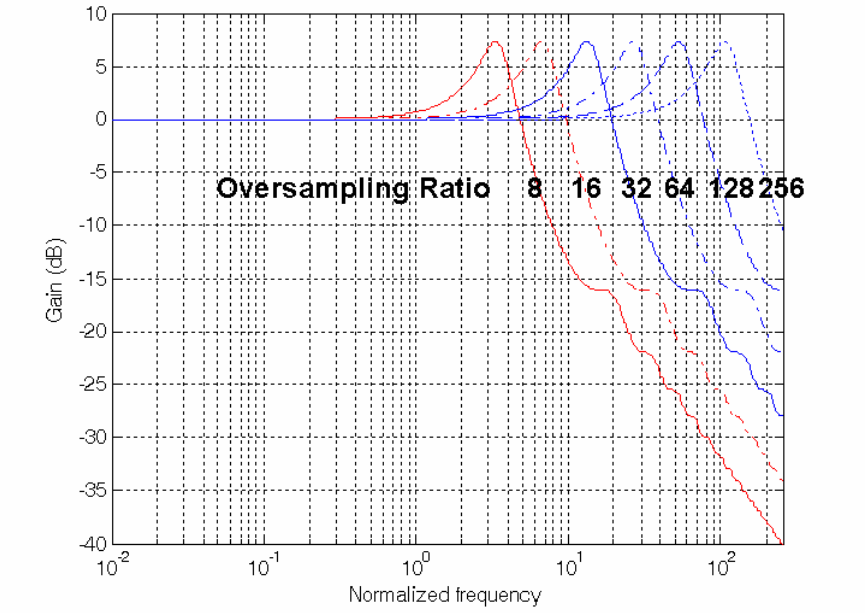

Figure 104: Continuous-time ΔΣ ADC frequency response normalized to signal band

Appendix D

INFORMATION THEORY

D.1 Entropy and Source Coding

A/D conversion can be regarded as a communication channel that involves noise, encoding, and decoding. Information theory provides several useful concepts on quantization, signal source encoding and decoding.

Let's assume a probabilistic experiment with a discrete source S as a random variable that takes symbols from a finite set of alphabet Γ given in (45). The probability distribution $P(S=s_k)$ of Γ is given in (46) and (47). The amount of information obtained after an observation of the event $S=s_k$ is defined in (48), where the observation is regarded as a resolution of uncertainty. When the logarithm base is 2, the measure of information becomes binary digit, i.e. bit. Then entropy H(Γ) of a discrete memoryless source with alphabet Γ as an average information content per source symbol is defined as (49) [90, 129-131].

(45): $\Gamma = \{s_0, s_1, \cdots, s_{K-1}\}$

(46): $P(S = s_k) = p_k, k = 0,1,\cdots,K-1$

(47): $\sum_{k=0}^{K-1} p_k = 1$

(48): $I(s_k) = \log\left(\frac{1}{p_k}\right)$

(49): $H(\Gamma) = \sum_{k=0}^{K-1} p_k \log\left(\frac{1}{p_k}\right) = E[I(s_k)]$

The entropy H(Γ) of a discrete memoryless source is bounded as given in (50). When there is no uncertainty, e.g. probability p_k=1 for some k and all other probabilities are zero, there is no information and H(Γ)=0. On the other hand, when all the events are equally probable, i.e. equiprobable, the upper bound as the maximum entropy H(Γ)=$\log_2$K is obtained, which means that it is the most difficult to predict an event. It can be easily shown that the entropy function H(Γ) is continuous, additive, nonnegative, and symmetric [90, 129-131].

(50): $0 \le H(\Gamma) \le \log_2 K$

The "information" in information theory should be related to the uncertainty and the resolution of uncertainty rather than the traditional idea of words such as knowledge [90, 129-131]. In the source encoding perspective, Shannon claimed in the noiseless coding theorem that $H(\Gamma)$ is the minimum average number of bits per source symbol required to represent a discrete memoryless source without loss [132].

D.2 Mutual Information and Channel Coding Theorem

When channel output Y is observed with channel input X, a conditional entropy H(X|Y) is defined as (51). The conditional entropy represents the amount of uncertainty left about the input after the observation. Therefore the difference between H(X) and the differential entropy will be the uncertainty resolved about the input after observation. The quantity is called the mutual information I(X; Y) and it is defined in (52). The mutual information is symmetric and nonnegative, which means that one will not lose any information with output observation. When the mutual information is maximized over input probability distribution, it represents channel capacity of a discrete memoryless channel in bits per channel use, as defined in (53). When maximum channel symbol rate is T_c and source symbol rate is T_s, the average information rate is $H(X)/T_s$ bits per second and channel capacity per unit time is C/T_c. Shannon's channel coding theorem or noisy coding theorem states that there exists a coding algorithm that enables zero probability of error transmission of data, when a relation given in (54) holds. This theorem only shows

that there is a scheme that achieves error-free communication and it does not shows a way to implement it [90, 129-132].

$$\textbf{(51):}\ H(X\,|\,Y)=\sum_{k=0}^{K-1}H(X\,|\,Y=y_k)\,p(y_k)$$

$$\textbf{(52):}\ I(X;Y)=H(X)-H(X;Y)$$

$$\textbf{(53):}\ C=\max_{p(x_j)} I(X;Y)$$

$$\textbf{(54):}\ \frac{H(X)}{T_s}\le\frac{C}{T_c}$$

D.3 Differential Entropy and Information Capacity Theorem

The differential entropy of a continuous random variable X h(X) is defined in (55). Not like the absolute entropy H(X) defined with a discrete source, the reference term "$-\log_2\Delta_x$" that goes to infinity as Δ goes to 0 is discarded for practical purpose as shown in (56). The differential entropy is upper bounded as given in (57), where Gaussian random variable with variance σ^2 attains maximum differential entropy. The definition of mutual information for continuous random variable X and Y is given in (58).

(55): $h(X) = \int_{-\infty}^{\infty} f_X(x) \log_2 \left[\frac{1}{f_X(x)} \right] dx$

(56): $H(X) = h(X) - \lim_{\Delta x \to 0} \log_2 \Delta x$

(57): $h(X) \leq \frac{1}{2} \log_2 (2\pi e \sigma^2)$

(58): $I(X;Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{X,Y}(x,y) \log_2 \left[\frac{f_X(x \mid y)}{f_X(x)} \right] dxdy$

For a discrete-time, memoryless Gaussian channel with transmission time T, band limit B, and power limit $E[X_k^2]=P$ for all k, let's assume input signal X_k, additive white Gaussian noise N_k, and received signal Y_k as (59), where K=2BT is the number of samples. The Gaussian noise has zero mean, power spectral density $N_0/2$, therefore, variance is $\sigma 2=N_0B$. The information capacity is defined with the maximum mutual information over input distributions and power limit as given in (60). For the maximum capacity, X_k should be Gaussian with average power P. The differential entropy $h(Y_k)$ and $h(N_k)$ are arranged as in (61) and (62) using property of Gaussian distribution. Therefore the information capacity in bits per transmission is (63) and capacity in bits per second is (64).

(59): $Y_k = X_k + N_k, k = 1,2,\cdots,K$

(60): $$C = \max_{f_{X_k}(x)} \{ I(X_k ; Y_k) : E[X_k^w] = P \}$$

(61): $$h(Y_k) = \frac{1}{2} \log_2 [2\pi e (P + \sigma_N^2)]$$

(62): $$h(N_k) = \frac{1}{2} \log_2 (2\pi e \sigma_N^2)$$

(63): $$C = \frac{1}{2} \log_2 \left(1 + \frac{P}{\sigma_N^2} \right)$$

(64): $$C = B \log_2 \left(1 + \frac{P}{N_0 B} \right)$$

Shannon's information capacity theorem claims that the capacity of a continuous channel with band limit B and power limit P is limited by these capacity relations, and that the given capacity of error-free transmission can be achieved through an encoding scheme. The transmitted energy per bit E_b can be defined with $P=E_bC$ to transform (64) into (65). The relation between C/B and E_b/N_0 plotted in Figure 105 is called the bandwidth efficiency diagram [90, 129-132].

(65): $$\frac{E_b}{N_0} = \frac{2^{C/B} - 1}{C/B}$$

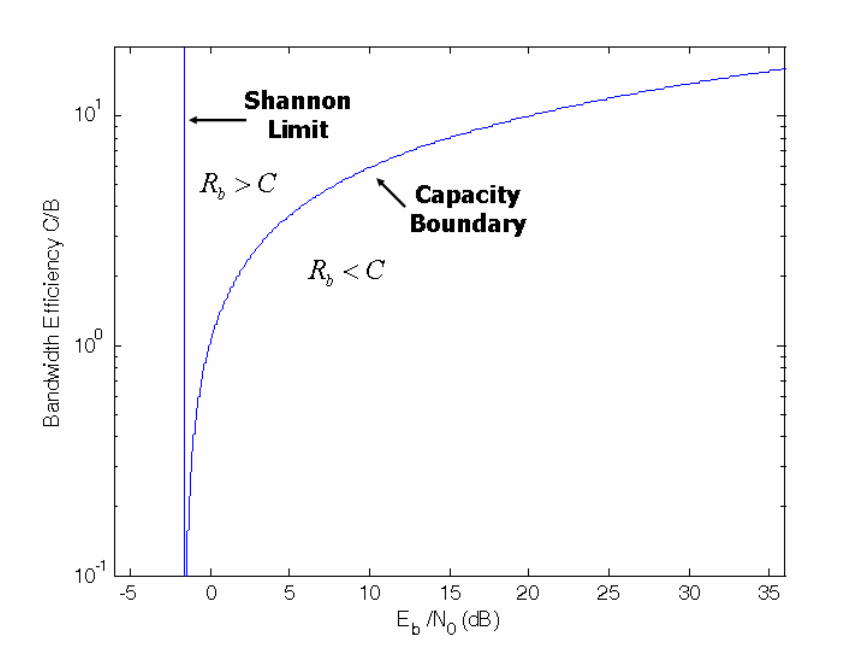

Figure 105: Bandwidth efficiency diagram

Appendix E

SI-CMOS TRANSISTOR SPICE MODELS

All circuit designs and simulations are based on AMI 1.5 μm n-well Si-CMOS chip fabrication services provided through MOSIS [8]. The process provides two poly layers, two metal layers, an NPN option, and PiP capacitors.

E.1 MOSIS SPICE Models

The following model list with run numbers is obtained from MOSIS web page [133].

```
T0CU    T14Z    T15L    T16Q    T16S
T16V    T17D    T18K    T1AT    T1AZ
T1CI    T1CL    T21M    T22X    T23F
T24P    T26W    T27G    T28N    T28P
T29V    T2AH    T2CR    T31A    T31E
T32P    T32Q    T33Z    T34D    T35O
T37C    T38F    T39T    T3AG
```

E.2 MOSIS SPICE Model Run Number T0CU

```
* DATE: Aug 23/01
* LOT: T0CU                 WAF: 5114
```

```
* Temperature_parameters=Default
.MODEL CMOSN NMOS (                                LEVEL   = 49
+VERSION = 3.1            TNOM    = 27             TOX     = 3.06E-8
+XJ      = 3E-7           NCH     = 7.5E16         VTH0    = 0.5461273
+K1      = 0.9248171      K2      = -0.0746521     K3      = 7.3546504
+K3B     = -1.3783456     W0      = 1E-7           NLX     = 1E-8
+DVT0W   = 0              DVT1W   = 0              DVT2W   = 0
+DVT0    = 0.6105593      DVT1    = 0.2813887      DVT2    = -0.3207057
+U0      = 687.6635171    UA      = 2.222506E-9    UB      = 1.213885E-18
+UC      = 5.939783E-11   VSAT    = 1.06244E5      A0      = 0.6091078
+AGS     = 0.1475122      B0      = 2.455572E-6    B1      = 5E-6
+KETA    = -5.207522E-3   A1      = 0              A2      = 1
+RDSW    = 3E3            PRWG    = -0.0591643     PRWB    = -0.0365956
+WR      = 1              WINT    = 7.73594E-7     LINT    = 2.483956E-7
+XL      = 0              XW      = 0              DWG     = -2.100159E-8
+DWB     = 3.664922E-8    VOFF    = -4.161505E-3   NFACTOR = 1.0577976
+CIT     = 0              CDSC    = 0              CDSCD   = 0
+CDSCB   = 4.95502E-6     ETA0    = -0.6952862     ETAB    = -0.2227648
+DSUB    = 0.6107704      PCLM    = 1.3283099      PDIBLC1 = 8.983088E-3
+PDIBLC2 = 1.979186E-3    PDIBLCB = 0.1            DROUT   = 0.0615405
+PSCBE1  = 2.205899E9     PSCBE2  = 5.040015E-10   PVAG    = 0.2668387
+DELTA   = 0.01           RSH     = 52.5           MOBMOD  = 1
+PRT     = 0              UTE     = -1.5           KT1     = -0.11
+KT1L    = 0              KT2     = 0.022          UA1     = 4.31E-9
+UB1     = -7.61E-18      UC1     = -5.6E-11       AT      = 3.3E4
+WL      = 0              WLN     = 1              WW      = 0
+WWN     = 1              WWL     = 0              LL      = 0
+LLN     = 1              LW      = 0              LWN     = 1
+LWL     = 0              CAPMOD  = 2              XPART   = 0.5
+CGDO    = 1.8E-10        CGSO    = 1.8E-10        CGBO    = 1E-9
+CJ      = 2.807209E-4    PB      = 0.9642875      MJ      = 0.5323927
+CJSW    = 1.202779E-10   PBSW    = 0.9809784      MJSW    = 0.1
+CJSWG   = 6.4E-11        PBSWG   = 0.9809784      MJSWG   = 0.1
+CF      = 0               )
*
.MODEL CMOSP PMOS (                                LEVEL   = 49
+VERSION = 3.1            TNOM    = 27             TOX     = 3.06E-8
+XJ      = 3E-7           NCH     = 2.4E16         VTH0    = -0.8476404
+K1      = 0.4513608      K2      = 2.379699E-5    K3      = 13.3278347
+K3B     = -2.2238332     W0      = 9.577236E-7    NLX     = 2.924346E-7
+DVT0W   = 0              DVT1W   = 0              DVT2W   = 0
+DVT0    = 2.097586       DVT1    = 0.6808189      DVT2    = -0.0568857
+U0      = 236.8923827    UA      = 3.833306E-9    UB      = 1.487688E-21
+UC      = -1.08562E-10   VSAT    = 1.659328E5     A0      = 0.7884225
+AGS     = 0.1890823      B0      = 3.435275E-6    B1      = 1.229777E-6
+KETA    = -1.960186E-3   A1      = 0              A2      = 0.364
+RDSW    = 3E3            PRWG    = 0.1155013      PRWB    = -0.3
+WR      = 1              WINT    = 7.565065E-7    LINT    = 9.504967E-8
+XL      = 0              XW      = 0              DWG     = -2.13917E-8
+DWB     = 3.857544E-8    VOFF    = -0.0877184     NFACTOR = 0.2508342
+CIT     = 0              CDSC    = 2.924806E-5    CDSCD   = 1.497572E-4
+CDSCB   = 1.091488E-4    ETA0    = 0.15903        ETAB    = -0.0240819
+DSUB    = 0.2873         PCLM    = 1.7338623      PDIBLC1 = 4.152377E-3
+PDIBLC2 = 1E-3           PDIBLCB = -1E-3          DROUT   = 0.0707408
+PSCBE1  = 3.324052E9     PSCBE2  = 1.718711E-6    PVAG    = 15
```

```
+DELTA   = 0.01           RSH     = 74.4          MOBMOD  = 1
+PRT     = 0              UTE     = -1.5          KT1     = -0.11
+KT1L    = 0              KT2     = 0.022         UA1     = 4.31E-9
+UB1     = -7.61E-18      UC1     = -5.6E-11      AT      = 3.3E4
+WL      = 0              WLN     = 1             WW      = 0
+WWN     = 1              WWL     = 0             LL      = 0
+LLN     = 1              LW      = 0             LWN     = 1
+LWL     = 0              CAPMOD  = 2             XPART   = 0.5
+CGDO    = 2.3E-10        CGSO    = 2.3E-10       CGBO    = 1E-9
+CJ      = 3.021506E-4    PB      = 0.7425959     MJ      = 0.4286385
+CJSW    = 1.472483E-10   PBSW    = 0.99          MJSW    = 0.1
+CJSWG   = 3.9E-11        PBSWG   = 0.99          MJSWG   = 0.1
+CF      = 0               )
```

Appendix F

MATLAB SCRPITS

F.1 Architectural Performance Comparison Simulation Scripts

F.1.1 Summarized Comparison Simulation with Signal Model A:

AllCaseAComparison.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Main Setting
warning off MATLAB:divideByZero;
k=1.38e-23;
T=300;
Nc=1e-10;
tFidx=[0 0.002 200];
Pin=1;
Fc=0.01:0.01:1.0;
FtType=[6];
Fs=2;
Kord=100;
T0=0.5;
C=1e-5;
R=1;
Nk=k*T/C;

AllSnrNyq=zeros(3,length(Fc));
AllSnrOvs=zeros(4,length(Fc));

%Input Signal Generation
Fidx=tFidx(2):tFidx(2):tFidx(3);
SigIdx=find(Fidx==1);
Bf=SigSpecGen(1,3,tFidx,Pin);
SH=SigSpecGen(1,3,tFidx,Pin);
SX=SigSpecGen(0,1,tFidx,Pin);
NX=SigSpecGen(0,1,tFidx,Nc);
```

```
Lpf=zeros(1,length(SX));
SiBf=zeros(1,length(SX));
SiNBf=zeros(1,length(SX));
NiBf=zeros(1,length(SX));
NiNBf=zeros(1,length(SX));
SnrBf=zeros(length(FtType),length(Fc));
SnrNBf=zeros(length(FtType),length(Fc));
SnrOvBf=zeros(length(FtType),length(Fc));
SnrOvNBf=zeros(length(FtType),length(Fc));

for k=1:length(Fc),
    for m=1:length(FtType),
        Lpf=SigSpecGen(Fc(k),FtType(m),tFidx,Pin);
        SiBf=((SX.*Bf).*Lpf).*SH;
        SiNBf=(SX.*Lpf).*SH;
        NiBf=((NX.*Bf+Nc).*Lpf+Nc).*SH+Nc+Nk;
        NiNBf=(NX.*Lpf+Nc).*SH+Nc+Nk;
        SpBf=sum(SiBf(1:SigIdx));
        SpNBf=sum(SiNBf(1:SigIdx));
        NpBf=[sum(SiBf(SigIdx+1:length(SiBf))) sum(NiBf(1:SigIdx))
sum(NiBf(SigIdx+1:length(SiBf)))];
        NpNBf=[sum(SiNBf(SigIdx+1:length(SiNBf))) sum(NiNBf(1:SigIdx))
sum(NiNBf(SigIdx+1:length(SiNBf)))];
        SnrBf(m,k)=10*log10(SpBf/sum(NpBf));
        SnrNBf(m,k)=10*log10(SpNBf/sum(NpNBf));
        SnrOvBf(m,k)=10*log10(SpBf/NpBf(2));
        SnrOvNBf(m,k)=10*log10(SpNBf/NpNBf(2));
    end
end

AllSnrNyq(1,:)=SnrBf;
AllSnrOvs(1,:)=SnrOvNBf;

%DirSHAnalCaseAVa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Main Setting
warning off MATLAB:divideByZero;
k=1.38e-23;
T=300;
Nc=1e-10;
tFidx=[0 0.002 200];
Pin=1;
Fc=0.01:0.01:1.0;
BfType=[6];
Fs=2;
Kord=100;
T0=0.5;
C=1e-10;
R=1;
Nk=k*T/C;

%Input Signal Generation
Fidx=tFidx(2):tFidx(2):tFidx(3);
SigIdx=find(Fidx==1);
Bf=zeros(size(Fidx));
```

```
SH=SigSpecGen(1,3,tFidx,Pin);
SX=SigSpecGen(0,1,tFidx,Pin);
NX=SigSpecGen(0,1,tFidx,Nc);
SiBf=zeros(size(SX));
SiNBf=zeros(size(SX));
NiBf=zeros(size(SX));
NiNBf=zeros(size(SX));
SnrBf=zeros(length(BfType),length(Fc));
SnrNBf=zeros(length(BfType),length(Fc));
SnrOvBf=zeros(length(BfType),length(Fc));
SnrOvNBf=zeros(length(BfType),length(Fc));

for k=1:length(Fc),
    for m=1:length(BfType),
        Bf=BfSpecGen(Fc(k),BfType(m),tFidx,Pin);
        SiBf=(SX.*Bf).*SH;
        SiNBf=SX.*SH;
        NiBf=(NX.*Bf+Nc).*SH+Nc+Nk;
        NiNBf=NX.*SH+Nc+Nk;
        SpBf=sum(SiBf(1:SigIdx));
        SpNBf=sum(SiNBf(1:SigIdx));
        NpBf=[sum(SiBf(SigIdx+1:length(SiBf))) sum(NiBf(1:SigIdx))
sum(NiBf(SigIdx+1:length(SiBf)))];
        NpNBf=[sum(SiNBf(SigIdx+1:length(SiNBf))) sum(NiNBf(1:SigIdx))
sum(NiNBf(SigIdx+1:length(SiNBf)))];
        SnrBf(m,k)=10*log10(SpBf/sum(NpBf));
        SnrNBf(m,k)=10*log10(SpNBf/sum(NpNBf));
        SnrOvBf(m,k)=10*log10(SpBf/NpBf(2));
        SnrOvNBf(m,k)=10*log10(SpNBf/NpNBf(2));
    end
end

AllSnrNyq(2,:)=SnrBf;
AllSnrOvs(2,:)=SnrOvNBf;

%DirADCAnanl
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Main Setting
warning off MATLAB:divideByZero;
k=1.38e-23;
T=300;
Nc=1e-10;
tFidx=[0 0.002 200];
Pin=1;
Fc=0.01:0.01:1.0;
BfType=[6];
Fs=2;
Kord=100;
T0=0.5;
C=1e-10;
R=1;
Nk=k*T/C;

%Input Signal Generation
Fidx=tFidx(2):tFidx(2):tFidx(3);
```

```
SigIdx=find(Fidx==1);
Bf=zeros(size(Fidx));
SH=SigSpecGen(1,3,tFidx,Pin);
SX=SigSpecGen(0,1,tFidx,Pin);
NX=SigSpecGen(0,1,tFidx,Nc);
SiBf=zeros(size(SX));
SiNBf=zeros(size(SX));
NiBf=zeros(size(SX));
NiNBf=zeros(size(SX));
SnrBf=zeros(length(BfType),length(Fc));
SnrNBf=zeros(length(BfType),length(Fc));
SnrOvBf=zeros(length(BfType),length(Fc));
SnrOvNBf=zeros(length(BfType),length(Fc));

WtBar= waitbar(0,'Processing');
for k=1:length(Fc),
    waitbar(k/length(Fc),WtBar)
    for m=1:length(BfType),
        Bf=BfSpecGen(Fc(k),BfType(m),tFidx,Pin);
        SiBf=SX.*Bf;
        SiNBf=SX;
        NiBf=NX.*Bf+Nc;
        NiNBf=NX;
        SpBf=sum(SiBf(1:SigIdx));
        SpNBf=sum(SiNBf(1:SigIdx));
        NpBf=[sum(SiBf(SigIdx+1:length(SiBf))) sum(NiBf(1:SigIdx))
sum(NiBf(SigIdx+1:length(SiBf)))];
        NpNBf=[sum(SiNBf(SigIdx+1:length(SiNBf))) sum(NiNBf(1:SigIdx))
sum(NiNBf(SigIdx+1:length(SiNBf)))];
        SnrBf(m,k)=10*log10(SpBf/sum(NpBf));
        SnrNBf(m,k)=10*log10(SpNBf/sum(NpNBf));
        SnrOvBf(m,k)=10*log10(SpBf/NpBf(2));
        SnrOvNBf(m,k)=10*log10(SpNBf/NpNBf(2));
    end
end
close(WtBar)

AllSnrNyq(3,:)=SnrBf;
AllSnrOvs(3,:)=SnrOvBf;
AllSnrOvs(4,:)=SnrOvNBf;
```

F.1.2 Signal Spectrum Generator: SigSpecGen.m

```
function x=SigSpecGen(Fc,Type,tFidx,Pin)
%
Fidx=tFidx(2):tFidx(2):tFidx(3);
switch Type
    case 1, %flat spectrum
        x=Pin*ones(size(Fidx));
    case 2, %ideal lpf
```

```
        x=(-sign(sign(Fidx-Fc)+0.5)+1)*0.5*Pin;
    otherwise
        if Type>=3,
            x=Pin./(1+(Fidx/Fc).^(2*(Type-2)));
        end
end
```

F.1.3 Buffer Frequency Spectrum Generator: BfSpecGen.m

```
function x=BfSpecGen(Fc,Type,tFidx,Pin)
Fidx=tFidx(2):tFidx(2):tFidx(3);
switch Type
    case 1, %flat spectrum
        x=Pin*ones(size(Fidx));
    case 2, %ideal lpf
        x=(-sign(sign(Fidx-Fc)+0.5)+1)*0.5*Pin;
    otherwise
        if Type>=3,
            x=Pin./(abs(1+(j*Fidx/Fc)).^(2*(Type-2)));
        end
end
```

F.2 ΔΣ Decoding Performance Simulation Scripts

F.2.1 Ideal Performance Simulation: DsmDecIdealTest.m

```
OSR=2.^(2:11);
AlgoNo=4;
Mse=zeros(AlgoNo,length(OSR));
Time=zeros(AlgoNo,length(OSR));
for m=1:length(OSR),
    N=OSR(m);
    X=[0:(1/(N*10)):1];
    Err=zeros(AlgoNo,length(X));
    tIdx=0:(N-1);
    Hopt=6*(tIdx+1).*(N-tIdx)/(N*(N+1)*(N+2));
    WB = waitbar(0,['Processing OSR ' num2str(OSR(m))]);
    for n=1:AlgoNo,
              waitbar(n/AlgoNo,WB);
              tic;
            for x=1:length(X),
                  [Y,Q]=DSM1(X(x)*ones(N+1,1),0.5);
              switch n
```

```
            case 1, %Linear Filter
                  Output=sum(Y.*Hopt');
                  Err(1,x)=Output-X(x);
            case 2, %Zoomer
                  Output=SingleZoomer(X(x),N,0,0.5,1,1,0.5);
                  Err(2,x)=Output-X(x);
            case 3, %NlogN
                        [A,B,Tp,Iter]=NlogNDecoder(Y(1:N),1);
                              Output=A/B;
                  Err(3,x)=(A/B)-X(x);
                  RcCount(x)=Iter;
            case 4, %Daeik's algorithm
                              Output=DaeikDSMDec(Y(1:N));
                  Err(4,x)=Output-X(x);
            otherwise
                  end %switch
            end %for x
        Time(n,m)=toc/length(X);
    end %for n
    close(WB);
    Mse(:,m)=mean(Err.^2,2);
end %for m
```

F.2.2 Ideal First-order ΔΣ ADC Model: DSM1.m

```
function [Y,Q]=DSM1(X,Ref)
%Delta Sigma Modulator 1st order Model Function
%X: Input Sequence
%Ref: Switching reference
%Y: Modulated Sequence
%Q: Quantization Noise Sequence
W=0;
X=X(:);
L=length(X);
Y=zeros(L,1);
Q=zeros(L,1);
for i=1:L,
    if W>Ref,
        Y(i)=1;
    end
    Q(i)=W-Y(i);
    W=W+X(i)-Y(i);
end
Y=Y(2:length(Y));
Q=Q(2:length(Q));
```

F.2.3 Ideal Single Zoomer Decoding: SingleZoomer.m

```
function Y=SingleZoomer(Xn,N,Q0,DC,Q1,K,Init)
% Xn : Input value, a number for dc, or a sequence
% N : Number of iteration;
% Q0 : Off value of Delta-sigma
% DC : On/Off decision point, half between Q0 and Q1
% Q1 : On value of Delta-sigma;
% K : range scaler 0-1
% Init : Initial integrator condition Q0-Q1
if length(Xn)==1,
    Xn=Xn*ones(N,1);
end
L=Q1-K*(Q1-Q0);
U=Q0+K*(Q1-Q0);
Un=zeros(N,1);
Qn=zeros(N,1);
Un(1)=Init;
if Un(1)>DC,
    Qn(1)=Q1;
else
    Qn(1)=Q0;
end
S=Qn(1);

Un(2)=Un(1)+Xn(1)-Qn(1);
Qn(2)=Quantize2(Un(2),Q0,DC,Q1);
if Un(2)>DC,
    Qn(2)=Q1;
else
    Qn(2)=Q0;
end
for n=2:N-1,
    S=S+Qn(n);
    Xb=S/n;
    Un(n+1)=Un(n)+Xn(n)-Qn(n);
    if Un(n+1)>DC,
        Qn(n+1)=Q1;
        L=max([L Xb]);
    else
        Qn(n+1)=Q0;
        U=min([U Xb]);
    end
end
Y=(L+U)/2;
```

F.2.4 Ideal Recursive Decoding: NlogNDecoder.m

```
function [A,B,Tp,Iter]=NlogNDecoder(Y,Iter);
```

```
%
[L,J,Tp]=NlogNTypeCheck(Y);
[A, B]=NlogNDecRecFinal(L,J,Y,Tp);
if Tp==1,
    A=B-A;
end
if B~=0 & A~=0,
    return
end
Lmax=max(L);
W=L-Lmax+1;
[rA,rB,rTp,Iter]=NlogNDecoder(W,Iter+1);
B=Lmax*rB+rA;
if Tp==0,
    A=rB;
else
    A=B-rB;
end
return
```

F.2.5 Ideal Correlative Decoding: QNCorrDec.m

```
function X=QNCorrDec(Q)
%Q should have even numbers of output
N=length(Q);
K=N/2;
R=zeros(K,1);
for k=1:K,
    R(k)=sum(Q(k+1:N).*Q(1:(N-k)))/(N-k);
end
[tL, Lest]=max(R);
X=sum(Q(1:Lest))/Lest;
```

F.2.6 Ideal Proposed Decoding: DaeikDSMDec.m

```
function Est=DaeikDSMDec(Y)
%
L=length(Y);
BL=0;
BU=1;
tS=0;
for k=1:L,
    tS=tS+Y(k);
    tXL=(tS-0.5)/k;
    tXU=(tS+0.5)/k;
    BL=max([tXL BL]);
```

```
    BU=min([tXU BU]);
end
Est=(BL+BU)/2;
```

F.3 Clock Jitter Generation and Simulation Analysis Scripts

F.3.1 Clock Generation Main Script: GenClockFiles.m

```
vA=10.^(-4:0.2:-1);
for M=1:length(vA),
    for K=1:20,
    ClockJitterGenerator(vA(M),K,M);
    end
end
```

F.3.2 Clock Jitter Generation Script: ClockJitterGenerator.m

```
function ClockJitterGenerator(A,K,M)
SampleNo=280;
Von=5;
RFTm=0.001;
UnitTime=0.5;
ClkDuty=0.25;
TimeUnit='u';
VoltUnit='v';

fCS=['ClkSet' num2str(M) 'A' num2str(K) '.spice'];
fCR=['ClkRst' num2str(M) 'A' num2str(K) '.spice'];

FidCS=fopen(fCS,'w');
FidCR=fopen(fCR,'w');
fprintf(FidCS,'VSet\tVSet\t0\tPWL\n+0%s\t%d%s\n', TimeUnit, 0, VoltUnit);
fprintf(FidCR,'VRst\tVRst\t0\tPWL\n+0%s\t%d%s\n', TimeUnit, Von, VoltUnit);
JR=[];
JS=[];
for k=1:SampleNo+10,
    Jitter=A*UnitTime*randn(1,1);
    TmA=(k-1)*UnitTime+UnitTime*ClkDuty-RFTm/2+Jitter;
    TmB=(k-1)*UnitTime+UnitTime*ClkDuty+RFTm/2+Jitter;
    Jitter=A*UnitTime*randn(1,1);
    TmC=k*UnitTime-RFTm/2+Jitter;
    TmD=k*UnitTime+RFTm/2+Jitter;
```

```
    fprintf(FidCR,'+%f%s\t%d%s\t', TmA, TimeUnit, Von, VoltUnit);
    fprintf(FidCR,'+%f%s\t%d%s\t', TmB, TimeUnit, 0, VoltUnit);
    fprintf(FidCR,'+%f%s\t%d%s\t', TmC, TimeUnit, 0, VoltUnit);
    fprintf(FidCR,'+%f%s\t%d%s\n', TmD, TimeUnit, Von, VoltUnit);
    Jitter=A*UnitTime*randn(1,1);
    TmA=(k-1)*UnitTime+UnitTime*0.5-RFTm/2+Jitter;
    TmB=(k-1)*UnitTime+UnitTime*0.5+RFTm/2+Jitter;
    Jitter=A*UnitTime*randn(1,1);
    TmC=(k-1)*UnitTime+UnitTime*ClkDuty+UnitTime*0.5-RFTm/2+Jitter;
    TmD=(k-1)*UnitTime+UnitTime*ClkDuty+UnitTime*0.5+RFTm/2+Jitter;
    fprintf(FidCS,'+%f%s\t%d%s\t', TmA, TimeUnit, 0, VoltUnit);
    fprintf(FidCS,'+%f%s\t%d%s\t', TmB, TimeUnit, Von, VoltUnit);
    fprintf(FidCS,'+%f%s\t%d%s\t', TmC, TimeUnit, Von, VoltUnit);
    fprintf(FidCS,'+%f%s\t%d%s\n', TmD, TimeUnit, 0, VoltUnit);
end

fclose(FidCS);
fclose(FidCR);
```

F.3.3 Generated Clock Jitter Spice Input Example: ClkRst1A1.m

```
VRst    VRst    0   PWL
+0u 5v
+0.124429u  5v  +0.125429u  0v  +0.499487u  0v  +0.500487u  5v
+0.624513u  5v  +0.625513u  0v  +0.999486u  0v  +1.000486u  5v
+1.124428u  5v  +1.125428u  0v  +1.499480u  0v  +1.500480u  5v
+1.624520u  5v  +1.625520u  0v  +1.999454u  0v  +2.000454u  5v
+2.124511u  5v  +2.125511u  0v  +2.499479u  0v  +2.500479u  5v
+2.624452u  5v  +2.625452u  0v  +2.999482u  0v  +3.000482u  5v
(.....)
+142.624493u    5v  +142.625493u    0v  +142.999637u    0v  +143.000637u    5v
+143.124386u    5v  +143.125386u    0v  +143.499467u    0v  +143.500467u    5v
+143.624417u    5v  +143.625417u    0v  +143.999524u    0v  +144.000524u    5v
+144.124587u    5v  +144.125587u    0v  +144.499478u    0v  +144.500478u    5v
+144.624537u    5v  +144.625537u    0v  +144.999599u    0v  +145.000599u    5v
```

F.3.4 Simulation Output Analysis Script: DataProcess.m

```
vA=10.^(-4:0.2:-1);
DSMStr='DSMSimOut'; %DSMSimOutT3AG953.out
SSCStr='SSCSimOut'; %SSCSimOutT3AG953.out
VPre='V';
NPre='N';
IPre='I';

IRange=10:10:990;
```

```
NI=length(IRange);
VRange=1:16;
NV=length(VRange);
NRange=1:20;
NN=length(NRange);
DSMSwpA=zeros(length(VRange)*length(NRange), length(IRange));

for v=1:NV,
    V=VRange(v);
    for n=1:NN,
        N=NRange(n);
        for k=1:NI,
            Input=IRange(k);
            FileSuffix=[VPre num2str(V) NPre num2str(N) IPre num2str(Input)];
            %LOAD Data
            eval(['load Data\' DSMStr FileSuffix '.out;']);
            eval(['DSMMat=' DSMStr FileSuffix ';']);
            eval(['clear ' DSMStr FileSuffix ';']);
            DSMSwpA(NN*(v-1)+n,k)=DSMDecA(DSMMat);
        end
    end
end

DSMSwpM=[];
DSMSwpV=[];
DSMSwpMSE=[];
for v=1:NV,
    DSMSwpM=[DSMSwpM; mean(DSMSwpA((1+NN*(v-1)):(NN*v),:))];
    DSMSwpV=[DSMSwpV; std(DSMSwpA((1+NN*(v-1)):(NN*v),:),0,1)];
    %DSMSwpMSE=[DSMSwpMSE; sum(DSMSwpV(v,:))/length(DSMSwpV(v,:))];
    tD=[];
    for k=1:NN,
        tD=[tD DSMSwpA(k+NN*(v-1),:)-DSMSwpM(v,:)];
    end
    tD=tD.^2;
    DSMSwpMSE=[DSMSwpMSE; sqrt(sum(tD)/length(tD)^2)];
end

tin=0.5e-6*280;
fin=1/tin;
dt=10.^(-8:0.1:-5);
tdt=dt*fin;
kj=2^8*pi*sqrt(6)*dt/tin;
nred=log(1+kj.^2)/2*log(2)*6.02;
```

F.4 Data Acquisition Unit Output Statistical Analysis Script:

DataAnalyze.m

```
Files_=dir('*.txt');
Files={Files_.name};
Data=[];
Time=[];
for k=1:length(Files),
    FileName=Files{k};
    eval(['load ' FileName ';']);
    eval(['tData=X' FileName(1:10) ';']);
    eval(['clear X' FileName(1:10) ';']);
    Time=[Time;
24*60*60*str2num(FileName(3:4))+60*60*str2num(FileName(5:6))+60*str2num(FileNam
e(7:8))+str2num(FileName(9:10))];
    Data=[Data; tData(1,:) tData(2,:) tData(3,:)];
end

dData=[];
mData=[];
mpData=[];
pData=[];
rData=[];
rdData=[];
rmData=[];

for k=1:length(Files),
    dData=[dData; Data(k,:)-Data(1,:)];
    pData=[pData; sum(Data(k,:))];
    mData=[mData; sum((dData(k,:)/24).^2)];
    mpData=[mpData; sum((dData(k,:)/pData(k)).^2)];
    rData=[rData; Data(k,:)/pData(k)];
    rdData=[rdData; rData(k,:)-rData(1,:)];
    rmData=[rmData; sum((rdData(k,:)/24).^2)];
end
dTime=Time-Time(1);
dmTime=dTime/60;
```

Appendix G

HSPICE SIMULATION SCRIPTS

G.1 HSPICE Scripts

G.1.1 DSM.cir

```
*Delta Sigma ADC Simulation
*.option post node
.option fast accuracy
.option gshunt=1e-15 cshunt=1e-15
.include 'DSM.spice'
.include 'ami150cmosmodel.spice'

*PowerSupply
VddA    Vdd!    0   PVsupply
VddB    Vdd     0   PVsupply
VgndA   GND!    0   0
VgndB   gnd     0   0
.param PVsupply=5

*Circuit Biasing
Vbs     Vbs     0   PVmid
Vref    Vref    0   PVmid
Ibs     Ibs     0   -1.05u
.param PVmid='PVsupply/2'

*Capacitor
Cint    VInt    0   1000fF

*Clocking
.include 'CktClkSet.spice'
.include 'CktClkRst.spice'
*Vset   VSet    0   PULSE(0 PVsupply 0 PTrsfl PTrsfl PClkDt PClock)
*Vrst   VRst    0   PULSE(0 PVsupply PClkPhs PTrsfl PTrsfl PClkDt PClock)
*.param     PTrsfl=1p
*.param     PClock=0.5u
```

```
*.param     PClkDtRt=0.25
*.param     PClkDt='PClock*PClkDtRt'
*.param     PClkPhs='PClock*0.5'

*Detector Input
.include 'CktDetInput.spice'

*Circuit Temperature
*.include 'CktTemp.spice'

*Simulation Command
.ic     v(VInt)=PVmid
.tran   PSimStp PSimEndTm
.param  PSimStp=0.5u
.param  PSimEndTm=150u
.print  v(Vout)
.end
```

G.1.2 SSC.cir

```
*Single-Slope ADC Simulation
*.option post node
.option fast accuracy
.option gshunt=1e-15 cshunt=1e-15
.include 'SSC.spice'
.include 'ami150cmosmodel.spice'

*PowerSupply
VddA    Vdd!    0   PVsupply
VddB    Vdd     0   PVsupply
VgndA   GND!    0   0
VgndB   gnd     0   0
.param PVsupply=5

*Voltage Biasing
VDtBs   VDtBs   0   PVmid
VApRf   VApRf   0   PVmid
.param PVmid='PVsupply/2'

*Current Biasing
IItBs   IItBs   0   0.08u
IApBs1  IApBs1  0   -1.0u
IApBs2  IApBs2  0   -2.0u
IApBs3  IApBs3  0   -2.0u
IApBs4  IApBs4  0   80u
IApBs5  IApBs5  0   1.15u

*Clocking
VRst    VRst    0   pulse(0 PVsupply 0 PTrsfl PTrsfl PSmpTime PCycle)
*VRst   VRst    0   0
```

```
VnRst   VnRst   0   pulse(PVsupply 0 0 PTrsfl PTrsfl PSmpTime PCycle)
*VnRst  VnRst   0   5
.param PTrsfl=1p
.param PCycle=10000u
.param PSmpTime=1u

*Capacitor
CInt    VInt    0   1000fF
CSmp    VSmp    0   1000fF

*Detector Input
.include 'CktDetInput.spice'

*Circuit Temperature
.include 'CktTemp.spice'

*Simulation Command
.ic v(VInt)=0
.tran PSimStp PSimEndTm
*.dc IDet 0 1u 0.01u
.param PSimStp=0.5u
.param PSimEndTm=130u
.print v(Vout)
.end
```

G.1.3 CktDetInput.spice

```
IDet    IDet    0   500n
```

G.1.4 CktTemp.spice

```
.temp   25
```

G.2 Circuit Extraction

G.2.1 ΔΣ ADC Extraction: DSM.spice

```
* # FILE NAME: /HOME/AZALEA/FRONTEND/CADENCE/SIMULATION/DSM1V1/HSPICES/
* extracted/netlist/DSM1v1.C.raw
```

```
* Netlist output for hspiceS.
* Generated on Mar 5 16:29:12 2004
* File name: ConverterComparison_DSM1v1_extracted.S.
* Subcircuit for cell: DSM1v1.
* Generated for: hspiceS.
* Generated on Mar  5 16:29:12 2004.
C24 0 22  3.13343999320615E-15 M=1.0
C26 VSET 0  2.80704008214644E-15 M=1.0
C28 IBS 0  2.78528001748971E-15 M=1.0
C30 OUT2 0  5.03744003907641E-15 M=1.0
C32 VDD! 22  2.41535997711655E-15 M=1.0
C34 VDD! 21  4.70015998980922E-15 M=1.0
C36 21 22  2.76239996351848E-15 M=1.0
C38 0 20  5.82079982561971E-15 M=1.0
C40 0 19  6.33840006034666E-15 M=1.0
C42 0 18  4.17120019622665E-15 M=1.0
C44 0 VRST  23.6479994522023E-15 M=1.0
C46 0 VINT  12.8159998211792E-15 M=1.0
C48 VSET 0  23.1424004037886E-15 M=1.0
C50 IBS 0  23.9192007671883E-15 M=1.0
C52 OUT2 22  5.09136003891625E-15 M=1.0
C54 OUT2 0  16.2648808131735E-15 M=1.0
C56 OUT1 21  5.09136003891625E-15 M=1.0
C58 OUT1 0  17.8371992740414E-15 M=1.0
C60 VOUT 0  2.14879997916504E-15 M=1.0
C62 IDET 0  14.108400151001E-15 M=1.0
C64 VDD! 7  3.69040015492611E-15 M=1.0
C66 VDD! 21  4.13136000155254E-15 M=1.0
C68 VDD! 18  2.57040004074903E-15 M=1.0
C70 VDD! 0  2.48880006298411E-15 M=1.0
C72 VDD! VSET  3.08576008508873E-15 M=1.0
C74 VDD! OUT2  2.28383990084696E-15 M=1.0
C76 VDD! OUT1  3.82527998738099E-15 M=1.0
C78 VDD! IDET  16.524799649302E-15 M=1.0
C80 VBS 0  23.3952007750284E-15 M=1.0
C82 VREF 0  22.5232006846208E-15 M=1.0
C84 0 20  2.5619199703976E-15 M=1.0
C86 0 19  5.28880003078944E-15 M=1.0
C88 0 VRST  3.3892799421772E-15 M=1.0
C90 0 VINT  2.90800005671205E-15 M=1.0
C92 IBS 0  2.01391993495193E-15 M=1.0
C94 OUT2 0  3.56192007990421E-15 M=1.0
C96 OUT1 0  5.37248010971459E-15 M=1.0
C98 IDET 1  2.74816006912571E-15 M=1.0
C100 VDD! VRST  2.41871994506007E-15 M=1.0
C102 VDD! VINT  2.70879993037446E-15 M=1.0
C104 VDD! VSET  2.3961600695429E-15 M=1.0
C106 VDD! OUT1  4.74255991804993E-15 M=1.0
C108 VBS 2  3.08608005178455E-15 M=1.0
M110 VDD! OUT2 OUT1 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M112 OUT1 21 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M114 VDD! OUT1 OUT2 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
```

```
M116 OUT2 22 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M118 VDD! 21 22 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M120 22 VSET VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M122 VDD! 22 21 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M124 21 VSET VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M126 20 VREF 7 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M128 19 VINT 7 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M130 7 18 12 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=2.55999995507628E-12 PD=11.1999997898238E-6 PS=1.60000001869776E-6 M=1
M132 12 6 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=2.55999995507628E-12
+AS=12.7999997753814E-12 PD=1.60000001869776E-6 PS=11.1999997898238E-6 M=1
M134 18 18 6 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M136 6 6 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M138 VINT 1 11 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=2.55999995507628E-12 PD=11.1999997898238E-6 PS=1.60000001869776E-6 M=1
M140 11 5 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=2.55999995507628E-12
+AS=12.7999997753814E-12 PD=1.60000001869776E-6 PS=11.1999997898238E-6 M=1
M142 2 1 10 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=2.55999995507628E-12 PD=11.1999997898238E-6 PS=1.60000001869776E-6 M=1
M144 10 5 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=2.55999995507628E-12
+AS=12.7999997753814E-12 PD=1.60000001869776E-6 PS=11.1999997898238E-6 M=1
M146 1 1 5 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M148 5 5 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M150 VDD! OUT2 VOUT VDD!  CMOSP  L=1.6E-6 W=19.2E-6 AD=76.7999969175648E-12
+AS=46.0800009260964E-12 PD=27.1999997494277E-6 PS=4.80000016978011E-6 M=1
M152 VOUT OUT2 VDD! VDD!  CMOSP  L=1.6E-6 W=19.2E-6 AD=46.0800009260964E-12
+AS=76.7999969175648E-12 PD=4.80000016978011E-6 PS=27.1999997494277E-6 M=1
M154 20 VRST 19 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M156 0 21 9 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=2.55999995507628E-12 PD=11.1999997898238E-6 PS=1.60000001869776E-6 M=1
M158 9 OUT2 OUT1 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=2.55999995507628E-12
+AS=12.7999997753814E-12 PD=1.60000001869776E-6 PS=11.1999997898238E-6 M=1
M160 0 22 8 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=2.55999995507628E-12 PD=11.1999997898238E-6 PS=1.60000001869776E-6 M=1
M162 8 OUT1 OUT2 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=2.55999995507628E-12
+AS=12.7999997753814E-12 PD=1.60000001869776E-6 PS=11.1999997898238E-6 M=1
M164 0 18 18 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M166 0 4 4 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M168 4 IBS IBS 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M170 0 OUT2 17 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
```

```
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M172 17 OUT1 4 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M174 0 17 3 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M176 3 IBS VINT 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M178 VBS 2 2 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M180 0 19 20 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M182 IDET 2 1 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M184 20 VSET 22 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M186 0 20 19 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M188 19 VSET 21 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M190 0 OUT2 VOUT 0  CMOSN  L=1.6E-6 W=8E-6 AD=31.9999998721343E-12
+AS=19.1999992293912E-12 PD=15.9999999596039E-6 PS=4.80000016978011E-6 M=1
M192 VOUT OUT2 0 0  CMOSN  L=1.6E-6 W=8E-6 AD=19.1999992293912E-12
+AS=31.9999998721343E-12 PD=4.80000016978011E-6 PS=15.9999999596039E-6 M=1
```

G.2.2 Single-slope ADC Extraction: SSC.spice

```
* # FILE NAME: /HOME/AZALEA/FRONTEND/CADENCE/SIMULATION/SSADC2V3/HSPICES/
* extracted/netlist/SSADC2v3.C.raw
* Netlist output for hspiceS.
* Generated on Mar 4 17:25:46 2004
* File name: ConverterComparison_SSADC2v3_extracted.S.
* Subcircuit for cell: SSADC2v3.
* Generated for: hspiceS.
* Generated on Mar  4 17:25:46 2004.
C25 S3 0  3.11167992854942E-15 M=1.0
C27 0 9  2.16720007587825E-15 M=1.0
C29 0 4  2.26240001464429E-15 M=1.0
C31 0 VRST  35.0872014277283E-15 M=1.0
C33 0 VINT  7.01600016012873E-15 M=1.0
C35 VDTBS 0  33.0647984578096E-15 M=1.0
C37 S2B 0  2.55279996665451E-15 M=1.0
C39 S2A 0  2.28560003531135E-15 M=1.0
C41 IAPBS5 0  32.520800017765E-15 M=1.0
C43 IAPBS4 0  32.520800017765E-15 M=1.0
C45 VOUT 0  33.5831995621882E-15 M=1.0
C47 IAPBS3 0  33.2735988555714E-15 M=1.0
C49 IAPBS2 0  33.2735988555714E-15 M=1.0
C51 IAPBS1 0  33.2735988555714E-15 M=1.0
C53 IDET 0  18.8520006490266E-15 M=1.0
C55 S1B 0  2.27240008561958E-15 M=1.0
```

```
C57 VSMP 0  8.16079991669376E-15 M=1.0
C59 VDD! 4  3.15279999000683E-15 M=1.0
C61 VDD! IDET  25.2567996221085E-15 M=1.0
C63 IITBS 0  32.5208000177765E-15 M=1.0
C65 NOUT 0  4.0124001533413E-15 M=1.0
C67 VAPRF 0  33.0647984578096E-15 M=1.0
C69 VNRST 0  33.0647984578096E-15 M=1.0
C71 S5 0  4.62719995986452E-15 M=1.0
C73 S5 VDD!  2.48399992727202E-15 M=1.0
C75 S4 0  3.20959990186733E-15 M=1.0
C77 S3 0  3.27999998648719E-15 M=1.0
C79 S3 VDD!  2.21760001854754E-15 M=1.0
C81 0 4  4.78055993364613E-15 M=1.0
C83 0 VRST  10.0695996747596E-15 M=1.0
C85 0 VINT  5.64007984060885E-15 M=1.0
C87 VDTBS 2  2.47071993296673E-15 M=1.0
C89 IAPBS5 0  2.25952010262363E-15 M=1.0
C91 IAPBS4 0  2.01391993495193E-15 M=1.0
C93 VOUT 0  2.55423992266484E-15 M=1.0
C95 IAPBS3 0  2.37815998413904E-15 M=1.0
C97 IAPBS2 0  2.04199992418636E-15 M=1.0
C99 IDET 1  2.24128009513745E-15 M=1.0
C101 VSMP 0  3.9943998561792E-15 M=1.0
C103 VDD! VRST  7.8609604177611E-15 M=1.0
C105 VDD! IAPBS1  2.03456001029415E-15 M=1.0
C107 NOUT VDD!  2.4076799293838E-15 M=1.0
C109 VAPRF VDD!  2.11119997487819E-15 M=1.0
C111 VNRST 0  5.39855983064407E-15 M=1.0
C113 VNRST VDD!  2.53839998303471E-15 M=1.0
C115 S3 0  8.20384020184259E-15 M=1.0
M117 VSMP VRST S5 VDD!  CMOSP  L=1.6E-6 W=9.6E-6 AD=38.3999984587824E-12
+AS=38.3999984587824E-12 PD=17.5999994098675E-6 PS=17.5999994098675E-6 M=1
M119 NOUT 13 VDD! VDD!  CMOSP  L=1.6E-6 W=12.8E-6 AD=51.1999991015255E-12
+AS=51.1999991015255E-12 PD=20.800000129384E-6 PS=20.800000129384E-6 M=1
M121 VOUT NOUT VDD! VDD!  CMOSP  L=1.6E-6 W=19.2E-6 AD=46.0800009260964E-12
+AS=76.7999969175648E-12 PD=4.80000016978011E-6 PS=27.1999997494277E-6 M=1
M123 VDD! NOUT VOUT VDD!  CMOSP  L=1.6E-6 W=19.2E-6 AD=76.7999969175648E-12
+AS=46.0800009260964E-12 PD=27.1999997494277E-6 PS=4.80000016978011E-6 M=1
M125 14 14 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M127 1 1 14 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M129 21 14 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=2.55999995507628E-12
+AS=12.7999997753814E-12 PD=1.60000001869776E-6 PS=11.1999997898238E-6 M=1
M131 2 1 21 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=2.55999995507628E-12 PD=11.1999997898238E-6 PS=1.60000001869776E-6 M=1
M133 22 14 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=2.55999995507628E-12
+AS=12.7999997753814E-12 PD=1.60000001869776E-6 PS=11.1999997898238E-6 M=1
M135 4 1 22 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=2.55999995507628E-12 PD=11.1999997898238E-6 PS=1.60000001869776E-6 M=1
M137 4 4 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M139 S1B 4 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M141 S2A S2A VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
```

```
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M143 S2B S2A VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M145 S3 S2B VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M147 IAPBS4 IAPBS4 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6
+AD=12.7999997753814E-12 AS=12.7999997753814E-12 PD=11.1999997898238E-6
+PS=11.1999997898238E-6 M=1
M149 S4 IAPBS4 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M151 IAPBS5 IAPBS5 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6
+AD=12.7999997753814E-12 AS=12.7999997753814E-12 PD=11.1999997898238E-6
+PS=11.1999997898238E-6 M=1
M153 S5 IAPBS5 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M155 9 9 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M157 12 12 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M159 13 12 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M161 23 15 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=2.55999995507628E-12
+AS=12.7999997753814E-12 PD=1.60000001869776E-6 PS=11.1999997898238E-6 M=1
M163 VINT IITBS 23 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=2.55999995507628E-12 PD=11.1999997898238E-6 PS=1.60000001869776E-6 M=1
M165 15 15 VDD! VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M167 IITBS IITBS 15 VDD!  CMOSP  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M169 S5 VNRST VSMP 0  CMOSN  L=1.6E-6 W=4E-6 AD=15.9999999360672E-12
+AS=15.9999999360672E-12 PD=12.0000004244503E-6 PS=12.0000004244503E-6 M=1
M171 20 10 0 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=2.55999995507628E-12
+AS=12.7999997753814E-12 PD=1.60000001869776E-6 PS=11.1999997898238E-6 M=1
M173 NOUT 9 20 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=2.55999995507628E-12 PD=4.80000016978011E-6 PS=1.60000001869776E-6 M=1
M175 VDD! VRST NOUT 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M177 VOUT NOUT 0 0  CMOSN  L=1.6E-6 W=8E-6 AD=19.1999992293912E-12
+AS=31.9999998721343E-12 PD=4.80000016978011E-6 PS=15.9999999596039E-6 M=1
M179 0 NOUT VOUT 0  CMOSN  L=1.6E-6 W=8E-6 AD=31.9999998721343E-12
+AS=19.1999992293912E-12 PD=15.9999999596039E-6 PS=4.80000016978011E-6 M=1
M181 IDET 2 1 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M183 VDTBS 2 2 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M185 3 IAPBS1 IAPBS1 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M187 0 3 3 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M189 5 S3 4 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M191 16 IAPBS1 5 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=2.55999995507628E-12
+AS=7.67999986522883E-12 PD=1.60000001869776E-6 PS=4.80000016978011E-6 M=1
M193 0 3 16 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=2.55999995507628E-12 PD=11.1999997898238E-6 PS=1.60000001869776E-6 M=1
```

```
M195 5 VAPRF S1B 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M197 6 IAPBS2 IAPBS2 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M199 0 6 6 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M201 7 S1B S2A 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M203 17 IAPBS2 7 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=2.55999995507628E-12
+AS=7.67999986522883E-12 PD=1.60000001869776E-6 PS=4.80000016978011E-6 M=1
M205 0 6 17 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=2.55999995507628E-12 PD=11.1999997898238E-6 PS=1.60000001869776E-6 M=1
M207 7 4 S2B 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M209 18 IAPBS3 S3 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=2.55999995507628E-12
+AS=12.7999997753814E-12 PD=1.60000001869776E-6 PS=11.1999997898238E-6 M=1
M211 0 8 18 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=2.55999995507628E-12 PD=11.1999997898238E-6 PS=1.60000001869776E-6 M=1
M213 8 IAPBS3 IAPBS3 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M215 0 8 8 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M217 0 S3 S4 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M219 0 S4 S5 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M221 10 9 9 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=7.67999986522883E-12
+AS=12.7999997753814E-12 PD=4.80000016978011E-6 PS=11.1999997898238E-6 M=1
M223 0 10 10 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=7.67999986522883E-12 PD=11.1999997898238E-6 PS=4.80000016978011E-6 M=1
M225 19 9 11 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=2.55999995507628E-12
+AS=12.7999997753814E-12 PD=1.60000001869776E-6 PS=11.1999997898238E-6 M=1
M227 0 10 19 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=2.55999995507628E-12 PD=11.1999997898238E-6 PS=1.60000001869776E-6 M=1
M229 11 VSMP 12 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M231 11 VINT 13 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
M233 0 VRST VINT 0  CMOSN  L=1.6E-6 W=3.2E-6 AD=12.7999997753814E-12
+AS=12.7999997753814E-12 PD=11.1999997898238E-6 PS=11.1999997898238E-6 M=1
```

Appendix H

PERL SCRIPTS

H.1 Spice Simulation Control Scripts

H.1.1 DC Sweep with Model Variation: AcqScriptModel.pl

```
#!/usr/bin/perl
$SimInp="CktDetInput.spice";
$DSMSimOut="DSM.lis";
$SSCSimOut="SSC.lis";
$DSMSimSrc="DSM.cir";
$SSCSimSrc="SSC.cir";
$ModelList="ModelList";
$ModelDir="CmosModel";
$DSMOutNameRoot="DSMSimOut";
$SSCOutNameRoot="SSCSimOut";
$DSMPatFile="DSM.header";
$SSCPatFile="SSC.header";
$IterNo=98;
$IterStep=10;
$IterOffset=10;
$Unit='n';
$x=0;
$Input=0;
$ModelCount=0;

open(MODEL, $ModelList)  || die("Cannot CMOS model list file:$ModelList");
while ($ModelName=<MODEL>)
{
    chop $ModelName;
    system("cp $ModelDir/$ModelName ami150cmosmodel.spice");
    for ($x=0; $x<=$IterNo; $x++)
    {
        $Input=$x*$IterStep+$IterOffset;
        WriteSimInput();
        system("hspice $DSMSimSrc > $DSMSimOut");
```

```
        system("hspice $SSCSimSrc > $SSCSimOut");
        $ModelName=~s/\.spice//;
        $ModelName=~tr/a-z/A-Z/;
        $ModelIdx="M$ModelCount";
        $InputIdx="I$Input";
        $DSMCvrtStr="$DSMOutNameRoot$ModelIdx$InputIdx.out";
        $SSCCvrtStr="$SSCOutNameRoot$ModelIdx$InputIdx.out";
        system("CvrtSimOut.pl -d 2.5 -s 1 -p 0 -m $DSMPatFile < $DSMSimOut >
$DSMCvrtStr");
        system("CvrtSimOut.pl -d 2.5 -s 1 -p 0 -m $SSCPatFile < $SSCSimOut >
$SSCCvrtStr");
    }
    $ModelCount++;
}
close(MODEL);

sub WriteSimInput {
    open(FOUT, ">$SimInp") || die("Cannot write sim input file:$SimInp");
    print FOUT "IDet\tIDet\t0\t$Input$Unit\n";
    close(FOUT);
}
```

H.1.2 DC Sweep with Temperature Variation: AcqScriptTemp.pl

```
#!/usr/bin/perl
$SimInp="CktDetInput.spice";
$SimTemp="CktTemp.spice";
$DSMSimOut="DSM.lis";
$SSCSimOut="SSC.lis";
$DSMSimSrc="DSM.cir";
$SSCSimSrc="SSC.cir";
$ModelList="ModelList";
$ModelDir="CmosModel";
$DSMOutNameRoot="DSMSimOut";
$SSCOutNameRoot="SSCSimOut";
$DSMPatFile="DSM.header";
$SSCPatFile="SSC.header";
$IterNo=98;
$IterStep=10;
$IterOffset=10;
$Unit='n';
$x=0;
$Input=0;
#$ModelCount=0;
$CktTempOffset=0;
$CktTempNo=10;
$CktTempStep=10;

for ($y=0; $y<=$CktTempNo; $y++)
{
```

```
    $CktTemp=$CktTempOffset+$y*$CktTempStep;
    for ($x=0; $x<=$IterNo; $x++)
    {
        $Input=$x*$IterStep+$IterOffset;
        WriteSimInput();
        system("hspice $DSMSimSrc > $DSMSimOut");
        system("hspice $SSCSimSrc > $SSCSimOut");
        $ModelName=~s/\.spice//;
        $ModelName=~tr/a-z/A-Z/;
        $ModelIdx="T$CktTemp";
        $InputIdx="I$Input";
        $DSMCvrtStr="$DSMOutNameRoot$ModelIdx$InputIdx.out";
        $SSCCvrtStr="$SSCOutNameRoot$ModelIdx$InputIdx.out";
        system("CvrtSimOut.pl -d 2.5 -s 1 -p 0 -m $DSMPatFile < $DSMSimOut >
$DSMCvrtStr");
        system("CvrtSimOut.pl -d 2.5 -s 1 -p 0 -m $SSCPatFile < $SSCSimOut >
$SSCCvrtStr");
    }
}

sub WriteSimInput {
    open(FOUT, ">$SimInp") || die("Cannot write sim input file:$SimInp");
    print FOUT "IDet\tIDet\t0\t$Input$Unit\n";
    close(FOUT);
    open(FOUT, ">$SimTemp") || die("Cannot write sim input file:$SimInp");
    print FOUT ".temp\t$CktTemp\n";
    close(FOUT);
}
```

H.2 Spice Data Manipulation Scripts

H.2.1 SPICE Analog Data Extractor: ExtSimOut.pl

```
#!/usr/bin/perl
$Mode=0;
$Step=1;
$Phase=0;
ArgvProcess();
$Count=$Step-$Phase;
while ($Line=<STDIN>) {
    if($Mode == -1) {
        if($Line !~ /^\s*\d/)
            $Mode=0;
            close(FFILTER);
            $SimCount++;
        }
        else {
            if($Count >= $Step) {
```

```
                $Count=0;
            }
            if($Count==0) {
                $Line =~ s/k/e3/g;
                $Line =~ s/x/e6/g;
                $Line =~ s/m/e-3/g;
                $Line =~ s/u/e-6/g;
                $Line =~ s/n/e-9/g;
                $Line =~ s/p/e-12/g;
                $Line =~ s/f/e-15/g;
                print "$Line";
            }
            $Count++;
        }
    }
    if($Mode ==0 && $Line eq "x\n") {
        $Mode++;
    }
    if($Mode > 0) {
        $Mode++;
        if($Mode>4) {
            $Mode=-1;
        }
    }
}

sub ArgvProcess {
    for($k=0; $k<($#ARGV-1); $k++) {
        if($ARGV[$k] eq '-s') {$Step="$ARGV[$k+1]";}
        if($ARGV[$k] eq '-p') {$Phase="$ARGV[$k+1]";}
        if($ARGV[$k] eq '-m') {$SearchPatternFile="$ARGV[$k+1]";}
        if($ARGV[$k] eq '-h') {
            print "Hspice lis data extractor\n";
            print "-s : Step\n";
            print "-p : Phase shift in integer\n";
            print "-h : This help file\n";
        }
    }
}
```

Appendix I

LABVIEW CODES FOR DATA ACQUISITION

The data acquisition hardware for ΔΣ oversampling ADC chip was National Instrument PCI-DIO-32HS data acquisition unit [61]. To automate the data acquisition process, a set of LabView [134] codes were programmed. Figure 106 shows the designed main control panel and Figure 107 shows the program structure. The program control flow is from top to bottom. There are 32-bits inputs and outputs in the PCI-DIO-32HS and a group of 8-bits makes a port. Each port can be an output or input port, though the ports 0 and 1 have a common input or output setting and the ports 2 and 3 have a common setting. In the code, ports 0 and 1 are input ports and ports 2 and 3 are output ports. The output ports send control signal according to the array addressing scheme by selecting a group of photo detector exclusively. Input data is collected and counted to get a first-order filtered or comb filtered estimation. The processed data is stored in the designated storage device with a time stamp for the following digital signal processing.

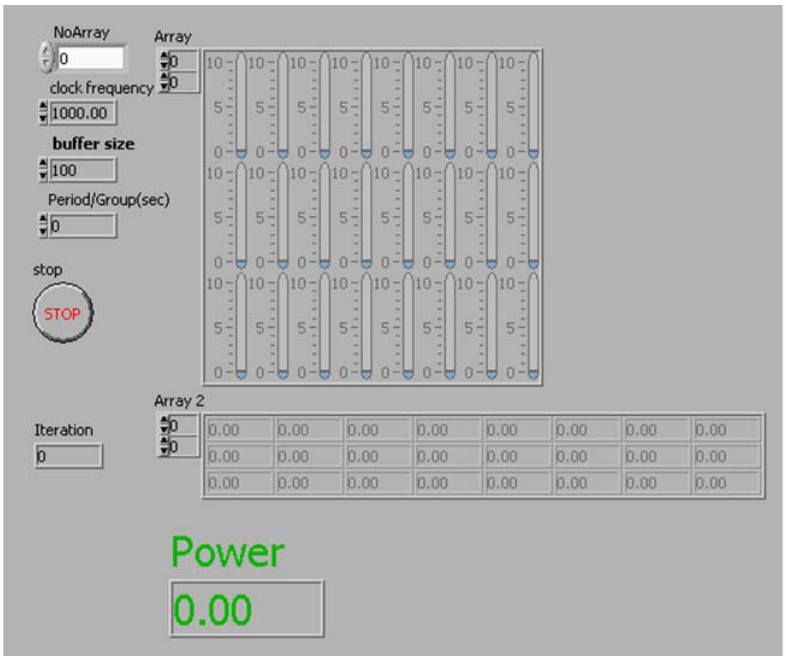

Figure 106: Main panel of data acquisition code

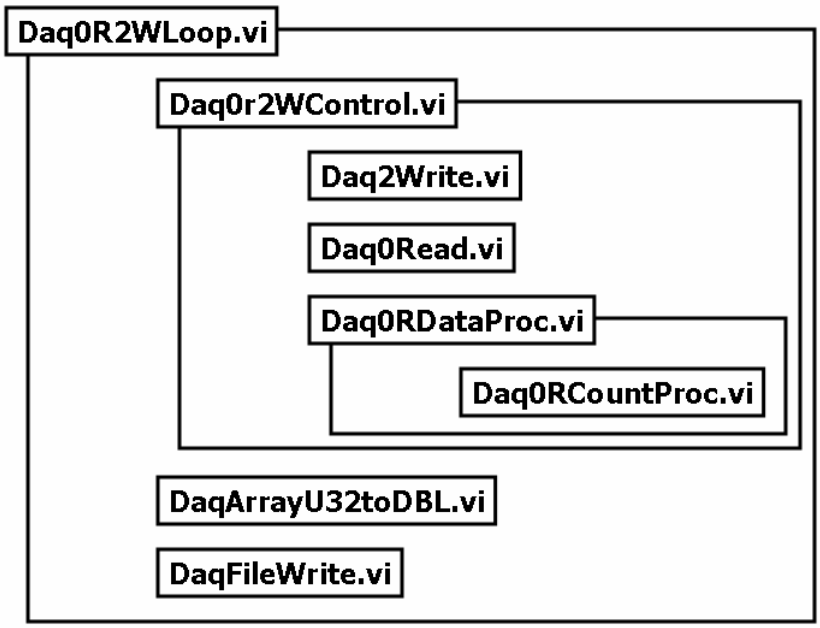

Figure 107: LabView code structure

I.1 LabView Codes

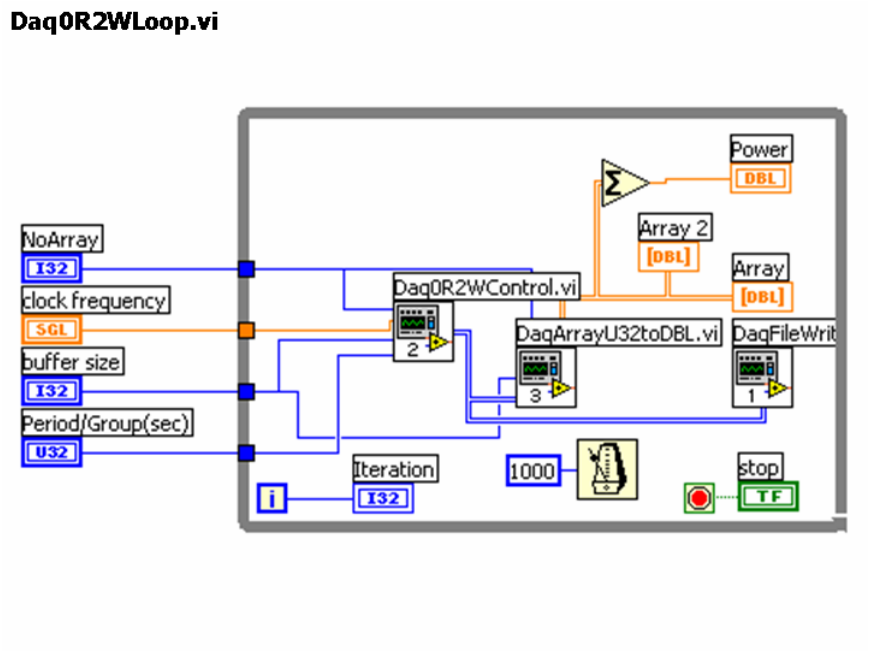

Figure 108: Daq0R2WLoop.vi

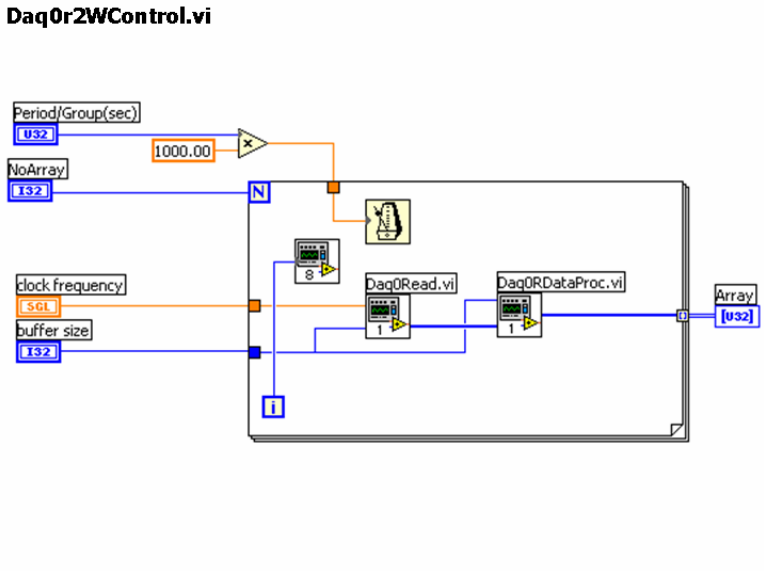

Figure 109: Daq0r2WControl.vi

Daq0RCountProc.vi

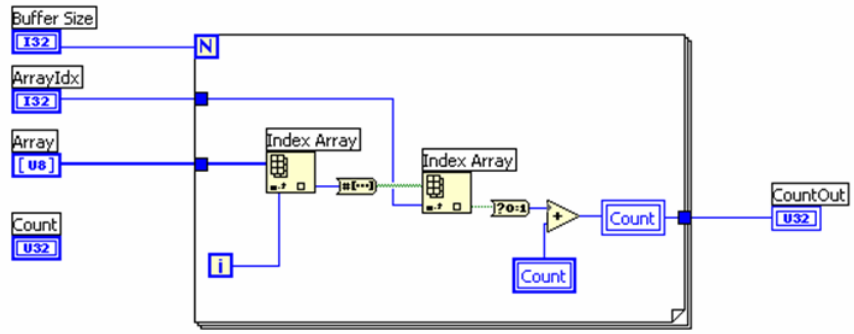

Figure 110: Daq0RCountProc.vi

Daq0RDataProc.vi

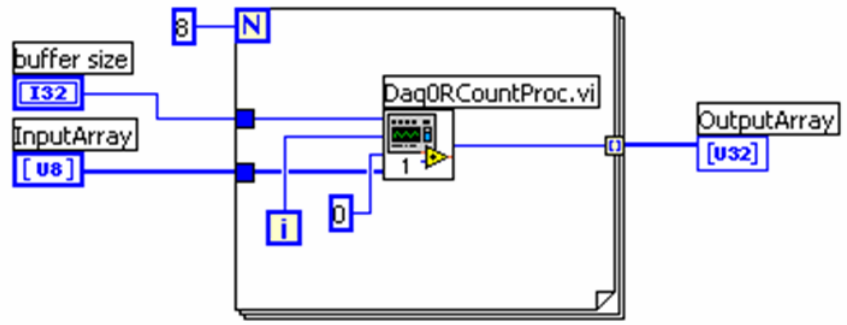

Figure 111: Daq0RDataProc.vi

Daq0Read.vi

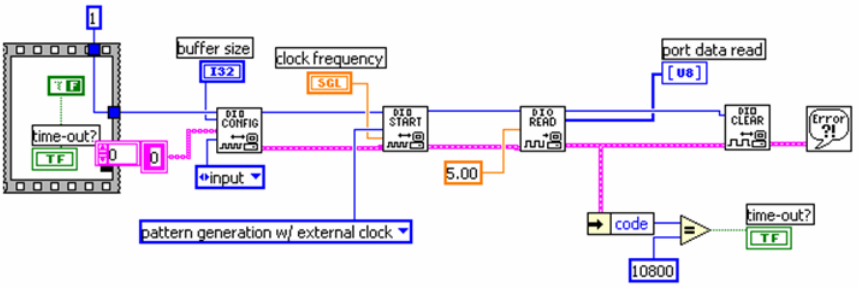

Figure 112: Daq0Read.vi

Daq2Write.vi

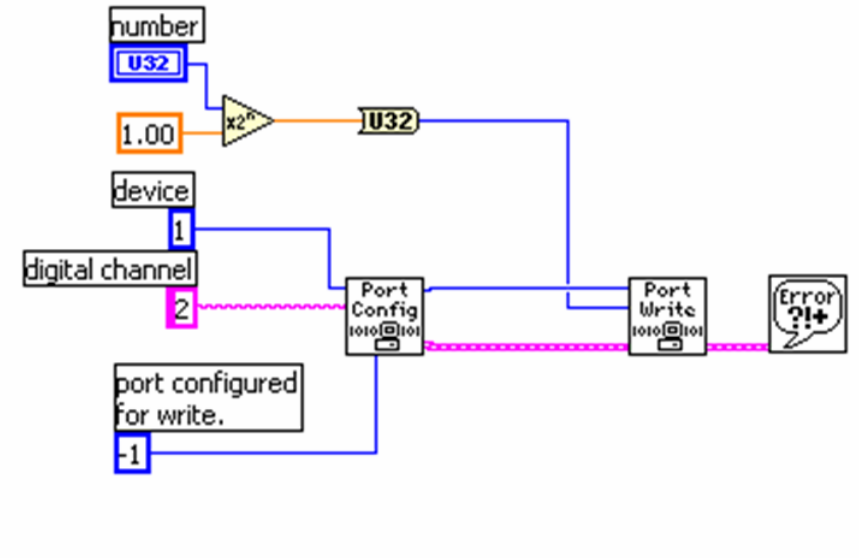

Figure 113: Daq2Write.vi

DaqArrayU32toDBL.vi

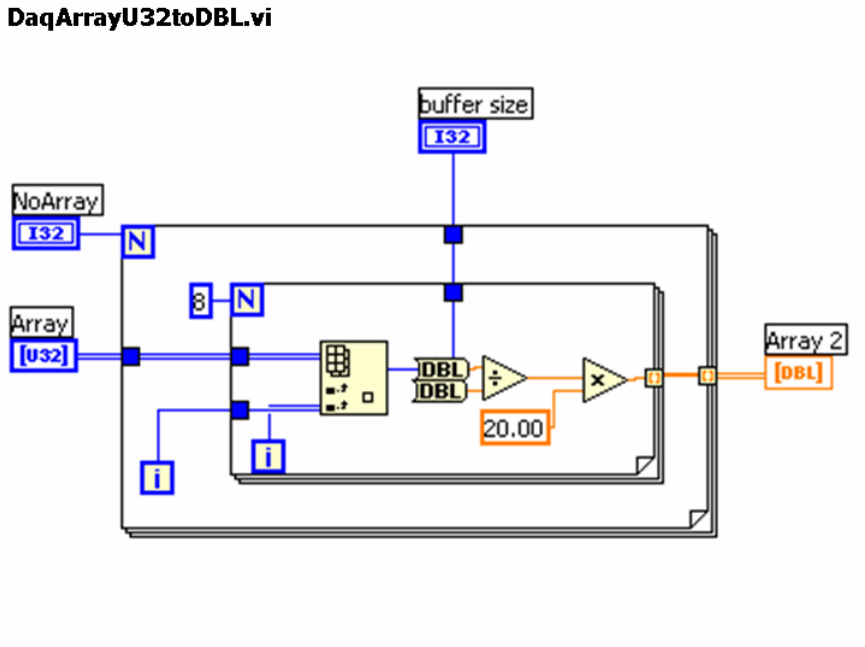

Figure 114: DaqArrayU32toDBL.vi

DaqFileWrite.vi

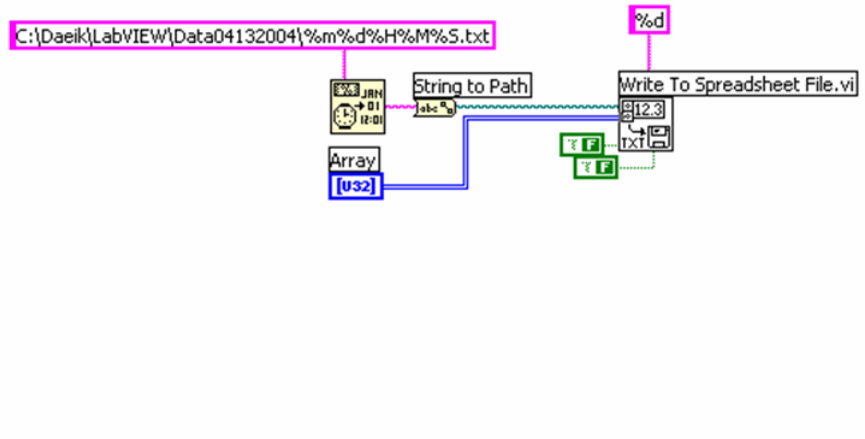

Figure 115: DaqFileWrite.vi

REFERENCES

[1] (2004) The Bio-Optoelectronic Sensor System. [Online]. Available: http://www.micro.uiuc.edu/boss/

[2] N. F. Hartman, "Optical sensing apparatus and method," U.S. Patent 4515430, July 10, 1990.

[3] D. P. Campbell, *et al.*, "Optical system-on-a-chip for chemical and biochemical sensing: the chemistry," *SPIE Proceedings*, vol. 3540, pp. 153-161, 1998.

[4] D. P. Campbell, *et al.*, "Reversible integrated optical evanescent field biosensor using chemical amplification for added sensitivity," *SPIE Proceedings*, vol. 3253, pp. 20-26, 1998.

[5] N. F. Hartman, *et al.*, "Rapid response biosensor for detection and identification of common foodborne pathogens," *SPIE Proceedings*, vol. 2345, pp. 128-137, 1994.

[6] N. F. Hartman, *et al.*, "Optical system-on-a-chip for chemical and biochemical sensing: the platform," *SPIE Proceedings*, vol. 3537, pp. 302-309, 1999.

[7] S. Cho, *et al.*, "Polymer waveguide optical interconnections on si cmos circuits," in Tech. Dig. Conference of Lasers and Electro-Optics, 2002.

[8] (2004) MOSIS Website. [Online]. Available: http://www.mosis.org/

[9] O. B. Milgrome, *et al.*, "A monolithic CMOS 16 channel, 12 bit, 10 microsecond analog to digital converter integrated circuit," *Nuclear Science, IEEE Transactions on*, vol. 40, pp. 721-723, 1993.

[10] W. R. Krenik, *et al.*, "A precision optical metering system for medical instrumentation," in Custom Integrated Circuits Conference, 1989., Proceedings of the IEEE 1989, 1989.

[11] P. E. Allen, *et al.*, *CMOS analog circuit design*, 2nd ed. New York: Oxford University Press, 2002.

[12] R. J. Baker, *et al.*, *CMOS circuit design, layout, and simulation*. New York: IEEE Press, 1998.

[13] R. J. Baker, *CMOS : mixed-signal circuit design*. New York: Wiley, 2002.

[14] M. J. Demler, *High-speed analog-to-digital conversion*. San Diego: Academic Press, 1991.

[15] R. J. v. d. Plassche, *CMOS integrated analog-to-digital and digital-to-analog converters*, 2nd ed. Boston: Kluwer Academic Publishers, 2003.

[16] M. Burns*, et al.*, *An introduction to mixed-signal IC test and measurement*. New York: Oxford University Press, 2001.

[17] D. Johns*, et al.*, *Analog integrated circuit design*. New York: John Wiley & Sons, 1997.

[18] M. P. Flynn*, et al.*, "A 400-Msample/s, 6-b CMOS folding and interpolating ADC," *Solid-State Circuits, IEEE Journal of*, vol. 33, pp. 1932-1938, 1998.

[19] S. Tsukamoto*, et al.*, "A CMOS 6-b, 400-MSample/s ADC with error correction," *Solid-State Circuits, IEEE Journal of*, vol. 33, pp. 1939-1947, 1998.

[20] M. J. M. Pelgrom*, et al.*, "A 25-Ms/s 8-bit CMOS A/D converter for embedded application," *Solid-State Circuits, IEEE Journal of*, vol. 29, pp. 879-886, 1994.

[21] (2004) Cirrus Logic A/D Converters. [Online]. Available: http://www.cirrus.com

[22] C. C. Cutler, "Transmission system employing quantization," U.S. Patent 2927962, March 8, 1960.

[23] A. Handkiewicz, *Mixed-signal systems : a guide to CMOS circuit design*. Piscataway, NJ: IEEE Press : Wiley, 2002.

[24] B. Razavi, *Design of analog CMOS integrated circuits*. Boston, MA: McGraw-Hill, 2001.

[25] M. Gustavsson*, et al.*, *CMOS data coverters for communications*. Boston: Kluwer Academic, 2000.

[26] S. R. Norsworthy*, et al.*, *Delta-sigma data converters : theory, design, and simulation*. New York: IEEE Press, 1997.

[27] B. Razavi, *Principles of data conversion system design*. New York: IEEE Press, 1995.

[28] S. Hein*, et al.*, *Sigma Delta modulators : nonlinear decoding algorithms and stability analysis*. Boston: Kluwer Academic Publishers, 1993.

[29] J. C. Candy, *et al.*, *Oversampling delta-sigma data converters : theory, design, and simulation*. Piscataway, NJ: IEEE Press, 1992.

[30] R. Gregorian, *et al.*, *Analog MOS integrated circuits for signal processing*. New York: Wiley, 1986.

[31] N. T. Thao, *et al.*, "Deterministic analysis of oversampled A/D conversion and decoding improvement based on consistent estimates," *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, vol. 42, pp. 519-531, 1994.

[32] A. R. Calderbank, *et al.*, "The pros and cons of democracy," *Information Theory, IEEE Transactions on*, vol. 48, pp. 1721-1725, 2002.

[33] (2004) IEEE Xplore. [Online]. Available: http://ieeexplore.ieee.org

[34] E. Dallago, *et al.*, "Comparison between PWM and sigma-delta modulation in a power factor correction system," in Power Electronics Specialists Conference, 2002. pesc 02. 2002 IEEE 33rd Annual, 2002.

[35] R. van de Plassche, "A sigma-delta modulator as an A/D converter," *Circuits and Systems, IEEE Transactions on*, vol. 25, pp. 510-514, 1978.

[36] J. Nieznanski, *et al.*, "Comparison of vector sigma-delta modulation and space-vector PWM," in Proc. 26th Annual Conference of the IEEE Industrial Electronics Society, 2000.

[37] C. Xu, *et al.*, "A highly integrated CMOS image sensor architecture for low voltage applications with deep submicron process," in Proc. 2002 IEEE ISCAS, 2002.

[38] L. A. Singer, *et al.*, "A 14-bit 10-MHz calibration-free CMOS pipelined A/D converter," in VLSI Circuits, 1996. Digest of Technical Papers., 1996 Symposium on, 1996.

[39] K. Nakamura, *et al.*, "An 85 mW, 10 b, 40 Msample/s CMOS parallel-pipelined ADC," *Solid-State Circuits, IEEE Journal of*, vol. 30, pp. 173-183, 1995.

[40] S. H. Lewis, *et al.*, "A pipelined 5-Msample/s 9-bit analog-to-digital converter," *Solid-State Circuits, IEEE Journal of*, vol. 22, pp. 954-961, 1987.

[41] M. Gottardi, *et al.*, "A CCD/CMOS image sensor array with integrated A/D conversion," in Proc. 1997 IEEE ISCAS, 1997.

[42] W. Chen, *et al.*, "Integrated 1.2 μm CMOS photodiodes, transimpedance amplifier, 12 bits A/D converter, and DSP interface for microinstrument applications," in

Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on, 1999.

[43] D. G. Gata*, et al.*, "A 1.1-V 270-μA mixed-signal hearing aid chip," *Solid-State Circuits, IEEE Journal of*, vol. 37, pp. 1670-1678, 2002.

[44] C. B. Wang, "A 20-bit 25-kHz delta-sigma A/D converter utilizing a frequency-shaped chopper stabilization scheme," *Solid-State Circuits, IEEE Journal of*, vol. 36, pp. 566-569, 2001.

[45] P. C. Maulik*, et al.*, "A 16-bit 250-kHz delta-sigma modulator and decimation filter," *Solid-State Circuits, IEEE Journal of*, vol. 35, pp. 458-467, 2000.

[46] W. H. Press, *Numerical recipes in C : the art of scientific computing*, 2nd ed. Cambridge [Cambridgeshire] ; New York: Cambridge University Press, 2002.

[47] B. Razavi, *Design of integrated circuits for optical communications*. Boston: McGraw-Hill, 2003.

[48] L. G. McIlrath, "A robust O(N log n) algorithm for optimal decoding of first-order ΣΔ sequences," *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, vol. 50, pp. 1942-1950, 2002.

[49] S. Hein*, et al.*, "Optimal decoding for data acquisition applications of sigma delta modulators," *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, vol. 41, pp. 602-616, 1993.

[50] R. M. Gray, "Spectral analysis of quantization noise in a single-loop sigma-delta modulator with DC input," *Communications, IEEE Transactions on*, vol. 37, pp. 588-599, 1989.

[51] J. Candy, "Decimation for Sigma Delta Modulation," *Communications, IEEE Transactions on [legacy, pre - 1988]*, vol. 34, pp. 72-76, 1986.

[52] S. Chu*, et al.*, "Multirate filter designs using comb filters," *Circuits and Systems, IEEE Transactions on*, vol. 31, pp. 913-924, 1984.

[53] F. Dachselt*, et al.*, "Rational cycle decoding algorithm for the first-order delta-sigma modulator," in Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on, 2001.

[54] M. R. Sherkat*, et al.*, "A novel decoder for ΣΔ modulator providing both high resolution and low latency," in Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on, 1999.

[55] R. W. Sandage, *et al.*, "Producing phototransistors in a standard digital CMOS technology," in Circuits and Systems, 1996. ISCAS '96., 'Connecting the World'., 1996 IEEE International Symposium on, 1996.

[56] (2004) HSPICE. [Online]. Available: http://www.synopsys.com/

[57] M. Ortmanns, *et al.*, "Fundamental limits of jitter insensitivity in discrete and continuous-time sigma delta modulators," in Proc. 2003 IEEE ISCAS, 2003.

[58] O. Oliaei, *et al.*, "Jitter effects in continuous-time ΣΔ modulators with delayed return-to-zero feedback," in IEEE 1998 International Conference on Electronics, Circuits and Systems, 1998.

[59] J. A. Cherry, *et al.*, "Clock jitter and quantizer metastability in continuous-time delta-sigma modulators," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on]*, vol. 46, pp. 661-676, 1999.

[60] (2004) Tektronix. [Online]. Available: http://www.tektronix.com

[61] (2004) National Instruments. [Online]. Available: http://www.ni.com/

[62] (2004) Keithley. [Online]. Available: http://www.keithley.com/

[63] J. Nakamura, *et al.*, "On-focal-plane signal processing for current-mode active pixel sensors," *Electron Devices, IEEE Transactions on*, vol. 44, pp. 1747-1758, 1997.

[64] (2004) Newport. [Online]. Available: http://www.newport.com/

[65] C. D. Motchenbacher, *et al.*, *Low-noise electronic system design*. New York: J. Wiley & Sons, 1993.

[66] A. A. Dorrington, *et al.*, "A simple microcontroller based digital lock-in amplifier for the detection of low level optical signals," in IEEE International Workshop on Electronic Design, Test and Applications, 2002.

[67] A. Mandelis, "Signal-to-noise ratio in lock-in amplifier synchronous detection: A generalized communications system approach with applications to frequency, time, and hybrid (rate window) photothermal measurements," *Rev. Sci. Instrum.*, vol. 65, pp. 3309-3323, 1994.

[68] W. B. Kuhn, *et al.*, "Dynamic range performance of on-chip RF bandpass filters," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on]*, vol. 50, pp. 685-694, 2003.

[69] A. I. Hussein, *et al.*, "Bandpass ΣΔ modulator employing undersampling of RF signals for wireless communication," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on]*, vol. 47, pp. 614-620, 2000.

[70] C. H. Leong, *et al.*, "An effective implementation of high-order bandpass sigma-delta modulators for high speed D/A applications," in Proc. 1997 IEEE ISCAS, 1997.

[71] I. J. O'Connell, *et al.*, "A high pass switched capacitor ΣΔ modulator," in 9th IEEE International Conference on Electronics, Circuits and Systems, 2002.

[72] A. Tabatabaei, *et al.*, "A wideband bandpass sigma-delta modulator for wireless applications," in Dig. Tech. Papers 1999 Symposium on VLSI Circuits, 1999.

[73] P. Ju, *et al.*, "A 22-kHz multibit switched-capacitor sigma-delta D/A converter with 92 dB dynamic range," *Solid-State Circuits, IEEE Journal of*, vol. 30, pp. 1316-1325, 1995.

[74] I. Galton, "Delta-sigma data conversion in wireless transceivers," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 50, pp. 302-315, 2002.

[75] E. Owen, "The elimination of offset errors in dual-slope analog-to-digital converters," *Circuits and Systems, IEEE Transactions on*, vol. 27, pp. 137-141, 1980.

[76] I. Wold, "Offset error compensation for integrating analog-to-digital converter," U.S. Patent 3942173, March 2, 1976.

[77] Y. Manoli, "A self-calibration method for fast high-resolution A/D and D/A converters," *Solid-State Circuits, IEEE Journal of*, vol. 24, pp. 603-608, 1989.

[78] E. Raisanen-Ruotsalainen, *et al.*, "An integrated time-to-digital converter with 30-ps single-shot precision," *Solid-State Circuits, IEEE Journal of*, vol. 35, pp. 1507-1510, 2000.

[79] H. Schmid, *Electronic analog/digital conversions*. New York,: Van Nostrand Reinhold Co., 1970.

[80] T. Hornak, *et al.*, "A high precision component-tolerant A/D convertor," *Solid-State Circuits, IEEE Journal of*, vol. 10, pp. 386-391, 1975.

[81] R. H. McCharles, *et al.*, "An algorithmic analog-to-digital converter," in 1977 IEEE ISSCC Dig. Tech. Papers, 1977.

[82] R. Webb, *et al.*, "A 12b A/D converter," in Solid-State Circuits Conference. Digest of Technical Papers. 1981 IEEE International, 1981.

[83] P. W. Li, *et al.*, "A ratio-independent algorithmic analog-to-digital conversion technique," *Solid-State Circuits, IEEE Journal of*, vol. 19, pp. 828-836, 1984.

[84] H.-S. Lee, *et al.*, "A self-calibrating 15 bit CMOS A/D converter," *Solid-State Circuits, IEEE Journal of*, vol. 19, pp. 813-819, 1984.

[85] R. V. d. Plaasche, "Dynamic element matching for high accuracy D/A converters," in 1976 IEEE ISSCC Dig. Tech. Papers, 1976.

[86] G. R. Ritchie, "Higher order interpolation analog to digital converters," University of Pennsylvania, 1977.

[87] H. A. Spang, *et al.*, "Reduction of quantizing noise by use of feedback," *IRE Trans. on Commun. Syst.*, pp. 373-380, 1962.

[88] F. d. Jager, "Delta modulation - a method of PCM transmission using the one unit code," *Philips Res. Rep.*, vol. 7, pp. 442-466, 1952.

[89] H. Inose, *et al.*, "A telemetering system by code modulation - ΔΣ modulation," *IRE Trans. Space Electron. Telemetry*, vol. SET-8, pp. 204-209, 1962.

[90] S. S. Haykin, *Communication systems*, 4th ed. New York: Wiley, 2001.

[91] J. Candy, *et al.*, "The Structure of Quantization Noise from Sigma-Delta Modulation," *Communications, IEEE Transactions on [legacy, pre - 1988]*, vol. 29, pp. 1316-1323, 1981.

[92] R. Gray, "Oversampled Sigma-Delta Modulation," *Communications, IEEE Transactions on [legacy, pre - 1988]*, vol. 35, pp. 481-489, 1987.

[93] R. M. Gray, "Quantization noise spectra," *Information Theory, IEEE Transactions on*, vol. 36, pp. 1220-1244, 1990.

[94] A. V. Oppenheim, *et al.*, *Discrete-time signal processing*. Englewood Cliffs, N.J.: Prentice Hall, 1989.

[95] A. V. Oppenheim, *et al.*, *Signals & systems*, 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 1997.

[96] L. Longo, *et al.*, "A 13 bit ISDN-band Oversampled ADC using Two-Stage Third Order Noise Shaping," in Proc. IEEE 1988 Conference on Custom Integrated Circuits, 1988.

[97] K. Matsumoto, *et al.*, "An 18b Oversampling A/D Converter for Digital Audio," in Solid-State Circuits Conference, 1988. Digest of Technical Papers. ISSCC. 1988 IEEE International, 1988.

[98] Y. Matsuya, *et al.*, "A 16-bit oversampling A-to-D conversion technology using triple-integration noise shaping," *Solid-State Circuits, IEEE Journal of*, vol. 22, pp. 921-929, 1987.

[99] S. R. Norsworthy, *et al.*, "A 14-bit 80-kHz sigma-delta A/D converter: modeling, design and performance evaluation," *Solid-State Circuits, IEEE Journal of*, vol. 24, pp. 256-266, 1989.

[100] Y. Geerts, *et al.*, "A high-performance multibit ΔΣ CMOS ADC," *Solid-State Circuits, IEEE Journal of*, vol. 35, pp. 1829-1840, 2000.

[101] M. R. Miller, *et al.*, "A multibit sigma-delta ADC for multimode receivers," *Solid-State Circuits, IEEE Journal of*, vol. 38, pp. 475-482, 2003.

[102] Y. Joo, *et al.*, "Smart CMOS focal plane arrays: a Si CMOS detector array and sigma-delta analog-to-digital converter imaging system," *Selected Topics in Quantum Electronics, IEEE Journal of*, vol. 5, pp. 296-305, 1999.

[103] Y. Joo, *et al.*, "Compact current input oversampling modulator design for a scalable high frame rate focal plane arrays," in Proc. 2000 IEEE ISCAS, 2000.

[104] Y. Botteron, *et al.*, "An investigation of bandpass sigma-delta A/D converters," in Proc. 40th IEEE Midwest Symposium on Circuits and Systems, 1997.

[105] T. Salo, *et al.*, "A 80-MHz bandpass ΔΣ modulator for a 100-MHz IF receiver," *Solid-State Circuits, IEEE Journal of*, vol. 37, pp. 798-808, 2002.

[106] T. Ueno, *et al.*, "A fourth-order bandpass ΔΣ modulator using second-order bandpass noise-shaping dynamic element matching," *Solid-State Circuits, IEEE Journal of*, vol. 37, pp. 809-816, 2002.

[107] V. T. Nguyen, *et al.*, "Advantages of high-pass ΔΣ modulators in interleaved ΔΣ analog to digital converter," in Proc. 45th IEEE Midwest Symposium on Circuits and Systems, 2002.

[108] A. K. Ong, *et al.*, "A two-path bandpass ΣΔ modulator for digital IF extraction at 20 MHz," *Solid-State Circuits, IEEE Journal of*, vol. 32, pp. 1920-1934, 1997.

[109] Y.-H. Chang, *et al.*, "Chopper-stabilized sigma-delta modulator," in Proc. 1993 IEEE ISCAS, 1993.

[110] S. Hein, *et al.*, "On the stability of sigma delta modulators," *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, vol. 41, pp. 2322-2348, 1993.

[111] J. Candy, "A Use of Double Integration in Sigma Delta Modulation," *Communications, IEEE Transactions on [legacy, pre - 1988]*, vol. 33, pp. 249-258, 1985.

[112] K. C.-H. Chao*, et al.*, "A higher order topology for interpolative modulators for oversampling A/D converters," *Circuits and Systems, IEEE Transactions on*, vol. 37, pp. 309-318, 1990.

[113] S. Ardalan*, et al.*, "An analysis of nonlinear behavior in delta-sigma modulators," *Circuits and Systems, IEEE Transactions on*, vol. 34, pp. 593-603, 1987.

[114] T. Ritoniemi*, et al.*, "Design of stable high order 1-bit sigma-delta modulators," in Circuits and Systems, 1990., IEEE International Symposium on, 1990.

[115] W. Chou*, et al.*, "Multistage sigma-delta modulation," *Information Theory, IEEE Transactions on*, vol. 35, pp. 784-796, 1989.

[116] A. Hirota*, et al.*, "A novel delta-sigma modulated DC-DC power converter operating under DC ripple voltage," in Proc. 25th Annual Conference of the IEEE Industrial Electronics Society, 1999.

[117] A. Eshraghi*, et al.*, "A time-interleaved parallel ΔΣ A/D converter," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on]*, vol. 50, pp. 118-129, 2003.

[118] D. P. Scholnik*, et al.*, "Space-time vector delta-sigma modulation," in Proc. 2002 IEEE ISCAS, 2002.

[119] R. Khoini-Poorfard*, et al.*, "Time-interleaved oversampling A/D converters: theory and practice," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on]*, vol. 44, pp. 634-645, 1997.

[120] I. Galton*, et al.*, "Oversampling parallel delta-sigma modulator A/D conversion," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on]*, vol. 43, pp. 801-810, 1996.

[121] E. Roza, "Poly-phase sigma-delta modulation," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on]*, vol. 44, pp. 915-923, 1997.

[122] E. Dijkstra*, et al.*, "On the use of modulo arithmetic comb filters in sigma delta modulators," in Proc. 1988 IEEE ICASSP, 1988.

[123] L. Luh, *et al.*, "A High-Speed Digital Comb Filter for Sigma-Delta Analog-to-Digital Conversion," in Proc. 42nd IEEE Midwest Symposium on Circuits and Systems, 2000.

[124] D. D. Kim, *et al.*, "A 1.4G Samples/sec Comb Filter Design for Decimation of Sigma-Delta Modulator Output," in Proc. 2003 IEEE ISCAS, 2003.

[125] P. Steiner, *et al.*, "A framework for analysis of high-order sigma-delta modulators," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on]*, vol. 44, pp. 1-10, 1997.

[126] H. Wang, "A geometric view of ΣΔ modulations," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on]*, vol. 39, pp. 402-405, 1992.

[127] A. K. Gupta, *et al.*, "Viterbi decoding and ΣΔ modulation," in Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on, 2002.

[128] E. Kreyszig, *Advanced engineering mathematics*, 7th ed. New York: Wiley, 1993.

[129] R. B. Ash, *Information theory*. New York,: Interscience Publishers, 1965.

[130] J. R. Pierce, *An introduction to information theory : symbols, signals & noise*, 2nd, rev. ed. New York: Dover Publications, 1980.

[131] F. M. Reza, *An introduction to information theory*. New York: Dover, 1994.

[132] C. E. Shannon, "A mathematical theory of communcication," *Bell System Tech. J.*, vol. 27, pp. 379-324, 623-656, 1948.

[133] (2004) MOSIS AMI Si-CMOS Models. [Online]. Available: http://www.mosis.org/Technical/Testdata/ami-abn-prm.html

[134] (2004) LabView, National Instruments. [Online]. Available: http://www.ni.com/labview/

VITA

Daeik D. Kim was born in Seoul, Republic of Korea in 1973. Having completed Paichai middle and high schools in 1989 and 1992 respectively, he was admitted to the School of Electrical Engineering, Seoul National University.

During his collegiate study, he was involved in an evangelical movement as a leader of Campus Crusade for Christ. He volunteered for military duty in 1995 and was deployed to Joint Security Area under the United Nations Command Security Battalion, which is located at the demilitarized zone between Republic of Korea and North Korea, until he was dismissed as a sergeant in 1997. After he received BS degree at Seoul National University in 1999, he was engaged with EduMTek Co., MGB Endoscopy Co. Ltd., and Bitnuri Co. as a freelancer engineer.

He began his graduate study with major advisor Professor Martin A. Brooke at the School of Electrical and Computer Engineering, Georgia Institute of Technology in 2000, and received MS and PhD degree in 2002 and 2004 with electronic design and application major and computer science minor. During graduate study, he was selected for Information and Telecommunication National Scholarship Program, provided by Ministry of Information and Communication, Republic of Korea. After completing his graduate studies, he began his career as a research engineer at the Department of Electrical and Computer Engineering, Duke University in 2004.

His research interest includes mixed-signal processing system design, and system-on-a-chip integration of heterogeneous function blocks for sensor and communication applications.