UNIVERSITY OF CALIFORNIA
Santa Barbara

Sigma Delta Stream Computation: A New Paradigm for Low Power and High Resolution Feedback Control

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

by

Joseph Sam Poverelli

Committee in Charge:

Professor Forrest Brewer, Chair

Professor Joao Hespanha

Professor Katie Byl

Professor Li-C. Wang

March 2020

The Dissertation of
Joseph Sam Poverelli is approved.

Professor Joao Hespanha

Professor Katie Byl

Professor Li-C. Wang

Professor Forrest Brewer, Committee Chairperson

December 2019

Sigma Delta Stream Computation: A New Paradigm for Low Power and High Resolution Feedback Control

by

Joseph Sam Poverelli

This dissertation is dedicated to my children. May they surpass me in all ways.

Acknowledgments

Without the help of many people, this work would never have been completed. I first would like to express my deepest gratitude to my adviser and mentor Forrest Brewer. Forrest always believed in me even when I doubted myself and I will always have great respect for him. I would like to thank Li-C Wang, Katie Byl, and Joao Hespanha for serving on my committee, imparting knowledge, and having patience with me. To my wife Stephanie for taking good care of our children and offering constant encouragement. To my father and late mother who always have been supportive in everything I do. I would also like to thank my fellow lab mates Merritt, Kunal, Carrie, Prashansa, David, and Aditya for all of their help, encouragement, and banter. I also have many thanks to give to Val de Veyra for all her help on the administrative side of things. My friends Kyle, Wade, Ben, and others are also of special note as they provided me with comradery and emotional support. Finally, I owe thanks to Fifth Gait Technologies and Wyatt Technologies for keeping me employed and fed during this time. Thank you all so much.

Curriculum Vitæ

Joseph Sam Poverelli

Education

2019	Ph.D. in Electrical and Computer Engineering (Expected), University of California, Santa Barbara.
2012	M.S. in Electrical and Computer Engineering, University of California, Santa Barbara.
2009	B.S. in Electrical Engineering, University of California, Santa Barbara.

Publications

Arya, K.; Poverelli, J.; Brewer, F., "Ongoing challenges in automated cyberphysical cross-domain design," *International Conference on Computing, Networking and Communications (ICNC), 2013.*, pp. 341-347, Jan 2013
doi: 10.1109/ICCNC.2013.6504106

Poverelli, J.; Brewer, F., "Direct ΣΔ Bitstream Processing for High Performance Feedback Control," *IEEE Conference on Control Technology and Applications (CCTA), 2019.*, pp. 444-449, Aug 2019
doi: 10.1109/CCTA.2019.8920720

Abstract

Sigma Delta Stream Computation: A New Paradigm for Low Power and High Resolution Feedback Control

by

Joseph Sam Poverelli

This dissertation describes a design and analysis methodology for $\Sigma\Delta$ bitstream filters and controllers in digital hardware. These circuits emulate continuous linear time invariant (LTI) models and directly process $\Sigma\Delta$ encoded bitstreams produced by $\Sigma\Delta$-based data converters. Since these converters are oversampled, there is a natural opportunity for clocking at the oversampling rate allowing for multiplierless, low latency designs. Direct processing of bitstreams also eliminates lowpass filtering and decimation necessary for conventional bit-parallel DSP. Combined, these changes reduce the hardware resources by more than an order of magnitude in FPGA implementations, with similar improvements in power overhead. MASH techniques (used extensively in data converters) are developed to allow for substantive improvement in resolution or reduction of the oversampling rate. These results have very substantive implications in the design of low-complexity, high performance controllers. In particular, these techniques can obviate conventional DSP augmented designs allowing for robust control in applications unreachable with current technology.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

Digital embedded controllers have become indispensable in modern life. Practically every modern convenience makes use of embedded digital control in order to improve performance, cost, and reliability. Lithium Ion batteries, for instance, contain an embedded charge controller running on a micro-processor to manage complex charging profiles that improve capacity and longevity as well as preventing cell detonation. The advent of digital technology has made embedded processors extremely inexpensive with the consequence that the vast majority of embedded controllers are implemented in software. Despite the effectiveness of micro-controllers in implementing complex control inexpensively, there are performance limitations that impair the ability to be a viable implementation solution for many applications requiring low latency and/or high bandwidth.

Software based control on a micro-controller requires several thousands of processor instructions to service interrupts and perform loop iterations. At practical clock rates, the latency at which a micro-controller can sample data, process a control algorithm, and update the control output is on the scale of tens of microseconds at best. For applications that require bandwidths above tens of kilohertz, digital signal processors (DSPs) are required. Having hardware architectures that are designed for high throughput and high bandwidth computation, DSPs are able

to reduce the latency of digital control implementations to hundreds of nanoseconds. The price for the reduced latency, however, is a substantial increase is component cost, as well as drastic increase in power consumption to the point where mobile (battery based) implementations are often impossible.

For control applications requiring bandwidths of a megahertz or more, the current options are logic-based hardware control using field programmable gate arrays (FPGAs) or continuous control implemented with analog components. Applications such as micro-electro-mechanical-systems (MEMS) and atomic force microscopy require very high bandwidth controllers, due to the small device scale. [59] FPGA programmable logic can accommodate complex controller designs that can run at hundreds of megahertz with the downside of being high cost and requiring high power consumption. Analog components on the other hand, can achieve control bandwidths of greater than hundreds of megahertz but have a litany of issues. These issues include controller hardware inflexibility, high cost, low reliability, high noise, and high sensitivity to parameter variations and drift which contribute to the complexity of such designs [7]. All of these issues contribute to the high cost of constructing controllers for low-latency systems. This is particularly onerous for MEMS devices where the base cost could be quite low due to economies of manufacturing scale.

This thesis presents an alternative strategy for implementing low complexity, low power, low latency, continuous time performance controllers in digital logic. The control implementation strategy does this by directly processing high resolution, high bandwidth $\Sigma\Delta$ encoded bitstreams. Both inputs and outputs can be encoded as sigma-delta bitstreams leading to inexpensive composability and a substantial reduction of communication costs. Additionally, the potential for stable operation in adverse environments is enhanced as this representation is not place sensitive (bit-weighted) and is inherently incremental. Thus, such controllers are candidates for satellite and high reliability applications as well as extreme low power (IOT) scenarios. $\Sigma\Delta$ converters are the premier data converter for high resolution, medium bandwidth applications. They are inexpensive, exceptional linear, and produce a compact oversampled bitstream representation as their natural outputs. This work allows direct connection between $\Sigma\Delta$ con-

verters and the controller hardware without need to perform parallel data conversion and thus eliminates the associated overhead and latency.

These ideas can be made more concrete by by means of controller block signal flow diagram. The signal chain samples an underlying continuous phenomena and converts it to a discrete signal representation, processes the sampled data via the discrete control law, and then converts the output sampled data back into an analog actuation signal. Such a signal chain is depicted in Figure 1.1.

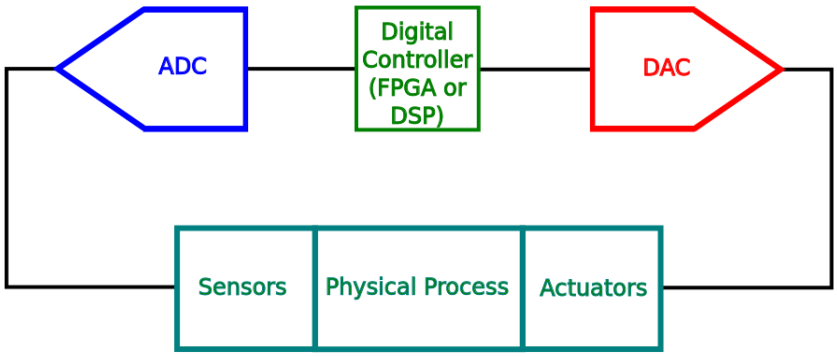


Figure 1.1: Conventional Embedded $\Sigma\Delta$ Control Loop

As shown above, the signal chain is comprised of the following components:

- Physical Process: This is the physical process that is intended to be controlled.
- Sensors/Transducers: Sensors and transducers are the components that convert a physical phenomenon (e.g. sound waves, pressure, displacement, etc.) to an electrical signal such as Voltage. These devices can range from microphones to accelerometers.
- Analog to Digital Converter (ADC): The ADC converts as electrical signal, typically a Voltage, into a series of time sampled discrete binary values.
- FPGA, DSP, μController: The control law in implemented discretely in a processing element such as a digital signal processor or μcontroller or in discrete hardware inside of an

FPGA. For the DSP and μcontroller, the control law is written in a low level programming language such as C in order more closely manage the binary mathematics that underlie the control law.

- Digital to Analog Converter (DAC): The DAC converts a discrete binary value into an electrical signal such as a Voltage.

- Actuator: The actuator is the component that physically interacts with the outside world. These could be motors, voice coils, electromagnets, etc. and are driven by the output of the DAC.

For a general embedded control system, each component is indispensable in the implementation of the control algorithm. However, proper care must be taken by the design engineer in choosing components that will meet or exceed the required closed loop performance constraints. For example, two of the most important considerations when selecting components are that of sampling frequency and resolution. ADCs, for instance, come in several varieties that have trade-offs in sampling frequency/bandwidth and resolution. For loops that require both high resolution and high bandwidth, finding an appropriate ADC may be expensive or outright impossible given the system constraints. In these cases analog controllers are used despite their issues regarding additional noise, environmental degradation, and hardware non-flexibility.

Currently, embedded digital control systems suffer pervasive limitations due to the digital algorithm implementation structured around processing 2's compliment signal representations at rates not far from Nyquist bounds. The reason for doing so is the intuitive simplicity of the time-shift/Z-transform model and is reinforced by the control design software (both system level and implementation) making this assumption. The ubiquity of these models and tools builds expectations of latency, complexity, and power consumption into the design methodology. Indeed, the view of many control engineers is that increasing the sampling bandwidth substantially beyond Nyquist adds complexity and expense as well as decreasing the stability of the implementation. This view is supported by classical analysis of coefficient resolution versus sampling rate, again given the time-shift implementation paradigm. In this thesis, simple controllers are implemented that exploit the inherent oversampling of the input ADC. Based on

digital integrators instead of time-shift operators, the new designs offer latencies on the scale of the oversampling clock period while demonstrating enhanced stability and relaxation of coefficient resolution. (As an example, a high-order bandbass filter can be directly implemented as a single filter element in contrast to composed bi-quad elements made to accommodate coefficient sensitivity at high sample rates). A side benefit of this methodology allows the removal of all hardware multipliers from the design allowing for substantive hardware power and complexity savings, without any latency cost.

1.1.1 Limitations in Conventional Discrete Control Algorithms

While DSPs and FPGAs have made digital control extremely viable in regards to the synthesis of complex control algorithms, there are a number of challenges that limit practical performance. Perhaps the most detrimental aspect of conventional controller designs is the of discrete shift operator constructions. This model is one where a the next controller state x depends on a function f of current values of x and input u at time index k which can be written as

$$x\left(k+1\right)=f\left(x\left(k\right),u\left(k\right)\right)$$

For the linear system variety, the Z-transform is the primary tool for analysis and frequency domain design. The shift operator and Z-transform constitute the predominant paradigm in discrete time controller design, analysis, and implementation. Although it may seem shocking, shift operator based controllers are actually poorly suited to implement controllers that require high performance in terms of simultaneous latency and resolution.

Consider the continuous time state space controller model

$$\dot{x}=Ax+Bu$$
$$y=Cx+Du$$

where $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, $u \in \mathbb{R}^k$, $A \in \mathbb{R}^{n \times n}$,$B \in \mathbb{R}^{n \times k}$,$C \in \mathbb{R}^{m \times n}$,$D \in \mathbb{R}^{m \times k}$, and $n, m, k \in \mathbb{Z}^+$. Applying a zero-order hold discretization method with a defined sampling period Δ, the discrete shift operator based equivalent model becomes

$$x^+ = A_z x + B_z u$$
$$y = C_z x + D_z u$$

where

$$\begin{aligned} A_z &= e^{A\Delta} \\ B_z &= \int_0^{\Delta} e^{A(t-\tau)} B d\tau \\ C_z &= C \\ D_z &= D \end{aligned}$$

Now suppose that one were to reduce Δ to a small value analogous to significantly increasing the sampling frequency. As the sampling period approaches zero the state transition matrix A_z becomes

$$\lim_{\Delta \to 0} A_z = \lim_{\Delta \to 0} e^{A\Delta} = I$$

The implication here is that for fast sampling controllers, the state transition matrix A_z approaches the identity matrix and imposes the issue of the system poles (i.e. the eigenvalues of A_z) gravitating to $z = 1$ on the real axis in the complex plane. System poles approaching a limit point on the stability boundary makes the overall control algorithm extremely sensitive to small perturbations in the controller coefficients. Stated another way, the controller requires extremely accurate coefficients to differentiate between the separate pole values. Unfortunately, embedded control systems, whether they are implemented on a DSP, FPGA, or other device, operate on quantized signal and coefficient values. The allowable number of bits to represent a number in real discrete controller implementations is limited, which in turn, limits the achievable sampling rate.

The sensitivity problem associated with shift-based control algorithms leads to controllers that are relatively complex requiring larger, more powerful, and more expensive components to implement. Perhaps the most tragic pitfall of the shift-based controller design is that it disincentives control designers from creating digital controllers that operate at higher bandwidths. Abstractly, controlling any system with lower latency should result in higher performance control. This valid intuition is not supported by the inherent singularity of the Z-transform as bandwidths increase. These issues are in the mathematical model, not in the physical reality. In the case of MEMs devices, where system time scales are significantly smaller due to the down scaling of physical size, high bandwidth is a must. This points to constructing a mathematical model efficiently supporting high bandwidth operation.

1.1.2 High Resolution Data Conversion

The sampling frequency of the digital controller will largely be determined by the physical time constants of the physical process itself. Typically, the sample frequency is chosen to be an order of magnitude higher than the highest frequency component of the physical process. After a sampling frequency has been chosen, it is also necessary to choose the resolution of the overall control system. Many factors can determine the required resolution such as system steady-state error bounds or the resolution the front end sensors/transducers. Once the the resolution has been found the first component of the embedded signal chain can be chosen, the analog to digital converter.

Analog to digital converters come in a variety of architectures each of which has its own trade offs. The most popular ADCs are the Dual-Slope ADC, the Successive Approximation/Pipeline ADC, the $\Sigma\Delta$ ADC, and the Flash ADC. When it comes the performance metrics of throughput (i.e. sample per second) and resolution, each ADC type occupies a different area in that two dimensional performance space. As can be seen in Figure 1.2, the various ADC types have different trade off in resolution and speed.

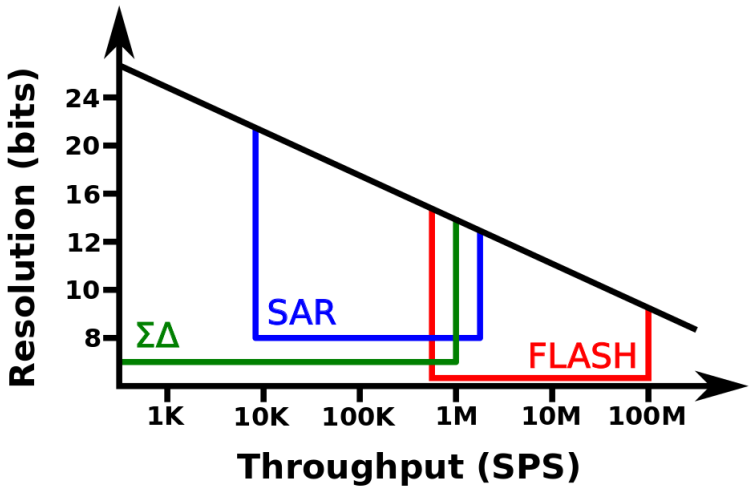


Figure 1.2: ADC Architecture Resolution vs. Throughput

In this work, we focus on ΣΔ ADC which is an oversampled converter whose output is a high speed stream of 1s and 0s. For high resolution and medium throughput requirements, ΣΔ data conversion is the premier encoding strategy for high performance applications [1]. As such, it find ubiquitous employment in many devices who directly supply the bitstream encoding (so-called PDM or pulse-density-modulation in this use). Everything from ADCs to MEMs microphones to class D amplifier stages currently exist on the market which directly produce or make use of ΣΔ encoded bitstream signals [10, 11, 6]. However, most designers treat bitstreams as a liability or burden which is demonstrated through how bitstreams are typically converted immediately to a sampled parallel representation.

Typical ΣΔ ADCs have an architecture shown in 1.3. On the front end is the ΣΔ converter itself which converters the analog signal input to that of an oversampled bitstream of ones and zeros. The bitstream is then lowpass filtered (usually 3rd-order integration) and then decimated to close to the Nyquist Bandwidth in order to convert the bitstream into a bit-parallel number representation. The bit-parallel number is then read out through an interface port, such as a serial peripheral interface (SPI) port, to a processing unit, such as a μcontroller, DSP, or FPGA.

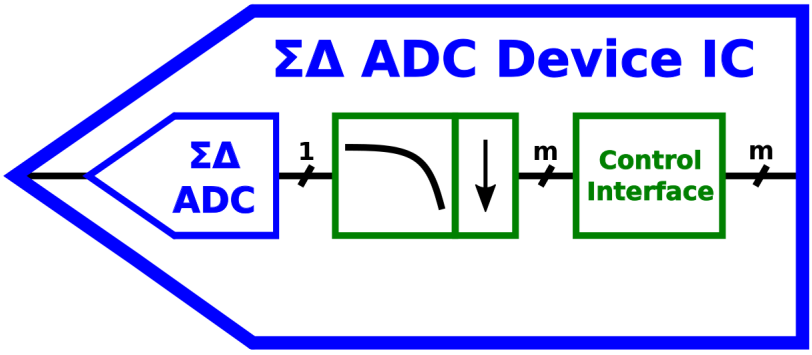


Figure 1.3: Typical ΣΔ ADC Architecture

The reason for the conversion is so that the data fits the conventional design paradigm; 2's compliment numbers running at Nyquist rates are easy to understand and design around while alternatives are not widely known. Bitstreams, on the other hand, are difficult to understand; they run at an oversampled rate and the amplitude/information of the signal is not apparent by merely inspecting the waveform. Indeed, in practice, the bit-representation noise is 50x times the signal amplitude. Despite the initial unintuitive nature of the bitstream, it will be shown that controllers with resolutions beyond 24-bits can be easily created directly from bit-stream data.

1.2 Direct ΣΔ Processing for High Performance Control

Through understanding the fundamental limitations imposed by the conventional signal chain and controller algorithm design, we now come to the question of whether or not we can supplant the conversion and interface logic and process ΣΔ encoded bitstreams directly. The answer to this quandary is a resounding yes and the primary exploration of this thesis.

Consider the embedded ΣΔ controller loop shown in figure 1.4. The diagram illustrates a signal chain where the bitstream from the front end ADC is routed directly to a purpose built controller and whose output is another ΣΔ encoded bitstream. The output bitstream can at this point be a simple one bit DAC (e.g. a half bridge driver stage). The advantages of such

a configuration are obvious. For starters, gone is the lowpass filtering and decimation of the bitstream leaving a simple one bit interface. The output of the controller is a one bit interface as well with both streams running at the oversampled clock rate. Not only is input and output interface complexity greatly reduced, but so is the controller latency. What is not seen in the diagram is that the controller will run at the oversampled clock rate in the FPGA fabric as well, creating a signal chain that has significantly reduced latency. As a consequence, controller bandwidth and stability margins increase substantially.

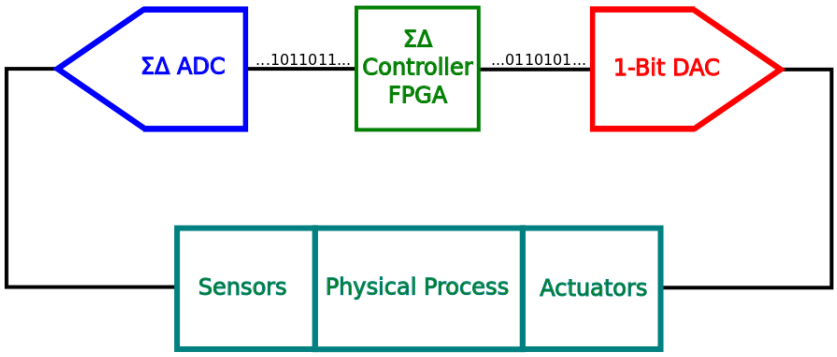


Figure 1.4: Embedded ΣΔ Control Loop

Another advantage obfuscated in the diagram is the complexity of the controller itself. As will be shown later chapters, the resource complexity of the controller in the FPGA fabric is substantially reduced compared to its conventional counterparts due to the simplification of arithmetic operations on one bit wide signals as opposed to multibit wide 2's compliment representations. The reduction in hardware complexity means that smaller, lower power, and less expensive FPGAs (or ASICs) can be used for implementations. This is due to the elimination of multipliers (the single largest digital component) from these designs.

1.3 Contributions and Thesis Outline

This dissertation presents a methodology for the optimized hardware design of $\Sigma\Delta$ controllers for use in embedded FPGA and ASIC systems. $\Sigma\Delta$ controllers are discrete circuits which emulate continuous time transfer functions and whose input and output are $\Sigma\Delta$ encoded binary bitstreams. The following contributions will be made in the remainder of the document:

1. $\Sigma\Delta$ Controller Design and Noise Analysis: For this contribution, the discrete $\Sigma\Delta$ Controller architecture will be described in great detail at the register transfer level (RTL) with attention given to choosing bitwidths in the internal signal paths of the circuit. Representation noise associated with $\Sigma\Delta$ encoded signals and the noise injection from these components are accurately modeled with an appropriate linear noise transfer function (NTF) model.

2. MASH $\Sigma\Delta$ Controllers: As a means to increase the overall resolution of a $\Sigma\Delta$ controller beyond what is possible with a single stage modulator, multistage noise shaping (MASH) $\Sigma\Delta$ converters will be introduced into the discrete controller architecture. By making modest modifications to the original controller circuitry, MASH based controllers have the capacity to achieve very high resolution at lower clock rates in comparison to the single stage designs. MASH based controllers can also be clocked at significantly lower rates to achieve the same resolution as the original single stage design. Details on how to design the MASH controller circuitry will be presented as well as an appropriate noise analysis.

3. Controller Optimization: Taking quantization noise and coefficient quantization/transfer function deviation into account, a convex optimization strategy is proposed for reducing the number of bits required for the internal signals of the controller architecture. The optimizer in turn reduces the number of flip flops required for the controller state. Given a controller design, coefficients and internal signal bitwidths are adjusted in order to meet noise and transfer function deviation metrics set by the designer.

4. Implementation and Resource and Power Analysis: Given a $\Sigma\Delta$ controller design, it will be shown how to implement the circuit in an FPGA. Various designs will be presented that

range in filter order, clock frequency, and resolution in order to assess the compactness and efficiency of the ΣΔ controller architecture. It will also be demonstrated that the designs can indeed fit within small low power FPGAs that lack any internal digital signal processing resources.

The remainder of this thesis is arranged as follows: Chapter 2 focuses on modeling and characterization of ΣΔ bitstreams. A linear frequency model of the modulator dynamics will be presented as well as how to use a power spectral density estimate of the bitstream to quantify the resolution of the encoding. Nonlinear phenomena that degrade the quality of the bitstream will also be explored as well as how to mitigate their effects. In chapter 3, the hardware description of the ΣΔ controller architecture is described. A design and performance analysis is presented as well in addition to an optimization strategy that reduces the number of bits required in the internal signals of the circuit. Chapter 4 expands on the architecture of the previous chapter by adapting it to process multistage noise shaping ΣΔ encoded input and output signals for high resolution controller applications. Chapter 5 delves into the hardware implementation of the controller architectures presented in the previous two chapters particularly in regards to FPGA design. In chapter 6, various controller/filter examples will be implements based on various design criteria and compared for resolution, power, and FPGA resource usage. The final chapter will illustrate several motivating control examples that take advantage of the high performance and low complexity of the ΣΔ controllers and show that these controllers offer a superior implementation compared to conventional embedded controller designs.

Bibliography

[1] N. N. Cikan and M. Aksoy. Analog to Digital Converters Performance Evaluation Using Figure of Merits in Industrial Applications. In *2016 European Modelling Symposium (EMS)*, pages 205–209, Nov 2016.

[2] M. B. Coskun, H. Alemansour, A. G. Fowler, M. Maroufi, and S. O. R. Moheimani. *Q* Control of an Active AFM Cantilever With Differential Sensing Configuration. *IEEE Transactions on Control Systems Technology*, pages 1–8, 2018.

[3] G. C. Goodwin, R. H. Middleton, and H. V. Poor. High-speed digital signal processing and control. *Proceedings of the IEEE*, 80(2):240–259, Feb 1992.

[4] D.A. Johns and D.M. Lewis. Sigma-delta based IIR filters. In *Circuits and Systems, 1991., Proceedings of the 34th Midwest Symposium on*, pages 210–213 vol.1, May 1991.

[5] Y. Matsuya, K. Uchimura, A. Iwata, et al. A 16-bit oversampling A-to-D conversion technology using triple-integration noise shaping. *IEEE Journal of Solid-State Circuits*, 22(6):921–929, Dec 1987.

[6] Maxim Integrated. *MAX98356: PDM Input Class D Audio Power Amplifier*, July 2013. Rev. 1.

[7] P Murphy, M Xie, Y Li, et al. Study of digital vs analog control. In *Power Electronics Seminar Proceedings (CPES Center for Power Electronics Systems)*, pages 203–206, 2002.

[8] Chiu-Wa Ng, Ngai Wong, H. Kwok-Hay So, and Tung-Sang Ng. Direct sigma-delta modulated signal processing in FPGA. In *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, pages 475–478, Sept 2008.

[9] R. Schreier and G.C. Temes. *Understanding Delta-Sigma Data Converters*. Wiley, 2004.

[10] STMicroelectronics. *MP34DT05-A: MEMS audio sensor omnidirectional digital microphone*, April 2019. Rev. 4.

[11] Texas Instruments. *ADS1204: Four 1-Bit, 10MHz, 2nd-Order Delta-Sigma Modulators*, February 2009.

[12] X. Wu and R. M. Goodall. One-bit processing for digital control. *IEE Proceedings - Control Theory and Applications*, 152(4):403–410, July 2005.

Chapter 2

$\Sigma\Delta$ Encoded Pulse Density Modulated Signals

This thesis is predicated on the notion of performing signal processing upon pulse density modulated (PDM) signal representations. This implies that provisions for both efficient input and output of PDM encoded stream data must be supported. PDM is a one bit wide oversampled bitstream signal typically produced from a $\Sigma\Delta$ modulator circuit and used extensively in analog to digital converter architecture designs. These bitstreams have a variety of unique properties such as frequency dependent representation noise that must be understood and characterized for effective use in arithmetic circuits. This chapter will describe PDM signals, how they are characterized and modeled, their drawbacks and limitations, and under what conditions $\Sigma\Delta$ modulators may be directly used as PDM encoders for embedded control loop implementations.

2.1 Pulse Density Modulation

Pulse density modulation is a signal representation in the form of a stream of bits; ones and zeros. Whereas the information in an analog signal or a discrete parallel bit word sample

(i.e. 2's complement) is contained within the amplitude, the information of a PDM signal is contained within the density of bits over a finite window of time. Take for instance the discrete sinusoidal signal and its PDM counterpart in figure 2.1.

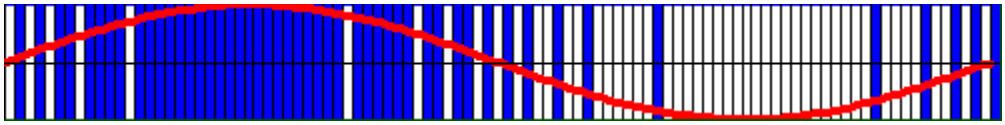


Figure 2.1: Pusle Density Modulated Sinusoid Signal

As can be seen in the figure, the density of ones increases when the sinusoidal amplitude is closer to max amplitude while the density of zeros increases when the sinusoidal amplitude is closer to minimum amplitude. While PDM bitstreams are a sequence of ones and zeros, the binary bit values of the stream can be interpreted in one of two ways:

- **Bipolar:** Suppose a discrete signal $x[n]$ is bounded in amplitude on the interval $[-A,\ A]$ where A is the maximum magnitude of $x[n]$. Then a 1 in the corresponding PDM encoded signal of $x[n]$ represents a value of A and a 0 represents a value of $-A$.
- **Unipolar:** Suppose a discrete signal $x[n]$ is bounded in amplitude on the interval $[0,\ A]$ where A is the maximum magnitude of $x[n]$. Then a 1 in the corresponding PDM encoded signal of $x[n]$ represents a value of A and a 0 represents a value of 0.

In this work, PDM signals will be interpreted as bipolar in order to take advantage of the symmetry of the representation and ability to simply represent negative values.

Given a PDM signal, it may become useful to demodulate the bitstream in order to recover the original pulse code modulated (PCM) signal. To first order, the original signal can be obtained via discrete integration of the bitstream. Suppose a bitstream signal $q[n] \in [-A,\ A]$. Its PCM counterpart can be obtained by

$$x[n] = \frac{1}{\Delta} \sum_{k=-\infty}^{n} q[k]$$

where Δ is the sampling period of the oversample rate. While simply integrating may not provide the best demodulation technique as opposed to high order lowpass filtering, it does hint at the fact that the signal information in $q[n]$ is contained within the baseband component. To better understand how to make use of bitstream signals and preserve their information, one must first understand how they are characterized. When encoding signals with PDM bitstreams, it is appropriate to ask questions about whether or not the bits are making an accurate representation. In particular, it is important to understand the nature of both the signal representation and the implicit noise inherent in representing a signal as a PDM bitstream. In the time domain, there is great difficulty in determining what the average value of the bitstream is at any given moment; one must first filter the bitstream to reveal its analog counterpart. However, in the frequency domain, the question becomes much clearer.

The power spectral density (PSD) of a PDM signal is the principle representation in which to analyze and characterize a bitstream as its signal and noise power versus frequency. The PSD reveals characteristics about a bitstream such as the signal it is encoding, the quality of the signal, and the distribution of the noise inherent in the representation. The anatomy of a power spectral density of a pulse density modulated signal can be seen in figure 2.2.

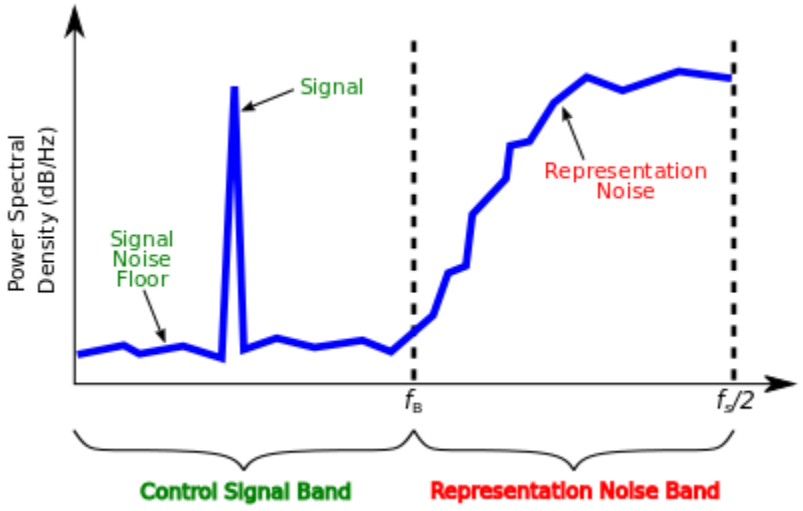


Figure 2.2: Anatomy of PDM Signal PSD

The PSD pictured is that of a PDM encoded discrete sinusoidal signal with a fundemental frequency of f_{sig}. The signal component can be seen as a spike at f_{sig} . Perhaps the most striking aspect of the PSD is the shape of the noise floor. The noise floor of the signal is flat in the lower frequencies up to f_B which corresponds to the input resolution of the encoded discrete signal or noise-floor associated with the input analog signal. (Abstractly, there is no lower bound on the PSD noise at low frequencies if sufficiently large resolution is used). In the higher frequencies, from f_B to $f_s/2$, the noise floor increases dramatically and saturates the power spectrum. The concentration of noise power in the higher frequencies is a hallmark of PDM signals and is in fact a by-product of the feedback correction nature of the encoder. That is, the high frequency noise is inherent in the representation but is pushed out of the signal band (i.e. frequencies below f_B) and into the representation noise band (i.e. frequencies above f_B). The quality of pushing or shaping the representation noise into high frequencies allows PDM bitstreams to achieve a very high quality encoding at the cost of the higher oversampled rate f_s versus the desired signal bandwidth f_B. Since the over-sampled signal is a binary bit-stream, it admits a variety of simple computation mechanisms all operated at the oversampled clock (sample) rate. The large ratio $f_s/\left(2f_B\right)$ allows for very low noise representation of the desired signal band, while the binary nature of the output admits single threshold (hence very linear) ADC. It is for this reason that analog sigma-delta converters are the ubiquitous solution for high-resolution analog signal conversion, at any rate allowing a reasonable over sampling ratio.

In this thesis, the characterization of a PDM bitstream will be derived from its power spectral density. Thus is it necessary to formally introduce a variety of definitions and concepts that will aid in their proper use in later chapters:

- Signal Band: The signal band is the frequency range[1] from DC to f_B which signal components can exist without the contamination of high levels of background noise.

[1]It is not strictly necessary for the signal band to be base-band and modulated converters exist, however, the design of computation schemes for such designs is relegated to future work. In this work, we shall assume based-band conversion.

- Representation Noise Band: The representation noise band is the frequency range from f_B to $f_s/2$ where the representation noise is pushed and dominates the signal spectral density.

- OSR: The oversample ratio (OSR) is the defined as the ratio between the oversampling frequency f_s and the Nyquist frequency $2f_B$

$$OSR = \frac{f_s}{2f_B}$$

where f_B is the signal bandwidth.

- SNR: The signal to noise ratio (SNR) is defined as the ratio between the signal power μ_x^2 and the noise variance σ_n^2 in the signal band from DC to f_B. The ratio is commonly expressed in units of decibels by

$$SNR\,(dB) = 10log_{10}\left(\frac{\mu_x^2}{\sigma_n^2}\right)$$

Both μ_x^2 and σ_n^2 can be computed directly from the power spectral density of the PDM signal by integrating over the corresponding spectral density of the signal band f_B. The noise variance for example can be computed by

$$\sigma_n^2 = \int_{-f_B}^{f_B} S_n\,(f)\,df$$

where $S_n\,(f)$ is the noise power spectral density. The SNR provides a metric for the quality of the encoded bitstream signal.

- SINAD: The signal to noise and distortion ratio is defined as the ratio between the signal power μ_x^2 and the noise variance σ_n^2 plus the distortion variance σ_d^2 in the signal band from DC to f_B. This ratio is commonly expressed in units of decibels by

$$SNR\,(dB) = 10log_{10}\left(\frac{\mu_x^2}{\sigma_n^2 + \sigma_d^2}\right)$$

Distortion can manifest as undesired frequency components in the power spectral density which lower the overall quality of the signal.

- ENOB: The effective number of bits (ENOB) is a measure of how many bits of signal information are contained in the bitstream signal based on given SNR or SINAD. The ENOB can be calculated by

$$ENOB = \frac{SINAD - 1.76}{6.02}$$

which is derived from and a direct comparison to Nyquist rate pulse code modulated encodings. [50]

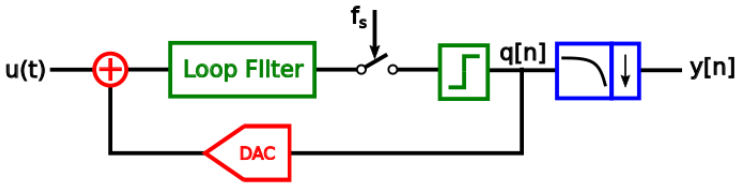
The above definitions and concepts provide a basis for the complete characterization of PDM signals that will be used throughout this thesis. It will become evident in later chapters that utilizing PDM signals in signal processing and control-centric hardware implementations requires the ability to maintain the shape and form of their PSD's for high quality computations. In fact, there will be many times when a signal must first be encoded into a PDM in the first place in order to change representations when it is convenient to do so. The circuit to do the conversion from analog/PCM to PDM encoding, the workhorse of this thesis, is the $\Sigma\Delta$ modulator.

2.2 $\Sigma\Delta$ Modulators

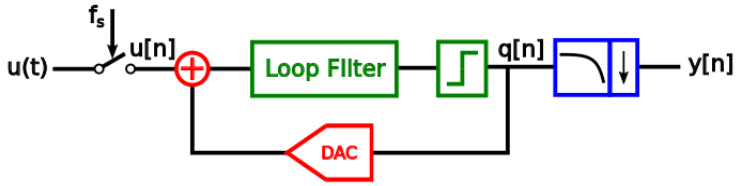
The $\Sigma\Delta$ modulator is the converter/encoder circuit that transforms an analog or pulse code modulated signal into a high resolution bitstream of 1's and 0's. While most widely utilized in analog to digital converter architectures, they also find use in digital to analog converters as well as discrete applications such as phase lock loops [20]. By making use of oversampling and noise shaping, $\Sigma\Delta$ modulators are able to create high SNR/ENOB PDM bitstream signals. This section will introduce and explain the $\Sigma\Delta$ theory of operation.

When designed for the purpose of analog to digital conversion, $\Sigma\Delta$ modulators come in two varieties: analog and discrete. Figure 2.3 illustrates block diagrams of discrete and analog $\Sigma\Delta$ ADC modulators with common blocks such as the loop filter, quantizer, and DAC. The

primary difference between the two is where the discrete sampling takes place; in front of the modulator for discrete and before the quantizer for analog. At the circuit level, the loop filter of the discrete modulated is implemented with a switched capacitor architecture. The analog loop filter can be implemented with gm-C, active RC, LC, or other circuit architectures [50].



(a) Continuous Time ΣΔ



(b) Discrete Time ΣΔ

Figure 2.3: a) Continuous Time and b) Discrete Time ΣΔ Modulators

Discrete ΣΔ modulators can be modeled as a discrete piece-wise affine (PWA) system with bimodal dynamics. Consider the following ΣΔ modulator PWA model

$$x^+ = \begin{cases} A_{\Sigma\Delta}x + B_{\Sigma\Delta}u + f_1 & \forall x_1 \geq 0 \\ A_{\Sigma\Delta}x + B_{\Sigma\Delta}u + f_2 & \forall x_1 < 0 \end{cases}$$

$$q = \begin{cases} 1 & \forall x_1 \geq 0 \\ -1 & \forall x_1 < 0 \end{cases}$$

where $A_{\Sigma\Delta} \in \mathbb{R}^{n\times n}$, x, $B_{\Sigma\Delta}$, f_1, $f_2 \in \mathbb{R}^n$, $n \in \mathbb{Z}^+$and $q \in [-1,\ 1]$. The nonlinear discontinuous nature of the dynamics imposed by the affine term f_i makes the modulator difficult to directly analyze and establish performance metrics. As will be seen shortly, the shape of the representation noise in Figure 2.2 is highly dependent upon the input signal to the modulator and requires that it hold to various limitations in order to produce a high quality signal encoding. To better ascertain the mysteries of the bitstream, it becomes necessary to fabricate a more tractable method for modeling its behavior.

2.2.1 $\Sigma\Delta$ Linearized Model

While the dynamics of $\Sigma\Delta$ modulators are highly nonlinear, there are key assumptions about the nature of the quantization noise which make the analysis more tractable. In order to characterize $\Sigma\Delta$ encoded bitstreams and estimate their performance, one must have a simple way of modeling the modulator dynamics. Fortunately, a simple model of the $\Sigma\Delta$ modulator can be derived by making key assumptions about the nature of its one bit quantizer.

Treating the one bit quantizer as an additive noise source, the dynamics of the $\Sigma\Delta$ modulator can be analyzed as a linear dynamic system with a stochastic noise input. The invocation of a linear dynamic $\Sigma\Delta$ model is a well known and widely used technique and has been presented by authors such as Temes and Schreier in [50]. The linearization of the quantizer can be seen in Figure 2.4 while the linearization transformation of a discrete $\Sigma\Delta$ modulator can be seen in figure 2.5.

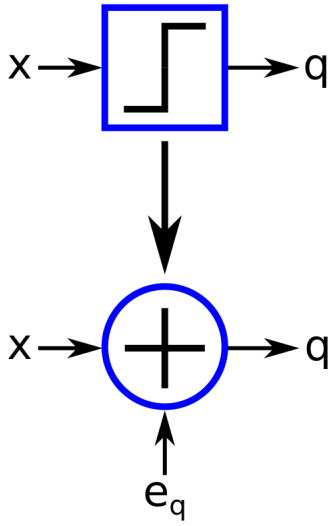


Figure 2.4: ΣΔ Quantizer Linearization

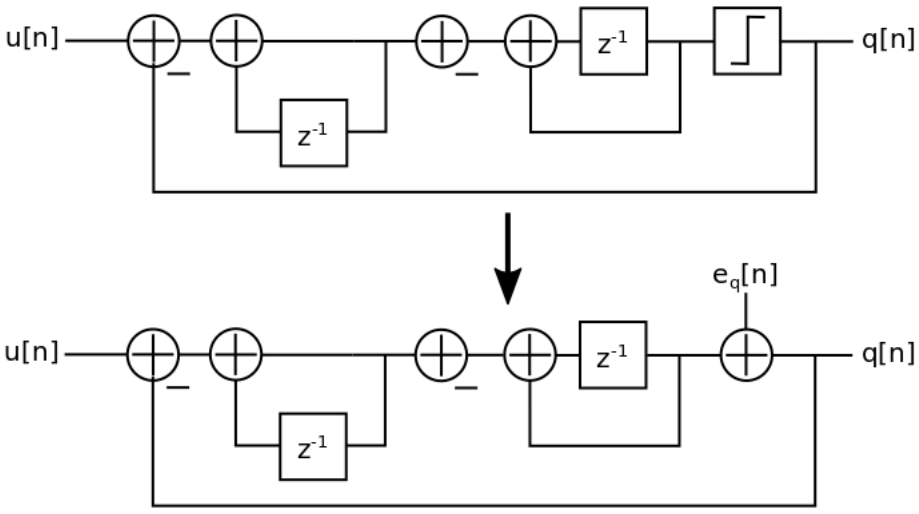


Figure 2.5: Linearization of 2nd Order ΣΔ

The linearized model allows one to modify the nonlinear state space dynamics of a modulator into the linear two input one output model:

$$x^+ = A_{\Sigma\Delta}x + B_{\Sigma\Delta}u + B_e e$$
$$q = C_{\Sigma\Delta}x + e$$

where e is the additive quantization noise input, $A_{\Sigma\Delta} \in \mathbb{R}^{n\times n}$, x, $B_{\Sigma\Delta}$, $B_e \in \mathbb{R}^{n\times 1}$, $C_{\Sigma\Delta} \in \mathbb{R}^{1\times n}$, e, $q \in \mathbb{R}$, and $n \in \mathbb{Z}^+$. Taking the z transform of the linear time domain state space model above, the two following transfer functions from u to q and from e_q to q can be found as

$$STF(z) = \frac{Q(z)}{U(z)} = C_{\Sigma\Delta}\left(zI - A_{\Sigma\Delta}\right)^{-1} B_{\Sigma\Delta}$$

$$NTF(z) = \frac{Q(z)}{E(z)} = C_{\Sigma\Delta}\left(zI - A_{\Sigma\Delta}\right)^{-1} B_e + 1$$

The signal transfer function, or STF, is the transfer function from the input to the output of the sigma delta without the addition of quantization noise. The noise transfer function, or NTF, is the quantization noise input to the output of the sigma delta without the signal component. The overall output response of $\Sigma\Delta$ modulator can be found by

$$Q(z) = STF(z)\,U(z) + NTF(z)\,E_q(z)$$

which allows the information to be split between a signal component and representation noise component. The concept of separable signal and noise components is paramount to formulating the power spectral density in order to ascertain the performance of the modulator. To estimate the PSD of the signal output the following assumption about the quantization noise are made:

1. The quantization error $e_q[n]$ is a wide sense stationary, white Gaussian random process.
2. The quantization error $e_q[n]$ is uncorrelated with itself and and the input sequence of the $\Sigma\Delta$ modulator.

3. The probability-density function of the quantization error $\rho(e_q)$ is uniform over the entire quantization range $\left[-\frac{q}{2},\ \frac{q}{2}\right]$.

$$\rho(e_q) = \begin{cases} \frac{1}{q} & |e_q| \leq \frac{q}{2} \\ 0 & |e_q| > \frac{q}{2} \end{cases}$$

where q is the quantization step.

While these assumptions are not always true, they do give a reasonable description of the actual quantization noise properties inherent in the nonlinear model for small and large amplitude signals with ample frequency content. From here, under these assumptions and the linear model, the PSD of the modulator output can be estimated.

Based on assumption that $\rho(e_q)$ is uniform over the quantization range $\left[-\frac{q}{2},\ \frac{q}{2}\right]$, we can find the mean of e_q as

$$\bar{e}_q = E\{e_q\} = \int_{-\infty}^{\infty} e_q \rho(e_q)\, de_q = \frac{1}{q} \int_{-q/2}^{q/2} e_q de_q = 0$$

and the variance of e_q as

$$\sigma_e^2 = E\left\{(e_q - \bar{e}_q)^2\right\} = \int_{-\infty}^{\infty} e_q^2 \rho(e_q)\, de_q = \frac{1}{q} \int_{-q/2}^{q/2} e_q^2 de_q = \frac{q^2}{12}$$

Due to the assumption that the noise is white, the nominal quantization noise power σ_e^2 is spread out uniformly over the entire frequency space. The power spectral density of the additive quantization noise input is thus

$$S_e(f) = \frac{\sigma_e^2}{f_s}$$

and

$$S_e(f) = \frac{\sigma_e^2}{f_s} = \frac{q^2}{12 f_s} = \frac{1}{3 f_s}$$

for a unipolar single bit quantizer (i.e. $q = 2$).

At this point, the filtering of $S_e(f)$ through the noise transfer function $NTF(f)$ must be taken into account. The noise transfer function $NTF(f)$ shapes the power spectral density of the additive quantization noise which leads to a noise power over the signal band to be found by the equation

$$\sigma^2_{\Sigma\Delta} = \int_{-f_b}^{f_b} S_e(f) \left| NTF(f) \right|^2 df$$

Typically, $NTF(f) \ll 1$ in the signal band and $NTF(f) \approx 1$ in the representation noise band. The noise transfer dynamics thus attenuate signal band noise energy and push the vast majority of the noise energy into high frequencies. This is called noise shaping. The signal transfer function $STF(f)$ on the other hand is typically unity gain in the frequency band allowing the signal to pass through while suppressing the representation noise. This leads to a very high SNR in the signal band of the PSD.

In order to characterize the performance of the modulator, sinusoidal inputs are generally chosen to calculate signal to noise ratios for individual frequency components. Supposing a sinusoid signal with period T and amplitude A, its average power over the signal band can be found by

$$\mu_x^2 = \frac{1}{T} \int_0^T \left(A cos\left(\frac{2\pi t}{T} \right) \right)^2 dt = \frac{A^2}{2}$$

Having the average noise power of a sinusoidal input at any frequency depend only upon the amplitude of the signal allows a convenient way of calculating SNR as given by

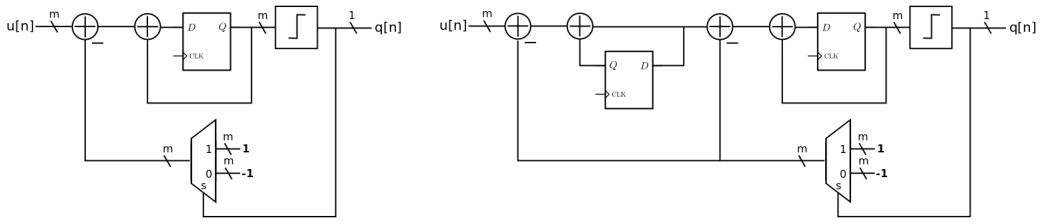
$$SNR_{dB} = 10 \cdot log_{10} \left(\frac{\mu_x^2}{\sigma^2_{\Sigma\Delta}} \right)$$

In principle, the SNR should remain constant in the signal band for a sinusoid at any frequency for a given modulator.

2.2.2 1st and 2nd Order Noise Shaping Representations

The noise transfer function of the ΣΔ modulator defines the noise shaping that manifests in the passband of the bitstream signal. To have a sharper noise transfer function is to have a potentially higher signal to noise ratio. The shape of the noise transfer function corresponds to the order of the modulator or rather the number of delay elements that exist in the loop.

The first and second order discrete ΣΔ modulators shown in figure 2.6, consists of one and two accumulators respectively as well as a 1 bit quantizer in the feedforward path.



(a) 1st Order Discrete ΣΔ (b) 2nd Order Discrete ΣΔ

Figure 2.6: 1st and 2nd Order Discrete ΣΔ Modulators

By imposing the linear additive noise model on the 1st and 2nd order modulators it becomes straightforward to derive their signal and noise transfer finctions. For the 1st order modulator the STF and NTF are

$$STF\,(z) = z^{-1}$$

$$NTF\,(z) = 1 - z^{-1}$$

while for the 2nd order modulator, the STF and NTF are

$$STF\,(z) = z^{-1}$$

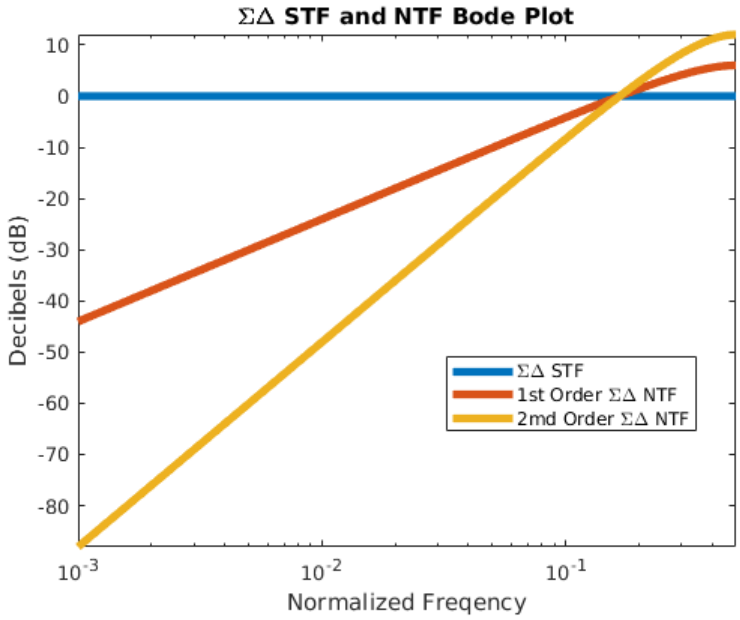


Figure 2.7: 1st and 2nd Order ΣΔ STF and NTF Bode Plot

$$NTF\left(z\right) = \left(1 - z^{-1}\right)^{2}$$

Figure 2.7 shows the Bode plot of the signal and noise transfer functions for the 1st and 2nd order modulators. As stated earlier, the signal transfer function for a discrete modulator is unity gain across the signal band while the noise transfer functions are akin to high pass filters.

The first and second order noise shaping shown in Figure 2.7 have slopes of 20 dB/decade and 40 dB/decade respectively. Higher order modulators with higher order noise shaping can achieve even steeper noise shaping but there is a catch. ΣΔ converters of third or higher order tend to become unstable with quickly varying input signals [50]. Specifically, a second-order modulator is unconditionally stable given a bound on input magnitude and the magnitude of the input time derivative (i.e. simple bandwidth limitation). For 3rd-order, a similar bound is required on the second derivative of the input as well. This does not translate to a simple band-stop filter so higher order modulators require substantial care in design [68].

The stability of the modulator is very important to the quality of the ΣΔ bitstream; if the modulator becomes unstable, the bitstream conversion can be wildly inaccurate. The reason for instability is due to the increase phase added by the additional integrators in the converter architecture. If an input is injected that is out of phase with the feedback from the one bit quantizer, the ΣΔ accumulators will begin to become unbounded and produce an incorrect bitstream coding of the input signal. It is well know that for third or higher order modulators, higher SNRs can be achieved, but only for more constrained input signals that have more slowly changing values. Second order modulators on the other hand can achieve 14 to 16 ENOB for modest OSRs of 128 to 256 and are stable over a wide class of input signals making them a practical choice for high quality encodings without having to worry too much about the modulator stability. For the reasons previously stated, the second order modulator model will used in the remainder of this thesis for bitstream encoder analysis and circuit architecture.

2.3 Conditions for Accurate ΣΔ Bitstream Encoding

While the linear model of the ΣΔ modulator is fairly accurate for a wide class of input signals, there are a two cases in which it will be difficult for a modulator to produce an accurate bitstream encoding. The two causes of potential encoding issues for ΣΔ modulators are code noise and limit cycles. By taking the two issues into account, steps can be taken to ensure that bitstream encodings maintain their accuracy and noise shaping.

2.3.1 Code Noise

The first major pitfall of PDM signals is that the number of bitstream codes that exist to represent a particular continuous value become more and more sparse as amplitudes approach the maximum and minimum limits. Originally introduced as "code noise" by Lindquist [18], the reality of limited codes has the ability to significantly reduce the signal to noise ratio of high amplitude signals.

To demonstrate the deleterious effects of code noise, suppose a PDM signal with a signal bandwidth of f_b and an oversample ratio of OSR. In order for the PDM signal to accurately

encoded a discrete or analog signal at its highest rate of change, it has, at the minimum, an OSR number of bits to do so. This puts a constraint on the $\Sigma\Delta$ modulator to settle and find a code in at least an OSR number of clock cycles. How fast a $\Sigma\Delta$ encoder settles depends upon its dynamics. Intuitively, higher order $\Sigma\Delta$ modulators have increased closed loop phase delay due to their addition integrators and as such reduce their ability to react to large amplitude and or quickly changing inputs.

Consider a PDM bitstream representation of all 1's (i.e. ...11111111111...) which is the code for a signal at maximum signal amplitude A. It becomes apparent that there is only one code to represent a maximum signal amplitude of A. Likewise an encoding of ...00000000000... is the only code that represents a signal amplitude of $-A$. When it comes to encoding a value that falls within the range $[A,\ -A]$, it will necessitate a k number of bits to be 1's out of a stream of OSR bits. The number of codes that exist in this context can be determined by invoking the binomial coefficient

$$C_k^{OSR} = \left(\frac{OSR}{k}\right) = \frac{OSR!}{k!\,(OSR-k)!}\ codes$$

Again, for a maximum value of A ($k = OSR$) and minimum value of $-A$ ($k = 0$), the number of codes is

$$C_0^{OSR} = C_{OSR}^{OSR} = 1\ code$$

For $k = 1$ and $k = OSR - 1$, the number of codes is

$$C_1^{OSR} = C_{OSR-1}^{OSR} = OSR\ codes$$

In contrast, a mid-range signal value of zero ($k = {}^{OSR}/_{2}$), which has an equal number of 1's and 0's, has

$$C_{{}^{OSR}/_{2}}^{OSR} = \frac{OSR!}{({}^{OSR}/_{2})!^2}\ codes$$

For $OSR = 16$, there are 12870 possible codes for a mid-range value of zero and only one code for a maximum value of A which illustrates a vast imbalance in the code density over

the representation space. Figure 2.8 shows the density of codes over the entire amplitude range $[A,\ -A]$ for an OSR of 16.

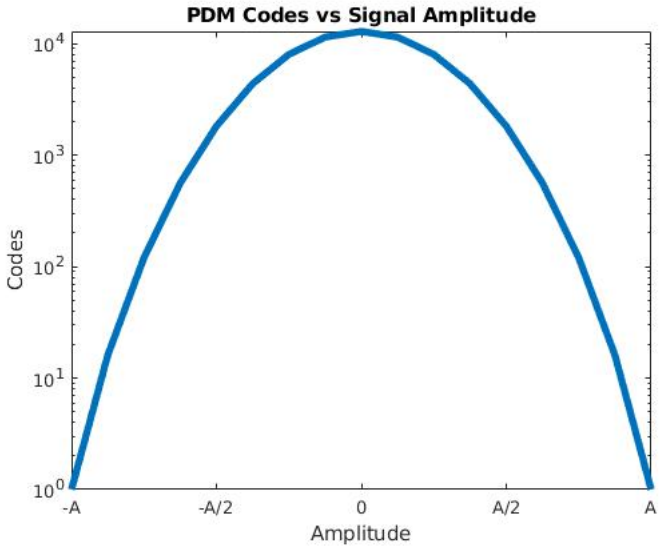


Figure 2.8: PDM Codes vs Signal Amplitude for $OSR = 16$

To make matters worse, only a small subset of the 2^{OSR} number of possible codes over an OSR length bitstream are possible for a given $\Sigma\Delta$ encoder. This means that while there may be an N number of codes to represent a certain signal amplitude, the dynamics of the modulator and the value of its present state prevent it from reaching every single one. Figure 2.9 shows a contour plot of SNR vs OSR and sinusoidal input amplitude for the 2nd order $\Sigma\Delta$ modulator shown in figure 2.6.

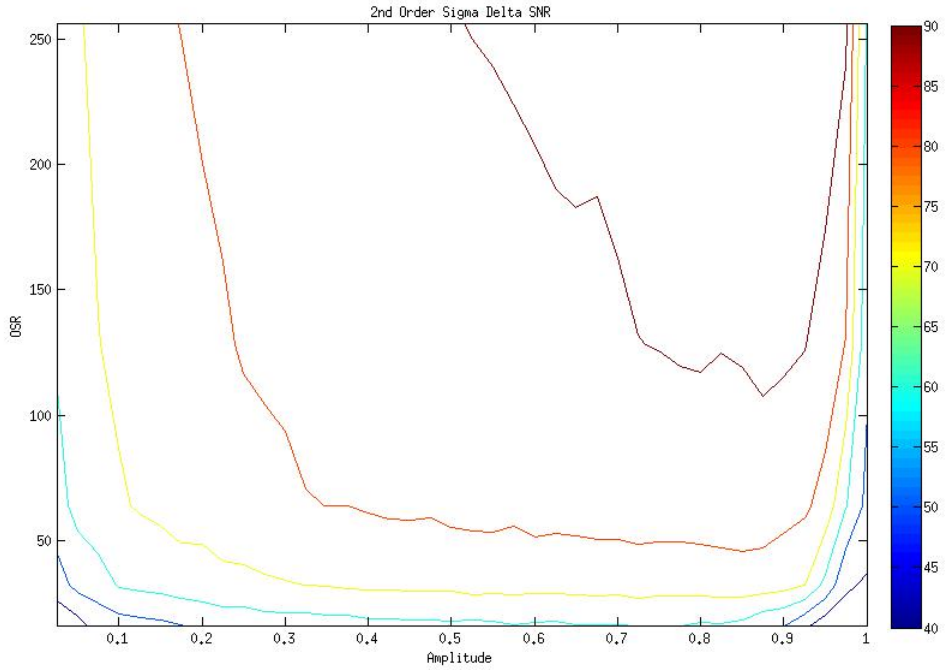


Figure 2.9: SNR vs OSR and Amplitude

In above plot, the input signal used to estimate the SNR is a sinusoid with a frequency of $^3/_4 f_B$. The contour plot shows what one would expect; an increase in SNR with an increase in OSR and amplitude. However, SNR rolls off precipitously as the amplitude approaches its maximum value. This roll-off demonstrates the decreased ability for the modulator to find the correct code for a quickly varying signal (relative to the Nyquist rate) at the sparse extremes of the representation space. The graph also shows that SNR rolls off lower in the amplitude range for lower OSR values. Again, a lower OSR means less time for the modulator to find the correct code for a given input signal amplitude and hence the roll off begins relatively sooner in the amplitude range.

To ensure that the bitstream encoding does not fall prey to code noise, one can simple limit the amplitude of the input signal to the modulator. In the case of the discrete second order modulator, the SNR vs OSR plot suggests that one could limit the input signal maximum amplitude to roughly 90% of the full scale input of the modulator. Doing so would ensure that

the required codes for encoding a signal range of $[0.9 \cdot A,\ -0.9 \cdot A]$ are not sparse and difficult for the modulator to find in time.

2.3.2 Limit Cycles

The primary source of distortion in the power spectral density of a $\Sigma\Delta$ encoded pulse density modulated constant (D.C.) signal is that of a limit cycle which manifest as repeating sequences in the output bitstream. Limit cycles create spurs that show up in the PSD and increase the integrated noise variance used to estimate the signal to noise ratio due to the distortion. As such, limit cycles are unwanted artifacts that reduce signal integrity by introducing frequency components that are not present in the input signal of the $\Sigma\Delta$ encoder.

Consider the first and second order $\Sigma\Delta$ encoders with the various DC input values and corresponding bitstream outputs in Table 2.1

DC Input Value	Bitstream Output
0 (1st Order)	...0101010101010101...
0 (2nd Order)	...0011001100110011...
+1/2 (1st Order)	...0111011101110111...
+1/2 (2nd Order)	...0110111101101111...
-3/4 (1st Order)	...1000000010000000...
-3/4 (2nd Order)	...1000000100000000...

Table 2.1: DC $\Sigma\Delta$ Input and Corresponding Bitstream Output

From the table, it is apparent that the corresponding limit cycle bitstream pattern is a function of the modulator dynamics. The repeating sequences will show up as peaks, or tones, in the power spectral density which add to the overall distortion and degrade the SNR. Typically, low frequency signals will be more apt to produce limit cycles as the modulator will have more time to find the proper bitstream code. In any case, limit cycles produce tones that are not welcome and reduce the overall SNR of the signal.

The answer to suppressing limit cycles is dithering. By adding a minute amount of noise to the input of a $\Sigma\Delta$ converter, one can effectively scramble the least significant bits of the converters state registers so that repeating sequences are eliminated. Several works detail

methods of dithering such as additive shaped least significant bit dithering on the ΣΔ input [16, 14, 22, 15, 17, 19].

It may often be the case however that additive dithering to a ΣΔ modulator is completely unnecessary. In most embedded control system applications, the physical system and the process of measuring a signal of interest will already include a noise floor. The noise floor can be attributed to such things as thermal noise or uncertainty in the front end transducer. This natural noise floor in of itself will randomize the lower bits of the ΣΔ converter state and quell any limit cycle that may be produced. In later chapters, a natural noise floor will be assumed on the front end of the embedded signal chain and will be specified in any practical examples.

2.4 Conclusion

This chapter presents a means to accurately model and characterize ΣΔ bitstream signals for use in the remainder of the thesis. A linear model of the discrete ΣΔ modulator was presented based on assumptions made of the single bit quantizer noise. The power spectral density estimation based on the linear modulator model has been introduced in order to accurately predict the quality, or SNR, of the bitstream. Issues that may invalidate the linear model approximation which include code noise and limit cycles have been addressed with suggestions on how to mitigate their performance degradation of the bitstream. In the following chapter, the linear quantization noise modular model will be used in order to design and characterize a filter architecture that directly process bitstream encodings.

Bibliography

[13] Jose De la Rosa and Rocio Del Rio. CMOS Sigma-Delta Converters: Practical Design Guide. *CMOS Sigma-Delta Converters: Practical Design Guide*, pages i–xxviii, 03 2013.

[14] B. Fitzgibbon, K. O'Neill, A. Grannell, et al. A spur-free MASH digital delta-sigma modulator with higher order shaped dither. In *2009 European Conference on Circuit Theory and Design*, pages 723–726, Aug 2009.

[15] B. Fitzgibbon, S. Pamarti, and M. P. Kennedy. A Spur-Free MASH DDSM With High-Order Filtered Dither. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 58(9):585–589, Sept 2011.

[16] K. Hosseini and M. P. Kennedy. Architectures for Maximum-Sequence-Length Digital Delta-Sigma Modulators. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(11):1104–1108, Nov 2008.

[17] M. P. Kennedy, B. Fitzgibbon, and K. Dobmeier. Spurious tones in digital delta sigma modulators with pseudorandom dither. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 2747–2750, May 2013.

[18] C. S. Lindquist. Code noise in delta-sigma modulators. In *Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers (Cat. No.98CH36284)*, volume 2, pages 1012–1016 vol.2, Nov 1998.

[19] H. Mo and M. P. Kennedy. Masked Dithering of MASH Digital Delta-Sigma Modulators With Constant Inputs Using Multiple Linear Feedback Shift Registers. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 64(6):1390–1399, June 2017.

[20] M. H. Perrott, M. D. Trott, and C. G. Sodini. A modeling approach for /spl Sigma/-/spl Delta/ fractional-N frequency synthesizers allowing straightforward noise analysis. *IEEE Journal of Solid-State Circuits*, 37(8):1028–1038, Aug 2002.

[21] R. Schreier and G.C. Temes. *Understanding Delta-Sigma Data Converters*. Wiley, 2004.

[22] J. Song and I. Park. Spur-Free MASH Delta-Sigma Modulation. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(9):2426–2437, Sept 2010.

[23] X. Wu and R. M. Goodall. One-bit processing for digital control. *IEE Proceedings - Control Theory and Applications*, 152(4):403–410, July 2005.

Chapter 3

Performance, Design, and Optimization of the $\Sigma\Delta$ Controller Architecture

Oversampled PDM signals provide an inexpensive encoding mechanism for time dependent signals. In this chapter, direct processing strategies are described to implement high performance filters/controllers using such streams for both input and output. Despite the ubiquitous utility of Z-transform and time-shift techniques in digital filters, it will be shown that they are of little use to construct filters based on PDM representations. In particular, operations between time-shifted oversampled streams require expensive trains of delay elements (since the stream is high rate) and unfortunately have extremely poor coefficient sensitivity and stability characteristics.

Instead, filters based on integrators and direct implementation of state-space variables will be described. Since the PDM representation is binary, filter coefficient scaling is simplified to the selection of a multi-bit constant or its complement to add to the integrator input. In this way, very efficient filters can be constructed without the multiplier scaling elements that are the cornerstone of conventional digital signal processing. Given the 50 years of continual

improvement in the performance of digital systems, operation at the oversampling rate is practical for such rates easily exceeding 1 GHz, leading to very high resolution filters with practical bandwidths exceeding 10MHz using these techniques. (In later chapters, we will show how to improve this to 50MHz and beyond).

While the architecture of PDM and integrator based filters has been proposed elsewhere, here we use the nature of the PDM representation and the sensitivity characteristics of the architecture to construct an optimizing algorithm minimizing the coefficient and integrator bit-widths subject to performance constraints. This minimizes the component (logic) footprint as well as the design power needed to implement the filter. Further, since the filter architecture latency is measured in small numbers of oversample periods, the filter latency is extremely low, enhancing its utility in feedback control applications.

In this chapter, a $\Sigma\Delta$ controller architecture that emulates transfer functions will be presented along with its register transfer level (RTL) design. The $\Sigma\Delta$ controller architecture will directly process bitstreams at the oversample rate and achieve near continuous time LTI control performance while having an extremely small multiplierless design that is both low power, low latency, and high resolution. In order to gauge performance, a novel quantization noise analysis and model will be presented in order to determine the output SNR of the controller logic ensemble. A metric for estimating transfer function variation due to coefficient quantization will be included as well. Using both the noise model and the transfer function variation estimate, a convex optimization scheme will be invoked in order to minimize the number of state register bits needed to construct the $\Sigma\Delta$ controller while still meeting user defined performance metrics and goals. The chapter will conclude with an example transfer function design which will outline the performance and capability of the $\Sigma\Delta$ controller architecture.

3.1 Mathematical Tools and Assumptions

Filters are a fundamental building block in embedded signal processing systems. Their ability to discriminate between and select designated frequency components from signals makes them indispensable in numerous applications ranging from telecommunications to control sys-

tems. In particular, the linear time invariant (LTI) variety are of great importance as they emulate transfer functions and can be implemented using simple mathematical operations such as addition, multiplication, and delays. Due to the relative ease of design and simple implementation, linear time invariant filters are extremely popular and have found great success in real time embedded signal processing and control applications.

The current paradigm in discrete filter design revolves around shift based algorithms. The shift operator q can be defined as

$$qx\left(k\Delta\right) = x\left(k\Delta + \Delta\right)$$

where $k \in \mathbb{Z}_{\geq 0}$ is the discrete time step, $\Delta \in \mathbb{R}$ is the sampling period, and $x \in \mathbb{R}$ is a state variable, and serves as the fundamental basis for storing filter state in discrete filter designs.

Direct filtering of oversampled bitstream signals implies running filter computations at the oversampled rate. In conventional signal processing and control implementations based on the forward shift operator q, sampling at rates much higher than the underlying system dynamics can lead to issues of ill conditioning and stability [62]. Consider the continuous time state space filter representation

$$\dot{x} = A_c x + B_c u$$
$$y = C_c x + D_c u$$

where $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, $u \in \mathbb{R}^k$, $A_c \in \mathbb{R}^{n\times n}$, ${}_cB \in \mathbb{R}^{n\times k}$, $C_c \in \mathbb{R}^{m\times n}$, $D_c \in \mathbb{R}^{m\times k}$, and $n, m, k \in \mathbb{Z}_{\geq 0}$. If one were to discretize the state transition matrix A_c using the zero order hold method and take the limit of the sampling period Δ as it approaches zero, the result would be

$$\lim_{\Delta\to 0} A_q = \lim_{\Delta\to 0} e^{A_c\Delta} = I$$

That is, as the sampling period approaches zero, the poles of the filters converge to $z = 1$ on the unit circle in the complex plane. In fixed point realizations where the coefficients of a filter are represented with finite precision, the possible locations of filter poles in the unit circle stability region of the Z domain become sparse in and around the real axis. Figure 3.1 illustrates this

effect for a fourth order lowpass filter while increasing the sample rate from 1kHz to 10kHz. As can be seen the poles and zeros converge quickly. What is more disturbing is that the black dots represent the quantization points for 5 bit coefficients in the filter.

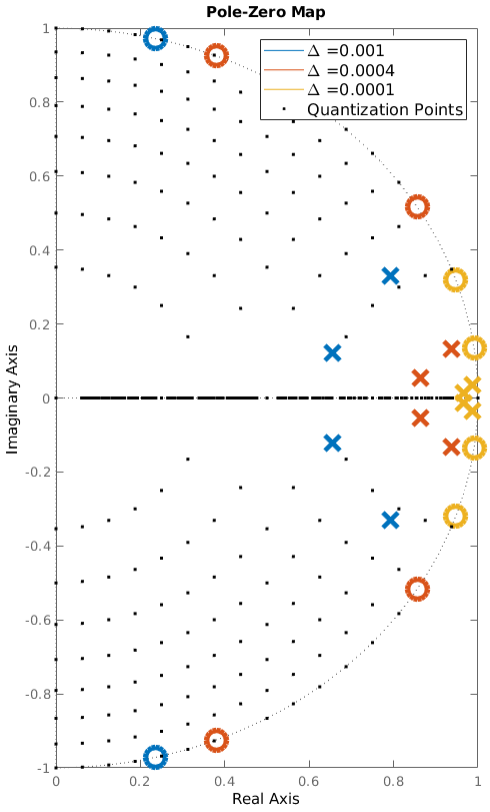


Figure 3.1: 5-bit Fixed Point Pole Locations for Direct Form IIR Filter

Notice that $z = 1$ is not a possible location within the unit circle for complex conjugate poles with distinct complex components. As sampling rates increase, discerning between distinct complex conjugate poles requires larger coefficient bitwidths leading to ever expanding logic implementations. Without, large coefficient bitwidths, the filter will experience severe filter magnitude and phase response deviations as well as the potential for instability. It seems counter intuitive that as the sample rate increases the numerical stability of the fixed point filter

implementation decreases, but must it be so? For oversampled bitstream processing where the resolution of the representation increases as the sampling period converges to zero, a different filter operator is required that does not increase the coefficient sensitivity to the point of absurdity. Thankfully, an alternative to the shift operator exists that mitigates these issues: the δ-operator.

The δ-operator is a difference based operator defined as

$$\delta x\left(k\Delta\right) \triangleq \frac{x\left(k\Delta+\Delta\right)-x\left(k\Delta\right)}{\Delta}$$

which is analogous to a discrete derivative approximation. Taking the limit of $\delta x\left(k\right)$ as the sampling period approaches zero we have

$$\lim_{\Delta\to 0}\delta x\left(k\Delta\right)=\lim_{\Delta\to 0}\frac{x\left(k\Delta+\Delta\right)-x\left(k\Delta\right)}{\Delta}=\frac{dx}{dt}$$

which demonstrates the operators ability to approximate continuous time dynamics with an increase in sampling frequency. The attractiveness of the δ-operator to emulate continuous dynamics is made more clear when using it as a basis for a linear dynamic system model.

By imposing a piece-wise input to the continuous variation of constants formula associated with the continuous time state space model, one can obtain the discrete δ state space representation written as

$$\begin{aligned}\delta x =& A_\delta x + B_\delta u\\ y =& C_\delta x + D_\delta u\end{aligned}$$

where

$$
\begin{aligned}
A_\delta &= \frac{e^{A_c\Delta} - I}{\Delta} \\
B_\delta &= \frac{1}{\Delta}\int_0^\Delta e^{A_c(t-\tau)} B_c u(\tau)\, d\tau \\
C_\delta &= C_c \\
D_\delta &= D_c
\end{aligned}
$$

Again, the main advantage of the δ based model is the ability for it to converge to continuous time dynamics with an increase in sampling frequency. This property can be demonstrated by taking the limit as the sampling period approaches zero of the state transition matrix A_δ. Doing so gives

$$\lim_{\Delta\to 0} A_\delta = \lim_{\Delta\to 0} \frac{e^{A_c\Delta} - I}{\Delta} = A_c$$

demonstrating that δ system dynamics do indeed converge to their continuous time counterpart. The discrete δ-operator based model avoids the singularity approach imposed by high sampling rates. Given oversampling rates of 500, this leads to very substantial improvements in coefficient sensitivity and hence design overhead [44, 33]. This effect is not seen in conventional DSP designs based on the shift operator which are very susceptible to coefficient round-off error. For every fixed point multiplication that exists in a filter implementation, the product must be rounded to a fixed bitwidth which creates a source of noise in the filter structure. The culmination of all round-off noise sources creates a significant increase in the output noise floor of the filter, especially as the order of the filter increases. For this reason, high order filters in practice are implemented as a series of second and first order sections so the multiplier quantization noise does not obfuscate the actual signal information [34].

The δ operator as a basis for filter implementation is key in the construction of bit-stream filters because it effectively filters the associated high frequency representation noise and attenuates it in branch nodes of filter architectures. The properties of the delta operator make it a suitable basis for construction in oversampled bit-serial filter designs. By taking great care

in the design of bitstream filters, it is not only possible to achieve low complexity and power and high performance designs, but it also provides an opportunity to solve problems that are not readily addressable with conventional DSP.

3.1.1 Shift-based Bitstream Design

In an attempt to create filters that have bitstream input and outputs, it will necessitate a ΣΔ modulator in loop on the output of the implementation. The shift based ΣΔ filter architecture shown in figure 3.2 illustrates a naive attempt to create multiplierless filter that directly processes bitstreams.

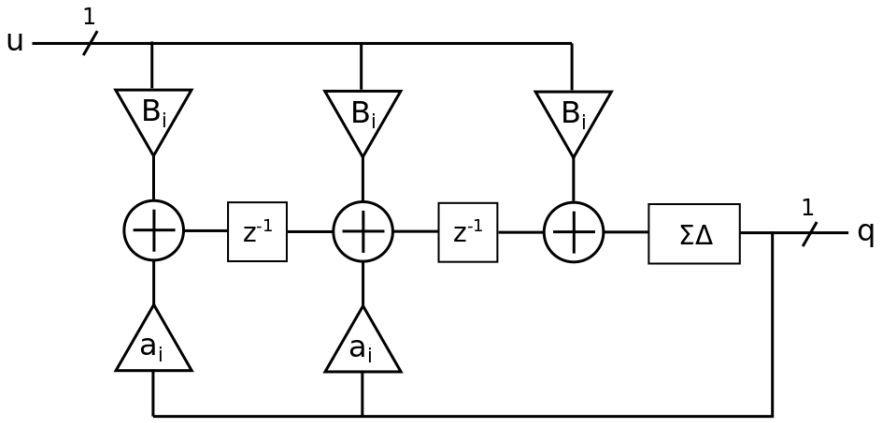


Figure 3.2: Shift Based ΣΔ Filter Architecture

Based on the direct form II transposed filter topology, the structure has the nice feature that each coefficient multiply points into the filter allowing for a single feedforward and feedback path. By placing a ΣΔ encoder in the loop, the filter can achieve a multiplierless design via single feedforward and feedback single bit signals. Unfortunately, the ΣΔ encoder introduces its own dynamics into the filter which cause undesirable effects.

Using the linear model of the ΣΔ modulator the transfer function for the second order shift based ΣΔ filter can be written as

$$\frac{Y(z)}{U(z)} = \frac{\left(b_0 + b_1 z^{-1} + b_2 z^{-2}\right) STF_{\Sigma\Delta}(z)}{1 + \left(a_1 z^{-1} + a_2 z^{-2}\right) STF_{\Sigma\Delta}(z)}$$

As can be seen, the signal transfer function shows up in the denominator. Suppose that the signal transfer function is that of a general second order modulator with $STF_{\Sigma\Delta}(z) = z^{-1}$. The introduction of a delay in the signal path adds phase delay and an extra pole in the denominator creating significant distortion in the transfer function response and in many cases instability. As an example, consider the case when $a_1 = 1.6$ and $a_2 = 0.8$. For the discrete controller, these coefficient values correspond to poles of $0.8 \pm 0.4j$. By placing a $\Sigma\Delta$ modulator in the loop with a single delay signal transfer function, the poles of the filter become $0.2225 \pm 1.3223j$ and 0.4449 which are a significant departure from the desired poles and render the filter unstable. Coupled with the numerical stability issue of fast sampling and the extra pole added by the output modulator, shift based versions of a $\Sigma\Delta$ filter are destined to fail.

3.1.2 Direct State-Space Filter

On the other hand, there is the δ operator which eliminates the numerical stability issues of its shift counterparts. One could implement a state space direct form of a filter and add discrete $\Sigma\Delta$ modulators in appropriate places where necessary to create a multiplier free design. Such a design would look like something in figure 3.3.

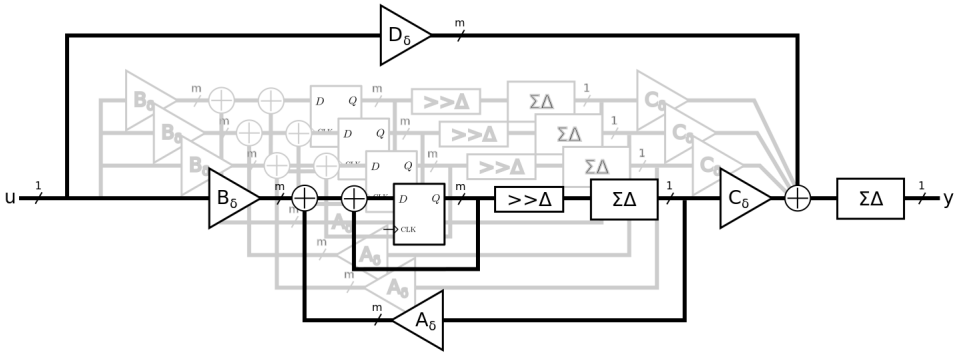


Figure 3.3: $\Sigma\Delta$ State Space Filter Architecture

The issue with the state space direct form is that it requires N+1 discrete $\Sigma\Delta$ modulators for each state variable with N being the order of the filter. More modulators means more complexity as well as more noise which could substantially reduce the effectiveness of the design SNR performance. Fortunately there is a canonical filter topology that serves as a great

compromise between performance and complexity and is based on a modified δ operator direct form II transposed structure.

3.1.3 Assumptions on Filter/Controller Architecture

In preparation for design and of the $\Sigma\Delta$ based filter/controller hardware platform, there are a variety of assumption about the conditions of the environment it is to accommodate. The assumptions on the controller are the following:

1. The controller is a single-input single-output, linear time-invariant, and minimal (being both observable and controllable) state space realization or proper rational transfer function.

2. Second order $\Sigma\Delta$ encodings are used in all bitstream paths and have a noise transfer function of

$$NTF_{\Sigma\Delta}(f) = \left(1 - e^{-j2\pi f}\right)^2$$

   where f is frequency in Hertz and $j = \sqrt{-1}$.

3. The input to the filter/controller is bounded not only in magnitude but also in frequency.

   (a) An input u to the controller is scaled such that

$$u(t) \in [-0.9,\, 0.9],\ \forall t$$

   This is to avoid distortion due to code noise discussed in chapter 2.

   (b) The input to any $\Sigma\Delta$ modulator must be band limited such that the gain crossover frequency of the closed loop is less than the control bandwidth f_B of the $\Sigma\Delta$ encoded bitstream. This is to ensure stability of the in loop $\Sigma\Delta$ modulators.

With these assumptions, it can be ensured the filter will operator in the desirable way without running into any edge constraints that would compromise stability and performance.

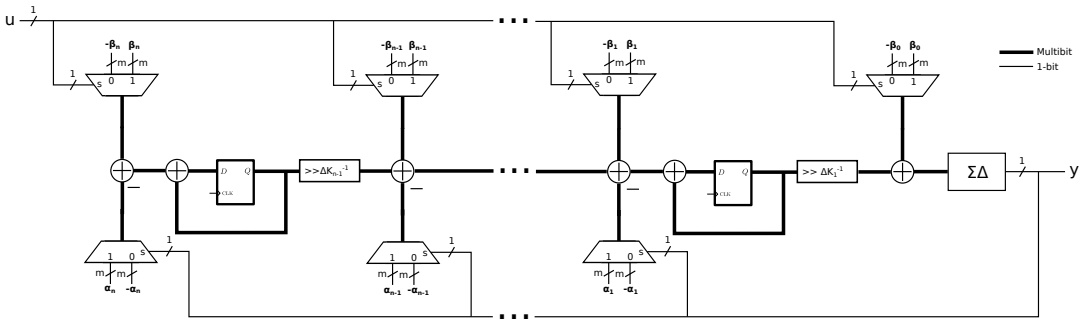


Figure 3.4: Sigma Delta IIR Filter

3.2 ΣΔ Filter Architecture

This section describes the discrete hardware architecture of the ΣΔ filter. A register transfer level diagram of the filter architecture can be seen in figure 3.4. This structure is very similar to the δ direct form II transposed structure presented in [52] with the main difference being that there now exists a discrete ΣΔ modulator after the filter output and before the feedback path. Although not shown, the input to the filter structure is also a ΣΔ encoded bitstream typically supplied by a front end ΣΔ ADC making both the input and feedback paths one bit wide.

Having both a 1-bit feedforward and feedback path has the advantage of allowing one to use multiplexers to implement the filter coefficient gains rather than using comparatively large 2's complement digital multipliers. In this case, the coefficients switch back and forth, according to the driving bitstream, between their positive and negative values. The output of a multiplexer implemented coefficient in conjunction with the dithering effect of the ΣΔ bitstream create, on average, the same low frequency product that would be obtainable with a conventional fixed point multiply operating at Nyquist sampling rates. Without the need of multipliers the entirety of the ΣΔ filter architecture can be constructed from just multiplexers, adders, registers, and bitshifts which includes the output ΣΔ modulator. Multiplierless filter design thus has an additional advantage of being considerably more compact in terms of circuit resources than their fixed point counterparts.

At the heart of the $\Sigma\Delta$ filter architecture is the filter node shown in figure 3.5. This is a point in the circuit in which corresponding coefficient scalings are added to the bit-shifted output of the previous node and accumulated in a discrete integrator. The inverse of the δ operator, which forms the basis for the model design, is implemented via the discrete integrator and a bitshift. In this paper, the scaled δ operator $\Delta\tilde{k}_{n-1}^{-1}$ gain will only be considered to occur after the accumulator. Since the coefficient gains, the output of the previous node, and the accumulator feedback share the same summation node, each signal can assume the same quantization step which sets the number of fractional bits in the representation. Choosing the proper number of fractional bits for each filter node will be discussed later in the chapter.

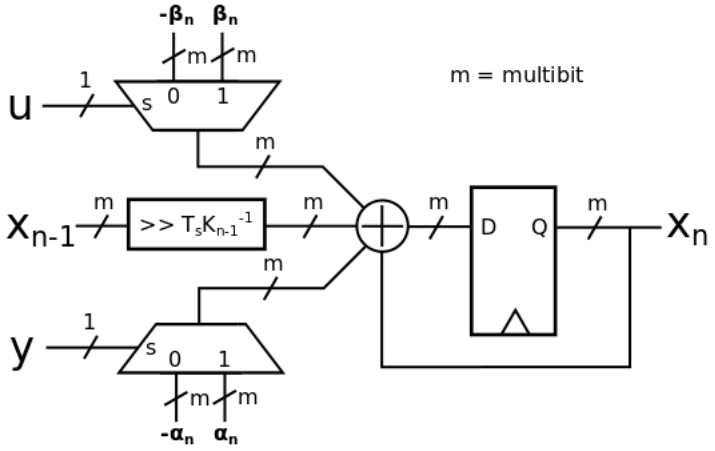


Figure 3.5: Sigma Delta Filter Node

It should also be noted that multiplexer switched filter coefficient gains do not require any rounding and truncation scheme associated with the product as with conventional bit parallel multipliers. The bitstream is simply scaled by the coefficients and the multiplexer multibit outputs are the same size as the coefficients themselves. This is a very important advantage to having 1-bit feedforward and feedback paths due to the fact the coefficient multiply truncation and rounding is eliminated. In conventional filtering, high order sections of filters are not used due in part to the accumulation of coefficient round-off noise. The problem is exacerbated

in shift operator based design while delta operator design are significantly more insensitive to coefficient round-off noise [53, 51, 39, 33]. As we shall see in Chapter 5, these problems are mostly eliminated in this architecture.

3.3 ΣΔ Filter Design and Performance Metrics

The ΣΔ filter architecture presented in the previous section has the capability of implementing nearly any δ-domain transfer function. In this section, the process of adapting a continuous time filter model to map into the ΣΔ filter structure and apply appropriate scaling is discussed.

3.3.1 Design by Emulation

Design by emulation is a popular technique in which a discrete filter is formulated via the discrete transformation of a continuous time filter design. Typically, controllers and filters are designed in the continuous domain and thus the design of a ΣΔ filter will begin with a continuous representation. The synthesized continuous time filter/controller can be described by the state space model

$$\dot{x} = A_c x + B_c u$$
$$y = C_c x + D_c u$$

where $x \in \mathbb{R}^n$, $y \in \mathbb{R}$, $u \in \mathbb{R}$, $A_c \in \mathbb{R}^{n \times n}$, $B_c \in \mathbb{R}^n$, $C_c \in \mathbb{R}^n$, $D_c \in \mathbb{R}$, and $m, n \in \mathbb{Z}^+$. The corresponding δ based model

$$\delta x = A_\delta x + B_\delta u$$
$$y = C_\delta x + D_\delta u$$

can be realized by using the relations

$$
\begin{aligned}
A_\delta &= \frac{e^{A_c\Delta} - I}{\Delta} \\
B_\delta &= \frac{1}{\Delta}\int_0^\Delta e^{A_c(t-\tau)} B_c u(\tau)\, d\tau \\
C_\delta &= C_c \\
D_\delta &= D_c
\end{aligned}
$$

where Δ is the sampling period. The discrete δ based transfer function $H_\delta(\delta)$, which emulates the input-output dynamics of the original continuous filter state spaced model, is constructed from the state space model as

$$
H_\delta(\delta) = C_\delta(\delta I - A_\delta)^{-1} B_\delta + D_\delta \;\; = \frac{N(\delta)}{D(\delta)}
$$

where

$$
\begin{aligned}
N(\delta) &= \beta_{\delta 0} + \beta_{\delta 1}\delta^{-1} + \cdots + \beta_{\delta n-1}\delta^{-(n-1)} + \beta_{\delta n}\delta^{-n} \\
&= \sum_{i=0}^{N} \beta_{\delta i}\delta^{-i}
\end{aligned}
$$

$$
\begin{aligned}
D(\delta) &= 1 + \alpha_{\delta 1}\delta^{-1} + \cdots + \alpha_{\delta n-1}\delta^{-(n-1)} + \alpha_{\delta n}\delta^{-n} \\
&= 1 + \sum_{i=1}^{N} \alpha_{\delta i}\delta^{-i}
\end{aligned}
$$

It should be noted here that the choice in Δ depends on the input bitstream and the amount of information content it contains. As previously discussed, the sample rate directly effects the effective number of bits contained in the bitstream encoding over a given bandwidth. Typically Δ will be set by a front end ADC or transducer to achieve the desired SNR/ENOB bitstream performance as described in chapter 2.

While the δ based state space model represents the correct dynamics of the filter, the matrices need to be altered in order to fit the filter model to the architecture layout of the $\Sigma\Delta$ filter. The process of mapping the original state space model can be done with the structural transformation similarity transform T_0 given by

$$T_0 = \begin{bmatrix} A_\delta^{n-1}T_1 & A_\delta^{n-2}T_1 & \dots & A_\delta T_1 & T_1 \end{bmatrix}$$

where

$$T_1 = \begin{bmatrix} C_\delta \\ C_\delta A_\delta \\ \vdots \\ C_\delta A_\delta^{n-1} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

as derived in [52]. Using the similarity transform T_0, the state space matrices of the δ model become$\tilde{A_\delta} = T_0^{-1}A_\delta T_0$, $\tilde{B_\delta} = T_0^{-1}B_\delta$, and $\tilde{C_\delta} = C_\delta T_0$. The $\sim$ indicates the application of the structural transformation to the filter model. As the name implies, the similarity transform converts the state space equations to those that describe the filter structure but leaves the overall filter dynamics unchanged, that is

$$H(\delta) = \tilde{C_\delta}(\delta I - \tilde{A_\delta})\tilde{B_\delta} + D_\delta$$

The structural transformation of the state space representation is important as it will allow the upcoming analysis of the internal filter dynamics at specific nodes in the filter.

3.3.2 Dynamic Range Scaling of State Variable Integrators

Within the architecture of the $\Sigma\Delta$ filter are the state variable integrators that implement the δ^{-1} operation that the dynamics are based upon. Given an arbitrary set of transfer function coefficients, it may well be the case that the transfer function from the input to the *ith* state variable x_i is poorly scaled. In the scenario that the scaling of a state variable integrator

is sufficiently small, the relevant signal information will be muddled with the noise floor. To maximize the SNR at each filter node, the internal integrators must be scaled appropriately.

By introducing scaling coefficients $k_1$through k_N that succeed the discrete δ^{-1} operators, the filter can be scaled at the output of the discrete integrators. The scaling matrix T_S defined as

$$T_s = diag\left[k_1^{-1}, (k_1k_2)^{-1}, \ldots, (k_1k_2 \ldots k_n)^{-1}\right]$$

can be introduced to augment the overall state space matrices as $\tilde{A_\delta'} = T_s^{-1}T_0^{-1}A_\delta T_0 T_s$, $\tilde{B_\delta'} = T_s^{-1}T_0^{-1}B_\delta$, and $\tilde{C_\delta'} = C_\delta T_0 T_s$. Here the ' indicates that scaling has been applied. As a result of the scaling, the filter coefficients become $\alpha_{\delta i} = \alpha_i k_1 k_2 \cdots k_n$ and $\beta_{\delta i} = \beta_i k_1 k_2 \cdots k_n$. The overall filter transfer function

$$H(\delta) = \tilde{C_\delta'}(\delta I - \tilde{A_\delta'})\tilde{B_\delta'} + D_\delta$$

using the transformed and scaled matrices contains the same dynamics as the non-transformed and non-scaled transfer function. While the input and output dynamics remain the same, the internal dynamics change substantially once the scaling coefficients are chosen.

Choosing the scaling coefficients begins with defining the transfer function from the input u to the *ith* integrator output x_i. Letting

$$f_i(\delta) = \frac{x_i(\delta)}{u(\delta)}$$

the transfer functions from the input to the *ith* state integrator can be written as

$$\begin{aligned} f(\delta) &= [f_1(\delta) \cdots f_n(\delta)]^T \\ &= \left(\delta I - \tilde{A}'_\delta\right)^{-1} \tilde{B}'_\delta \\ &= \left(\delta I - T_s^{-1}T_0^{-1}A_\delta T_0 T_s\right)^{-1} T_s^{-1}T_0^{-1}B_\delta \\ &= \left(T_s^{-1}T_0^{-1}\left(\delta T_0 T_s - A_\delta T_0 T_s\right)\right)^{-1} T_s^{-1}T_0^{-1}B_\delta \\ &= \left(\left(\delta I - A_\delta\right)T_0 T_s\right)^{-1} B_\delta \\ &= T_s^{-1}T_0^{-1}\left(\delta I - A_\delta\right)^{-1} B_\delta \end{aligned}$$

From here, a proper measure of the $f(\delta)$ gain is required to make an appropriate choice for k_1 through k_N. The p-norm for discrete δ based filters defined as

$$\left\| H(\frac{e^{j\omega}-1}{\Delta}) \right\|_p = \left[\frac{1}{2\pi} \int_{-\pi}^{\pi} |H\left(\frac{e^{j\omega}-1}{\Delta}\right)|^p d\omega \right]^{1/p}$$

is the measure that will be used to determine the internal gain of the filter structure. For this paper, the ∞-norm will be used for scaling which will ensure that the fixed point filter integrators do not overflow. To utilize the full dynamic range of each variable and maximize its SNR, the following relation should be made

$$\begin{aligned} ||f(\delta)||_\infty &= T_s^{-1}||T_0^{-1}\left(\delta I - A_\delta\right)^{-1} B_\delta||_\infty \\ &= \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T \end{aligned}$$

such that each integrator node is normalized. The scaling coefficients in T_s can then be found using back substitution.

In implementing the scaling coefficients in the filter structure, it is advantageous to continue the multiplierless theme of the architecture. Rather than fixed point multiplies, the scaling coefficients can be chosen as a power of two multiply/divide (i.e. a bit shift). Since

the magnitude of the signal is constant after each multiplexer coefficient (it merely switches between its positive and negative value based on the $\Sigma\Delta$ bitstream) we can scale the input from the previous integrator stage so that the ∞-norm is equal to the max value of the summed coefficients to maximize dynamic range at each signal node. To reduce the overall filter architecture complexity, the scaling coefficients should be adjusted according to

$$\tilde{k}_i = \frac{2^{\lfloor log_2 \Delta \cdot k_i^{-1} \rfloor}}{\Delta}$$

Doing so will render the $\Delta \cdot k_i^{-1}$ multiplication after the state registers a power of two. The multiplication can thus be implemented with a bit shift. Rounding after each scaling shift will be assumed from here on. At this point, the filter coefficients are completely described with the appropriate scaling and can be used in further analysis to establish noise gain and register and coefficient bitwidth selection for the modified $\Sigma\Delta$ filter architecture.

3.3.3 Coefficient Sensitivity

To construct a fixed point filter, it becomes necessary to quantize the filter coefficients. For two's compliment representations the coefficients will have a bitwidth $b_w = b_i + b_f + 1$ where b_i is the integer bits and b_f are the fractional bits. Doing so however, introduces deviations into the transfer function response. To choose the correct quantization of filter coefficients in order to bound the induced transfer function error, the filter sensitivity must be analyzed.

To choose the coefficient bitwidths, the the sensitivity of the transfer function must be evaluated for small deviations in the filter coefficients in order to bound the induced variation. Let the sensitivity of a transfer function $G(\delta)$ with respect to a parameter m be defined as

$$S_m^{G(\delta)}(\delta) = \frac{\partial G(\delta)}{\partial m}$$

Using the sensitivity measure $S_m^{G(\delta)}(\delta)$, an estimate for the transfer function response deviation $\epsilon_m(\delta)$ with respect to a parameter deviation δ_m takes the form of $\epsilon_m(\delta) = \delta_m S_m^{G(\delta)}(\delta)$.

Applying the previously defined sensitivity measure to the feedback and feedforward coefficients $\tilde{\alpha}$ and $\tilde{\beta}$, the sensitivity functions with respect to the ith element of $\tilde{\alpha}$ and $\tilde{\beta}$ coefficients can be written as

$$
\begin{aligned}
S_{\alpha_i}^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) &= \frac{\partial \tilde{H}'_{\Sigma\Delta}(\delta)}{\partial \alpha_i} \\
&= \tilde{H}'_{\Sigma\Delta}(\delta) \cdot \left[T_s^T T_0^T \left(\delta I - A_\delta^T \right)^{-1} C_\delta^T \right] \\
S_{\beta_i}^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) &= \frac{\partial \tilde{H}'_{\Sigma\Delta}(\delta)}{\partial \beta_i} \\
&= T_s^T T_0^T \left(\delta I - A_\delta^T \right)^{-1} C_\delta^T \\
S_{\beta_0}^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) &= \frac{\partial \tilde{H}'_{\Sigma\Delta}(\delta)}{\partial \beta_0} \\
&= C_\delta \left(\delta I - A_\delta \right)^{-1} A_\delta T_0 \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T + 1
\end{aligned}
$$

As can be seen in the filter node diagram in figure 3.5, the α_i and β_i coefficient multiplexers see the same summation node. It is appropriate then that both coefficients are quantized to the same level. By combing the coefficient sensitivities at each node, a sensitivity matrix, $S^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta)$, can be formulated as

$$S^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) = \begin{bmatrix} S_0^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) \\ S_1^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) \\ S_2^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) \\ \vdots \\ S_N^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) \end{bmatrix}^T$$

$$= \begin{bmatrix} S_{\beta_0}^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) \\ S_{\beta_1}^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) + S_{\alpha_1}^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) \\ S_{\beta_2}^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) + S_{\alpha_2}^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) \\ \vdots \\ S_{\beta_N}^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) + S_{\alpha_N}^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) \end{bmatrix}^T$$

At this point it is of interest to find the variation of the ideal transfer function magnitude, $\Delta\left|\tilde{H}'_{\Sigma\Delta}(e^{j\omega})\right|$, with respect to Δ_{qi} changes in the *ith* numerator and denominator coefficients. The overall magnitude of the transfer function sensitivity $S_i^{\tilde{H}'_{\Sigma\Delta}(\delta)}\left(e^{j\omega}\right)$ is

$$\left|S_i^{\tilde{H}'_{\Sigma\Delta}(\delta)}\left(e^{j\omega}\right)\right| = \left|S_{\beta_i}^{\left|\tilde{H}'_{\Sigma\Delta}(e^{j\omega})\right|}(e^{j\omega}) + S_{\alpha_i}^{\left|\tilde{H}'_{\Sigma\Delta}(e^{j\omega})\right|}(e^{j\omega})\right|$$

$$\leq \left|S_{\beta_i}^{\tilde{H}'_{\Sigma\Delta}(e^{j\omega})}(e^{j\omega}) + S_{\alpha_i}^{\tilde{H}'_{\Sigma\Delta}(e^{j\omega})}(e^{j\omega})\right|$$

The right hand side of the above inequality can be used as a conservative estimate for $\left|S_i^{\tilde{H}'_{\Sigma\Delta}(\delta)}\left(e^{j\omega}\right)\right|$. An approximation to the transfer function variation can then be computed by

$$\Delta\left|\tilde{H}'_{\Sigma\Delta}(\delta)\right| \leq \Delta_{q0}\left|S_{\beta_0}^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta)\right| + \sum_{i=1}^{N}\Delta_{qi}\left[\left|S_{\beta_i}^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) + S_{\alpha_i}^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta)\right|\right]$$

or more compactly as

$$\Delta \left| \tilde{H}'_{\Sigma\Delta}(\delta) \right| \leq \left| S^{\tilde{H}'_{\Sigma\Delta}(\delta)}(\delta) \right| \Delta_q$$

where $\Delta_q = \begin{bmatrix} \Delta_{q_0} & \Delta_{q_1} & \cdots & \Delta_{q_N} \end{bmatrix}^T$ is a vector quantization levels at each filter node. The transfer function magnitude variation, $\Delta \left| \tilde{H}'_{\Sigma\Delta}(\delta) \right|$, due to coefficient quantization will later be used as a performance metric for choosing the appropriate quantization levels of the filter nodes with the overall $\Sigma\Delta$ filter architecture.

3.3.4 Noise Analysis

The noise analysis of the $\Sigma\Delta$ filter differs from conventional analysis in two distinct ways. First, there is the inherent representation noise associated with the input and output bitstreams that are injected into the filter. Second, there are no fixed point coefficient multiplier products that need truncation. For the $\Sigma\Delta$ filter, noise is injected into the circuit that propagates to the output from three separate sources:

1. The representation noise of the input $\Sigma\Delta$ encoded bitstream to the filter output.
2. The additive representation noise of the output $\Sigma\Delta$ modulator to the filter output.
3. The coupling coefficient round-off quantization taking place after each discrete integrator.

As previously stated, there is no round-off quantization noise due to coefficient multiplies since no fixed point multipliers exist in the filter architecture; scaling coefficients are selected as power of two bitshifts. It is possible then to create third order or higher filters without coefficient multiply round-off quantization noise drowning out the signal information at each stage of the filter.

3.3.4.1 Input $\Sigma\Delta$ Representation Noise Propagation

Representation noise from the input $\Sigma\Delta$ follows the same path as the input signal to the filter output. That is, the input representation noise is filtered by the transfer function $\tilde{H}'_{\Sigma\Delta}(\delta)$. Let $NTF_{\Sigma\Delta_1}(\delta)$ be the noise transfer function of the input modulator where the

quantizer noise has a spectral density of $\eta_{\Sigma\Delta_1}$ and f be frequency in Hertz. The output noise variance injected by the input $\Sigma\Delta$ modulator, $\sigma^2_{\Sigma\Delta_1}$, is given by

$$\begin{aligned}
\sigma^2_{\Sigma\Delta_1} &= \int_{-f_B}^{f_B} \eta_{\Sigma\Delta_1} \left| NTF_{\Sigma\Delta_1}(f) \left(\tilde{H}'_{\Sigma\Delta}(f) + \Delta\tilde{H}'_{\Sigma\Delta}(f) \right) \right|^2 df \\
&\leq \int_{-f_B}^{f_B} \eta_{\Sigma\Delta_1} \left[\left| NTF_{\Sigma\Delta_1}(f)\tilde{H}'_{\Sigma\Delta}(f) \right|^2 + \left| NTF_{\Sigma\Delta_1}(f)\Delta\tilde{H}'_{\Sigma\Delta}(f) \right|^2 \right] df \\
&= \int_{-f_B}^{f_B} \eta_{\Sigma\Delta_1} \left[\tilde{H}'^*_{\Sigma\Delta}(f) NTF^*_{\Sigma\Delta_1}(f) NTF_{\Sigma\Delta_1}(f)\tilde{H}'_{\Sigma\Delta}(f) \right] df \\
&+ \Delta \left[\int_{-f_B}^{f_B} \eta_{\Sigma\Delta_1} \left[S^*(f)\, NTF^*_{\Sigma\Delta_1}(f) NTF_{\Sigma\Delta_1}(f) S(f) \right] df \right] \Delta_q
\end{aligned}$$

Note here that the transfer function deviation, $\Delta\tilde{H}'_{\Sigma\Delta}(f)$, is taken into account when quantizing the internal bitwidths of the filter. The overall output variance contributed by the input bitstream representation noise is

$$\sigma^2_{\Sigma\Delta_1} = \sigma^2_{nom_1} + \Delta_q^T H_{\Sigma\Delta_1} \Delta_q$$

where

$$\sigma^2_{nom_1} = \eta_{\Sigma\Delta} \int_{-f_B}^{f_B} \tilde{H}'^*_{\Sigma\Delta}(f) NTF^*_{\Sigma\Delta_1}(f) NTF_{\Sigma\Delta_1}(f)\tilde{H}'_{\Sigma\Delta}(f) df$$

and

$$H_{\Sigma\Delta_1} = \eta_{\Sigma\Delta} \int_{-f_B}^{f_B} S^*(f)\, NTF^*_{\Sigma\Delta_1}(f) NTF_{\Sigma\Delta_1}(f) S(f)\, df$$

3.3.4.2 Output $\Sigma\Delta$ Representation Noise Propagation

Representation noise from the output $\Sigma\Delta$ is injected at the quantizer noise input of the digital output modulator to the filter output. The transfer function, $E(\delta)$, from the output

$\Sigma\Delta$ quantizer noise input to the output of the filter is

$$\begin{aligned}
E(\delta) &= (\tilde{A}'_\delta, A_o \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T, \tilde{C}'_\delta, 1)_\delta \\
&= C_\delta T_0 T_s \left(\delta I - T_s^{-1} T_0^{-1} A_\delta T_0 T_s\right)^{-1} \times \\
&T_s^{-1} T_0^{-1} A_\delta T_0 T_s K \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T + 1 \\
&= C_\delta T_0 T_s \left[T_s^{-1} T_0^{-1} \left(\delta I T_0 T_s - A_\delta T_0 T_s\right)^{-1}\right] \times \\
&T_s^{-1} T_0^{-1} A_\delta T_0 \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T + 1 \\
&= C_\delta \left(\delta I - A_\delta\right)^{-1} A_\delta T_0 \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T + 1
\end{aligned}$$

where $A_o = \tilde{A}'_\delta K$ and $K = diag\begin{pmatrix} \tilde{k}_1 & \tilde{k}_2 & \cdots & \tilde{k}_N \end{pmatrix}$. As a result of coefficient quantization, $E(\delta)$ will vary slightly from its ideal formulation. The sensitivity of $E(\delta)$ must be taken into account then when calculating the output noise variance estimate.

The sensitivity function $S^{E(\delta)}(\delta)$ of $E(\delta)$ can be defined as

$$S^{E(\delta)}(\delta) = \begin{bmatrix} S_0^{E(\delta)}(\delta) \\ S_1^{E(\delta)}(\delta) \\ S_2^{E(\delta)}(\delta) \\ \vdots \\ S_N^{E(\delta)}(\delta) \end{bmatrix}^T = \begin{bmatrix} 0 \\ S_{\alpha_1}^{E(\delta)}(\delta) \\ S_{\alpha_2}^{E(\delta)}(\delta) \\ \vdots \\ S_{\alpha_N}^{E(\delta)}(\delta) \end{bmatrix}^T$$

where

$$S_{\alpha_i}^{E(\delta)}(\delta) \quad = \quad \frac{\partial E(\delta)}{\partial \alpha_i} = \frac{\delta^{-i} E(\delta)}{D(\delta)}$$

The transfer function magnitude deviation of $E(\delta)$ can then be written as

$$\Delta |E(\delta)| \leq \sum_{i=1}^{N} \Delta_i \left| S_{\alpha_i}^{E(\delta)}(\delta) \right|$$

Let $NTF_{\Sigma\Delta_2}(\delta)$ be the noise transfer function of the output modulator where the quantizer noise has a spectral density of $\eta_{\Sigma\Delta_2}$. The output noise variance injected by the input $\Sigma\Delta$ modulator, $\sigma^2_{\Sigma\Delta_1}$, is given by

$$\begin{aligned}
\sigma^2_{\Sigma\Delta_2} &= \int_{-f_B}^{f_B} \eta_{\Sigma\Delta} \left| NTF_{\Sigma\Delta_2}(f) \left(E(f) + \Delta E(f) \right) \right|^2 df \\
&\leq \int_{-f_B}^{f_B} \eta_{\Sigma\Delta} \left[\left| NTF_{\Sigma\Delta_2}(f) E(f) \right|^2 + \left| NTF_{\Sigma\Delta_2}(f) \Delta E(f) \right|^2 \right] df \\
&= \int_{-f_B}^{f_B} \eta_{\Sigma\Delta} \left[E^*(f) NTF^*_{\Sigma\Delta_2}(f) NTF_{\Sigma\Delta_2}(f) E(f) \right] df \\
&\quad + \Delta_q^T \left[\int_{-f_B}^{f_B} \eta_{\Sigma\Delta} \left[S^{E(\delta)*}(f) NTF^*_{\Sigma\Delta_2}(f) NTF_{\Sigma\Delta_2}(f) S^{E(\delta)}(\delta) \right] df \right] \Delta_q
\end{aligned}$$

or more compactly as

$$\sigma^2_{\Sigma\Delta_2} = \sigma^2_{nom_2} + \Delta_q^T H_{\Sigma\Delta_2} \Delta_q$$

where

$$\sigma^2_{nom_2} = \eta_{\Sigma\Delta} \int_{-f_B}^{f_B} E^*(f) NTF^*_{\Sigma\Delta_2}(f) NTF_{\Sigma\Delta_2}(f) E(f) df$$

and

$$H_{\Sigma\Delta_2} = \eta_{\Sigma\Delta} \int_{-f_B}^{f_B} S^{E(\delta)*}(f) NTF^*_{\Sigma\Delta_2}(f) NTF_{\Sigma\Delta_2}(f) S^{E(\delta)}(\delta) df$$

3.3.4.3 Output Noise Propagation due to Scaling Coefficient Rounding

The third source of noise in the $\Sigma\Delta$ filter structure is from the rounding that occurs directly after the scaling coefficient bitshift implementations. Again, a rounding scheme is

assumed in order to maintain a zero bias. It is also assumed that the rounding noise can be modeled as an additive white Gaussian source. Under that assumption, the spectral density of the rounding noise is

$$\eta_{k_i}(f) = \frac{\sigma_e^2}{f_s} = \frac{\Delta_{qi}^2}{12 \cdot f_s}$$

where σ_e^2 is the noise variance of the additive rounding noise source.

To begin finding the total noise contribution from the scaling coefficient rounding, the transfer functions from the additive rounding noise inputs $e_{k_1}$to the output y must be found. Taking note of the fact that the noise input for k_1 shares the same summation node as that of β_0,the transfer function $g_0(\delta) = \frac{y(\delta)}{e_{k_1(\delta)}}$ can be found as

$$\begin{aligned}
g_0(\delta) &= (\tilde{A_\delta'}, A_o \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T, \tilde{C_\delta'}, 1)_\delta \\
&= C_\delta T_0 T_s \left(\delta I - T_s^{-1} T_0^{-1} A_\delta T_0 T_s\right)^{-1} \times \\
&T_s^{-1} T_0^{-1} A_\delta T_0 T_s K \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T + 1 \\
&= C_\delta T_0 T_s \left[T_s^{-1} T_0^{-1} \left(\delta I T_0 T_s - A_\delta T_0 T_s\right)^{-1}\right] \times \\
&T_s^{-1} T_0^{-1} A_\delta T_0 \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T + 1 \\
&= C_\delta \left(\delta I - A_\delta\right)^{-1} A_\delta T_0 \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T + 1
\end{aligned}$$

The output noise variance $\sigma_{k_1}^2$ contributed by the rounding of the k_1 scaling coefficient can then be found as

$$\sigma_{k_1}^2 = \int_{-f_B}^{f_B} \eta_{e_{k_1}}(f) \left|g_0(f)\right|^2 df$$

Similarly, the transfer function from $e_{k_i}$to the output y shares the same summation node as $\tilde{\alpha}_i$ and $\tilde{\beta}_i$. The transfer function $g_i(\delta) = \frac{y(\delta)}{e_{k_i(\delta)}}$ can be found as

$$\begin{aligned}
g(\delta) &= \begin{bmatrix} g_1(\delta) & \cdots & g_N(\delta) \end{bmatrix}^T \\
&= \left[\tilde{C}'_\delta \left(\delta I - \tilde{A}'_\delta\right)^{-1}\right]^T \\
&= \left[C_\delta T_0 T_s \left(\left(\delta T_s^{-1} T_0^{-1} I - T_s^{-1} T_0^{-1} A_\delta\right) T_0 T_s\right)^{-1}\right]^T \\
&= \left[C_\delta \left(\delta T_s^{-1} T_0^{-1} I - T_s^{-1} T_0^{-1} A_\delta\right)^{-1}\right]^T \\
&= \left[C_\delta \left(T_s^{-1} T_0^{-1} (\delta I - A_\delta)\right)^{-1}\right]^T \\
&= \left[C_\delta (\delta I - A_\delta)^{-1} T_0 T_s\right]^T \\
&= T_s^T T_0^T \left(\delta I - A_\delta^T\right)^{-1} C_\delta^T
\end{aligned}$$

where the total output noise variance $\sigma^2_{k_i{}'}$ for the ith scaling coefficient rounding can written as

$$\sigma^2_{k_i{}'} = \int_{-f_B}^{f_B} \eta_{e_{k_i}}(f) \left|g_{i-1}(f)\right|^2 df$$

Taking all rounding contributions into consideration, the total output variance from scaling coefficient rounding σ_k^2 can be written as

$$\begin{aligned}
\sigma_k^2 &= \Delta_q^T \left[\int_{-f_B}^{f_B} \left(\frac{1}{3f_s}\right) g_k^*(\delta)\, g_k(\delta)\, df\right] \Delta_q \\
\sigma_k^2 &= \Delta_q^T H_k \Delta_q
\end{aligned}$$

where

$$g_k(\delta) = \begin{bmatrix} g_0(\delta) & g_1(\delta) & \cdots & g_{N-1}(\delta) & 0 \end{bmatrix}^T$$

and

$$H_k = \int_{-f_B}^{f_B} \left(\frac{1}{3f_s}\right) g_k^*(\delta)\, g_k(\delta)\, df$$

The total filter output noise propagation from all noise sources can be written as:

$$\sigma_{total}^2 = \sigma_{\Sigma\Delta 1}^2 + \sigma_{\Sigma\Delta 2}^2 + \sum_{k=1}^{n} \sigma_{k_i}^2$$

3.3.4.4 Noise Floor in Integrator Sections due to ΣΔ Representation Noise

The representation noise from the input and output ΣΔ modulators is an effect that propagates itself throughout the internal nodes of the ΣΔ filter architecture. Once a ΣΔ modulator architecture and OSR have been selected, the representation noise sets a fixed noise floor in all parts of the filter. Decreasing scaling coefficient round-off noise is therefore limited to the floor set by the representation noise of the input and feedback ΣΔ modulators. Reducing the quantization level Δ_{q_i} for the fixed point representation in the *ith* filter node will increase the overall size of the filter circuitry without any reduction in overall internal filter noise. It is therefore prudent to select Δ_{q_i}based on the condition

$$\sigma_{k_i}^2 \geq \sigma_{x_{i-1}\Sigma\Delta}^2$$

where $\sigma_{k_i}^2$ is in band quantization noise variance due to scaling coefficient rounding and $\sigma_{x_{i-1}\Sigma\Delta}^2$ is the in band noise variance due to the input and feedback ΣΔ modulators at the *ith* integrator node. The in band quantization noise from the rounding that occurs after each integrator stage coupling coefficient is

$$\sigma_{k_i}^2 = \frac{\Delta_{q_i}^2}{12 \cdot OSR}$$

It is a fact that $\sigma_{x_{i-1}\Sigma\Delta}^2$ puts a limit onto how small one can make $\sigma_{k_i}^2$ through choosing the quantization step size Δ_{q_i} (i.e. $\sigma_{k_i}^2 \not< \sigma_{x_{i-1}\Sigma\Delta}^2$). The bound on the size of Δ_q can be written as

$$\sigma_{k_i}^2 \geq \sigma_{x_i \Sigma\Delta}^2$$
$$\frac{\Delta_{q_i}^2}{12 \cdot OSR} \geq \sigma_{x_i \Sigma\Delta}^2$$
$$\Delta_{q_i}^2 \geq 12 \cdot OSR \cdot \sigma_{x_i \Sigma\Delta}^2$$
$$\Delta_{q_i} > \sqrt{12 \cdot OSR \cdot \sigma_{x_i \Sigma\Delta}^2}$$

To calculate $\sigma_{x_i \Sigma\Delta}^2$, the transfer function from the input and output $\Sigma\Delta$ modulator noise input to the state variables must be derived. For the input modulator to the *ith* integrator state, the transfer function is

$$\begin{aligned}
f_{x\Sigma\Delta_1}(\delta) &= \left[f_{x_1\Sigma\Delta_1}(\delta) \cdots f_{x_n\Sigma\Delta_1}(\delta)\right]^T \\
&= \left(\delta I - \tilde{A}_\delta'\right)^{-1} \tilde{B}_\delta' \\
&= \left(\delta I - T_s^{-1} T_0^{-1} A_\delta T_0 T_s\right)^{-1} T_s^{-1} T_0^{-1} B_\delta \\
&= \left(T_s^{-1} T_0^{-1} \left(\delta T_0 T_s - A_\delta T_0 T_s\right)\right)^{-1} T_s^{-1} T_0^{-1} B_\delta \\
&= \left(\left(\delta I - A_\delta\right) T_0 T_s\right)^{-1} B_\delta \\
&= T_s^{-1} T_0^{-1} \left(\delta I - A_\delta\right)^{-1} B_\delta
\end{aligned}$$

and the transfer function from the output modulator to the *ith* integrator state can be written as

$$
\begin{aligned}
f_{x\Sigma\Delta_2}(\delta) &= \left[f_{x_1\Sigma\Delta_2}(\delta)\cdots f_{x_n\Sigma\Delta_2}(\delta)\right]^T \\
&= (\delta I - \tilde{A}'_\delta)A_O \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T \\
&= \left(\delta I - T_s^{-1}T_0^{-1}A_\delta T_0 T_s\right)^{-1} T_s^{-1}T_0^{-1}A_\delta T_0 T_s K^{-1} \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T \\
&= \left(T_s^{-1}T_0^{-1}\left(\delta T_0 T_s - A_\delta T_0 T_s\right)\right)^{-1} T_s^{-1}T_0^{-1}A_\delta T_0 T_s K^{-1} \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T \\
&= \left(\left(\delta I - A_\delta\right) T_0 T_s\right) A_\delta T_0 T_s K^{-1} \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T \\
&= T_s^{-1}T_0^{-1}\left(\delta I - A_\delta\right)^{-1} A_\delta T_0 T_s K^{-1} \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T
\end{aligned}
$$

The noise variance at each node integrator stage due to the input and output modulators with respective noise transfer functions $NTF_{\Sigma\Delta_1}(f)$ and $NTF_{\Sigma\Delta_2}(f)$ can then be calculated as

$$
\begin{aligned}
\sigma^2_{x_i\Sigma\Delta_1} &= \eta_e(f)\int_{-f_B}^{f_B} \left|f_{x_i\Sigma\Delta_1}(f) NTF_{\Sigma\Delta_1}(f)\right|^2 df \\
\sigma^2_{x_i\Sigma\Delta_2} &= \eta_e(f)\int_{-f_B}^{f_B} \left|f_{x_i\Sigma\Delta_2}(f) NTF_{\Sigma\Delta_2}(f)\right|^2 df
\end{aligned}
$$

where

$$
\eta_e(f) = \frac{\sigma_e^2}{f_s} = (3f_s)^{-1}
$$

The total noise variance at each integrator stage due to the input and output $\Sigma\Delta$ representation noise is

$$
\sigma^2_{x_i\Sigma\Delta} = \sigma^2_{x_i\Sigma\Delta_1} + \sigma^2_{x_i\Sigma\Delta_2}
$$

where

$$\sigma^2_{x\Sigma\Delta} = \begin{bmatrix} \sigma^2_{x_1\Sigma\Delta} \\ \sigma^2_{x_2\Sigma\Delta} \\ \vdots \\ \sigma^2_{x_n\Sigma\Delta} \end{bmatrix}$$

Having calculated $\sigma^2_{x\Sigma\Delta}$, the lower bound on the quantization steps of the filter will be

$$\Delta_q > \sqrt{12 \cdot OSR \cdot \sigma^2_{x\Sigma\Delta}}$$

and can be used in determining the size of the fixed point signals in the filter architecture.

3.4 Filter Bitwidth Optimization

At this stage in the $\Sigma\Delta$ filter design, the filter coefficients and signal paths of the discrete hardware must be quantized. Using the quantization noise and coefficient sensitivity metrics derived above, one could naively choose the quantization levels, Δ_q, that would meet a given noise and sensitivity constraint, but perhaps one could do better. Given a noise and sensitivity constraint, it is possible to write an optimization routine that maximizes the quantization levels so that minimum number of fractional bits are required to represent the coefficients and state register values. This has the additional advantage of taking into account the slight errors introduced in other stages by selected resolutions.

By drawing on the principals of convex optimization [24], the following optimization strategy seeks to minimize internal bitwidths of the filter architecture by maximizing the quantization levels:

$$
\begin{aligned}
&maximize\ c^T \Delta_q && (objective\ function)\\
&\sqrt{H_k}\Delta_q \leq \sqrt{\gamma_{noise} - \sigma^2_{nom_1} - \sigma^2_{nom_2}} && (quantization\ noise\ bound)\\
&\left|S^{\tilde{H}'_{\Sigma\Delta}(f_d)}(f_d)\right| \Delta_q \preceq \rho(f_d) && (sensitivity\ bound)\\
&\sqrt{12 \cdot f_B \cdot \sigma^2_{x\Sigma\Delta}} \preceq \Delta_q \preceq 1 && (\Delta_q\ range\ bound)
\end{aligned}
$$

where $c = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}$. The objective function of the optimization routine relies on the quantization noise and sensitivity bounds of the filter as well as the value limit imposed by the representation noise floor created by the input and output $\Sigma\Delta$ modulators. There are two performance bounds chosen by the designer in this model:

1. The noise bound $\gamma_{noise} \in \mathbb{R}_{\geq 0}$ which sets the upper bound on the resolution of the filter over the specified signal band.

2. The magnitude sensitivity bound $\rho(f) \in \mathbb{R}^n$ which sets an upper bound on the magnitude variation of the filter across the specified signal band.

Steps taken to construct this optimization routine require first that the frequency band of interest (typically from DC to f_B) be discretized so that a convex optimization solver can solve the problem over a discrete set of points. Creating a discretized vector of frequencies f_D will be in the form

$$
f_D = \begin{bmatrix} f_0 & f_1 & \cdots & f_M \end{bmatrix}
$$

where M is the number of samples.

Next, f_D must be used to create the sensitivity matrix $\left|S^{\tilde{H}'_{\Sigma\Delta}(f_d)}(f_D)\right|$ for use in the sensitivity bound of the filter. The resulting matrix will be of the form

$$\left|S^{\tilde{H}'_{\Sigma\Delta}(f_d)}(f)\right| = \begin{bmatrix} \left|S^{\tilde{H}'_{\Sigma\Delta}}_{01}(f_0)\right| & \left|S^{\tilde{H}'_{\Sigma\Delta}}_{02}(f_1)\right| & \cdots & \left|S^{\tilde{H}'_{\Sigma\Delta}}_{0K}(f_M)\right| \\ \left|S^{\tilde{H}'_{\Sigma\Delta}}_{11}(f_0)\right| & \left|S^{\tilde{H}'_{\Sigma\Delta}}_{12}(f_1)\right| & \cdots & \left|S^{\tilde{H}'_{\Sigma\Delta}}_{1K}(f_M)\right| \\ \left|S^{\tilde{H}'_{\Sigma\Delta}}_{21}(f_0)\right| & \left|S^{\tilde{H}'_{\Sigma\Delta}}_{22}(f_1)\right| & \cdots & \left|S^{\tilde{H}'_{\Sigma\Delta}}_{2K}(f_M)\right| \\ \vdots & \ddots & \ddots & \vdots \\ |S_{N1}(f_0)| & |S_{N2}(f_1)| & \cdots & |S_{NK}(f_M)| \end{bmatrix}$$

Now the performance bounds must be chosen by a designer. For the transfer function sensitivity, $\rho(f_D)$ must be chosen to put an upper bound on the transfer function variation that the filter design would tolerate. The bound vector will be in the vector form of

$$\rho(f_D) = \begin{bmatrix} \rho(f_0) & \rho(f_1) & \cdots & \rho(f_M) \end{bmatrix}$$

The last performance bound to chose in the output noise bound of the filter γ_{noise}. The noise bound puts an upper bound on the maximum tolerable noise level on the output of the filter over the frequency range f_D. For a required output ENOB of 14 for instance, the bound should be chosen in absolute terms to be $\gamma_{noise} = 1e-8$.

3.4.1 Choosing Filter Bitwidths

When it comes to assigning bitwidths to the interior signal paths of the $\Sigma\Delta$ filter architecture, there are two considerations that must be made: the bitwidths of the coefficients and the bitwidths of the state registers. Since the numerical representation of these values will be in two's compliment form, each value will consist of a bitwidth $B_w \in \mathbb{Z}^+$ given by

$$B = I + F + 1$$

where $I \in \mathbb{Z}^+$ are the integer bits, $F \in \mathbb{Z}^+$ are the fractional bits, and a one for the sign bit. For the bitwidths of the coefficients, it must again be pointed out that the corresponding α and β coefficients share the same summation node as shown in figure 3.6. For the number of integer bits I that should be assigned to the coefficients at the *ith* node is

$$I_i = \lceil max\left(log_2\left(\beta_{\delta i}\right), log2\left(\alpha_{\delta i}\right)\right)\rceil$$

where $\lceil\cdot\rceil$ is the ceiling operator. The number for fractional bits F at the *ith* node can be determined by

$$F_i = \begin{cases} \lceil |log2\left(\Delta_{qi}\right)| \rceil, & \Delta_{qi} < 1 \\ 0, & otherwise \end{cases}$$

where Δ_{qi} is determined by the previously described optimization routine. The *ith* coefficient bitwidth B_i can then be determined by I_i and F_i according to .

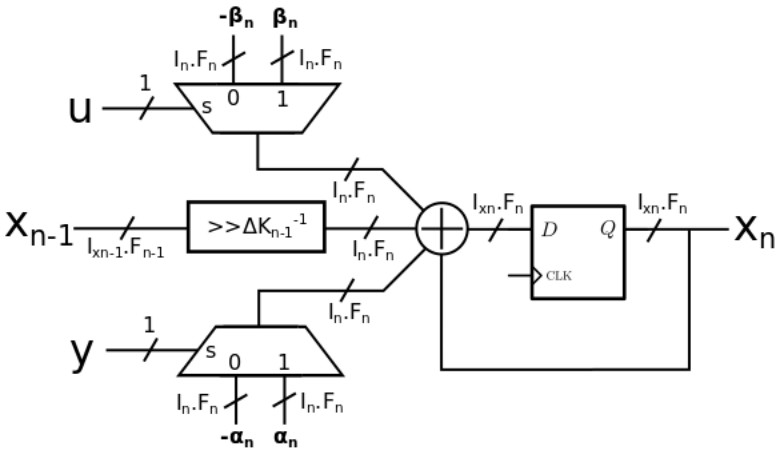


Figure 3.6: Sigma Delta Filter Node Bitwidths

In regards to setting the register bitwidths, notice again that the α, β, and previous scaled state variable all share the same summation node. It is appropriate then to set the number of fractional bits of the proceeding integrator register to F_i. The number of integer bits I_{xi} of the state register can be found by

$$I_{xi} = \lceil |log2\left(\Delta\right)| \rceil$$

leaving the total bitwidth of the ith register

$$B_{xi} = I_{xi} + F_i + 1$$

3.5 Design Example

To demonstrate the efficacy of the ΣΔ filter architecture and the bitwidth optimization strategy previously outlined, a small sample of filter designs will be simulated. Power and resource utilization will also be estimated by mapping the filter designs into a low cost Artix-7 XC7A12T-1CPG238I FPGA from Xilinx®. This FPGA was chosen as Xilinx tools support Matlab Simulink based design interfaces as well as providing direct measurement of the resources and power requirements. The design is mapped entirely into the logic fabric, none of the DSP resources are used, hence these results are applicable to very low cost FPGA's from Lattice or Actel without DSP support, but fitting very low power or cost design niches.

The design parameters of the filter are listed in table 4.1. The ΣΔ filter will be designed to emulate an eight order bandpass Chebyshev type II filter. Normally with a shift based filter, a high order filter would have to be broken into second order sections to mitigate coefficient multiply coefficient quantization but for the ΣΔ filter, this is unnecessary do to there being zero noise of this type. The filter to follow is constructed as a single element.

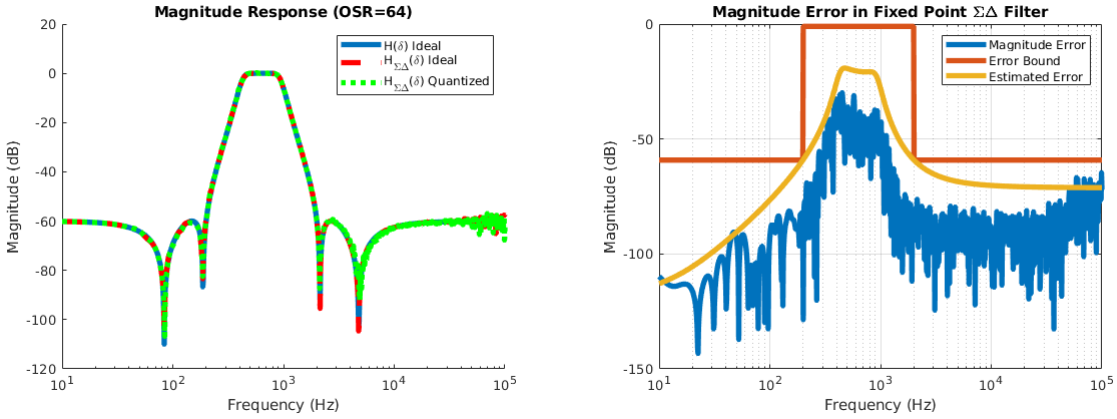
Parameter	Value
Filter Type	Bandpass Chebyshev Type II
Passband	200 - 2000 Hz
Signal Band	DC - 10 kHz
Order	8
Stopband Attenuation	60 dB

Table 3.1: Design Parameters of ΣΔ Filter

Using a white noise input stimulus, the magnitude response of the filter was estimated for design parameters of OSR equal to 64, a target ENOB of 12 ENOB in the passband and a magnitude deviation of 1 db in the passband and stopband from the ideal magnitude response. The appropriate ρ was chosen to be

$$\rho(f) = \begin{cases} 0.831 & \forall f \in [200,\ 2000] \\ 0.0011 & otherwise \end{cases}$$

to achieve the desired deviation bound and used as a constraint in the bitwidth optimization routine. The resulting magnitude and magnitude error response are shown below in figure 4.12.



(a) Magnitude Response

(b) Magnitude Error

Figure 3.7: Filter Magnitude Response ($OSR = 64$, $ENOB = 12$)

As can be seen, the magnitude response is quite faithful to the ideal magnitude curve of the filter design both for the ideal ΣΔ filter (where the simulation runs on floating point numbers) and the quantized fixed point ΣΔ filter implementation. The magnitude error plot shows that the quantized filter error is below the ρ design bound and demonstrates that the sensitivity analysis based estimated error bound is quite close to the actual measured deviation.

OSR	Design ENOB	Calculated ENOB	Power (μW) (Static/Dynamic)	LUT/FF
32	10	10.35	58/>1	796/250
64	12	12.21	58/2	850/259
128	14	14.33	58/3	902/267
256	16	16.02	58/4	957/273

Table 3.2: Filter Design Parameters

Table 4.2 lists the power and resource usage after mapping the filter into a Xilinx Artix-7 XC7A12T-1CPG238I FPGA. The XC7A12T is currently the smallest Artix-7 FPGA

that Xilinx provides. From the reported resources usage, one could easily fit multiple high order filters into the device where the power is dominated by the static power of the device. The design ENOB was also easily reached by the optimization program for each OSR value.

3.6 Previous Work

The concept of directly processing bitstreams has garnered moderate attention over the last four decades. Motivated primarily by reduced complexity embedded signal chains and smaller logic footprints, many have contributed to the growing field of bitstream processing in the areas of implementing mathematical operations for signal processing [37, 38, 67, 36, 26, 27], FIR filters [40, 32, 45], and IIR filters [65, 29, 30, 25]. The prior works regarding IIR filters being germane to the dissertation, serve as a basis for the design of the $\Sigma\Delta$ filter proposed in this chapter. Extending on those prior works in this chapter is the sensitivity analysis, noise analysis, bitwidth optimization, and RLT design that allows one to characterize and construct such $\Sigma\Delta$ based IIR filters with a reliable prediction of performance.

3.7 Conclusions

In this chapter, the $\Sigma\Delta$ filter architecture was introduced along with a description on how to design the hardware to emulate any specified LTI filter design. A cogent representation and quantization noise analysis and coefficient sensitivity analysis was presented which give way to two important performance metrics: signal to noise ratio and transfer function deviation. The SNR and deviation metrics were used to construct an optimization strategy for reducing the coefficient bitwidths (and ultimately the overall size of the filter hardware implementation) by maximizing the individual coefficient quantization steps. Using the noise and sensitivity analysis in conjunction with the optimization routine, an 8th order filter was designed, simulated, and routed in a low power Xilinx Artix-7 FPGA. The simulation showed that the quantized $\Sigma\Delta$ held to the design metrics while using very little FPGA resources and power at varying oversample ratios.

Bibliography

[24] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[25] L. Fernandes, C. A. Leme, and J. Franca. Programmable IIR bitstream filters. In *38th Midwest Symposium on Circuits and Systems. Proceedings*, volume 1, pages 576–579 vol.1, Aug 1995.

[26] H. Fujisaka, R. Kurata, M. Sakamoto, and M. Morisue. Bit-stream signal processing and its application to communication systems. *IEE Proceedings - Circuits, Devices and Systems*, 149(3):159–166, June 2002.

[27] H. Fujisaka, N. Masuda, M. Sakamoto, and M. Morisue. Arithmetic circuits for single-bit digital signal processing. In *ICECS'99. Proceedings of ICECS '99. 6th IEEE International Conference on Electronics, Circuits and Systems (Cat. No.99EX357)*, volume 3, pages 1389–1392, Sep. 1999.

[28] G. C. Goodwin, R. H. Middleton, and H. V. Poor. High-speed digital signal processing and control. *Proceedings of the IEEE*, 80(2):240–259, Feb 1992.

[29] D. A. Johns and D. M. Lewis. Design and analysis of delta-sigma based IIR filters. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 40(4):233–240, April 1993.

[30] D. A. Johns, D. M. Lewis, and D. Cherepacha. Highly selective 'analog' filters using /spl Delta//spl Sigma/ based IIR filtering. In *1993 IEEE International Symposium on Circuits and Systems*, pages 1302–1305 vol.2, May 1993.

[31] D.A. Johns and D.M. Lewis. Sigma-delta based IIR filters. In *Circuits and Systems, 1991., Proceedings of the 34th Midwest Symposium on*, pages 210–213 vol.1, May 1991.

[32] J. Juni and R. W. Stewart. Implementing adaptive DSP algorithms using oversampled sigma delta strategies. In *IEE Colloquium on Oversampling and Sigma-Delta Strategies for DSP*, pages 9/1–9/9, Nov 1995.

[33] G. Li and M. Gevers. Comparative study of finite wordlength effects in shift and delta operator parameterizations. *IEEE Transactions on Automatic Control*, 38(5):803–807, May 1993.

[34] Sanjit K. Mitra. *Digital Signal Processing.* McGraw-Hill Science/Engineering/Math, 2005.

[35] Chiu-Wa Ng, Ngai Wong, H. Kwok-Hay So, and Tung-Sang Ng. Direct sigma-delta modulated signal processing in FPGA. In *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, pages 475–478, Sept 2008.

[36] Chiu-Wa Ng, Ngai Wong, H. Kwok-Hay So, and Tung-Sang Ng. Quad-level bit-stream signal processing on FPGAs. In *ICECE Technology, 2008. FPT 2008. International Conference on*, pages 309–312, Dec 2008.

[37] C.W. Ng, N. Wong, and T.S. Ng. Efficient FPGA implementation of bit-stream multipliers. *Electronics Letters*, 43(9):496–497, April 2007.

[38] C.W. Ng, N. Wong, and T.S. Ng. Quad-level bit-stream adders and multipliers with efficient FPGA implementation. *Electronics Letters*, 44(12):722–724, June 2008.

[39] T. Song, E. G. Collins, and R. H. Istepanian. Improved closed-loop stability for fixed-point controller implementation using the delta operator. In *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, volume 6, pages 4328–4332 vol.6, June 1999.

[40] S. Summerfield, S. M. Kershaw, and M. B. Sandler. VLSI design of a sigma-delta bitstream FIR filter. In *1994 IEE Colloquium on Digital and Analogue Filters and Filtering Systems (Digest No. 1994/233)*, pages 3/1–3/5, Nov 1994.

[41] Ngai Wong and Tung-Sang Ng. Roundoff noise minimization in a modified direct-form delta operator IIR structure. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 47(12):1533–1536, Dec 2000.

[42] Ngai Wong and Tung-Sang Ng. A generalized direct-form delta operator-based IIR filter with minimum noise gain and sensitivity. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 48(4):425–431, April 2001.

[43] Ngai Wong and Tung-Sang Ng. Improved roundoff noise performance in a direct-form IIR filter using a modified delta operator. In *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, volume 2, pages 773–776 vol. 2, May 2001.

[44] J. Wu, G. Li, R. H. Istepanian, and J. Chu. Shift and delta operator realisations for digital controllers with finite word length considerations. *IEE Proceedings - Control Theory and Applications*, 147(6):664–672, Nov 2000.

[45] Yifei Liu and Wei Tang. A Delta Sigma based Finite Impulse Response Filter for EEG Signal Processing. In *2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1–4, Aug 2015.

Chapter 4

High Resolution ΣΔ Controller Designs

While a single ΣΔ encoded bitstream is capable of high resolution, there is a limit to how much information it can convey at a specified oversample ratio. For the typical second order ΣΔ modulator, it is possible to obtain 14 to 16 effective bits of resolution with oversample ratios of 256 or more. For effective signal bandwidths on the order of tens to hundreds of kilohertz this would imply sampling frequencies of at least tens of megahertz. For embedded systems that must be limited in power and clock frequency, achieving effective resolutions above 16 bits becomes a decreasingly effective. The question is then, could anything be done to increase resolution of the encoded bitstream without using an unreasonably large OSR value? As it turns out, the answer to the problem lies with the use of MASH ΣΔ modulation.

Since the information capacity of one bitstream is rather limited, the only way to increase it without increasing the sampling frequency is to increase the number of bits of the ΣΔ encoded signal. Introduced in [48], Multistage Noise Shaping (MASH) ΣΔ modulators accomplish this task by introducing multiple ΣΔ modulators to not only encode the signal information but also the single bit quantization error of the signal stream in a separate stream. The MASH technique creates a higher order of noise suppression which in turn produces a

higher signal to noise ratio compared to a conventional modulator [50]. The resulting oversampled multibit signal in such a configuration is capable of achieving 20 or more bits of effective resolution for modest oversample ratios.

In this chapter, it will be shown how to modify the $\Sigma\Delta$ filter architecture from the previous chapter with MASH $\Sigma\Delta$ encoding for the purpose of creating filter and controller implementations that can achieve over 20 bits of effective resolution with only a modest increase in circuit complexity and at lower oversampling rates. An initial overview of MASH $\Sigma\Delta$ modulation is presented followed by a description of the modified MASH $\Sigma\Delta$ filter architecture. Scaling, coefficient sensitivity, and noise performance of the MASH $\Sigma\Delta$ controller will presented with the chapter concluding with a modified circuit minimization strategy for the overall discrete architecture.

4.1 Multistage Noise Shaping (MASH) $\Sigma\Delta$ Modulation

MASH $\Sigma\Delta$ modulation consists of multiple modulators that encode signal and quantization error information in order to produce a composite signal that achieves greater in band representation noise suppression. The trademark qualities of a MASH encoded signal PSD will have a steeper representation noise slope and a lower in band noise floor which contribute to a much higher SNR compared to signal stage $\Sigma\Delta$ modulators. Figure 4.1 shows a comparison of the typical PSDs associated with a single stage and MASH modulator encoded signal.

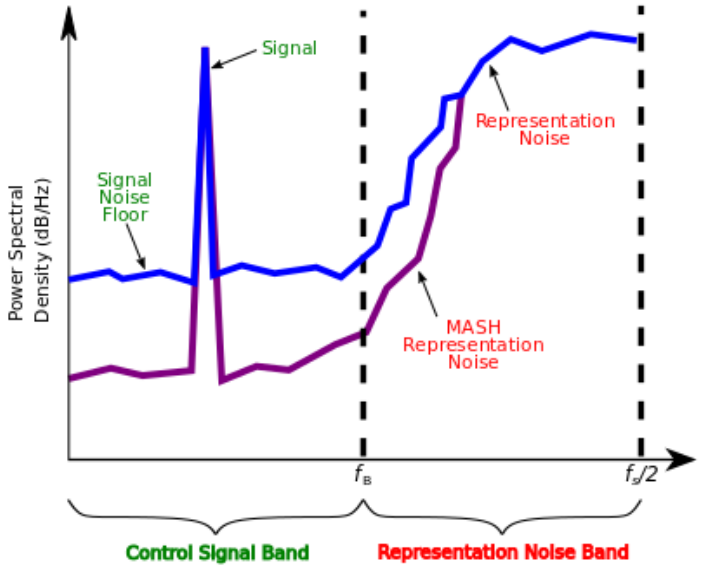


Figure 4.1: MASH Power Spectral Density

The block diagram of a generalized two stage MASH ΣΔ converter can be seen in figure 4.2.

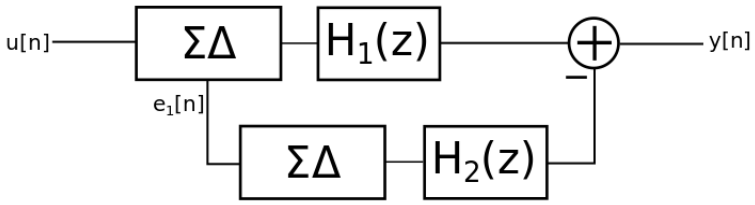


Figure 4.2: Conventional 2 Stage MASH

It is shown in the above figure that the input ΣΔ encodes the input signal u which is then filtered by $H_1(z)$. The quantization error of the first stage ΣΔ modulator $E_1(z)$ is encoded by the second stage modulator and filtered by $H_2(z)$. The output of $H_2(z)$ is then subtracted from the output of $H_1(z)$ to produce a composite output with high order noise suppression. The discrete logic of $H_1(z)$ and $H_2(z)$ and the subtractor circuit is known as cancellation logic. The purpose of the cancellation logic is to eliminate any cross terms that include the first stage quantization error e_1 that may appear in the output y_{sig} leaving only a high order noise shaping

of the secondary quantization noise e_2. To demonstrate the high order noise suppression that is achievable with MASH, the following derivation will assume a two stage discrete converter using second order modulators for signal and quantization encoding.

Assuming the linear $\Sigma\Delta$ model, the outputs of the signal and noise encoded paths are

$$Y_{sig}(z) = H_1(z)\left[STF_1(z)U(z) + NTF_1(z)E_1(z)\right]$$
$$Y_{error}(z) = H_2(z)\left[STF_2(z)E_1(z) + NTF_2(z)E_2(z)\right]$$

where $H_1(z)$ and $H_2(z)$ are cancellation circuitry that will be discussed later. The composite output $Y(z)$ can be obtain by subtracting $Y_{error}(z)$ from $Y_{sig}(z)$ which yields

$$\begin{aligned} Y(z) &= Y_{sig}(z) - Y_{error}(z) \\ &= H_1(z)\left[STF_1(z)U(z) + NTF_1(z)E_1(z)\right] \\ &\quad -H_2(z)\left[STF_2(z)E_1(z) + NTF_2(z)E_2(z)\right] \\ &= H_1(z)STF_1(z)U(z) + \left[H_1(z)NTF_1 - H_2(z)STF_2\right]E_1(z) \\ &\quad -H_2(z)NTF_2(z)E_2(z) \end{aligned}$$

From here it is necessary to cancel the cross term $\left[H_1(z)NTF_1(z) - H_2(z)STF_2(z)\right]E_1(z)$ in order to extinguish any representation noise contributed from the first stage modulator. This can be done by setting the cross term to zero and rewriting the term as

$$\begin{aligned} H_1(z)NTF_1(z) - H_2(z)STF_2(z) &= 0 \\ H_1(z)NTF_1(z) &= H_2(z)STF_2(z) \end{aligned}$$

which leads to the following relationship

$$H_1(z) = STF_2(z)$$

$$H_2(z) = NTF_1(z)$$

For the purposes of this dissertation, the general second order $\Sigma\Delta$ modulator signal and noise transfer functions will be used giving

$$STF_1 = z^{-1}$$

$$STF_2 = z^{-1}$$

$$NTF_1 = \left(1 - z^{-1}\right)^2$$

$$NTF_2 = \left(1 - z^{-1}\right)^2$$

The overall output for a two stage second order MASH $\Sigma\Delta$ converter can then be written as

$$Y(z) = STF_1(z)\, STF_2(z)\, U(z) - NTF_1(z)\, NTF_2(z)\, E_2(z)$$

$$Y(z) = z^{-2}(z)\, U(z) - \left(1 - z^{-1}\right)^4 E_2(z)$$

The equation for $Y(z)$ indicates that the input signal is delayed by merely two oversampled clock cycles while the quantization noise of the second stage modulator is attenuated by fourth order shaping. In the information band of the oversampled signal, the input signal effectively sees zero phase delay while the representation noise is attenuated with a steep fourth order noise shaping.

Figure 4.3 plots SNR in decibels vs OSR for ideal single, double, and triple stage MASH converters using second order modulators. What is indicated in this plot is that for relatively low oversample ratios, MASH converters can achieve comparable resolutions to that of a single

stage modulator. In fact, a two stage MASH converter can achieve about 16 bits of resolution at an OSR of about 32 which would require an OSR of about 256 for a single stage converter. For OSRs above 64, the two stage MASH converter can theoretically achieve 18 or more bits of effective resolution.

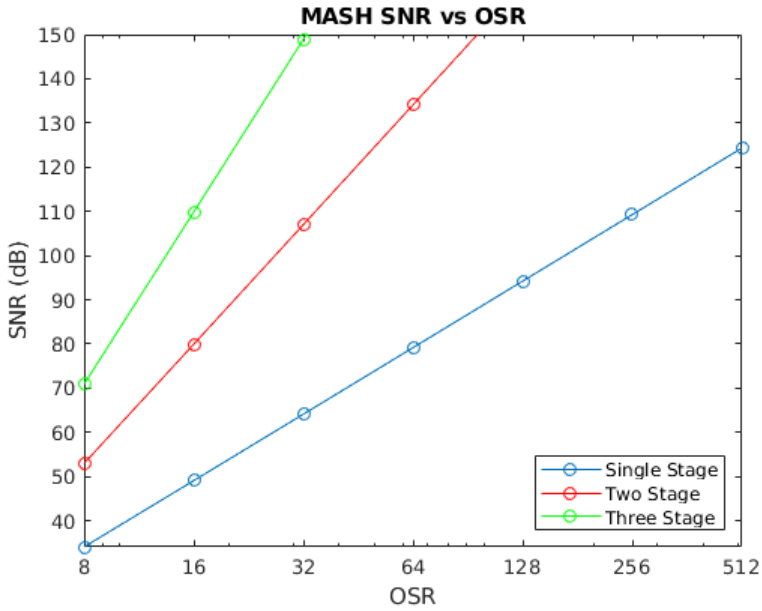


Figure 4.3: Ideal MASH 2-2 SNR vs OSR

The importance of the MASH $\Sigma\Delta$ encoder is that it allows one to obtain high resolution information encodings at relatively low oversample ratios compared to that of a single stage $\Sigma\Delta$ modulator. By decreasing the overall sample frequency, MASH based designs can potentially reduced power dissipation (due to dynamic power dissipation) and circuit complexity (due to the more relaxed time scaling required in δ-operator based filter designs). Another advantage is that low speed grade and cost effective parts could be used while still achieving high bandwidth and high resolution conversion.

In this thesis, two stage second order MASH converters will be assumed for every MASH data conversion. In addition, a signal and noise transfer function of transfer function of $STF_{M\Sigma\Delta} = z^{-2}$ and $NTF_{M\Sigma\Delta} = \left(1 - z^{-1}\right)^4$ will be assumed as well. The remainder of the

chapter will describe how to modify the ΣΔ filter architecture with MASH encodings in order to increase its performance.

4.1.1 MASH ΣΔ for Direct Signal Processing

Using a second order two stage MASH architecture to encode signal information for direct signal processing requires a bit of thought. For the purposes of filter design, one would like to eliminate the need for fixed point multipliers when processing ΣΔ encoded signals but the output of a MASH encoder is multibit. Fortunately, the MASH output can takes on a very small set of values.

For direct signal processing, rather than immediately combining y_{sig} and y_{error}, the proposed method is to keep them separate as shown in figure 4.4. In this way, downstream digital logic can be designed to process the signal and quantization error separately as well as allow for a simple two bit interface. For the encoded signal information we have $y_{sig} \in [-1,\ 1]$ which is a bitstream. For the encoded error information we have $y_{error} \in [-4,\ -2,\ 0,\ 2,\ 4]$ and requires 4 bits of representation.

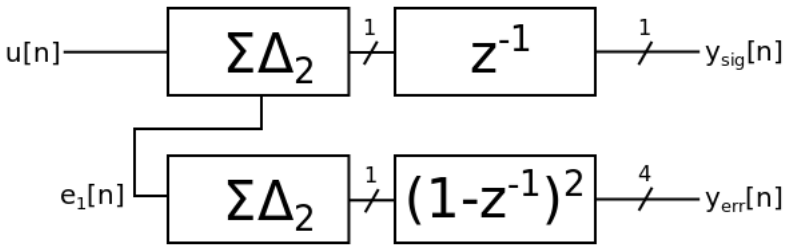


Figure 4.4: Proposed 2 Stage MASH

The signal path cancellation logic $H_1(z)$ is a single delay and can be implemented using a single flip flop. The error path cancellation logic $H_2(z)$ for a two stage MASH encoder with a first stage second order modulator is a double differentiator. The output of the second stage modulator is a bitstream making the implementation of $H_2(z)$ quite simple. A digital hardware implementation of $H_2(z)$ can be seen in figure 4.5.

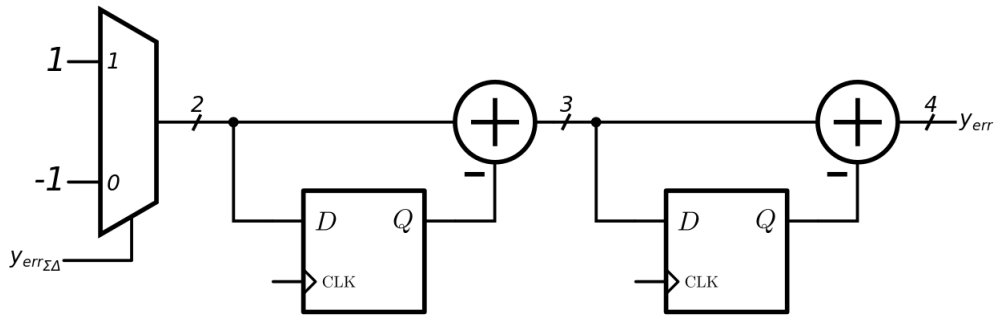


Figure 4.5: Double Differentiator Circuit

A simple multiplexer, five flip flops, and a three bit and four bit adder, are all that is needed to construct the double differentiator logic. The output y_{error} can be synthesized from the bitstream output of the second stage MASH modulator using only a minimal amount of logic.

If a device with a integrated MASH $\Sigma\Delta$ encoder produced both signal and quantization error bitstreams, the cancellation logic could easily by implemented in the programmable logic of an FPGA or the discrete design of an ASIC. Moving the cancellation logic of the MASH encoder from the front end transducer of the signal chain to the device intended to implement the discrete logic would require only a two wire interface and would maintain the low latency of the overall design. The ultimate goal is to allow for the creation of a two bit input/output interface for embedded controllers such as shown in figure 4.6.

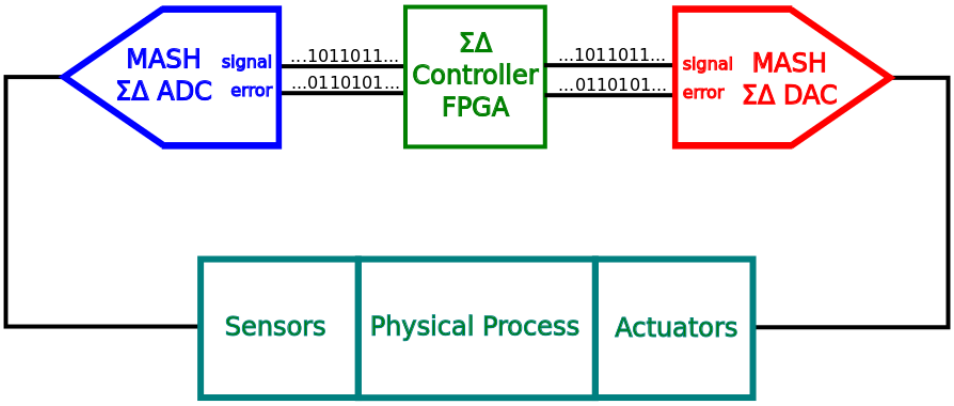


Figure 4.6: MASH ΣΔ Embedded Control System

In this way, by simply increasing the interface to two bits, MASH modulators can be used to dramatically increase SNR of the total embedded control system while maintaining the same latency and dynamic performance at a lower OSR.

4.2 MASH ΣΔ Filter Architecture

The MASH ΣΔ controller filter architecture seen below in figure 4.7 is a modified version of the the architecture presented in the previous chapter. While the internal signal paths of the controller take on a two's compliment fixed point representation, the feedforward and feedback paths are now five bit wide MASH ΣΔ encoded signals.

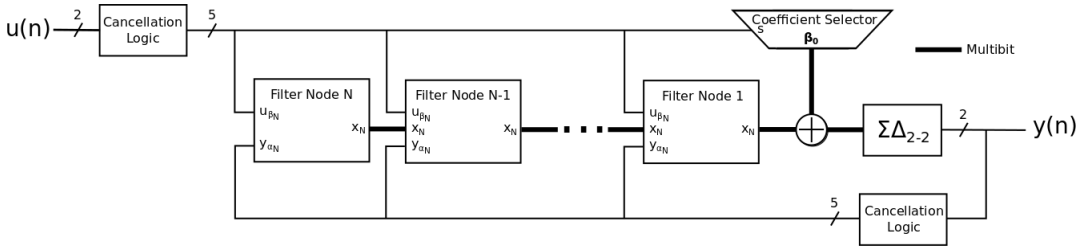


Figure 4.7: MASH IIR Filter

For the single stage $\Sigma\Delta$ modulator design, the one bit input and output channels allowed for simple implementation of coefficient multiplies using multiplexers. However with a five bit MASH channel, coefficient multiply implementations require a slightly more involved design.

The first change in the filter is that there now exists a discrete two stage second order MASH encoder in the output. An RTL diagram of the MASH modulator used in this thesis can be seen in figure 4.8. The proposed discrete MASH modulator maintains the signal and noise transfer functions assumed in the beginning of the chapter for the analysis and design of the overall filter architecture. It is possible to choose another topology for the discrete MASH modulator so long as it maintains the signal and noise transfer functions assumed for implementing the discrete cancellation logic.

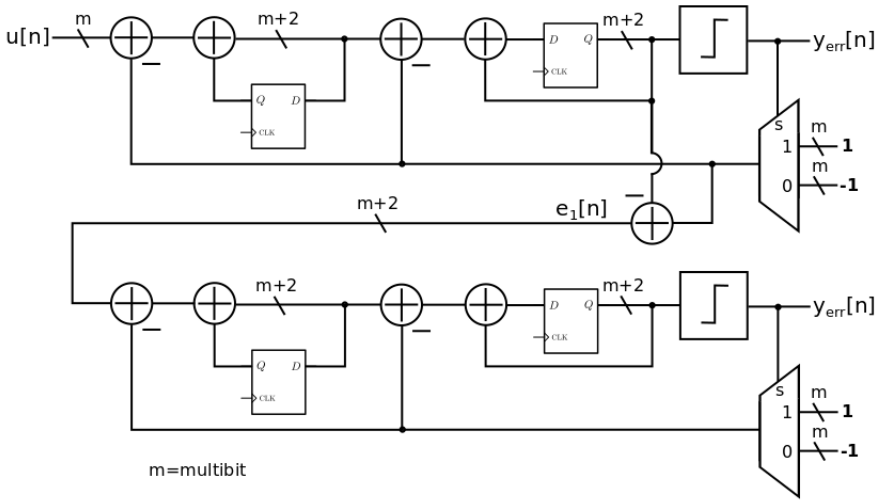


Figure 4.8: Discrete MASH $\Sigma\Delta$ Circuit

The internal MASH filter nodes shown in figure 4.9 still make use of of the δ operator implemented as an accumulator circuit and a corresponding bitshift which invokes the time constant and magnitude scaling of the controller. What is different however is the coefficient selector circuit for both the α and β coefficients. The selector circuits take in a five bit MASH encoding and output an appropriately scaled coefficient value.

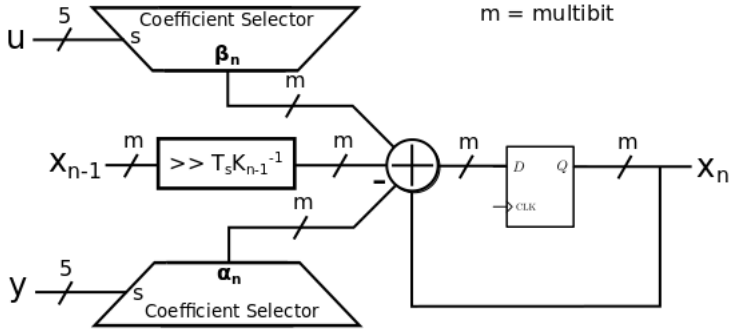


Figure 4.9: MASH IIR Filter Node

At this point, it is important to understand that the five bit MASH $\Sigma\Delta$ encoded signal is actually a composite of the 1 bit y_{signal} signal and the four bit two's compliment y_{error} signal. The allowable values that each signal can take is $y_{signal} \in [-1,\ 1]$ and $y_{error} \in [-4,\ -2,\ 0,\ 2,\ 4]$. Using this fact, the MASH2-2 coefficient selector circuit can be seen below in figure 4.10.

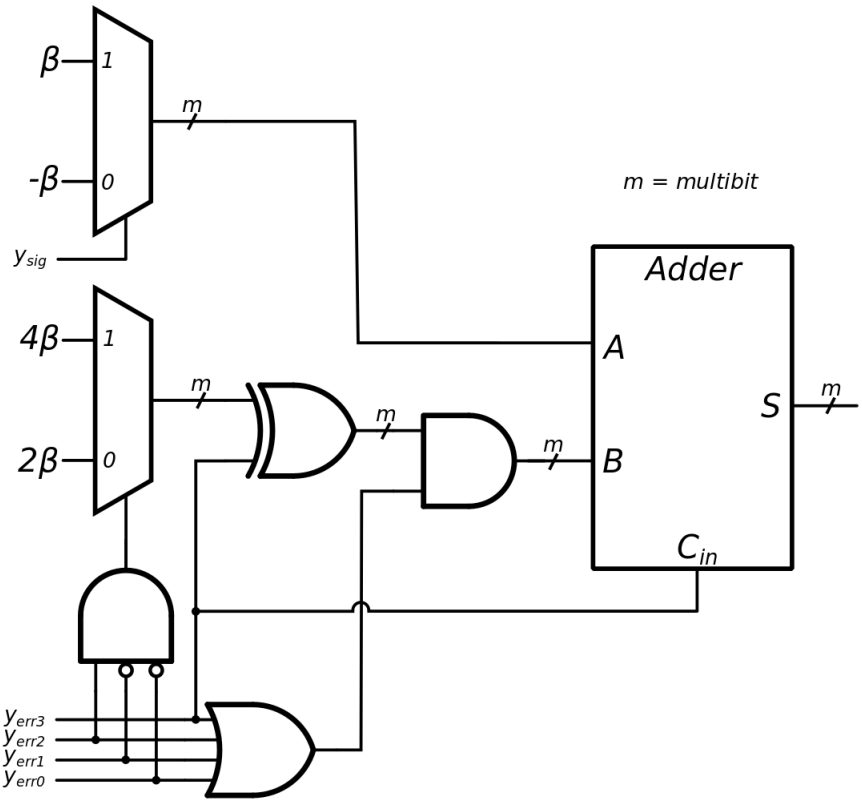


Figure 4.10: MASH 2-2 Coefficient Selector Circuit

The coefficient selector circuit operates by subtracting the corresponding scaled β coefficient value given by y_{error} from the switched β coefficient value of y_{signal}. For the y_{signal} branch of the circuit, the coefficient multiply of $\beta \cdot y_{signal}$ is implemented as before with a signal multiplexer switched by the signal bitstream. For $\beta \cdot y_{error}$, the circuit first detects whether or not $|y_{error}| = 2$ or $|y_{error}| = 4$. If so, a multiplexer switches the appropriate scaled β coefficient to an exclusive OR gate which provides the one's compliment of the product should y_{error} have a negative sign. For negative numbers, the ones compliment of the product is then added to the value of the y_{signal} branch plus one on the adder carry to form the correctly scaled output product value. For positive numbers the scaled product is fed through to be added to the product of the y_{signal} branch. In the case of $y_{error} = 0$, the AND gate sets all bits to zero feeding into the adder on the y_{error} branch which nullifies any contribution to the final product value

of the coefficient selector circuit. The allowable values that the coefficient selector output y_{out} can take is $y_{out} \in [-5\beta,\ -3\beta,\ -\beta,\ 0,\ \beta,\ 3\beta,\ 5\beta]$.

From a top level perspective, the MASH $\Sigma\Delta$ is a two bit input and two bit output hardware circuit that implements transfer functions. Using slight modifications to the $\Sigma\Delta$ filter architecture, a new platform for high resolution transfer functions implementations can be realized with only a slight modification to the overall circuit complexity.

4.3 MASH $\Sigma\Delta$ Controller Design and Performance Metrics

Like the $\Sigma\Delta$ filter architecture, the MASH $\Sigma\Delta$ filter architecture implements a transfer function which could either be a LTI filter or controller. While they may have differences in resolution performance, mathematically the two implementation versions are equivalent and produce the same transfer function input/output relationship. That being said, the quantization noise and sensitivity analysis of the MASH version is very similar to the analysis presented in the previous chapter. The main difference is in the widths of the data paths interior of the filter and the order of the noise transfer functions of the input and output MASH $\Sigma\Delta$ modulators.

To design a MASH filter/controller one must follow the same steps as outlined in the previous chapter. Those steps are as follows:

1. Begin with continuous state space realization of filter/controller.

2. Determine the OSR and clock rate for which the filter will run such that the required SNR is achieved on the front end modulator/converter.

3. Discretize state space model to δ domain and apply structural transformation (T_0) and scaling (T_s).

4. Calculate the representation noise and sensitivity metrics assuming the two stage second order MASH noise transfer function of $NTF_{\Sigma\Delta}(z) = \left(1 - z^{-1}\right)^4$.

5. Set the magnitude deviation bound (ρ) and quantization noise bound (γ_{noise}) and run the quantization step optimization routine.

4.3.1 Choosing Filter Bitwidths

Like the single stage $\Sigma\Delta$ filter the bitwidths of the MASH filter must be chosen carefully for all of the internal signal paths. Again the bitwidths of importance are those of the coefficients and state registers. The numerical representation of these signals is 2's compliment form and consist of bitwidths $B \in \mathbb{Z}_{\geq 0}$ given by

$$B = I + F + 1$$

where $I \in \mathbb{Z}_{\geq 0}$ are the integer bits, $F \in \mathbb{Z}_{\geq 0}$ are the fractional bits, and a one for the sign bit.

When it comes to the α and β coefficients for the MASH implementation, it must be taken into account that the output of the coefficient selector takes on the values from the set $[-5\beta,\ -3\beta,\ -\beta,\ 0,\ \beta,\ 3\beta,\ 5\beta]$. Normally the integer bits I_i for the nth stage can be found via

$$I_n = \lceil max\left(log_2\left(\beta_{\delta n}\right), log2\left(\alpha_{\delta n}\right)\right)\rceil$$

since the the coefficients share the same summation node, but an additional four bits must be used to represent the maximum absolute value of 5β. Therefore, the integer bits for the coefficient selector circuit should be adjusted to

$$I_n = \lceil max\left(log_2\left(\beta_{\delta n}\right), log2\left(\alpha_{\delta n}\right)\right)\rceil + 4$$

The number for fractional bits F at the nth node can be determined by

$$F_n = \begin{cases} \lceil |log2\left(\Delta_{qn}\right)| \rceil, & \Delta_{qn} < 0 \\ 0, & otherwise \end{cases}$$

where Δ_{qn} is determined by the optimization routine detailed in chapter three.

Next, the integrator register widths must be set, Taking into account that the integrator nodes are unity scaled, the number of integer bits I_{xi} of the state register can be found by

$$I_{xn} = \lceil |log2\,(\Delta)| \rceil$$

The integrator register sharing the same summation node as the output of the coefficient selector circuits should have the same fractional bits F_n determined above. The total bitwidth of the nth register can then be written as

$$B_{xn} = I_{xn} + F_n + 1$$

Figure 4.11 shows a diagram of a MASH filter node and the appropriate bitwidths of each internal signal path.

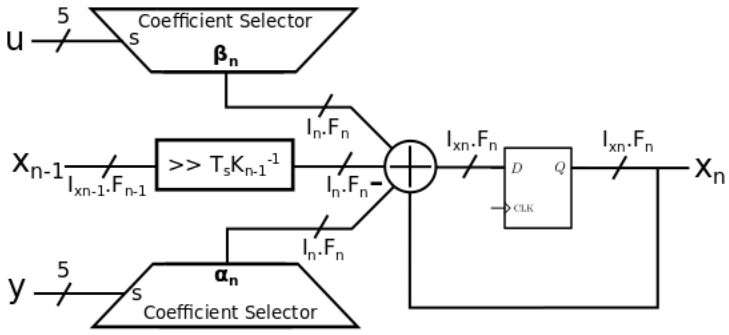


Figure 4.11: MASH ΣΔ Filter Node Bitwidths

4.4 Design Example

Like the previous chapter, an 8th order Chebychev Type II bandpass filter implementation will be used to demonstrate the effectiveness of the MASH filter architecture. Table 4.1 lists the design parameters of the filter. Again, the filter was constructed without breaking it up into to smaller sections such as second order.

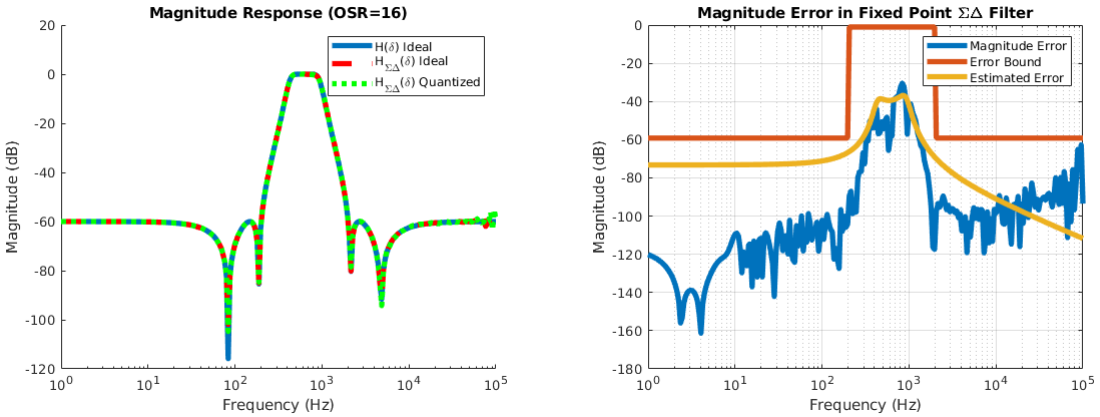
Parameter	Value
Filter Type	Bandpass Chebyshev Type II
Passband	200 - 2000 Hz
Signal Band	DC - 10 kHz
Order	8
Stopband Attenuation	60 dB

Table 4.1: Design Parameters of MASH ΣΔ Filter

The filter was simulated with an OSR of 16 and a target ENOB of 16 in the passband. The magnitude deviation was chosen to be 1 db in the passband and stopband from the ideal magnitude response. The appropriate ρ was chosen to be

$$\rho(f) = \begin{cases} 0.831 & \forall f \in [200,\ 2000] \\ 0.0011 & otherwise \end{cases}$$

to achieve the desired deviation bound and used as a constraint in the bitwidth optimization routine. Figure4.12 shows the magnitude response and error for the filter with an OSR of 16.



(a) Magnitude Response

(b) Magnitude Error

Figure 4.12: Filter Magnitude Response and Error

By varying the OSR and noise requirements of the filter and mapping the design into an Artix-7 XC7A12T-1CPG238 FPGA from Xilinx, table 4.2 was generated. The magnitude error bound, ρ, was used for each of the four filter designs in the table. The table demonstrates

that the optimization routine was able to achieve the desired ENOB while using a minimal amount of FPGA resources and relatively low OSRs.

Design	OSR	Design ENOB	Calculated ENOB	Power (μW)	LUT/FF
1	16	13	13.37	2/58	1785/375
2	32	18	18.25	3/58	1893/389
3	64	22	22.13	6/58	1997/401
4	128	26	26.57	9/58	2011/413

Table 4.2: Filter Design Resource Usage

4.5 Conclusion

In this chapter, the MASH $\Sigma\Delta$ filter architecture was presented. As a modified version of the single stage filter design from the previous chapter, the MASH architecture allows filters and controllers to achieve twenty or more bits of effective resolution at modest oversample ratios. The increase in resolution comes at only a modest increase in circuit complexity of which the multiplierless design can easily fit inside small low power FPGAs. The two bit IO interface of the MASH architecture also reduces complexity when interfacing components in the embedded signal chain.

An 8th order bandpass filter was constructed and simulated to demonstrate the performance of the MASH based filter design. At an OSR of 64, the filter was able to achieve over 22 bits of effective resolution in its passband. The resources required to place and route the filter were small enough to fit five copies of the filter in the smallest available Artix-7 FPGA. The magnitude response of the filter was also virtually indistinguishable from the initial transfer function magnitude design.

Bibliography

[46] G. C. Goodwin, R. H. Middleton, and H. V. Poor. High-speed digital signal processing and control. *Proceedings of the IEEE*, 80(2):240–259, Feb 1992.

[47] D.A. Johns and D.M. Lewis. Sigma-delta based IIR filters. In *Circuits and Systems, 1991., Proceedings of the 34th Midwest Symposium on*, pages 210–213 vol.1, May 1991.

[48] Y. Matsuya, K. Uchimura, A. Iwata, et al. A 16-bit oversampling A-to-D conversion technology using triple-integration noise shaping. *IEEE Journal of Solid-State Circuits*, 22(6):921–929, Dec 1987.

[49] Chiu-Wa Ng, Ngai Wong, H. Kwok-Hay So, and Tung-Sang Ng. Direct sigma-delta modulated signal processing in FPGA. In *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, pages 475–478, Sept 2008.

[50] R. Schreier and G.C. Temes. *Understanding Delta-Sigma Data Converters*. Wiley, 2004.

[51] Ngai Wong and Tung-Sang Ng. Roundoff noise minimization in a modified direct-form delta operator IIR structure. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 47(12):1533–1536, Dec 2000.

[52] Ngai Wong and Tung-Sang Ng. A generalized direct-form delta operator-based IIR filter with minimum noise gain and sensitivity. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 48(4):425–431, April 2001.

[53] Ngai Wong and Tung-Sang Ng. Improved roundoff noise performance in a direct-form IIR filter using a modified delta operator. In *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, volume 2, pages 773–776 vol. 2, May 2001.

[54] X. Wu and R. M. Goodall. One-bit processing for digital control. *IEE Proceedings - Control Theory and Applications*, 152(4):403–410, July 2005.

Chapter 5

Implementation and Realization of $\Sigma\Delta$ Filters

Having developed a method for designing and characterizing $\Sigma\Delta$ filters, this chapter will explore the design trade-offs and performance of such filters against conventional digital implementations. The claims of this thesis are that these new filter/controller implementations offer lower power, lower complexity, high resolution, and continuous time performance compared to conventional time-shift based filter constructions. Optimizing various filter implementations under different design constraints allows exploration of not only the gross morphology, but also fine-scale trends and trade-offs inherent in these designs. Unfortunately, these explorations are by no means comprehensive, issues such as design decomposition and cascade realizations are out of scope of the currently implemented optimization techniques, however, as these implementations offer the practical ability for direct implementation of relatively high-order IIR filter designs, and the practical interest in such designs for anti-aliasing, demodulation and other uses, such filters are used in this chapter as benchmarks.

This chapter will first begin with addressing practical and pragmatic issues in the design and evaluation of $\Sigma\Delta$ filter implementations. Next, a conventional filter design will be presented to give a complexity and cost comparison. Various single stage and MASH $\Sigma\Delta$ filter

designs will then be presented and mapped into a small Xilinx FPGA while comparing trade-offs in OSR, ENOB, and magnitude deviation. Xilinx tools are used for the majority of comparisons as they offer logic/placement, routing, timing and power estimation for a variety of optimized time-shift implementation strategies and have a nicely constructed MATLAB/Simulink® library simplifying filter performance evaluation. The chapter will conclude with remarks on the effectiveness of the design methodology presented in chapters three and four to design $\Sigma\Delta$ filters compared to conventional filters of comparable response and resolution.

5.1 Pragmatic Issues in $\Sigma\Delta$ Filter Design and Simulation

Designing $\Sigma\Delta$ filters, a key pragmatic issue is that of accurate characterization and noise estimation. Given the potential for very low levels of noise in the lower end of the operating band, with simultaneously high levels of representation noise in the mid to high band, accurate estimation requires substantial simulation effort and attention to details. This is especially true for MASH designs where the potential for very high resolution and noise floors below -150dB are practical. $\Sigma\Delta$ filter/controller performance estimation issues are primarily centered on computing accurate power spectral density (PSD) estimates of bitstream channels. From the PSD, the SNR, ENOB, and transfer function estimates can be calculated. If the PSD is not accurate, then false conclusions can be made about the overall performance of a $\Sigma\Delta$ filter design.

5.1.1 Accurate PSDs and Related Estimations

Power spectral densities are a requirement for characterizing bitstreams and $\Sigma\Delta$ filters. When running simulations with oversampled bitstreams, one must take into account that the underlying dynamics that are encoded in the signals are two to three orders of magnitude slower that the sampling rate. To accurately measure low frequencies, it is essential that deep simulations be performed.

Consider the two transfer function estimates of figure 5.1. It is clear that $5 \cdot 10^5$ samples is not enough data points to accurately estimate the transfer function magnitude response. On

the other hand, 10^7 samples allows for a very nice transfer function estimation and magnitude response plot. As a rule of thumb, bitstream PSDs should require at least 10^7 samples for a Nyquist frequency of less than 100kHz and $OSR \leq 256$ in order to get an accurate estimation.

5.1.2 Calculating SNR

The signal to noise ratio again is the ratio of signal power, μ_{sig}^2, to noise variance, σ_{noise}^2, and is calculated via

$$SNR_{dB} = 10log_{10}\left(\frac{\mu_{sig}}{\sigma_{noise}}\right)$$

Since the filter/controller design is normalized across the entire signal band, the SNR should be taken at points in the frequency space equal to 0 dB. For a filter, the appropriate input stimulus is a full scale sine wave (90% full scale for $\Sigma\Delta$ due to code noise) in order to calculate the SNR. In the case of a bandpass filter, the sine wave frequency should be chosen so it is in the pass-band where the signal information matters and is not attenuated. There may be cases however where the SNR is calculated differently from integrating the noise floor from DC to Nyquist. If the signal bandwidth happens to be a subset of the inferred full signal frequency range, it will potentially render a higher SNR. The higher SNR is a direct consequence of integrating less noise over a smaller frequency band in the PSD.

Calculating the SNR of a sine wave $\Sigma\Delta$ bitstream can be accomplished using the following Matlab code:

```
%one-sided PSD of Sigma Delta bitstream (q_bitstream)
N = length(q_bitstream);
xdft = fft(q_bitstream);
xdft = xdft(1:N/2+1);
psdx = (1/(fs*N)) * abs(xdft).^2;
psdx(2:end-1) = 2*psdx(2:end-1);
bin = fs/N;
freq = 0:bin:fs/2;
```

```
%calulation of SNR assuming full scale sine-wave of frequency fsig
fsig_index = find(freq == fsig);
psig = psdx(fsig_index)/bin;
freq(fsig_index) = [];
psdx(fsig_index) = [];
index = find(freq <= fb);
SNR = 10*log10(psig/(2*trapz(freq(index),psdx(index))));
ENOB = (SNR-1.72)/6.02;
```

5.1.3 Transfer Function Estimation

Transfer function estimation was accomplished by using the 'tfestimate' Matlab function. The function estimates the transfer function response of a filter by dividing the cross power spectral density of the input and output, $P_{yx}(f)$, by the power spectral density of the input, $P_{xx}(f)$, where x and y are the bitstream input and output of the filter/controller. The transfer function estimation can be written as

$$H_{\Sigma\Delta}(f) = \frac{P_{yx}(f)}{P_{xx}(f)}$$

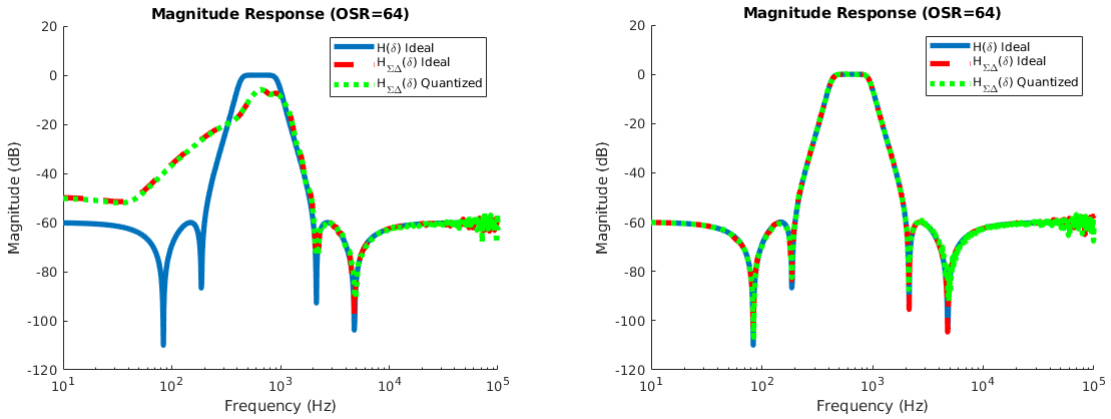
For an input stimulus, a white Gaussian noise source should be used due to its uniform power spectral density. White Gaussian noise will create sufficient frequency content across the entire frequency spectrum allowing for accurate estimation of $H_{\Sigma\Delta}(f)$.

For transfer function estimates that are to follow, a Hanning window was used over eight equal length sections of the data for every transfer function estimation in this thesis. The following Matlab code snippet demonstrates how to accomplish this task:

```
nx = max(size(input));
na = 8;
w = hanning(floor(nx/na));
```

```
[T_est, ~] = tfestimate(input,output,w,[],f,fs);
```

To demonstrate the effect of insufficient samples in the computation of a PSD, figure 5.1 shows the transfer function estimation of an 8th order bandpass filter for $5{\cdot}10^5$ and 10^7 samples. As can be seen, the magnitude estimation plot for $5 \cdot 10^5$ samples fails to encapsulate the low frequency information required to accurately compute PSDs of the input and output of the filter below 1kHz. In contrast the 10^7 sample magnitude estimate contains the low frequency information required for the plot and thus the magnitude response of the filter is well estimated.



(a) $5 \cdot 10^5$Sample Transfer Function Estimate

(b) 10^7 Sample Transfer Function Estimate

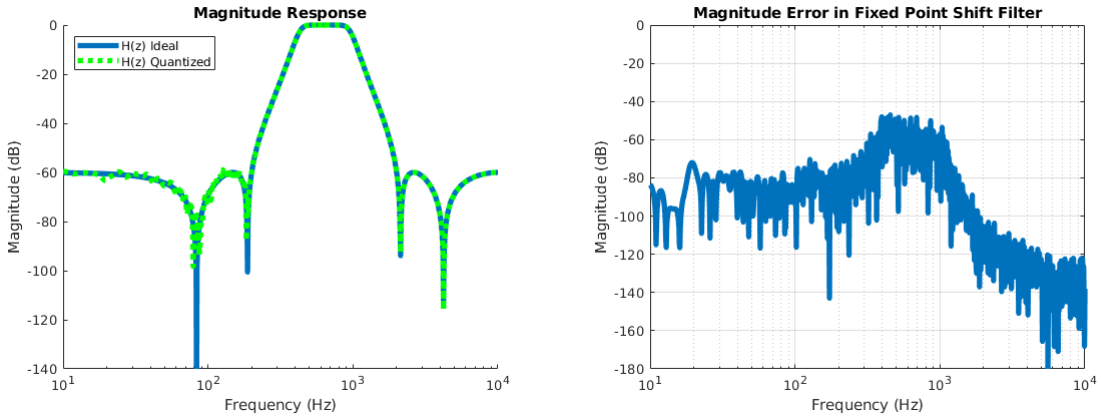
Figure 5.1: Insufficient Sample PSD Effects

5.1.4 Matlab/Simulink, Xilinx System Generator, and Vivado

The characterization and evaluation of all filter/controller designs used in this thesis were carried out in the Matlab/Simulink environment using Xilinx System Generator® blocks [55]. Bit accurate Xilinx System Generator blocks were placed in the Simulink environment in order to implement adders, multiplexers, registers and other digital components found in the $\Sigma\Delta$ system designs. Default parameters were used in all cases when mapping the Simulink models into Vivado FPGA place and route.

5.1.5 Design of Conventional Comparison Filter

To gauge the relative savings in $\Sigma\Delta$ filter implementations to conventional designs, a shift based filter is first created and implemented in a Xilinx Artix-7 XC7A12TL FPGA. The filter is the same 8th order Chebychev II bandpass filter presented in chapter 3. The filter has a pass band of 200 Hz to 2 kHz, a stopband attenuation of 60dB, and a sampling rate of 20 kHz. The 8th order filter was broken into four second order sections in order to mitigate quantization noise associated with coefficient multiplication rounding. The estimated magnitude response and associated error can be seen in figure 5.2



(a) Shift Filter Magnitude Response

(b) Shift Filter Magnitude Error

Figure 5.2: Magnitude Response and Error for Shift Based Bandpass Filter Design

Using the default Vivado implementation settings, the shift based filter was implemented in two ways: using the FPGA DSP blocks and mapping the entire design into LUTs and flip flops. The results can be seen in table 5.1.

Design	LUTs	FFs	DSPs	ENOB	IO Pins	Power (static/dynamic) mW
1	2171	204	20	14.17	32	58/1
2	6851	204	n/a	14.17	32	58/1

Table 5.1: Band-pass Shift Filter FPGA Resource Utilization

The results from the FPGA implementation of the shift based bandpass filter is that conventional filter designs require substantial physical resources. For the Xilinx Artix-7 XC7A12TL,

only one 8th order filter with 24 bit coefficients can be fit inside with room for little else if mapped entirely into logic. With use of DSP slices, about three 8th order filters could fit inside the FPGA. For more complex filters or MIMO designs, a much larger and potentially more expensive FPGA must be used. IO Pins are also used in large quantity unless a serialized interface such as SPI is used, but this increases latency which is detrimental in high bandwidth control applications.

5.2 $\Sigma\Delta$ Filter Resource and Power Utilization

To demonstrate the low resource usage of $\Sigma\Delta$ filter designs, this section will analyze several different filter types and design over a variety of specification constraints. In particular, lowpass and bandpass filters will be implemented in both signal stage modular designs as well as MASH designs. Parameters such as the order of the filter, the OSR, the required ENOB, and the allowed transfer function variation bound will be varied and compared.

5.2.1 Single Stage Designs

Single stage designs are those discussed in chapter three with a single bitstream input and output. Simulations of the filters were performed in the Matlab/Simulink environment using the Xilinx System Generator Library for implementing bit accurate performance. For performance evaluation in a FPGA, the Xilinx Vivado® design software was used.

5.2.1.1 Lowpass Filter Designs

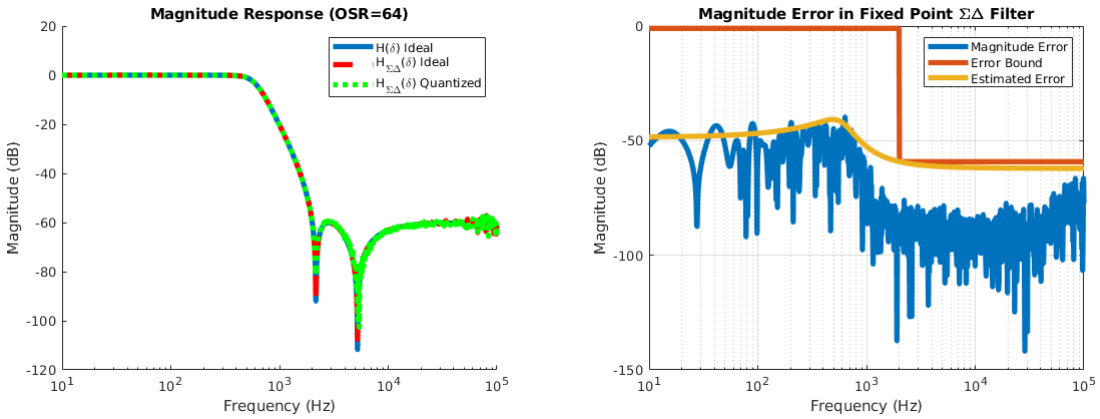
The following lowpass filter designs are based on a Chebychev II type filter with a cutoff frequency of 2kHz and a stopband attenuation specification of 60 dB, designed with different OSRs, magnitude error bounds, orders, and ENOB. The following magnitude error bounds used in the various designs, ρ_{LP1}, ρ_{LP2}, ρ_{LP3}, allow for 1dB deviation in the passband and 1dB, 5dB, and 10dB deviation in the stopband respectively:

$$\rho_{LP1}(f) = \begin{cases} 0.8913 & \forall f \leq 2000\ Hz \\ 0.0011 & otherwise \end{cases}$$

$$\rho_{LP2}(f) = \begin{cases} 0.8913 & \forall f \leq 2000\ Hz \\ 0.0018 & otherwise \end{cases}$$

$$\rho_{LP3}(f) = \begin{cases} 0.8913 & \forall f \leq 2000\ Hz \\ 0.0032 & otherwise \end{cases}$$

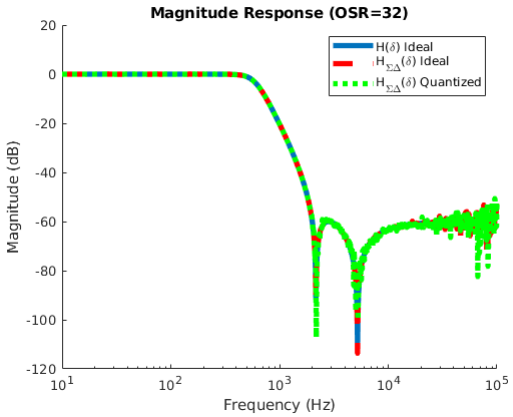
The examples of transfer function estimates can be found in figure 5.3, figure 5.4, and figure 5.5. In the magnitude response plots, the blue lines represent the ideal (continuous-time) magnitude curve, the red dashed lines represent an ideal ΣΔ filter simulated with full floating point precision, and the green dotted lines represent the bit accurate ΣΔ filter. As can be seen from the magnitude response plot , the quantized ΣΔ filters do an excellent job at emulating the ideal response with the plots virtually overlapping one another even at low OSRs. These plots were generated by using a white Gaussian random input stimulus and the transfer function estimation method previously described.



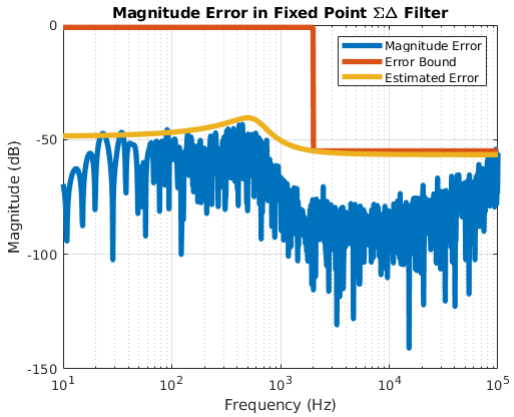
(a) Magnitude Response

(b) Magnitude Error

Figure 5.3: Lowpass Filter (4th order, OSR=64, ρ_{LP1})

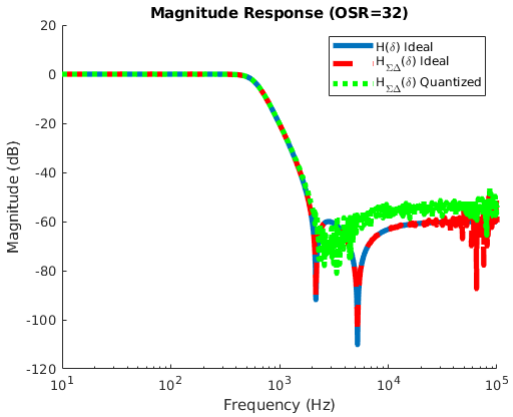


(a) Magnitude Response

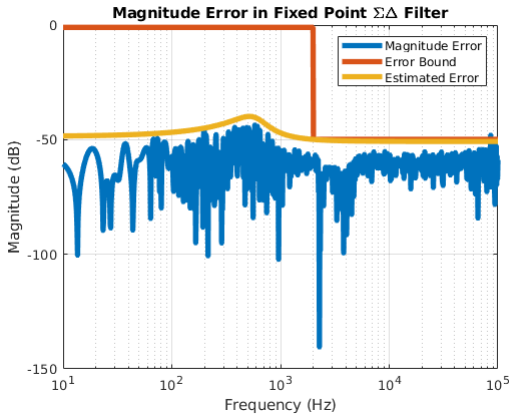


(b) Magnitude Error

Figure 5.4: Lowpass Filter (4th order, OSR=32, ρ_{LP2})



(a) Magnitude Response



(b) Magnitude Error

Figure 5.5: Lowpass Filter (4th order, OSR=32, ρ_{LP3})

Towards the Nyquist frequency of the magnitude response, it can be seen that the ΣΔ filter implementation becomes more noisy and begins to deviate from the ideal. This is due

to the rise in representation noise at the filter output which reduces the mutual information between the input and output bitstreams. To defeat this effect, one simply needs to increase the oversampling frequency.

Another effect seen in the magnitude response plots worth noting is the relaxation of the stopband attenuation. Figure 5.3 shows a magnitude response using ρ_{LP1} as a design input in the bitwidth optimization and as can be seen, the magnitude response is tight in the stopband of the filter. In contrast, figure 5.5 shows the magnitude response using ρ_{LP3} as a design constraint corresponding to a 10 dB deviation in the stopband. As a result, the quantized filter response has a higher stopband attenuation of about 50 dB as opposed to the ideal stopband attenuation of 60 dB.

The magnitude error plots show the red ρ_{LP} bound, the gold estimated error bound computed from the actual quantization levels, and the actual magnitude error which is defined as the absolute difference between the ideal and quantized $\Sigma\Delta$ magnitude responses. For all filters, it is evident that not only does the actual magnitude error in blue fit below the red error bound, but it also fits relatively tight against the gold estimated error bound. (The gold estimation is on the specification boundary as it is the metric used by the bit-width optimization procedure). These estimates suggest that the sensitivity analysis and optimization strategy outlined in chapter 3 is a sound predictor of the error imposed by quantization in the $\Sigma\Delta$ filter itself..

Using ρ_{LP1}, ρ_{LP2}, ρ_{LP3}, in addition to varying other parameters, the $\Sigma\Delta$ filters can be mapped into an actually programmable device to evaluate their performance in terms of resource usage, power, and resolution. For the remainder of this chapter, all filter designs will be routed in a Xilinx Artix-7 XC7A12TL via Xilinx System Generator and Vivado® for performance evaluation. Table 5.2 shows the measured ENOB for various parameterized versions of the Chebychev II lowpass filter. The table demonstrates as the OSR increases in conjunction with the design ENOB, the implemented $\Sigma\Delta$ filters easily exceed the specification. This is an indication that the ρ_{LP} magnitude error bound is the dominate constraint in the optimization and that the filter dynamics can actually attenuate representation noise in the signal band for a higher SNR. Another insight, is that as the stopband attenuation is relaxed going from ρ_{LP1} to

ρ_{LP3}, the ENOB goes down by a small amount of one to one and a half ENOB. The degradation in the magnitude response increases representation noise and quantization noise in the signal band.

OSR	Order	Design ENOB	ENOB (ρ_{LP1})	ENOB (ρ_{LP2})	ENOB (ρ_{LP3})
32	2	10	14.72	13.95	13.08
64	2	12	15.17	14.32	13.72
128	2	14	15.67	14.87	14.50
256	2	15	16.22	15.45	15.17
32	4	10	14.31	13.39	12.44
64	4	12	14.73	14.05	13.34
128	4	14	15.29	14.71	14.54
256	4	15	15.81	15.37	15.25

Table 5.2: Lowpass Filter Implementation (Cheby II f_band=2kHz, f_b=100kHz, Rs = 60 db)

Moving onto FPGA utilization, table 5.3 shows the number of flip flops (FFs) and look-up tables (LUTs) used over various OSR, order, and magnitude error bounds. The table clearly shows that for higher OSR/ENOB and increased order, more FFs and LUTs will be needed. The increase in resource usage though measures only 10-15% between OSRs of 32 and 256 for both 2nd and 4th order filters. A slight decrease resource usage is noticed as the magnitude error bound is relaxed. This is due to coefficients and subsequently internal filter signals requiring less bits to meet the relaxed magnitude error constraints in the stopband of the filter.

OSR	Order	FF/LUTs (ρ_{LP1})	FF/LUTs (ρ_{LP2})	FF/LUTs (ρ_{LP3})
32	2	80/242	79/239	76/229
64	2	82/247	81/244	79/239
128	2	84/250	83/255	82/251
256	2	86/254	84/260	84/260
32	4	130/461	128/413	126/403
64	4	133/467	132/420	131/415
128	4	137/473	136/432	135/430
256	4	142/482	140/443	140/443

Table 5.3: Lowpass Filter Implementation (Cheby II f_band=2kHz, f_b=10kHz, Rs = 60 db)

The last performance metric to consider for the ΣΔ filter implementations is power dissipation. After place and route of the filter designs into the Xilinx Artix-7 FPGA within Vivado, the Vivado power estimator was used under default specifications. The results of the power consumption of each filter is listed in table 5.4.

OSR	Order	Power (static/dynamic mW) (ρ_{LP1})	Power (ρ_{LP2})	Power (ρ_{LP3})
32	2	58/1	58/1	58/1
64	2	58/2	58/2	58/2
128	2	58/3	58/3	58/3
256	2	58/4	58/4	58/4
32	4	58/1	58/1	58/1
64	4	58/2	58/2	58/2
128	4	58/3	58/3	58/3
256	4	58/6	58/6	58/6

Table 5.4: Lowpass Filter Implementation (Cheby II f_band=2kHz, f_b=100kHz, Rs = 60 db)

The first noticeable thing about the power dissipation is that the static power is 58 milliwatts; a consequence of the FPGA device simply being powered on. The dynamic power is much less ranging from 1 to 6 milliwatts for OSRs of 32 to 256 respectively in 4th order designs. Since the size of the overall FPGA realization is on the same level in each order category, the dynamic power dissipation is the same across the varying magnitude error constraints.

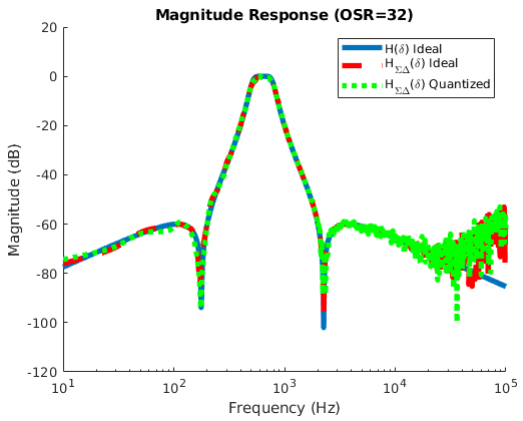
5.2.1.2 Bandpass Filter Designs

Switching to a bandpass filter design, the following filter implementations will be based off a 6th order Chebychev type II filter with a bandpass of 0.2-2kHz and a stopband attenuation of 60 dB. Two magnitude error bounds will be used for the subsequent designs as well and are as follows:

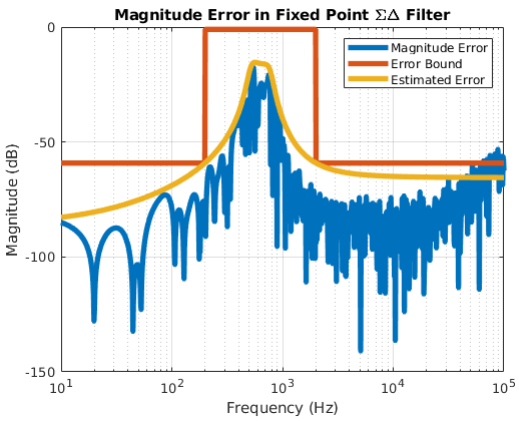
$$\rho_{BP1}(f) = \begin{cases} 0.8913 & 200\ Hz \leq f \leq 2000\ Hz \\ 0.0011 & otherwise \end{cases}$$

$$\rho_{BP2}(f) = \begin{cases} 0.8913 & 200\ Hz \leq f \leq 2000\ Hz \\ 0.0018 & otherwise \end{cases}$$

Like the lowpass filter designs of the previous section, the quantized ΣΔ designs using ρ_{BP1} and ρ_{BP2} fit quite nicely over the ideal response as shown in figure 5.6 and figure 5.7 even for an OSR of 32. Once again, the beginning rise in the output representation noise of the filter creates a noisy transfer function estimation around Nyquist in both magnitude response plots. The magnitude error however, mostly stays below the design stopband magnitude error bound of ρ_{BP1} and ρ_{BP2} shown in the magnitude error plots. The magnitude error plots also demonstrate that the estimated magnitude error provides a good estimate of the actual magnitude error of the quantized filter; the bounds are tight especially in the passband region.

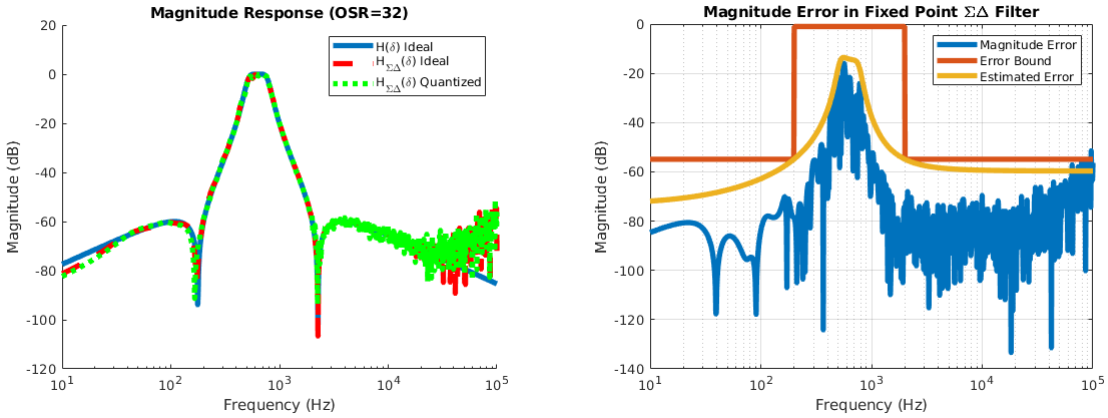


(a) Magnitude Response



(b) Magnitude Error

Figure 5.6: Bandpass Filter (6th order OSR=32, ρ_{BL1})



(a) Magnitude Response

(b) Magnitude Error

Figure 5.7: Bandpass Filter (6th order OSR=32, ρ_{BP2})

In terms of resolution performance, table 5.5 shows that bandpass filter designs easily achieve the required ENOB. The fact that the actual ENOB overshoots the design parameter suggests that the magnitude error bound was the dominate constraint in the optimization routine.

OSR	Order	Design ENOB	ENOB (ρ_{BP1})	ENOB (ρ_{BP2})
32	6	10	13.54	13.22
64	6	12	14.07	14.34
128	6	14	14.68	14.81
256	6	15	15.26	15.26

Table 5.5: Bandpass Filter Implementation (Cheby II f_band=2kHz, f_b=10kHz, Rs = 60 db)

Mapping the the 6th order bandpass filter designs into the same Artix-7 FPGA as before resulted in the resource usage shown in table 5.6. Increasing the filter order by two increases the resource usage around the magnitude as the low-pass filter order being increased from two to four. Even though the 6th order filter is larger, it is still possible to fit over ten of these into the Artix-7 FPGA. While increasing the OSR and subsequent resolution of the bandpass filters

certainly increases the resource usage, relaxing the magnitude error bound decreases it but only slightly. the descrease in complexity for reduced error bounds coincides with the fact that the magnitude error bound was the dominant constraint in the optimization routine.

OSR	Order	FF/LUTs (ρ_{BP1})	FF/LUTs (ρ_{BP2})
32	6	575/185	568/181
64	6	607/197	605/192
128	6	645/208	639/201
256	6	670/217	660/213

Table 5.6: Bandpass Filter Implementation (Cheby II f_band=2kHz, f_b=10kHz, Rs = 60 db)

Power dissipation for the 6th order bandpass filters is reported in table 5.7. Increasing the filter order has certainly made the circuit larger and as a consequence, is dissipating more dynamic power than the fourth order lowpass filter. However, the point that over ten of these filters can fit into the smallest Artix-7 FPGA means that multichannel and MIMO systems do not need a large and higher power FPGA for implementation but rather a smaller device with lower static dissipation.

OSR	Order	Power (static/dynamic mW)	Power (static/dynamic mW)
32	6	58/1	58/1
64	6	58/3	58/3
128	6	58/6	58/6
256	6	58/9	58/9

Table 5.7: Bandpass Filter Implementation (Cheby II f_band=2kHz, f_b=100kHz, Rs = 60 db)

5.3 MASH $\Sigma\Delta$ Filter Implementation

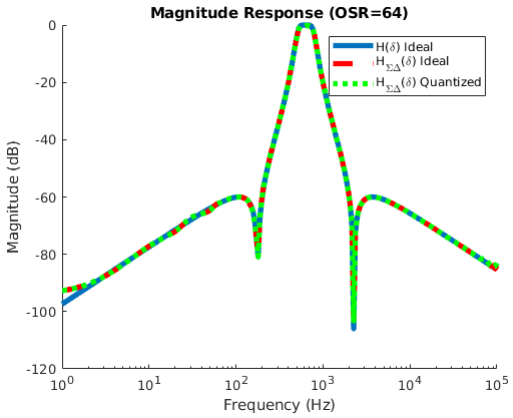
To evaluate the MASH $\Sigma\Delta$ constructions, a series of bandpass filters were created over varying parameters like the prior single stage versions. The bandpass filter was of the Chebychev

II type with a passband of 0.2-2kHz and a 60 dB stopband attenuation. The magnitude error bounds ρ_{BP1} and ρ_{BP2} are used in the following designs and are defined as follows:

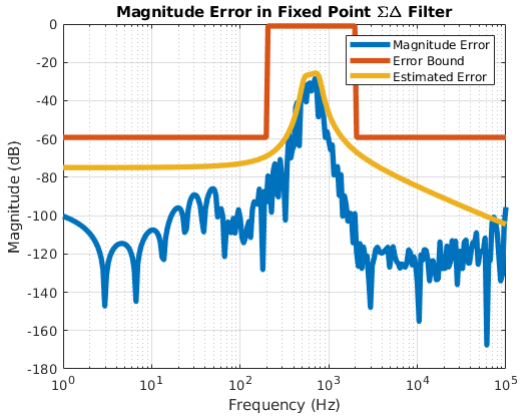
$$\rho_{BP1}(f) = \begin{cases} 0.8913 & 200\ Hz \leq f \leq 2000\ Hz \\ 0.0011 & otherwise \end{cases}$$

$$\rho_{BP2}(f) = \begin{cases} 0.8913 & 200\ Hz \leq f \leq 2000\ Hz \\ 0.0018 & otherwise \end{cases}$$

Simulating the 6th and 8th order bandpass filters resulted in the magnitude response and error plots shown in figure 5.8, figure 5.9, and figure 5.10. Due to the high resolution of the MASH $\Sigma\Delta$ filter, the magnitude response plots are almost indistinguishable from the ideal case. The rise in the representation noise on the output of the single stage design at Nyquist is of no consequence in the MASH design due to the 4th order noise shaping of the two stage second order $\Sigma\Delta$ modulator architecture. In fact this phenomena is clearly seen when taking the magnitude error plots into account. At an OSR of 64, there is about a 100 dB attenuation in the magnitude error of the quantized MASH filter for both the 6th and 8th order bandpass filter designs. It is also worth noting that the estimated error bound in gold is not touching the ρ_{BP} error bound in red at any point in the signal band. The suggests that the dominate constraint in the optimization routine for the MASH construction is the resolution constraint of 22 ENOB. The estimated error bound is again tight is several places compared to the actual magnitude error, namely the passband and at Nyquist. The tightness of the estimated magnitude deviation demonstrates that the sensitivity analysis is adequate for MASH constructions as well.

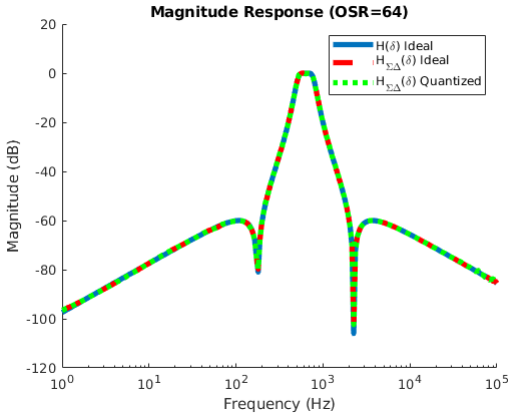


(a) Magnitude Response

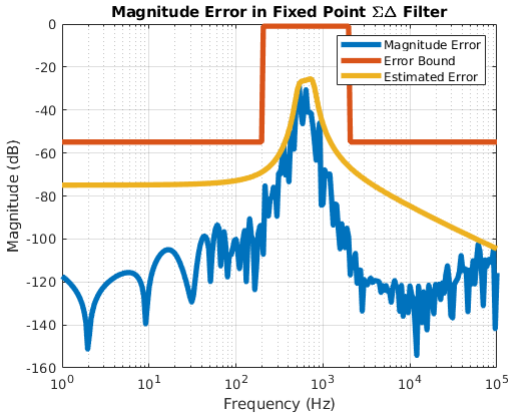


(b) Magnitude Error

Figure 5.8: MASH Bandpass Filter (6th order OSR=64, ρ_{BL1})

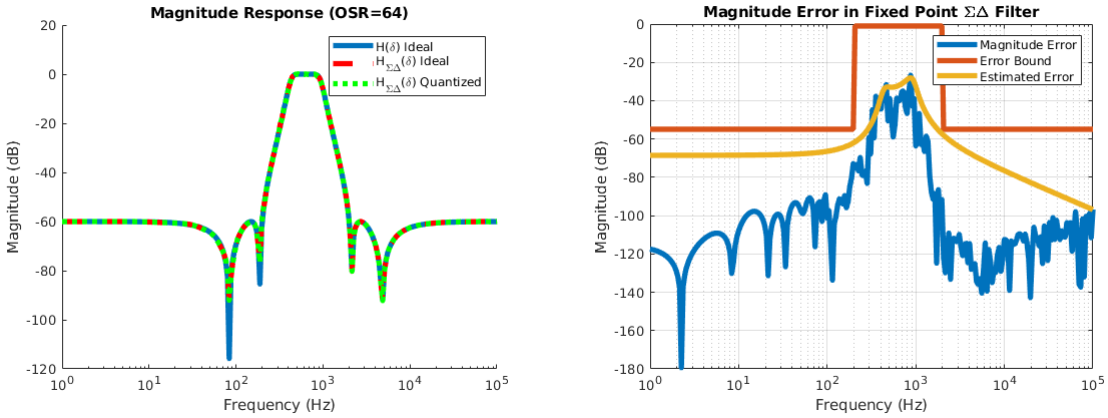


(a) Magnitude Response



(b) Magnitude Error

Figure 5.9: MASH Bandpass Filter (6th order OSR=64, ρ_{BL2})



(a) Magnitude Response

(b) Magnitude Error

Figure 5.10: MASH Bandpass Filter (8th order OSR=64, ρ_{BL2})

Apart from the previous performance plots, simulating the MASH filters over varying parameters has yielded the resolution data tabulated in table 5.9. The first comment about this data is that MASH achieves a very high ENOB for relatively low OSRs. For example, the MASH filters were able to accomplish around 16 ENOB for an OSR of 16 compared to the same ENOB in a single stage design at an OSR of 256. In addition, MASH designs can easily achieve over 20 or more bits of effective resolution at OSRs above 64. In regards to varying the ρ_{BP} design parameter, there is a slight decrease in ENOB for a relaxation in stopband attenuation error which is on par with single stage designs.

OSR	Order	Design ENOB	ENOB (ρ_{BP1})	ENOB (ρ_{BP2})
16	6	16	16.21	16.14
32	6	18	18.31	18.07
64	6	22	22.37	22.12
128	6	24	24.25	24.01
16	8	16	16.14	16.03
32	8	18	18.27	18.07
64	8	22	22.23	22.11
128	8	24	24.21	24.06

Table 5.8: MASH Bandpass Filter Implementation (Cheby II f_band=2kHz, f_b=10kHz, Rs = 60 db)

For resource usage, the MASH filter designs were mapping into the Xilinx Artix-7 XC7A12TL FPGA via Vivado. Table 5.8 shows the resource usage for the filters of different orders, OSRs, and magnitude deviation bounds. Not surprisingly, the MASH filter implementations are larger than than the single stage designs with similar OSR by about 60-70%. This is due to significantly larger overhead with the cancellation logic and multiple coefficient selector circuits. For 20+ ENOB designs the MASH filter requires 2100-2400 LUTs, where most of the expansion is coefficient switching logic for high-resolution coefficients. However, compared to a conventional design, the 24 ENOB 8th order MASH filter is about a *third* the size of a 24-bit conventional 8th order filter that only achieves 14 ENOB. At least three of the 24 ENOB 8th order filters could fit within the smallest Artix-7 FPGA using only 4 IO pins each.

OSR	Order	FF/LUTs (ρ_{BP1})	FF/LUTs (ρ_{BP2})
16	6	302/1316	300/1304
32	6	324/1487	321/1467
64	6	343/1601	340/1591
128	6	386/1784	384/1757
16	8	355/1738	349/1689
32	8	391/1964	384/1892
64	8	427/2157	411/2078
128	8	465/2411	451/2356

Table 5.9: MASH Bandpass Filter Implementation (Cheby II f_band=2kHz, f_b=10kHz, Rs = 60 db)

When evaluation for power, the MASH filters generated the power dissipation numbers shown in table 5.10. For OSRs of 16 and 32, the MASH filters produce power dissipation numbers similar to single stage designs for similar ENOB. The dynamic power dissipation for higher resolution designs jump substantially for large ENOB. Note, these power figures are somewhat surprising, given the OSR-clock operating speed of the filters. Despite running as much as 128x as fast as conventional filters, the dynamic power is reduced by the small footprint of the design and for comparable resolution to 24-bit conventional designs, is below the static power of the

smallest conventional Xilinx FPGA as of 2019. Based in 28nm technology, the Xilinx parts are optimized for large-scale signal processing tasks – these techniques would allow substantial (typically 10x) reduction in part size required for similar tasks and resolutions and further, make no use of embedded DSP core resources.

OSR	Order	Power (static/dynamic mW) (ρ_{BP1})	Power (ρ_{BP2})
16	6	58/1	58/1
32	6	58/4	58/4
64	6	58/8	58/8
128	6	58/13	58/13
16	8	58/2	58/2
32	8	58/6	58/6
64	8	58/11	58/11
128	8	58/17	58/17

Table 5.10: MASH Bandpass Filter Implementation (Cheby II f_band=2kHz, f_b=10kHz, Rs = 60 db)

There are families of ultra-low power FPGAs that currently fill niche markets where programability and firmware update of "glue" logic is a direct benefit. Such FPGAs were never intended for intensive signal processing and contain no embedded DSP cores or fast-add bypass logic, but illustrate the potential of these new filtering techniques. The Lattice uPower® family contains several parts with static power in the few mW scale. For instance, the 6th order bandpass filters for both single stage and MASH designs running at an OSR of 64 dissipate 4 mW and 8 mW of power when implemented into a Lattice ICE40LP8K FPGA.

5.3.1 The Relative Size of Things

Having implemented several different $\Sigma\Delta$ filter designs under various design parameters and constraints, a picture of their relative complexity and cost becomes apparent. The follow is a list of the relative size of filter/controller implementations in an Artix-7 FPGA:

- Conventional shift based filter (8th order):
  - 20 DSP slices, ~2000 LUTs, 200 FFs

  - 0 DSP slices, ~7000 LUTs, 200 FFs
  - ~14 ENOB
  - 30+ IO pins for parallel interfacing
  - Requires complex interfacing between filter/controller and ADC and DAC
- ΣΔ based filter (2nd to 8th order):
  - 100-1000 LUTs, 70- 250 FFs
  - 10-16 ENOB
  - Only two IO pins required
  - no interfacing between filter/controller and ΣΔ ADC and DAC
    - Direct bitstream processing
- MASH ΣΔ based filter (2nd to 8th order):
  - 800-2400 LUTs, 150 500 FFs
  - 16-24+ ENOB
  - Only four IO pins required
  - no interfacing between filter/controller and ΣΔ ADC and DAC
    - Direct bitstream processing

5.4 Conclusion

This chapter began by introducing pragmatic issues involving the accurate simulation and evaluation of ΣΔ bitstream filters/controllers. Those issues center around using enough simulation samples to produce accurate PSDs, how to compute SNR, and how to obtain transfer function magnitude estimations of the filters. From there a series of lowpass and bandpass filter designs were implemented and evaluated for both signal stage and MASH constructions. The filter designs were created over varying oversample ratios, magnitude error bounds, resolution bounds, and filter orders. The designs were then implemented in a Xilinx Artix-7 XC7A12TL

FPGA and evaluated for SNR, power dissipation, and resource usage. For signal stage designs, it was found that for 100-1000 LUTs and 70-250 FFs, quantized filters 2nd to 8th order with 10-16 ENOB could be achieved while the noise and sensitivity analysis provided a good estimate of the predicted SNR and magnitude error of the filter implementation. For MASH, 800-2400 LUTs and 150-500 FFs, quantized filters 2nd to 8th order with 16-24 ENOB could be achieved. Multiple single stage and MASH filters are able to fit inside the smallest Artix-7 FPGA from Xilinx while only requiring two to four IO pins and no interfacing circuitry with outside components. Compared to a conventional bit parallel filters, these filters are at least an order of magnitude lower in complexity when taking into account resource usage, IO pin usage, and interfacing allowing for several continuous time performance filters/controllers to fit within very compact FPGAs. In terms of power, the $\Sigma\Delta$ filters were on par with conventional designs in an FPGA on a one to one basis due to dominate static power dissipation. However, since multiple $\Sigma\Delta$ filters can fit in a small lower power device, the need for larger and higher power devices are unnecessary contributing to a sizable decrease in system power requirements.

Bibliography

[55] Xilinx, The address of the publisher. *Vivado Design Suite User Guide: Model-Based DSP Design Using System Generator*, v2019.2 edition, 10 2019. An optional note.

Chapter 6

$\Sigma\Delta$ Based Digital Control

In the previous chapters, it was shown that $\Sigma\Delta$ filters could be designed and characterized in a methodical and accurate way. The fact that they can emulate transfer function dynamics makes them a prime candidate for implementing embedded LTI controllers. This chapter will describe the advantages of using $\Sigma\Delta$ filters as digital controllers as well as how to integrate them into current popular control architectures such as LQG and H_∞. Lastly, a series of motivating examples will be described that demonstrate the advantages and feasibility of the $\Sigma\Delta$ paradigm.

Prior work on $\Sigma\Delta$ control revolves around mostly simple implementations of controllers (typically PI or PID) for a variety of applications such as motor control [72, 56, 70, 57, 71, 58]. These works demonstrate the effectiveness of even the simplest of controllers to provide adequate performance for the task at hand. This chapter will expand on prior works as well as the previous chapters to discuss implementation strategies for embedded $\Sigma\Delta$ closed loop controllers.

6.1 Control Specific Advantages of $\Sigma\Delta$ Filters

$\Sigma\Delta$ based digital controllers have several advantages over conventional designs, most notably substantially lower latency. A slightly hidden advantage is the use of integrators instead of time-shift operators. This has the consequence that the integrator contents at any given time

are simply the controller implementation state variables. In particular, the parallel output is always available as well as the ΣΔ stream, substantially simplifying low-noise ADC outputs as well simplified debugging in simulation.

6.1.1 Controller Latency

It should be apparent by now that the ΣΔ filter/controller is a low latency implementation do to the fact that it runs at the oversample rate. The total latency of the controller includes the latency of both the ADC and the DAC, which do not require a data conversion interface, as well as the FPGA controller implementation. The total latency can be represented by

$$T_{total} = T_{ADC} + T_{DAC} + T_{SDC}$$

where

$$T_{SDC} = 3\Delta$$

Typical ΣΔ ADC latencies are on the order of one to a few oversample clocks (in advanced MASH designs) when using serial outputs, but this degrades to 1 sample period (or more depending on the communication interface) when using the bit-parallel output. On the other hand, many kinds of actuator are directly controllable with pulses from a ΣΔ stream since these are often 100-200 times the intended actuation bandwidth. This is particularly true for integrated capacitance (MEMS drives) and voice-coil drive circuits. In motor control systems, such high rates of switching are expensive in terms of driver power, so pulse compression is often used. Finally, since the final stage accumulator state is bit-parallel output, a zero latency output can be read by a suitable DAC to effect low noise drive at very low latency. Overall then, extremely low latency control can be achieved whenever ΣΔ stream outputs are available from the sensors or input ADCs. The advent of PDM parts allows for a substantial variety of such components.

6.2 Adapting Output Feedback Control Designs for ΣΔ Based Implementations

In most cases, control designers do not have direct access to a systems state variables and have to rely on synthesizing controllers that utilized the measured output variables. These controller designs are called output feedback controllers and come in a variety of flavors. This section will briefly discuss three types of controllers and how they can be easily scaled and adapted to an equivalent embedded ΣΔ controller implementation.

6.2.1 Scaling of Controllers

In digital control implementations, one must scale the control parameters such that input and output signals to the embedded signal chain do not overflow and cause distortion. It is also the case that front end of the signal chain be scaled such that the signal being measured utilizes the full dynamic range of the ADC so that the highest SNR can be achieved. For ΣΔ controllers, the front end and back end modulators must have signals scaled to 90% of the full scale input (i.e. -0.9 to 0.9) but the overall controller has a gain, $k \in \mathbb{R}$, typically much larger than one. So what to do?

Fortunately the problem of scaling can be easily solved by invoking an ADC gain , $k_u \in \mathbb{R}$, and DAC gain, $k_y \in \mathbb{R}$, in the design where the controller $C(\delta)$ can be written as

$$C(\delta) = kC'(\delta) = k_y k_u C'(\delta)$$

where

$$k = \|C(\gamma)\|_\infty = \sup_\omega \left| C\left(\frac{e^{j\omega}-1}{\Delta}\right)\right|_{\gamma=\frac{e^{j\omega}-1}{\Delta}}$$

The controller for ΣΔ implementation, $C'(\gamma)$, is the normalized version of the original controller design where

$$\left|C'(\gamma)\right|_\infty = 1$$

The ADC and DAC gains k_u and k_y can be chosen according to l_1 scaling methods proposed in [69].

6.2.2 Set-point Tracking

Set-point tracking is a common requirement of many control systems where the plant must follow a specified reference signal. Such control loops find use in motor control, industrial process control, and so on. For a ΣΔ controller implementation, some modifications to the hardware architecture are required to make set-point tracking realizable.

Consider the ΣΔ reference tracking controller diagram in figure 6.1. The ΣΔ controller in this case has two bitstream inputs for the reference as well as the feedback inputs. The reference command to be followed is first ΣΔ encoded using a discrete modulator inside the FPGA and the feedback comes directly from the front end ΣΔ ADC. Typically, the feedback signal is subtracted from the reference command in order to produce an error signal that the controller can act upon. In this case there is no need for the additional subtractor circuit.

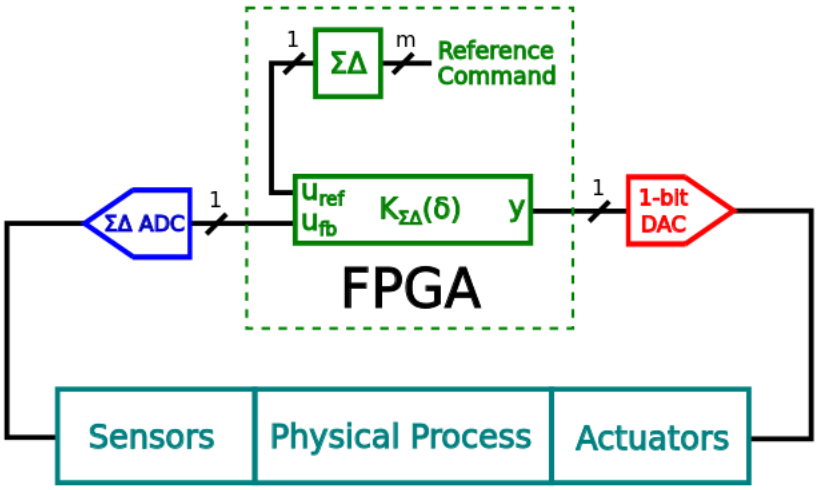


Figure 6.1: Reference Tracking Embedded ΣΔ Controller

The set point ΣΔ controller architecture is illustrated in figure 6.2. From an RTL perspective, the controller is identical to the architecture presented in chapter 3 with the difference being two input multiplexers in the feedforward path. These two input multiplexers take the feedback and reference bitstreams and combine them to form appropriately scaled feedforward coefficient multiplies. This strategy again removes the need for hardware multiplies by bitstream encoding the reference command and pushing the scaling into multiplexers. The increase complexity for this modification is nominal and allows one to implement set point controllers relatively easily.

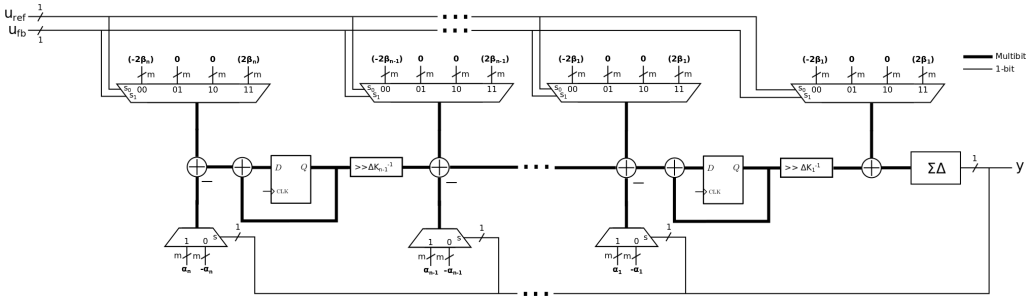


Figure 6.2: RTL Diagram of Reference Tracking ΣΔ Controller

6.2.3 LQG Regulation

Linear-Quadratic-Gaussian (LQG) control is a optimal design technique for synthesizing output feedback regulators and can easily be integrated into a ΣΔ controller [64]. Beginning with the problem formulation, the LQG regulator design assumes that following state-space system

$$\dot{x} = Ax + Bu + \bar{B}w$$

$$y = Cx + v$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^k$, $w \in \mathbb{R}^q$, and $y, n \in \mathbb{R}^q$. In this case, w and v are modeled additive Gaussian noise inputs. The goal of the LQG regulator is to drive y to zero while under external noise disturbance by driving the model input u with the appropriate control effort.

The controller consists of a linear quadratic regulator (LQR) gain vector, K,in conjunction with a Kalman filter. The purpose of the Kalman filter is to estimate the internal states of the plant and provide them for use by the LQR feedback gains. The LQG regulator system diagram can be seen in the figure below

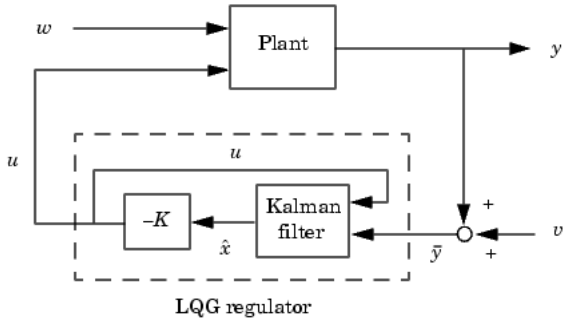


Figure 6.3: LQG Regulator System Diagram

The LQG regulator is designed to minimizing the cost function

$$J(u) = \int_0^\infty \left\{ x^T Q x + 2x^T N u + u^T R u \right\} dt$$

where Q, N, and R are weighting matrices on the control actuation performance. Algebraic Riccati eqautions are solved in order to determine the regulator gain vector K and Kalman filter gain matrix L. K and L are then used to formulate the composite state space regulator equations

$$\dot{\hat{x}} = (A - LC - BK)\,\hat{x} + Ly$$
$$u = -K\hat{x}$$

By taking the Laplace transform of the LQG regulator state space representation, the regulator can be transformed into the transfer function

$$C(s) = K(sI - A + LC + BK)^{-1} L$$

Now having a continuous time transfer function representation of the LQG controller, scaling and $\Sigma\Delta$ control design outlined in chapter 3 can be used to synthesize a digital embedded controller.

6.2.4 H_∞ Control Design

For the H_∞ control problem, the designer seeks to find a controller, K, that stabilizes a plant, P, while meeting various performance constraints [63]. Given the system

$$\begin{bmatrix} z \\ v \end{bmatrix} = \begin{bmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix}$$

with exogenous input w, control input u, performance outputs z, and measured plant output v , the controller

$$u = K(s)\, v$$

is chosen to minimize the performance gain

$$z = F_\ell(P, K)\, w$$

where

$$F_\ell(P, K) = P_{11} + P_{12} K (I - P_{22} K)^{-1} P_{21}$$

Minimizing the performance gain $F_l(P, K)$ is akin to minimizing

$$\|F_\ell(P, K)\|_\infty = \sup_\omega \bar{\sigma}\left(F_\ell(P, K)(j\omega)\right)$$

Once K is found, one has a continuous time transfer function that is ready to be scaled and transformed into an embedded $\Sigma\Delta$ controller implementation. Typically H_∞ controllers are high order and great pains are taken to reduce the controller order of such designs due to conventional discrete shift based limitations [61, 60, 66]. Fortunately, $\Sigma\Delta$ controllers can be made to relatively high order and emulate very high performance H_∞.

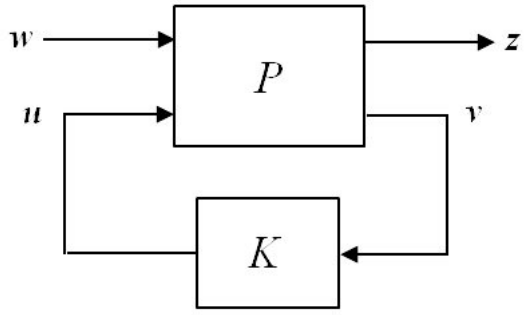


Figure 6.4: H_∞ Controller Diagram

6.3 MIMO Configurations

When it comes to MIMO controller implementations which can be produces be LQG and H_∞ design techniques, there is great flexibility in how the controller can be configured. The following is a list of various ways one can take the individual single input single output $\Sigma\Delta$ controllers and piece them together to construct larger systems:

1. MIMO Configuration with Parallel Output: In the case where a design calls for the use of parallel DACs, the parallel outputs of the individual $\Sigma\Delta$ controllers can be taken and routed to a parallel DAC as shown in figure 6.5. The advantage here is having a high resolution output running at the oversample rate minus any $\Sigma\Delta$ representation noise provided that each controller is strictly proper (i.e. $\beta_{\delta 0} = 0$).

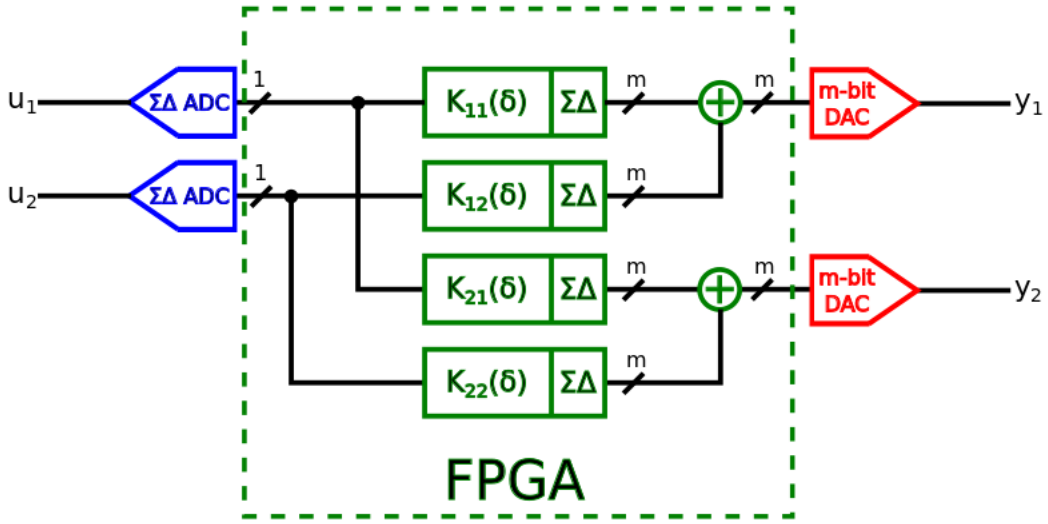


Figure 6.5: MIMO Configuration with Parallel Output

2. MIMO Configuration with ΣΔ Encoded Output: Having MIMO controller ΣΔ encoded outputs, should they be desired, require only a slight increase in complexity. For the MIMO controller shown in figure6.6, the multibit outputs of the individual controllers are summed and then ΣΔ encoded with a discrete modulator before being routed to a one bit DAC.

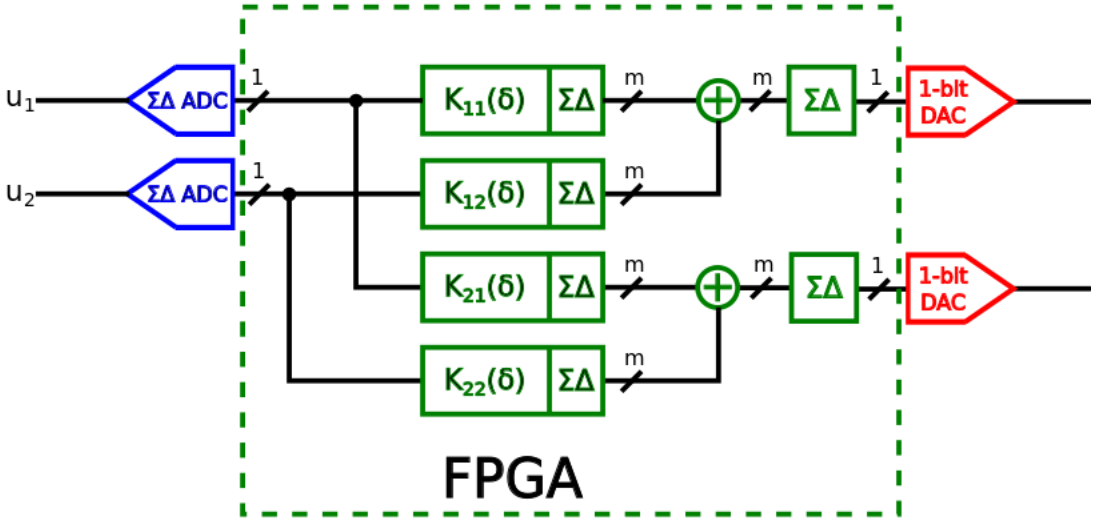


Figure 6.6: MIMO Configuration with ΣΔ Encoded Output

The one bit outputs of this configuration lend themselves nicely to driving loads via a simple output power half bridge stage. Figure 6.7 illustrates what a one bit output driver stage could look like.

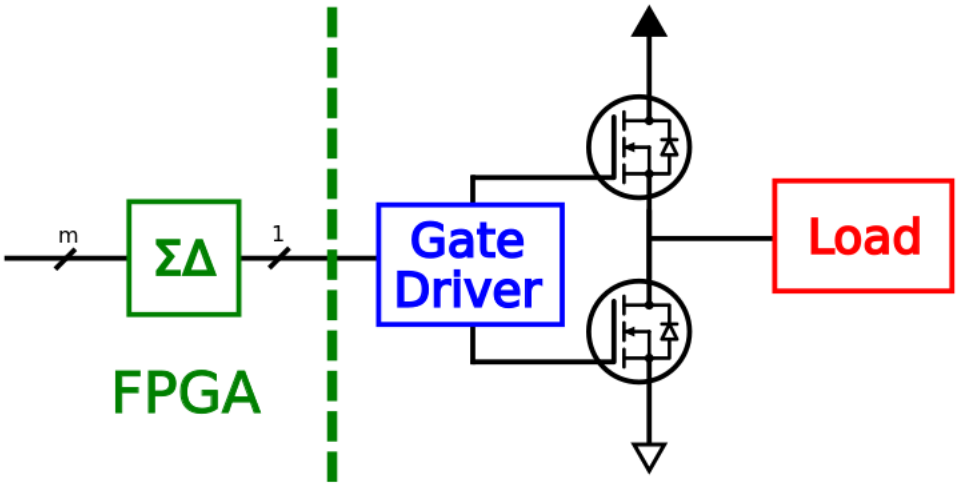


Figure 6.7: ΣΔ Output Half-Bridge Drive

3. MIMO Configuration with Pulse Width Modulated Output: In some cases where the high frequency switching of a PDM representation may be deleterious to performance, it is possible to encoded the multibit output of the controller with a pulse width modulation module. When driving inductive loads for instance, switching at the oversample frequency may reduce efficiency in the drive electronics due to the high transition rates. Figure 6.8 demonstrates how to adapt the ΣΔ MIMO controller by encoding the outputs with a PWM module.

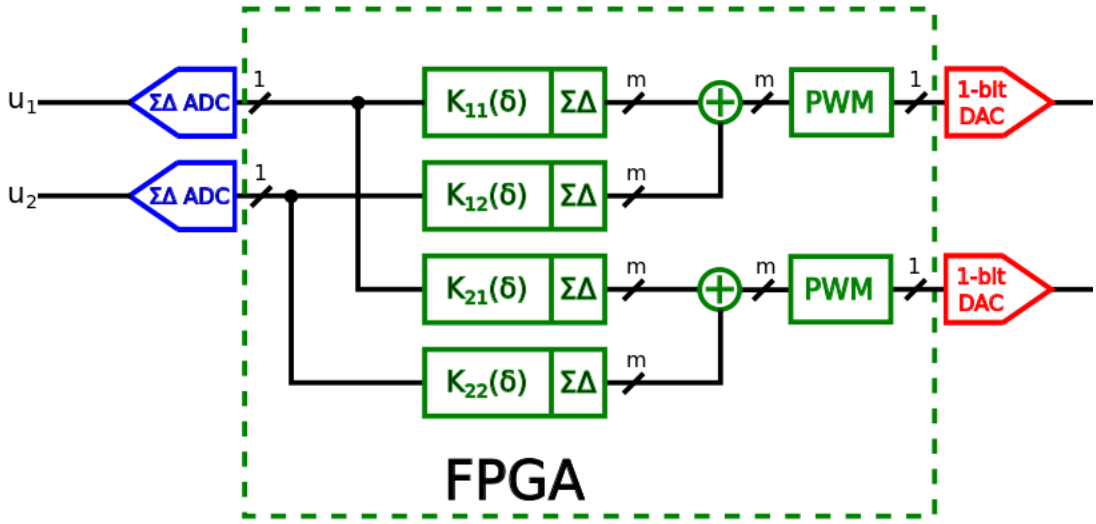


Figure 6.8: MIMO Configuration with Pulse Width Modulated Output

6.4 Motivation Examples

In this section, two motivating examples are presented: the inverted pendulum controller and an AFM cantilever controller. These examples serve to demonstrate that not only can the ΣΔ controllers emulate continuous time controllers, but they do so with high resolution and precision.

6.4.1 Inverted Pendulum

The inverted pendulum on a cart is a well known problem in control where a feedback controller is designed to balance a pendulum upright. The problem is interesting for a variety of reasons such as the upright equilibrium point being unstable and the fact that the system has non-minimum phase zeros. For this example, an LQG controller will be designed, as previously described, to stabilize the pendulum in the upright position.

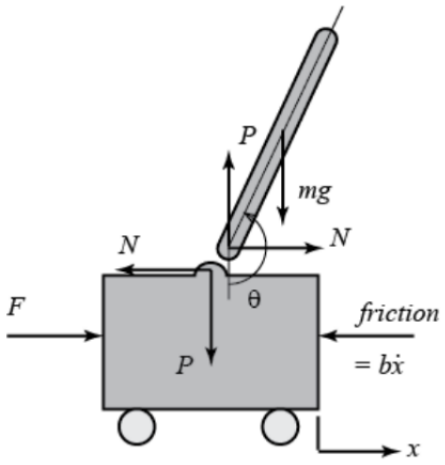


Figure 6.9: Inverted Pendulum on a Cart

Consider the inverted pendulum in figure 6.9. A model of the system can be created with state vector x_{state} defined as

$$x_{state} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$$

where x is the cart displacement and θ is the angle of the pendulum from the upright position. A state space model linearized around the upright equilibrium for a small inverted pendulum is given as

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -15.1372 & -3.0448 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 37.2252 & 31.6122 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 3.3871 \\ 0 \\ -8.3294 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$D = 0$$

The system has one input, the input Voltage to the DC motor that drives the cart, and two outputs, the cart position measurement and the angle of the pendulum. The LQG controller designed for the upright stabilization yielded the two continuous time controllers

$$K_x(s) = \frac{-2994s^3 - 2.842e04s^2 + 4.553e05s + 1.156e05}{s^4 + 140.8s^3 + 5721s^2 + 2.569e04s + 4.403e04}$$

$$K_\theta(s) = \frac{9104s^3 + 1.636e05s^2 + 1.352e06s + 3.682e06}{s^4 + 140.8s^3 + 5721s^2 + 2.569e04s + 4.403e04}$$

The pendulum controller FPGA implementation can be seen in figure 6.10. The controller implementation reads both the x and θ measurements via second order $\Sigma\Delta$ modulators and drives them straight into the K_x and K_θ controllers. The multibit output of the controllers are them summed and then bitstream encoded before driving a one bit DAC.

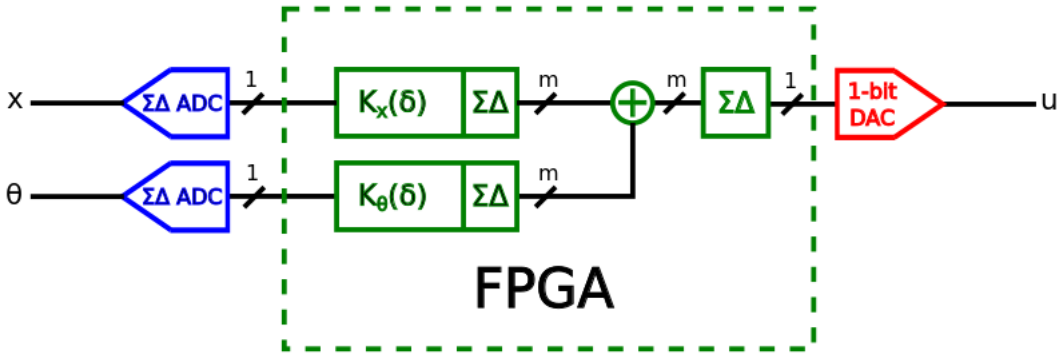


Figure 6.10: Pendulum Controller FPGA Implementation

After the initial design of each ΣΔ sub-controller with an OSR of 128 and $f_b = 1kHz$, the controller was mapped into an IE40LP8K FPGA from Lattice. The resource usage for this controller was 967 LUTs and 276 FFs with a power dissipation of only 1 mW.

A simulation of the closed loop system seen in figure 6.11 represents the pendulum starting away from equilibrium by $-\frac{\pi}{32}$ and then being "tapped" at t=0.5 seconds. The simulation shows that the ΣΔ pendulum response in red overlays the continuous time response in blue. This demonstrates that the ΣΔ controller design is effective at emulating its continuous time counterpart.

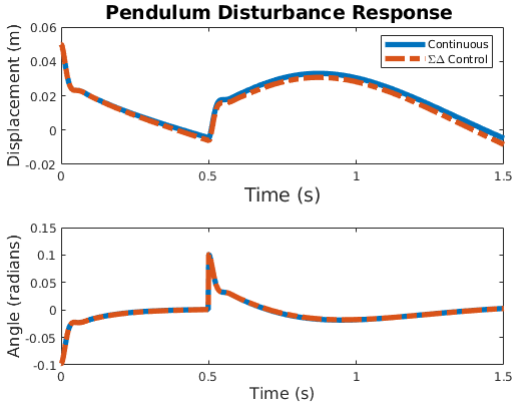


Figure 6.11: Pendulum Controller Time Simulation

6.4.2 AFM Cantilever Q-Control

Atomic force microscopy is a method of scanning that requires high bandwidth and high resolution. The size of the scanning probe for such systems is on the order of microns and has dynamics in the tens and hundreds of kilohertz. Digital feedback control at these rates can be tricky and expensive; perfect for $\Sigma\Delta$ control.

The Q controller presented in [59] describes an AFM cantilever with a differential sensing interface. The goal of the Q controller is to provided feedback control that quells the first oscillatory mode of the cantilever tip at about 49 kHz. This allows higher performance and greater accuracy for the cantilever operation in tapping mode, especially for actuation signals close to resonance.

Figure 6.12 illustrates a $\Sigma\Delta$ version of the Q controller. The design samples the output of the sensing circuit with a $\Sigma\Delta$ ADC which provides the bitstream signal encoding for the FPGA based control. The bit parallel output of the $\Sigma\Delta$ controller is routed from the final state integrator and into an ideal parallel DAC running at the oversample rate. A parallel DAC is used in this case in order to not excite any unmodeled higher order cantilever resonance modes with bitstream representation noise.

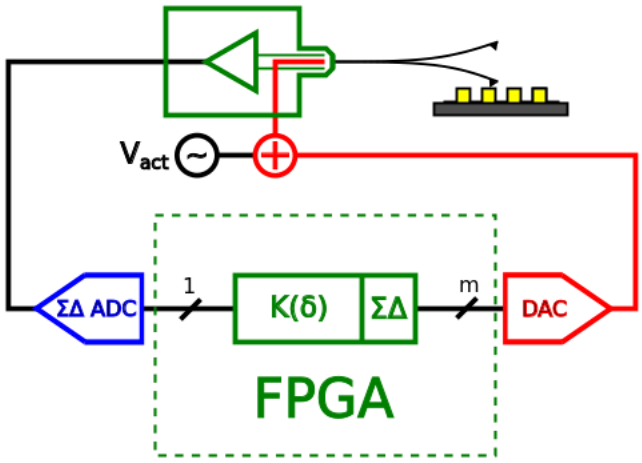


Figure 6.12: AFM Cantilever Q Control Loop Implementation

The estimated transfer function estimate from the actuation input Voltage to the output Voltage of the sense amplifier circuit is given as

$$G(s) = \frac{-0.73s^3 + 1.28 \cdot 10^5 s^2 - 6.57 \cdot 10^{10} s + 1.45 \cdot 10^{16}}{s^3 + 1.12 \cdot 10^6 s^2 + 9.52 \cdot 10^{10} s + 1.05 \cdot 10^{17}}$$

and the PPF controller is given as

$$C(s) = \frac{k_c \omega_c^2}{s^2 + 2\zeta_c \omega_c s + \omega_c^2}$$

where $k_c = 0.959$, $\zeta_c = 0.178$, and $\omega_c = 48.547$ kHz. Simulations were performed with the continuous time controller, a conventional discrete controller, and an $\Sigma\Delta$ controller implementation. It is important to note that the conventional discrete controller is implemented by first decimating an input $\Sigma\Delta$ modulator running at 30.72 MHz with a 3rd order Sinc filter at a rate of 25 and 50. The discrete controller as well as the output DAC then runs at the decimated clock rate. The discrete controller was derived from the continuous controller using the bilinear

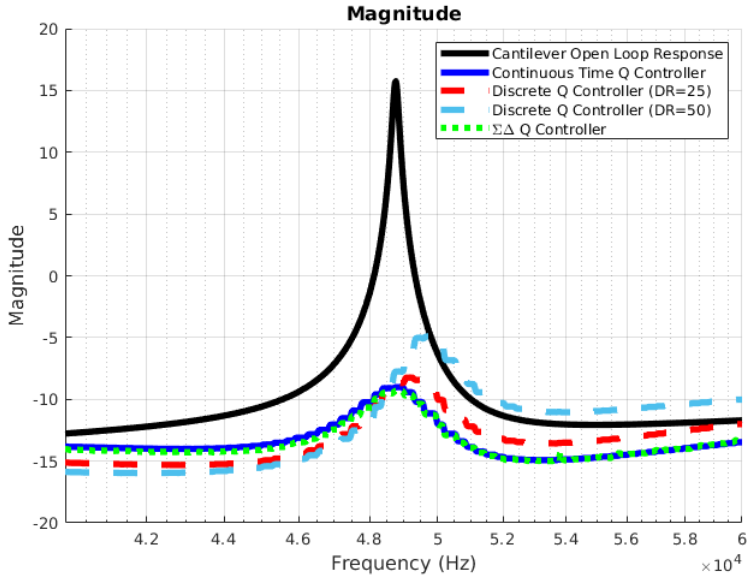


Figure 6.13: AFM Continuous, Discrete, and ΣΔ Q-Controller Magnitude Response

transform at the appropriate sampling period. A plot of the closed loop magnitude response performance can be seen in figure 6.13 with the SDC running at a clock frequency of 30.72 MHz. Using Welch's power spectral density estimates from the actuation Voltage input and sense Voltage output of the Simulink model, the closed loop magnitude response of the continuous, discrete, and ΣΔ Q controllers were determined and plotted.

ΣΔ controllers running at three different clock frequencies using the controller parameters seen in table 6.1 were simulated with their corresponding closed loop magnitude response plotted in figure 6.12. Table 6.2 lists the relevant design information associated with each controller. While there is no converter or sensor noise modeled in the simulation, the SDC control band SNR can still be estimated and measured from the fixed point implementation. The SNR numbers assume a full scale sinusoidal input while integrating the noise PSD over the entire control band from DC to f_B. It is also of note to specify that the ideal Q controller coefficients were chosen to obtain a specific closed loop Q value. The SDC control loop magnitude response varies slightly from the ideal model due to the discretization process at different sampling frequencies.

Table 6.2: ΣΔ Q Controller Fixed Point Parameters

Oversample Frequency	Measured SDC SNR	Power	LUT/FF
$7.68MHz$	$61.47dB$	$1.94mW$	178/76
$15.36MHz$	$71.16dB$	$3.12mW$	195/82
$30.72MHz$	$86.57dB$	$4.36mW$	214/88

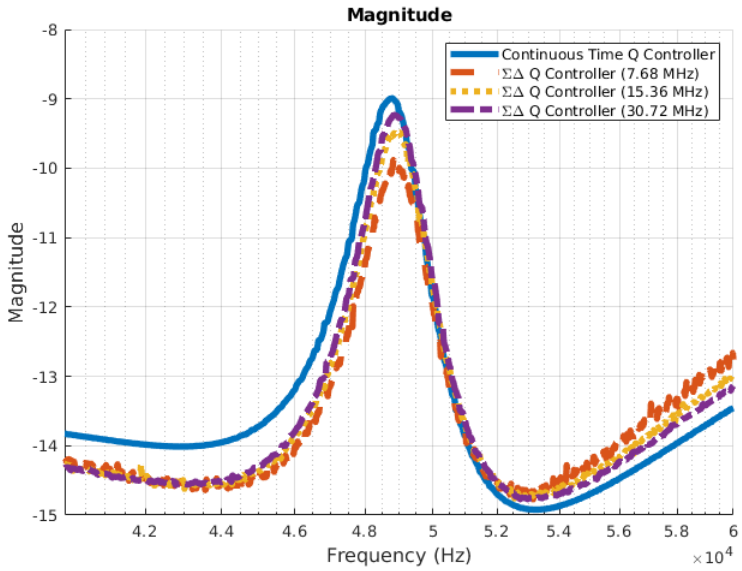


Figure 6.14: AFM ΣΔ Q-Controller Magnitude Response

Table 6.1: ΣΔ Q Controller Parameters

Parameter	Controller 1	Controller 2	Controller 3
f_B	60 kHz	60 kHz	60 kHz
f_s	7.68 MHz	15.36 MHz	30.72 MHz
Controller Latency	1.3 μs (5 clock cycles)	326 ns (5 clock cycles)	163 ns (5 clock cycles)
Estimated SDC SNR	61dB (9.84 ENOB)	76.5dB (12.4 ENOB)	91.5dB (14.9 ENOB)

To estimate resource utilization and power dissipation in an FPGA, the three ΣΔ controllers were mapped into the low power ICE40LP8K-CM81 FPGA from Lattice Semiconductor. The Lattice iCEcube2 software was used to synthesize, place, and route the hardware description of the controllers to determine the number of flip flops and look up tables (LUT) used as well as estimate the power dissipation in each scenario. The ICE40LP FPGA from Lattice does not contain any built in DSP hardware accelerators and although it might typically be used for glue logic or interfacing, the low complexity and multiplierless design of the ΣΔ controller allows normally resource hungry controller implementations to be mapped into low cost, low power, low resource FPGAs. In fact, the ICE40LP8K FPGA can fit 30 to 40 second order controllers based on the resource utilization numbers given above. One could also implement a higher order controller to dampen higher frequency resonance peaks in the Q controller which would further increase an AFM's scanning speed.

These simulations show that the ΣΔ Q controller is able to emulate a continuous time controller transfer function by directly processing the bitstream from the front end modulator. The entire ΣΔ Q controller has a signal chain latency of only five clock cycles which is a minute fraction of the multiple decimated clock cycles required for the conventional discrete case. Consequently, the conventional discrete controller closed loop magnitude response deviates substantially from that of the ideal continuous controller. When increasing the oversample clock frequency, the ΣΔ Q controller response holds closer to the continuous case but at a cost. In terms of power and footprint, doubling the clock frequency increases the power dissipation by a factor of about 50% in the controller while the LUT/FF utilization increases only modestly. Faster clock speeds also allow for higher controller SNRs which is demonstrated in the estimated and actual output noise powers. The estimated output noise power from Table 6.1 matches quite well to the actually noise power listed in Table 6.2 which, again, validates the accuracy of the noise analysis.

6.5 Conclusions

In this chapter, a variety of implementation schemes for embedded ΣΔ controllers were presented and discussed. So long as a controller can be converted to an LTI transfer function model, the ΣΔ controller architecture can accommodate any design whether it be a PID, LQG, or H_∞. MIMO systems can also be easily adapted by linking single input and output ΣΔ controllers. There is also flexibility in the output type that a controller can provide whether it a multibit output, a bitstream output, or a PWM output as a means to drive loads at a lower switching frequency. Two control examples were also presented. The inverted pendulum control problem demonstrated the feasibility of linking two ΣΔ controllers to form a composite output and provide continuous time performance. The AFM cantilever control problem demonstrated that ΣΔ controllers can easily perform high bandwidth control in high performance applications. The low latency of the ΣΔ AFM controller in conjunction with being δ operator based allowed it to easily outperform the high latency discrete controllers and achieve performance mirroring an ideal continuous controller. These controllers were also successfully integrated into a low power non-DSP Lattice ICE40LP FPGA using minimal resources and operating on only a couple milliwatts.

Bibliography

[56] D. Al-Makhles, A. Swain, and N. Patel. Delta-Sigma based bit-stream controller for a D.C. motor. In *TENCON 2012 IEEE Region 10 Conference*, pages 1–5, Nov 2012.

[57] D. Almakhles, N. Pyle, H. Mehrabi, A. Swain, and A. P. Hu. Single-bit modulator based controller for capacitive power transfer system. In *2016 IEEE 2nd Annual Southern Power Electronics Conference (SPEC)*, pages 1–6, Dec 2016.

[58] D. J. Almakhles, R. Sakthivel, A. Swain, U. Subramaniam, and K. Almustafa. A Generalized One-Bit Control System Using a ΔΣ -Quantizer. *IEEE Access*, 7:117009–117018, 2019.

[59] M. B. Coskun, H. Alemansour, A. G. Fowler, M. Maroufi, and S. O. R. Moheimani. *Q* Control of an Active AFM Cantilever With Differential Sensing Configuration. *IEEE Transactions on Control Systems Technology*, pages 1–8, 2018.

[60] E. Gershon. Robust Reduced-order Hâ Output-Feedback Control of Retarded Stochastic Linear Systems. *IEEE Transactions on Automatic Control*, 58(11):2898–2904, Nov 2013.

[61] P. J. Goddard and K. Glover. Controller reduction: weights for stability and performance preservation. In *Proceedings of 32nd IEEE Conference on Decision and Control*, pages 2903–2908 vol.3, Dec 1993.

[62] G. C. Goodwin, R. H. Middleton, and H. V. Poor. High-speed digital signal processing and control. *Proceedings of the IEEE*, 80(2):240–259, Feb 1992.

[63] M. Green and D.J.N. Limebeer. *Linear Robust Control*. Dover Books on Electrical Engineering. Dover Publications, Incorporated, 2012.

[64] J.P. Hespanha. *Linear Systems Theory*. Princeton University Press, 2009.

[65] D.A. Johns and D.M. Lewis. Sigma-delta based IIR filters. In *Circuits and Systems, 1991., Proceedings of the 34th Midwest Symposium on*, pages 210–213 vol.1, May 1991.

[66] D. Mustafa and K. Glover. Controller reduction by H/sub infinity /-balanced truncation. *IEEE Transactions on Automatic Control*, 36(6):668–682, June 1991.

[67] Chiu-Wa Ng, Ngai Wong, H. Kwok-Hay So, and Tung-Sang Ng. Direct sigma-delta modulated signal processing in FPGA. In *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, pages 475–478, Sept 2008.

[68] Michael Ruppert and S O. Reza Moheimani. Multimode Q Control in Tapping-Mode AFM: Enabling Imaging on Higher Flexural Eigenmodes. *IEEE Transactions on Control Systems Technology*, 24:1–11, 10 2015.

[69] M. Steinbuch, G. Schootstra, and Hoon-Toh Goh. Closed-loop scaling in fixed-point digital control. *IEEE Transactions on Control Systems Technology*, 2(4):312–317, Dec 1994.

[70] A. Swain, D. Almakhles, Y. Hou, N. Patel, and U. Madawala. A Sigma-Delta Modulator based PI controller for bidirectional inductive power transfer systems. In *2016 IEEE 2nd Annual Southern Power Electronics Conference (SPEC)*, pages 1–6, Dec 2016.

[71] C. Wanigasekara, D. Almakhles, L. Zhou, et al. Design of âÎ£ Based PID Controller for Wind Energy Systems. In *2019 IEEE International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I CPS Europe)*, pages 1–6, June 2019.

[72] X. Wu and R. M. Goodall. One-bit processing for digital control. *IEE Proceedings - Control Theory and Applications*, 152(4):403–410, July 2005.

Chapter 7

Conclusions

This dissertation presented a design methodology for implementing hardware implementations of LTI controllers that directly process ΣΔ bitstreams. These filters have extremely low latency as a result of operating at the oversample ratio of the input bitstreams and thus achieve near-continuous-time response. When implemented via FPGA, these controllers require substantially smaller resources (e.g. ~10x) look up tables and minimal flip flops compared to conventional bit-parallel designs. As a consequence, extremely large MIMO designs can fit within small, low cost, and low power realizations. Further, the use of integrators vs. time-shift operators in the formulation and construction has the hidden benefit of relaxed coefficient resolution sensitivity allowing for practical single-filter realizations of 8th and higher degree digital IIR filters.

These benefits are greatly enhanced by the implementation of MASH realizations, allowing effective 4th order behavior of the sigma-delta modulation scheme and thus greatly reduced oversampling requirements at high resolution (e.g. 16 ENOB) and the ability to reach ultra-high resolution (e.g. 24+ ENOB) with hardware that is still far smaller than conventional designs. Both MASH and single-stage filters are fully supported by a simple additive NTF model which has been shown to be highly accurate despite the required approximations. MASH benefits in the predictive stability of the filters (allowing 2nd-order modulators throughout) while only requiring 2 bit-serial, non-positional interfaces.

This combination of features is particularly valuable in high performance controller implementations. In particular, the very low latency (on the scale of 1 OSR clock period) potentially simplifies the design of digital controllers realizing continuous-time transient response without the classic issues of analog noise, component drift and calibration. That these filters can be realized in extremely small footprints allows for the design and implementation of micro-power filters and compensators with superior characteristics matching the potentials of advanced sigma-delta modulator designs. In particular, applications such as hearing-aids (currently designed around low-power CPUs and dissipating about 1.1mW) could be run in low-power FPGA (firmware programmable) or even lower power ASIC implementations. Such designs could reach 10uW for the digital signal processing (compression, filtering) portion of the power use, roughly tripling the overall battery life (about 50-100 hours currently). Other direct uses could be found in AFM microscope controllers (improving the performance of the DSP part by more than 10x) thus substantially reducing hardware costs. (Both the analog and digital parts of the controller could be simplified).

Finally, the widespread adoption of high-resolution (1 and 2 bit PDM signaling) would simplify wiring and communication requirements while improving system robustness and sensitivity to transient faults such as ionizing radiation based soft-errors. Techniques for protecting adders and clock distribution schemes are well known, all long-wire communications are via non-place value bit-streams for which soft-errors are simply low-sensitivity noise sources.

7.1 Future Work

Having laid the groundwork for creating reliable high performance LTI $\Sigma\Delta$ controllers, there are many avenues where this work can be extended.

7.1.1 Design Partition and Cascade Realizations

A natural extension to the optimization procedures is the separation of eigenvalues into frequency similar groups and partitioning the design based on these groups. This has the benefit of allowing substantially smaller coefficients and accumulators as well as the possibility of

reduced power use while maintaining overall resolution and performance. Since the representation error of cascaded filters in non-correlated, only the associated white-noise NTF components accumulate. This allows for substantial design flexibility in controller applications where noise sensitive portions of the response can be accommodated.

7.1.2 Modulated (non-baseband) Controllers and Filters

The great majority of $\Sigma\Delta$ ADC and DAC devices are built at base-band. The issue of 1/f noise, however, make such simple implementations very difficult or impossible in sensitive sensing systems. Classical capacitive microphones and microphone pre-amplifiers have typical preamp voltage gains of 60-75db. Maintaining a 65+db SNR places substantial strain on amplifier design as well as requiring physically large (and thus expensive) sensing elements. The classical solution is to modulate the element at 1MHz, perform the amplification using 1 MHz IF and then demodulate to base band. This solution is used extensively in (very expensive) precision audio measurement systems. Instead of the demodulator, a $\Sigma\Delta$ ADC with 1 MHz modulation could be used. Such a design could be demodulated digitally, providing substantial cost savings and allowing for custom calibration of the sensor non-idealalities in NVram. These designs are rather simple extensions to the current filtering scheme.

7.1.3 Nonlinear Control

While linear controllers are certainly the more popular design choice, in reality most real world systems are nonlinear. Saturation, quadratic effects, trigonometric terms, and other nonlinear phenomena present them selves in a variety of important control problems. For instance, in robotic control the robot kinematics include trigonometric functions and nonlinear terms. Partial feedback linearization is a popular way to handle the nonlinear terms of the robotic kinematics while providing sufficient control of the robot dynamics. To implement bitstream control in these situations, one must devise a way to implement state machines that emulate trigonometric functions while directly processing bitstreams. Future work will revolve around devising $\Sigma\Delta$ based filters and state machines that have the ability to emulate trigono-

metric and common nonlinear functions and how to apply them to different classes of nonlinear control problems.

3 Phase Motor Control

Multi phase motor control requires complicated algorithms and conversions between stationary and rotating reference frames to provide adequate and efficient control. Typically implemented in software via DSPs and running anywhere from ten to one hundred kilohertz, modern motor control invoke considerable latency in their implementation. Given that ΣΔ controllers are extremely low latency, the opportunity to provide higher performance control with possibly greater efficiency is available. Rather than take the conventional approach using Park and Clarke transformations [73] for the reference frame conversions in the feedforward and feedback, it is possible to construct compact ΣΔ oscillator circuits to generate the oscillating waveforms necessary for commutation. Figure 7.1 shows an RTL diagram for a two phase sine and cosine generator. Such a circuit could be used to drive a two phase motor.

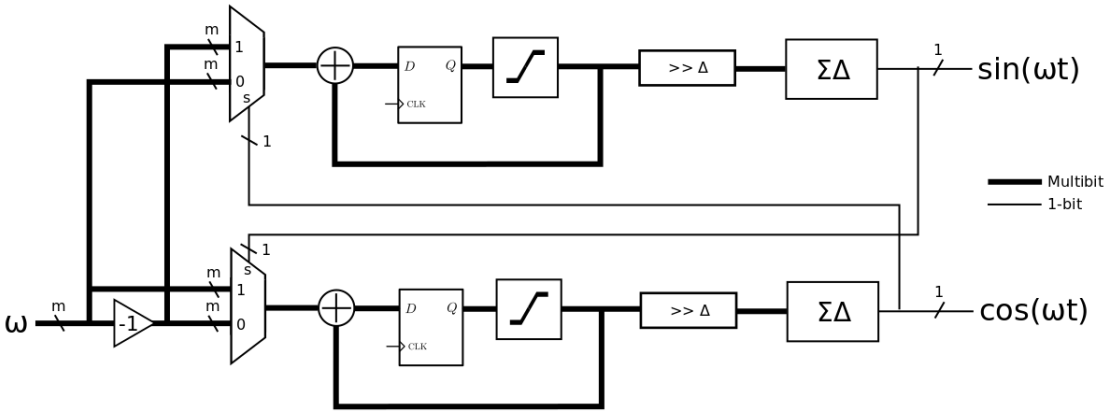


Figure 7.1: Two Phase Oscillator Circuit

Figure 7.2 shows a time simulation of very high resolution sinusoids coming from the multibit outputs of the oscillator. For future work regarding three phase motors, a simple three

phase oscillator circuit will be devised and utilized into a novel low latency and high performance motor control architecture.

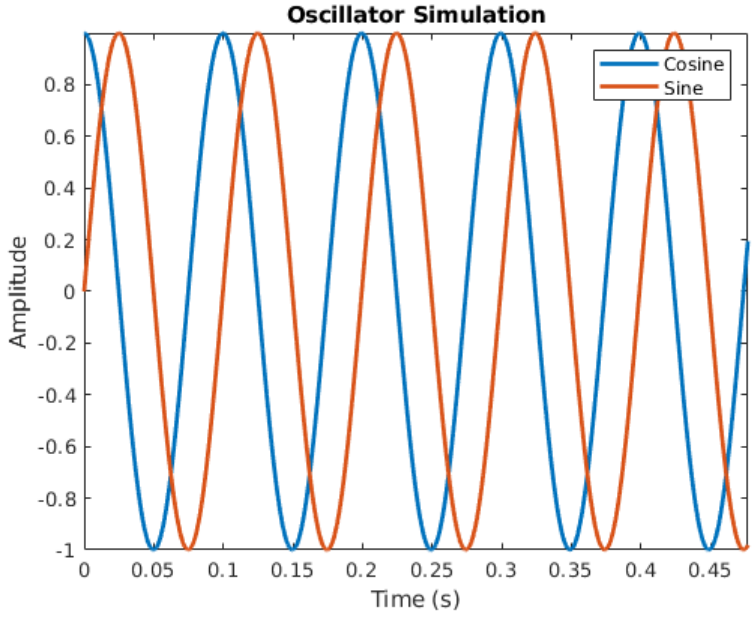


Figure 7.2: Two Phase Oscillator Simulation

Reduced Driver Switching

Driving inductive loads comes up very often in embedded control applications. Whether it is a voice coil for a speaker, a motor coil, or a electromagnet, these actuators are typically driven via PWM into a half-bridge power stage. Unfortunately there comes a point of diminishing returns when quickly switching power transistor switching stages; power loss in the switching transistors increases dramatically when close to the switching timescale limits of the device. In order to drive these types of stages directly via a bitstream, the number of switching transitions must be lowered to reduce power loss. Further work will include devising a scheme to reduce switching losses due to direct drive via oversample rate bitstreams.

Bibliography

[73] Microchip Technology Inc. *Sensored (Encoder-Based) Field Oriented Control of Three-Phase Permanent Magnet Synchronous Motor (PMSM)*, ds00002757a-rev1 edition, 2018.